ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SONIDO E IMAGEN

Título del proyecto:

ENTORNO EDUCATIVO PARA EL APRENDIZAJE DE SISTEMAS DIGITALES

Juan María Pérez Azpeitia

Francisco Javier Arregui San Martín

Pamplona, Jueves, 22 de Abril de 2010

Desde esta página quiero hacer mis agradecimientos a todas las personas que han colaborado tanto en la realización de este proyecto como a lo largo de todos estos años.

Mis agradecimientos …..

…….. en primer lugar, para mis padres, por todo lo que han hecho por mí en este momento y en toda mi vida.

…….. a mi tutor, Patxi Arregui, por la confianza y ayuda mostrada, así como por la posibilidad de dejarme hacer lo que me gusta.

……. a mi primo Javier, también por todo lo que ha hecho por mí y por ser una pieza importante para haber llegado hasta aquí.

…….. de igual manera, a primo Miguel, por permitirme vivir en su casa todos estos años.

…….. a todo el resto de familiares, abuelos, tíos y primos, por su apoyo.

…….. por último, no olvidarme todos mis amigos, debido a todo lo que disfruto con ellos.

Para todos, GRACIAS.

INDICE

CAPÍTULO 1.

INTRODUCCIÓN Y OBJETIVOS

1.- INTRODUCCIÓN Y OBJETIVOS

El siglo XX ha traído consigo numerosos avances tecnológicos, lo cual ha permitido que en la actualidad contemos con ordenadores personales, agendas electrónicas de bolsillo, teléfonos móviles con conexión a Internet y múltiples aplicaciones, etc.

El desarrollo de estos dispositivos se produce en base a la evolución que han sufrido en las últimas décadas los circuitos integrados, llegando cada día a ser más veloces y densos. En este ámbito, los circuitos integrados constituyen los bloques básicos de cualquier sistema digital, en el cual la información se procesa de forma discreta.

Este proyecto gira en torno al aprendizaje de sistemas digitales y, fundamentalmente, en cuanto a dispositivos de lógica programable del tipo FPGA. Los objetivos que se persiguen son los siguientes:

- Introducción al concepto de dispositivo de lógica programable. Ver la evolución que estos dispositivos han sufrido, así como su implicación en los diferentes campos de la ingeniería.
- Analizar y definir un entorno educativo adecuado basado en dispositivos de lógica programable para su utilización en un laboratorio de prácticas. Para ello es necesario hacer el correspondiente estudio de mercado.
- Desarrollar un laboratorio de prácticas para las asignaturas 'Estructura y Tecnología de Computadores' de la titulación 'Ingeniería Técnica en Informática de Gestión', y 'Circuitos Electrónicos Digitales' de la titulación 'Ingeniería de Telecomunicación', en las que se realiza el estudio de sistemas digitales.
- Estudio de una aplicación de mayor complejidad orientada a temas vistos en la titulación 'Ingeniería Técnica de Telecomunicación Especialidad Sonido e Imagen'.
- Como punto adicional del proyecto, en el Anexo 2, se realiza el estudio de un robot basado en microcontroladores PIC: Robot Moway.

- ***LABORATORIO DE PRÁCTICAS***

El objetivo de la realización de este punto es el de realizar la modernización del material existente en el laboratorio de prácticas para las asignaturas mencionadas anteriormente. Hasta ahora, en el laboratorio se viene utilizando el software de desarrollo Max+Plus II® junto con la tarjeta educacional UP2, ambos de la compañía Altera®. Dicho material va a ser sustituido por el software Quartus II® y la tarjeta educacional DE2, también ambos de la compañía Altera®. Este software, en su versión para estudiantes, se encuentra disponible para ser descargado gratuitamente desde la página Web de la compañía, así como las nuevas tarjetas educacionales ya se encuentran disponibles en el laboratorio. Por tanto, el trabajo de este proyecto consiste en adaptar el material anteriormente desarrollado para el laboratorio al nuevo entorno de aprendizaje.

El material se compone de lo siguiente:

- **Manual de las tarjetas 'UP2' y 'DE2' de Altera:** permite conocer las características del hardware disponible en el laboratorio, además de realizar un estudio sobre las familias de los dispositivos programables integrados en las mismas.

Aunque, como se ha mencionado, se va a pasar a trabajar con la tarjeta DE2, se incluyen nociones de la tarjeta UP2, debido a la posibilidad de tener que recurrir a la utilización de esta tarjeta por la falta del número suficiente de tarjetas DE2 en un determinado momento por diferentes motivos.

- **Manual del software Quartus II:** el objetivo es tener un manual adaptado y orientado a la iniciación en este entorno de desarrollo, que sirva como ayuda en la realización de los guiones de prácticas. Para ello, no se incluyen todas las posibilidades que el software ofrece, sino las necesarias para llevar a cabo el desarrollo de las prácticas.

- **Guiones de prácticas para la titulación 'Ingeniería Técnica en Informática de Gestión'**: se trata de un conjunto de ocho prácticas el cual permite aplicar los conocimientos teóricos adquiridos en la asignatura 'Estructura y Tecnología de Computadores'.

- **Guiones de prácticas para la titulación 'Ingeniería de Telecomunicación':** debido a la menor carga de horas prácticas de la asignatura 'Circuitos Electrónicos Digitales', en este caso, el contenido se resume a dos guiones que permiten aplicar los conocimientos teóricos de la mencionada asignatura.

- ***APLICACIÓN ESPECÍFICA***

Para esta parte del proyecto, en primer lugar, es necesario acudir al estudio de mercado realizado, con el objetivo de elegir el hardware adecuado para su implementación.

De esta manera, se realiza el estudio de un entorno educativo orientado al trabajo con elementos de vídeo.

La aplicación consiste en un sistema completo de vídeo, en el que mediante una cámara digital se realiza la adquisición de imágenes de forma continua para que a su vez sean visualizadas en una pantalla LCD, todo ello controlado por un dispositivo de lógica programable. Este diseño, en su forma original, está disponible para ser descargado desde la página Web de Altera. En la aplicación, este diseño queda modificado, permitiendo jugar con la distribución de los colores de la imagen final mostrada en la pantalla LCD.

- ***ANEXO 2. ROBOT MOWAY***

Como ampliación, en cuanto a sistemas digitales se refiere, este anexo recoge la explicación de un robot programable en base a microcontroladores PIC. De esta manera, se dan a conocer características de este tipo de microcontroladores. Asimismo, este robot representa un completo entorno de aprendizaje para aplicaciones de robótica.

CAPÍTULO 2.

ANTECEDENTES FPGA

2.- ANTECEDENTES FPGA

En este apartado se da una visión de la evolución de las herramientas disponibles para el diseñador de sistemas digitales en cuanto a la realización de diseños de lógica programable, dando una visión más extensa de los dispositivos FPGA que son los empleados durante la realización del presente proyecto.

2.1.- INTRODUCCIÓN A LA LÓGICA PROGRAMABLE

Un amplio espectro de posibilidades está disponible para el diseño e implementación de circuitos lógicos, tal como se ve en la Figura 1.

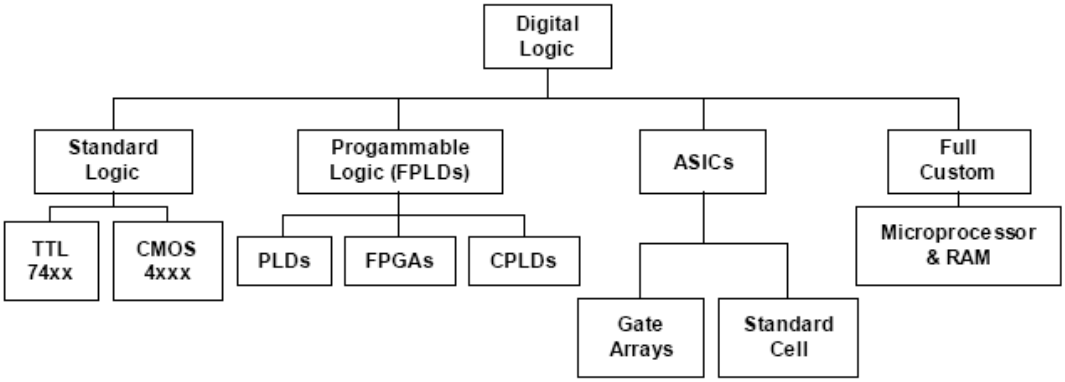


Figura 1. Alternativas para la implementación de un sistema digital en un circuito integrado.
(Gráfico extraído de [2])

Tradicionalmente el diseño de sistemas digitales se realizaba mediante el uso de circuitos integrados, tales como SSI y MSI TTL. La característica principal de estos circuitos es que son empleados para realizar funciones fijas definidas por el diseñador del sistema.

Para la realización de sistemas complejos que exigen un elevado número de circuitos integrados (CI), se utilizan circuitos diseñados a medida que únicamente sirven para implementar una aplicación en concreto. Estos circuitos son los denominados ASIC (Application Specific Integrated Circuit) y, en general, son producidos por los fabricantes de circuitos integrados con las especificaciones proporcionadas por el usuario.

Los equipos realizados mediante ASICs ocupan menos espacio, son más fiables, consumen menos energía y, realizando la fabricación en grandes series, resultan más baratos que los equipos equivalentes diseñados con circuitos integrados.

Por otra parte, existe la posibilidad de realizar sistemas electrónicos completamente personalizables, diseñados a petición del cliente para resolver una determinada aplicación, integrando todos los elementos necesarios en un sólo chip, lo que se denomina sistema *full-custom(VLSI).* De esta forma, se consiguen sistemas para ser empleados en aplicaciones que requieran una alta complejidad computacional, si bien es necesario el uso de complejas herramientas de diseño basadas en el uso del ordenador, además de ser sistemas que conllevan altos costes en el diseño y fabricación debido a los altos plazos de diseño. Por todo ello, resulta una solución válida para la producción de grandes series (microprocesadores, chips de memoria RAM, etc), pero poco efectiva para diseñadores.

La necesidad de obtener diseños de forma rápida ha llevado a la creación y evolución de los dispositivos de lógica programable, con las ventajas que estos conllevan en cuanto a simplicidad de uso, bajo coste y flexibilidad. La idea comenzó con las memorias de sólo lectura (ROM) y ha evolucionado hasta sistemas que integran dispositivos programables, memorias y lógica configurable, todo ello en un mismo chip.

El gráfico presentado en la Figura 2 muestra el equilibrio entre los parámetros de las diferentes tecnologías.

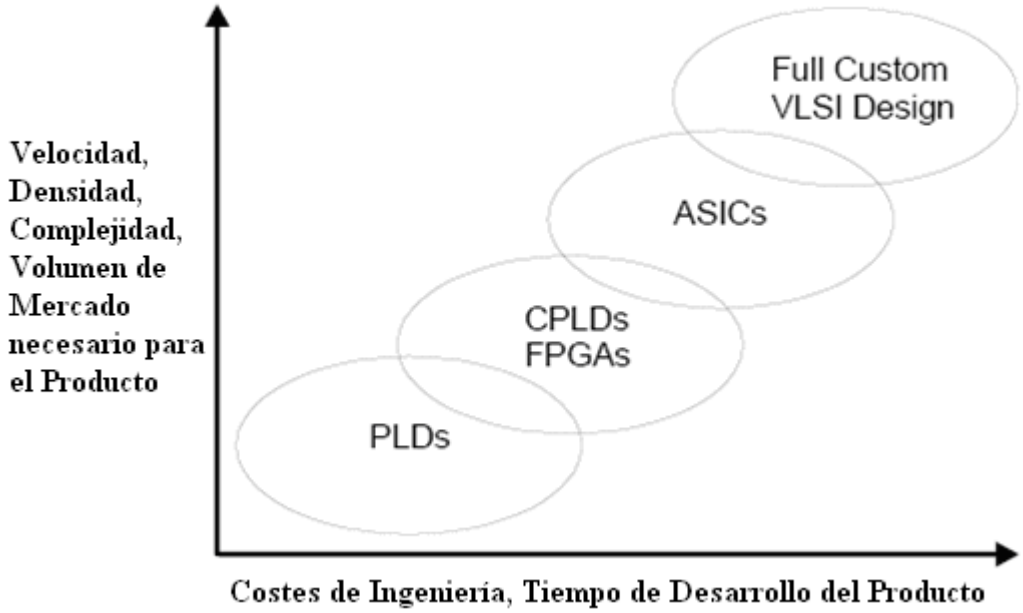


Figura 2. Características de las distintas tecnologías
(Gráfico extraído de [2])

2.2.- EVOLUCIÓN DE LA LÓGICA PROGRAMABLE

2.2.1.- CONCEPTO DE LÓGICA PROGRAMABLE

La lógica programable se basa en dispositivos de lógica programable (PLD), los cuales son circuitos formados por matrices de puertas lógicas y flip-flops, y pueden ser configurados por el usuario para ejecutar una o varias funciones lógicas, consiguiendo reemplazar varios circuitos integrados estándares o de funciones fijas.

2.2.2.- CLASIFICACIÓN DE LOS PLD'S

A continuación, se muestra la evolución en cuanto a dispositivos de lógica programable, desde estructuras simples como ROMs, hasta CPLDs (Complex Programmable Logic Device) y FPGAs (Field Programmable Gate Array).

Los PLD's se pueden clasificar en:

- SPLD's (Simplex Programmable Logic Device)
- CPLD's (Complex Programmable Logic Device)
- FPGA's (Field Programmable Gate Arrays)

2.2.2.1.- SPLD – Simplex Programmable Logic Device

Son dispositivos basados en el empleo de matrices lógicas programables. Es el tipo de PLD empleado para realizar tareas relativamente sencillas, como el diseño de circuitos combinacionales y secuenciales no muy complejos, a medida y a un bajo coste.

Existen cuatro tipos de dispositivos SPLD, atendiendo a la forma de programación de los mismos: PROM, PLA, PAL y GAL.

- **Dispositivos PROM (Programmable Read Only Memory)**

Antes de la aparición de dispositivos específicos para el diseño de sistemas digitales, los chips de sólo lectura (ROM) se utilizaban para crear distintas funciones de lógica combinacional. Por tanto, el chip de memoria es equivalente a una serie de circuitos lógicos separados, en número igual a la cantidad de entradas que ésta contenga.

La Figura 3 muestra una arquitectura PROM con matriz de AND's fijas y arreglo programable de puertas OR.

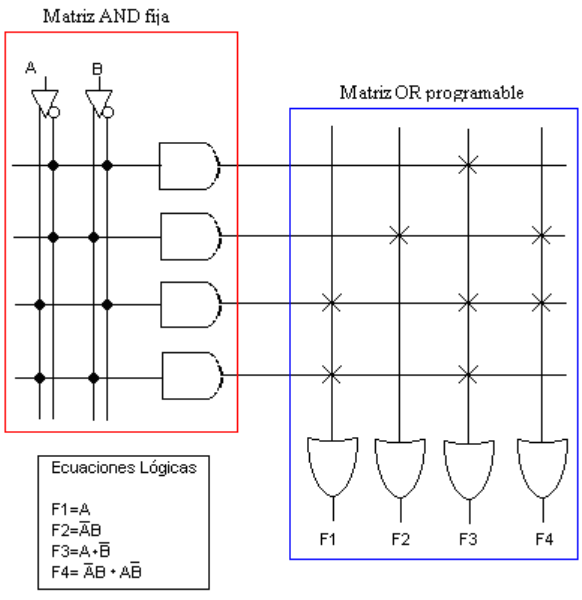


Figura 3. Ejemplo de dispositivo PROM
(Imagen original extraída de [3])

El uso de chips de memoria PROM conlleva una serie de desventajas respecto a los circuitos lógicos dedicados:

- Mayor lentitud.
- Mayor consumo de potencia.
- Uso ineficiente del espacio.
- No pueden ser usados para circuitos de lógica secuencial por sí solos, ya que no contienen biestables. Para ello es necesario el uso de registros TTL externos.

- **Dispositivos PLA (Programmable Logic Array)**

Estos dispositivos también son conocidos como FPLA (Field Programmable Logic Array), debido a que es el usuario, y no el fabricante, quien los programa.

Los dispositivos PLA son PLD's en los cuales se pueden programar las uniones tanto en la matriz de puertas AND como en la matriz de puertas OR. Debido a ello, son los dispositivos más flexibles, si bien presentan desventajas en cuanto a tamaño y velocidad, debido a la inclusión de transistores adicionales en la matriz de puertas OR. Principalmente son usados para la construcción de máquinas de estados. Para otro tipo de aplicaciones, los dispositivos PAL resultan más efectivos.

La siguiente figura muestra un ejemplo básico de arquitectura PLA:

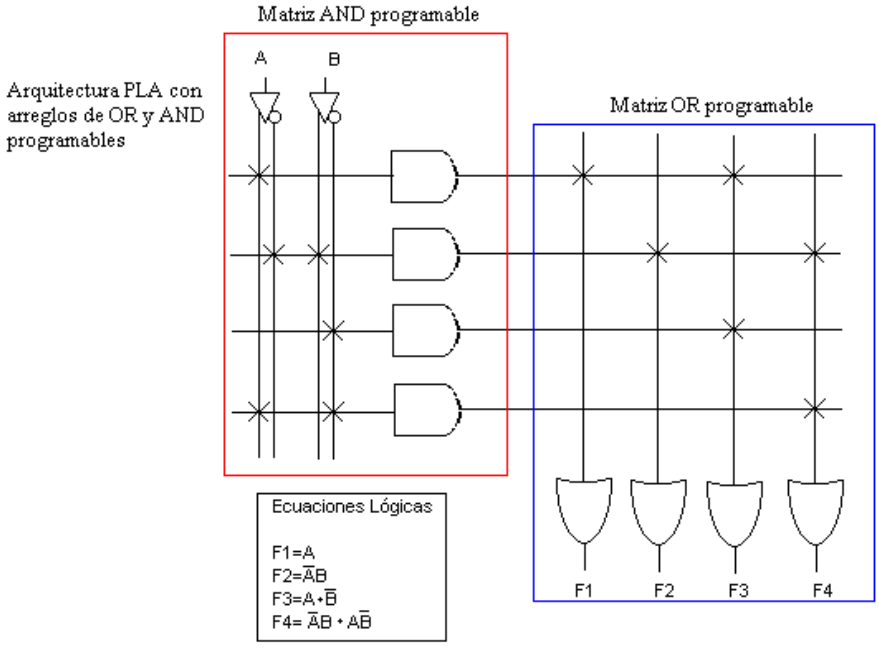


Figura 4. Ejemplo de arquitectura PLA
(Imagen original extraída de [3])

- **Dispositivos PAL (Programmable Array Logic)**

Los dispositivos PAL mejoran las desventajas de los PLA en cuanto a retardos en los fusibles, debido a que únicamente se programa la matriz de puertas AND dejando el arreglo de OR fijo. Para compensar la reducción de flexibilidad, debida al nivel OR fijo, se produjeron diversas variaciones, modificando el número de entradas y salidas, así como variaciones en los tamaños de las puertas OR.

La Figura 5 muestra la arquitectura básica de un dispositivo PAL:

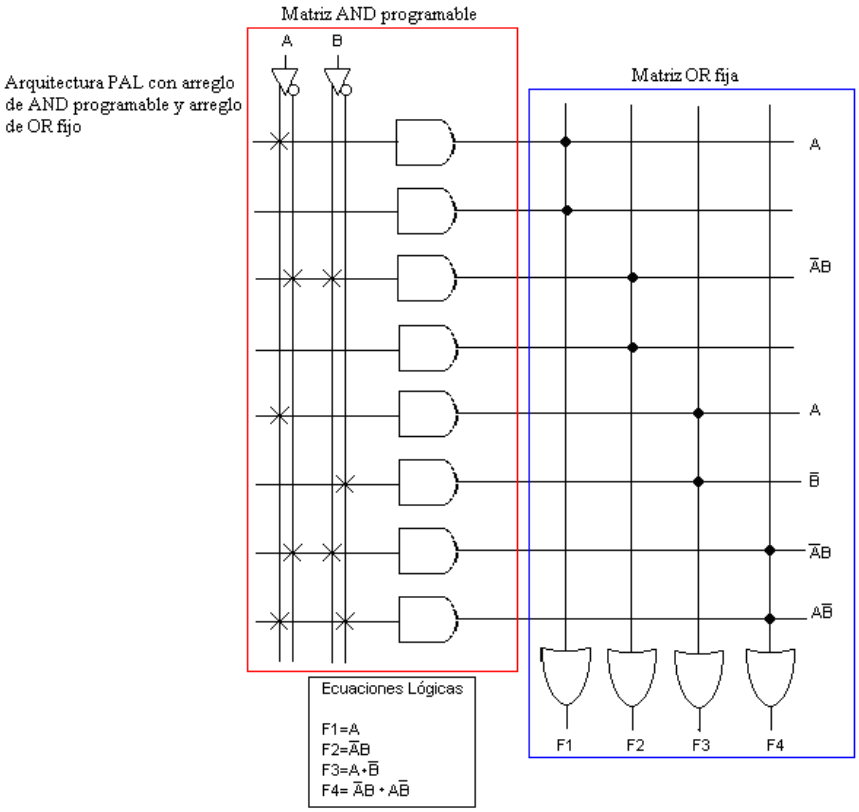


Figura 5. Ejemplo de estructura de un dispositivo PAL
(Imagen original extraída de [3])

La introducción del dispositivo PAL aumentó la eficiencia de los diseños de lógica programable y, de hecho, se trataba del dispositivo programable más común. Es implementado con tecnología TTL y ECL, además de, normalmente, contener flip-flops conectados a las salidas de las puertas OR para poder realizar circuitos secuenciales.

Estos dispositivos resultan de especial importancia ya que son la base de algunas de las más nuevas y complejas arquitecturas surgidas posteriormente.

- **Dispositivos GAL (Generic Array Logic)**

Los dispositivos GAL surgen como una innovación de PAL. Este dispositivo tiene propiedades lógicas similares al PAL pero con las diferencias de poder ser borrado y reprogramado. Es implementado mediante tecnología E^2CMOS (Electrically Erasable CMOS).

2.2.2.2.- CPLD – Complex Programmable Logic Device

Los dispositivos CPLD presentan similares características a los SPLD, con la diferencia de ser versiones superiores a éstos, lo que conlleva un aumento en la capacidad lógica, aproximadamente 50 veces superior respecto a los SPLD.

Un CPLD consta de múltiples bloques lógicos conectados mediante una matriz de interconexión interna centralizada. La matriz de interconexión se programa para conectar de forma selectiva distintas señales de entrada a un nivel de puertas AND programable, conectadas a su vez a un nivel de puertas OR fijo.

De la misma manera, las salidas de las puertas OR son conectadas a macrocélulas configurables que permiten al usuario mayor flexibilidad en los diseños.

2.2.2.3.-FPGA - Field Programmable Gate Arrays

Los FPGA's son el resultado de la combinación de dos tecnologías diferentes: los dispositivos lógicos programables (PLD) y los circuitos integrados de aplicación específica (ASIC). De esta manera, se obtiene un dispositivo con un alto nivel de integración, como el caso de los ASIC's, pero también con las ventajas de estandarización y programación por parte del usuario que ofrecen los CPLD's.

Los FPGA's se definen como dispositivos lógicos de propósito general programables por los usuarios, compuestos de bloques lógicos comunicados por interconexiones programables.

La siguiente figura resume el aspecto genérico de un FPGA:

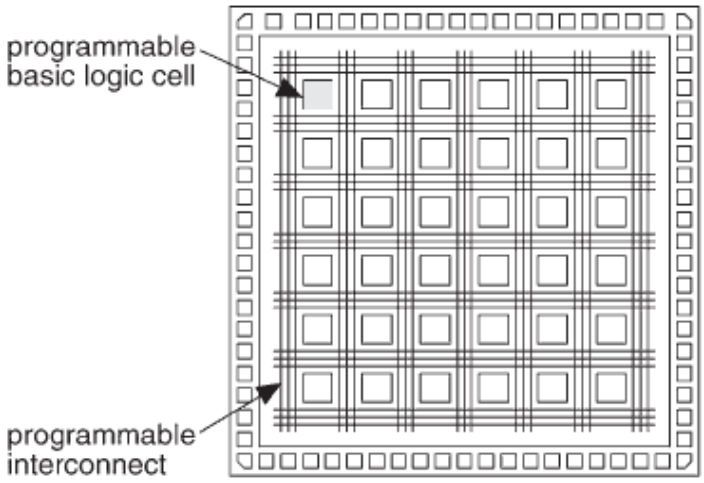


Figura 6. Arquitectura FPGA
(Imagen extraída de [4])

Una tendencia reciente está siendo la combinación de los bloques lógicos e interconexiones de los FPGA's con microprocesadores y periféricos relacionados. Otra alternativa es hacer uso de núcleos de procesadores en la lógica del dispositivo FPGA (ejemplo de Nios y Nios II de Altera).

Muchos FPGA modernos soportan la reconfiguración parcial del sistema, permitiendo que una parte del diseño sea reprogramada mientras las demás partes siguen funcionando.

Hoy en día, los FPGA's están presentes en campos tan diversos como la automoción, la electrónica de consumo ó la investigación espacial. Dicha tecnología es de especial importancia en todas aquellas industrias que requieren computación a alta velocidad.

En el apartado de 'arquitectura de los dispositivos FPGA' se explica más detalladamente la estructura de estos dispositivos.

2.3.- COMPARATIVA FPGA – CPLD – ASIC

La tecnología FPGA fue inventada en el año 1984 por Ross Freeman y Bernard Vonderschmitt, cofundadores de la compañía Xilinx, y surgen como la evolución de los CPLD's.

Tanto los dispositivos CPLD como FPGA contienen un gran número de elementos lógicos programables. Midiendo la densidad de elementos lógicos programables en puertas lógicas equivalentes, los dispositivos CPLD constan del orden de decenas de miles de puertas lógicas equivalentes, mientras que en un dispositivo FPGA se puede llegar hasta cientos de miles e incluso millones de éstas.

Aparte de la medida de la densidad, la diferencia fundamental entre FPGA's y CPLD's radica en su arquitectura. La arquitectura de los CPLD's es más rígida y se basa en una o más sumas de productos programables cuyos resultados son recogidos por un reducido número de biestables síncronos o flip-flops. Por otro lado, la arquitectura de las FPGA's se basa en un gran número de pequeños bloques, donde cada uno de estos bloques es utilizado para implementar sencillas operaciones lógicas, recogidas a su vez por biestables síncronos. De esta manera, la gran libertad disponible para la interconexión de estos bloques dota a las FPGA's de una gran flexibilidad.

Otra diferencia importante entre FPGA's y CPLD's consiste en que en la mayoría de FPGA's se pueden encontrar funciones de alto nivel, como sumadores y multiplicadores e incluso bloques de memoria, integrados en la propia matriz de interconexiones.

Hasta este punto se han resumido las principales diferencias entre FPGA's y CPLD's. A continuación, la Tabla 1 recoge las principales características que definen a los dispositivos FPGA y ASIC, aumentando la información dada en la Figura 2 del presente capítulo, con objeto de observar las similitudes y diferencias existentes entre ambos.

Características	ASIC	FPGA
Tipo de producto	Específico	Estándar
Consumo de energía	Bajo	Medio
Fiabilidad	Alta	Alta
Confidencialidad	Alta	Baja
Coste de desarrollo	Alto	Bajo
Complejidad	Alta	Alta
Programación	Sólo en proceso de fabricación	Por el usuario (reprogramable)
Simulación	Complicada	Fácil
Verificaciones en fases previas al diseño final	Imposible	Posible
Cambios en el diseño	Muy costosos	Posibles en cualquier momento
Comprobación del dispositivo	Específica para cada diseño	Comprobada por el fabricante

Tabla 1. Características ASIC – FPGA

Los parámetros expuestos en la anterior tabla sirven al usuario para decantarse entre una tecnología u otra. En primer lugar, el alto coste de desarrollo y el hecho de la baja tolerancia a errores o reprogramabilidad del sistema suelen ser factores de descarte para el empleo de sistemas basados en ASIC's, en favor del uso de FPGA's, si bien, en segundo lugar, el bajo consumo de potencia suele ser el factor principal que obliga a usar esta tecnología.

2.4.- ARQUITECTURA DE LOS DISPOSITIVOS FPGA

Un dispositivo FGPGA consta de tres tipos de elementos programables:

1. Bloques Lógicos Configurables (CLB, Configurable Logic Block).
2. Matrices de interconexión (SM, Switching Matrix).
3. Bloques de entrada/salida (IOB, Input/Output Blocks).

Además de los elementos mencionados, presentan líneas de interconexión agrupadas en canales verticales y horizontales. Finalmente, existen células de memoria de configuración (CMC, Configuration Memory Cell) distribuidas a lo largo de todo el chip, las cuales permiten almacenar toda la información necesaria para realizar la configuración de los elementos programables. Estas células de configuración suelen consistir en bloques de memoria RAM y son inicializadas en el proceso de carga del programa de configuración.

La siguiente figura muestra la estructura básica de un dispositivo FPGA, indicando los bloques mencionados:

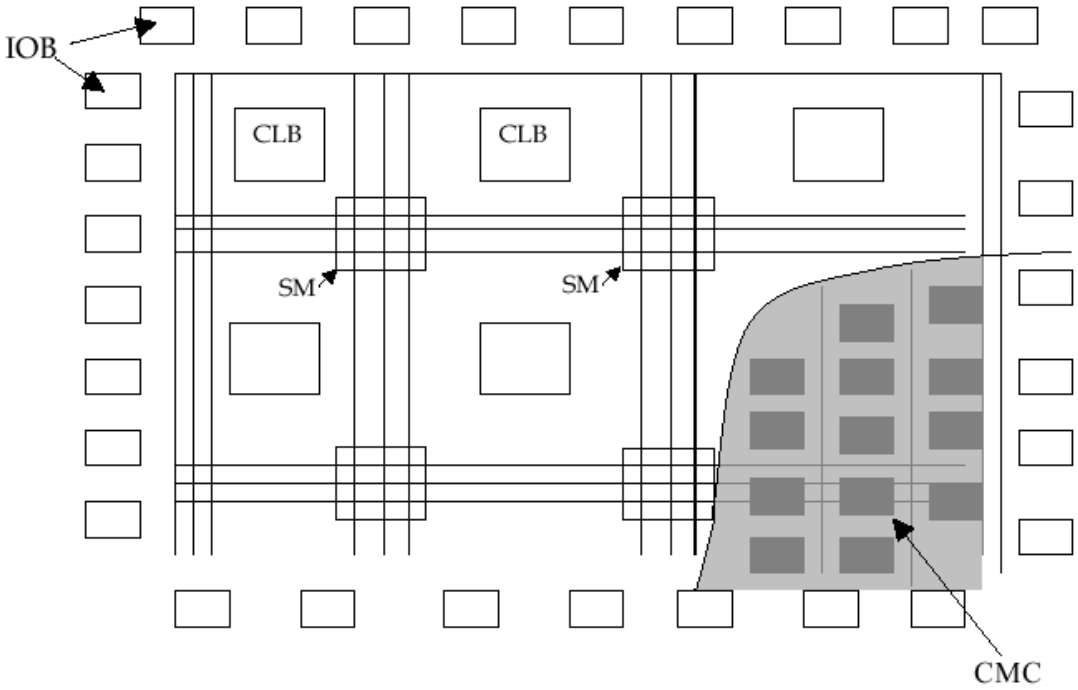


Figura 7. Arquitectura básica FPGA
(Imagen extraída de [5])

2.4.1.- BLOQUES LÓGICOS CONFIGURABLES (CLB)

Son los elementos que constituyen el núcleo del dispositivo FPGA. Cada CLB consta de una parte de lógica combinacional y de una serie de registros de almacenamiento.

Los registros de almacenamiento son empleados en el caso de diseños de lógica secuencial.

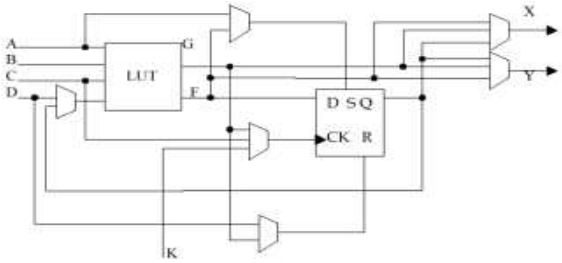


Figura 8. Bloque CLB
(Imagen extraída de [5])

La sección de lógica combinacional, normalmente basada en una LUT (Look Up Table), permite implementar cualquier función booleana a partir de sus variables de entrada. Además, se recurre al uso de multiplexores como elementos adicionales de direccionamiento de los datos del CLB.

2.4.2.-MATRICES DE INTERCONEXIÓN (SM)

Son dispositivos de conmutación distribuidos de forma uniforme por el dispositivo FPGA. Internamente están formados por transistores que permiten la unión de las diferentes líneas de interconexión de propósito general, permitiendo conectar señales de unas líneas a otras.

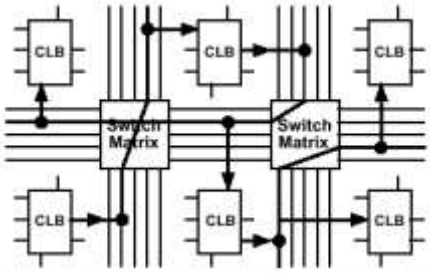


Figura 9. Matrices de interconexión
(Imagen extraída de [5])

Además de las líneas de interconexión de propósito general, existen las denominadas "Líneas Directas" y "Líneas Largas". Las primeras permiten la conexión directa entre bloques, sin la necesidad de pasar por ninguna matriz de interconexión, mientras que las segundas son líneas conductoras de gran longitud, horizontales y verticales, que atraviesan el dispositivo desde sus extremos.

La Figura 10 muestra ejemplos de conexión mediante estas líneas.

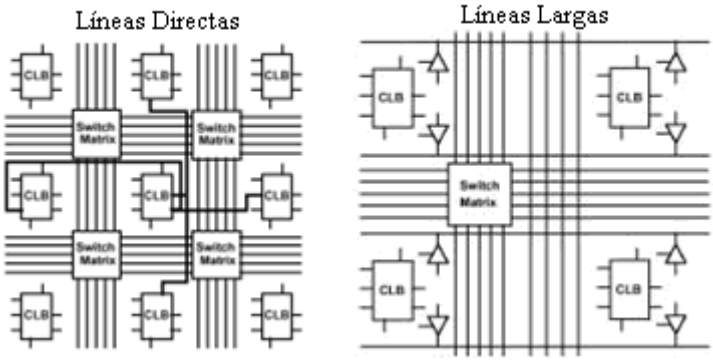


Figura 10. Líneas de interconexión
(Imagen extraída de [5])

2.4.3.- BLOQUES DE ENTRADA/SALIDA (IOB)

Tal como se observa en la Figura 7, el contorno de la FPGA está constituido por bloques de entrada/salida, donde éstos son configurables por parte del usuario. Cada uno de los bloques disponibles puede ser configurado independientemente para funcionar como entrada, salida ó línea bidireccional.

2.5.- HERRAMIENTAS DE PROGRAMACIÓN

Los lenguajes utilizados como código fuente para el desarrollo de diseños con dispositivos de lógica programable son los denominados 'lenguajes de descripción hardware (HDL, Hardware Description Language)'.

También existen programas de captura de esquemas, útiles para la realización de diseños sencillos, sin tener que recurrir a la complejidad de un lenguaje HDL, si bien este sistema no es útil para diseños complicados.

Un lenguaje HDL permite definir las interconexiones y el comportamiento de un circuito electrónico, sin la utilización de diagramas esquemáticos.

Los pasos en el diseño de un sistema son los siguientes:

1.- Definir la tarea o tareas que tiene que realizar el circuito.
2.- Escribir el programa usando un lenguaje HDL.
3.- Comprobación de la sintaxis y simulación del programa.
4.- Programación del dispositivo y comprobación del funcionamiento.

Un rasgo común de estos lenguajes suele ser la independencia del hardware empleado y la modularidad o jerarquía, es decir, una vez implementado el diseño, éste puede ser usado dentro de otro diseño más complejo y con otro dispositivo compatible.

Entre los lenguajes de programación disponibles, destacan los siguientes:

- ABEL HDL: Advanced Boolean Expression Language. Se trata de un lenguaje limitado, utilizado para el diseño de sistemas de baja complejidad.
- Verilog HDL: Es un lenguaje de descripción hardware de alto nivel para dispositivos más complejos. Presenta una sintaxis similar a la del lenguaje de programación C, sin embargo, la característica principal que lo define es que difiere de los lenguajes de programación convencionales, al no seguir una ejecución estrictamente lineal de las sentencias incluidas en los correspondientes diseños.
- VHDL: Acrónimo que representa la combinación VHSIC (Very High Speed Integrated Circuit) y HDL (Hardware Description Language). Al igual que Verilog, es un lenguaje de descripción hardware de alto nivel empleado para el desarrollo de sistemas complejos. Como característica principal, en VHDL existen varias formas de diseñar un mismo circuito:
  - Funcional: Describir la forma en que se comporta el circuito. La descripción es secuencial, asemejando a los leguajes de programación convencionales.
  - Flujo de datos: Descripción de asignaciones concurrentes de señales, es decir, descripción en paralelo.
  - Estructural: Descripción del circuito con instancias de componentes.
  - Mixta: Combinación de todas o alguna de las formas anteriores.

2.6.- MERCADO ACTUAL DE LAS FPGA

El uso de dispositivos FPGA está consolidado a nivel mundial. Los porcentajes de utilización de estos dispositivos respecto a tecnologías anteriores varía según el tipo de industria. Por ejemplo, destacan los altos porcentajes de utilización en las industrias aeroespaciales (64%) ó industrias de vídeo (62%), mientras que el sector de la automoción se sigue decantando por tecnologías anteriores basadas en ASIC's.

El empleo de FPGA's no conlleva la exclusión del resto de ASIC's. En muchos casos, las primeras son empleadas en la realización de prototipos de los ASIC finales.

A su vez, dispositivos de baja densidad como CPLD's pueden ser utilizados junto con las FPGA's en un mismo diseño.

Otro punto importante es la invasión de las FPGA en el mercado de los DSP. Actualmente, cerca de un 40% de los usuarios de FPGA emplean estos chips en la realización de diferentes aplicaciones de DSP.

A continuación se muestra el listado de los principales fabricantes de FPGA:

- Actel
- Altera
- Atmel
- Cypress
- Lattice
- Lucent Technologies
- QuickLogic
- Xilinx

La siguiente figura muestra la cuota de mercado que tienen las respectivas compañías fabricantes:

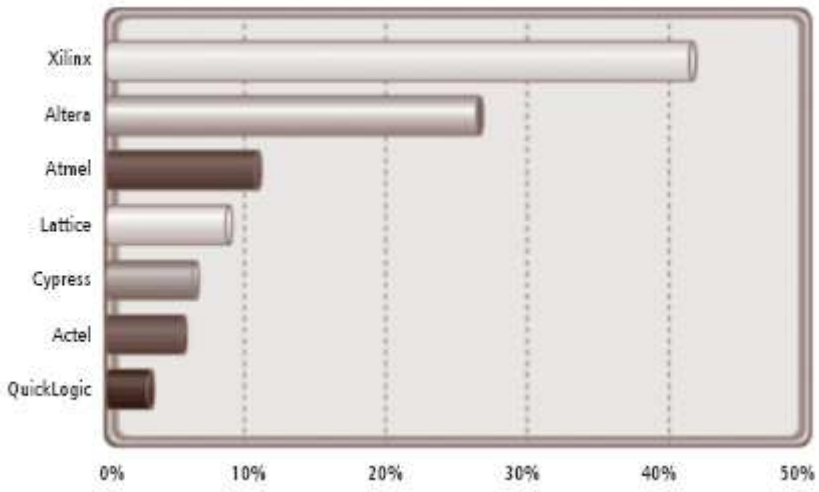


Figura 11. Cuota de mercado de fabricantes de FPGA
(Imagen extraída de [4])

2.7.- CONCLUSIÓN

Los circuitos integrados están omnipresentes en gran cantidad de productos industriales. Entre sus alternativas destacan los dispositivos lógicos programables, que combinan muchos de los beneficios de los circuitos integrados dedicados a funciones específicas con la gran ventaja de poder implementar circuitos 'a la medida' del cliente.

CAPÍTULO 3.

ESTUDIO DE MERCADO

3.- ESTUDIO DE MERCADO

Siguiendo con el apartado 'Mercado Actual de las FPGA' descrito en el capítulo anterior, el cual presenta los principales fabricantes de dispositivos FPGA, en este capítulo se analiza la oferta de cada uno de dichos fabricantes, con el objetivo de realizar un estudio de mercado que permita identificar tanto las compañías como los productos más adecuados para definir un entrono educativo basado en sistemas digitales.

A continuación se hace una recopilación del listado de principales fabricantes:
ACTEL, ALTERA, ATMEL, CYPRESS, LATTICE, QUICKLOGIC, XILINX

Dado que el objetivo de este estudio es encontrar entornos educativos adecuados para su utilización tanto en prácticas de laboratorio como en posibles proyectos adicionales, se valora especialmente aquellas compañías que sigan el siguiente perfil:

- Compañías que ofertan un programa universitario en el que se dan facilidades a las organizaciones académicas para el acceso a sus productos y recursos, consiguiendo importantes descuentos en la adquisición de tarjetas educacionales destinadas al laboratorio.
- Disponer de un software libre de desarrollo de sistemas digitales, descargable desde la correspondiente página Web, con el objetivo de que esté disponible para cualquier persona interesada en el aprendizaje de sistemas digitales.
- Los productos ofertados deben cumplir las siguientes características:
  - Kit completo para la programación del sistema (cable de descarga, CD-ROM con drivers de configuración, etc).
  - PLD de tecnología RAM para tener un número ilimitado de reprogramaciones.
  - Variedad de recursos (LED's, displays, interruptores, pulsadores, diferentes puertos de entrada/salida, etc) con el objetivo de permitir diseñar un completo programa de prácticas para el alumno en cuanto a diseños de sistemas digitales.
  - Manuales de usuario para la ayuda y entendimiento de los dispositivos.

Según los requisitos mencionados, se decide centrar el estudio en las compañías Altera y Xilinx, descartando el resto de compañías, debido a razones como: no ofrecer un programa universitario para la adquisición de sus productos o no ofrecer un software libre de desarrollo. Además, los productos ofertados por estas compañías no están orientados a la realización de prácticas de laboratorio a nivel de iniciación, pese a que son productos potentes, ya que no ofrecen la variedad de recursos necesaria para la realización de un programa de prácticas completo.

A continuación, la Tabla 1, recoge la oferta de las compañías Altera y Xilinx en cuanto a tarjetas educacionales orientadas a prácticas de sistemas digitales en laboratorio, exponiendo las características de cada producto, con el objetivo de realizar la comparación entre los mismos.

VER TABLA ‘ESTUDIO DE MERCADO’

Analizando la oferta presentada en la tabla anterior, en primer lugar, cabe destacar que todas las tarjetas tienen la capacidad de ser utilizadas bajo el software libre ofrecido por las respectivas compañías, excepto la tarjeta XUPV2P de la compañía Xilinx.

Las tarjetas básicas de ambas compañías, UP2 (Altera) y Basys (Xilinx), presentan unas características suficientes para la realización del conjunto de prácticas expuesto en el Capítulo 7 del presente proyecto, si bien para su posible utilización en proyectos de mayor complejidad tienen importantes carencias, especialmente en cuanto a cantidad de memoria y puertos de comunicación.

La tarjeta educacional DE0 ofrecida por la compañía Altera, se trata de una placa dirigida a estudiantes para el ensayo y realización de sencillos diseños de apoyo, ofreciendo menores recursos para la realización de un programa de prácticas de laboratorio.

En cuanto a los dispositivos programables integrados en las respectivas tarjetas, las familias Cyclone II, por parte de Altera, y Spartan 3E, por parte de Xilinx, presentan dispositivos con características similares. Ambas familias presentan dispositivos capaces de implementar aplicaciones con alto volumen de datos y alta flexibilidad en los diseños, todo ello a un bajo coste.

Por otra parte, las tarjetas DE3 (Altera) y XUPV5-LX110T (Xilinx), constan de dispositivos programables integrados con unos recursos muy superiores a los necesarios para la utilización en un laboratorio de prácticas, lo que conlleva el consecuente aumento de precio de las placas.

Centrando el análisis en las tarjetas de nivel intermedio (DE1, DE2, DE2-70, Nexys 2 y Spartan 3E Starter) se obtienen las siguientes conclusiones:

- Tanto los dispositivos de las placas de Altera como de Xilinx cuentan con núcleos de procesadores integrados (NIOS II y MicroBlaze-PicoBlaze) que permiten obtener mayores prestaciones en cuanto a flexibilidad y velocidad de procesamiento.
- Las tarjetas de Xilinx ofrecen mayor cantidad de memoria RAM (excepto en el caso de la tarjeta DE2-70 de Altera) y memoria Flash, si bien las placas de Altera cuentan con la posibilidad de ampliación de memoria a través de tarjetas externas.
- Las placas de Altera ofrecen mayores posibilidades en cuanto a puertos de comunicación, especialmente en el caso de puertos de entrada/salida de audio y vídeo.
- En las placas de Altera existe un mayor número de LED's, displays, interruptores y pulsadores que en el caso de las tarjetas de la compañía Xilinx, de hecho la tarjeta Spartan 3E Starter de Xilinx no ofrece estos recursos. Esto es un factor importante de cara a la realización de prácticas de laboratorio.
- Otro factor importante es el precio de las tarjetas, donde en el caso de Xilinx las tarjetas tienden hacia precios más reducidos.

Además de las tarjetas educacionales vistas hasta el momento, las compañías ofrecen posibilidades de expansión, a través de distintos productos, según la finalidad ó aplicación que se le quiera dar al sistema.

En este caso, dado que la última parte del presente proyecto trata de una aplicación orientada al trabajo con imágenes digitales, se decidió realizar una búsqueda para elegir los

accesorios necesarios para este tipo de aplicación. En este campo, la compañía Altera ofrece mayores posibilidades que Xilinx, por lo que, a continuación, se analizan los productos existentes de este fabricante.

En concreto, el material buscado se corresponde con un kit que consta de una tarjeta educacional (de las expuestas en la Tabla 1), una cámara digital de 5 megapíxeles y un panel táctil LCD de 4.3 pulgadas.

La Figura 1 muestra una imagen de este tipo de kit:

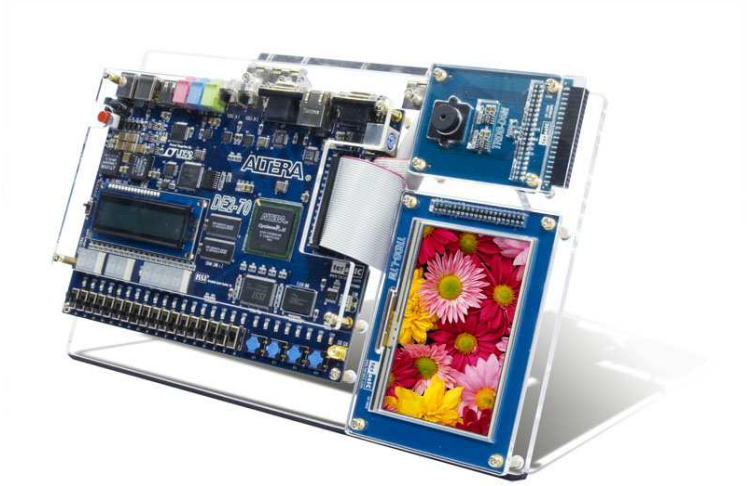


Figura 1. Kit DE2-70 + Cámara Digital 5M + Panel LCD
(Imagen extraída del enlace Web de Altera Corporation)

La siguiente tabla resume las diferentes posibilidades, con sus respectivos precios, para conformar el kit:

Producto	Precio Unitario	Precio Programa Universitario
Tarjeta DE2-70 + Cámara Digital 5M + Panel LCD 4.3”	$854	$584
Tarjeta DE2 + Cámara Digital 5M + Panel LCD 4.3”	$750	$524
Tarjeta DE1 + Cámara Digital 5M + Panel LCD 4.3”	$405	$380

Tabla 2. Kits para aplicación de vídeo

Como se puede observar en la Tabla 2, la diferencia entre un kit y otro radica en la elección de la tarjeta educacional que lo compone. En este caso, observando la Tabla 1, se ve como la tarjeta educacional DE2-70 es la que consta de mayores recursos, al disponer de un dispositivo más potente que el del resto de tarjetas de los distintos kits, así como de mayor cantidad de memoria. Estos factores han sido los fundamentales a la hora de escoger el primero de los kits (Tarjeta DE2-70 + Cámara Digital 5M + Panel LCD 4.3”) expuestos en la Tabla 2, aunque el precio sea mayor, ya que la diferencia de potencial entre esta placa y el resto hace que tenga más posibilidades a la hora de programarla y de realizar distintos diseños.

Por lo tanto, se optó por la compra del kit **“Tarjeta DE2-70 + Cámara Digital 5M + Panel LCD 4.3”** como herramienta para la aplicación final del presente proyecto.

Cabe decir también que se puede realizar la adquisición tanto de la cámara digital como del panel táctil por separado, en el posible caso de que se disponga de la tarjeta educacional correcta con anterioridad. La siguiente tabla muestra los precios de estos elementos individualmente:

Producto	Precio Unitario
Cámara Digital 5M	$85
Panel LCD 4.3”	$170

Tabla 3. Precios de cámara digital y panel LCD

CAPÍTULO 4.

COMPARATIVA 'MAX+PLUS II – QUARTUS II'

4.- COMPARATIVA 'MAX+plus II – QUARTUS II'

4.1.- INTRODUCCIÓN

Este capítulo permite el estudio del programa QUARTUS II en comparación con MAX+PLUS II, viendo las posibilidades que ofrece el nuevo programa y su metodología de trabajo.

La comparación incluye, como principal aspecto a tener en cuenta, las familias de dispositivos lógicos programables que son soportadas por cada uno de los programas mencionados. Además, se presenta el nuevo entorno de trabajo y se facilita la adaptación al nuevo software de desarrollo.

4.2.- FAMILIAS DE DISPOSITIVOS SOPORTADOS

El software QUARTUS II soporta los dispositivos actuales del mercado y la mayoría de dispositivos soportados por el software MAX+PLUS II, si bien el nuevo software no es capaz de soportar ningún tipo de dispositivo o paquete que haya quedado obsoleto.

Las familias de dispositivos soportados por cada uno de los entornos de desarrollo se exponen en la siguiente tabla:

Dispositivo Soportado	Quartus II	Max+Plus II
Arria® GX	SI	NO
Stratix® Series	SI	NO
Cyclone® Series	SI	NO
HardCopy® Series	SI	NO
MAX® II	SI	NO
Classic™	NO	SI
MAX 3000A	SI	SI
MAX 7000S/AE/B	SI	SI
MAX 7000E	NO	SI
MAX 9000	NO	SI
ACEX® 1K	SI	SI
FLEX® 6000	SI	SI
FLEX 8000	NO	SI
FLEX 10K	SI(1)	SI
FLEX 10KA	SI	SI
FLEX 10KE	SI(1)	SI
APEX™ II	SI	NO
APEX 20K	SI	NO

Tabla 1: Familias de dispositivos soportados

(1) Algunas versiones no nos soportadas.

4.3.- VISIÓN GENERAL DE DISEÑO CON QUARTUS II

El software de diseño Quartus II incluye las siguientes herramientas globales para el desarrollo de los respectivos diseños:

- Project Navigator
- Node Finder
- Tcl Console
- Messages
- Status

A continuación se explica brevemente cada una de ellas:

- **Project Navigator**

La ventana 'Project Navigator' es similar a la ventana 'Hierarchy Display' en Max+Plus II. La utilidad de esta ventana es la de dar información sobre los proyectos que se están llevando a cabo. En el caso de 'Project Navigator', además, se incluye información adicional, tales como registros y recursos de memoria utilizados.

- **Node Finder**

La ventana 'Node Finder' ofrece unas funcionalidades equivalentes al cuadro de diálogo 'Search Node Database' del programa Max+Plus II. Esta herramienta permite buscar y utilizar cualquier elemento almacenado en la base de datos del proyecto.

- **Tcl Console**

La herramienta 'Tcl Console' permite la entrada de comandos y scripts Tcl que permiten la realización de asignaciones, análisis temporales, obtención de información acerca de los distintos diseños, además de poder automatizar y personalizar totalmente la forma de trabajo con los elementos disponibles en Quartus II. Esta funcionalidad no es posible realizarla con el software Max+Plus II.

- **Messages**

La ventana 'Messages' es similar a la ventana 'Message Processor' de Max+Plus II. Esta herramienta permita la visualización de información de los diferentes procesos, mensajes de alerta o mensajes de error. En el software Quartus II, también es posible utilizar esta herramienta para localizar algún nodo relativo a un mensaje perteneciente a varias ventanas de trabajo.

- **Status**

La ventana 'Status' muestra información similar a la de la ventana 'Compiler' en Max+Plus II. Permite la visualización del progreso y del tiempo transcurrido de cada etapa de la compilación del proyecto.

De la misma manera que los nombres y algunas de las funcionalidades de las herramientas mencionadas anteriormente han sufrido variaciones, la apariencia del entorno de desarrollo también ha sido actualizada.

La Figura 1 muestra un ejemplo del nuevo entorno de desarrollo.

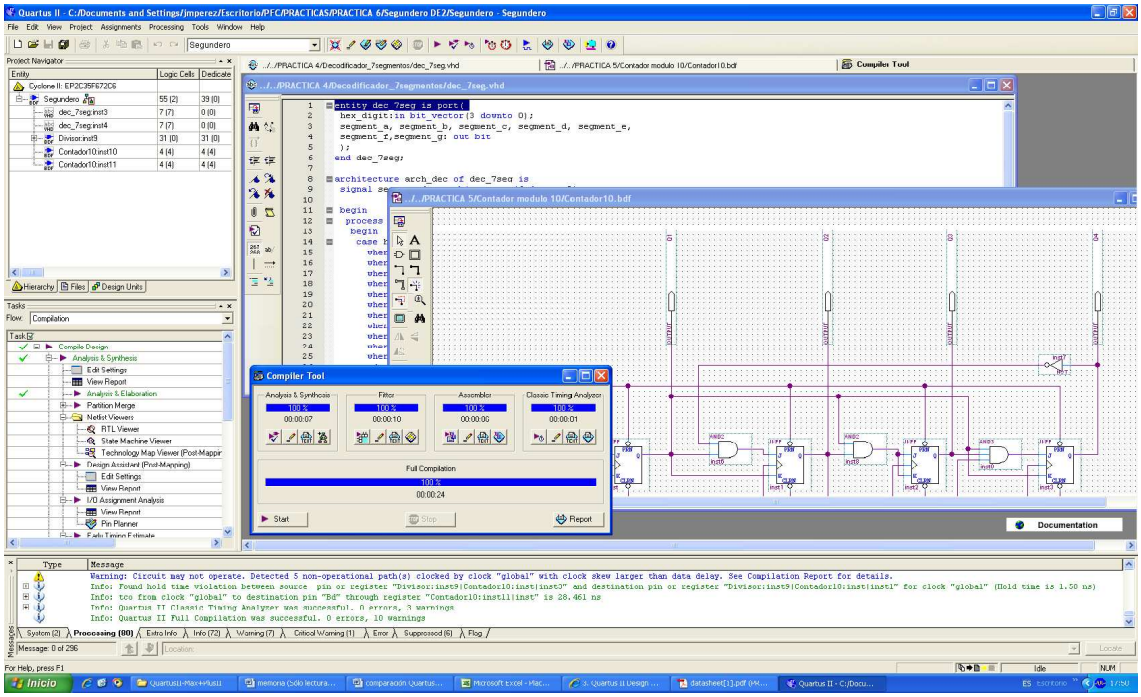


Figura 1. Entorno de Quartus II

Es interesante comentar que para usuarios acostumbrados a trabajar en el entrono de Max+Plus II, el software Quartus II permite modificar su apariencia. De esta manera, se consigue que la barra de herramientas donde se alojan los respectivos iconos que dan acceso a las diferentes herramientas del software, mantenga los mismos símbolos que los utilizados en el caso de Max+Plus II.

Las siguientes figuras muestran las distintas barras de herramientas, consecuencia de utilizar una apariencia u otra:

Figura 2. Barra de herramientas de Max+Plus II
(Imagen extraída de [8])

Figura 3. Barra de herramientas de Quartus II

Para realizar este cambio y obtener mayor información al respecto, consultar el capítulo 6: 'Manual del Software Quartus II' en el que se describen los pasos a seguir.

A continuación, con objeto de facilitar la adaptación al nuevo entorno de trabajo, se resumen, en la Tabla 2, las equivalencias entre los iconos de los comandos más utilizados a la hora de trabajar con estos sistemas.

MAX+PLUS II Software	QUARTUS II Software
Hierarchy Display	View menu, Utility Windows, Project Navigator
Graphic Editor	Block Editor
Symbol Editor	Block Symbol Editor
Text Editor	Text Editor
Waveform Editor	Waveform Editor
Floorplan Editor	Timing Closure Floorplan
Compiler	Compiler Tool
Simulator	Simulator Tool
Timing Analyzer	Timing Analyzer Tool
Programmer	Programmer
Message Processor	Messages
Open Project	Open Project
Set project To Current File	Set as Top-Level Entity
Save & Check	Start Analysis & Synthesis
Save & Compile	Start Compilation
Save & Simulate	Start Simulation

Tabla 2. Iconos de comandos Max+Plus II – Quartus II
(Imágenes de iconos extraídos de [8])

4.4.- CONCLUSIÓN

El software de desarrollo Quartus II está dotado de todas las capacidades y características que posee el software Max+Plus II, pero preparado para la realización de diseños con dispositivos lógicos que exigen unos mayores recursos y capacidad de procesamiento.

CAPÍTULO 5.

MANUAL TARJETAS ‘UP2’ – ‘DE2’ DE ALTERA

5.- MANUAL TARJETAS 'UP2' – 'DE2' DE ALTERA

Este capítulo muestra una descripción de las tarjetas educacionales disponibles en el laboratorio: UP2 y DE2, ambas de la compañía Altera.

Asimismo, se hace una descripción detallada de las familias de dispositivos lógicos programables FLEX10K y Cyclone II, ya que están involucradas en las respectivas tarjetas.

Por último, se resumen las características básicas de los dispositivos que llevan integrados ambas placas con el objetivo de poder establecer una comparación entre ellos y ver la evolución que supone el paso de utilizar una tarjeta u otra.

5.1.- TARJETA EDUCACIONAL UP2

5.1.1.- DESCRIPCIÓN DE LA TARJETA EDUCACIONAL UP2

La tarjeta educacional UP2 presenta útiles características que facilitan el diseño de circuitos electrónicos digitales. Se trata de una tarjeta de entrenamiento que incluye diferentes componentes, los cuales serán explicados posteriormente, que posibilitan la realización de diversas prácticas. Esta tarjeta se caracteriza por incorporar dos dispositivos lógicos programables (PLD's) de dos familias diferentes de Altera, MAX7000 y FLEX10K. El primero de los dispositivos es el EPM7128S (correspondiente a la familia MAX7000) y el segundo el EPF10K70 (de la familia FLEX10K). Ambos dispositivos pueden ser programados o configurados sin necesidad de sacarlos de la placa. En el caso de utilizar esta tarjeta para la realización de prácticas, se utilizará el dispositivo EPF10K70, ya que el dispositivo EPM7128S está basado en tecnología EEPROM, por lo que el número de veces que se puede programar y borrar es limitado, siendo además este dispositivo menos potente que el EPF10K70.

La Figura 1 muestra una imagen de la tarjeta educacional UP2.

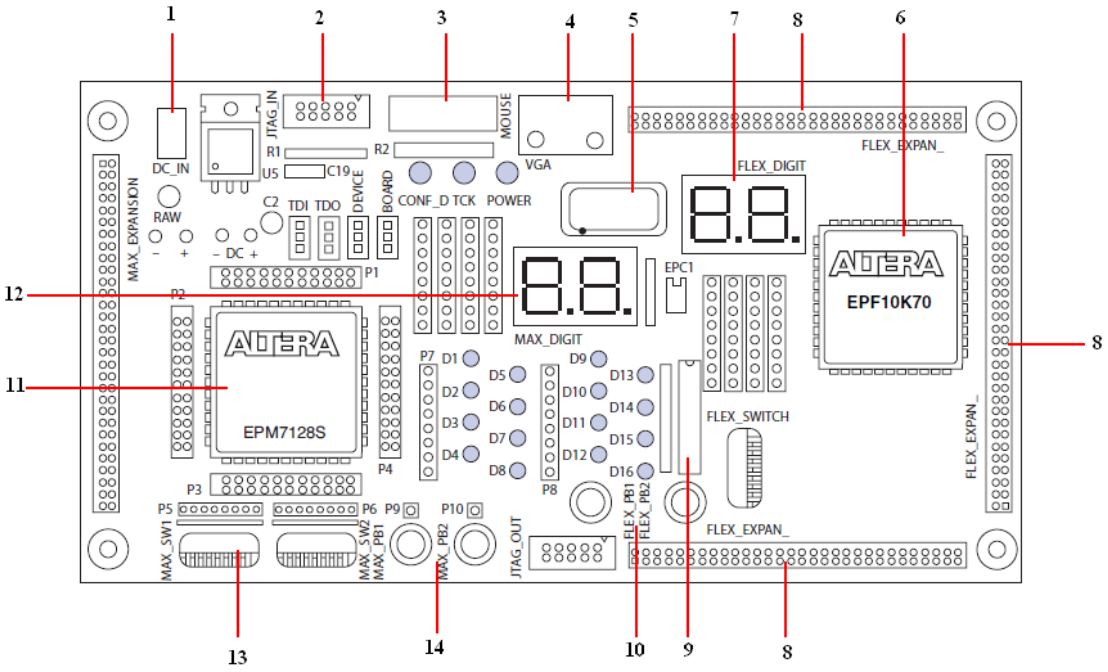


Figura 1. Tarjeta educacional UP2
(Imagen original extraída de [3])

Observando la Figura 1, el listado de componentes contenidos en la placa es el siguiente:

1- Alimentación de 9v.
2- Puerto JTAG de entrada.
3- Puerto PS/2.
4- Puerto VGA.
5- Oscilador de 25.175 MHz.
6- Dispositivo FLEX10K70 de Altera.
7- Display 7 segmentos correspondiente al dispositivo FLEX10K70.
8- Salidas/Entradas externas para el dispositivo FLEX10K70.
9- Interruptores para el dispositivo FLEX10K70.
10- Pulsadores del dispositivo FLEX10K70.
11- Dispositivo MAX7128S de Altera.
12- Display 7 segmentos correspondiente al dispositivo MAX7128S.
13- Interruptores para el dispositivo MAX7128S.
14- Pulsadores del dispositivo MAX7128S.

La placa UP2 de Altera dispone de dos dispositivos, cada uno de ellos con sus diferentes componentes de entrada y salida, como son: diodos LED, diplays de 7 segmentos, pulsadores, etc. Los elementos correspondientes al dispositivo FLEX10K70 están directamente conectados a los pines de dicho dispositivo, por lo que su conexión viene impuesta, sin posibilidad de ser modificada. En el caso del dispositivo MAX7128S los diferentes elementos no están directamente conectados al dispositivo (excepto los displays), por lo que es posible configurar estos elementos para su utilización tanto desde su propio dispositivo como también desde el FLEX10K70.

En el manual original de la placa UP2 (ofrecido por Altera) se indica que todos los diodos LED que inicialmente pertenecen al dispositivo MAX7128S, están únicamente conectados por sus ánodos, dejando la conexión de los cátodos libre, por lo que estos diodos podrán ser conectados sin ningún problema tanto al MAX7128S como al FLEX10K70 mediante cables. De esta forma, si los diodos LED están conectados por sus ánodos, bastará con un cero lógico por los mismos para iluminarlos. Esto también sucede con los diodos LED de los displays.

Algo similar sucede con los bancos de interruptores y pulsadores del dispositivo MAX7128S. Los interruptores dan una señal a nivel alto cuando dicho interruptor está abierto, y una señal a nivel bajo si está cerrado. Dado que en el MAX7128S las señales procedentes de los interruptores se conducen mediante cables, se pueden llevar estas señales a ambos dispositivos.

En el caso de los pulsadores, éstos dan una señal a nivel alto mientras no sean presionados, de lo contrario, al ser presionados, ofrecen una señal a nivel bajo. Al igual que pasa con los interruptores, los dos pulsadores que en principio pertenecen al dispositivo MAX7128S también podrán ser usados por el FLEX10K70, realizando las oportunas conexiones mediante cables a los pines de este dispositivo en lugar del MAX7128S.

En resumen, aunque no se utilizará el dispositivo MAX7128S para la realización de prácticas, debido a su limitado número de reprogramaciones, sí que es posible utilizar los elementos que lo acompañan, al poder ser éstos conectados por medio de cables con el dispositivo FLEX10K70 a través de sus puertos de expansión.

A continuación, debido a su importancia, se explica con detalle la familia de dispositivos FLEX10K.

5.1.2.-DISPOSITIVOS FLEX10K

La familia de dispositivos programables integrados FLEX10K pertenece a la categoría de las FPGA's y ofrece la flexibilidad de la lógica programable tradicional junto con la utilización de matrices de puertas integradas.

Los dispositivos FLEX10K están basados en elementos reconfigurables SRAM de tecnología CMOS. Estos dispositivos constan de hasta 250000 puertas, lo que proporciona la densidad, velocidad y rasgos necesarios para integrar sistemas completos en un único dispositivo.

Los dispositivos FLEX10K son reconfigurables, es decir, es posible realizar todas las pruebas necesarias antes de obtener el resultado final. Pueden ser configurados en la misma tarjeta para la función específica que sea requerida.

La arquitectura FLEX10K es similar a la de matrices de puertas integradas. Como en el caso de matrices de puertas estándar, las matrices de puertas integradas aplican la lógica general, generando la arquitectura denominada "mar de puertas". Además, las matrices de puertas integradas constan de áreas dedicadas a la implementación de grandes funciones especializadas, aumentando la velocidad en comparación con las matrices de puertas estándar. Sin embargo, generalmente las megafunciones integradas no pueden ser adaptadas, lo que limita la flexibilidad en el diseño para el programador. Por el contrario, los dispositivos FLEX10K son programables, proporcionando al diseñador un total control tanto sobre las megafunciones integradas como sobre la lógica general. A su vez, se facilita la realización de cambios en el diseño durante la fase de depuración del mismo.

Cada dispositivo FLEX10K contiene una matriz integrada y una matriz lógica. La matriz integrada se utiliza para implementar distintas funciones de memoria o funciones de lógica compleja, como procesado digital de señal (DSP), microcontroladores y funciones de transformación de datos. La matriz lógica, en cambio, realiza la función de un "mar de puertas", es decir, se utiliza para implementar la lógica general, como pueden ser contadores, sumadores, máquinas de estados o multiplexores. La combinación de ambas matrices proporciona un buen rendimiento y densidad de matrices de puertas integradas, permitiendo a los diseñadores llevar a cabo un sistema completo en un solo dispositivo.

Los dispositivos FLEX10K contienen un interfaz que permite la configuración en serie o en paralelo desde el microprocesador, tanto de manera síncrona como asíncrona.

A continuación se exponen distintas características de la familia de dispositivos FLEX10K:

- Proporciona la integración de Sistema en un Chip Programable (SOPC).
- Presenta entre 10000 y 250000 puertas típicas.
- Consta de un máximo de 40960 bits de memoria RAM: 2048 bits por cada bloque de matrices integradas (EAB), las cuales pueden ser utilizadas sin reducir la capacidad lógica.

En el punto 5.3 se presentan las características concretas del dispositivo EPF10K70.

5.1.3.- CONFIGURACIÓN DEL DISPOSITIVO EPF10K70

El procedimiento completo para realizar la descarga de un determinado diseño en el dispositivo EPF10K70 se muestra en el capítulo 'Manual de Quartus II'.

La tarjeta educacional UP2 contiene cuatro jumpers de tres pines (TDI, TDO, DEVICE y BOARD) para seleccionar los diferentes modos de configuración. Se puede configurar la placa para programar cada uno de los dispositivos por separado, los dos juntos o varias tarjetas conectadas en cascada.

En este caso interesa la configuración que permite volcar el programa en el dispositivo EPF10K70, para lo cual la posición de los jumpers deberá ser la que se indica en la Figura 2.

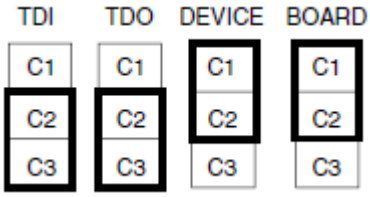


Figura 2. Configuración de los jumpers de la tarjeta
(Imagen extraída de [3])

La configuración del dispositivo se realiza mediante el cable especial de Altera denominado ByteBlaster, el cual se conectará al puerto JTAG de entrada de la tarjeta UP2 y al puerto paralelo del PC con el que se está trabajando, tal y como se aprecia en la Figura 3. Este cable enviará los datos de configuración desde el software Quartus II hasta el dispositivo.

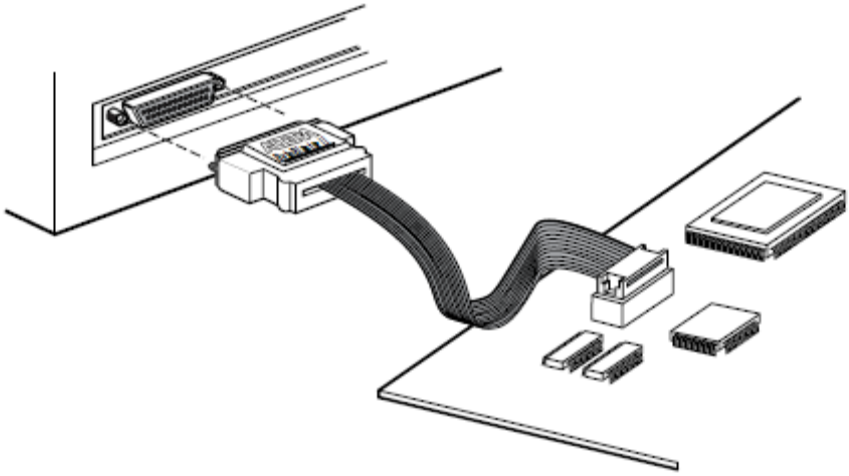


Figura 3. Conexión mediante cable ByteBlaster
(Imagen extraída de [3])

5.2.- TARJETA EDUCACIONAL DE2

En primer lugar, antes de realizar la correspondiente descripción de la placa, se expone la serie de pasos a seguir para la correcta instalación del controlador de la tarjeta DE2 en el PC.

5.2.1.- INSTALACIÓN. CONTROLADOR USB-BLASTER

Los elementos necesarios para la instalación de la tarjeta DE2, proporcionados en el mismo paquete, no son más que el adaptador de alimentación de 9 voltios y el cable USB.

Conectar el cable de alimentación a su correspondiente conector en la placa. Conectar el cable USB en el conector dedicado para ello en la placa, situado junto a la entrada de alimentación, denominado USB Blaster Port. Conectar el otro extremo del cable USB a un puerto cualquiera del PC.

El PC reconocerá que ha encontrado un nuevo hardware, el cual deberá ser instalado a través de su correspondiente controlador. Automáticamente, el PC lanza el asistente para la instalación del nuevo hardware obteniendo la ventana mostrada en la Figura 4 en pantalla.

Figura 4. Nuevo hardware encontrado

Dado que el controlador a instalar no se aloja en ninguna ubicación Web, seleccionar la tercera opción 'No, not this time', y presionar el botón 'Next'. De esta manera, aparece la ventana mostrada en la Figura 5.

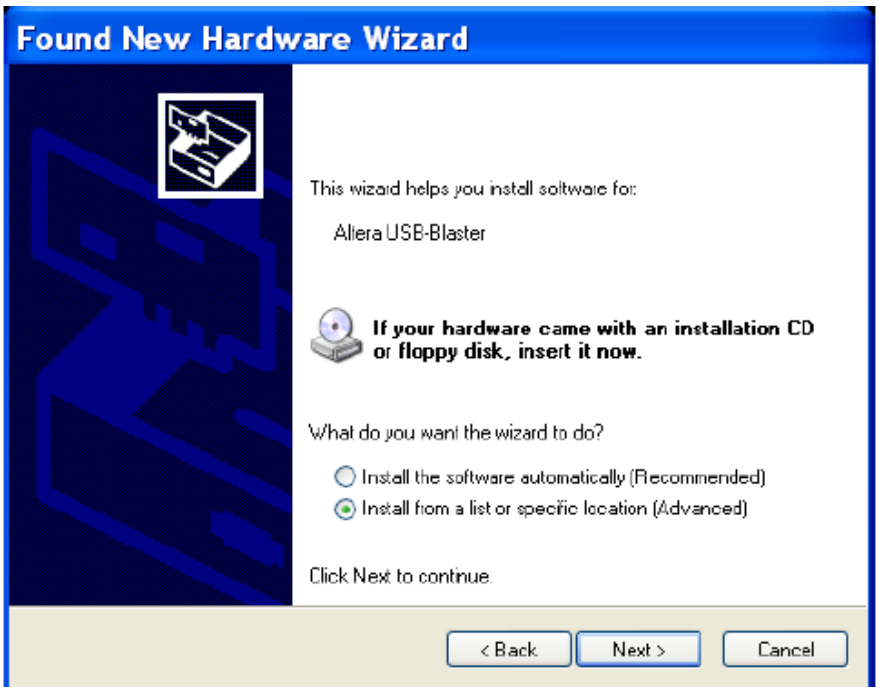


Figura 5. Buscar la localización del controlador

El driver a instalar se encuentra en la propia carpeta de instalación de Quartus II, alojada en la unidad de disco duro del PC. Para introducir la ruta del driver, seleccionar la segunda opción 'Install from a list or specific location' y presionar el botón 'Next', apareciendo la siguiente ventana:

Figura 6. Especificar la dirección del driver

Para buscar la localización del driver, pulsar sobre el botón 'Browse' y seguir la siguiente cadena: '*altera\quartus\drivers\usb−blaster'*. Seguidamente pulsar el botón 'Next' y obviar la siguiente pantalla en la que se indica la compatibilidad del controlador con 'Windows' ya que no es necesario realizar ningún test a dicho controlador.

Por último aparece la ventana mostrada en la Figura 7, la cual indica que el proceso de instalación de la placa en el PC ha concluido satisfactoriamente.

Figura 7. Confirmación de la instalación.

5.2.2.- DESCRIPCIÓN DE LA TARJETA EDUCACIONAL DE2

La tarjeta educacional DE2, al igual que la tarjeta UP2, está diseñada con la intención de ser la base para el aprendizaje sobre dispositivos lógicos programables. Para ello, dicha tarjeta cuenta con una serie de componentes que posibilitan la realización de una gran variedad de ejercicios.

El elemento principal de la placa es el dispositivo 'Altera Cyclone® II 2C35 FPGA', al cual se conectan el resto de componentes importantes de la placa.

Para el caso de experimentos sencillos, la placa DE2 cuenta con el suficiente número de interruptores (switches), diodos LED y displays de siete segmentos. Para experimentos más avanzados, existen chips de memoria SRAM, SDRAM y Flash, así como un display de 16x2 caracteres. Además, es posible la realización de experimentos que involucren señales de sonido y vídeo, para los cuales la placa lleva incorporados conectores estándar tales como '24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks' y 'VGA DAC (10-bit high-speed triple DACs) with VGA-out connector'. Por último, para diseños en grandes proyectos, es posible utilizar las conexiones USB y Ethernet, así como utilizar una tarjeta externa de memoria o conectar otras placas de diseño a través de sus conectores de expansión.

La Figura 8 muestra una imagen de la tarjeta educacional DE2.

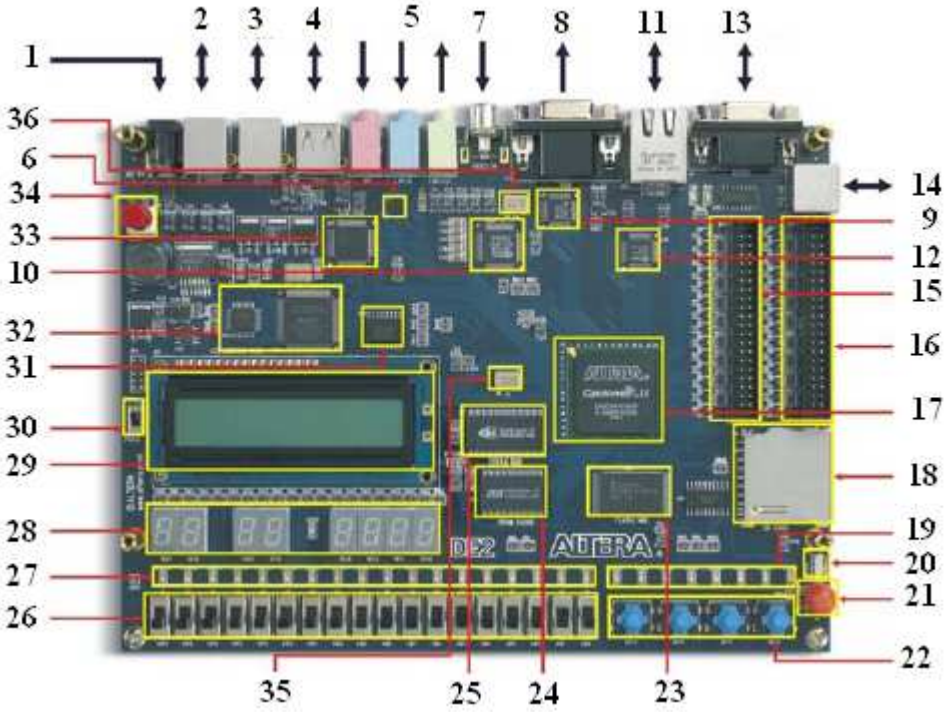


Figura 8. Tarjeta educacional DE2
(Imagen original extraída de [9])

Observando la figura anterior, los componentes que componen la placa son los siguientes:

1.- Entrada de alimentación, 9V DC.
2.- Puerto USB Blaster.
3.- Puerto USB Device.
4.- Puerto USB Host.
5.- Entrada de línea, salida de línea y entrada de micrófono.
6.- CODEC de audio de 24 bits.
7.- Entrada de video.
8.- Conector de salida VGA.
9.- VGA 10-bit DAC.
10.- Decodificador de TV (NTSC/PAL).
11.- Puerto Ethernet 10/100.
12.- Controlador Ethernet 10/100.
13.- Conexión RS-232.
14.- Conector PS2 de ratón ó teclado.
15.- Conector de expansión (JP2) de 40 pines.
16.- Conector de expansión (JP1) de 40 pines.
17.- Dispositivo Altera Cyclone® II 2C35 FPGA.
18.- Socket para memoria externa SD.
19.- 8 LEDs color verde.
20.- Transceptor IrDA.
21.- SMA external Clock.
22.- 4 pulsadores.
23.- Flash 4-Mbyte.
24.- Memoria SRAM 512-Kbyte.
25.- Memoria SDRAM 8-Mbyte.

26.- 18 interruptores.
27.- 18 LEDs color rojo.
28.- Displays de 7 segmentos.
29.- Módulo LCD 16x2.
30.- Interruptor RUN/PROG para selección de modos de programación JTAG/AS.
31.- Elemento de configuración de dispositivos serie Altera - EPCS16.
32.- Controlador Altera USB Blaster.
33.- Controlador USB Host/Slave.
34.- Interruptor Power ON/OFF.
35.- Oscilador de 50 MHz.
36.- Oscilador de 27 MHz.

- Diagrama de bloques:

La Figura 9 muestra el diagrama de bloques de la placa. En dicho diagrama se ve que todas las comunicaciones entre elementos de la placa son hechas a través del dispositivo principal Cyclone II FPGA, por lo que esta estructura dota al programador de la máxima flexibilidad, teniendo la posibilidad de implementar cualquier tipo de diseño.

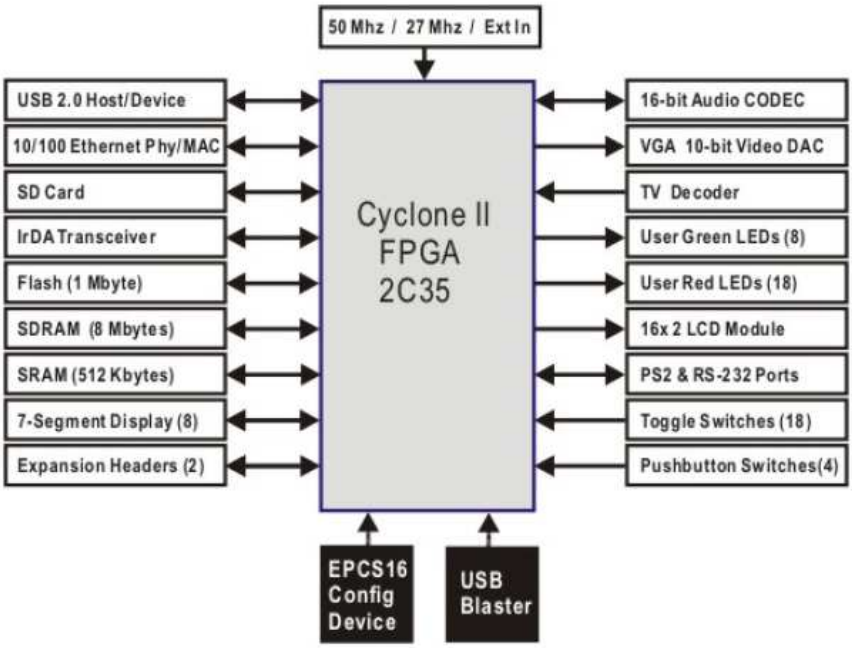


Figura 9. Diagrama de bloques de la placa DE2
(Imagen extraída de [9])

A continuación se da una descripción detallada de la familia de dispositivos Cyclone II FPGA, a la cual pertenece el elemento principal de la tarjeta educacional DE2.

5.2.3.- DISPOSITIVOS CYCLONE II FPGA

La familia de dispositivos programables integrados Cyclone II pertenece a la categoría de las FPGA. Los dispositivos que están bajo la denominación Cyclone® FPGA están construidos en base a una política de bajo consumo energético. Con memoria integrada, interfaces para memoria externa y circuitos de gestión de reloj, estos dispositivos son una solución óptima para aplicaciones de alto nivel a bajos costes.

De esta manera, los dispositivos de la familia Cyclone proporcionan una alternativa de bajo coste para aplicaciones actualmente basadas en ASIC's. De cara al diseñador, se aprecian diferentes novedades, tales como el aumento de la complejidad de los diseños, inclusión de nuevos estándares emergentes y reducción de ciclos de diseño.

La arquitectura de la familia Cyclone está basada en elementos lógicos (LE's) alineados verticalmente, bloques de memoria integrados y dispositivos PLL (phase-locked loops), que están rodeados de elementos de entrada y salida (IOE's).

Un sistema de interconexión altamente eficiente junto con un sistema de reloj para la sincronización proporciona la conectividad entre cada una de las estructuras, permitiendo a las señales de reloj y datos recorrer todo el dispositivo. La Figura 10 muestra el esquema básico de la arquitectura de un dispositivo de la familia Cyclone.

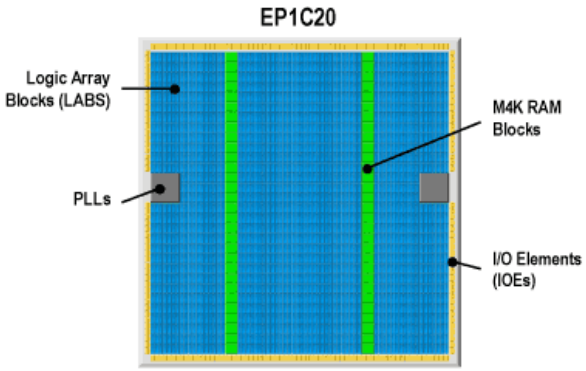


Figura 10. Estructura básica dispositivo Cyclone
(Imagen extraída de [8])

Dentro del área reservada para elementos de entrada y salida, se establecen bancos de señales (I/O) alrededor del dispositivo, ofreciendo importantes posibilidades, mientras se reduce la zona de consumo de energía. Estos elementos incluyen soporte para estándares tales como SSTL-2, SSTL-3 y LVDS I/O. Cada elemento de entrada o salida contiene tres registros que permiten la implementación de aplicaciones DDR (double data rate) además de circuitos asociados para otras posibles aplicaciones que el programador desee implementar.

Cada uno de estos bancos está equipado con un circuito dedicado que hace la función de interfaz para un bloque de memoria externa. Este circuito simplifica la transferencia de datos con los dispositivos de memoria externa, incluyendo dispositivos DDR-SDRAM y FCRAM.

Siguiendo los pasos de la primera generación de dispositivos Cyclone, la compañía Altera lanza la familia de dispositivos Cyclone® II FPGA como una extensión de la primera, igualmente basada en una política de bajo consumo energético. Dicha extensión conlleva una densidad de elementos lógicos (LE's) dentro de un rango entre 4608 y 68.416 LE's, un máximo de 622 pines utilizables por el usuario y una cantidad de memoria integrada de 1,1 Mbits. Asimismo, proporciona una señal global de reloj y un máximo de cuatro dispositivos PLL. La señal global de reloj está constituida por un máximo de 16 líneas que conducen esta señal a través de todos los puntos del dispositivo.

Las FPGA's Cyclone II son fabricadas en obleas de 300 mm, incorporando el material dieléctrico 'TSMC's 90 nm low-k'. Este proceso de fabricación asegura una disponibilidad rápida y de bajo coste. De esta manera, los dispositivos Cyclone II son capaces de soportar complejos sistemas digitales en un único chip, a un coste que rivaliza con el de un circuito integrado diseñado para una aplicación específica (ASIC).

El bajo coste, junto con las optimizadas funciones que ofrecen los dispositivos de la familia Cyclone II, hace que éstos se conviertan en una solución ideal en los campos de la automoción, consumo, comunicaciones, procesado de vídeo, aplicaciones de prueba y medición, además de otras soluciones de final de mercado.

Por otra parte, los dispositivos Cyclone II son capaces de trabajar conjuntamente con el procesador integrado de carácter general Nios® II, el cual permite implementar soluciones integradas de procesamiento realizadas de costumbre por el sistema.

5.2.4.- CONFIGURACIÓN DEL DISPOSITIVO CYCLONE II FPGA.

El procedimiento completo para realizar la descarga de un determinado diseño en el dispositivo Cyclone II EP2C35 se muestra en el capítulo 'Manual de Quartus II'.

La placa DE2 contiene un chip de memoria EEPROM que almacena los datos de configuración del dispositivo Cyclone II EP2C35. Estos datos de configuración son automáticamente cargados en el dispositivo al mismo tiempo en el que se aplica alimentación a la placa. Mediante el uso del software Quartus II es posible realizar la reprogramación de la FPGA en cualquier momento, así como cambiar los datos de configuración almacenados en el chip de memoria EEPROM.

La placa DE2 de Altera puede ser configurada de dos diferentes maneras: modo JTAG y modo AS, siendo el primero de ellos el que va a ser utilizado en las prácticas. Los datos de configuración son enviados desde el PC, con el software Quartus II activo, a través de un puerto USB cualquiera, hasta la placa a través del puerto USB situado más a la izquierda de la misma, denominado USB Blaster, junto a la entrada de alimentación (mirar Figura 8). Es fundamental, para realizar esta conexión tener el controlador USB-Blaster instalado.

En el modo JTAG (Joint Test Action Group), los datos de configuración son enviados directamente al dispositivo FPGA. Al configurar el dispositivo de este modo, éste retiene la configuración indicada mientras no se corte la alimentación. La configuración se pierde cuando la placa deja de ser alimentada. El segundo modo de configuración es el denominado Active Serial (AS). En este caso, los datos de configuración son guardados en un bloque de memoria flash, por lo que dichos datos no se pierden en caso de desconectar la alimentación y la FPGA no ha de ser reconfigurada.

A continuación se describen los pasos a seguir para la programación con cada uno de los modos.

- Configuración en modo JTAG:

La Figura 11 ilustra el esquema de configuración JTAG. Los pasos a seguir son los siguientes:

1.- Asegurarse de que se está aplicando alimentación a la tarjeta DE2.

2.- Realizar la conexión USB entre el PC y el puerto USB Blaster de la tarjeta.

3.- Configurar el circuito JTAG de programación seleccionando la posición RUN en el interruptor RUN/PROG situado en el margen izquierdo de la placa.

4.- Por último, mediante la herramienta de programación de Quartus II, seleccionar el archivo de configuración, el cual tendrá una extensión *.sof*, y realizar la descarga del mismo.

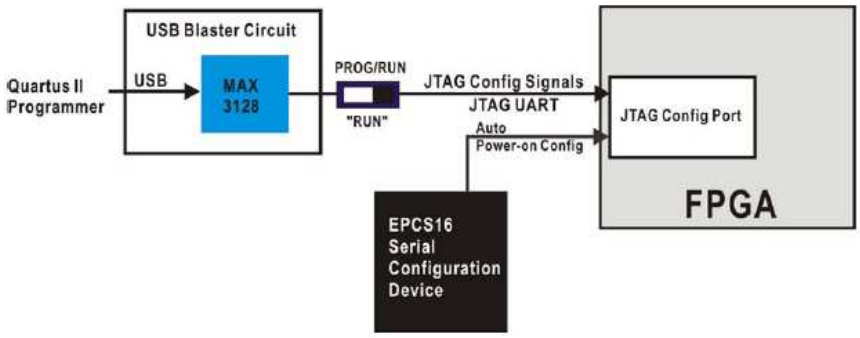


Figura 11. Esquema de configuración JTAG
(Imagen extraída de [9])

- Configuración en modo AS:

En la Figura 12 se observa el esquema resultante para la configuración en modo AS. Los pasos a seguir son los siguientes:

1.- Asegurarse de que se está aplicando alimentación a la tarjeta.

2.- Realizar la conexión USB entre el PC y el puerto USB Blaster de la placa.

3.- Configurar el circuito de programación en modo AS seleccionando la posición PROG en el interruptor RUN/PROG situado en el margen izquierdo de la placa.

4.- Mediante el programador del software Quartus II, seleccionar el archivo de configuración correspondiente, el cual contará con extensión *.pof*, y realizar la descarga del mismo.

5.- Una vez finalizada la operación de programación, volver a situar el interruptor RUN/PROG en la posición RUN y realizar un reset de la placa, apagándola y volviéndola a encender. De esta manera, los nuevos datos de configuración almacenados en el dispositivo EPCS16, serán cargados en la FPGA.

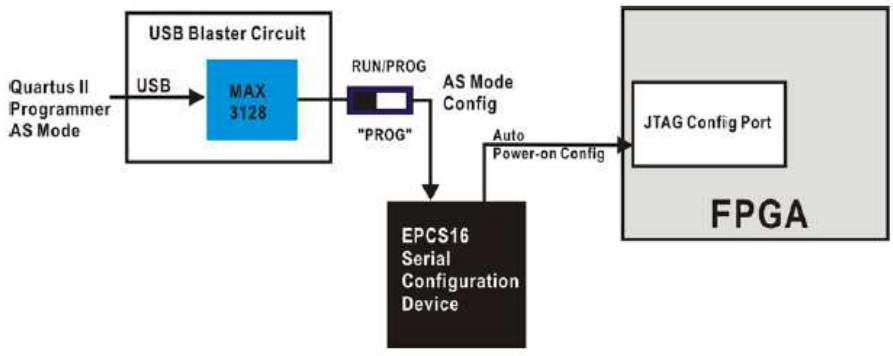


Figura 12. Esquema de configuración AS
(Imagen extraída de [9])

5.3.- FLEX10K70 – CYCLONE II 2C35

La Tabla 1 muestra las características de los dispositivos EPF10K70 y Cyclone II EP2C35, correspondientes a las tarjetas educacionales UP2 y DE2 respectivamente. Además, también se incluyen las características del dispositivo que lleva integrado la placa DE1, Cyclone II EP2C20, que es un dispositivo intermedio entre ambos, con el objetivo de ver las diferencias existentes entre ellos y su consecuente evolución.

Características	EPF10K70	Cyclone II EP2C20	Cyclone II EP2C35
Puertas típicas (lógica y RAM)	70.000	-	-
Puertas máximas del sistema	118.000	-	-
Elementos lógicos (LE's)	3.744	18.752	33.216
Bloques de matrices lógicas (LAB's)	468	1.172	2.076
Bloques M2K RAM (bloques de 2 Kbits)	9	-	-
Bloques M4K RAM (bloques de 4 Kbits)	-	52	105
Bits totales de RAM	18.432	239.616	483.840
Multiplicadores integrados (18x18)	-	26	35
PLLs	-	4	4
Pines de entrada/salida para el usuario	358	315	475

Tabla 1. Comparación de características

Observando la tabla se comprueba la notable evolución de las características entre dispositivos, lo que permite al usuario un amplio abanico de posibilidades, desde diseños sencillos hasta diseños de mayor complejidad que requieran mayores recursos.

5.4.- GLOSARIO

- **ByteBlaster:** Cable de descarga paralelo que permite al usuario programar y configurar los dispositivos de la placa UP2.

- **Chip:** Grupo de funciones lógicas definidas como una unidad. La asignación de un chip a un determinado dispositivo puede ser realizada por el usuario o por la herramienta de compilación.

- **CODEC:** es la abreviatura de codificador-decodificador. Los códecs pueden codificar el flujo de datos de una señal y recuperarlo del mismo modo para su posterior reproducción.

- **DSP:** Un procesador digital de señales es un sistema basado en un procesador o microprocesador que posee un juego de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad.

- **EEPROM:** Acrónimo de ***E****lectrically* ***E****rasable* ***P****rogrammable* ***R****ead-****O****nly* ***M****emory*. Se trata de un tipo de memoria reprogramable en la cual los contenidos se pueden borrar sometiendo al dispositivo a las señales eléctricas apropiadas para ello.

- **Elemento Lógico (LE):** Es el bloque básico en la estructura de cualquier dispositivo.

- **Ethernet:** es un estándar de redes de ordenadores de área local.

- **FPGA:** Acrónimo inglés ***F****ield* ***P****rogrammable* ***G****ate* ***A****rray*. Es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar.

- **Jumper:** Término asignado al elemento conductor usado para conectar dos terminales para cerrar un circuito eléctrico o electrónico.

- **LCD:** acrónimo inglés de **L**iquid **C**rystal **D**isplay. Pantalla de cristal líquido.

- **NTSC:** sistema de codificación para señales de televisión utilizado principalmente en Estados Unidos.

- **PAL:** sistema de codificación para señales de televisión utilizado principalmente en Europa.

- **PLD:** Acrónimo inglés ***P****rogrammable* ***L****ogic* ***D****evice*. Es un dispositivo electrónico programable, formado por múltiples bloques lógicos.

- **PLL:** Acrónimo inglés ***P****hase-****L****ocked* ***L****oops*. Se trata de un sistema realimentado, en el que las magnitudes realimentadas son la frecuencia y la fase.

- **Programación AS:** (**A**ctive **S**erial), modo de programación donde los datos de configuración son guardados en un bloque de memoria flash, por lo que dichos datos no se pierden en caso de desconectar la alimentación

- **Programación JTAG:** (**J**oint **T**est **A**ction **G**roup), los datos de configuración son enviados directamente al dispositivo FPGA. Al configurar el dispositivo de este modo, éste retiene la configuración indicada mientras no se corte la alimentación.

- **RAM: R**andom-**a**cces **m**emory o memoria de acceso aleatorio. Es el área de trabajo para la mayor parte del software de un PC.

- **Tecnología CMOS**: del inglés **C**omplementary **M**etal **O**xide **S**emiconductor, es una de las familias lógicas empleadas en la fabricación de circuitos integrados.

- **USB Blaster:** Circuito contenido en la placa DE2 para la programación de la misma a través de un cable USB conectado al PC.

- **VGA:** acrónimo de **V**ideo **G**raphics **A**rray. Se trata de un sistema gráfico de pantallas para PCs desarrollado por IBM.

CAPÍTULO 6.

MANUAL DEL SOFTWARE 'QUARTUS II'

Antes de comenzar con el manual del software Quartus II, se muestran la serie de pasos a seguir para proceder a la descarga de la versión 'Web Edition' del mencionado software.

DESCARGA DEL PROGRAMA QUARTUS II WEB EDITION DE ALTERA

Acudir a la página web de Altera Corporation: http://www.altera.com.

Desde la página principal, acudir a la zona de descarga (esquina superior derecha, en el botón 'Download Center').

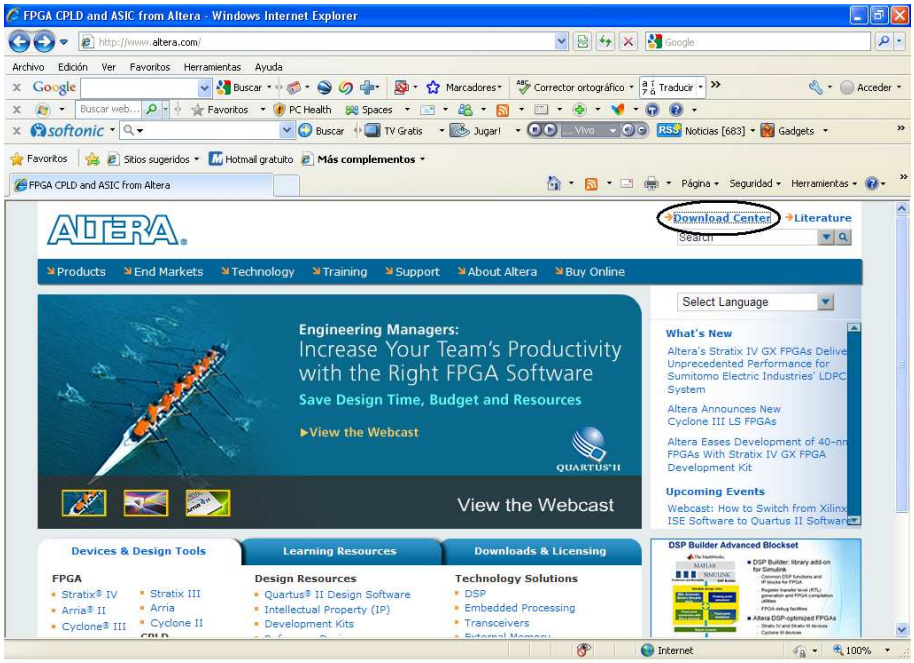


Figura 1. Página principal de Altera Corporation

Acceder al enlace que permite la descarga del software Quartus II Web Edition, tal y como indica la figura 2.

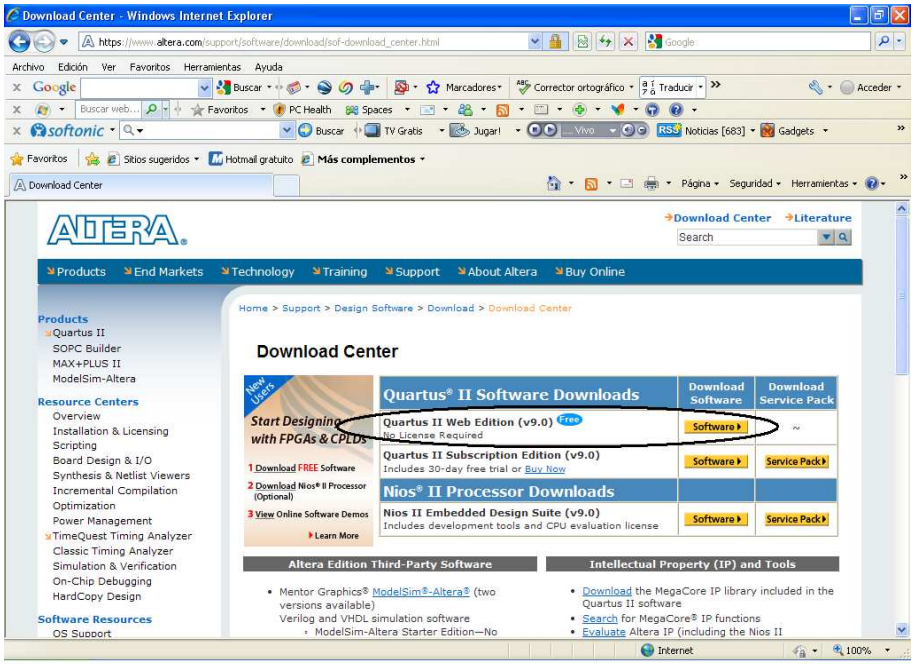


Figura 2. Centro de descarga de Altera

Entre las opciones que aparecen en la tabla que se muestra en la Figura 3, seleccionar la opción 'Quartus II Web Edition Software Service Pack 2'.

Tener en cuenta que el tamaño de la descarga es de 1,31 GB.

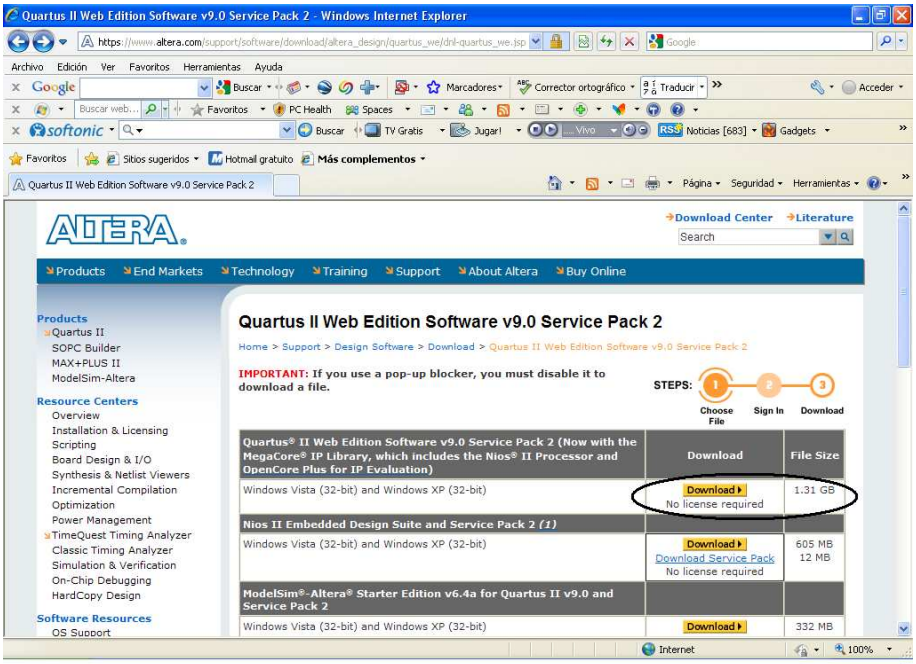


Figura 3. Descarga de Quartus II Web Edition Software

Es necesario registrarse como nuevo usuario para poder descargar el programa (Create Your Altera.com Account). Se debe introducir una cuenta de usuario que será empleada por Altera para el envío de la confirmación de nuevo usuario.

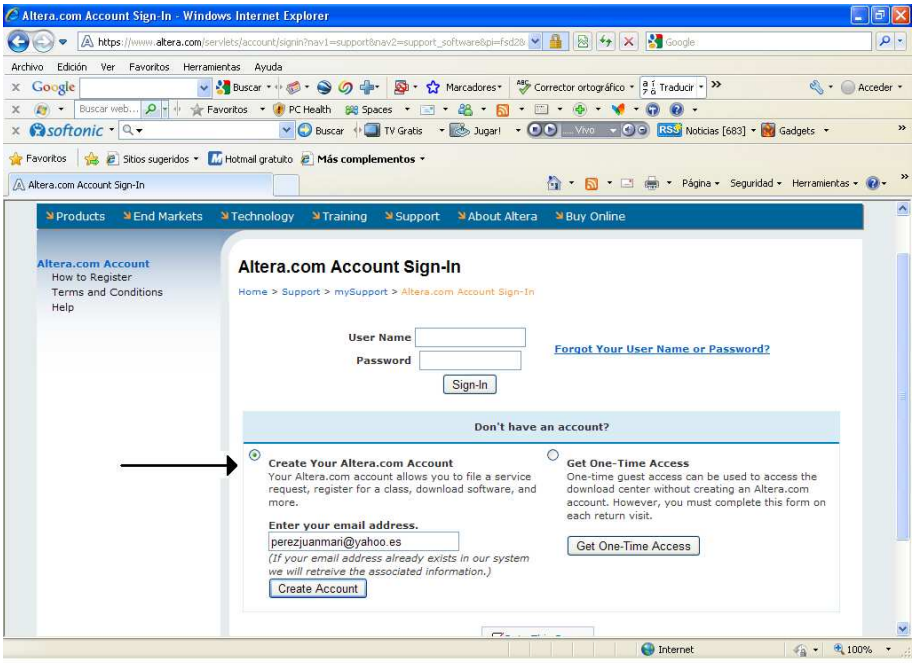


Figura 4. Identificación de Usuario

Introducir los datos de los campos obligatorios (los marcados por asterisco).

- Nota: no introducir acentos ni 'ñ' ya que la página utiliza diccionario inglés.

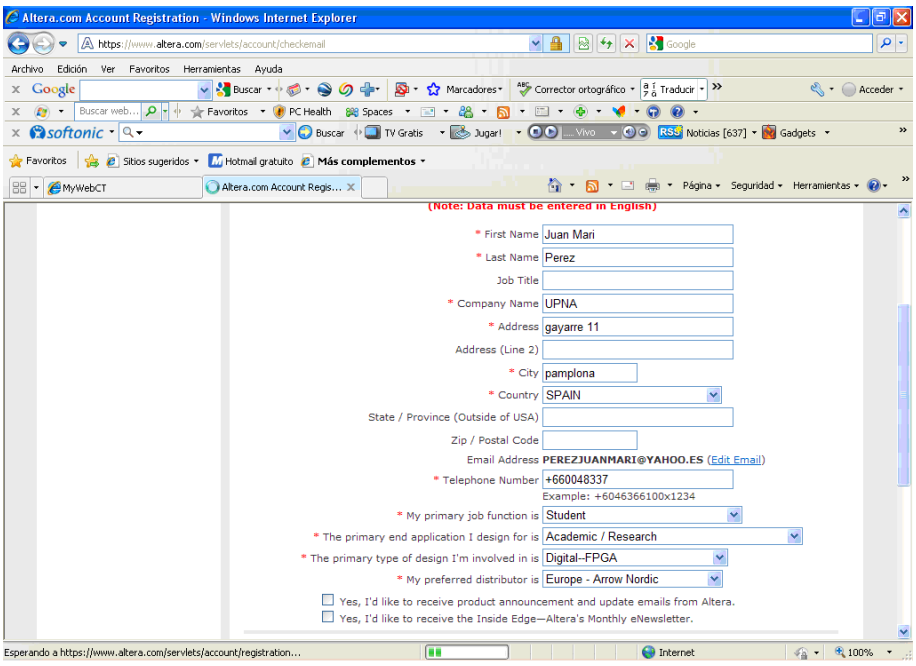


Figura 5. Introducción de Datos de Usuario

Después de registrarse da comienzo la descarga. Es necesario que los 'Pop-up' de Windows estén desactivados para que la descarga se inicie automáticamente.

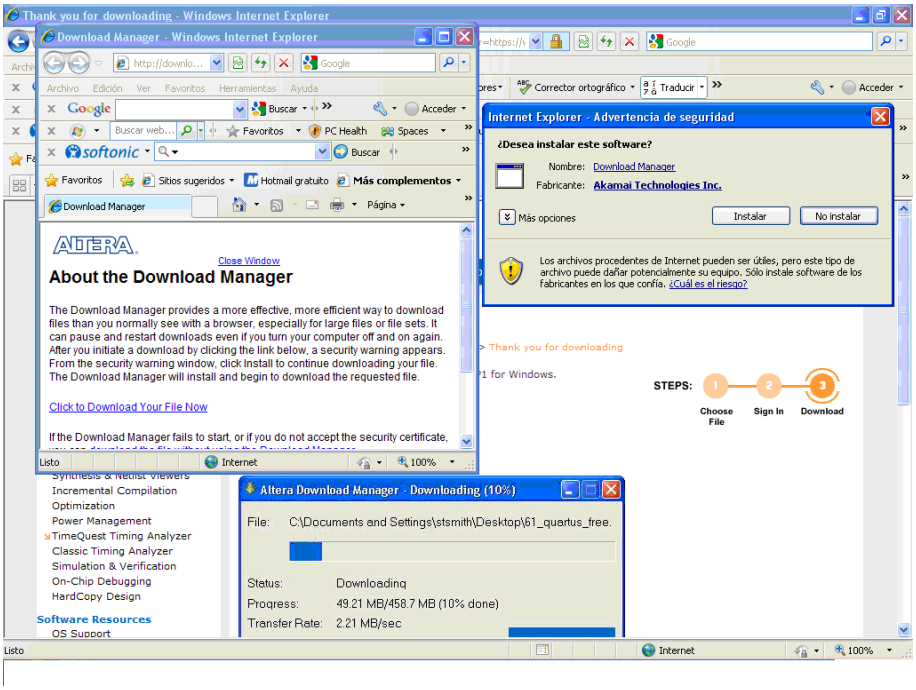


Figura 6. Descarga automática de Quartus II

En caso de que los 'Pop-up' de Windows estén activados, aparecen los mensajes de la Figura 6. En este caso, se debe presionar sobre 'Click to Download Your File Now' para comenzar la descarga.

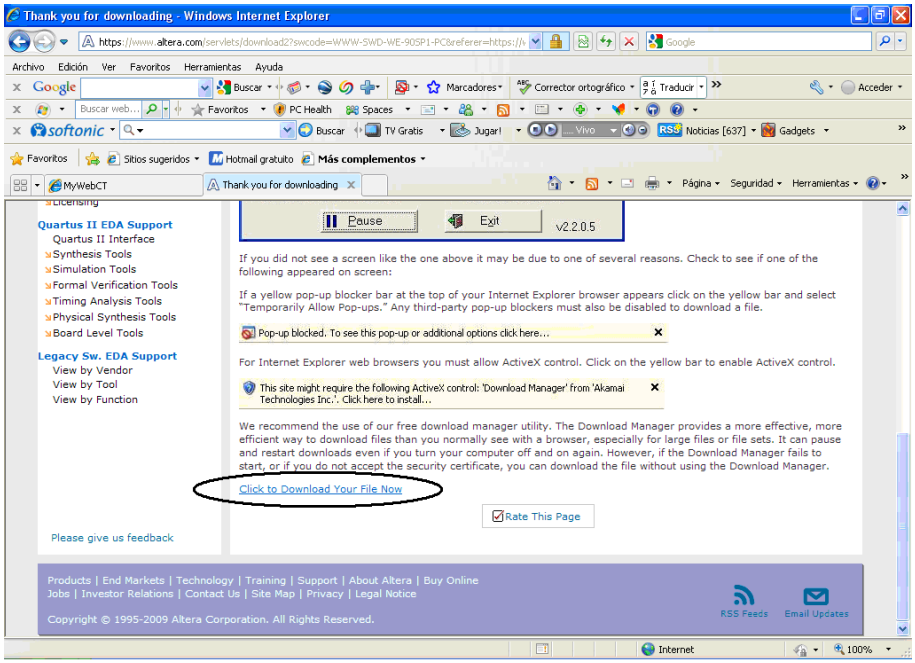


Figura 7. Descarga manual de Quartus II

Hay que tener en cuenta que el tamaño de la descarga es de 1,31 GB, por tanto, es necesario disponer de una conexión a Internet que soporte un tiempo razonable la descarga.

Es recomendable, antes de la instalación de cualquier programa, leer las instrucciones que el diseñador del software facilita, tanto para ahorrar tiempo en la instalación como para asegurarse de que el ordenador en el que se pretende ejecutar el software es compatible con el mismo. A continuación se incluye el contenido de la guía de instalación tal y como aparece en la propia página de Altera.

Install Program File

Perform the following steps to install the Quartus II Web Edition software:

1. *Create a directory on your hard disk to store the Quartus II Web Edition software and license file. The default location is c:\quartus2we.*
2. *Run the quartusii_< version information >_web_edition_single.exe program by double-clicking on the file name in Windows Explorer. Replace* < version information > *with the version number of the software. For example, the download file for Quartus II Web Edition software version 4.0 is quartusii_40_web_edition_single.exe.*
3. *The installation program guides you through the installation process. Specify the directory you created in step 1 as the installation directory. Altera recommends that you install the Quartus II Web Edition software in a different directory from any other version of the Quartus II software.*
4. *When installation is complete, you can read the Quartus II software readme.txt file by clicking on it in the Altera option of the Start menu.*
5. *Continue the installation by setting up your license file.*

Además, se puede consultar más información en el archivo **readme.txt** creado en la instalación del programa.

6.-MANUAL DEL SOFTWARE 'QUARTUS II'

6.1.- INTRODUCCIÓN

En este capítulo se realiza un manual introductorio al software QUARTUS II® de la compañía Altera®. El objetivo que se pretende alcanzar con la realización de dicho manual para el alumno es el de dar a conocer tanto el material como la forma de trabajar con dispositivos lógicos programables (PLD's). Asimismo, sirve de ayuda en la realización de las prácticas del curso a la vez que se adquiere un conocimiento básico sobre la utilización de este tipo de dispositivos para que, en un futuro, sirva para desarrollar aplicaciones más complejas.

Este manual es un elemento de apoyo en el que se recoge una determinada manera de trabajar, si bien no existe un único camino para realizar una misma operación.

Asimismo, el software Quartus II tiene diversas partes que no han sido tratadas en el presente manual por considerarse que se escapan de los objetivos que se pretenden obtener con la realización de las prácticas de los respectivos cursos. Por ello, para más información sobre el tema, se puede consultar la página oficial de Altera, en la que se pueden encontrar enlaces de interés sobre la diversa literatura de Quartus II. De entre la literatura existente cabe destacar lo siguiente:

- Introduction to the Quartus II Software
- Quartus II Handbook Version 9.0.

Enlace: www.altera.com

Nota: Quartus II ofrece la posibilidad de trabajar con el entorno gráfico del software que se viene utilizando hasta ahora en el laboratorio, Max+Plus II.

Al ser éste un manual pensado para alumnos que no han tenido contacto con ninguno de los dos software mencionados anteriormente se recomienda la utilización del programa Quartus II con su propio entorno gráfico, si bien puede ser de utilidad para personas con experiencia en el entrono de Max+Plus II en cuanto a sencillez en alguna de las operaciones, incluir en la barra de herramientas principal el menú desplegable de Max+Plus II tal y como se indica más adelante en el presente manual.

6.2.- SOFTWARE QUARTUS II DE ALTERA

El software de diseño Quartus II de Altera ofrece un completo entorno de diseño que se adapta fácilmente a las necesidades de los respectivos diseños. Este software incluye diferentes soluciones para la todas las fases del diseño de FPGA's y CPLD's.

El ciclo de diseño utilizado por el programa se puede ver en la Figura 1. Básicamente, las etapas del diseño son: entrada de datos y definición del diseño, verificación del proyecto, procesado y simulación, programación del dispositivo. Todo ello será explicado más adelante con mayor detalle.

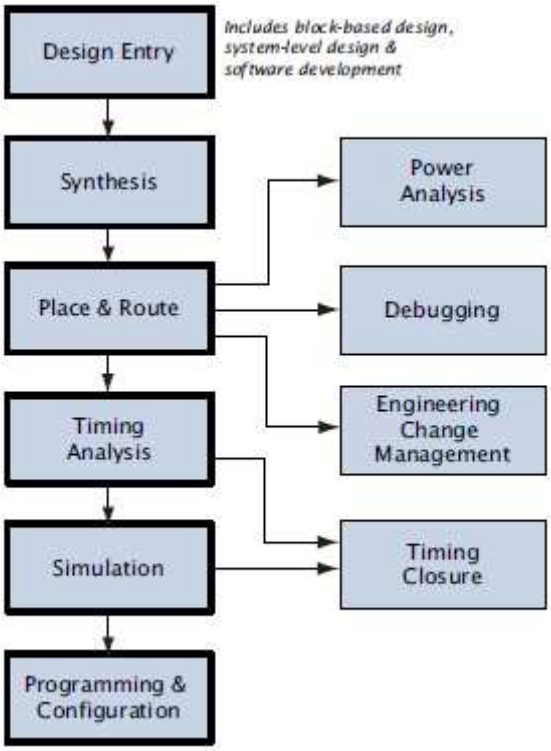


Figura 1. Ciclo de diseño
(Imagen extraída de [8])

Muchas características y órdenes son compartidas por las diferentes aplicaciones de Quartus II, por lo que el conocer una de ellas facilita el aprendizaje de las demás. De esta manera, se tendrán numerosas órdenes que se repetirán en cada una de las diferentes aplicaciones, como puede ser abrir proyectos, asignar los dispositivos de los proyectos, etc, así como diversas herramientas de diseño y tareas, como por ejemplo, la asignación de los diferentes pines. La Figura 2 muestra el entrono de trabajo de Quartus II.

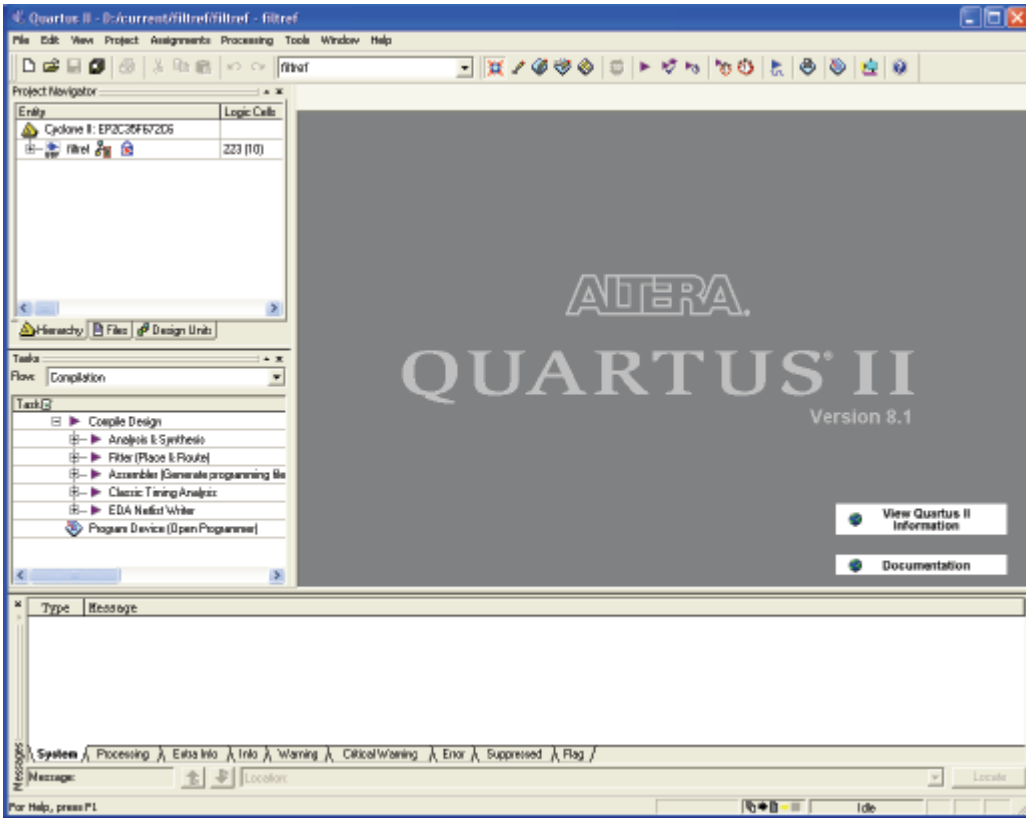


Figura 2: Entorno gráfico de Quartus II

Asimismo, Quartus II permite trabajar con diferentes aplicaciones al mismo tiempo. Por ejemplo, es posible abrir múltiples archivos de diseño y compartir información entre ellos, mientras que por otro lado se está compilando o simulando otro proyecto.

Quartus II trabaja con una estructura jerárquica que permite entender la organización del programa, tal y como se puede apreciar en la Figura 3. En esta presentación jerárquica se puede pinchar directamente en cada elemento, por lo que es posible moverse de un nivel a otro de manera sencilla. Cuando se abra un archivo de diseño, automáticamente se iniciará el editor de diseño apropiado.

Figura 3: Presentación jerárquica de un diseño

6.3.- GUÍA SIMPLIFICADA PARA REALIZAR UN DISEÑO

Los pasos a llevar a cabo en la realización de un proyecto de diseño de principio a fin, se pueden simplificar de la siguiente manera (un desarrollo más extenso se verá más adelante):

1.- Crear un nuevo proyecto (un archivo o una jerarquía de varios archivos) con alguna de las herramientas que contiene Quartus II.

2.- Especificar el nombre del proyecto. En caso de que dicho proyecto contenga varios archivos, definir el nombre del archivo más importante, es decir, el superior jerárquicamente.

3.- Seleccionar los archivos de diseño que se quieren incluir en el proyecto así como la familia de dispositivos que se quiere utilizar en la compilación. También es posible permitir al compilador seleccionar un dispositivo automáticamente.

4.- Una vez diseñado el proyecto, se procede a la compilación del mismo. Esta operación permite tanto detectar errores en el diseño, como crear el archivo de configuración a descargar en el dispositivo programable. .

5.- Si el proyecto se compila satisfactoriamente, es posible, de manera opcional, pero recomendable, realizar una simulación y un análisis temporal. Para realizar la simulación, primero se deben crear las formas de onda de las entradas en el Editor de Señales.

6.- Por último se realiza el volcado y programación del dispositivo lógico que se esté empleando para la realización del proyecto.

6.4.- ENTORNO GRÁFICO DE QUARTUS II

Tal y como se apunta en el apartado 'Introducción' del presente capítulo, el entorno de trabajo a utilizar va a ser el del software Quartus II, con la posibilidad de incluir en la barra de herramientas principal el menú desplegable de Max+Plus II. Esto puede ser útil para personas con experiencia en Max+Plus II debido a que desde este menú es posible abrir de forma más intuitiva las herramientas de diseño con las que vamos a trabajar en la realización de las prácticas, como son: Editor Gráfico (Graphic Editor), Editor de Símbolos (Symbol Editor), Editor de Texto (Text Editor) y Editor de Señales (Waveform Editor).

Para realizar esta operación se accede al menú desplegable 'Tools' de la barra de herramientas principal y se selecciona la opción 'Customize…'. Tras la realización de estos pasos aparece la ventana que se muestra en la Figura 4.

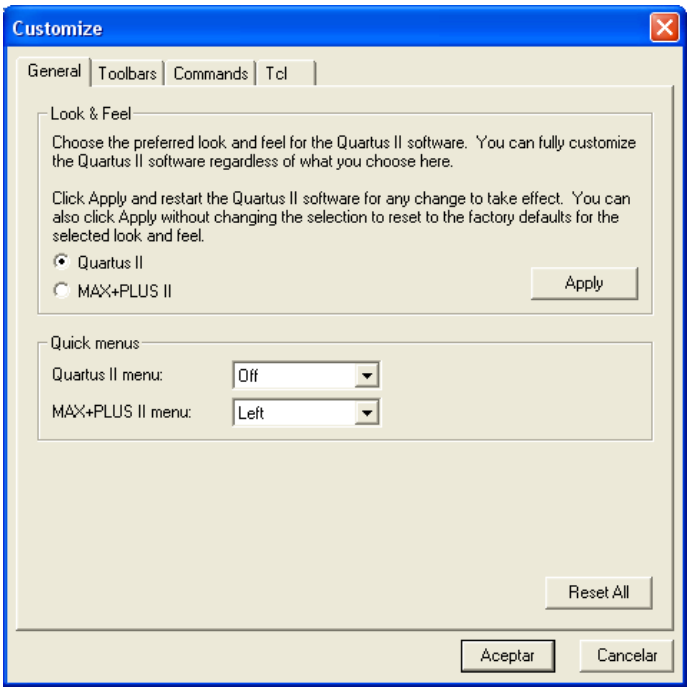


Figura 4: Inclusión del menú de Max+Plus II

Teniendo en cuenta las opciones que aparecen seleccionadas en la ventana de la Figura 4 obtendremos el resultado mostrado a continuación en la Figura 5:

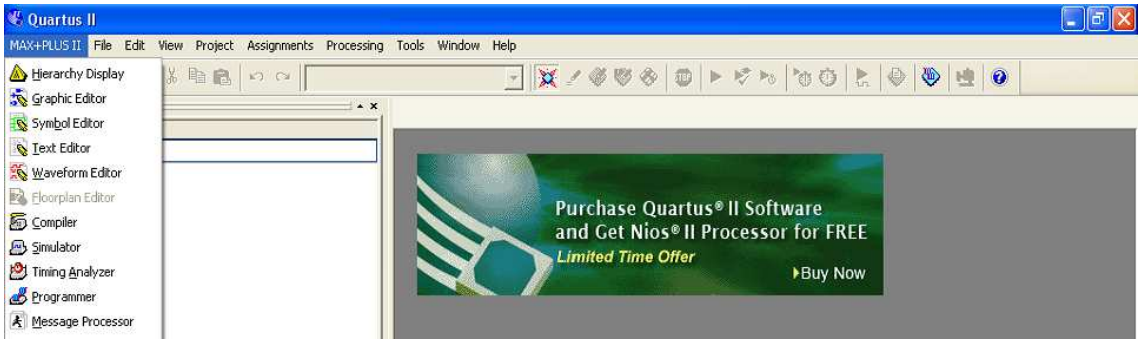


Figura 5: Entorno Quartus II + menú desplegable Max+Plus II

6.5.- DEFINICIÓN DEL PROYECTO

El primer paso a la hora de realizar un diseño con Quartus II consiste en la definición del proyecto en el que se va a operar. Un proyecto consiste en englobar una serie de archivos de diseño dentro de un subdirectorio dado.

A la hora de definir el proyecto, se debe incluir el nombre del archivo de diseño que ocupa un lugar más alto en la jerarquía de archivos, el cual podrá ser simulado y sintetizado.

Para definir el proyecto, es necesario acudir a la opción 'New Project Wizard' situada en el menú 'File', que es el asistente para la definición de proyectos. Este asistente ofrece una serie de ventanas que permiten realizar las siguientes operaciones:

- Seleccionar el subdirectorio de trabajo.

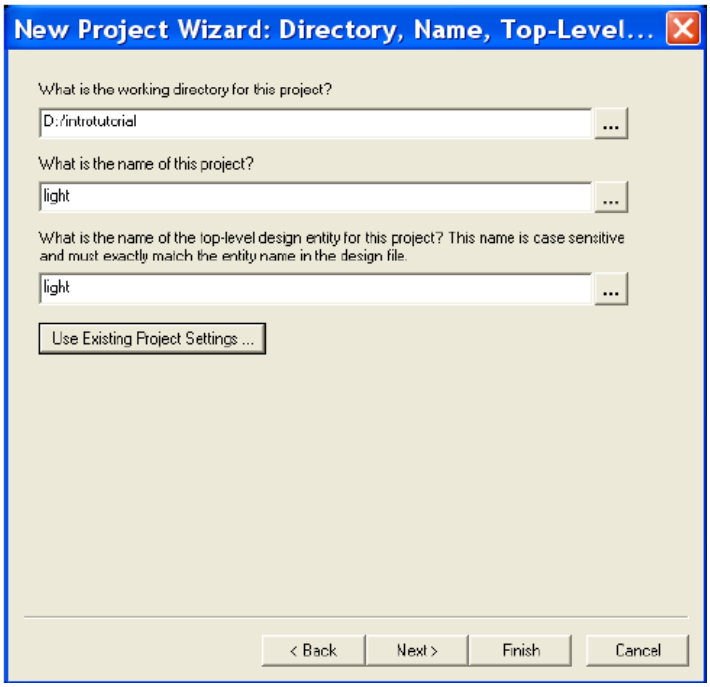


Figura 6. Subdirectorio de trabajo

- **Importante: Cada nuevo proyecto debe ser almacenado en su propio directorio de trabajo, de no ser así, se pueden producir conflictos a la hora de realizar la compilación (proceso explicado posteriormente en el presente manual) de cada uno de los diferentes diseños.**

- Incluir librerías y archivos de otros proyectos que sean necesarios.

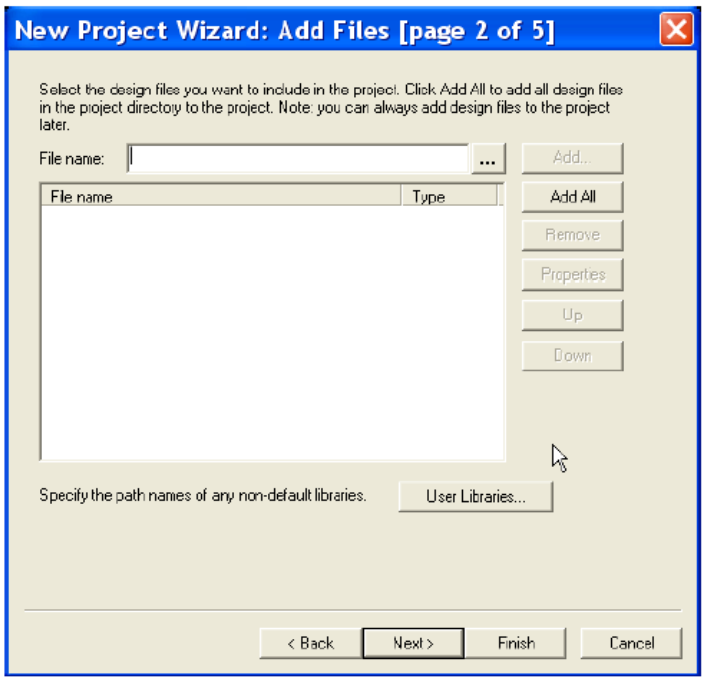


Figura 7. Inclusión de archivos de diseño específicos

- Seleccionar la familia de dispositivos, así como el dispositivo específico con el que se va a trabajar en el proyecto.

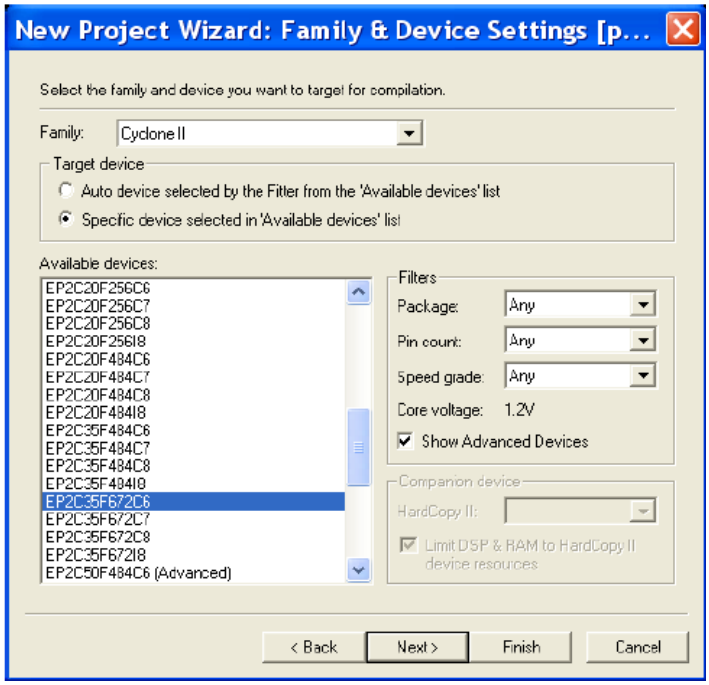


Figura 8. Selección de la familia y dispositivo de trabajo

- A través de la siguiente ventana, el usuario puede seleccionar otras herramientas auxiliares para poder ser usadas:

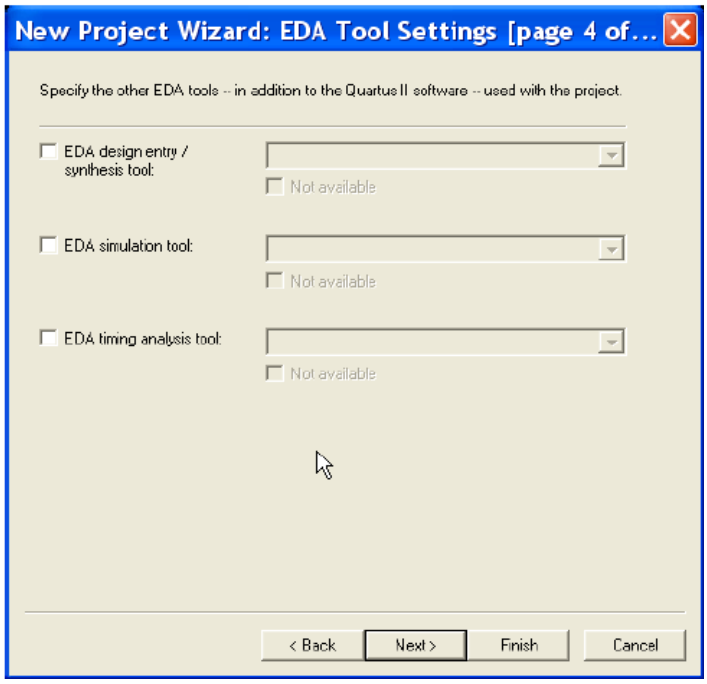


Figura 9. Herramientas auxiliares

- Como último paso, se presenta una ventana que contiene un sumario con todos los pasos anteriores

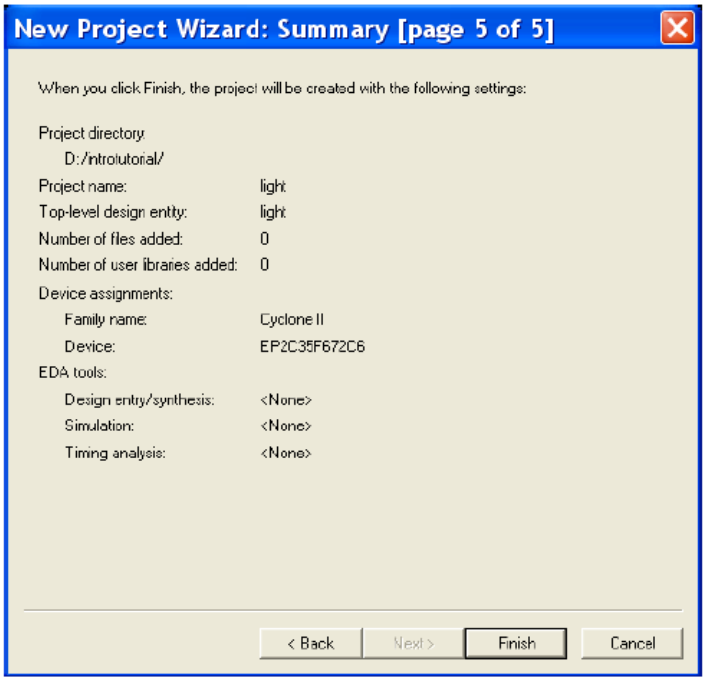


Figura 10. Sumario de la definición del proyecto

6.6.- EDITOR GRÁFICO (GRAPHIC EDITOR)

La herramienta Editor Gráfico permite la realización de diseños conectando gráficamente distintos bloques, donde cada cual realiza distintas funciones. Estos bloques se pueden dividir en tres grandes grupos:

- Funciones primitivas
- Megafunciones
- Otras funciones

En el grupo de funciones primitivas se encuentran las funciones básicas para construir un circuito lógico, como son las puertas lógicas (AND, OR, NAND, NOR, XOR, XNOR, NOT), las señales de alto y bajo nivel (Vcc, Gnd), elementos de entrada y salida (Inputs, Outputs), diferentes flip-flops, etc.

El apartado de Megafunciones cuenta con funciones más complejas, que el usuario puede configurar a su medida y que están completamente optimizadas. Se pueden encontrar contadores, multiplexores, decodificadores, registros móviles, y otras muchas más funciones, las cuales permiten modificar a voluntad casi cualquier parámetro, como por ejemplo escoger las entradas y salidas que deberá tener la función.

La sección definida como 'Otras funciones' cuenta con una carpeta denominada 'maxplus2' donde se encuentran diferentes elementos que ofrecen distintas funciones como son multiplexores, demultiplexores, codificadores, decodificadores, gran parte de la familia TTL (identificada con el número 74), etc.

Para comenzar a trabajar con el Editor Gráfico, tal y como hemos definido el entorno de trabajo, existen dos opciones. La primera de ellas, por ser la forma que por defecto emplea Quartus II, consiste en abrir un nuevo documento. Para ello, se pincha sobre el folio en blanco que aparece en la parte superior izquierda de la pantalla, como se puede apreciar en la Figura 11.

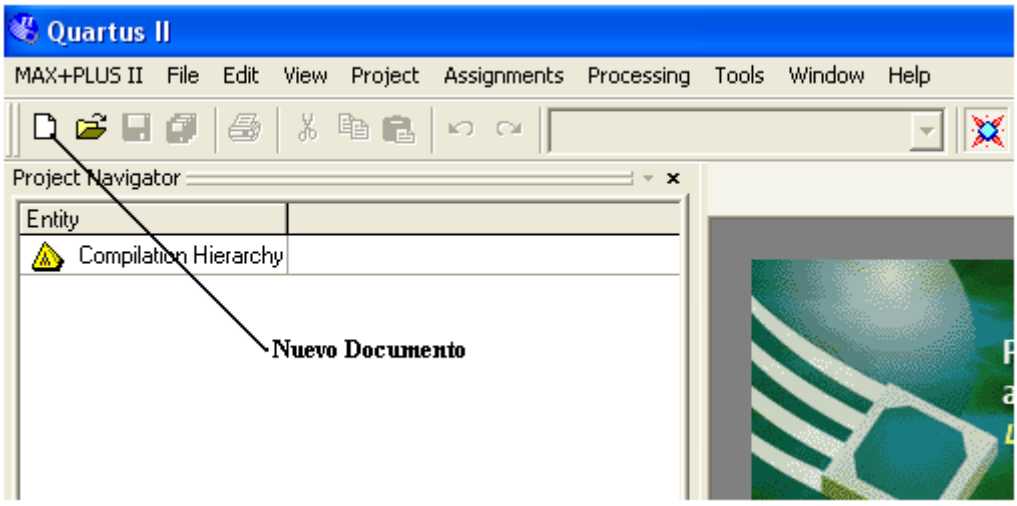


Figura 11: Abrir nuevo documento

Aparecerá entonces el menú que se muestra en la Figura 12. En dicho menú, se seleccionará la opción 'Block Diagram / Schematic File' para así ya estar en condiciones de comenzar a usar el Editor Gráfico.

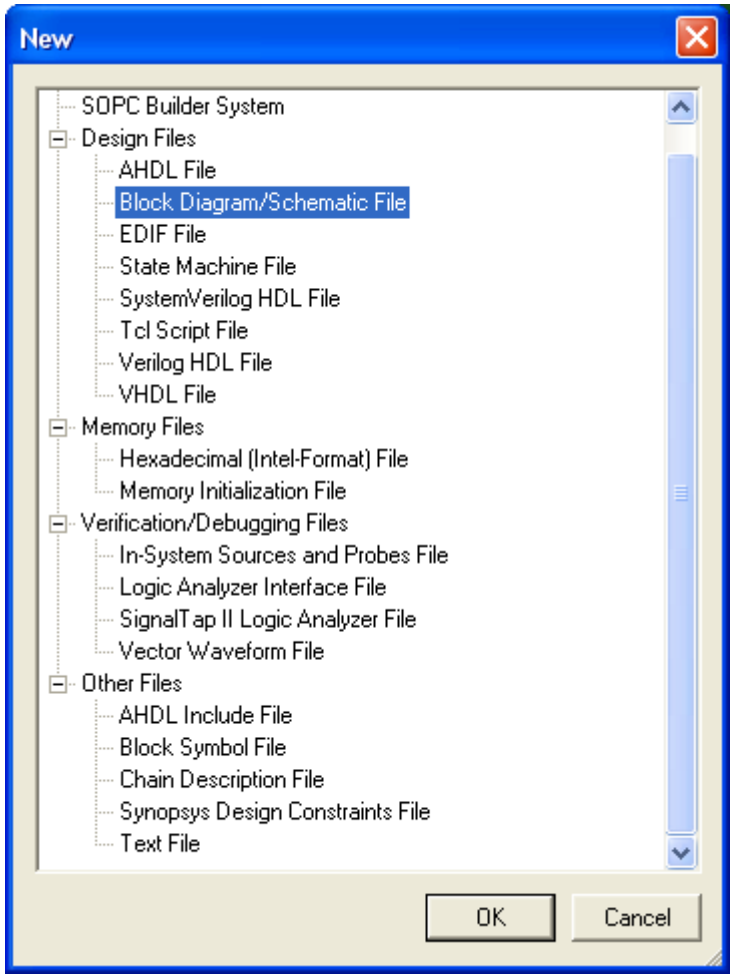


Figura 12: Selección del Editor Gráfico

La segunda de las opciones, recurriendo al menú desplegable de Max+Plus II, simplemente consiste en pinchar sobre la opción 'Graphic Editor' según se muestra en la Figura 13.

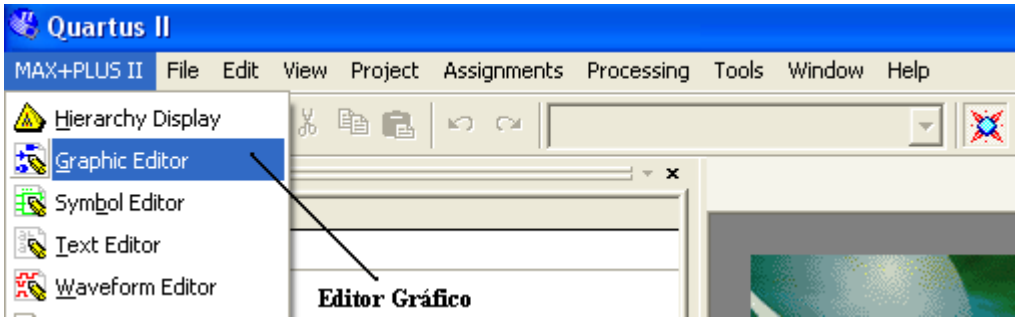


Figura 13: Selección del Editor Gráfico desde menú de Max+Plus II

La figura 14 muestra la ventana del Editor Gráfico. En esta figura se puede ver a la izquierda de la pantalla la barra de herramientas del Editor, donde sus elementos se detallan a continuación.

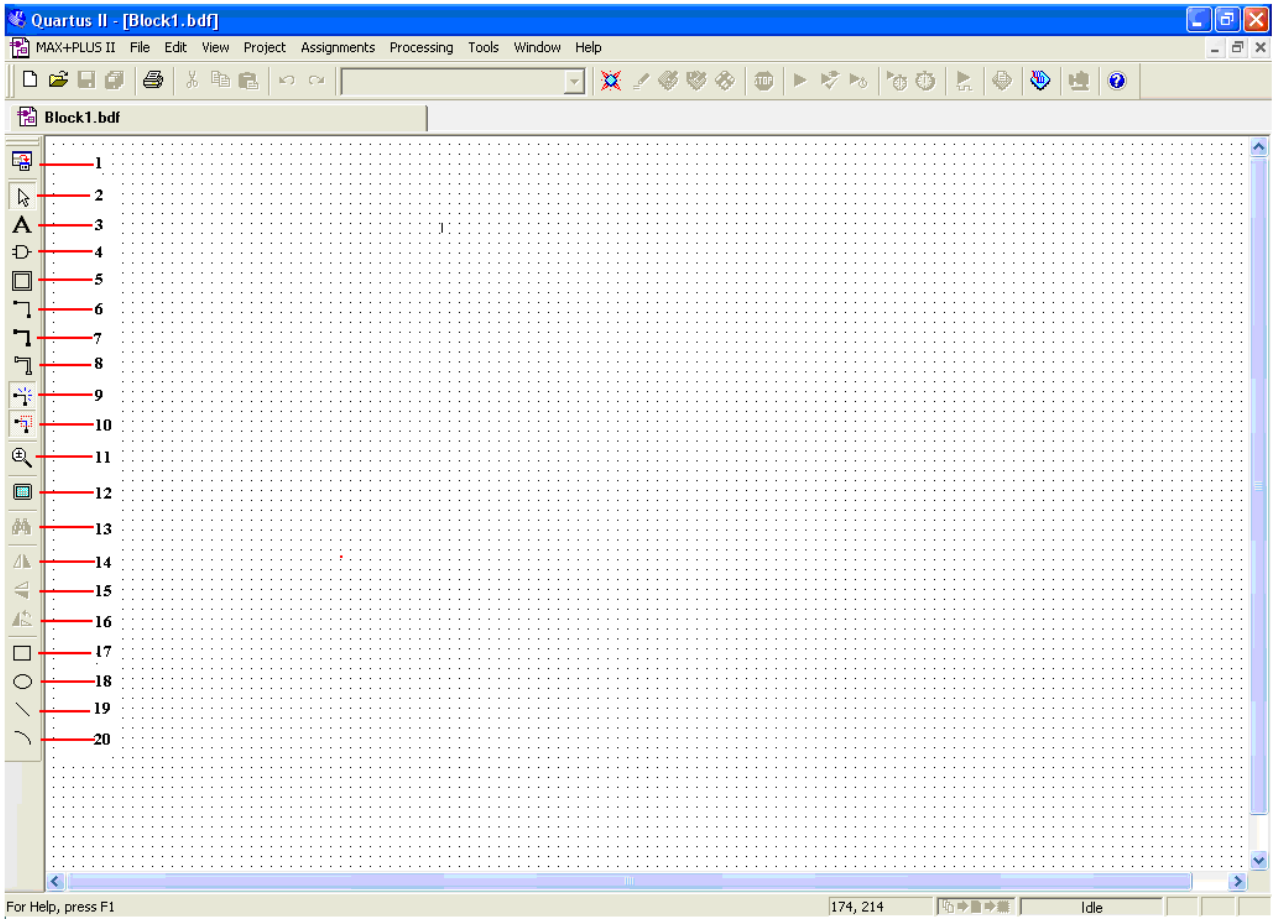


Figura 14: Ventana del Editor Gráfico

Las herramientas indicadas en la Figura 14, que posee el Editor Gráfico son las siguientes:

1- Detach Window: Botón para minimizar – agrandar la pantalla.
2- Selection Tool: Cursor de selección.
3- Text Tool: Herramienta de inclusión de texto.
4- Symbol Tool: Herramienta de inclusión de símbolos.
5- Block Tool: Herramienta de inclusión de bloques.
6- Orthogonal Node Tool: Botón para dibujar líneas ortogonales.
7- Orthogonal Bus Tool: Botón para dibujar líneas de bus ortogonales.
8- Orthogonal Conduit Tool: Botón para dibujar conductos ortogonales.
9- Use Rubberbanding: Ayuda a mover objetos preservando la conectividad.
10- Use Partial Line Selection: Ayuda a seleccionar parte de una línea.
11- Zoom Tool: Zoom in (botón izquierdo), Zoom out (botón derecho).
12- Full Screen: Ver el Editor Gráfico en pantalla completa.
13- Find: Botón de búsqueda.
14- Flip Horizontal: rotación horizontal.
15- Flip Vertical: rotación vertical.
16- Rotate Left 90: Rotación a la izquierda 90º.
17- Rectangle Tool: Botón para dibujar rectángulos.
18- Oval Tool: Botón para dibujar círculos / óvalos.
19- Line Tool: Botón para dibujar líneas rectas.
20- Arc Tool: Botón para dibujar líneas curvas.

- Inclusión de símbolos:

Para tener acceso a las diferentes funciones que facilita el software, se puede hacer: 1) pinchando sobre el botón de la herramienta 'Symbol Tool'; 2) hacer doble clic con el botón izquierdo ó pulsar el botón derecho del ratón sobre la pantalla del Editor Gráfico. A continuación se escoge la opción 'Insert Symbol', apareciendo la ventana que se muestra en la Figura 15. En esta ventana se busca la función, símbolo o puerta deseada entre las distintas carpetas que las recogen. También es posible escribir directamente el nombre del componente deseado en caso de conocerse. Para ello, escribir dicho nombre en el espacio reservado para esto después de 'Name'.

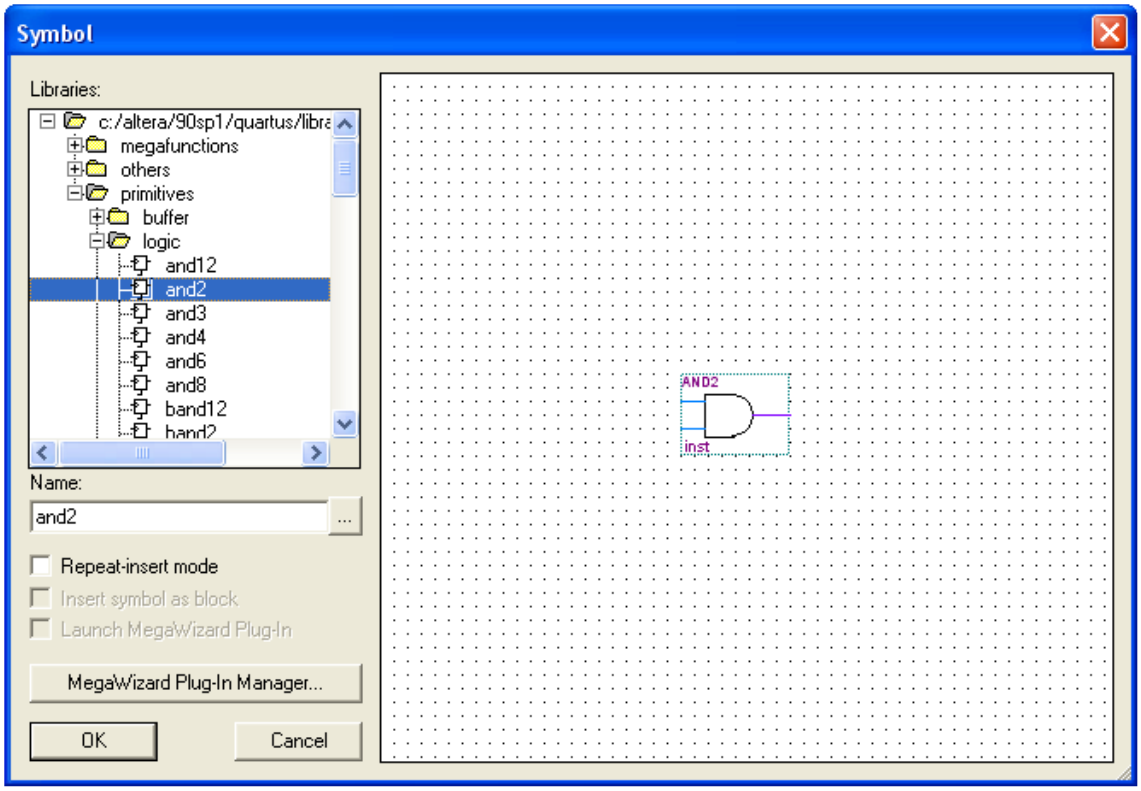


Figura 15. Selección de símbolos

- Como vemos en la Figura 15, en su parte inferior izquierda, aparecen varias opciones que es posible utilizar a la hora de incluir un símbolo. En la realización de las prácticas, activar la opción 'Repeat-insert mode' es útil en los casos en los que se quiere insertar un mismo símbolo en repetidas ocasiones.

En el caso de la inclusión de símbolos de entrada o salida (input, output), estos deben tener un nombre de identificación. Se puede asignar este nombre de identificación de dos formas diferentes:

- La primera de ellas consiste en hacer doble clic con el botón izquierdo del ratón sobre la leyenda 'pin-name' dentro del símbolo y entonces escribir el nombre.

- La segunda opción es hacer clic con el botón derecho del ratón sobre el símbolo y seleccionar la opción 'Properties' del menú que se despliega. A continuación aparece la ventana que se muestra en la figura 16.b y se introduce el nombre deseado dentro de la pestaña 'General' en el apartado denominado 'Pin name(s)'.

- La ventana que aparece dentro de esta opción no sólo permite cambiar el nombre del símbolo, además permite otras variaciones como asignar un valor por defecto, modificar las propiedades de líneas y texto.

Figura 16.a. Editar nombre

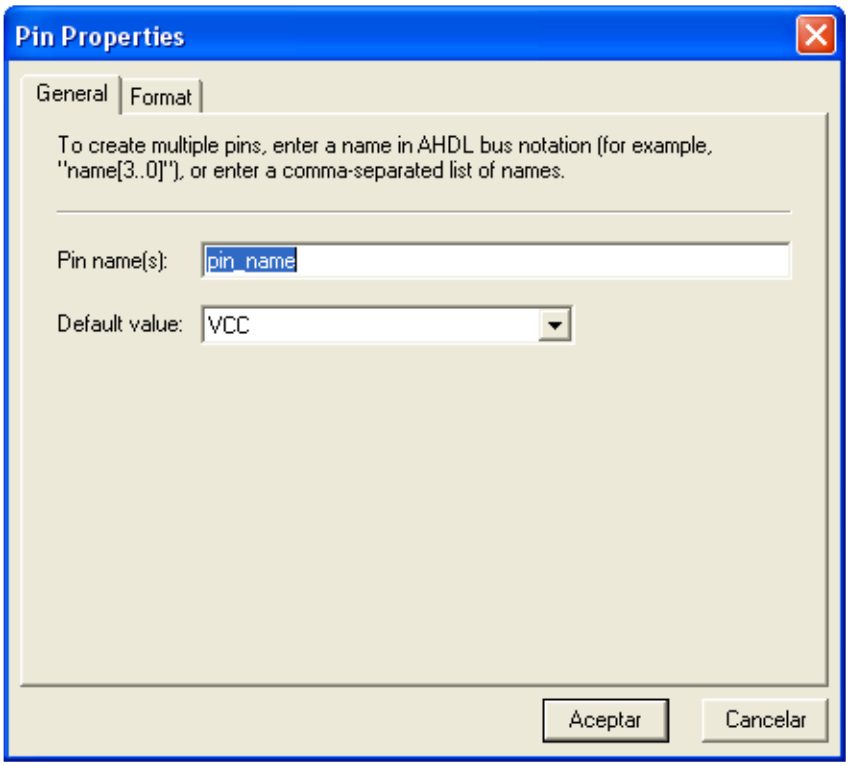


Figura 16.b. Editar propiedades

- *Conexión de símbolos:*

Una vez que se hayan incluido los símbolos que se desea utilizar en el diseño, se procede a su conexión.

La conexión entre los símbolos se hace con tres tipos de líneas diferentes, dependiendo del tipo de señal que se necesite conducir:

- Conexión normal: se representa por una línea delgada que sólo tiene capacidad para llevar una señal, es decir, representa un bit.
- Conexión en bus: se representa por una línea gruesa y permite llevar varias señales a diferentes destinos, es decir, representa vectores de bits. Los buses deben tener un nombre, y éste deberá ser igual que el de su respectiva entrada, tal y como se ve a continuación.

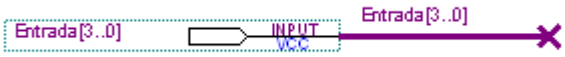


Figura 17: Bus de señales

- Conexión mediante conducto: se representa como dos líneas delgadas y permite llevar varias señales que el usuario añade según desee.

El nombre del bus se introduce haciendo clic con el botón derecho del ratón sobre la línea del bus y seleccionando la opción 'properties' del menú desplegable, teniendo como resultado la siguiente ventana:

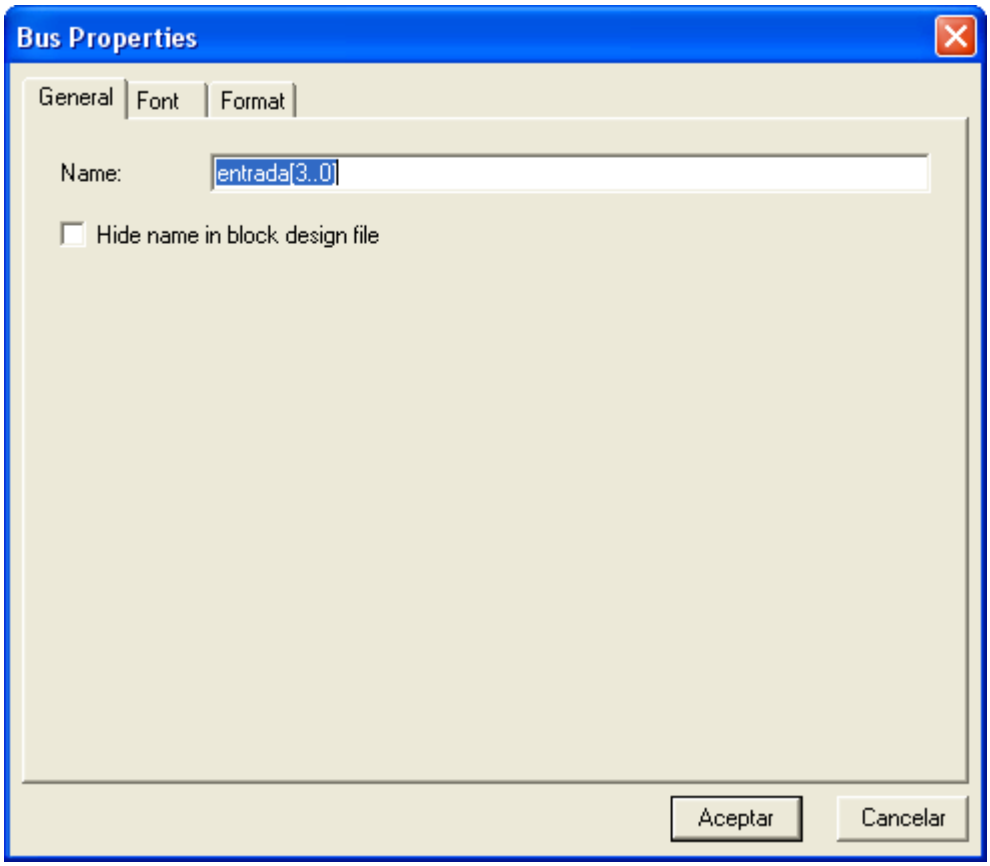


Figura 18. Propiedades de bus

Tal y como ocurría en la edición del nombre de los símbolos, desde la ventana mostrada en la Figura 18, también es posible modificar las propiedades de texto y formato del bus.

Además, en el menú desplegable que aparece cada vez que se hace clic con el botón derecho del ratón sobre cualquier tipo de línea tenemos la opción de poder cambiar entre los tipos de línea mencionados anteriormente.

En la Figura 17 se ve que entre corchetes aparece un tres y un cero separados por dos puntos. Esto quiere decir que el bus 'entrada' tiene cuatro señales diferentes, las cuales serán: entrada 3, entrada 2, entrada 1 y entrada 0.

Para dirigir las diferentes señales del bus a sus destinos, no es necesario que el destino esté conectado directamente con el bus. Se puede hacer la conexión por el nombre del bus o por el nombre de cada señal del bus.

Hay que asegurarse de utilizar diferentes nombres para cada uno de los pines. Los cables y pines con el mismo nombre son conectados automáticamente sin que aparezca un cable visible en el diagrama esquemático.

La figura 19 muestra un ejemplo de conexiones (línea, bus, conducto)

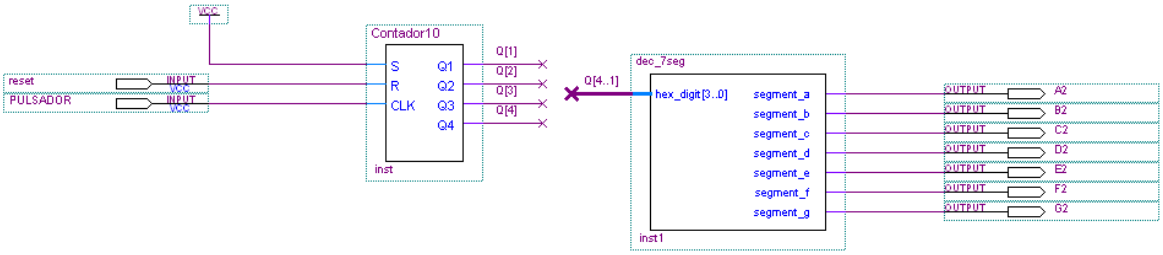


Figura 19: Ejemplo de conexiones

Además de lo visto hasta ahora, el Editor Gráfico permite manipular los diseños como cualquier paquete de dibujo, es decir, existe la posibilidad de rotar, obtener imágenes de espejo vertical u horizontal, cambiar el diseño de líneas, etc.

- *Guardar el proyecto:*

Por último, para guardar el proyecto se accede a la opción 'File' de la barra de herramientas principal y se hace clic sobre 'Save As…' dentro del menú desplegable. Será ahora cuando se introduzca el nombre del proyecto, que por haber sido realizado mediante el Editor Gráfico deberá ir acompañado de la extensión '.bdf'.

Al ejecutar el paso anterior, si previamente no se ha definido un nuevo proyecto sino que directamente se ha abierto el Editor Gráfico, el software pregunta si se quiere crear un nuevo proyecto. Al responder afirmativamente aparece la serie de pasos que definen las características del mismo. Estos pasos son los siguientes:

I. Definición de nombre y directorio de almacenamiento del proyecto. Inclusión de archivos anteriores que quieran ser utilizados en el presente proyecto.
II. Selección de la familia y dispositivo que estemos utilizando.
III. Selección de otras herramientas EDA.
IV. Ventana resumen de los pasos anteriores.

Una vez terminado el diseño del circuito deseado, el siguiente paso es el de la compilación, para ver si existen errores en la construcción del mismo.

- Nota: una vez guardado el proyecto, para volver a abrirlo en otra ocasión, seleccionar el archivo con extensión '.qpf' para abrir el proyecto completo. A continuación, abrir el archivo de diseño que se quiera.
- Nota: no guardar los proyectos con nombres de funciones existentes en la librería del software, ya que a la hora de compilar dará errores.

6.7.- EDITOR DE TEXTO (TEXT EDITOR)

Además de mediante el Editor Gráfico, también es posible la realización de diseños utilizando el Editor de Texto que contiene Quartus II. La utilización del Editor de Texto se hace necesaria ya que la programación en el Editor Gráfico se puede volver muy tediosa para diseños con un mínimo de complicación. Por lo tanto, se utiliza el Editor Gráfico para construir diseños sencillos y para unir diferentes bloques hechos en el Editor de Texto, para así poder ver gráficamente el diagrama de bloques que conforma el diseño y facilitar la compresión del mismo.

El Editor de Texto que contiene Quartus II permite elegir entre las siguientes opciones para la programación:

- AHDL: Altera Hardware Description Language.
- EDIF: Electronic Design Interchange Format.
- SystemVerilog HDL: System Verilog Hardware Description Language.
- Tcl script: Tool Command Language script.
- Verilog HDL: Verilog Hardware Description Language.
- VHDL: Very High Speed Integrated Circuit (VHSIC) Hardware Description Language.

En las practicas se hará una introducción al lenguaje de programación VHDL, ya que éste es un lenguaje de programación estándar y universalmente aceptado, lo cual permitirá volcar los diseños en los dispositivos de Altera o en los de cualquier otra compañía.

Para comenzar a utilizar el Editor de Texto existen dos opciones, tal y como pasaba con el Editor Gráfico. La primera de las opciones, definida por defecto en Quartus II, consiste en abrir un nuevo documento, para lo cual se pincha sobre el icono del folio en blanco que aparece en la parte superior izquierda de la pantalla.

A continuación, aparecerá el menú que se muestra en la Figura 20 y, en dicho menú, se selecciona la opción 'VHDL File'. Ahora se estará en condiciones de utilizar el Editor de Texto con el lenguaje de programación VHDL.

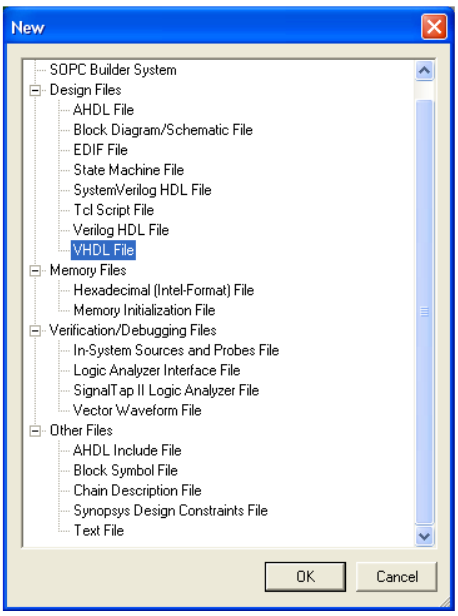


Figura 20. Nuevo archivo VHDL

La segunda opción para abrir un nuevo documento de texto, es acudir a la pestaña 'MAX+PLUS II' de la barra de herramientas principal y seleccionar la opción 'Text Editor' de su correspondiente menú desplegable.

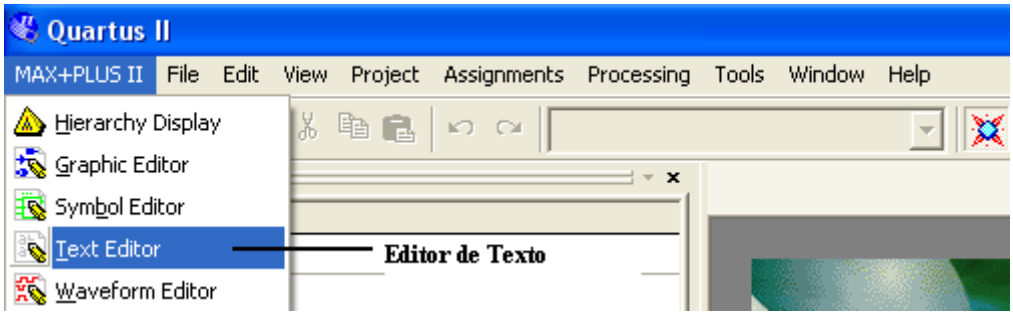


Figura 21. Selección del Editor de Texto mediante el menú de Max+Plus II

Al abrir el Editor de Texto de esta manera, se genera un archivo de texto con extensión '.txt'. Para que el programa sepa que la programación se va a realizar con el lenguaje VHDL se debe guardar el proyecto según un archivo correspondiente a este lenguaje. Para ello se accede a la pestaña 'File' y se selecciona la opción 'Save As…'. Una vez hecho esto, aparece la ventana en la que se define el nombre del proyecto y el tipo de archivo que se va a guardar. Por tanto, en nuestro caso, se selecciona el tipo 'VHDL File' que cuenta con la extensión '.vhd'.

Como se ha comentado anteriormente, en las prácticas se va a realizar una introducción a la programación con VHDL, por tanto, las nociones acerca de este lenguaje se verán en el correspondiente guión de prácticas.

Una vez definido el proyecto, para guardarlo se siguen los mismos pasos que en el caso del Editor Gráfico. Se accede a la pestaña 'File' situada en la barra de herramientas principal y se selecciona la opción 'Save As…' de su correspondiente menú desplegable. En este momento aparece la ventana en la cual se introduce el nombre del proyecto acompañado de la extensión '.vhd' por haber sido realizado con el Editor de Texto en VHDL.

6.8.- COMPILADOR (COMPILER)

Una vez que se ha terminado de realizar el diseño deseado, bien sea a través del Editor Gráfico o del Editor de Texto, se procede a compilar el proyecto. En esta fase se lleva a cabo la detección de posibles errores existentes a la hora de construir el diseño gráficamente o en la sintaxis del mismo en el caso de trabajar con el Editor de Texto.

Es necesario que antes de iniciar la compilación del proyecto, éste haya sido previamente guardado.

El acceso a la herramienta de compilación de Quartus II puede realizarse de dos maneras diferentes:

- La primera de ellas consiste en acceder al menú desplegable de la pestaña 'Processing' de la barra de herramientas principal y seleccionar la opción 'Compiler Tool', como se ve en la Figura 22.
- La segunda opción consiste en iniciar directamente la compilación accediendo al icono asignado para ello, tal y como muestra la Figura 22.

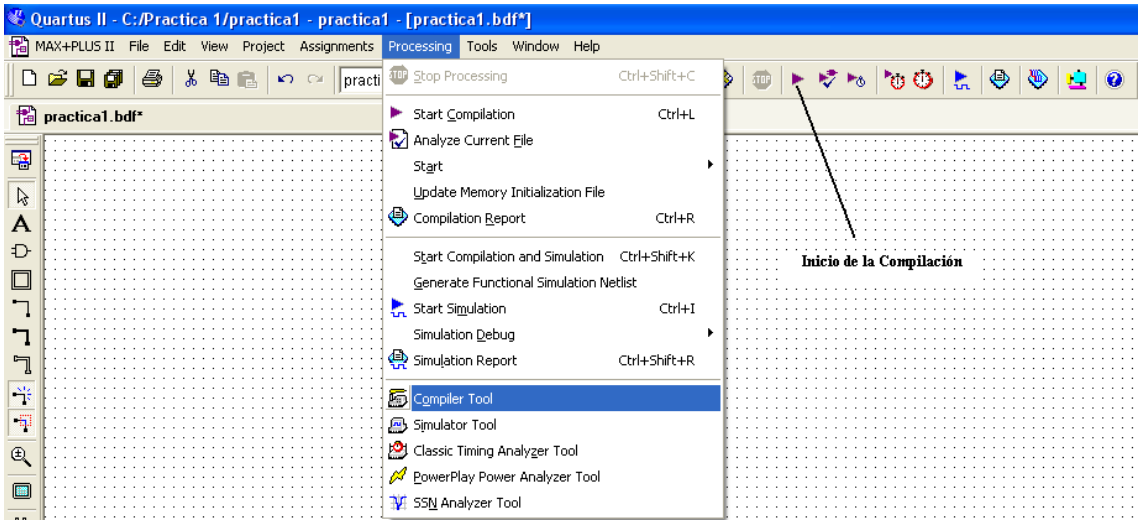


Figura 22. Apertura de la herramienta de compilación

En el caso de escoger la primera opción para abrir el compilador, una vez seguidos los pasos mencionados anteriormente, aparece la ventana mostrada en la Figura 23.

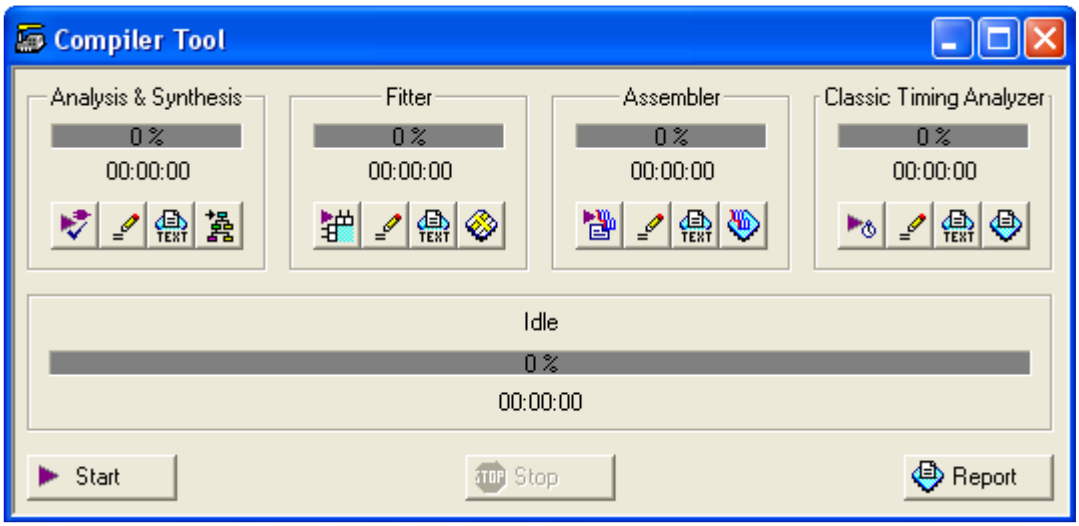


Figura 23. Ventana del Compilador

Como se puede ver en la Figura 23, el compilador de Quartus II permite realizar la compilación completa o por partes según interese al usuario. El proceso de compilación completo comienza cuando se pulsa el botón 'Start', y éste puede ser interrumpido mediante el botón 'Stop'. Para ver las características del proceso es necesario recurrir al botón 'Report'.

En el caso de que se quiera realizar la compilación por partes, es necesario tener en cuenta el orden de compilación, es decir, por ejemplo no se puede recurrir a la herramienta 'Assembler' sin haber pasado anteriormente por las herramientas 'Analysis & Synthesis' y 'Fitter' en este orden.

Conforme se realiza la compilación del proyecto, se despliega la ventana de mensajes, mostrada en la Figura 24, que indica los diferentes sucesos surgidos en el proceso de compilación.

Pueden existir mensajes de error, escritos en rojo y que provocan la interrupción del proceso de compilación, mensajes de precaución, marcados en azul, y mensajes de información, escritos en verde.

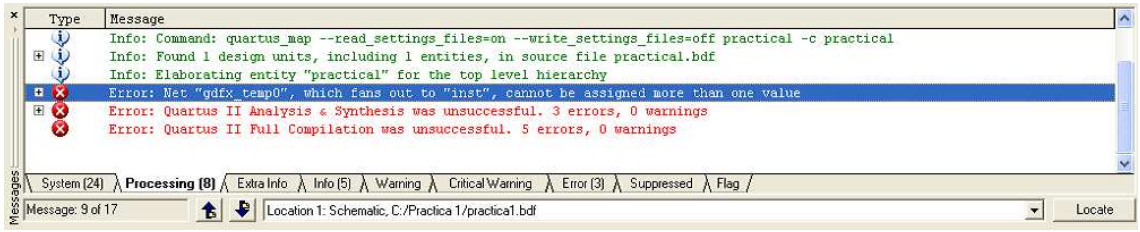


Figura 24. Ventana de errores

Si se tiene algún error, es posible seleccionarlo bien con el ratón o bien con las flechas de desplazamiento entre mensajes para después poder localizarlo.

Para localizar el error se pulsa el botón 'Locate' e inmediatamente regresa al editor en el que se diseñó el circuito y resalta el símbolo o el texto erróneo.

Si por el contrario, la compilación del proyecto resulta satisfactoria, aparecerá en pantalla la ventana que se muestra en la Figura 25.

Figura 25. Compilación satisfactoria

Al pulsar el botón 'Aceptar' se da por concluido el proceso de compilación del proyecto y ya se está en condiciones de comenzar la simulación del mismo.

Durante la compilación, el programa selecciona el dispositivo asignado previamente en la definición del proyecto. Si se hubiera obviado este paso, el programa selecciona un dispositivo automáticamente. De todos modos siempre se podrá cambiar el dispositivo posteriormente. Para ello se accede al menú desplegable 'Assignments' y se escoge la opción 'Device' apareciendo en pantalla la ventana mostrada en la Figura 26.

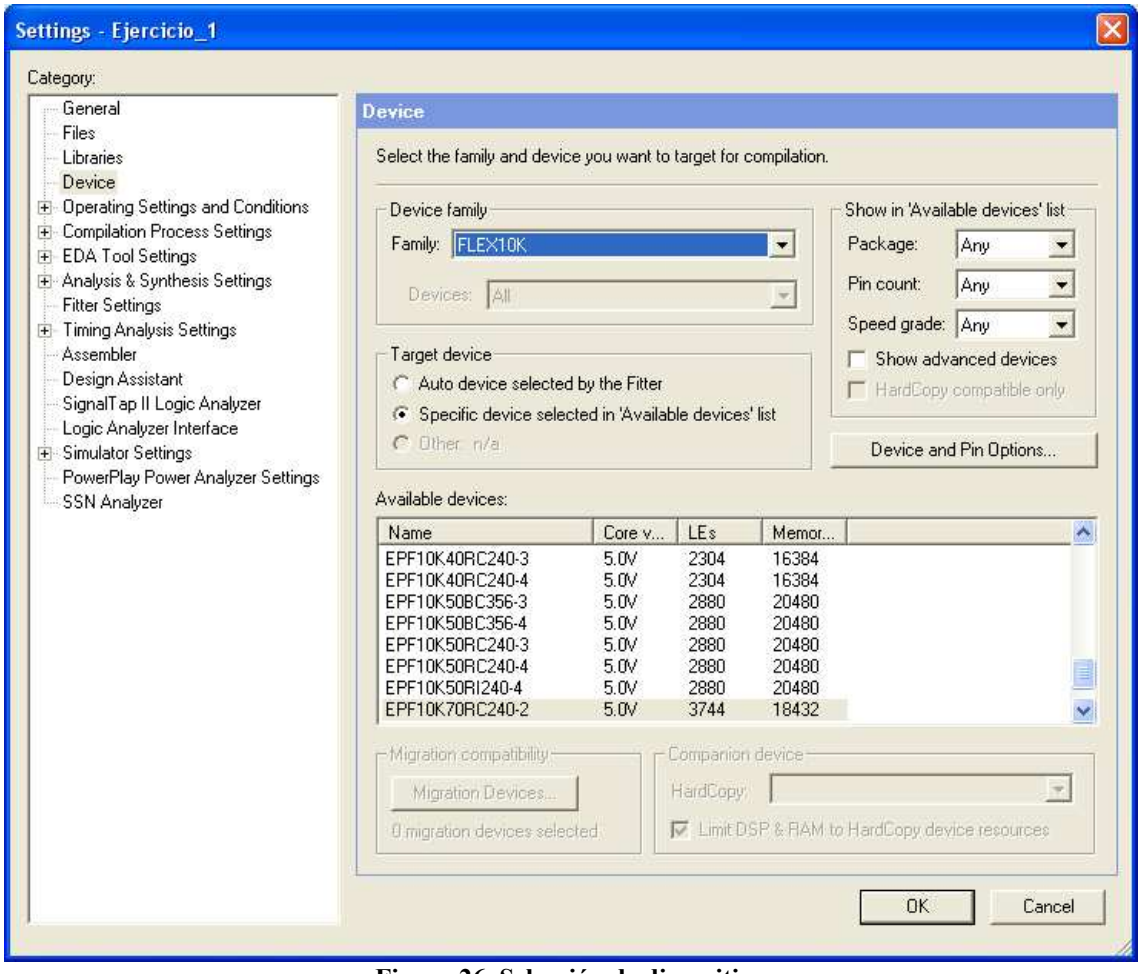


Figura 26. Selección de dispositivo

- *RTL Viewer:*

Como resultado de la compilación, Quartus II genera un diagrama esquemático de las conexiones hardware que deberán ser programadas en el dispositivo programable, lo que permite comprobar si las conexiones realizadas en el diseño están bien realizadas.

Para acceder a esta herramienta se debe acudir al menú 'Tools' y, a continuación, seleccionar la opción 'Netlist Viewers' seguido de 'RTL Viewer'.

Un ejemplo de conexiones mediante esta herramienta se ve en la siguiente figura:

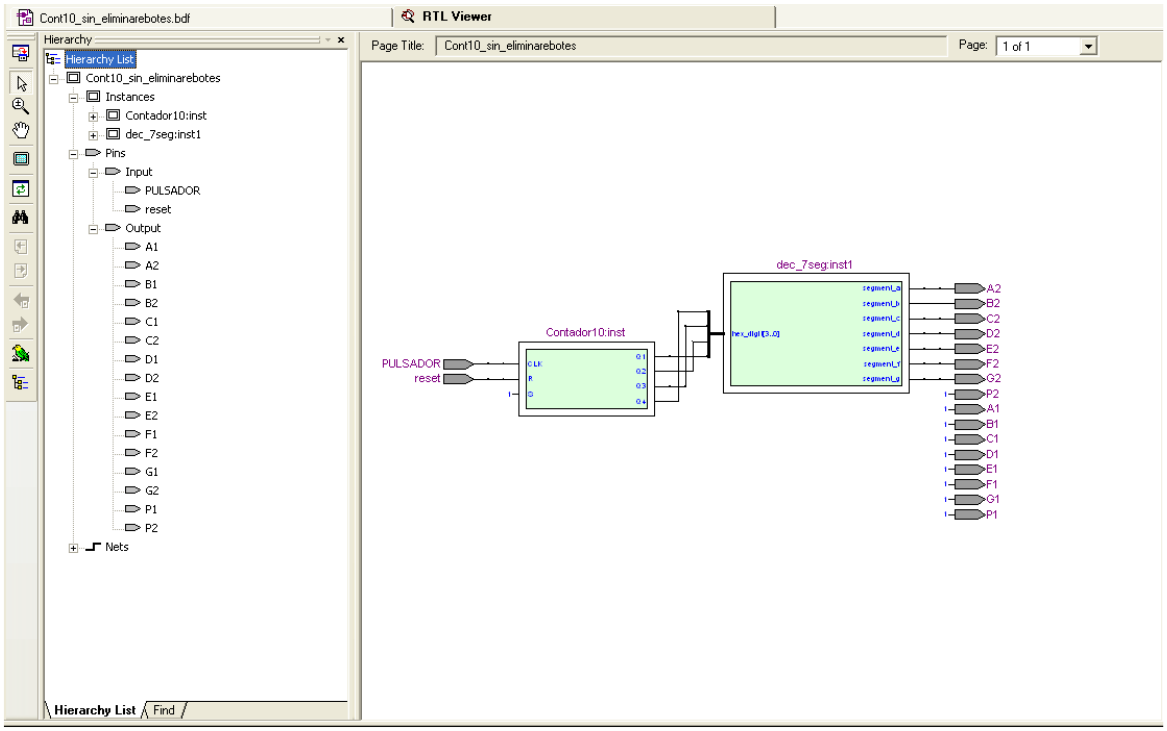


Figura 27. RTL Viewer

6.9.- EDITOR DE SÍMBOLOS (SYMBOL EDITOR)

Existen diseños en los que, debido a su complejidad, resulta interesante dividir el problema a resolver en problemas más pequeños, de mayor facilidad de resolución. Cada una de las partes en las que se ha dividido el diseño serán posteriormente unidas, compiladas y simuladas. Es por este motivo por lo que el software de desarrollo, Quartus II, ofrece la posibilidad de crear como símbolo cada una de las partes mencionadas anteriormente, teniendo la posibilidad de que estos símbolos puedan ser integrados en otros diseños más complicados.

Para crear un símbolo se deben seguir los siguientes pasos:

1- En primer lugar, el diseño deberá estar compilado, pudiendo haber sido hecho tanto en el Editor Gráfico como en el Editor de Texto.
2- Acceder al menú desplegable 'File' en la barra de herramientas principal y seleccionar la opción 'Create / Update' – 'Create Symbol Files For Current File'
3- Una vez seguidos estos pasos, aparece la ventana que permite guardar el símbolo en la carpeta deseada y con el nombre que se quiera asignar al símbolo. Debido a la utilización del Editor de Símbolos, el símbolo queda guardado seguido de la extensión '.bsf'.
4- Si tras la inclusión de un símbolo en un diseño, este quiere ser modificado, se debe acceder a la opción 'Edit Selected Symbol' bien sea desde el menú desplegable 'Edit' de la barra de herramientas principal o bien pinchando con el botón derecho del ratón sobre el símbolo a editar.

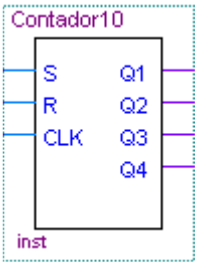


Figura 28. Ejemplo de símbolo

El software asigna por defecto al símbolo el mismo nombre que tiene el proyecto, aunque este puede ser modificado. Al símbolo se le puede dar el formato que se quiera, pudiendo cambiar el tamaño y forma del símbolo, introducir un texto de referencia o un texto explicativo del símbolo, etc. Una vez modificado el símbolo, éste volverá a ser guardado y estará listo para ser usado en cualquier oto diseño más complejo. Para guardar el símbolo se vuelve a acceder al menú 'File' y se escoge la opción 'Save'.

No será necesario utilizar el Editor de Símbolos cada vez que se cree un símbolo, ya que en la mayoría de las ocasiones la configuración y formato que asigna el software por defecto será adecuada. Asimismo, se recomienda la creación del símbolo una vez comprobado el correcto funcionamiento del diseño.

6.10.- EDITOR DE SEÑALES (WAVEFORM EDITOR)

Una vez construido y compilado el diseño, resulta necesario comprobar el correcto funcionamiento del mismo antes de que se implemente sobre el sistema, en este caso, sobre las tarjetas educacionales DE2 ó UP2.

Este paso puede ser obviado, siendo posible la comprobación del correcto funcionamiento directamente sobre el sistema físico, sin embargo esto no es aconsejable ya que la simulación a menudo ofrece una visión más detallada de la operación del diseño. Además, es conveniente realizar la simulación ya que en el diseño se pueden producir errores que podrían llegar a afectar al dispositivo si se implementara directamente sobre el mismo. Es por esto que la simulación se considera una herramienta fundamental a la hora de obtener buenos resultados en los respectivos diseños.

- Selección de las señales a observar

A la hora de realizar una simulación lo primero que se debe hacer es seleccionar un grupo de señales lógicas, que son las que se utilizarán para la comprobación del correcto funcionamiento del diseño. Estas señales incluyen tanto las entradas como las salidas globales del diseño así como ciertas señales internas o intermedias. La inclusión de estos grupos de señales se realiza con la herramienta 'Editor de Señales' disponible en el software Quartus II. La forma de abrir esta herramienta consiste en acceder al icono del folio en blanco situado en la parte superior derecha del entorno gráfico y a continuación seleccionar la opción 'Vector Waveform File', tal y como muestra la Figura 29.

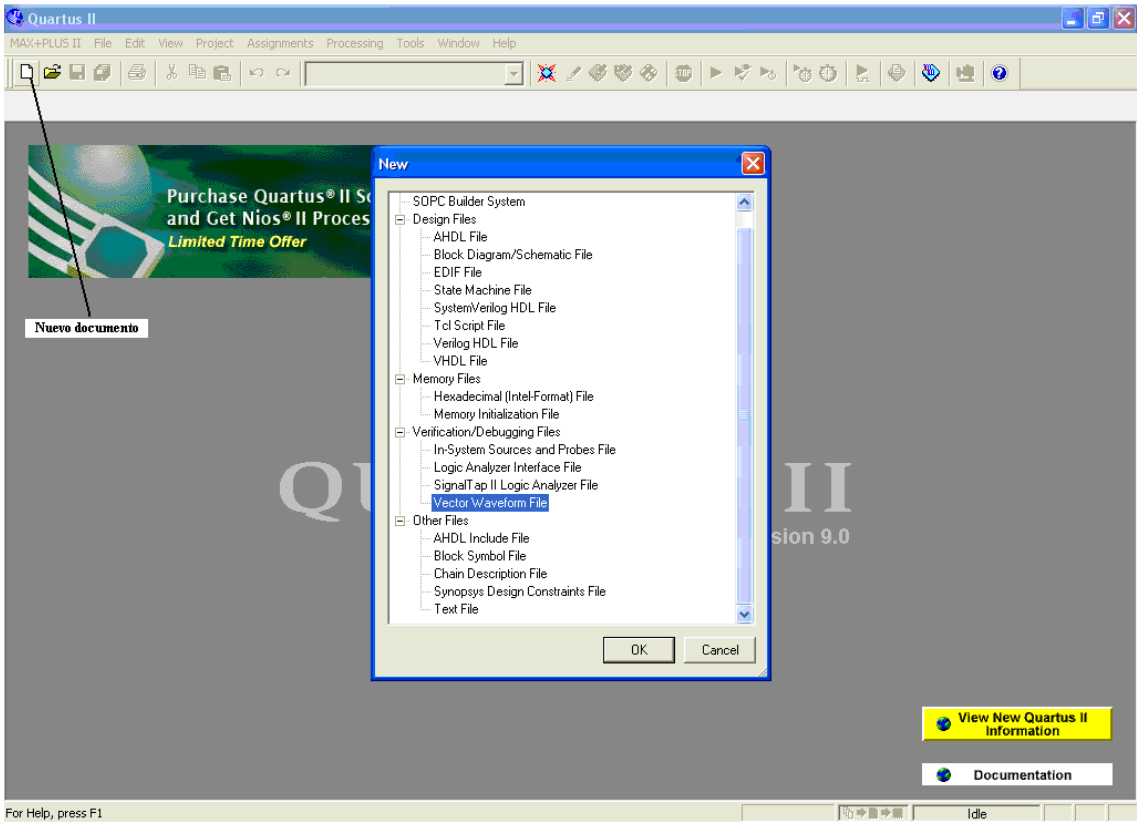


Figura 29. Cómo abrir el Editor de Señales

Una segunda manera de abrir el Editor de Señales consiste en acceder al menú desplegable MAX+PLUS II situado en la barra de herramientas principal y seleccionar la opción 'Waveform Editor', como se ve en la Figura 30.

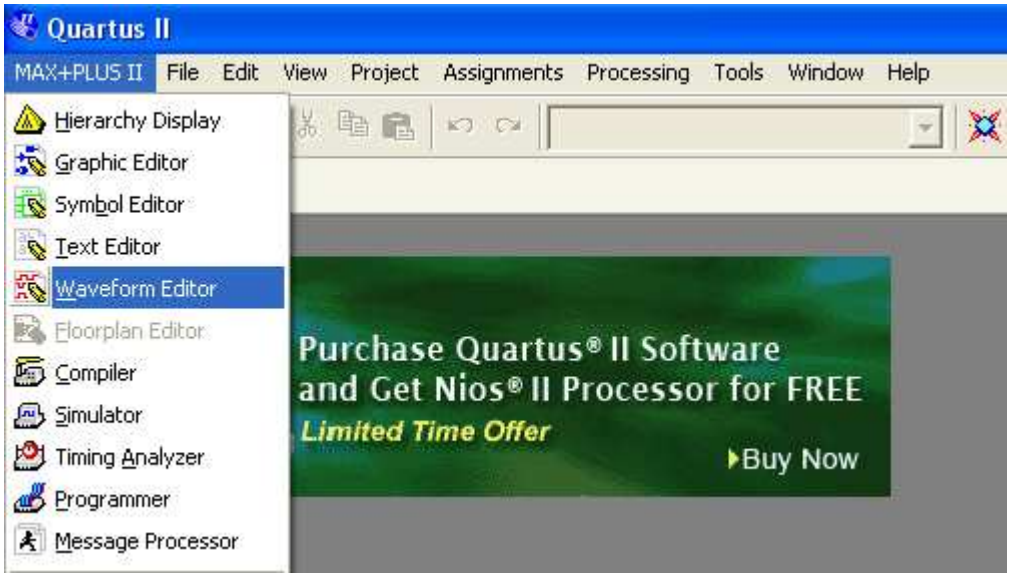


Figura 30. Abrir Editor de Señales desde menú MAX+PLUS II

Recurriendo a cualquiera de las dos formas de abrir el Editor de Señales se obtiene la ventana que se muestra a continuación en la Figura 31.

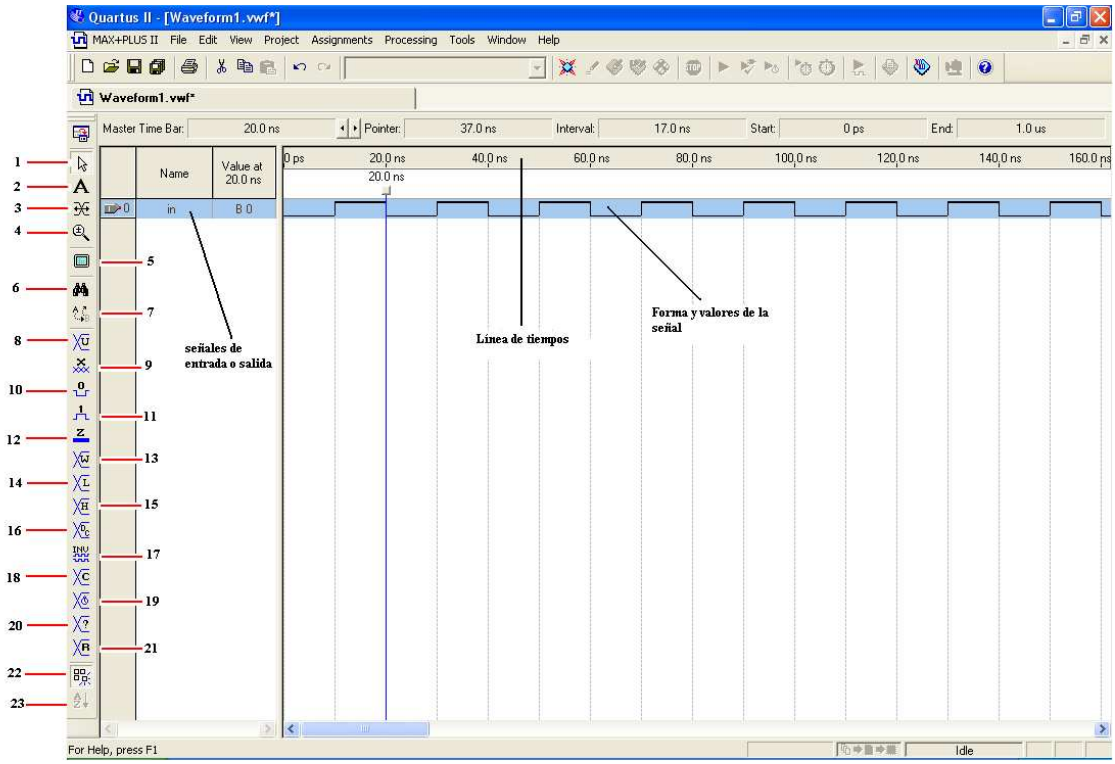


Figura 31. Editor de Señales

Las diferentes herramientas que posee el Editor de Señales se definen a continuación:

1- Selection Tool: Selecciona un intervalo de onda y posibilita asignarle el valor deseado a la onda.
2- Text Tool: Coloca un texto de referencia en alguna parte de la onda.
3- Waveform Editing Tool: Cambia el valor de la onda en un intervalo (se arrastra el ratón sobre el intervalo que se desea cambiar).
4- Zoom Tool: Zoom in / zoom out.
5- Full Screen: Permite maximizar la ventana del Editor de Señales.
6- Find: Realiza una búsqueda según un texto especificado.
7- Replace: Herramienta para reemplazar texto.
8- Unintialized (U): Sobrescribe la entrada seleccionada con un determinado nivel lógico.
9- Forcing Unknown (X): Asigna valor indeterminado a la onda.
10- Forcing Low (0): Asigna valor bajo a la onda.
11- Forcing High (1):Asigna valor alto a la onda.
12- High Impedance (Z): Asigna valor de alta impedancia lógica a la onda.
13- Weak Unknown (W): Sobrescribe la entrada seleccionada con un nivel desconocido.
14- Weak Low (L): Sobrescribe la entrada seleccionada con un nivel bajo.
15- Weak High (H): Sobrescribe la entrada seleccionada con un nivel alto.
16- Don`t care (DC): Sobrescribe la entrada seleccionada donde no importa el nivel lógico.
17- Invert: Invierte el valor de la onda en un intervalo dado.
18- Count value: Asigna un determinado periodo de onda a la señal.
19- Overwrite clock: Asigna un valor de reloj para circuitos secuenciales.
20- Arbitraty value: Sobrescribe la entrada seleccionada con un valor arbitrario.
21- Random value: Sobrescribe la entrada seleccionada con un valor aleatorio, forzando este valor al tamaño del periodo.
22- Snap to grid: Fuerza la selección a la longitud entre líneas de tiempo.
23- Sort: Ordena las diferentes señales por nombre y tipo.

En esta misma ventana, visualizada en la Figura 31, en la parte superior izquierda de la pantalla bajo la etiqueta 'Name' aparecerán los nombres de las entradas y salidas seleccionadas para la simulación. Los valores de las mismas aparecen en la parte derecha de la pantalla a su misma altura bajo la línea de tiempos de referencia.

Se debe ahora elegir las señales a estudiar. Para ello es necesario pulsar con el botón derecho del ratón sobre la parte izquierda de la pantalla, en el lugar donde se ven los nombres de las señales, para que aparezca la ventana que se muestra a continuación en la Figura 32.

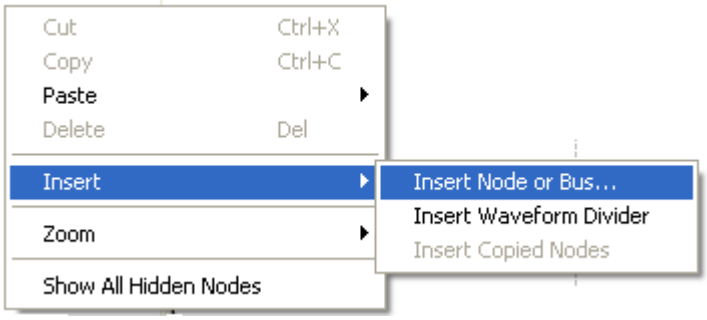


Figura 32. Acceso a la selección de nodos

Siguiendo los pasos que se muestran en la Figura 32 (Insert – Insert Node or Bus) aparece la ventana mostrada en la Figura 33, la cual permite elegir el tipo de señales que interesa mostrar para la simulación del diseño.

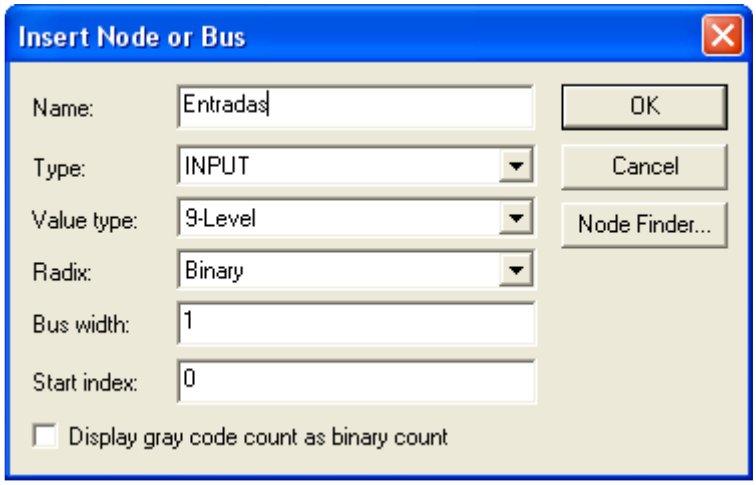


Figura 33. Tipos de nodo

Una vez seleccionado el tipo de señales que interesa mostrar, pulsando sobre el botón 'Node Finder' es posible acceder a la ventana que se muestra en la Figura 34, donde se realiza la selección del conjunto de señales que se quieren observar durante la simulación.

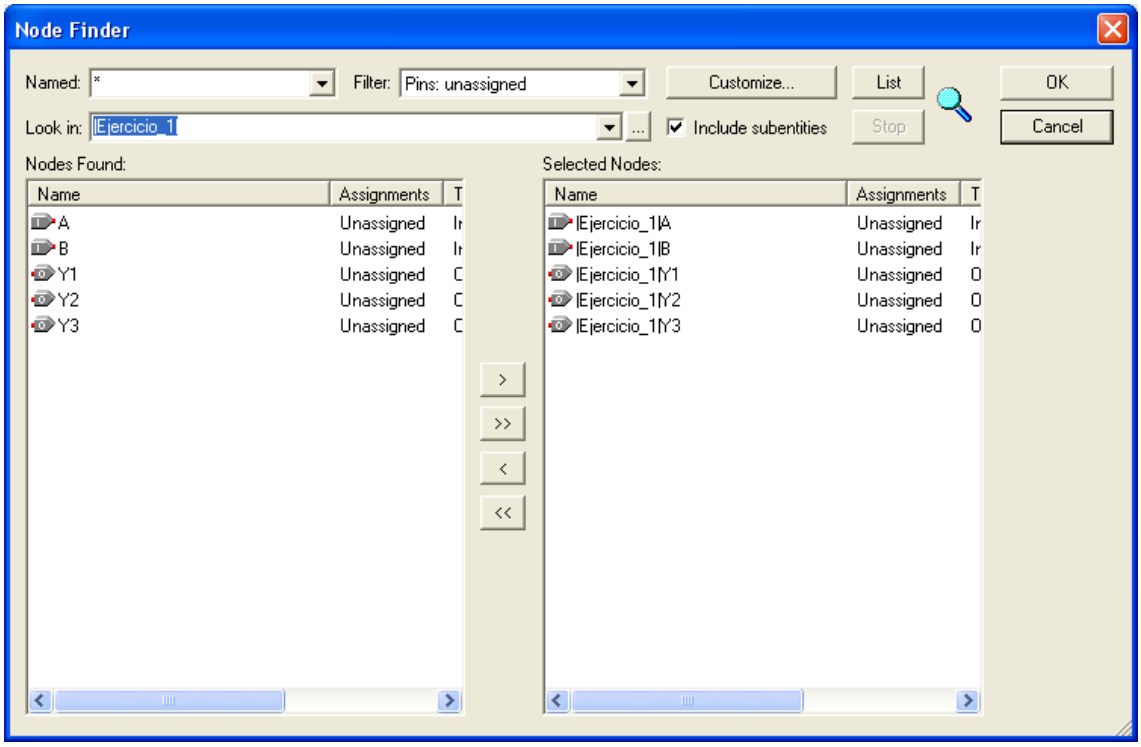


Figura 34. Ventana de selección de nodos

Pulsando el botón 'List', situado en la parte superior de la ventana, aparecerán todas las señales del tipo que se haya seleccionado en la ventana correspondiente a la Figura 33, para ahora seleccionar de entre estas señales las que interesa analizar, pasándolas a la parte derecha de la pantalla mediante los botones con las flechas de dirección.

El botón 'Customize' permite realizar un filtrado, según distintos parámetros, de las señales que se quieren utilizar.

La operación queda aceptada cuando se pulsa el botón 'OK', apareciendo en el editor las señales seleccionadas. Es posible que estas señales aparezcan desordenadas, para ordenarlas bastará con arrastrarlas mediante el ratón hasta la posición deseada.

6.11.- SIMULADOR (SIMULATOR)

La finalidad de esta herramienta consiste en la comprobación del correcto funcionamiento de los respectivos diseños antes de que éstos sean descargados en la tarjeta educacional. Este paso es importante realizarlo ya que podrían existir errores en los diseños que incluso podrían dañar la placa.

Al igual que en el resto de herramientas, existen diferentes opciones a la hora de abrir la herramienta del simulador.

- La primera de las opciones consiste en acceder al menú desplegable 'Processing' situado en la barra de herramientas principal y elegir la opción 'Simulator Tool', como se ve en la Figura 35.
- La segunda opción consiste en iniciar directamente la simulación pulsando sobre el icono asignado para ello.

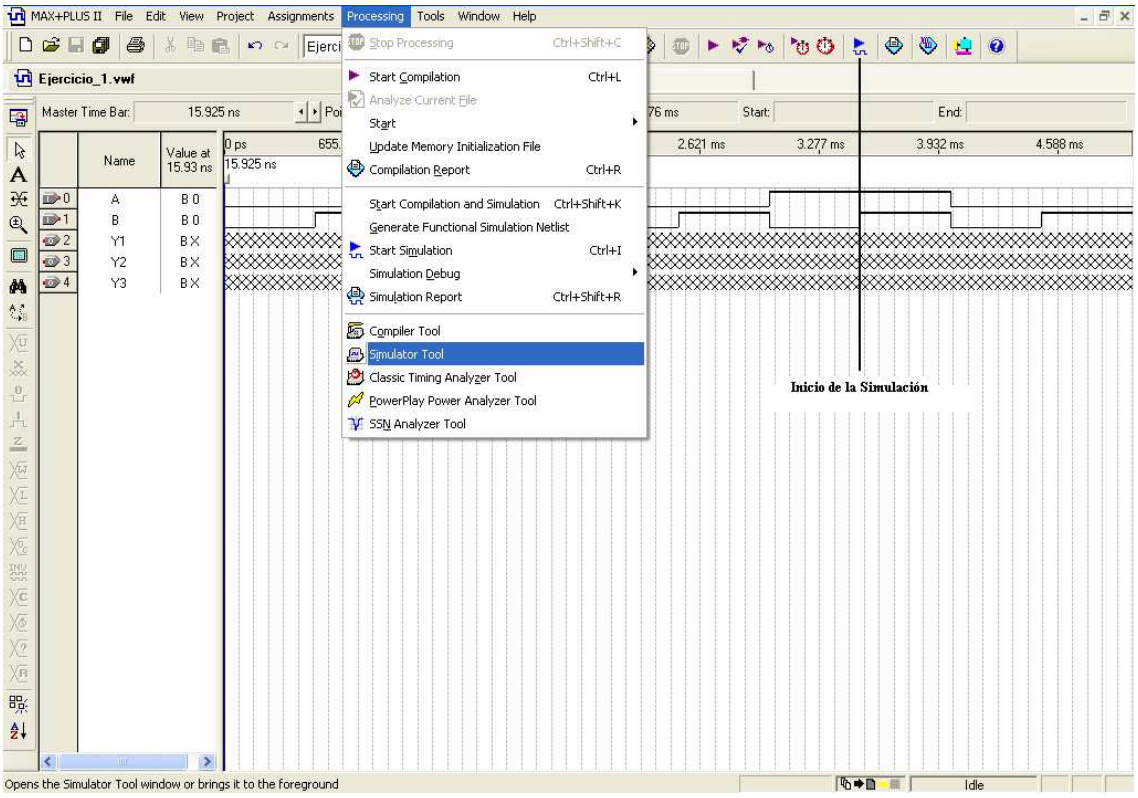


Figura 35. Acceso a la herramienta de simulación

Si se opta por elegir la primera manera de abrir esta herramienta aparece una nueva ventana en pantalla, mostrada en la Figura 36, la cual muestra el estado del simulador. Basta con pulsar el botón 'Start' para comenzar con la ejecución de la simulación.

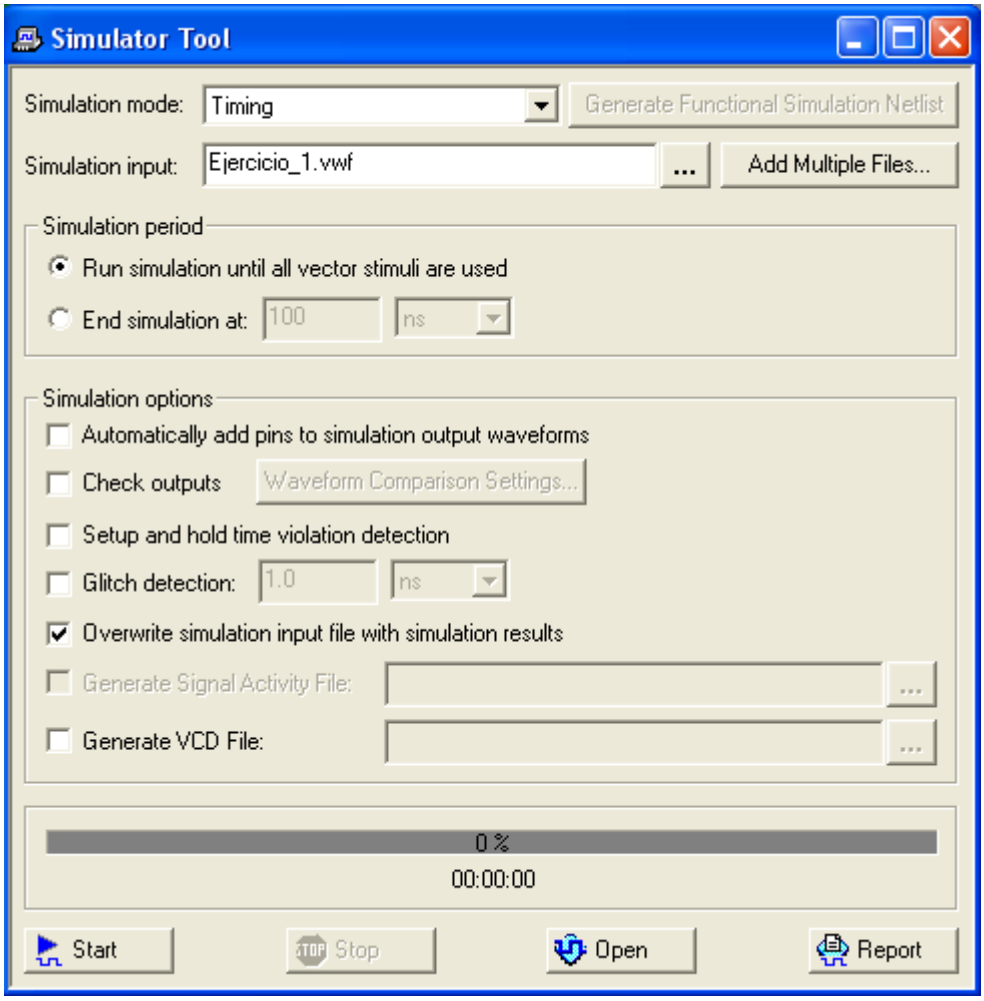


Figura 36. Ventana del simulador

Un mensaje similar al mostrado en la Figura 37 aparecerá una vez concluido el proceso.

Figura 37. Fin de la simulación

Una vez terminada la simulación se está en disposición de poder ver y analizar los resultados. Para ello, en la misma ventana del simulador existen los botones 'Open' y 'Report'.

Si se pulsa sobre el botón 'Open' se modifica y se guarda el archivo original creado desde el Editor de Señales, pudiendo observar en él tanto las entradas como las salidas resultado de la simulación.

Si por el contrario, se pulsa sobre el botón 'Report' podremos observar el resultado de la simulación (entradas y salidas), pero el archivo original creado con el Editor de Señales permanecerá sin cambios.

Si se opta por abrir la herramienta del simulador de la segunda de las maneras indicadas anteriormente, se consigue iniciar directamente la simulación sin tener que pasar previamente por la ventana de la herramienta, mostrada en la Figura 36.

El resultado que se obtiene es el mismo que el obtenido si se pulsa el botón 'Report' en la ventana del simulador.

Una vez mostrados los resultados en pantalla, se deberá comprobar que éstos tienen sentido y son lo que cabría esperar. Esto en algunos casos será un proceso sencillo, mientras que en otros será más complicado, dependiendo de la complejidad del diseño. Por este motivo es importante tener bien claro las entradas y/o salidas que se quieren observar y qué es lo que se pretende conseguir con el diseño.

6.12.-PROGRAMADOR (PROGRAMMER). VOLCADO DE DISEÑOS A LA TARJETA EDUCACIONAL.

El programador de Quartus II es la herramienta utilizada para volcar los diseños a un PLD. Es este caso, los dispositivos de los que se dispone, son los dos que contiene la tarjeta educacional UP2, y el dispositivo instalado sobre la placa DE2, todos ellos de la compañía Altera:

- EPM7128S: es un dispositivo de la familia MAX 7000S, basada en elementos EEPROM, por lo que el número de veces que se puede programar es limitado (superior a 100). El dispositivo almacenará el programa sin necesidad de ser alimentado.
- EPF10K70: dispositivo de la familia FLEX 10K, basado en elementos SRAM reconfigurables, por lo que el número de veces que se puede programar es ilimitado. Al interrumpirse la alimentación, el programa descargado en este dispositivo se borrará.
- 2C35 FPGA: dispositivo de la familia Cyclone II, capaz de ser programado un número ilimitado de veces. Según el modo de programación, es posible que la información volcada a la placa se borre o se mantenga al cortar la alimentación.

La programación en un dispositivo programable es el último paso en el desarrollo de un diseño. Los archivos necesarios para este propósito son generados en Quartus II después de que el diseño ha sido satisfactoriamente compilado. La familia de dispositivos MAX 7000S usa los archivos de extensión '.pof' (Programmer Object File) como formato para la programación, mientras que la familia de dispositivos FLEX 10K emplea los archivos con extensión '.sof' (SRAM Object File) para la misma operación. La familia Cyclone II soporta ambos tipos de archivo.

Estos archivos son generados por el compilador, incluyendo todos los datos para la correcta configuración del correspondiente PLD. Por tanto, los archivos de extensión '.pof' configuran elementos EEPROM mientras que los archivos '.sof' sirven para elementos basados en FPGA SRAMs.

- Volcado de diseños a la tarjeta educacional UP2:

A continuación se presenta la serie de pasos que permiten la configuración de la tarjeta educacional UP2 de Altera, a través del software Quartus II:

1.- Debido a las características de los dispositivos existentes en la placa, los diseños van a ser volcados sobre el dispositivo de la familia FLEX 10K ya que éste permite ser reconfigurado ilimitadamente. Para ello, la tarjeta debe ser configurada a través de los puentes de la placa de tal forma que sólo se permita la configuración del dispositivo EPF10K70, tal como se muestra en la siguiente figura:

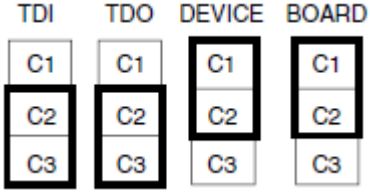


Figura 39. Disposición de puentes para la configuración sólo del dispositivo EPF10K70
(Imagen extraída de [3])

El dispositivo EPF10K70 de la placa UP2 es programado mediante la conexión hardware ByteBlaster II a través del terminal JTAG de la placa.

2.- En el caso de no haber escogido el dispositivo programable correspondiente a la hora de definir el proyecto en Quartus II es necesario acceder a la ventana mostrada en la Figura 40, donde se escoge la familia y dispositivo a programar. Para ello acceder al menú 'Assignments' seguido de la opción 'Device'.

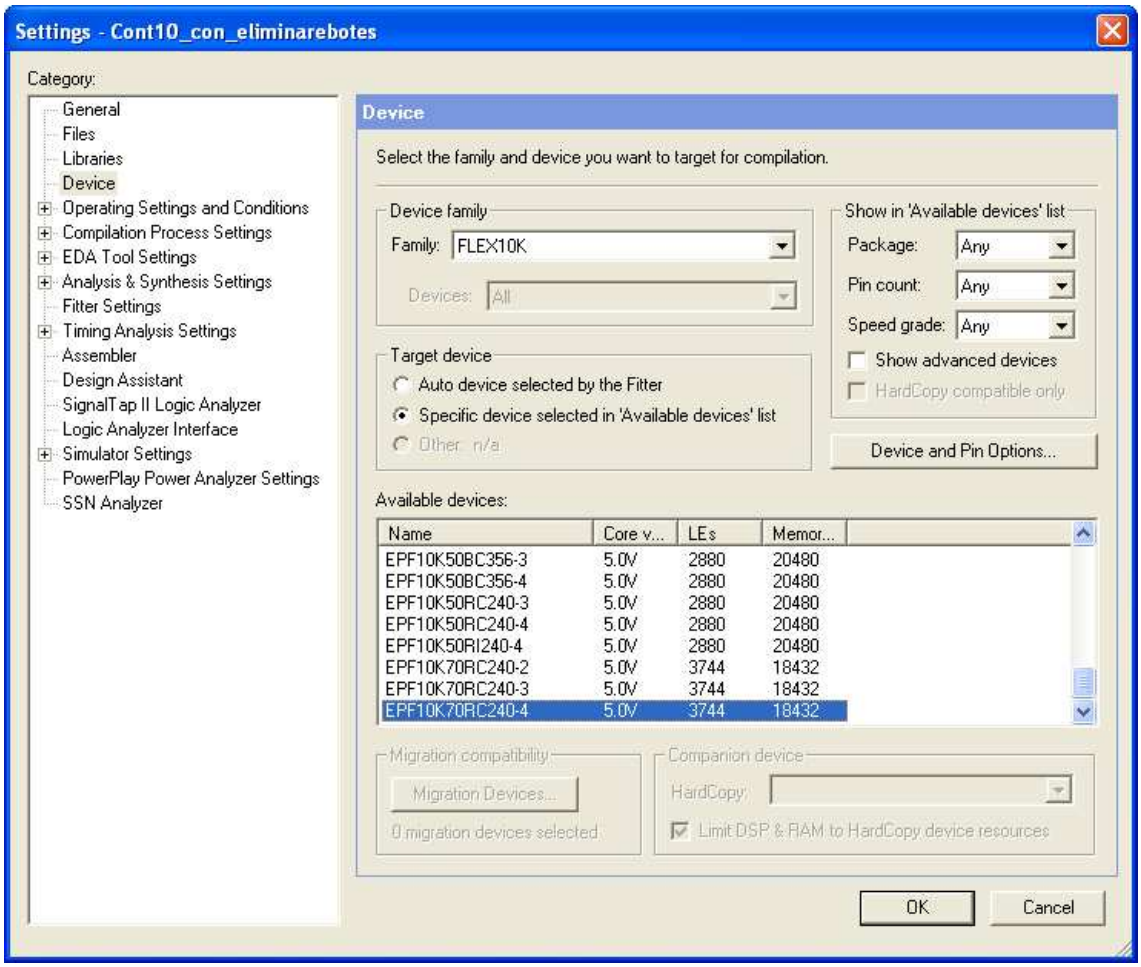


Figura 40. Asignación del dispositivo a programar

En el caso de la tarjeta UP2 se debe escoger la familia de dispositivos FLEX10K, y dentro de ella el dispositivo EPF10K70RC240-4.

Una vez seleccionado el dispositivo se acepta la operación pulsando el botón 'OK' y se compilará el proyecto de nuevo para que éste sea actualizado al nuevo dispositivo.

3.- Mediante la ventana de asignación de pines, se asignan a los puertos de entrada y salida del diseño los pines correspondientes del dispositivo PLD que se está utilizando. El acceso a esta ventana se obtiene desde el menú 'Assignments' seguido de la opción 'Assignment Editor', obteniendo una ventana como la mostrada en la Figura 41.

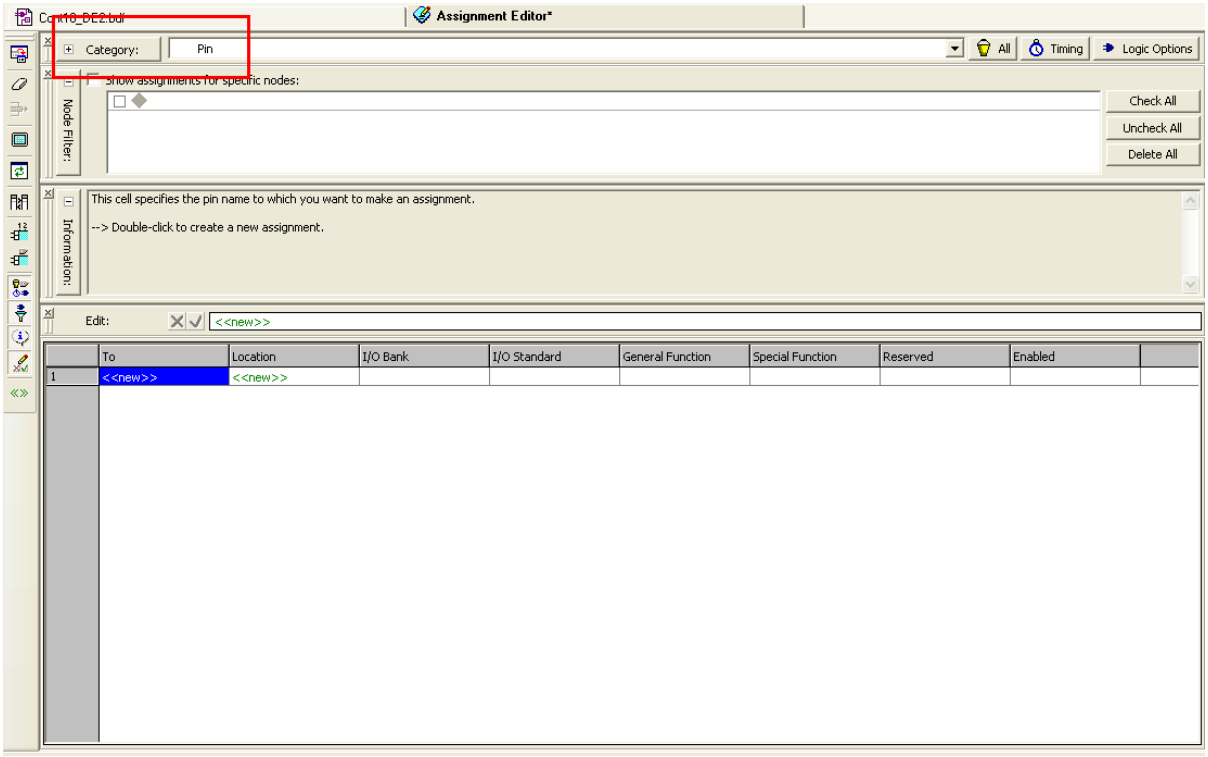


Figura 41. Ventana de la herramienta de asignación de pines

Observando la figura anterior, para comenzar con la asignación de pines, seleccionar bajo el apartado 'categoría', recuadrado en rojo, la opción 'pin'. A continuación hacer doble clic sobre la entrada <<new>> sombreada en color azul, obteniendo un menú desplegable en el que aparecen cada uno de los puertos que se pretenden asignar, tal y como muestra la siguiente figura:

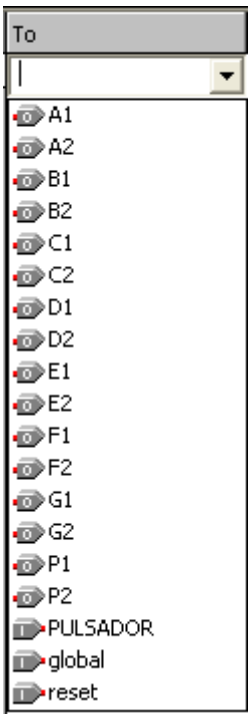


Figura 42. Menú desplegable con los nombres de las entradas y salidas

A partir de este menú, seleccionar el pin que va a ser asignado y, a continuación, acudir a la columna denominada 'Location' haciendo doble clic sobre su correspondiente celda para que aparezca el menú desplegable que permite asignar el pin con las características exigidas por cada entrada. Este menú es mostrado en la Figura 43.

Location	I/O Bank	I/O Standard	General Function	Special Fun
		3.3-V LVTTL		
PIN_M2	I/O Bank 2	Row I/O	LVDS27n	
PIN_M3	I/O Bank 2	Row I/O	LVDS27p, DPCLK0/DQS0L/CQ1L	
PIN_M4	I/O Bank 2	Row I/O	LVDS28p	
PIN_M5	I/O Bank 2	Row I/O	LVDS28n	
PIN_M19	I/O Bank 5	Row I/O	LVDS123p	
PIN_M20	I/O Bank 5	Row I/O	LVDS123n	
PIN_M21	I/O Bank 5	Row I/O	LVDS124n	
PIN_M22	I/O Bank 5	Row I/O	LVDS122p	
PIN_M23	I/O Bank 5	Row I/O	LVDS122n	
PIN_M24	I/O Bank 5	Row I/O	LVDS125p	
PIN_M25	I/O Bank 5	Row I/O	LVDS125n	
PIN_N1	I/O Bank 2	Dedicated Clock	CLK1, LVDSCLK0n, Input	
PIN_N2	I/O Bank 2	Dedicated Clock	CLK0, LVDSCLK0p, Input	
PIN_N9	I/O Bank 2	Row I/O	LVDS31p	
PIN_N18	I/O Bank 5	Row I/O	LVDS110p	
PIN_N20	I/O Bank 5	Row I/O	LVDS124p	
PIN_N23	I/O Bank 5	Row I/O	LVDS126p, DPCLK7/DQS0R/CQ1R	
PIN_N24	I/O Bank 5	Row I/O	LVDS126n	
PIN_N25	I/O Bank 5	Dedicated Clock	CLK4, LVDSCLK2p, Input	
PIN_N26	I/O Bank 5	Dedicated Clock	CLK5, LVDSCLK2n, Input	

Figura 43. Ventana de pines disponibles

De esta manera, consultado el manual de la placa y las tablas de disposición de pines que se exponen en el anexo del presente manual, se irán configurando cada una de las entradas y salidas con su correspondiente pin.

Una vez que se ha terminado con la asignación de pines para este diseño, guardar la operación y volver a compilar para que el dispositivo se quede con su nueva configuración. Además, el usuario puede usar esta misma asignación de pines para posteriores diseños en la que sea requerida. Acudiendo al menú 'File' situado en la barra de herramientas principal mediante la opción 'Export' se crea un archivo con extensión '.csv' capaz de ser leído por el software Microsoft Excel. A continuación, en un nuevo proyecto, es posible importar esta asignación escogiendo el menú 'Assignments' seguido de la opción 'Import Assignments' seleccionando el archivo con extensión '.csv' que se desea importar.

4.- El último paso en el proceso de programación del PLD se realiza mediante la herramienta 'programador' con la que cuenta el software Quartus II.
Para comenzar la programación del dispositivo, acceder bien al menú 'Tools' y seleccionar la opción 'Programmer' o bien acudir directamente a su icono. 

La ventana de esta herramienta es la mostrada en la Figura 44.

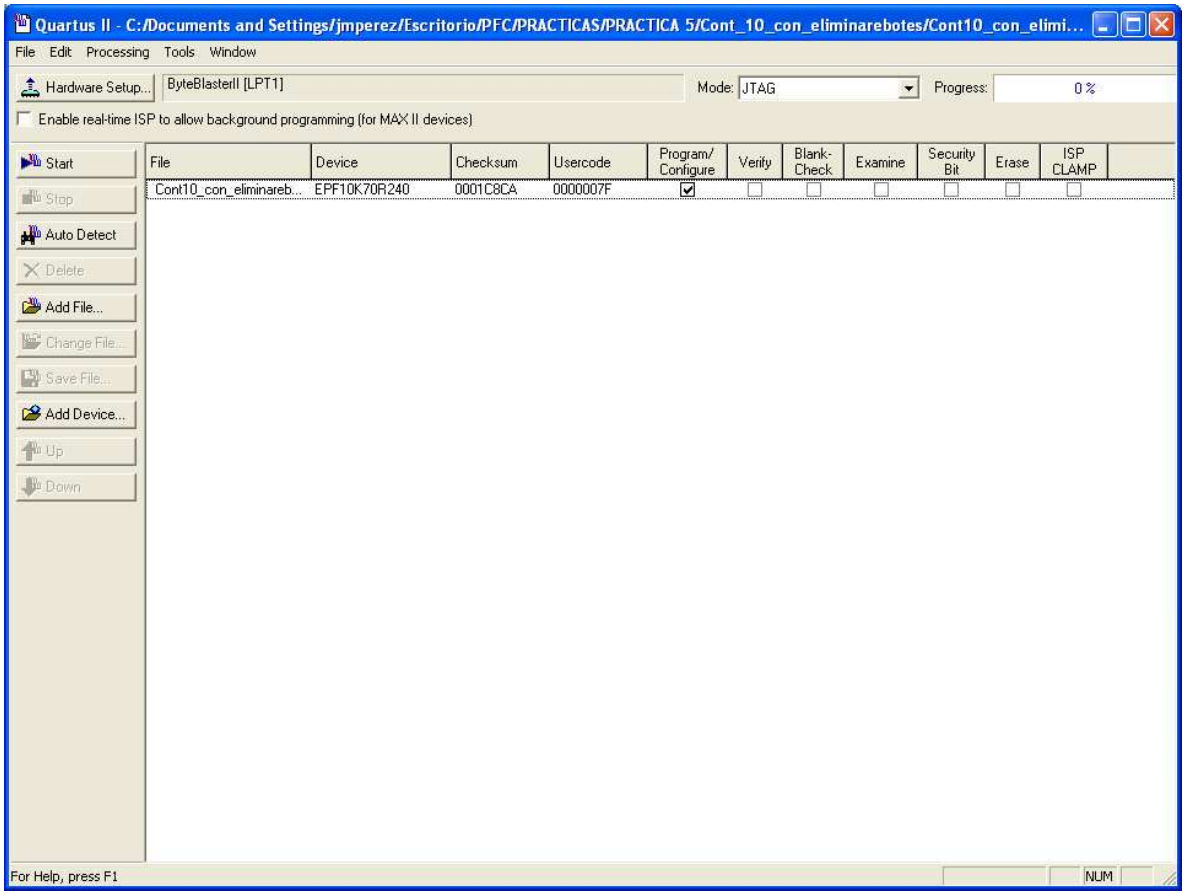


Figura 44. Ventana del programador

En esta ventana es posible asignar el hardware usado para la programación del dispositivo, seleccionar el archivo de diseño que va a ser volcado (con extensión '.sof' en nuestro caso y generado por el compilador) y especificar el propio dispositivo que va a ser programado.

En el caso de que el entorno de Quartus II esté siendo utilizado por primera vez para la programación del dispositivo, es necesario asignar el hardware que permite dicha programación. Para hacer esto, pinchar sobre el botón 'Hardware Setup' situado en la parte superior izquierda de la ventana, y en la nueva ventana que surge añadir el hardware a emplear, que en nuestro caso es el denominado 'ByteBlaster II' a través del puerto LPT1. ByteBlaster II es un sistema programador de dispositivos desarrollado por Altera para dispositivos de Altera, y permite conectar la placa con el puerto paralelo del PC. Por otra parte, el modo de programación es JTAG, que es la forma por la que ByteBlaster se conecta con la placa UP2.

En el caso de que el archivo de programación no aparezca automáticamente en la ventana de configuración o si el usuario quiere cambiar de archivo, se debe acceder a la barra de herramientas asociada al programador y seleccionar el archivo apropiado. De la misma manera, es posible que el dispositivo previsto para la programación no aparezca o no sea el deseado, por lo que se debe recurrir de la misma manera.

Una vez que se ha completado la configuración en la ventana, es necesario seleccionar la opción 'Program/Configure' y pulsar sobre el botón 'Start' que da comienzo a la programación. Cuando la programación está siendo realizada, aparecen los correspondientes mensajes en la ventana de mensajes, así como la barra de progreso, que se encuentra en la parte superior derecha, muestra el estado de la programación.

- Volcado de diseños a la tarjeta educacional DE2

A continuación se presenta la serie de pasos que permiten la configuración y volcado de los respectivos diseños a la tarjeta educacional DE2.

1.- Asignación de pines.

Durante el proceso de compilación, el compilador de Quartus II es libre de elegir cualquier pin de la FPGA seleccionada que le sirvan de entradas y salidas. Sin embargo, la placa DE2 requiere especificar las conexiones entre la FPGA y el resto de componentes de la placa.

Para realizar la especificación de los pines es necesario utilizar la herramienta 'Assignment Editor' situada dentro del menú desplegable 'Assignments' de la barra de herramientas principal. Siguiendo estos pasos se obtiene la ventana que se muestra a continuación:

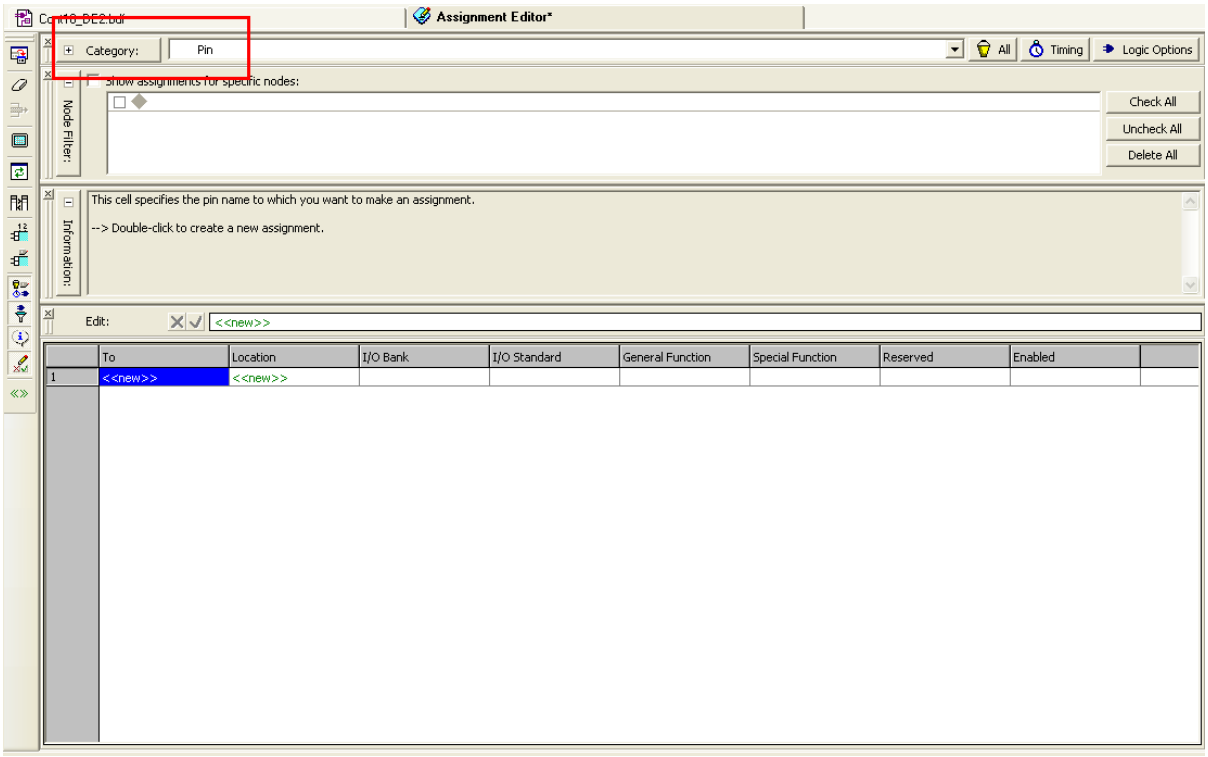


Figura 45. Ventana de la herramienta de asignación de pines

Observando la figura anterior, para comenzar con la asignación de pines, seleccionar bajo el apartado 'categoría', recuadrado en rojo, la opción 'pin'. A continuación hacer doble clic sobre la entrada <<new>> sombreada en color azul, obteniendo un menú desplegable en el que aparecen cada uno de los puertos que se pretenden asignar, tal y como muestra la Figura 46.

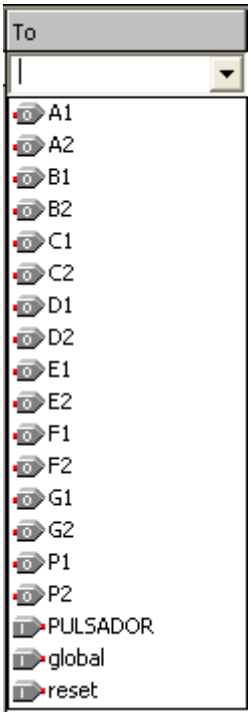


Figura 46. Menú desplegable con los nombres de las entradas y salidas

A partir de este menú, seleccionar el pin que va a ser asignado y, a continuación, acudir a la columna denominada 'Location' haciendo doble clic sobre su correspondiente celda para que aparezca el menú desplegable que permite asignar el pin con las características exigidas por cada entrada. Este menú es mostrado en la Figura 47.

Location	I/O Bank	I/O Standard	General Function	Special Fun
		3.3-V LVTTL		
PIN_M2	I/O Bank 2	Row I/O	LVDS27n	
PIN_M3	I/O Bank 2	Row I/O	LVDS27p, DPCLK0/DQS0L/CQ1L	
PIN_M4	I/O Bank 2	Row I/O	LVDS28p	
PIN_M5	I/O Bank 2	Row I/O	LVDS28n	
PIN_M19	I/O Bank 5	Row I/O	LVDS123p	
PIN_M20	I/O Bank 5	Row I/O	LVDS123n	
PIN_M21	I/O Bank 5	Row I/O	LVDS124n	
PIN_M22	I/O Bank 5	Row I/O	LVDS122p	
PIN_M23	I/O Bank 5	Row I/O	LVDS122n	
PIN_M24	I/O Bank 5	Row I/O	LVDS125p	
PIN_M25	I/O Bank 5	Row I/O	LVDS125n	
PIN_N1	I/O Bank 2	Dedicated Clock	CLK1, LVDSCLK0n, Input	
PIN_N2	I/O Bank 2	Dedicated Clock	CLK0, LVDSCLK0p, Input	
PIN_N9	I/O Bank 2	Row I/O	LVDS31p	
PIN_N18	I/O Bank 5	Row I/O	LVDS110p	
PIN_N20	I/O Bank 5	Row I/O	LVDS124p	
PIN_N23	I/O Bank 5	Row I/O	LVDS126p, DPCLK7/DQS0R/CQ1R	
PIN_N24	I/O Bank 5	Row I/O	LVDS126n	
PIN_N25	I/O Bank 5	Dedicated Clock	CLK4, LVDSCLK2p, Input	
PIN_N26	I/O Bank 5	Dedicated Clock	CLK5, LVDSCLK2n, Input	

Figura 47. Ventana de pines disponibles

De esta manera, consultado el manual de la placa y las tablas de disposición de pines que se exponen en el anexo del presente manual, se irán configurando cada una de las entradas y salidas con su correspondiente pin.

Una vez que se ha terminado con la asignación de pines para este diseño, guardar la operación y volver a compilar para que el dispositivo se quede con su nueva configuración. Además, el usuario puede usar esta misma asignación de pines para posteriores diseños en la que sea requerida. Acudiendo al menú 'File' situado en la barra de herramientas principal mediante la opción 'Export' se crea un archivo con extensión '.csv' capaz de ser leído por el software Microsoft Excel. A continuación, en un nuevo proyecto, es posible importar esta asignación escogiendo el menú 'Assignments' seguido de la opción 'Import Assignments' seleccionando el archivo con extensión '.csv' que se desea importar.

2.- Programación:

El dispositivo FPGA debe ser programado y configurado si sobre él se quiere implementar el circuito diseñado. El archivo de configuración requerido para la operación ha sido creado previamente en el proceso de compilación.

Para los diseños que requieran ser volcados en la realización de las prácticas se va a seguir el modo de programación JTAG (ver capítulo 4 'Manual de las tarjetas UP2 y DE2' para mayor información), por lo que se deben seguir los pasos descritos a continuación.

- Modo de programación JTAG

En primer lugar, la placa DE2 cuenta con un conmutador de dos posiciones RUN/PROG, el cual ha de ser situado en la posición RUN.

A continuación, en el software Quartus II, acceder al menú 'Tools' situado en la barra de herramientas principal y seleccionar la opción 'Programmer'. También es posible acceder directamente a su icono.

Siguiendo estos pasos, se obtiene la ventana mostrada en la Figura 48.

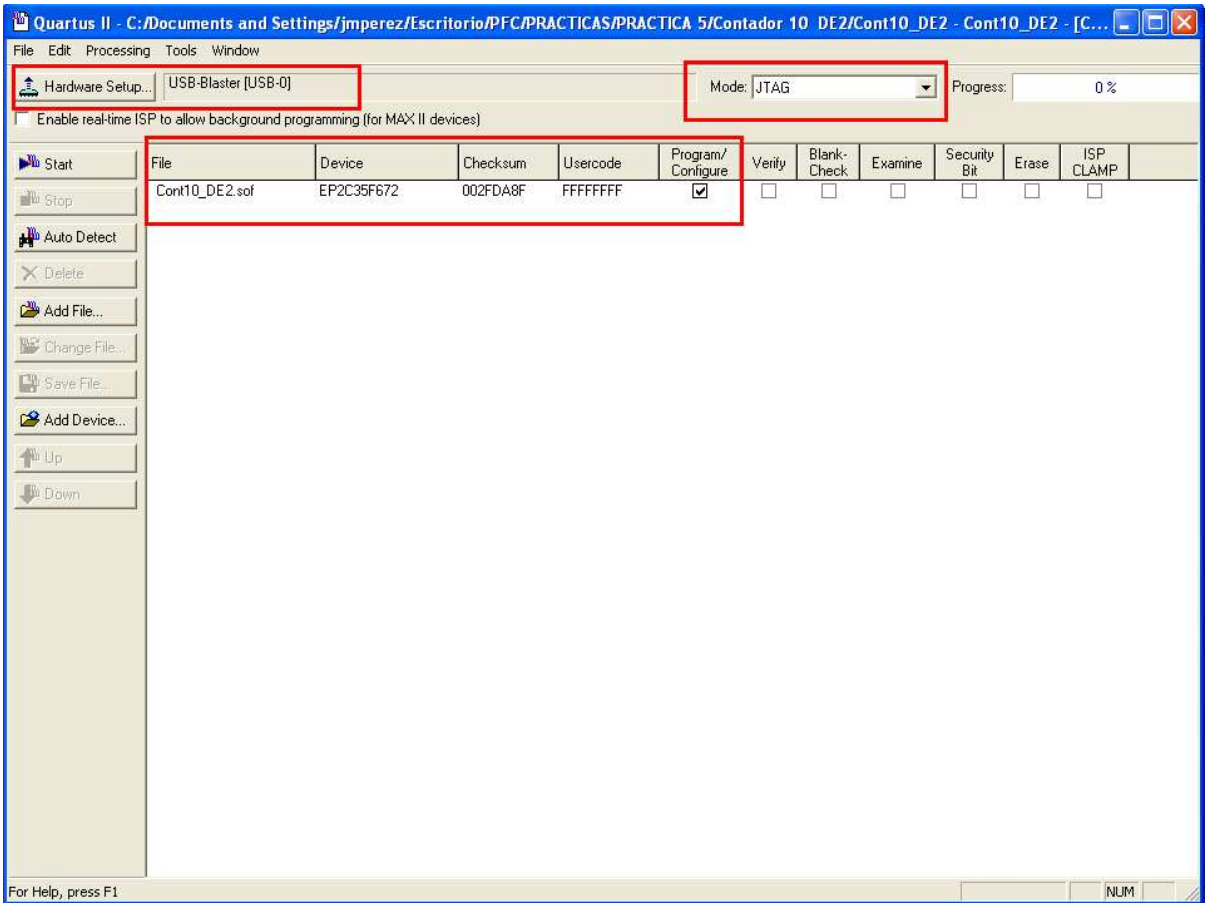


Figura 48. Ventana del Programador

En esta ventana es necesario especificar tanto el hardware como el modo de programación que va a ser utilizado. Si estos parámetros no vienen ya dados por defecto, seleccionar JTAG en la casilla 'Mode'. Asimismo, si el hardware USB-Blaster no ha sido el elegido, pulsar sobre el botón 'Hardware Setup' situado en la parte superior izquierda de la ventana y seleccionar la opción USB-Blaster en la nueva ventana, mostrada en la Figura 49.

Además, se observa en la Figura 48, cómo el archivo de programación, con extensión '.sof', también ha sido cargado por defecto. En caso de no ser así, pulsar sobre el botón 'Add Files' de la barra de herramientas adjunta al programador. Éste es el archivo de configuración creado por el compilador y que incluye toda la información necesaria para la programación del dispositivo FPGA. Comprobar también que el dispositivo que aparece en la misma línea es el EP2C35F672, que es el correspondiente a la placa DE2. Si todo es correcto, hacer clic sobre la casilla de la opción 'Program/Configure' y presionar el botón 'Start' de la barra de herramientas. Cuando la programación está siendo realizada, aparecen los correspondientes mensajes en la ventana de mensajes, así como la barra de progreso, que se encuentra en la parte superior derecha, muestra el estado de la programación. Un diodo LED de la placa se ilumina cuando la configuración del dispositivo ha concluido satisfactoriamente.

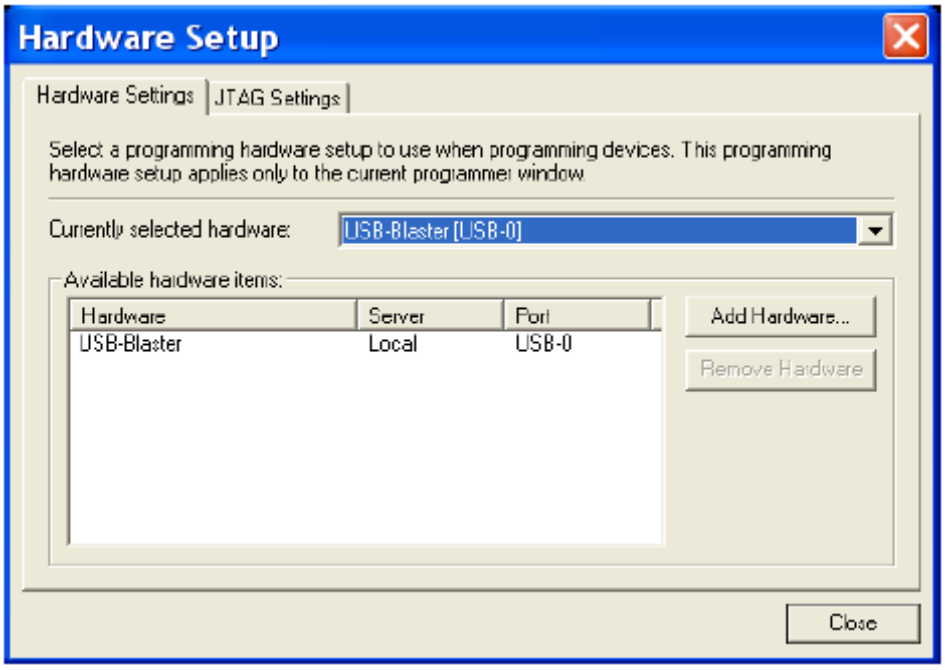


Figura 49. Ventana de selección de hardware

6.13.- EDITOR FLOORPLAN

Este editor permite asignar elementos lógicos a los pines físicos del dispositivo y a celdas lógicas de manera gráfica. Además, se puede ver cómo queda la programación del chip (i.e. entradas, salidas, reservadas, etc). La ventana del Editor Floorplan es la mostrada en la Figura 50.

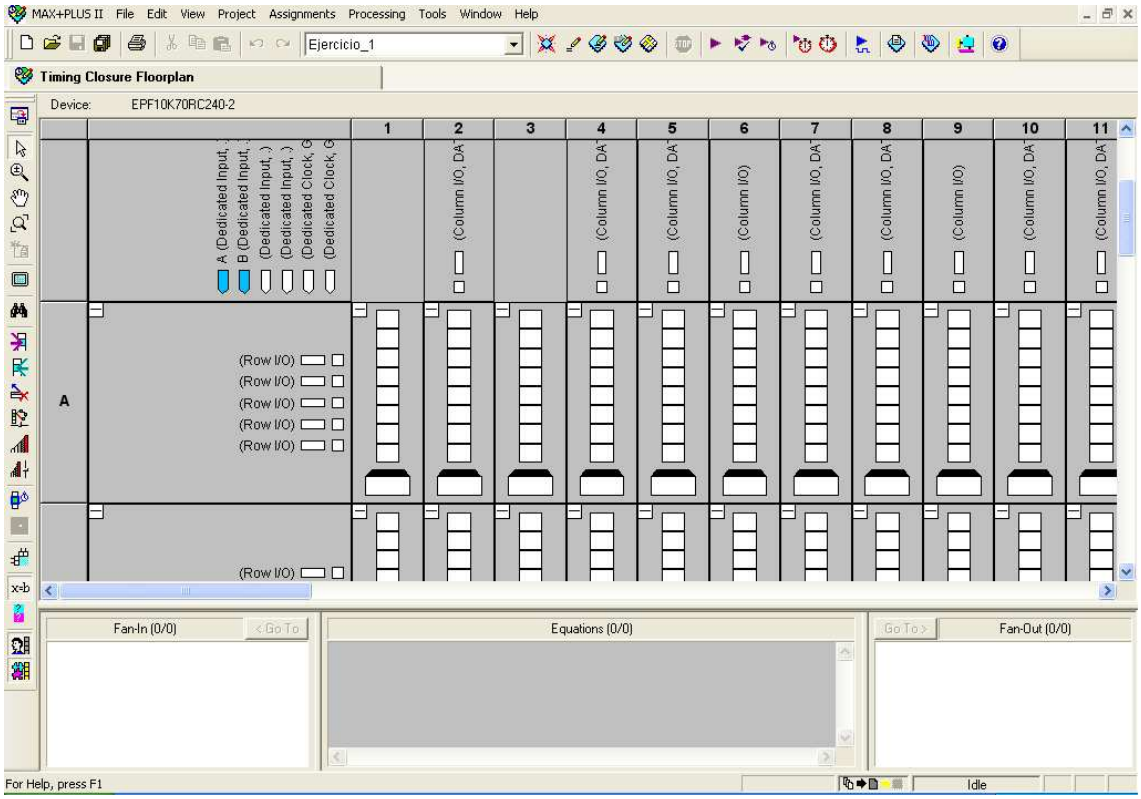


Figura 50. Ventana del Editor Floorplan

6.14.- GLOSARIO

- **Archivo de diseño:** Archivo que contiene el diseño lógico de un proyecto de Quartus II. A continuación se enumeran los diferentes archivos de diseño:
  - AHDL File (.tdf)
  - Block Diagram / Schematic File (.bdf)
  - EDIF File (.edf)
  - State Machine File (.smf)
  - System Verilog HDL File (.sv)
  - Tcl Script File (.tcl)
  - Verilog HDL File (.v)
  - VHDL File (.vhd)

- **Bit:** Acrónimo de ***Bi**nary dig**it***. (dígito binario). Un bit o dígito binario puede representar uno de esos dos valores, **0** ó **1**.

- **Bus:** Línea capaz de transportar múltiples señales entre los componentes de un diseño.

- **ByteBlaster:** Cable de descarga paralelo que permite al usuario programar y configurar los dispositivos de la placa UP2.

- **Chip:** Grupo de funciones lógicas definidas como una unidad. La asignación de un chip a un determinado dispositivo puede ser realizada por el usuario o por la herramienta de compilación.

- **Compilador:** Herramienta que se encarga de la detección de posibles errores existentes a la hora de construir el diseño gráficamente o en la sintaxis del mismo. Genera un archivo '.qdf' que contiene toda la configuración del dispositivo.

- **CPLD:** Acrónimo inglés ***C**omplex **P**rogrammable **L**ogic **D**evice*. Es un dispositivo electrónico programable, formado por múltiples bloques lógicos.

- **Editor Gráfico:** Herramienta que permite el diseño de circuitos esquemáticamente. Genera un archivo con extensión '.bdf'.

- **Editor de Texto:** Herramienta que permite la creación de diseños según los distintos lenguajes de programación con los que cuenta el software.

- **EEPROM:** Acrónimo de ***E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory*. Se trata de un tipo de memoria reprogramable en la cual los contenidos se pueden borrar sometiendo al dispositivo a las señales eléctricas apropiadas para ello.

- **Elemento Lógico (LE):** Es el bloque básico en la estructura de cualquier dispositivo.

- **Estructura jerárquica:** Es el orden de archivos llevado en un proyecto, según su importancia. Es importante en un proyecto de Quartus II conocer el archivo de mayor jerarquía, ya que los demás archivos estarán contenidos en él.

- **Driver:** Su traducción se corresponde con 'Controlador de dispositivo'. Es un programa informático que permite al sistema operativo interactuar con un periférico.

- **FPGA:** Acrónimo inglés ***F**ield **P**rogrammable **G**ate **A**rray*. Es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar.

- **Jumper:** Es un elemento para interconectar dos terminales de manera temporal.

- **Nodo:** Un nodo representa un cable que conduce una señal que se desplaza entre diferentes componentes lógicos de un diseño. En el Editor Gráfico, los nodos se representan por medio de líneas finas; en archivos de texto se tratan de nombres simbólicos y en el Editor de Señales son formas de onda.

- **Número de pin:** Número utilizado para asignar una entrada o una salida en un archivo de diseño, el cual se corresponde con un número de pin en un dispositivo físico.

- **Pin:** Es una entrada o salida de cualquier dispositivo.

- **PLL:** Acrónimo inglés ***P**hase-**L**ocked **L**oops*. Se trata de un sistema realimentado, en el que las magnitudes realimentadas son la frecuencia y la fase.

- **Programador:** Herramienta de Quartus II que permite la configuración de un dispositivo en base al diseño realizado en el software.

- **Programmer Object File (.pof):** Archivo binario generado por el módulo Ensamblador de la herramienta de compilación. Este archivo contiene la información necesaria para la configuración del dispositivo de la familia MAX 7000 de la placa UP2 de Altera®.

- **Proyecto:** Un proyecto consiste en todos los archivos que están asociados a un diseño en particular, incluyendo todos los subarchivos correspondientes a otros diseños que se necesite incluir. El nombre del proyecto es el mismo que el dado al archivo de diseño más importante jerárquicamente. En Quartus II un archivo de proyecto cuenta con extensión 'qpf'.

- **Programación AS:** (**A**ctive **S**erial), modo de programación donde los datos de configuración son guardados en un bloque de memoria flash, por lo que dichos datos no se pierden en caso de desconectar la alimentación

- **Programación JTAG:** (**J**oint **T**est **A**ction **G**roup), los datos de configuración son enviados directamente al dispositivo FPGA. Al configurar el dispositivo de este modo, éste retiene la configuración indicada mientras no se corte la alimentación.

- **RAM: R**andom-**a**cces **m**emory o memoria de acceso aleatorio. Es el área de trabajo para la mayor parte del software de un PC.

- **Señales Lógicas:** Una señal lógica o binaria es aquella que únicamente puede tomar dos valores fijos.

- **SRAM Object File (.sof):** Archivo binario generado por el compilador, el cual contiene toda la información necesaria para configurar los dispositivos de las familias FLEX 10K (Placa UP2) y Cyclone II (Placa DE2).

- **Symbol File (.bsf):** Archivo gráfico creado por el Editor de Símbolos. Un símbolo representa un diseño con el mismo nombre del proyecto y puede ser usado por medio del Editor Gráfico en otros diseños.

- **USB Blaster:** Circuito contenido en la placa DE2 para la programación de la misma a través de un cable USB conectado al PC.

- **VHDL:** acrónimo que representa la combinación de VHSIC y HDL, donde VHSIC es el acrónimo de ***V**ery **H**igh **S**peed **I**ntegrated **C**ircuit* y HDL es a su vez el acrónimo de ***H**ardware **D**escription **L**anguage*. Es un lenguaje de programación usado para diseñar circuitos digitales.

- **Vector Waveform File (.vwf):** Archivo de diseño gráfico de señales creado mediante el Editor de Señales de Quartus II.

ANEXO 1. CONEXIÓN DE PINES DEL DISPOSITIVO EPF10K70

Los pines del dispositivo EPF10K70 de la familia FLEX 10K, soportado en la tarjeta educacional UP2, están preasignados a los interruptores y displays de la tarjeta. A continuación se presentan dichas asignaciones, identificando cada elemento en su correspondiente tabla.

- *Pulsadores FLEX-PB1 y FLEX-PB2*

Estos pulsadores trabajan con lógica negativa, es decir, darán un cero lógico cuando sean presionados.

Pulsador	Pin FLEX10K
FLEX_PB1	28
FLEX_PB2	29

Tabla 1. Asignación de pines de los pulsadores.

- *Interruptores del dispositivo*

Tener en cuenta que un pin de entrada se pone a uno lógico cuando el interruptor está abierto y a cero lógico cuando el interruptor está cerrado.

Interruptor	Pin FLEX10K
Flex_Int_1	41
Flex_Int_1	40
Flex_Int_1	39
Flex_Int_1	38
Flex_Int_1	36
Flex_Int_1	35
Flex_Int_1	34
Flex_Int_1	33

Tabla 2. Asignación de pines de los interruptores.

- *Displays de siete segmentos*

La figura mostrada a continuación permite identificar los nombres de los distintos segmentos que componen los displays.

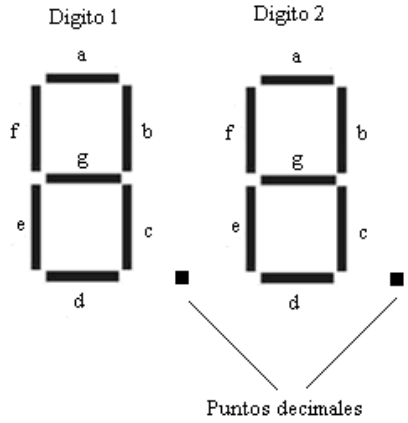


Figura 1. Displays 7-segmentos
(Imagen extraída de [3])

Es importante tener en cuenta las señales que controlan los displays, puesto que los diodos que los forman se activan con un cero lógico.

Los dos displays correspondientes al dispositivo FLEX10K70 se encuentran directamente conectados al mismo. La siguiente tabla muestra la asignación de los correspondientes pines.

Segmento	Pin Display 1	Pin Display 2
a	6	17
b	7	18
c	8	19
d	9	20
e	11	21
f	12	23
g	13	24
Punto decimal	14	25

Tabla 3. Asignación de pines de los displays

- *Interfaz VGA*

Señal	Pin Puerto VGA	Pin FLEX10K
Red	1	236
Green	2	237
Blue	3	238
GND	6,7,8,10,11	-
Sinc. Horizontal	13	240
Sinc. Vertical	14	239
Sin conexión	4,5,9,15	-

Tabla 4. Asignación de pines del interfaz VGA

- *Conexión PS/2*

Señal	Pin PS/2	Pin FLEX10K
CLICK	1	30
DATOS	3	31
VCC	5	-
GND	2	-

Tabla 5. Asignación de pines de la conexión PS/2

- Entradas/Salidas externas del dispositivo FLEX10K70

Nº Agujero	Señal/Pin	Nº Agujero	Señal/Pin
1	RAW	2	GND
3	VCC	4	GND
5	VCC	6	GND
7	Sin conexión	8	D11/90
9	D12/92	10	D13/210
11	D14/212	12	DEV_CLR/209
13	DEV_OE/213	14	DEV_CLK2/211
15	45	16	46
17	48	18	49
19	50	20	51
21	53	22	54
23	55	24	56
25	61	26	62
27	63	28	64
29	65	30	66
31	67	32	68
33	70	34	71
35	72	36	73
37	74	38	75
39	76	40	78
41	79	42	80
43	81	44	82
45	83	46	84
47	86	48	87
49	88	50	94
51	95	52	97
53	98	54	99
55	100	56	101
57	VCC	58	GND
59	VCC	60	GND

Tabla 6. Asignación de pines para el puerto de expansión 'A'

Nº Agujero	Señal/Pin	Nº Agujero	Señal/Pin
1	RAW	2	GND
3	VCC	4	GND
5	VCC	6	GND
7	Sin conexión	8	D11/90
9	D12/92	10	D13/210
11	D14/212	12	DEV_CLR/209
13	DEV_OE/213	14	DEV_CLK2/211
15	109	16	110
17	111	18	113
19	114	20	115
21	116	22	117
23	118	24	119
25	120	26	126
27	127	28	128
29	129	30	131
31	132	32	133
33	134	34	136
35	137	36	138
37	139	38	141
39	142	40	143
41	144	42	146
43	147	44	148
45	149	46	151
47	152	48	153
49	154	50	156
51	157	52	158
53	159	54	161
55	162	56	163
57	VCC	58	GND
59	VCC	60	GND

Tabla 7. Asignación de pines para el puerto de expansión 'B'

Nº Agujero	Señal/Pin	Nº Agujero	Señal/Pin
1	RAW	2	GND
3	VCC	4	GND
5	VCC	6	GND
7	Sin conexión	8	D11/90
9	D12/92	10	D13/210
11	D14/212	12	DEV_CLR/209
13	DEV_OE/213	14	DEV_CLK2/211
15	175	16	181
17	182	18	183
19	184	20	185
21	186	22	187
23	188	24	190
25	191	26	192
27	193	28	194
29	195	30	196
31	198	32	199
33	200	34	201
35	202	36	203
37	204	38	206
39	207	40	208
41	214	42	215
43	217	44	218
45	219	46	220
47	221	48	222
49	223	50	225
51	226	52	227
53	228	54	229
55	230	56	231
57	VCC	58	GND
59	VCC	60	GND

Tabla 8. Asignación de pines para el puerto de expansión 'C'

ANEXO 2. CONEXIÓN DE PINES DEL DISPOSITIVO FPGA 2C35

Los pines del dispositivo Cyclone II FPGA 2C35 de la tarjeta educacional DE2 están preasignados a los diferentes elementos que componen la misma (interruptores, displays, etc). A continuación se presentan las tablas de dichas asignaciones.

- *Interruptores y Pulsadores*

La placa DE2 consta de 4 pulsadores 1 18 interruptores. Tener en cuenta que un pin de entrada se pone a nivel alto ('1' lógico) cuando el pulsador o interruptor está abierto y a nivel bajo ('0' lógico) cuando los mismos están cerrados.

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_N25	Toggle Switch[0]
SW[1]	PIN_N26	Toggle Switch[1]
SW[2]	PIN_P25	Toggle Switch[2]
SW[3]	PIN_AE14	Toggle Switch[3]
SW[4]	PIN_AF14	Toggle Switch[4]
SW[5]	PIN_AD13	Toggle Switch[5]
SW[6]	PIN_AC13	Toggle Switch[6]
SW[7]	PIN_C13	Toggle Switch[7]
SW[8]	PIN_B13	Toggle Switch[8]
SW[9]	PIN_A13	Toggle Switch[9]
SW[10]	PIN_N1	Toggle Switch[10]
SW[11]	PIN_P1	Toggle Switch[11]
SW[12]	PIN_P2	Toggle Switch[12]
SW[13]	PIN_T7	Toggle Switch[13]
SW[14]	PIN_U3	Toggle Switch[14]
SW[15]	PIN_U4	Toggle Switch[15]
SW[16]	PIN_V1	Toggle Switch[16]
SW[17]	PIN_V2	Toggle Switch[17]

Tabla 1. Asignación de pines de los interruptores

Signal Name	FPGA Pin No.	Description
KEY[0]	PIN_G26	Pushbutton[0]
KEY[1]	PIN_N23	Pushbutton[1]
KEY[2]	PIN_P23	Pushbutton[2]
KEY[3]	PIN_W26	Pushbutton[3]

Tabla 2. Asignación de pines de los pulsadores

- *Diodos LED*

Existen 27 diodos LED controlables por el usuario sobre la placa DE2, de los cuales 18 son de color rojo y 9 de color verde.

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_AE23	LED Red[0]
LEDR[1]	PIN_AF23	LED Red[1]
LEDR[2]	PIN_AB21	LED Red[2]
LEDR[3]	PIN_AC22	LED Red[3]
LEDR[4]	PIN_AD22	LED Red[4]
LEDR[5]	PIN_AD23	LED Red[5]
LEDR[6]	PIN_AD21	LED Red[6]
LEDR[7]	PIN_AC21	LED Red[7]
LEDR[8]	PIN_AA14	LED Red[8]
LEDR[9]	PIN_Y13	LED Red[9]
LEDR[10]	PIN_AA13	LED Red[10]
LEDR[11]	PIN_AC14	LED Red[11]
LEDR[12]	PIN_AD15	LED Red[12]
LEDR[13]	PIN_AE15	LED Red[13]
LEDR[14]	PIN_AF13	LED Red[14]
LEDR[15]	PIN_AE13	LED Red[15]
LEDR[16]	PIN_AE12	LED Red[16]
LEDR[17]	PIN_AD12	LED Red[17]
LEDG[0]	PIN_AE22	LED Green[0]
LEDG[1]	PIN_AF22	LED Green[1]
LEDG[2]	PIN_W19	LED Green[2]
LEDG[3]	PIN_V18	LED Green[3]
LEDG[4]	PIN_U18	LED Green[4]
LEDG[5]	PIN_U17	LED Green[5]
LEDG[6]	PIN_AA20	LED Green[6]
LEDG[7]	PIN_Y18	LED Green[7]
LEDG[8]	PIN_Y12	LED Green[8]

Tabla 3. Asignación de pines para diodos LED

- Displays 7-segmentos

La placa DE2 provee de 8 displays de 7 segmentos. Estos displays están dispuestos en dos parejas y un grupo de cuatro, con el fin de poder conseguir cifras de diferentes tamaños. La siguiente figura muestra la disposición y nombres de los diodos que componen un display de 7 segmentos:

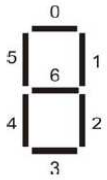


Figura 1. Display 7 segmentos

Signal Name	FPGA Pin No.	Description	Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_AF10	Seven Segment Digit 0[0]	HEX4[0]	PIN_U9	Seven Segment Digit 4[0]
HEX0[1]	PIN_AB12	Seven Segment Digit 0[1]	HEX4[1]	PIN_U1	Seven Segment Digit 4[1]
HEX0[2]	PIN_AC12	Seven Segment Digit 0[2]	HEX4[2]	PIN_U2	Seven Segment Digit 4[2]
HEX0[3]	PIN_AD11	Seven Segment Digit 0[3]	HEX4[3]	PIN_T4	Seven Segment Digit 4[3]
HEX0[4]	PIN_AE11	Seven Segment Digit 0[4]	HEX4[4]	PIN_R7	Seven Segment Digit 4[4]
HEX0[5]	PIN_V14	Seven Segment Digit 0[5]	HEX4[5]	PIN_R6	Seven Segment Digit 4[5]
HEX0[6]	PIN_V13	Seven Segment Digit 0[6]	HEX4[6]	PIN_T3	Seven Segment Digit 4[6]
HEX1[0]	PIN_V20	Seven Segment Digit 1[0]	HEX5[0]	PIN_T2	Seven Segment Digit 5[0]
HEX1[1]	PIN_V21	Seven Segment Digit 1[1]	HEX5[1]	PIN_P6	Seven Segment Digit 5[1]
HEX1[2]	PIN_W21	Seven Segment Digit 1[2]	HEX5[2]	PIN_P7	Seven Segment Digit 5[2]
HEX1[3]	PIN_Y22	Seven Segment Digit 1[3]	HEX5[3]	PIN_T9	Seven Segment Digit 5[3]
HEX1[4]	PIN_AA24	Seven Segment Digit 1[4]	HEX5[4]	PIN_R5	Seven Segment Digit 5[4]
HEX1[5]	PIN_AA23	Seven Segment Digit 1[5]	HEX5[5]	PIN_R4	Seven Segment Digit 5[5]
HEX1[6]	PIN_AB24	Seven Segment Digit 1[6]	HEX5[6]	PIN_R3	Seven Segment Digit 5[6]
HEX2[0]	PIN_AB23	Seven Segment Digit 2[0]	HEX6[0]	PIN_R2	Seven Segment Digit 6[0]
HEX2[1]	PIN_V22	Seven Segment Digit 2[1]	HEX6[1]	PIN_P4	Seven Segment Digit 6[1]
HEX2[2]	PIN_AC25	Seven Segment Digit 2[2]	HEX6[2]	PIN_P3	Seven Segment Digit 6[2]
HEX2[3]	PIN_AC26	Seven Segment Digit 2[3]	HEX6[3]	PIN_M2	Seven Segment Digit 6[3]
HEX2[4]	PIN_AB26	Seven Segment Digit 2[4]	HEX6[4]	PIN_M3	Seven Segment Digit 6[4]
HEX2[5]	PIN_AB25	Seven Segment Digit 2[5]	HEX6[5]	PIN_M5	Seven Segment Digit 6[5]
HEX2[6]	PIN_Y24	Seven Segment Digit 2[6]	HEX6[6]	PIN_M4	Seven Segment Digit 6[6]
HEX3[0]	PIN_Y23	Seven Segment Digit 3[0]	HEX7[0]	PIN_L3	Seven Segment Digit 7[0]
HEX3[1]	PIN_AA25	Seven Segment Digit 3[1]	HEX7[1]	PIN_L2	Seven Segment Digit 7[1]
HEX3[2]	PIN_AA26	Seven Segment Digit 3[2]	HEX7[2]	PIN_L9	Seven Segment Digit 7[2]
HEX3[3]	PIN_Y26	Seven Segment Digit 3[3]	HEX7[3]	PIN_L6	Seven Segment Digit 7[3]
HEX3[4]	PIN_Y25	Seven Segment Digit 3[4]	HEX7[4]	PIN_L7	Seven Segment Digit 7[4]
HEX3[5]	PIN_U22	Seven Segment Digit 3[5]	HEX7[5]	PIN_P9	Seven Segment Digit 7[5]
HEX3[6]	PIN_W24	Seven Segment Digit 3[6]	HEX7[6]	PIN_N9	Seven Segment Digit 7[6]

Tabla 4. Disposición de pines de los Displays de 7 segmentos

- *Entradas de reloj:*

La placa DE2 incluye dos osciladores que generan señales de reloj de 27 MHz y 50 MHz. Además, la placa incluye un conector SMA que puede ser usado para conectar una señal de reloj externa a la misma.

Signal Name	FPGA Pin No.	Description
CLOCK_27	PIN_D13	27 MHz clock input
CLOCK_50	PIN_N2	50 MHz clock input
EXT_CLOCK	PIN_P26	External (SMA) clock input

Tabla 5. Asignación de pines para las entradas de reloj

- *Módulo LCD:*

El módulo LCD puede ser usado para mostrar un determinado texto que sea generado en el diseño.

Signal Name	FPGA Pin No.	Description
LCD_DATA[0]	PIN_J1	LCD Data[0]
LCD_DATA[1]	PIN_J2	LCD Data[1]
LCD_DATA[2]	PIN_H1	LCD Data[2]
LCD_DATA[3]	PIN_H2	LCD Data[3]
LCD_DATA[4]	PIN_J4	LCD Data[4]
LCD_DATA[5]	PIN_J3	LCD Data[5]
LCD_DATA[6]	PIN_H4	LCD Data[6]
LCD_DATA[7]	PIN_H3	LCD Data[7]
LCD_RW	PIN_K4	LCD Read/Write Select, 0 = Write, 1 = Read
LCD_EN	PIN_K3	LCD Enable
LCD_RS	PIN_K1	LCD Command/Data Select, 0 = Command, 1 = Data
LCD_ON	PIN_L4	LCD Power ON/OFF
LCD_BLON	PIN_K2	LCD Back Light ON/OFF

Tabla 6. Asignación de pines del módulo LCD.

- *Conectores de expansión*

La placa DE2 contiene dos conectores de expansión de 40 pines cada uno, de los cuales, en cada conector, 36 de estos pines están directamente conectados al dispositivo Cyclone II FPGA.

Signal Name	FPGA Pin No.	Description	Signal Name	FPGA Pin No.	Description
GPIO_0[0]	PIN_D25	GPIO Connection 0[0]	GPIO_1[0]	PIN_K25	GPIO Connection 1[0]
GPIO_0[1]	PIN_J22	GPIO Connection 0[1]	GPIO_1[1]	PIN_K23	GPIO Connection 1[1]
GPIO_0[2]	PIN_E26	GPIO Connection 0[2]	GPIO_1[2]	PIN_M22	GPIO Connection 1[2]
GPIO_0[3]	PIN_E25	GPIO Connection 0[3]	GPIO_1[3]	PIN_M23	GPIO Connection 1[3]
GPIO_0[4]	PIN_F24	GPIO Connection 0[4]	GPIO_1[4]	PIN_M19	GPIO Connection 1[4]
GPIO_0[5]	PIN_F23	GPIO Connection 0[5]	GPIO_1[5]	PIN_M20	GPIO Connection 1[5]
GPIO_0[6]	PIN_J21	GPIO Connection 0[6]	GPIO_1[6]	PIN_N20	GPIO Connection 1[6]
GPIO_0[7]	PIN_J20	GPIO Connection 0[7]	GPIO_1[7]	PIN_M21	GPIO Connection 1[7]
GPIO_0[8]	PIN_F25	GPIO Connection 0[8]	GPIO_1[8]	PIN_M24	GPIO Connection 1[8]
GPIO_0[9]	PIN_F26	GPIO Connection 0[9]	GPIO_1[9]	PIN_M25	GPIO Connection 1[9]
GPIO_0[10]	PIN_N18	GPIO Connection 0[10]	GPIO_1[11]	PIN_N24	GPIO Connection 1[11]
GPIO_0[11]	PIN_P18	GPIO Connection 0[11]	GPIO_1[11]	PIN_P24	GPIO Connection 1[11]
GPIO_0[12]	PIN_G23	GPIO Connection 0[12]	GPIO_1[12]	PIN_R25	GPIO Connection 1[12]
GPIO_0[13]	PIN_G24	GPIO Connection 0[13]	GPIO_1[13]	PIN_R24	GPIO Connection 1[13]
GPIO_0[14]	PIN_K22	GPIO Connection 0[14]	GPIO_1[14]	PIN_R20	GPIO Connection 1[14]
GPIO_0[15]	PIN_G25	GPIO Connection 0[15]	GPIO_1[15]	PIN_T22	GPIO Connection 1[15]
GPIO_0[16]	PIN_H23	GPIO Connection 0[16]	GPIO_1[16]	PIN_T23	GPIO Connection 1[16]
GPIO_0[17]	PIN_H24	GPIO Connection 0[17]	GPIO_1[17]	PIN_T24	GPIO Connection 1[17]
GPIO_0[18]	PIN_J23	GPIO Connection 0[18]	GPIO_1[18]	PIN_T25	GPIO Connection 1[18]
GPIO_0[19]	PIN_J24	GPIO Connection 0[19]	GPIO_1[19]	PIN_T18	GPIO Connection 1[19]
GPIO_0[20]	PIN_H25	GPIO Connection 0[20]	GPIO_1[21]	PIN_T21	GPIO Connection 1[21]
GPIO_0[21]	PIN_H26	GPIO Connection 0[21]	GPIO_1[21]	PIN_T20	GPIO Connection 1[21]
GPIO_0[22]	PIN_H19	GPIO Connection 0[22]	GPIO_1[22]	PIN_U26	GPIO Connection 1[22]
GPIO_0[23]	PIN_K18	GPIO Connection 0[23]	GPIO_1[23]	PIN_U25	GPIO Connection 1[23]
GPIO_0[24]	PIN_K19	GPIO Connection 0[24]	GPIO_1[24]	PIN_U23	GPIO Connection 1[24]
GPIO_0[25]	PIN_K21	GPIO Connection 0[25]	GPIO_1[25]	PIN_U24	GPIO Connection 1[25]
GPIO_0[26]	PIN_K23	GPIO Connection 0[26]	GPIO_1[26]	PIN_R19	GPIO Connection 1[26]
GPIO_0[27]	PIN_K24	GPIO Connection 0[27]	GPIO_1[27]	PIN_T19	GPIO Connection 1[27]
GPIO_0[28]	PIN_L21	GPIO Connection 0[28]	GPIO_1[28]	PIN_U20	GPIO Connection 1[28]
GPIO_0[29]	PIN_L20	GPIO Connection 0[29]	GPIO_1[29]	PIN_U21	GPIO Connection 1[29]
GPIO_0[30]	PIN_J25	GPIO Connection 0[30]	GPIO_1[31]	PIN_V26	GPIO Connection 1[31]
GPIO_0[31]	PIN_J26	GPIO Connection 0[31]	GPIO_1[31]	PIN_V25	GPIO Connection 1[31]
GPIO_0[32]	PIN_L23	GPIO Connection 0[32]	GPIO_1[32]	PIN_V24	GPIO Connection 1[32]
GPIO_0[33]	PIN_L24	GPIO Connection 0[33]	GPIO_1[33]	PIN_V23	GPIO Connection 1[33]
GPIO_0[34]	PIN_L25	GPIO Connection 0[34]	GPIO_1[34]	PIN_W25	GPIO Connection 1[34]
GPIO_0[35]	PIN_L19	GPIO Connection 0[35]	GPIO_1[35]	PIN_W23	GPIO Connection 1[35]

Tabla 7. Asignación de pines de los conectores de expansión.

- *Interfaz VGA:*

La placa DE2 incluye un conector de 16 pines D-SUB para salida VGA. La sincronización de las señales VGA proviene directamente del dispositivo Cyclone II FPGA. El conector denominado ADV7123 se usa para llevar las señales analógicas de salida RGB (red, green, blue).

Signal Name	FPGA Pin No.	Description
VGA_R[0]	PIN_C8	VGA Red[0]
VGA_R[1]	PIN_F10	VGA Red[1]
VGA_R[2]	PIN_G10	VGA Red[2]
VGA_R[3]	PIN_D9	VGA Red[3]
VGA_R[4]	PIN_C9	VGA Red[4]
VGA_R[5]	PIN_A8	VGA Red[5]
VGA_R[6]	PIN_H11	VGA Red[6]
VGA_R[7]	PIN_H12	VGA Red[7]
VGA_R[8]	PIN_F11	VGA Red[8]
VGA_R[9]	PIN_E10	VGA Red[9]
VGA_G[0]	PIN_B9	VGA Green[0]
VGA_G[1]	PIN_A9	VGA Green[1]
VGA_G[2]	PIN_C10	VGA Green[2]
VGA_G[3]	PIN_D10	VGA Green[3]
VGA_G[4]	PIN_B10	VGA Green[4]
VGA_G[5]	PIN_A10	VGA Green[5]
VGA_G[6]	PIN_G11	VGA Green[6]
VGA_G[7]	PIN_D11	VGA Green[7]
VGA_G[8]	PIN_E12	VGA Green[8]
VGA_G[9]	PIN_D12	VGA Green[9]
VGA_B[0]	PIN_J13	VGA Blue[0]
VGA_B[1]	PIN_J14	VGA Blue[1]
VGA_B[2]	PIN_F12	VGA Blue[2]
VGA_B[3]	PIN_G12	VGA Blue[3]
VGA_B[4]	PIN_J10	VGA Blue[4]
VGA_B[5]	PIN_J11	VGA Blue[5]
VGA_B[6]	PIN_C11	VGA Blue[6]
VGA_B[7]	PIN_B11	VGA Blue[7]
VGA_B[8]	PIN_C12	VGA Blue[8]
VGA_B[9]	PIN_B12	VGA Blue[9]
VGA_CLK	PIN_B8	VGA Clock
VGA_BLANK	PIN_D6	VGA BLANK
VGA_HS	PIN_A7	VGA H_SYNC
VGA_VS	PIN_D8	VGA V_SYNC
VGA_SYNC	PIN_B7	VGA SYNC

Tabla 8. Asignación de pines del conector ADV7123.

- *CODEC de audio de 24 bits:*

La placa DE2 contiene un dispositivo de audio de alta calidad de 24 bits. Este dispositivo cuenta con entrada de micrófono, entrada y salida de línea. Este chip de audio es conectado a través de un bus serie al dispositivo Cyclone II FPGA.

Signal Name	FPGA Pin No.	Description
AUD_ADCLRCK	PIN_C5	Audio CODEC ADC LR Clock
AUD_ADCDAT	PIN_B5	Audio CODEC ADC Data
AUD_DACLRCK	PIN_C6	Audio CODEC DAC LR Clock
AUD_DACDAT	PIN_A4	Audio CODEC DAC Data
AUD_XCK	PIN_A5	Audio CODEC Chip Clock
AUD_BCLK	PIN_B4	Audio CODEC Bit-Stream Clock
I2C_SCLK	PIN_A6	I2C Data
I2C_SDAT	PIN_B6	I2C Clock

Tabla 9. Asignación de pines del CODEC de audio.

- *Puerto serie RS-232:*

La placa DE2 utiliza un conector D-SUB de 9 pines para sus comunicaciones a través del estándar RS-232.

Signal Name	FPGA Pin No.	Description
UART_RXD	PIN_C25	UART Receiver
UART_TXD	PIN_B25	UART Transmitter

Tabla 10. Asignación de pines para la conexión RS-232.

- *Puerto serie PS/2:*

La placa DE2 emplea el estándar PS/2 a través de su conector PS/2 para ratón ó teclado.

Signal Name	FPGA Pin No.	Description
PS2_CLK	PIN_D26	PS/2 Clock
PS2_DAT	PIN_C24	PS/2 Data

Tabla 11. Asignación de pines para la conexión PS/2.

- *Controlador FastEthernet:*

La placa DE2 proporciona soporte Ethernet a través del chip de control Davicom DM9111A Fast Ethernet.

Signal Name	FPGA Pin No.	Description
ENET_DATA[0]	PIN_D17	DM9000A DATA[0]
ENET_DATA[1]	PIN_C17	DM9000A DATA[1]
ENET_DATA[2]	PIN_B18	DM9000A DATA[2]
ENET_DATA[3]	PIN_A18	DM9000A DATA[3]
ENET_DATA[4]	PIN_B17	DM9000A DATA[4]
ENET_DATA[5]	PIN_A17	DM9000A DATA[5]
ENET_DATA[6]	PIN_B16	DM9000A DATA[6]
ENET_DATA[7]	PIN_B15	DM9000A DATA[7]
ENET_DATA[8]	PIN_B20	DM9000A DATA[8]
ENET_DATA[9]	PIN_A20	DM9000A DATA[9]
ENET_DATA[10]	PIN_C19	DM9000A DATA[10]
ENET_DATA[11]	PIN_D19	DM9000A DATA[11]
ENET_DATA[12]	PIN_B19	DM9000A DATA[12]
ENET_DATA[13]	PIN_A19	DM9000A DATA[13]
ENET_DATA[14]	PIN_E18	DM9000A DATA[14]
ENET_DATA[15]	PIN_D18	DM9000A DATA[15]
ENET_CLK	PIN_B24	DM9000A Clock 25 MHz
ENET_CMD	PIN_A21	DM9000A Command/Data Select, 0 = Command, 1 =Data
ENET_CS_N	PIN_A23	DM9000A Chip Select
ENET_INT	PIN_B21	DM9000A Interrupt
ENET_RD_N	PIN_A22	DM9000A Read
ENET_WR_N	PIN_B22	DM9000A Write
ENET_RST_N	PIN_B23	DM9000A Reset

Tabla 12. Asignación de pines para la conexión Fast Ethernet.

- *Decodificador TV:*

La placa DE2 se encuentra equipada con el chip decodificador para dispositivos analógicos ADV7181 TV. Este chip es un decodificador integrado de vídeo que automáticamente detecta y convierte señales de televisión de los estándares analógicos (NTSC, PAL y SECAM) al formato de muestreo de datos 4:2:2 para televisión digital. El decodificador de televisión está conectado con el dispositivo Cyclone II FPGA a través del bus serie I2C.

Signal Name	FPGA Pin No.	Description
TD_DATA[0]	PIN_J9	TV Decoder Data[0]
TD_DATA[1]	PIN_E8	TV Decoder Data[1]
TD_DATA[2]	PIN_H8	TV Decoder Data[2]
TD_DATA[3]	PIN_H10	TV Decoder Data[3]
TD_DATA[4]	PIN_G9	TV Decoder Data[4]
TD_DATA[5]	PIN_F9	TV Decoder Data[5]
TD_DATA[6]	PIN_D7	TV Decoder Data[6]
TD_DATA[7]	PIN_C7	TV Decoder Data[7]
TD_HS	PIN_D5	TV Decoder H_SYNC
TD_VS	PIN_K9	TV Decoder V_SYNC
TD_CLK27	PIN_C16	TV Decoder Clock Input.
TD_RESET	PIN_C4	TV Decoder Reset
I2C_SCLK	PIN_A6	I2C Data
I2C_SDAT	PIN_B6	I2C Clock

Tabla 13. Asignación de pines del decodificador de TV.

- *Conexión USB:*

Signal Name	FPGA Pin No.	Description
OTG_ADDR[0]	PIN_K7	ISP1362 Address[0]
OTG_ADDR[1]	PIN_F2	ISP1362 Address[1]
OTG_DATA[0]	PIN_F4	ISP1362 Data[0]
OTG_DATA[1]	PIN_D2	ISP1362 Data[1]
OTG_DATA[2]	PIN_D1	ISP1362 Data[2]
OTG_DATA[3]	PIN_F7	ISP1362 Data[3]
OTG_DATA[4]	PIN_J5	ISP1362 Data[4]
OTG_DATA[5]	PIN_J8	ISP1362 Data[5]
OTG_DATA[6]	PIN_J7	ISP1362 Data[6]
OTG_DATA[7]	PIN_H6	ISP1362 Data[7]
OTG_DATA[8]	PIN_E2	ISP1362 Data[8]
OTG_DATA[9]	PIN_E1	ISP1362 Data[9]
OTG_DATA[10]	PIN_K6	ISP1362 Data[10]
OTG_DATA[11]	PIN_K5	ISP1362 Data[11]
OTG_DATA[12]	PIN_G4	ISP1362 Data[12]
OTG_DATA[13]	PIN_G3	ISP1362 Data[13]
OTG_DATA[14]	PIN_J6	ISP1362 Data[14]
OTG_DATA[15]	PIN_K8	ISP1362 Data[15]
OTG_CS_N	PIN_F1	ISP1362 Chip Select
OTG_RD_N	PIN_G2	ISP1362 Read
OTG_WR_N	PIN_G1	ISP1362 Write
OTG_RST_N	PIN_G5	ISP1362 Reset
OTG_INT0	PIN_B3	ISP1362 Interrupt 0
OTG_INT1	PIN_C3	ISP1362 Interrupt 1
OTG_DACK0_N	PIN_C2	ISP1362 DMA Acknowledge 0
OTG_DACK1_N	PIN_B2	ISP1362 DMA Acknowledge 1
OTG_DREQ0	PIN_F6	ISP1362 DMA Request 0
OTG_DREQ1	PIN_E5	ISP1362 DMA Request 1

Tabla 14. Asignación de pines para la conexión USB.

- *Puerto IrDA:*

Signal Name	FPGA Pin No.	Description
IRDA_TXD	PIN_AE24	IRDA Transmitter
IRDA_RXD	PIN_AE25	IRDA Receiver

Tabla 15. Asignación de pines para puerto IrDA.

- *Bloques de memoria SDRAM/SRAM/Flash:*

La placa DE2 contiene 8 Mbytes de memoria SDRAM, 512 KBytes de memoria SRAM y 4 Mbytes de memoria Flash.

Signal Name	FPGA Pin No.	Description
DRAM_ADDR[0]	PIN_T6	SDRAM Address[0]
DRAM_ADDR[1]	PIN_V4	SDRAM Address[1]
DRAM_ADDR[2]	PIN_V3	SDRAM Address[2]
DRAM_ADDR[3]	PIN_W2	SDRAM Address[3]
DRAM_ADDR[4]	PIN_W1	SDRAM Address[4]
DRAM_ADDR[5]	PIN_U6	SDRAM Address[5]
DRAM_ADDR[6]	PIN_U7	SDRAM Address[6]
DRAM_ADDR[7]	PIN_U5	SDRAM Address[7]
DRAM_ADDR[8]	PIN_W4	SDRAM Address[8]
DRAM_ADDR[9]	PIN_W3	SDRAM Address[9]
DRAM_ADDR[10]	PIN_Y1	SDRAM Address[10]
DRAM_ADDR[11]	PIN_V5	SDRAM Address[11]
DRAM_DQ[0]	PIN_V6	SDRAM Data[0]
DRAM_DQ[1]	PIN_AA2	SDRAM Data[1]
DRAM_DQ[2]	PIN_AA1	SDRAM Data[2]
DRAM_DQ[3]	PIN_Y3	SDRAM Data[3]
DRAM_DQ[4]	PIN_Y4	SDRAM Data[4]
DRAM_DQ[5]	PIN_R8	SDRAM Data[5]
DRAM_DQ[6]	PIN_T8	SDRAM Data[6]
DRAM_DQ[7]	PIN_V7	SDRAM Data[7]
DRAM_DQ[8]	PIN_W6	SDRAM Data[8]
DRAM_DQ[9]	PIN_AB2	SDRAM Data[9]
DRAM_DQ[10]	PIN_AB1	SDRAM Data[10]
DRAM_DQ[11]	PIN_AA4	SDRAM Data[11]
DRAM_DQ[12]	PIN_AA3	SDRAM Data[12]
DRAM_DQ[13]	PIN_AC2	SDRAM Data[13]
DRAM_DQ[14]	PIN_AC1	SDRAM Data[14]
DRAM_DQ[15]	PIN_AA5	SDRAM Data[15]
DRAM_BA_0	PIN_AE2	SDRAM Bank Address[0]
DRAM_BA_1	PIN_AE3	SDRAM Bank Address[1]
DRAM_LDQM	PIN_AD2	SDRAM Low-byte Data Mask
DRAM_UDQM	PIN_Y5	SDRAM High-byte Data Mask
DRAM_RAS_N	PIN_AB4	SDRAM Row Address Strobe
DRAM_CAS_N	PIN_AB3	SDRAM Column Address Strobe
DRAM_CKE	PIN_AA6	SDRAM Clock Enable
DRAM_CLK	PIN_AA7	SDRAM Clock
DRAM_WE_N	PIN_AD3	SDRAM Write Enable
DRAM_CS_N	PIN_AC3	SDRAM Chip Select

Tabla 16. Asignación de pines de memoria SDRAM

Signal Name	FPGA Pin No.	Description
SRAM_ADDR[0]	PIN_AE4	SRAM Address[0]
SRAM_ADDR[1]	PIN_AF4	SRAM Address[1]
SRAM_ADDR[2]	PIN_AC5	SRAM Address[2]
SRAM_ADDR[3]	PIN_AC6	SRAM Address[3]
SRAM_ADDR[4]	PIN_AD4	SRAM Address[4]
SRAM_ADDR[5]	PIN_AD5	SRAM Address[5]
SRAM_ADDR[6]	PIN_AE5	SRAM Address[6]
SRAM_ADDR[7]	PIN_AF5	SRAM Address[7]
SRAM_ADDR[8]	PIN_AD6	SRAM Address[8]
SRAM_ADDR[9]	PIN_AD7	SRAM Address[9]
SRAM_ADDR[10]	PIN_V10	SRAM Address[10]
SRAM_ADDR[11]	PIN_V9	SRAM Address[11]
SRAM_ADDR[12]	PIN_AC7	SRAM Address[12]
SRAM_ADDR[13]	PIN_W8	SRAM Address[13]
SRAM_ADDR[14]	PIN_W10	SRAM Address[14]
SRAM_ADDR[15]	PIN_Y10	SRAM Address[15]
SRAM_ADDR[16]	PIN_AB8	SRAM Address[16]
SRAM_ADDR[17]	PIN_AC8	SRAM Address[17]
SRAM_DQ[0]	PIN_AD8	SRAM Data[0]
SRAM_DQ[1]	PIN_AE6	SRAM Data[1]
SRAM_DQ[2]	PIN_AF6	SRAM Data[2]
SRAM_DQ[3]	PIN_AA9	SRAM Data[3]
SRAM_DQ[4]	PIN_AA10	SRAM Data[4]
SRAM_DQ[5]	PIN_AB10	SRAM Data[5]
SRAM_DQ[6]	PIN_AA11	SRAM Data[6]
SRAM_DQ[7]	PIN_Y11	SRAM Data[7]
SRAM_DQ[8]	PIN_AE7	SRAM Data[8]
SRAM_DQ[9]	PIN_AF7	SRAM Data[9]
SRAM_DQ[10]	PIN_AE8	SRAM Data[10]
SRAM_DQ[11]	PIN_AF8	SRAM Data[11]
SRAM_DQ[12]	PIN_W11	SRAM Data[12]
SRAM_DQ[13]	PIN_W12	SRAM Data[13]
SRAM_DQ[14]	PIN_AC9	SRAM Data[14]
SRAM_DQ[15]	PIN_AC10	SRAM Data[15]
SRAM_WE_N	PIN_AE10	SRAM Write Enable
SRAM_OE_N	PIN_AD10	SRAM Output Enable
SRAM_UB_N	PIN_AF9	SRAM High-byte Data Mask
SRAM_LB_N	PIN_AE9	SRAM Low-byte Data Mask

Tabla 17. Asignación de pines de memoria SRAM.

Signal Name	FPGA Pin No.	Description
FL_ADDR[0]	PIN_AC18	FLASH Address[0]
FL_ADDR[1]	PIN_AB18	FLASH Address[1]
FL_ADDR[2]	PIN_AE19	FLASH Address[2]
FL_ADDR[3]	PIN_AF19	FLASH Address[3]
FL_ADDR[4]	PIN_AE18	FLASH Address[4]
FL_ADDR[5]	PIN_AF18	FLASH Address[5]
FL_ADDR[6]	PIN_Y16	FLASH Address[6]
FL_ADDR[7]	PIN_AA16	FLASH Address[7]
FL_ADDR[8]	PIN_AD17	FLASH Address[8]
FL_ADDR[9]	PIN_AC17	FLASH Address[9]
FL_ADDR[10]	PIN_AE17	FLASH Address[10]
FL_ADDR[11]	PIN_AF17	FLASH Address[11]
FL_ADDR[12]	PIN_W16	FLASH Address[12]
FL_ADDR[13]	PIN_W15	FLASH Address[13]
FL_ADDR[14]	PIN_AC16	FLASH Address[14]
FL_ADDR[15]	PIN_AD16	FLASH Address[15]
FL_ADDR[16]	PIN_AE16	FLASH Address[16]
FL_ADDR[17]	PIN_AC15	FLASH Address[17]
FL_ADDR[18]	PIN_AB15	FLASH Address[18]
FL_ADDR[19]	PIN_AA15	FLASH Address[19]
FL_ADDR[20]	PIN_Y15	FLASH Address[20]
FL_ADDR[21]	PIN_Y14	FLASH Address[21]
FL_DQ[0]	PIN_AD19	FLASH Data[0]
FL_DQ[1]	PIN_AC19	FLASH Data[1]
FL_DQ[2]	PIN_AF20	FLASH Data[2]
FL_DQ[3]	PIN_AE20	FLASH Data[3]
FL_DQ[4]	PIN_AB20	FLASH Data[4]
FL_DQ[5]	PIN_AC20	FLASH Data[5]
FL_DQ[6]	PIN_AF21	FLASH Data[6]
FL_DQ[7]	PIN_AE21	FLASH Data[7]
FL_CE_N	PIN_V17	FLASH Chip Enable
FL_OE_N	PIN_W17	FLASH Output Enable
FL_RST_N	PIN_AA18	FLASH Reset
FL_WE_N	PIN_AA17	FLASH Write Enable

Tabla 18. Asignación de pines de memoria Flash.

CAPÍTULO 7.

OBJETIVOS DE LOS GUIONES DE PRÁCTICAS

7.- OBJETIVOS DE LOS GUIONES DE PRÁCTICAS

El objetivo de las prácticas expuestas en el presente proyecto, es que el alumno aprenda a realizar diseños basados en dispositivos de lógica programable. Para ello, dichos diseños se realizarán utilizando el software QUARTUS II® y la tarjeta educacional DE2, ambos de al compañía Altera®. Se pretende que el alumno sea capaz de aplicar los conceptos teóricos obtenidos a lo largo del curso, así como adquirir unos conocimientos básicos al respecto, de forma que en un futuro pueda enfrentarse a problemas más complejos.

En este apartado se hará una breve descripción de los objetivos que se persiguen con la realización de cada una de las prácticas. Es necesario tener en cuenta la carga de horas prácticas de cada una de las asignaturas para las cuales se desarrollan los respectivos planes.

Los objetivos globales de todo el conjunto de prácticas son los siguientes:

- Consolidación y puesta en práctica de los conocimientos teóricos adquiridos por el alumno en las asignaturas 'Estructura y Tecnología de Computadores' y 'Circuitos Electrónicos Digitales' respectivamente. Para ello se sincronizarán, en la medida de lo posible, las sesiones prácticas con las sesiones teóricas.

- Familiarización con el diseño lógico. Conocer y saber utilizar técnicas para el desarrollo de los diseños, así como comprender sus respectivos resultados.

- Realización de distintos circuitos digitales por medio de la utilización de dispositivos lógicos programables. El PLD utilizado será de la familia de dispositivos Cyclone II®, en concreto el EP2C35F672C6, de la compañía Altera®, el cual viene integrado en la tarjeta educacional DE2.

- Manejo del software de diseño QUARTUS II® de la compañía Altera®, creado específicamente para el diseño con dispositivos de lógica programable.

Para conseguir lo especificado anteriormente se ha desarrollado el correspondiente plan de sesiones prácticas para cada una de las asignaturas. A continuación se especifican los objetivos particulares de cada práctica incluida en los diferentes planes.

- **Asignatura 'Estructura y Tecnología de Computadores' de la titulación 'Ingeniería Técnica en Informática de Gestión'.**

- Práctica 1. Puertas Lógicas.

Se trata de una práctica de aprendizaje, en la que el alumno tendrá un primer contacto con el software QUARTUS II®, el cual será empleado en el resto de las prácticas. Para ello se realizarán los mismos ejercicios realizados en la práctica cero basados en circuitos con puertas lógicas, ya conocidos por el alumno, pero utilizando el entorno gráfico del software mencionado anteriormente para su diseño. De esta manera, el alumno comprobará cómo realizar un diseño, además de aprender a interpretar sus resultados, comparándolos con los obtenidos en la práctica anterior.

El primer ejercicio del presente guión se realizará de manera guiada, con el fin de que el alumno obtenga una base con los pasos a seguir a lo largo de la realización de un diseño, y lo pueda aplicar así en los siguientes ejercicios y prácticas.

Asimismo, en este guión se muestra el método de trabajo a seguir en el resto de las prácticas, donde, en primer lugar, se exponen ejercicios semi-resueltos que sirven como base para los posteriores ejercicios de diseño.

- Práctica 2. Circuitos Aritméticos.

El objetivo de esta práctica es el de mostrar y diseñar alguno de los circuitos lógicos capaces de realizar operaciones aritméticas básicas mediante números binarios, tales como la suma, resta ó multiplicación. Para ello se simularán distintos diseños realizados mediante el Editor Gráfico del software y se comprobará su correcto funcionamiento.

- Práctica 3. Circuitos Lógicos Combinacionales.

En esta práctica se pretende que el alumno consolide los conocimientos adquiridos en las prácticas anteriores basados en la realización de circuitos lógicos.

Además, se dan a conocer tres elementos básicos de la electrónica digital en el campo de las comunicaciones: Multiplexor, demultiplexor/decodificador y comparadores. Se pretende utilizar estos elementos para realizar funciones lógicas.

- Práctica 4. Introducción a VHDL.

Se trata de una práctica de aprendizaje, en la que se hace una introducción al manejo y comprensión del lenguaje de programación VHDL. Para ello la práctica comienza con una breve introducción a dicho lenguaje, para proceder, a continuación, con la realización de una serie de ejercicios semi-guiados que deberán ser estudiados por parte del alumno. Entre dichos ejercicios, una parte de ellos está orientada a ver cómo realizar los ejercicios vistos hasta ahora en las anteriores prácticas desde un Editor de Texto mediante el lenguaje de programación VHDL, mientras que la otra parte de los mismos está orientada hacia algoritmos que serán utilizados en prácticas posteriores.

- Práctica 5. Biestables y Contadores.

El objetivo de esta práctica es dar a conocer el elemento básico de los circuitos secuenciales, el biestable. Además se verá alguna de sus aplicaciones, como los contadores.

Para ello se simulará el funcionamiento de un biestable JK, que a su vez servirá para conocer el funcionamiento de un biestable comercial, el 7476, con la finalidad de que los alumnos tengan un contacto con un elemento comercial.

Además, se simulará el comportamiento de dos contadores síncronos, uno ascendente y otro descendente.

En esta práctica se comienza a realizar volcados de diseños a la tarjeta educacional DE2. Para ello, se presenta un primer ejercicio en el cual se realizará el volcado del mismo de forma guiada ya que será la metodología a seguir en el resto de ejercicios que exigen volcado a la placa.

- Práctica 6. Diseño y Volcado de un Cronómetro.

El objetivo de esta práctica es el de aprender a realizar aplicaciones más avanzadas a partir de circuitos secuenciales. En concreto, se va a trabajar con divisores de frecuencia capaces de obtener una señal de frecuencia 1Hz, partiendo de la señal de reloj de la placa DE2 (27MHz ó 50MHz) y emplear esta señal para diseñar el segundero de un reloj.

- Práctica 7. Máquina de Estados.

En esta práctica se pretende que el alumno consolide los conocimientos teóricos adquiridos sobre máquinas de estados y vea algún ejemplo práctico de las mismas.

Asimismo, se profundiza más en el lenguaje de programación VHDL mediante la realización de dos diseños de máquinas de estados, uno guiado que sirva como base y otro de diseño.

- Práctica 8. Diseño Libre.

El objetivo de esta práctica es que el alumno realice un diseño con total libertad, aplicando para ello los conocimientos adquiridos a lo largo de la realización de las prácticas de la asignatura.

- **Asignatura 'Circuitos Electrónicos Digitales' de la titulación 'Ingeniería de Telecomunicación'.**

Debido a la menor carga lectiva de prácticas de esta asignatura, el presente plan consta de una recopilación de aquellos temas que se consideran más útiles para el alumno, en función de los contenidos teóricos de la asignatura, en base al plan establecido en el apartado anterior.

- Práctica 2. Entorno Software de Programación.

Este guión de prácticas consta de dos partes. En primer lugar se trata de una práctica de aprendizaje en la que el alumno tiene un primer contacto con el software de simulación QUARTUS II®. En esta parte se repite alguno de los ejercicios vistos en la práctica uno basados en circuitos lógicos, ya conocidos por el alumno, con el fin de adaptarse al entorno gráfico de programación y comparar los resultados obtenidos.

El primer ejercicio del presente guión se realizará de forma guiada para que el alumno obtenga los pasos a seguir en la realización de un diseño para posteriores ejercicios.

La segunda parte del guión trata de mostrar alguno de los circuitos lógicos capaces de realizar operaciones aritméticas básicas a partir de números binarios, tales como la suma, resta ó multiplicación. Además, en el apartado de ejercicios de diseño, se refuerza lo aprendido en la práctica anterior en cuanto a circuitos combinacionales, mediante la realización de dos ejercicios basados en comparadores.

Se simularán los distintos diseños realizados mediante el Editor Gráfico con el fin de poder comprobar su correcto funcionamiento.

- Práctica 3. Biestables y Contadores.

El objetivo de esta práctica es dar a conocer el elemento básico de los circuitos secuenciales, el biestable, y ver alguna de sus aplicaciones, tales como los contadores.

Para ello se realizará la simulación de un biestable JK, que a su vez servirá para conocer el funcionamiento de un circuito comercial, el biestable 7476. Además se deberá simular y comprender el funcionamiento de dos contadores síncronos, uno ascendente y otro descendente.

En este guión se aprende a realizar el volcado de diseños a la tarjeta educacional. Para ello se presenta un ejercicio guiado que permite ver los pasos seguidos en la operación para poder aplicarlos en los posteriores ejercicios.

CAPÍTULO 8.

GUIONES DE PRÁCTICAS PARA LA TITULACIÓN

'INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN'

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

(1º Ingeniería Técnica en Informática de Gestión)

Práctica 1: Puertas Lógicas.

- Entorno Software de Simulación.

El haber estudiado el manual introductorio del software es muy importante y facilitará en gran medida el desarrollo de la práctica.

Es muy importante entender perfectamente esta práctica y obtener soltura en el manejo del software, ya que será el punto de partida para el resto de las prácticas.

1.- OBJETIVOS

a) Introducción al manejo del software Quartus II® utilizado para la implementación de diseños lógicos empleando PLD`s, no llegando a programar en esta práctica las tarjetas educacionales DE2 ó UP2.
b) Se repetirán los ejercicios realizados en la Práctica 0 con el software indicado anteriormente, realizando el primero de los ejercicios de manera guiada con el fin de observar que los resultados obtenidos en la Práctica 0 coinciden con los simulados en la actual práctica.

2.- MATERIAL

- Ordenador personal con el software QUARTUS II® de Altera®.
- Manual introductorio al software.
- Guión de prácticas.

3.- DESARROLLO DE LA PRÁCTICA

3.1.- PUERTAS NOT, AND Y OR

El diseño a introducir será el siguiente:

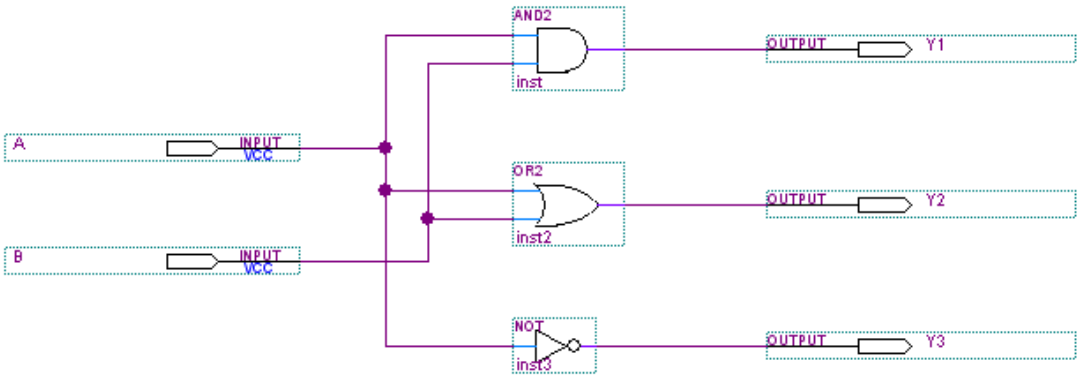


Figura 1. Diseño Ejercicio 3.1

Para la implementación del circuito y posterior comprobación de resultados se seguirán los pasos que se indican a continuación.

1º- Realización Gráfica del Diseño.

b) Inicializar el software Quartus II.

c) El primer paso a la hora de realizar un diseño con Quartus II consiste en la definición del proyecto en el que se va a operar. Un proyecto consiste en englobar una serie de archivos de diseño dentro de un subdirectorio dado.
Para definir el proyecto, es necesario acudir a la opción 'New Project Wizard' situada en el menú 'File', que es el asistente para la definición de proyectos. Este asistente ofrece una serie de ventanas que permiten realizar las siguientes operaciones:

I. Definición de nombre y directorio de almacenamiento del proyecto.

- Importante: No se pueden crear dos proyectos en un mismo directorio de trabajo, ya que posteriormente, en el proceso de compilación, se producirá conflicto entre ambos.

II. Inclusión de archivos anteriores que quieran ser utilizados en el presente proyecto. Para esta práctica no es necesario añadir ningún archivo.

III. Selección de la familia y dispositivo que vayamos a utilizar. Aunque en esta práctica los diseños no van a ser implementados sobre las tarjetas educacionales, es bueno acostumbrarnos desde ahora a seleccionar el dispositivo EPF10K70RC240-4 de la familia FLEX10K, en el caso de la tarjeta educacional UP2, y el dispositivo EP2C35F672C6 de la familia Cyclone II, en el caso de la placa DE2, para la compilación del diseño. Esto se muestra en la Figura 2.

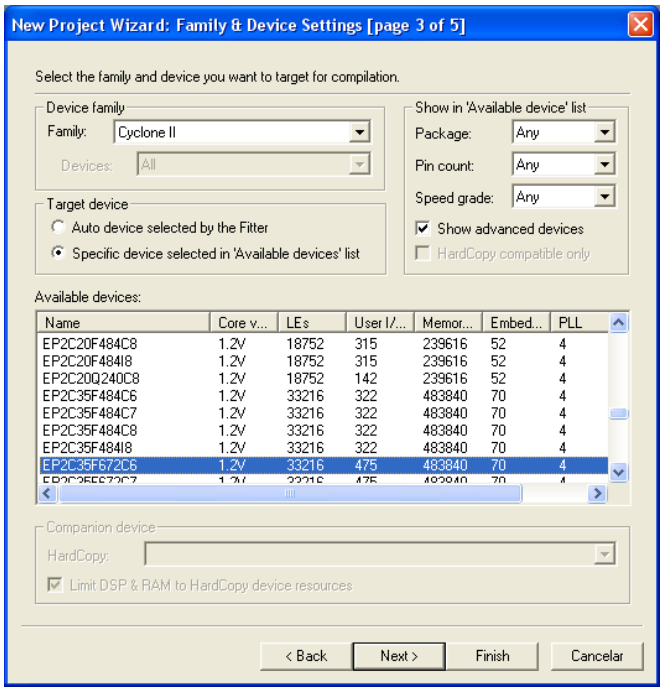


Figura 2. Selección de familia y dispositivo para la compilación

IV. Selección de otras herramientas EDA. No es necesario ninguna herramienta adicional, luego pasamos directamente al siguiente paso.

V. Por último, aparece una ventana que muestra un resumen de lo incluido en los pasos anteriores. Pulsando sobre el botón 'Finish' quedará definido el nuevo proyecto y se estará en disposición de empezar a trabajar.

d) Abrir un nuevo archivo para el Editor Gráfico. Para ello se debe acceder al icono que representa un folio en blanco y, en la nueva ventana, seleccionar la opción 'Block Diagram/Schematic File'.

e) Introducir los elementos que forman el circuito a implementar, mostrado en la Figura 1. Esto se puede hacer pinchando con el botón derecho del ratón sobre la pantalla, eligiendo la opción 'Insert' seguido de 'Symbol…' en el menú que se despliega. Al realizar esto aparecerá la ventana que se muestra en la Figura 3 en la cual se puede escribir directamente el nombre del componente a utilizar o bien buscarlo en las diferentes librerías existentes. Los componentes que nosotros estamos buscando se encuentran en la carpeta de primitivas, opción 'logic', sin embargo al ser componentes conocidos y muy sencillos, se buscarán directamente por su nombre en la casilla 'Name'. Los componentes a utilizar son: input, and2 ('and' con dos entradas), or2 ('or' con dos entradas), not y output. Mediante la opción 'Repeat-insert mode' se incluye cada símbolo tantas veces como se quiera.

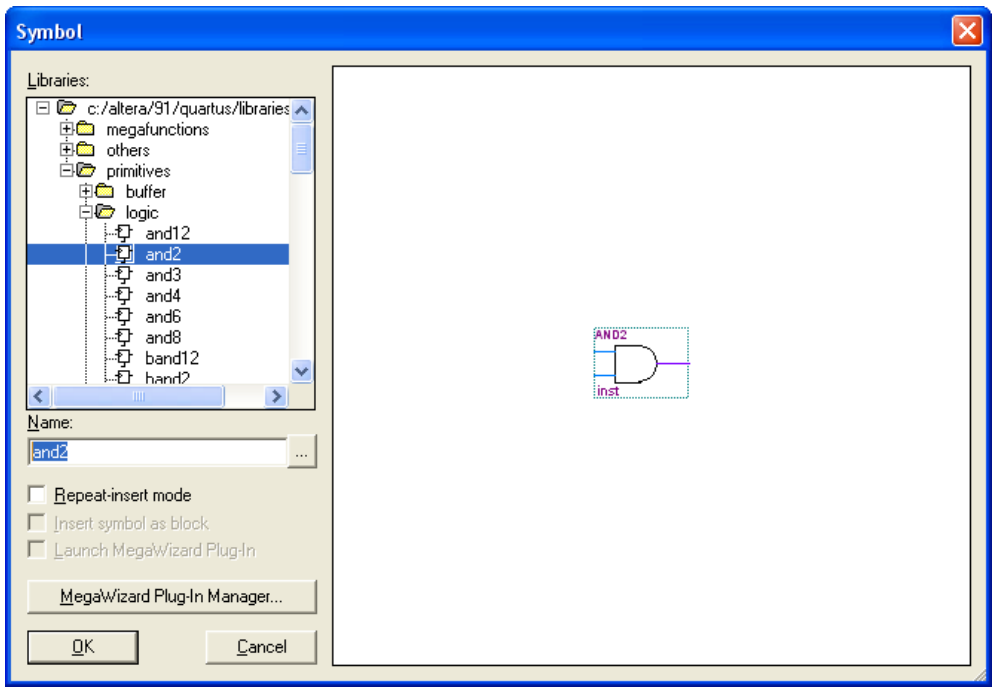


Figura 3. Inclusión de símbolos

f) Dibujar las conexiones. Para ello se pulsa sobre la herramienta 'Orthogonal Node Tool' y con el botón izquierdo del ratón pulsado se unen los elementos que interese.

g) Asignar nombres a las entradas y salidas. Para ello, hacer doble clic con el botón izquierdo o un clic con el botón derecho sobre cada una de las entradas y salidas. En el caso de utilizar el botón derecho, aparecerá un menú desplegable en el que se debe seleccionar la opción 'Properties'. A continuación en la casilla 'Pin name (s)' introducir el nombre deseado.

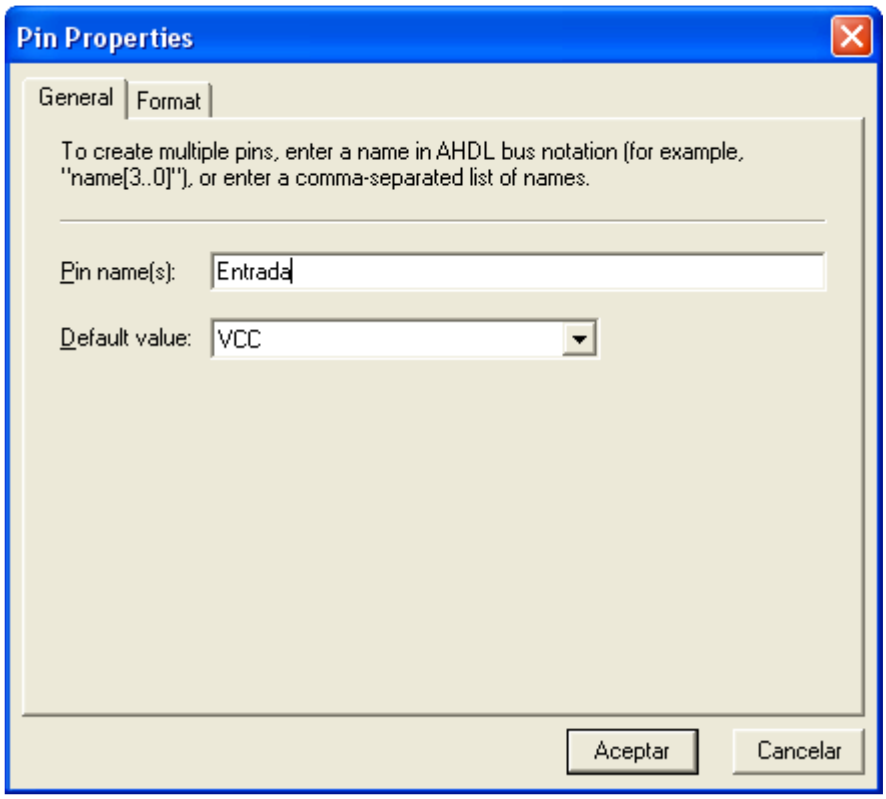


Figura 4. Introducción de nombres

Es imprescindible usar nombres únicos para los diferentes pines. Los pines y cables con el mismo nombre son conectados automáticamente sin que aparezca un cable visible en el diagrama esquemático.

h) Guardar el diseño en la carpeta definida para el proyecto. Para ello acceder a la orden 'Save' del menú 'File'.

Al llegar a este punto ya se ha introducido el diseño, por lo que, a continuación, se procederá a la compilación del mismo.

2º- Compilación del Diseño

La compilación del diseño conlleva la detección de posibles errores en la construcción o sintaxis del mismo, sintetiza el diseño lógico, produce información temporal para la simulación, ajusta el diseño al PLD seleccionado y genera el archivo necesario para el volcado del programa a la tarjeta de simulación.

Antes de iniciar la compilación es necesario que el diseño haya sido previamente guardado.

- **Importante: Tener en cuenta que en el caso de proyectos en los que se incluyen archivos de diseño de proyectos anteriores, el proceso de compilación se hace sobre el archivo jerárquicamente superior.**

El acceso a la herramienta de compilación de Quartus II puede realizarse de dos maneras diferentes:

- La primera de ellas consiste en acceder al menú desplegable de la pestaña 'Processing' de la barra de herramientas principal y seleccionar la opción 'Compiler Tool', como se ve en la Figura 5.
- La segunda opción consiste en iniciar directamente la compilación accediendo al icono asignado para ello, tal y como muestra la Figura 5. 

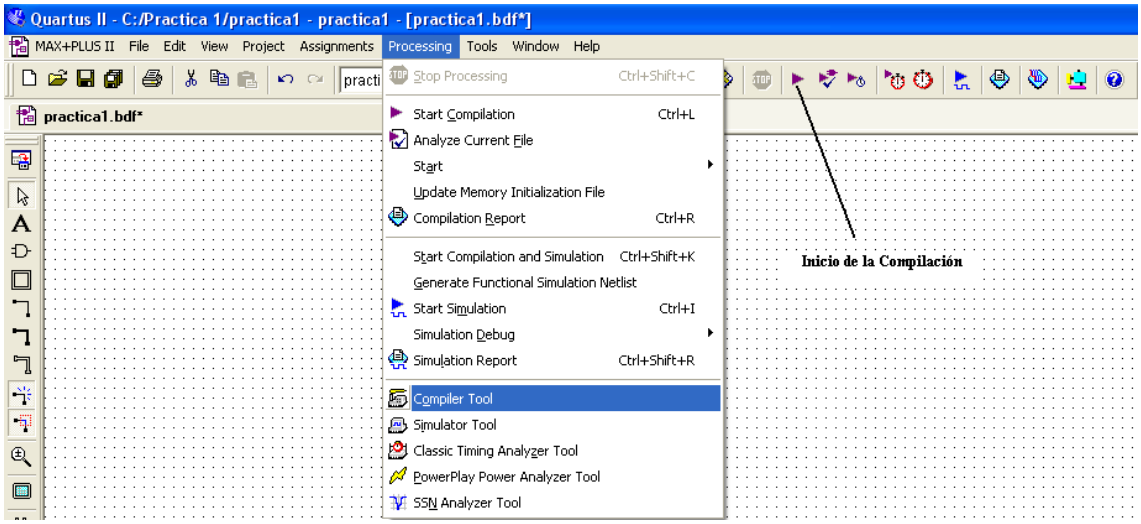


Figura 5. Apertura de la herramienta de compilación

Se recomienda que el alumno escoja la primera de las opciones ya que de esta manera es posible ver la ventana de la herramienta de compilación y los procesos que se van dando en la misma.

La Figura 6 muestra la ventana del compilador.

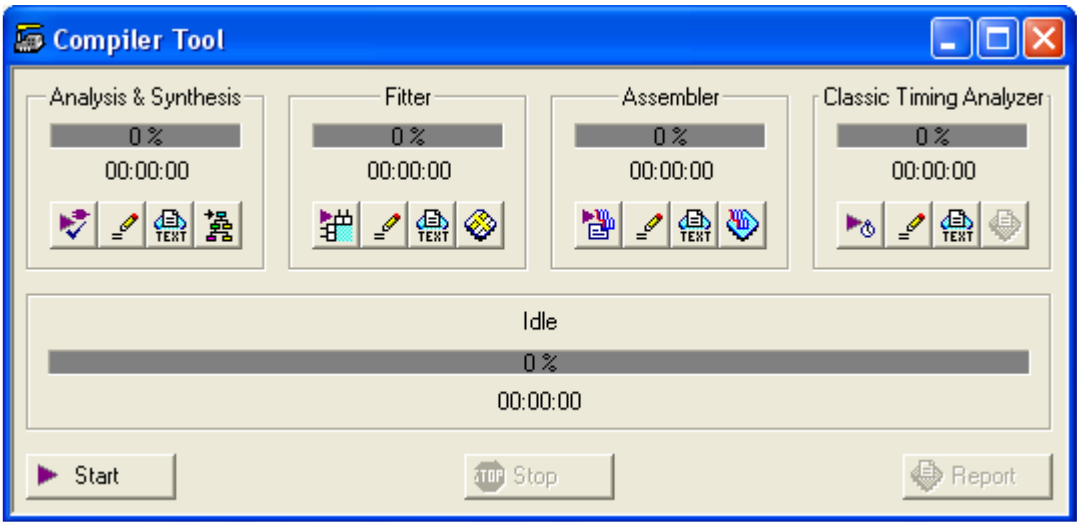


Figura 6. Ventana del compilador

El proceso de compilación completo comienza cuando se pulsa el botón 'Start', y éste puede ser interrumpido mediante el botón 'Stop'. Para ver las características del proceso es necesario recurrir al botón 'Report'.

Tras finalizar el proceso de compilación se presentará una ventana en la que se da información acerca de los posibles errores o advertencias que pudiera haber. Si no hay ningún error, se continuará con el proceso de simulación. Si por el contrario, hubiese algún error, éstos aparecerán en la ventana de mensajes del compilador en color rojo, y habrán de ser localizados y corregidos. Para localizar el error basta con seleccionarlo y pulsar el botón 'Locate'. Al hacer esto, el programa regresa al editor en el que se diseñó el circuito y resalta el símbolo o el texto erróneo.

Durante la compilación, el programa selecciona el dispositivo asignado previamente en la definición del proyecto. Si se hubiera obviado este paso el programa selecciona un dispositivo automáticamente. De todos modos siempre se podrá cambiar el PLD posteriormente.

3º- Edición de Señales

Para comprobar el correcto funcionamiento del diseño generado resulta necesario definir las señales de prueba que se van a introducir en el mismo. Esto se realiza mediante el Editor de Señales y los pasos a seguir son los siguientes:

a) Abrir un nuevo archivo del Editor de Señales. Para ello se pincha sobre el icono que representa una hoja en blanco y se selecciona la opción 'Vector Waveform File' desde la nueva ventana.
b) Fijar el tiempo de análisis desde el menú 'Edit' en la opción 'End Time…'. Fijar un valor alrededor de 5-10ms.
c) Introducir las señales que se quieren estudiar. Para ello se debe acceder al menú desplegable 'Edit' situado en la barra de herramientas principal y seleccionar la opción 'Insert' seguido de 'Insert Node or Bus…', con lo que se abrirá el siguiente cuadro.

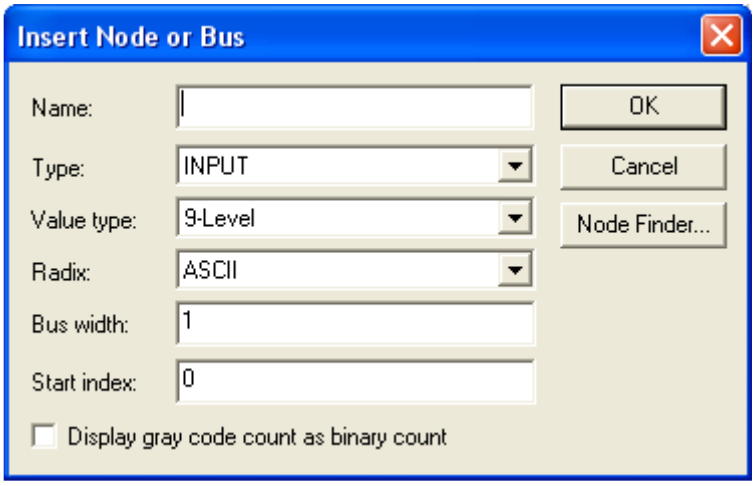


Figura 7. Tipo de señales

Esta ventana permite elegir el tipo de señales que interesa mostrar para la simulación del diseño. En nuestro caso dejaremos los parámetros que vienen por defecto y pulsaremos sobre el botón 'Node Finder…' para acceder a la siguiente ventana.

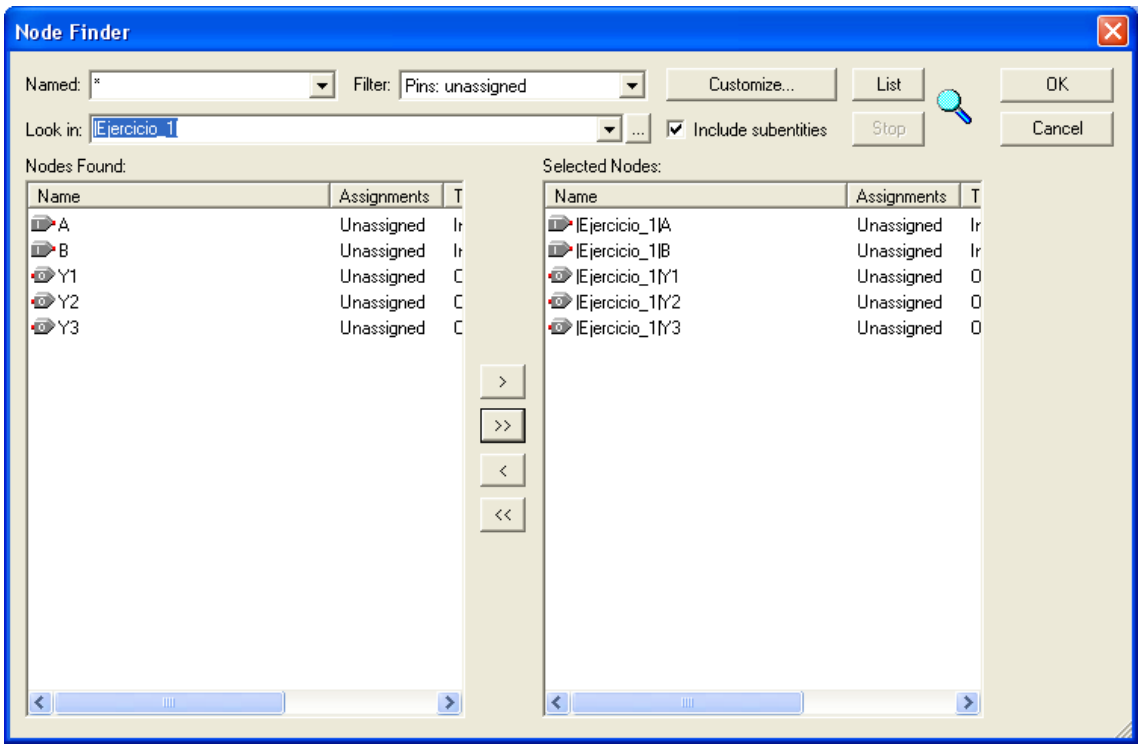


Figura 8. Selección de señales

En la ventana mostrada en la Figura 8 pincharemos sobre el botón 'List', con lo que nos aparecerán como señales disponibles todas las entradas y salidas correspondientes al diseño del proyecto en el que estamos trabajando. Aquí se han de seleccionar las señales con las que interesa trabajar, que en este caso particular son todas. A continuación se pulsará sobre el botón que indica la flecha hacia la derecha para incluir las señales en el Editor, por lo que ya están disponibles para la simulación. Para aceptar la operación pulsaremos sobre el botón 'OK'.

Aparecerá la siguiente ventana:

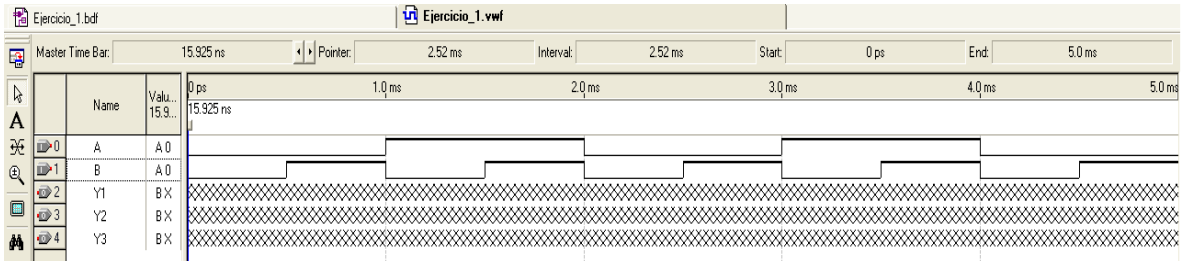


Figura 9. Señales para la simulación

En esta ventana se ve como las señales de entrada no tienen ningún valor asignado, por lo que se deberá elegir unos valores para las mismas de forma que representen los casos que interese estudiar.

d) Para simular los pulsadores utilizaremos señales cuadradas con una frecuencia de 1 KHz para la entrada A y otra de 500 Hz para la entrada B (Esto se hace para que los retardos introducidos por las puertas no distorsionen los resultados que se quieren obtener). De esta manera se obtendrán las distintas posibilidades de la tabla de verdad del circuito.

Lo primero que debemos hacer es seleccionar ambas señales de entrada y pinchar sobre ellas con el botón derecho del ratón, obteniendo la siguiente ventana.

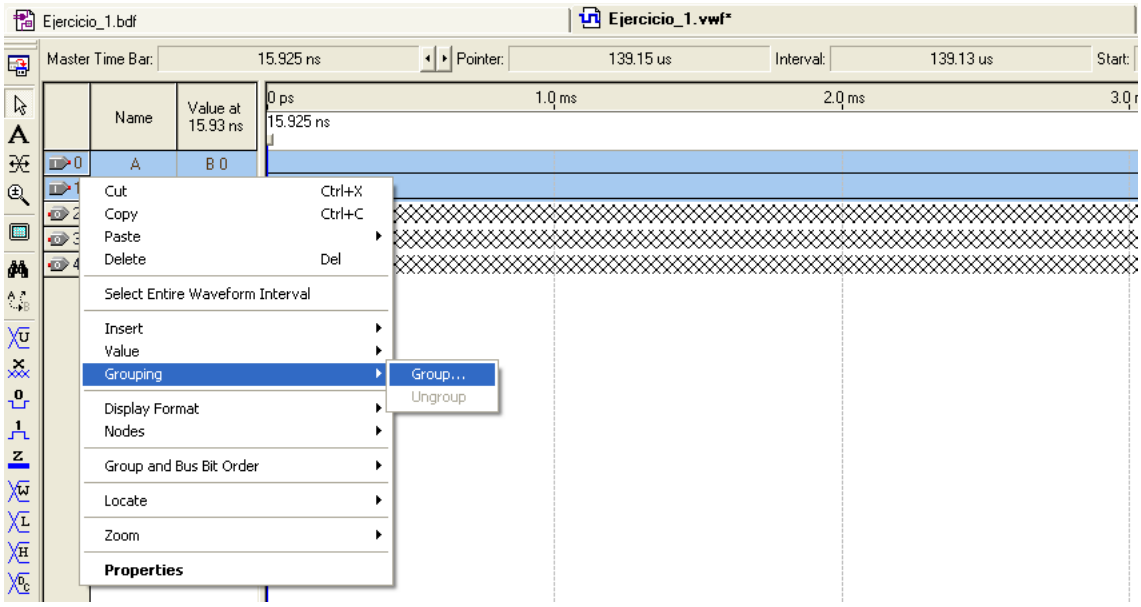


Figura 10. Agrupación de señales

En ella se elige la opción 'Grouping' seguido de 'Group…' con lo que se consigue agrupar las dos señales. Aparecerá una ventana en la que se le dará un nombre cualquiera al grupo. El siguiente paso será definir ese grupo. Para ello hay que pinchar con el botón derecho sobre el grupo creado y seleccionar la opción 'Value/Count Value…'. En la nueva ventana pinchar sobre la pestaña 'Timing' y fijar la opción 'Count Every' a la mitad del periodo más pequeño de interés. Esto se hace así porque esta opción indica el tiempo para el cual ocurre el cambio de estado de la señal. Si introducimos, por ejemplo, un valor de 0.5 ms, obtendremos una señal de 1 Khz.

A continuación, se deshace el grupo para ver claramente cada una de las señales por separado. Esto se hace pulsando con el botón derecho del ratón sobre el grupo y seleccionar la opción 'Grouping/Ungroup'.

**Nota: Asegurarse de que la opción 'Snap to grid' del menú 'View' se encuentre desactivada (por defecto suele estar activada), de lo contrario no se podrán fijar los semiperiodos que nosotros queramos.*

e) Si es necesario, hacer un zoom in o zoom out, para ver claramente la señal. La apariencia de las señales de entrada se muestra en la Figura 11.

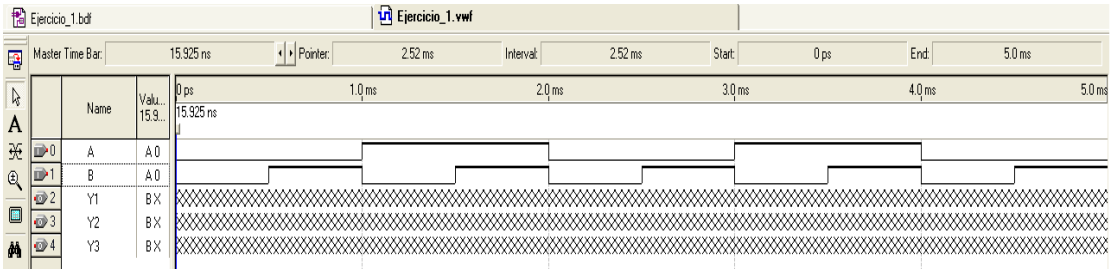


Figura 11. Señales de entrada

Como se puede observar, se obtienen todas las combinaciones posibles de una tabla de verdad de dos entradas.

f) Guardar el archivo, que al ser generado desde el Editor de Señales tendrá una extensión '.vwf'. Ya se han preparado las señales de entrada para proceder a la simulación y obtener los resultados correspondientes a las señales de salida.

4º- Simulación del Diseño.

La finalidad de este apartado es la comprobación del correcto funcionamiento del diseño realizado. Este paso es necesario hacerlo antes de realizar el volcado de los respectivos diseños a la tarjeta, debido a que podrían existir errores en los mismos que podrían estropearla.

Al igual que el resto de herramientas, tenemos varias opciones para acceder a la herramienta de simulación:

- La primera de ellas consiste en acceder al menú desplegable 'Processing' y elegir la opción 'Simulator Tool', como se ve en la Figura 12.

- La segunda opción consiste en iniciar directamente la simulación pulsando sobre el icono asignado para ello. 

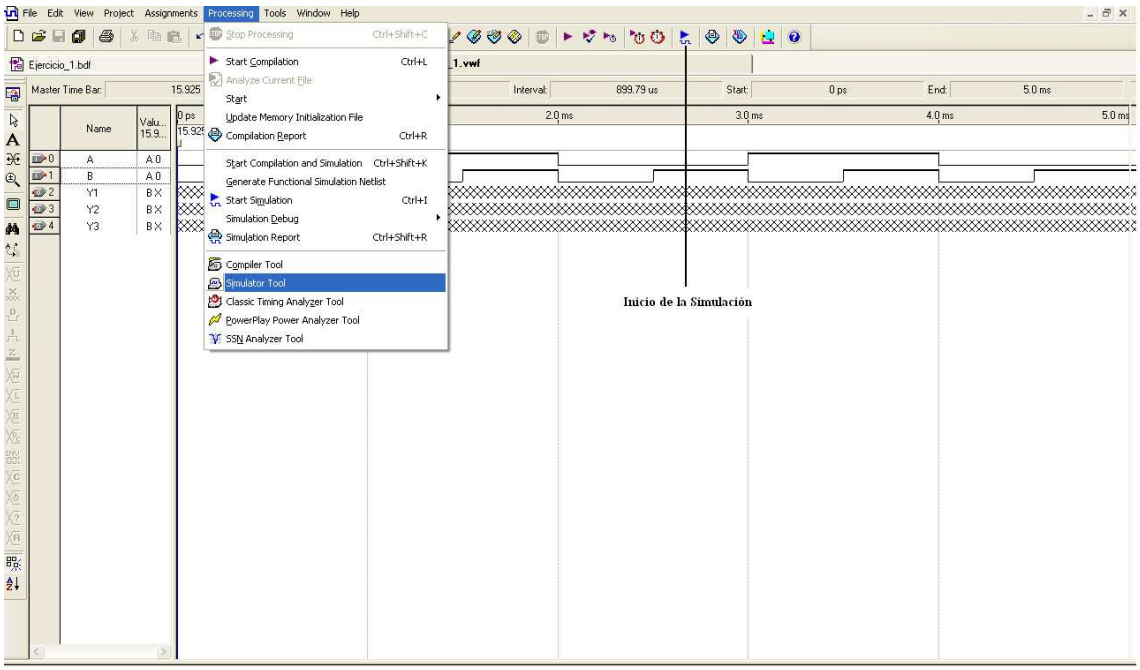


Figura 12. Acceso a la herramienta de simulación

Se recomienda realizar el acceso mediante la primera de las maneras ya que de esta forma tenemos acceso a la ventana de la herramienta, mostrada en la Figura 13.
Basta con pulsar el botón 'Start' para que comience la simulación.

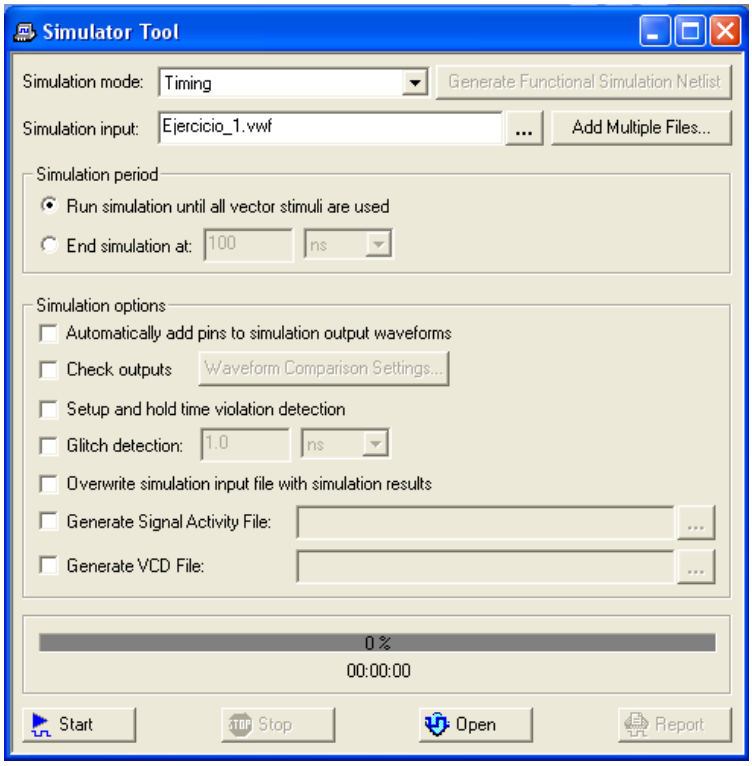


Figura 13. Ventana del simulador

Una vez terminada la simulación estaremos en disposición de poder ver los resultados. Para ello, en la misma ventana del simulador existen los botones 'Open' y 'Report'.

Si pulsamos sobre el botón 'Open' se modifica y se guarda el archivo original creado desde el Editor de Señales, pudiendo observar en el mismo tanto las entradas como las salidas resultado de la simulación.

Si pulsamos sobre el botón 'Report' podremos observar el resultado de la simulación (entradas y salidas), pero el archivo original creado con el Editor de Señales permanecerá sin cambios.

Comprobar que los resultados son los correctos.

3.2.- PUERTAS NAND, NOR Y XOR

En este segundo ejercicio el diseño a introducir será el siguiente:

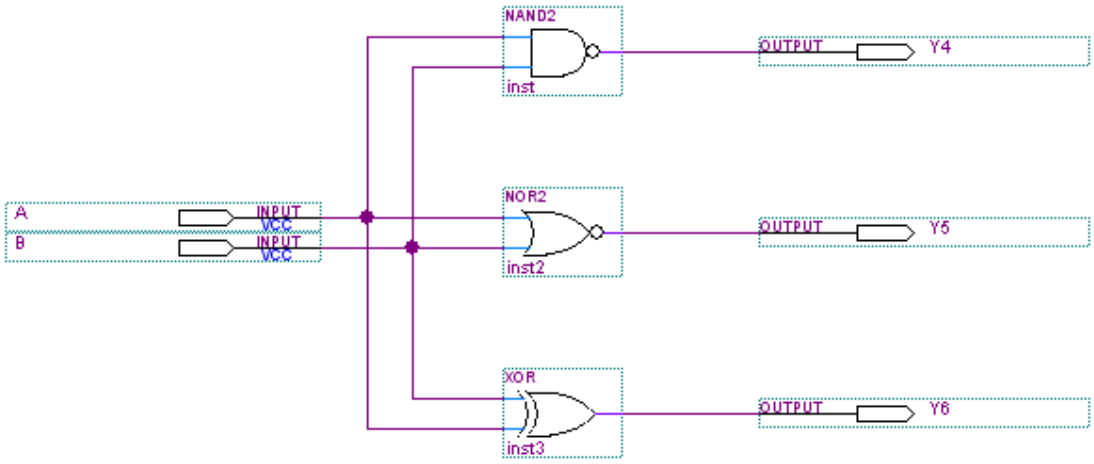


Figura 14. Diseño Ejercicio 3.2

Realizar los mismos pasos que en el apartado anterior y comprobar que los resultados son correctos y que coinciden con los obtenidos en la práctica cero.

4.- EJERCICIOS DE DISEÑO

Implementar los problemas de diseño de la práctica cero con el software Quartus II. Realizar las correspondientes simulaciones y ver que los resultados se ajustan a lo obtenido en la práctica anterior.

4.1.- Veredicto de un Tribunal

Construir un circuito capaz de decidir el veredicto de un tribunal manteniendo en secreto el voto de sus componentes. El tribunal está formado por un presidente y tres vocales. Cada uno de los cuatro miembros del tribunal dispone de un botón mediante el cual votará (SI = '1', NO = '0'). La salida del circuito consistirá en una luz indicadora que se encenderá cuando la mayoría de los miembros vote SI (MAYORÍA "SI" $\Rightarrow$ SALIDA a '1'). En caso de empate, el presidente tiene el voto de calidad.

Sólo se podrá usar un máximo de ocho puertas AND de dos entradas y seis puertas NOT.

4.2.- Función XOR

Realizar la función XOR, utilizando para ello un máximo de cuatro puertas NAND.

4.3.- Circuito Combinacional

Diseñar un circuito combinacional que sea capaz de:

a) Detectar un error en la codificación de un número decimal en BCD. Es decir, que si la combinación introducida a sus entradas no representa ningún número decimal codificado en BCD su salida E se encuentre a nivel alto ('1' lógico).

b) Detectar que el número decimal, en BCD, introducido en sus entradas, no está comprendido entre 3 y 7. Es decir, si el número es menor que 3 o mayor que 7 la salida R se pondrá a nivel alto.

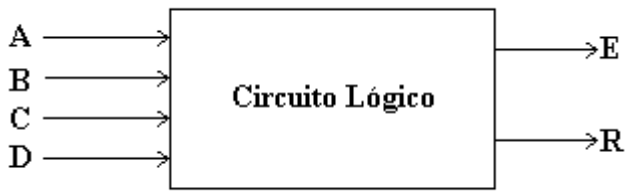


Figura 15. Circuito combinacional

Se emplearán un máximo de ocho puertas NOR para la implementación del diseño.

Para cada uno de los tres ejercicios de diseño se deberá entregar:

- Tabla de verdad
- Función lógica (Karnaugh)
- Esquema del circuito
- Cronograma o simulación, sobre una base de tiempos que muestre todos los posibles estados de las entradas y salidas.

RESOLUCIÓN PRÁCTICA 1

A continuación se presentan las soluciones de los diferentes ejercicios que componen esta práctica.

3.1.- PUERTAS NOT, AND Y OR

El diseño a introducir en el siguiente:

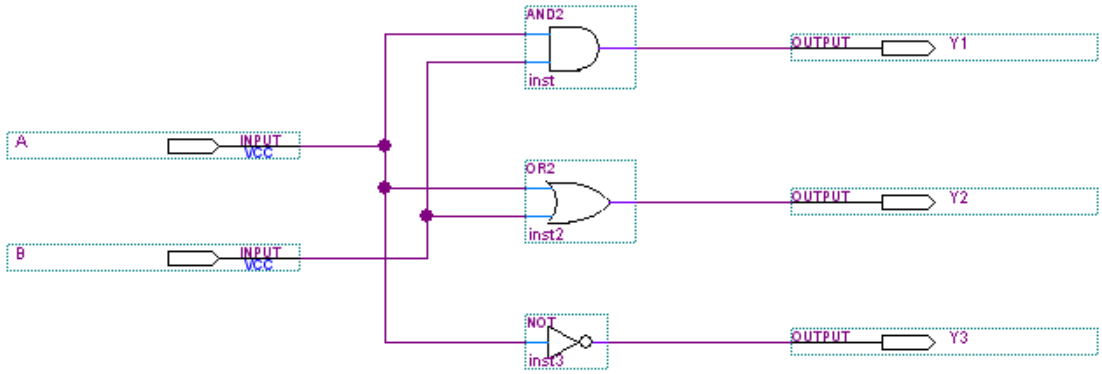


Figura 1. Circuito Ejercicio 3.1

Tras realizar la compilación del diseño, se procede a la simulación, obteniendo los siguientes resultados:

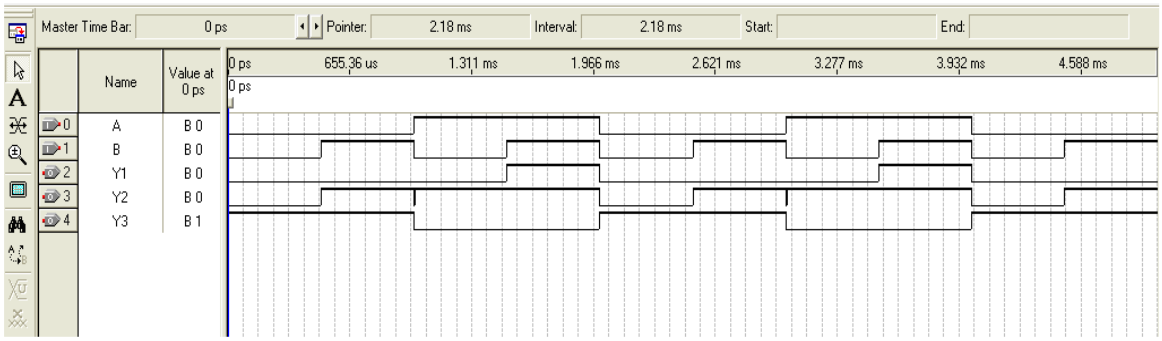


Figura 2. Resultados simulación Ejercicio 3.1

A partir de los resultados de la simulación, se obtiene la tabla de verdad del circuito:

A	B	Y1	Y2	Y3
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Tabla 1

Si analizamos la tabla de verdad vemos que efectivamente, los resultados son los esperados:

- Y1 se corresponde con el producto (puerta AND) $A \cdot B$
- Y2 se corresponde con la suma (puerta OR) $A + B$
- Y3 se corresponde con $\overline{A}$ (puerta NOT)

Observaciones:

En la simulación se puede ver un pico de nivel bajo en la salida 'Y2', que justamente coincide con los flancos de subida y bajada de las señales 'A' y 'B' respectivamente.

La señal 'Y2' corresponde a la salida de la puerta OR y este pico es debido a que hay un instante en el que la señal 'B' ha cambiado a nivel bajo antes de que la señal 'A' haya cambiado a nivel alto, durante la transición de estados lógicos.

Este efecto se puede apreciar con detalle en la siguiente Figura.

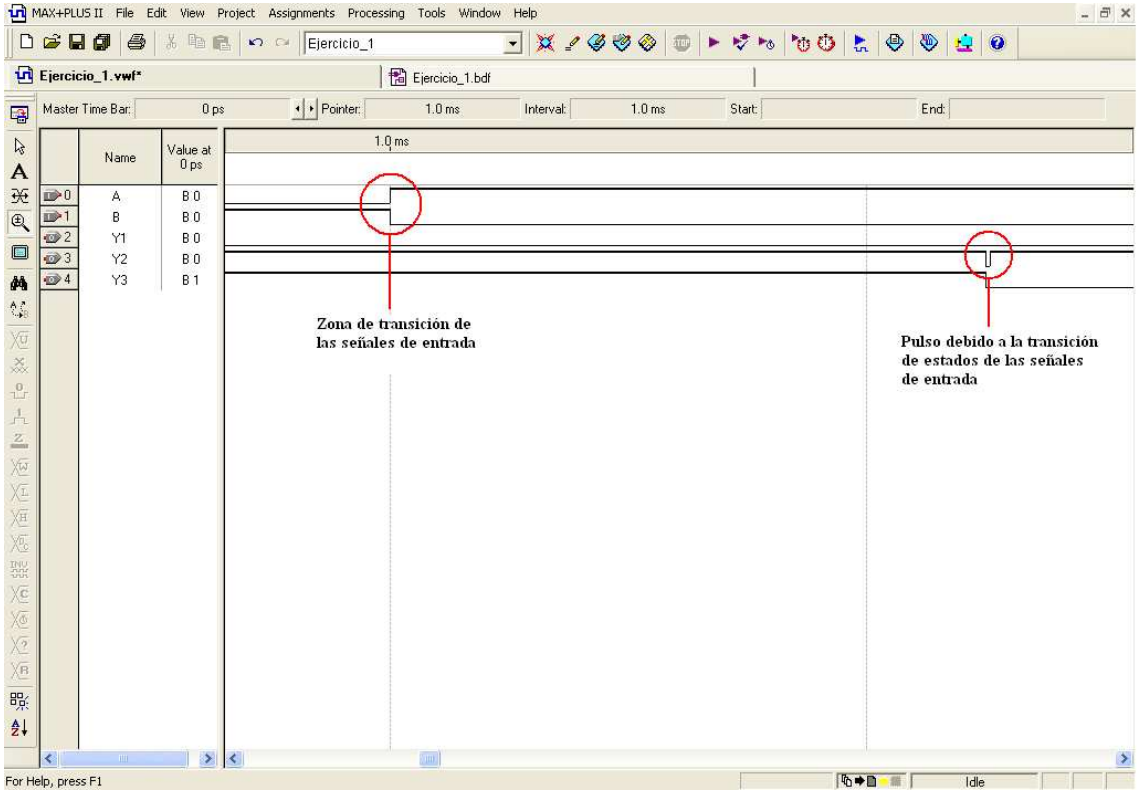


Figura 3. Efectos de la transición de estados

Como se puede apreciar en la Figura 3, el pulso aparece con el retardo que introduce la puerta OR.

3.2.- PUERTAS NAND, NOR Y XOR

El diseño a introducir para este ejercicio es el siguiente:

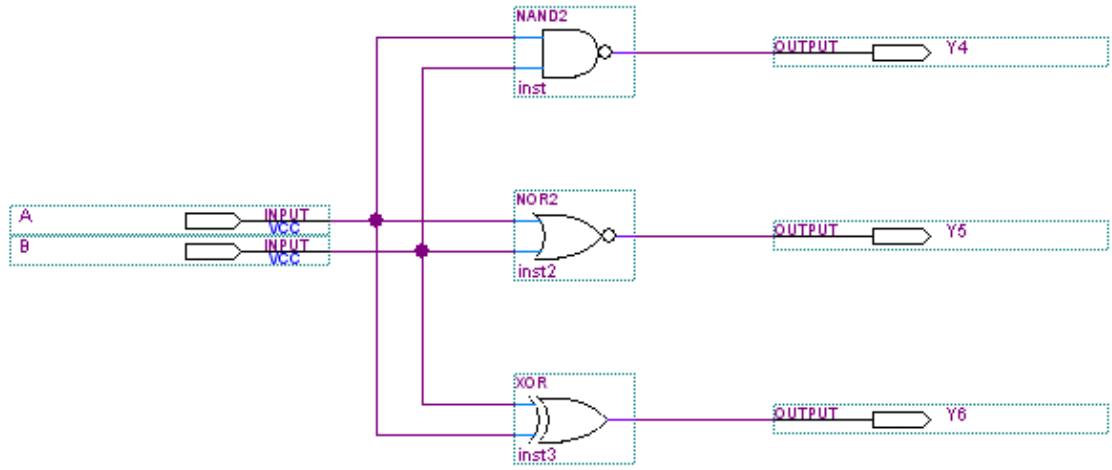


Figura 4. Circuito Ejercicio 3.2

Tras el proceso de simulación se obtienen los siguientes resultados:

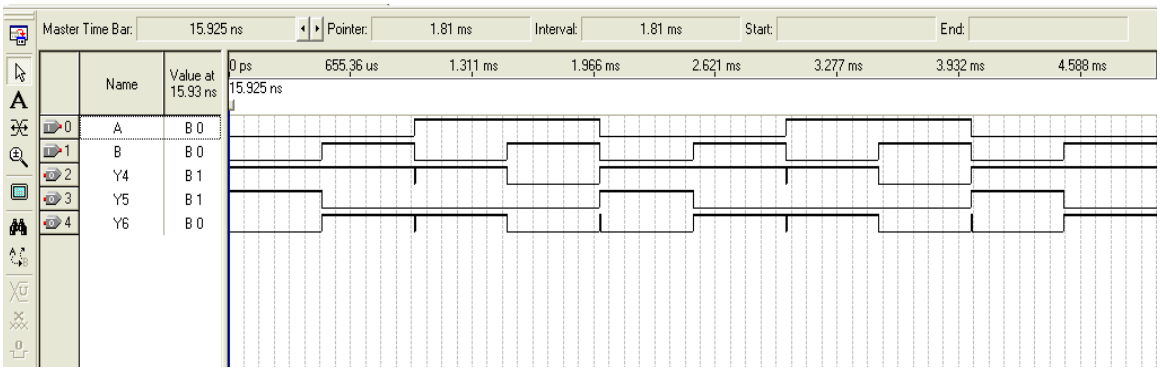


Figura 5. Resultado simulación Ejercicio 3.2

A partir de la simulación realizada se obtiene la siguiente tabla de la verdad:

A	B	Y1	Y2	Y3
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

Tabla 2

Viendo las combinaciones de la tabla de la verdad se observa que:

- Y4 corresponde al producto $\overline{A \bullet B}$ (puerta NAND)
- Y5 corresponde a la suma $\overline{A + B}$ (puerta NOR)
- Y6 corresponde a la función $A \oplus B$ (puerta XOR)

Observaciones:

En el resultado de esta simulación, el motivo de los pulsos que aparecen en las señales de salida 'Y4' e 'Y6' es el mismo que el comentado en el apartado anterior.

4.- EJERCICIOS DE DISEÑO

4.1.- Veredicto de un Tribunal

Considerando tres vocales y un presidente, hace un total de cuatro entradas las cuales se denominan A, B, C, y D, siendo A la correspondiente al presidente y que tiene valor doble en caso de empate. Por tanto, la tabla de verdad queda de la siguiente manera:

D	C	B	A	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Tabla 3

A continuación, tomando las salidas con valor igual a uno, se obtiene la función que representa en circuito mediante el método de Karnaugh, tal como se muestra:

DC/BA	00	01	11	10
00	0	0	1	0
01	0	1	1	0
11	0	1	1	1
10	0	1	1	0

Por tanto, queda la siguiente función:

$$Y = DA + CA + BA + DCB = \overline{\overline{DA} \cdot \overline{CA} \cdot \overline{BA} \cdot \overline{DCB}}$$

La representación del circuito mediante puertas lógicas es el mostrado en la Figura 6:

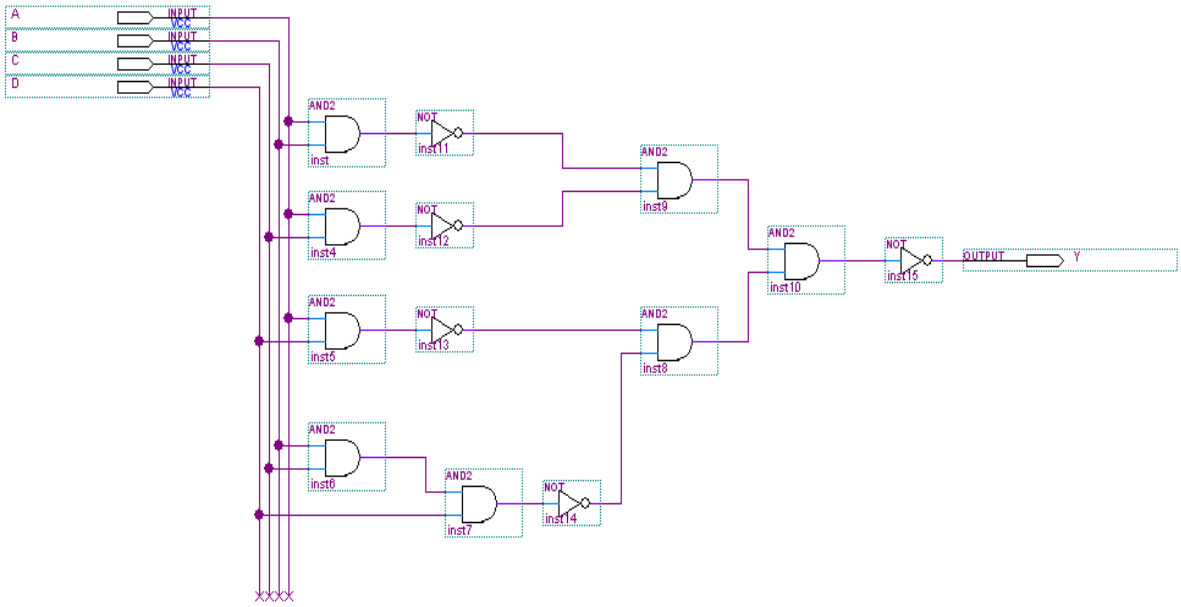


Figura 6. Circuito Ejercicio de diseño 1

A continuación se realiza una simulación mediante el Editor de Señales para comprobar el correcto funcionamiento del circuito. Se toman las señales de entrada con las siguientes frecuencias:

Señal D → 125 Hz
Señal C → 250 Hz
Señal B → 500 Hz
Señal A → 1KHz

Estos valores se toman para que no influyan los retardos de las puertas lógicas y así poder ver adecuadamente todas las combinaciones posibles de las señales de entrada.

Se puede optar por realizar la simulación con señales de distintas frecuencias a las utilizadas en este caso, pero teniendo siempre en cuenta que las frecuencias escogidas no sean excesivamente grandes ya que entonces los retardos sí que podrían influir en los resultados.

La simulación obtenida es la que se muestra a continuación en la Figura 7.

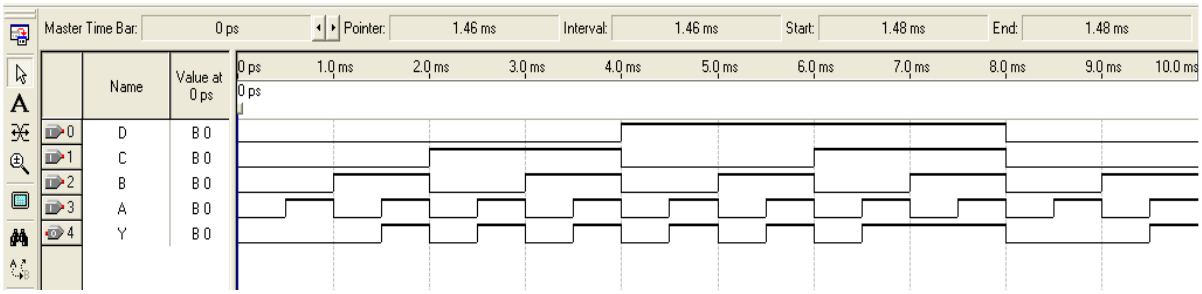


Figura 7. Resultado simulación Ejercicio de diseño 1

Como se puede comprobar, la señal de salida 'Y' coincide con la tabla de verdad del diseño.

4.2.- Función XOR

La tabla de verdad de una puerta XOR es la siguiente:

B	A	Z
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 4

La función resultante en base a la tabla anterior es:

$$Z = \overline{B}\,A + B\,\overline{A}$$

Como el ejercicio pide implementar la función sólo con puertas NAND, la función obtenida debe ser adaptada a ello.

La función 'Z' equivale a:

$$Z = \overline{B}\,A + A\,\overline{A} + B\,\overline{A} + B\,\overline{B} = (\overline{B} + \overline{A}) \cdot A + (\overline{B} + \overline{A}) \cdot B$$

Si ahora se niega la función dos veces y se aplica el Teorema de DeMorgan:

$$Z = \overline{\overline{(\overline{B} + \overline{A}) \cdot A + (\overline{B} + \overline{A}) \cdot B}}$$

$$= \overline{\overline{(\overline{B} + \overline{A}) \cdot A} \cdot \overline{(\overline{B} + \overline{A}) \cdot B}}$$

$$= \overline{\overline{BA} \cdot A} \cdot \overline{\overline{BA} \cdot B}$$

La función puede ser implementada únicamente con cuatro puertas NAND, ya que existe un término repetido en la misma.

El circuito lógico resultante es el mostrado en la Figura 8.

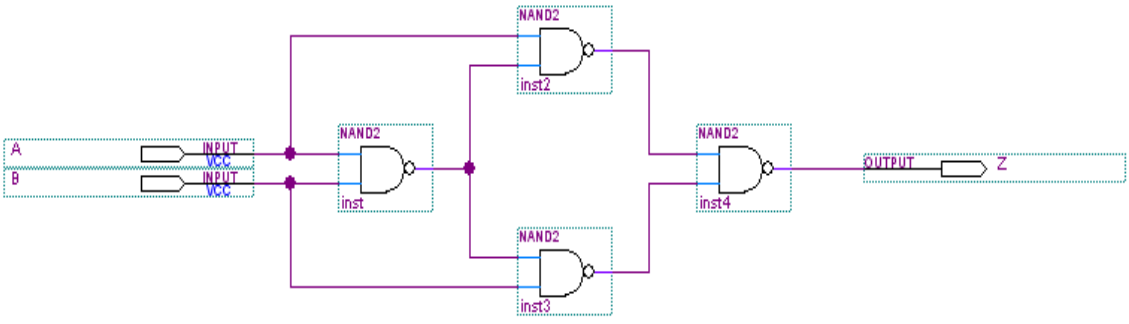


Figura 8. Circuito Ejercicio de diseño 2

Las frecuencias empleadas para las señales de entrada del circuito son las siguientes:

Señal B → 500 Hz
Señal A → 1 KHz

Realizado la correspondiente simulación, mostrada en la Figura 9, se puede comprobar como el resultado de la señal 'Z' coincide con el de la tabla de verdad del circuito y, por tanto, el diseño es correcto.

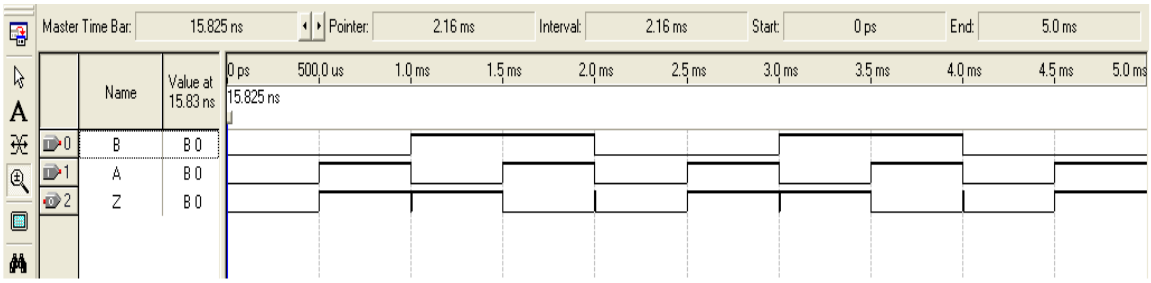


Figura 9. Resultado simulación Ejercicio de diseño 2

Como se puede apreciar en el resultado, vuelven a aparecer picos en la señal 'Z'. Las causas son las explicadas anteriormente.

4.3.- Circuito Combinacional

La tabla de verdad correspondiente al circuito combinacional existente en la Figura 15, junto lo que se nos pide en los respectivos apartados, es la siguiente:

D	C	B	A	E	R
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	1	x
1	0	1	1	1	x
1	1	0	0	1	x
1	1	0	1	1	x
1	1	1	0	1	x
1	1	1	1	1	x

Tabla 5

Como se puede apreciar, cuando la función 'E' nos indica que no estamos ante una codificación BCD, el valor de la función 'R' es irrelevante. Por tanto, cabe destacar que la función 'E' tiene prioridad respecto a la función 'R'.

Simplificando cada una de las funciones mediante el método de Karnaugh se obtiene lo siguiente:

Función E:

DC/BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

Función R:

DC/BA	00	01	11	10
00	1	1	0	1
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

Obteniendo las siguientes expresiones para las funciones 'E' y 'R':

$$E = DC + DB = D(C + B) = \overline{\overline{D(C + B)}} = \overline{\overline{D} + \overline{(C + B)}}$$

$$R = \overline{C}\,\overline{B} + \overline{D}\,\overline{C}A = \overline{C}(\overline{B} + \overline{D}A) = \overline{\overline{\overline{C}(\overline{B} + \overline{D}A)}} = \overline{C + \overline{(\overline{B} + \overline{(D + A)})}}$$

Observando las respectivas ecuaciones vemos que son necesarias 3 puertas NOR para la función 'E' y cuatro para la función 'R'. De esta manera, el circuito lógico resultante es el mostrado en la Figura 10.

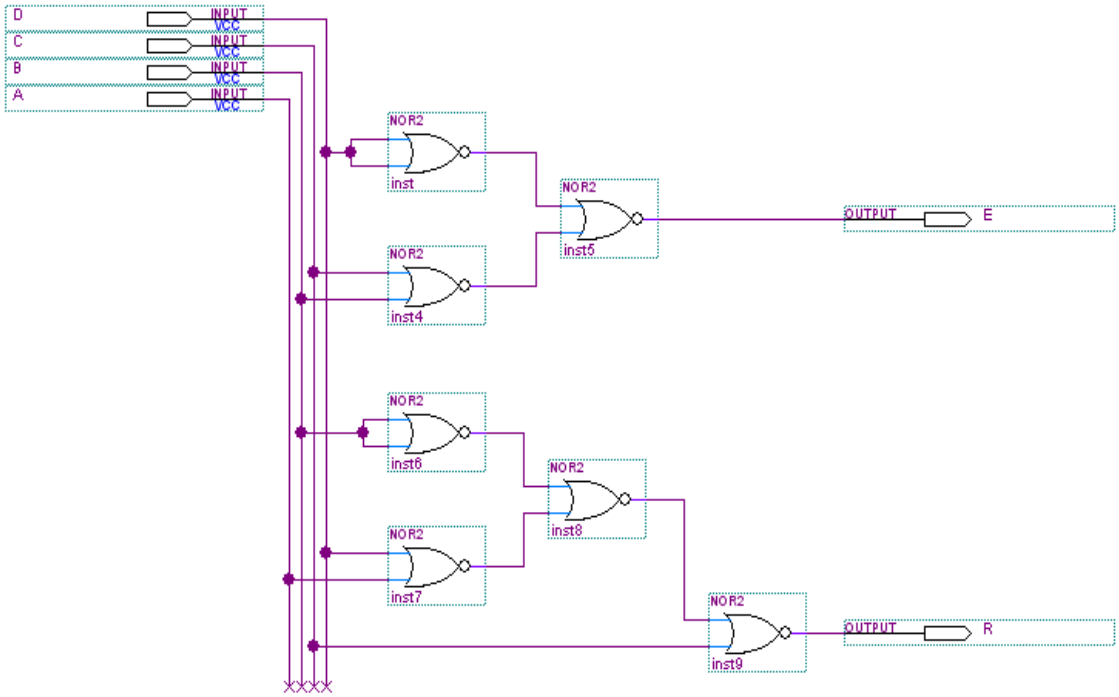


Figura 10. Circuito Ejercicio de diseño 3

La Figura 11 muestra el resultado de la simulación, donde las formas de onda coinciden con los valores de la tabla de la verdad.

Se toman las señales de entrada con las siguientes frecuencias:

Señal D → 125 Hz
Señal C → 250 Hz
Señal B → 500 Hz
Señal A → 1KHz

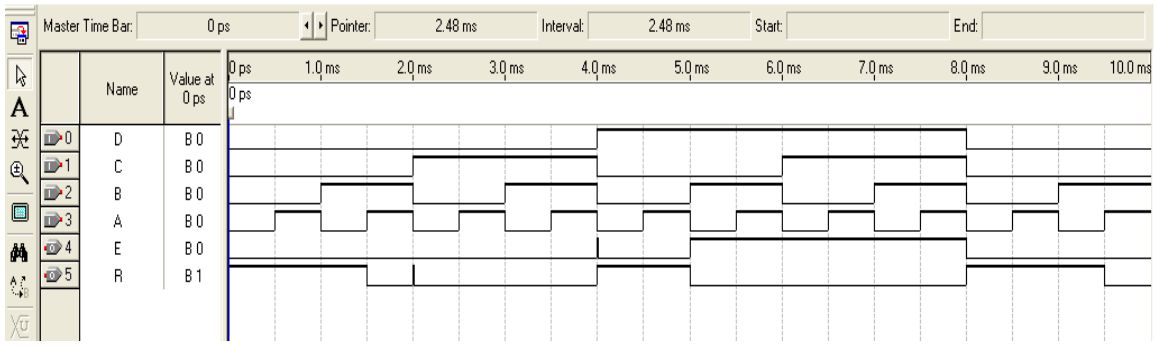


Figura 11. Resultado simulación Ejercicio de diseño 3

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

(1º Ingeniería Técnica en Informática de Gestión)

Práctica 2: Circuitos Aritméticos

Las reglas para las operaciones aritméticas básicas son comunes a todos los sistemas de numeración. La aritmética binaria permite realizar cualquier tipo de operación, la diferencia está en que el sistema binario es el empleado por los microprocesadores y éstos, solo son capaces de sumar y restar.

Hay que tener en cuenta que la suma es la base de las operaciones aritméticas. La resta se puede sustituir por la suma empleando números negativos, el producto es una sucesión de sumas, la división es una sucesión de restas, etc.

1.- OBJETIVOS

Mostrar algunos de los circuitos lógicos que realizan operaciones aritméticas básicas (suma, resta, multiplicación) con números binarios. Para ello se simularán distintos diseños realizados con el Editor Gráfico del software Quartus II.

2.- MATERIAL

- Ordenador personal con el software QUARTUS II®.
- Manual introductorio al software.
- Guión de prácticas.

3.- DESARROLLO DE LA PRÁCTICA

3.1.- SUMADORES BINARIOS

Los circuitos sumadores parten de un circuito sencillo llamado semisumador. Este circuito es capaz de sumar un bit más otro bit, pero no es capaz de tener en cuenta un acarreo anterior. Para corregir esta falta surge el sumador total, que sí es capaz de emplear el acarreo anterior. Partiendo de un sumador total y mediante la combinación del número necesario de los mismos es posible realizar sumas de más de un bit.

Las reglas para la suma binaria de dos bits son las siguientes:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0 \rightarrow$ Acarreo o carry = 1

La suma de dos números binarios de longitud mayor a un bit se realiza por parejas de bits. Hay que tener en cuenta la situación de sumar 1+1 donde se lleva 1 de acarreo. Este dato se debe incluir al realizar la suma de la pareja siguiente.

En esta práctica se van a realizar tres diseños, donde el primero de ellos se utilizará en el segundo y éste en el tercero. Debido a ello, es necesario crear el símbolo correspondiente a cada uno de los diseños, tal y como se explica en el apartado 3.1.1. A continuación, el símbolo puede ser insertado en el diseño que lo requiera.

3.1.1.- Semisumador

El semisumador es un circuito lógico que no es capaz de añadir a la suma un carry anterior. Este circuito suma dos bits, obteniendo a la salida la suma (S) de dichos bits y el posible acarreo (C).

Se comenzará este diseño definiendo un nuevo proyecto y abriendo un nuevo documento en el Editor Gráfico, diseñando el circuito semisumador mostrado en la Figura 1:

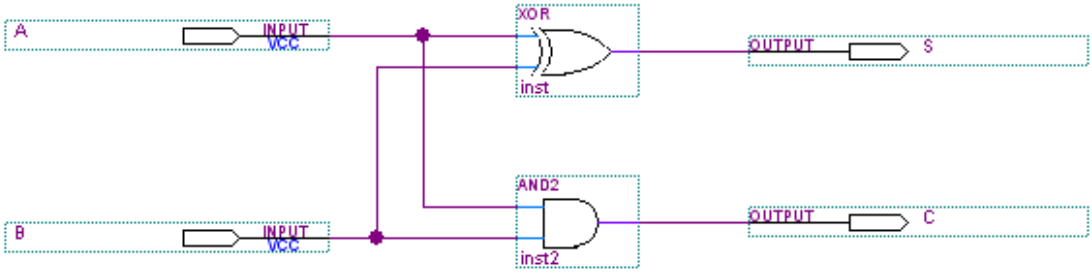


Figura 1. Circuito lógico del semisumador

En este diseño se emplean los símbolos XOR y AND2 (puerta AND de dos entradas), los cuales están disponibles en la librería 'primitives'.

Una vez realizado el circuito, se grabará el diseño y se compilará para ver si hay posibles errores, tal y como se hizo en la práctica anterior. A continuación, se llevará a cabo la simulación para ver si los resultados son los esperados. Si no lo son, se debe tener en cuenta que el fallo debe estar en el diseño. Un error habitual es que aparezca un mensaje de precaución (marcado en verde) a la hora de compilar, el cual permite seguir con la simulación, pero puede que los resultados no sean los esperados.

Se comprobarán los resultados obtenidos conforme a la tabla de la verdad del semisumador que se muestra a continuación:

B1	A1	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabla 1

Además, una vez compilado el diseño se procederá a la creación de su correspondiente símbolo, para poder ser utilizado en los siguientes diseños, tal y como se explica a continuación.

La creación de un símbolo bien puede ser en base a un diseño realizado con el Editor Gráfico o bien con el Editor de Texto. Para ello, una vez introducido, guardado y compilado el diseño, se accede al menú desplegable 'File' y se escoge la opción 'Create/Update' seguido de la opción 'Create Symbol Files for Current File', tal y como se puede ver en la siguiente ventana.

**Nota: El nombre elegido para cada uno de los diseños tiene que ser individual, sin poder repetirse, para que en el momento que interese crear un nuevo símbolo para un diseño no se tenga ningún problema. Asimismo, no se puede nombrar a los diseños con nombres ya ocupados por símbolos de las librerías, ni utilizar la letra 'ñ' en los mismos.*

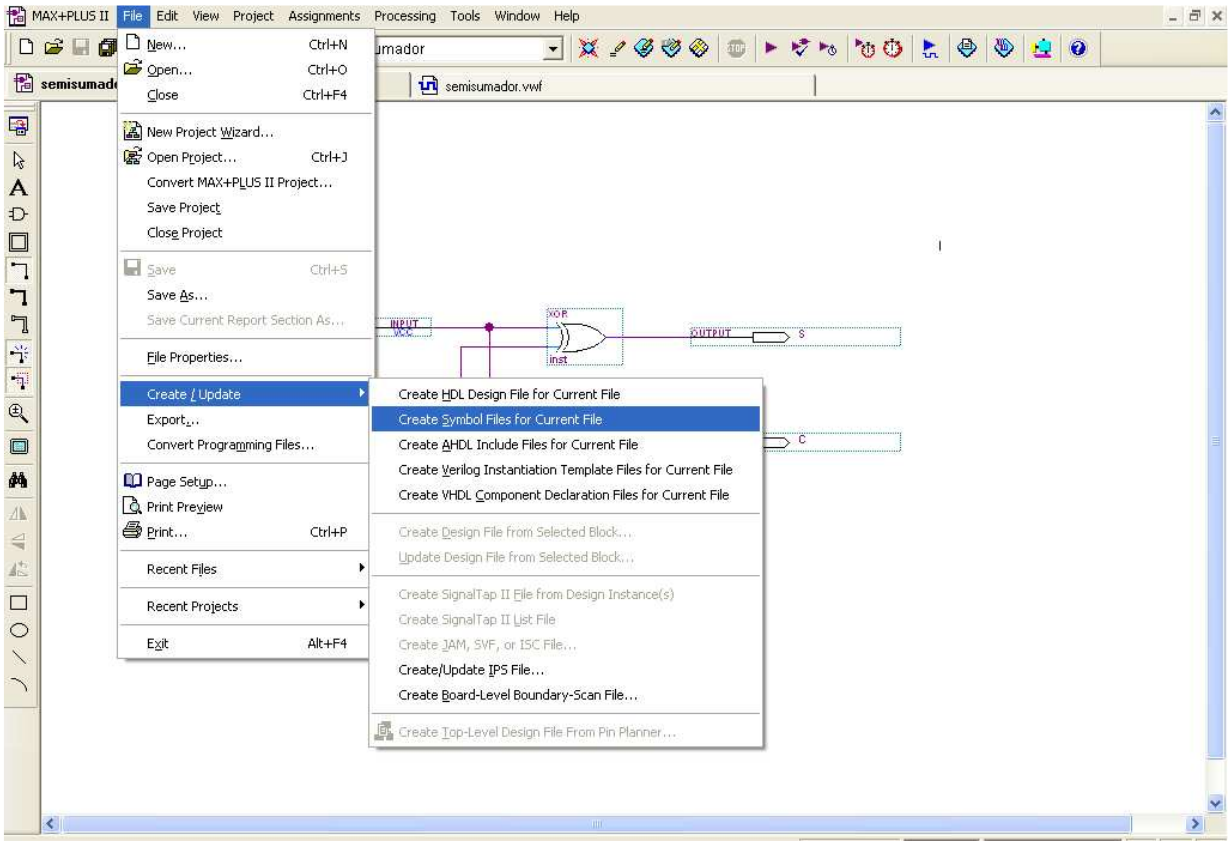


Figura 2. Acceso a la creación de símbolos

De esta manera se generará automáticamente un símbolo con las entradas y salidas del diseño, el cual se comportará de la misma forma, y tendrá el nombre del diseño con extensión '.bsf'. Este nuevo símbolo podrá ser utilizado en posteriores diseños de igual manera que los símbolos que ya existían en las librerías. Por ello, para la inclusión de este símbolo basta con acceder a la opción 'Insert Symbol' que aparece en el menú desplegable al pulsar con el botón derecho sobra la pantalla del Editor Gráfico.

- Importante: Para usar el símbolo creado en otro diseño es fundamental que éste se encuentre dentro del mismo proyecto. Si se crea otro proyecto se deberá incluir el archivo de diseño correspondiente al símbolo a la hora de definir este nuevo proyecto (por ejemplo, archivo con extensión '.bdf' correspondiente al Editor Gráfico).

El símbolo generado es el que se muestra en la Figura 3. Es posible cambiar las propiedades del símbolo (tamaño, nombres de las entradas y salidas, etc) desde el Editor de Símbolos. Para profundizar en ello se puede consultar el manual introductorio al software.

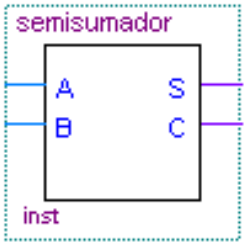
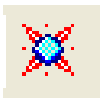


Figura 3. Símbolo del semisumador

Una forma de ver si el símbolo está creado o no, es pulsando sobre el símbolo de la derecha denominado 'Project Navigator' que representa jerárquicamente el diseño.



3.1.2.- Sumador completo

El semisumador sólo sirve para sumar dos bits. La suma de números binarios de más de un bit no se puede hacer sólo con semisumadores, ya que hay que tener en cuenta el posible acarreo de la pareja anterior. Para solucionar esto será necesario otro circuito lógico, el sumador completo.

El sumador completo realiza la suma de dos bits (A y B), teniendo en cuenta el acarreo (Ci) de la pareja anterior. A la salida se obtiene la suma (S) y el posible acarreo (Co).

Se puede realizar un sumador completo mediante dos semisumadores y una puerta OR. El diseño del circuito lógico es el que se muestra en la Figura 4.

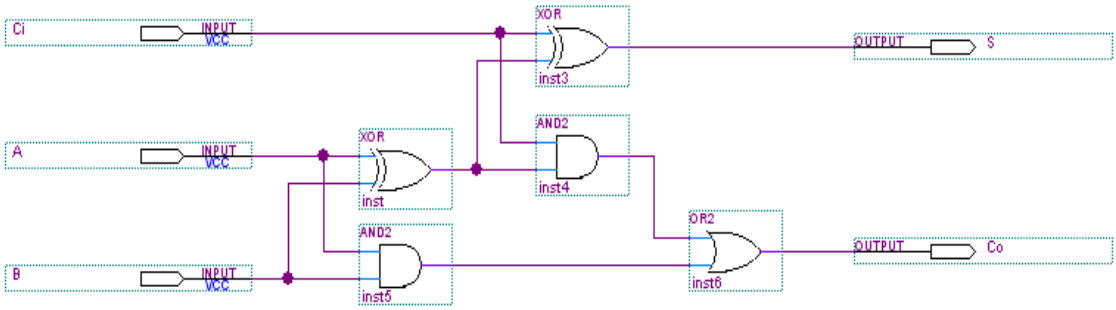


Figura 4. Circuito lógico del sumador completo

De cara a la implementación de circuitos más complejos, resulta más sencillo utilizar símbolos previamente creados. En este caso es posible utilizar el símbolo del semisumador generado en el apartado anterior. Para poder utilizar este símbolo, crearemos un nuevo archivo en el Editor Gráfico dentro de un nuevo proyecto (recordar guardar cada proyecto en un nuevo directorio) en el que ha sido incluido el semisumador. De esta manera, el diseño queda como se muestra en la Figura 5.

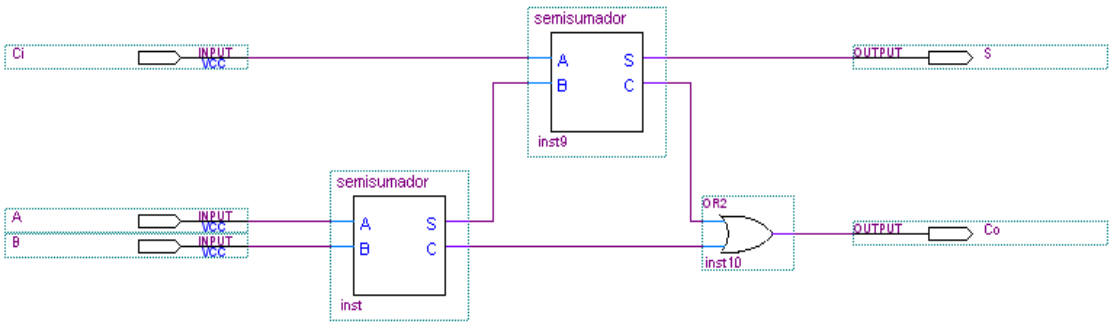


Figura 5. Sumador completo mediante la inclusión del símbolo semisumador

Una vez compilado, se simulará el sumador completo y se comprobarán que los resultados coinciden con los de su tabla de la verdad, mostrada a continuación:

Ci	B1	A1	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabla 2

En este caso, también se creará el símbolo que representará al sumador completo, que será usado en el siguiente apartado.

3.1.3.- Sumador binario paralelo de números de cuatro bits con entrada de acarreo

La suma de dos números de N bits se puede realizar con N-1 sumadores completos y un semisumador (para la pareja menos significativa) conectados en cascada. Sin embargo, si queremos un sumador para dos números de N bits con entrada de acarreo previo, son necesarios N sumadores completos.

```
                          C_in
      A4    A3    A2    A1
+
      B4    B3    B2    B1
_____________________________
C_out S4    S3    S2    S1
```

El circuito lógico que representa a este sumador se ve en la Figura 6.

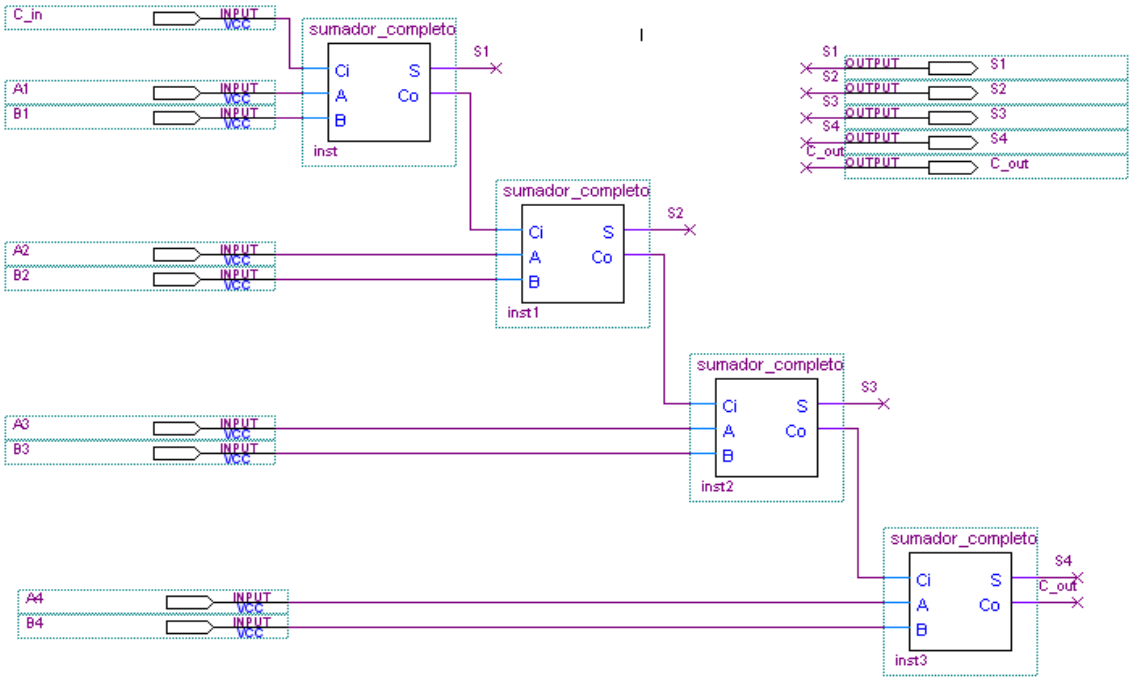


Figura 6. Circuito lógico del sumador paralelo de cuatro bits y entrada de acarreo

Como se puede observar en la figura anterior, existen dos maneras de realizar las conexiones. En primer lugar, uniendo los distintos elementos mediante hilos y, en segundo lugar, dando el mismo nombre al origen y destino de la conexión.

Se realizará este nuevo diseño generando un nuevo proyecto, donde se incluirán los archivos generados anteriormente (semisumador y sumador completo) para así poder incluir sus respectivos símbolos sin problemas.

Nuevamente se procederá a compilar el diseño y se utilizará el simulador para comprobar las siguientes operaciones:

a) C_in = 0; B = 0011; A = 0001;
b) C_in = 0; B = 1011; A = 0110;
c) C_in = 1; B = 0010; A = 0101;
d) C_in = 1; B = 0111; A = 1100;

Para realizar la simulación de las operaciones que se piden, es necesario introducir a mano los valores de las señales, utilizando para ello los iconos ‘0’ y ‘1’ de la barra de herramientas situada en el margen izquierdo del Editor de Señales, tal y como se ve en la Figura 7.

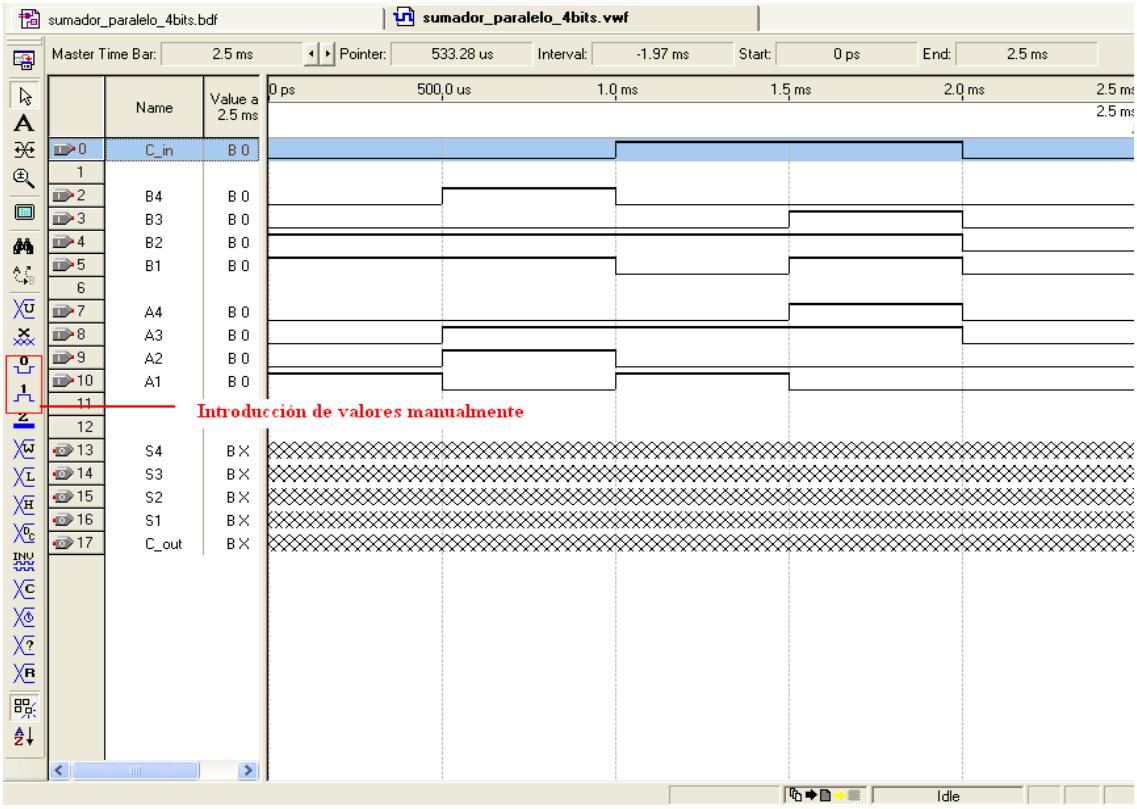


Figura 7. Introducción de las señales para las operaciones

A la hora de introducir valores manualmente es conveniente activar la opción 'Snap to Grid' situada en el menú desplegable 'View'. De esta manera se consigue forzar el valor introducido al tiempo del periodo.

También se deberá crear un símbolo de este diseño, el cual será empleado posteriormente.

- Circuitos Restadores:

Este tipo de circuitos serán realizados mediante sumadores, empleando bien el convenio CA1 o bien el convenio CA2.

$$\text{RESTA} \rightarrow B - A = B + (-A) = \begin{Bmatrix} B + CA1(A); ConvenioCA1 \\ B + CA2(A); ConvenioCA2 \end{Bmatrix}$$

4.- EJERCICIOS DE DISEÑO

4.1.- Sumador de palabras de ocho bits

Diseñar un circuito capaz de sumar dos números o palabras de ocho bits cada una, utilizando para ello el dispositivo 7483, que es un sumador binario de cuatro bits.

Para la comprobación del circuito, realizar la simulación y efectuar las siguientes operaciones:

a) 23+48
b) 12+122+1
c) 64+128
d) 252+182

¿Cuál es el número más elevado que se puede obtener a la salida?

4.2- Multiplicador binario paralelo de dos bits

Diseñar un circuito combinacional con el que se pueda obtener el producto (multiplicación) de dos números binarios de dos bits cada uno (A2 A1 y B2 B1) presentados en paralelo. Se utilizarán puertas AND y semisumadores, en concreto, seis puertas AND2 y dos puertas XOR.

La operación resultante para el diseño sería la siguiente:

		A2	A1
		B2	B1
		B1 A2	B1 A1
	B2 A2	B2 A2	
C	P2	P1	P0

Dibujar el circuito con el Editor Gráfico y comprobar su correcto funcionamiento.

4.3- Sumador / Restador (CA1)

Diseñar un Sumador / Restador para números binarios (A y B) de cuatro bits utilizando el complemento a uno (CA1) para la resta. El circuito tendrá una entrada de control C para seleccionar la suma o la resta (C=0 suma y C=1 resta). Para la realización del circuito se utilizará el símbolo del 'sumador binario paralelo de números de cuatro bits con entrada de acarreo' y cuatro puertas XOR.

Dibujar el circuito con el Editor Gráfico y comprobar su correcto funcionamiento con el simulador, realizando las siguientes operaciones:

a) 4 + 2
b) 3 – 1
c) 1 – 5

4.4- Sumador / Restador (CA2)

Diseñar un Sumador / Restador para números binarios (A y B) de cuatro bits utilizando el complemento a dos (CA2) para la resta. El circuito tendrá una entrada de control C para seleccionar la suma o la resta (C=0 suma y C=1 resta). Para la realización del circuito se utilizará el símbolo del 'sumador binario paralelo de números de cuatro bits con entrada de acarreo' y cuatro puertas XOR.

Dibujar el circuito con el Editor Gráfico y comprobar su correcto funcionamiento con el simulador, realizando las siguientes operaciones:

d) 4 + 2
e) 3 – 1
f) 1 – 5

RESOLUCIÓN PRÁCTICA 2. CIRCUITOS ARITMÉTICOS

A continuación se muestra la resolución de los diferentes ejercicios que componen esta práctica.

3.1.- SUMADORES BINARIOS

3.1.1.- Semisumador

El circuito a introducir es el que se muestra en la Figura 1.

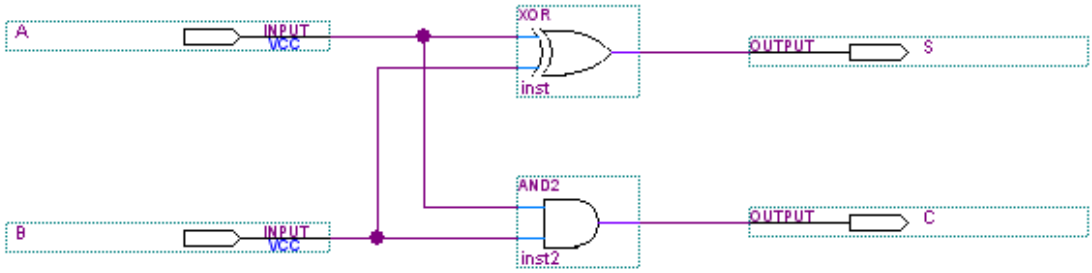


Figura 1. Circuito semisumador

Tras el proceso de simulación se obtienen los siguientes resultados:

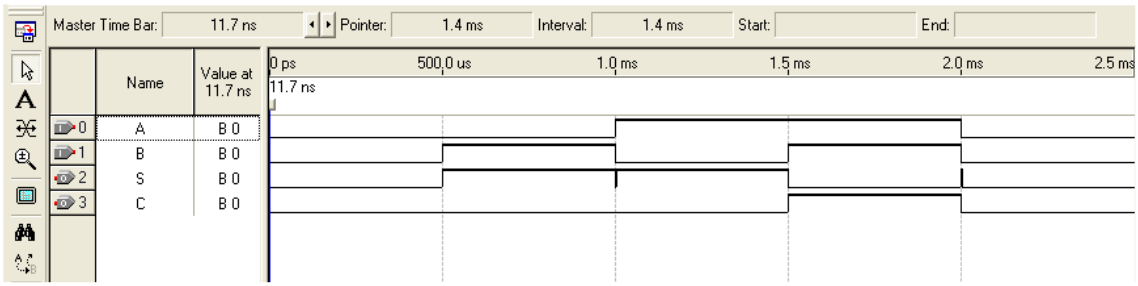


Figura 2. Resultados del semisumador

A partir de la simulación realizada se obtiene la tabla de la verdad mostrada a continuación, la cual coincide con la expuesta en el guión, correspondiente al semisumador:

B1	A1	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabla 1

3.1.2.- Sumador completo

El circuito lógico del sumador completo es el siguiente:

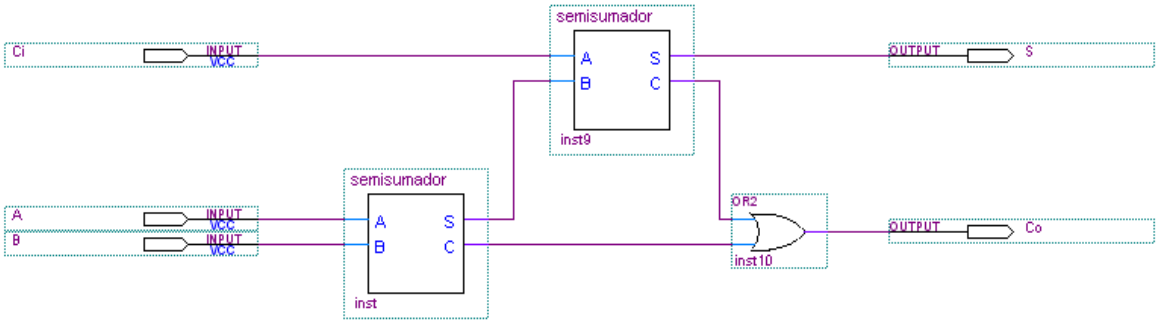


Figura 3. Circuito sumador completo

Los resultados obtenidos se muestran a continuación, en la Figura 4:

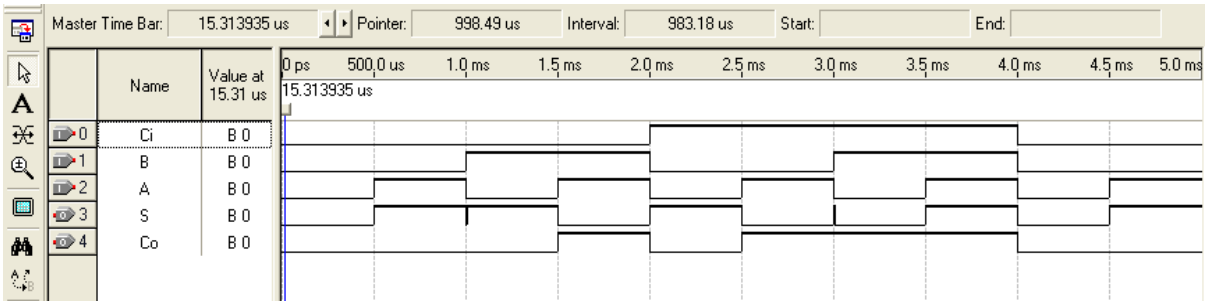


Figura 4. Resultados del sumador completo

Tras la simulación se obtiene la tabla de la verdad correspondiente a un sumador de dos bits con entrada de acarreo:

Ci	B1	A1	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabla 2

Como se puede comprobar, la tabla se corresponde con la del sumador completo.

3.1.3.- Sumador binario paralelo de números de cuatro bits con entrada de acarreo.

El diseño a introducir es el mostrado en la Figura 5.

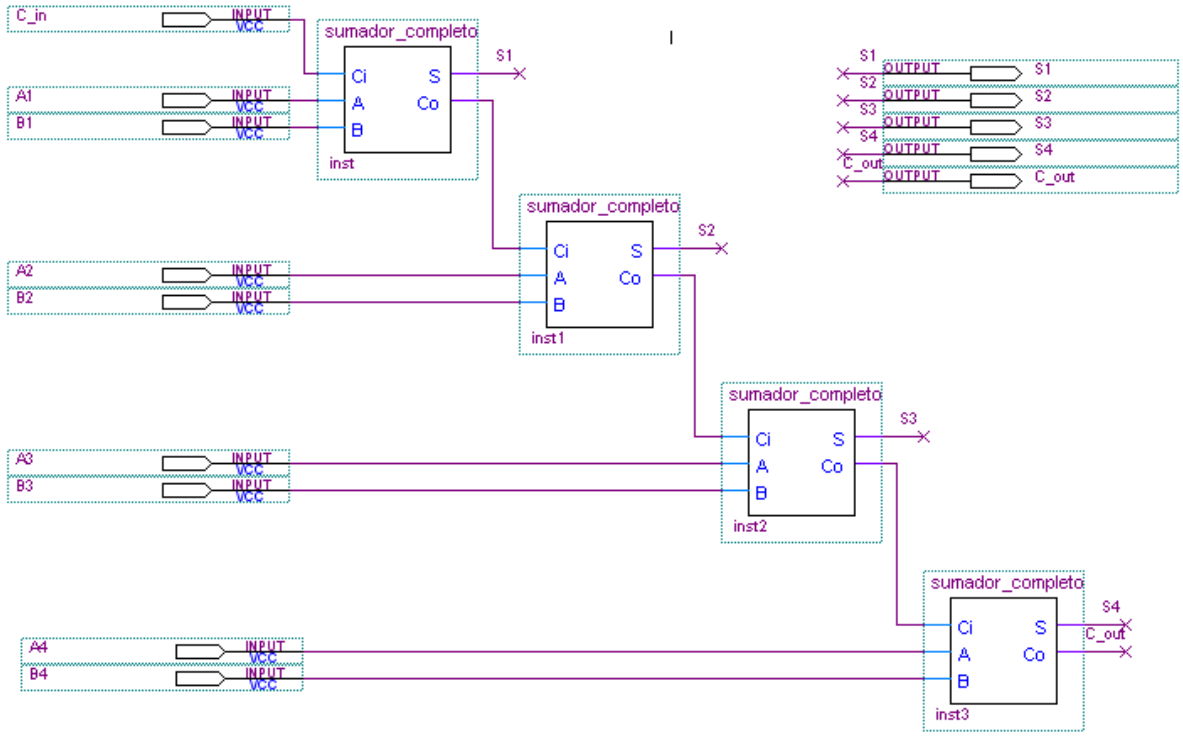


Figura 5. Circuito sumador paralelo de cuatro bits y entrada de acarreo

La suma que realiza este circuito es de la siguiente manera:

				C_in
	A4	A3	A2	A1
+				
	B4	B3	B2	B1
C_out	S4	S3	S2	S1

A continuación se muestran los resultados de los diferentes casos que se exigen como prueba:

a)	C_in	0
	B	0011
	A	0001
C_out=0		0100

b)	C_in	0
	B	1011
	A	0110
C_out=1		0001

c)

	C_in	1
	B	0010
	A	0101
C_out=0		1000

d)

	C_in	1
	B	0111
	A	1100
C_out=1		0100

A partir de estas operaciones, la simulación queda de la siguiente manera:

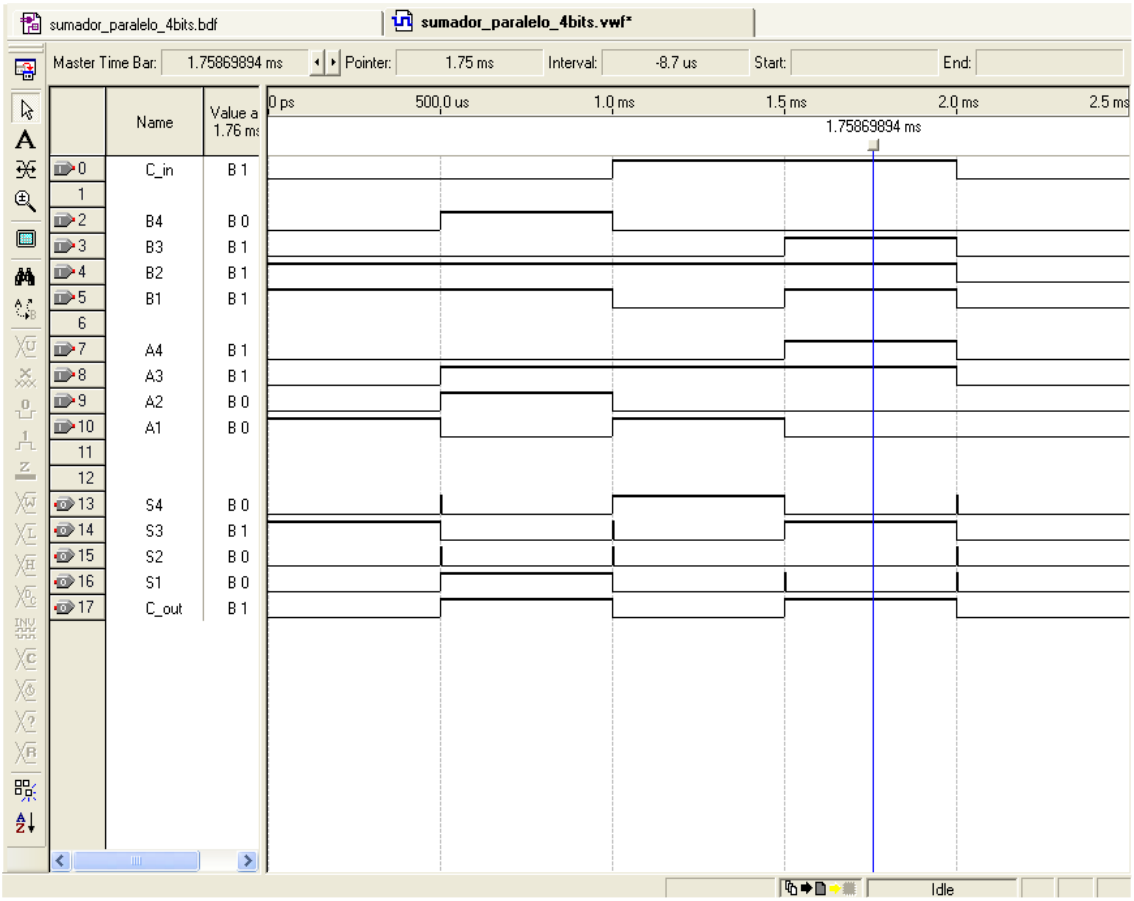


Figura 6. Resultados del sumador paralelo de 4 bits

Como se puede comprobar, los resultados coinciden con lo calculado anteriormente, luego es correcto.

4.- EJERCICIOS DE DISEÑO

4.1.- Sumador de palabras de ocho bits

Teniendo como base el dispositivo 7483, que es un sumador de palabras de cuatro bits, realizando la concatenación de dos de ellos es posible realizar el sumador binario de números de ocho bits. Por tanto, el diseño queda de la siguiente manera:

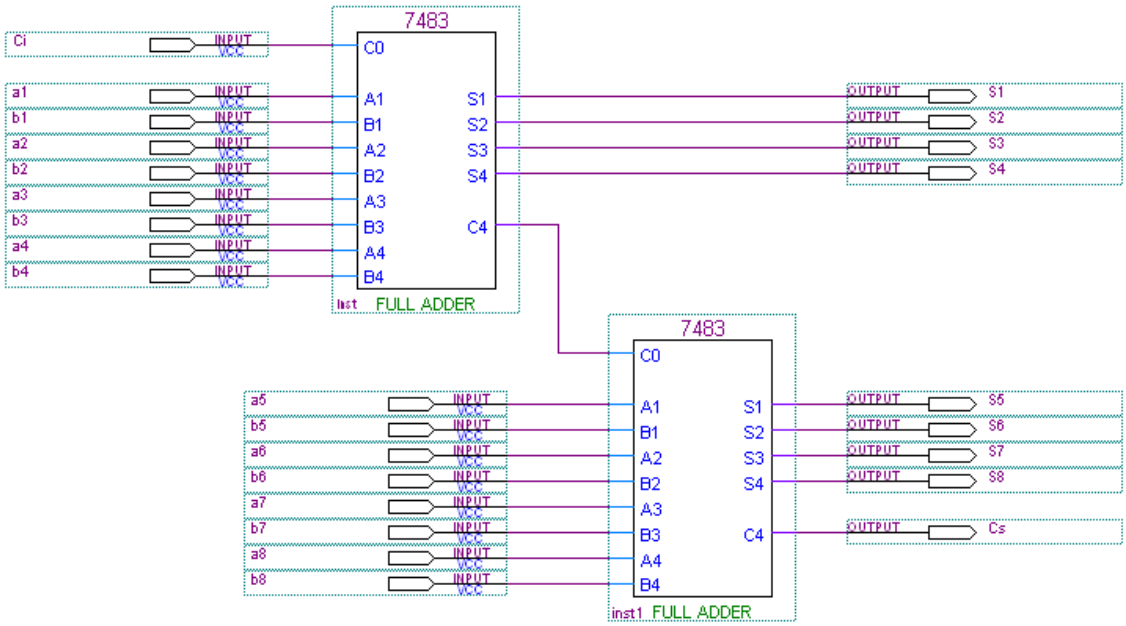


Figura 7. Circuito sumador de palabras de 8 bits

Las entradas ‘a1’ y ‘b1’ se corresponden con los bits de menor peso, mientras que las entradas ‘a8’ y ‘b8’ son los bits de mayor peso. Asimismo, la salida ‘S1’ es la de menor peso, mientras que la salida ‘S8’ es la de mayor peso.

Para comprobar el correcto funcionamiento del circuito se han tomado los siguientes ejemplos:

a) 23 + 48 = 71
b) 12 + 122 + 1 = 135
c) 64 + 128 = 192
d) 252 + 182 = 434

La figura 8 muestra el resultado de esta simulación, con el fin de poder ver los valores obtenidos.

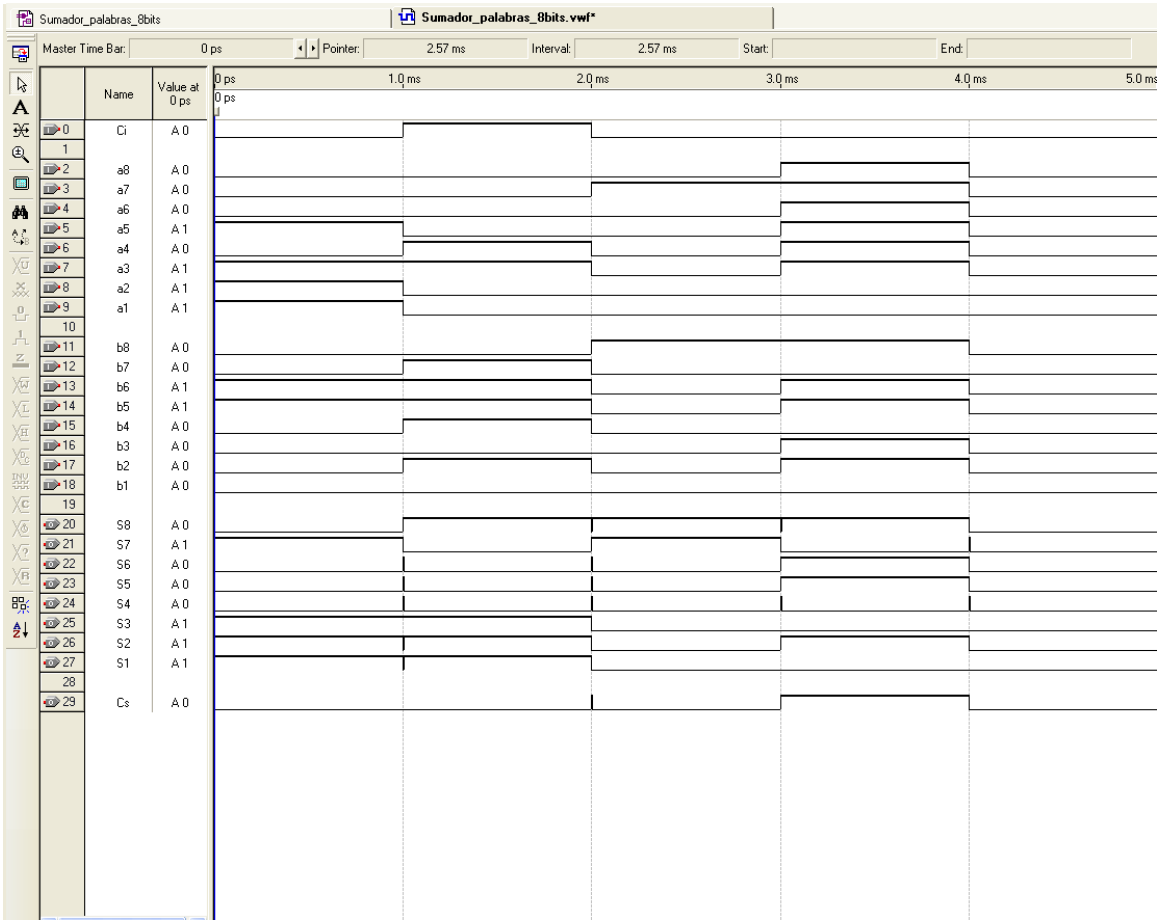


Figura 8. Resultados del sumador de palabras de 8 bits

Como se puede observar, en los tres primeros casos se obtiene el resultado esperado, mientras que el cuarto caso muestra el resultado de un desbordamiento, activando el carry de salida. Esto es debido a que el mayor número que se puede obtener a la salida es $2^8 = 256$, como consecuencia de disponer de ocho salidas de datos.

Las señales de entrada en esta simulación han de ser insertadas a mano.

4.2- Multiplicador binario paralelo de dos bits

El producto de dos números binarios, A y B se corresponde con:

		A2	A1
		B2	B1
		B1 A2	B1 A1
	B2 A2	B2 A2	
C	P2	P1	P0

De esto se deduce lo siguiente:

P0 = B1 A1
P1 = B1 A2 + B2 A1
P2 = B2 A2 (más un posible acarreo procedente de P1)
C = posible acarreo final

A partir de las expresiones anteriores, el circuito estará formado por cuatro puertas AND para realizar los productos de las parejas de bits y dos semisumadores, donde el segundo estará conectado con la salida de acarreo del primero.

El circuito lógico correspondiente al multiplicador binario queda de la siguiente manera:

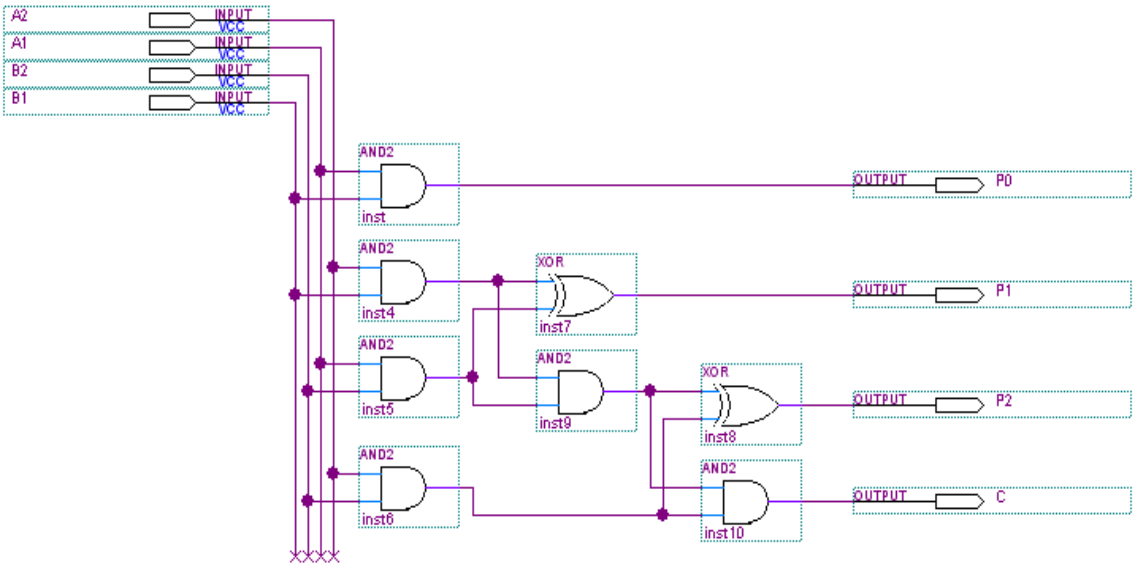


Figura 9. Circuito del multiplicador binario paralelo de dos bits

Como prueba se han tomado los siguientes ejemplos:

a)	B	1 0
	A	0 1
C=0		0 1 0

b)	B	1 1
	A	1 1
C=1		0 0 1

La simulación de estos ejemplos se muestra en la Figura 10.

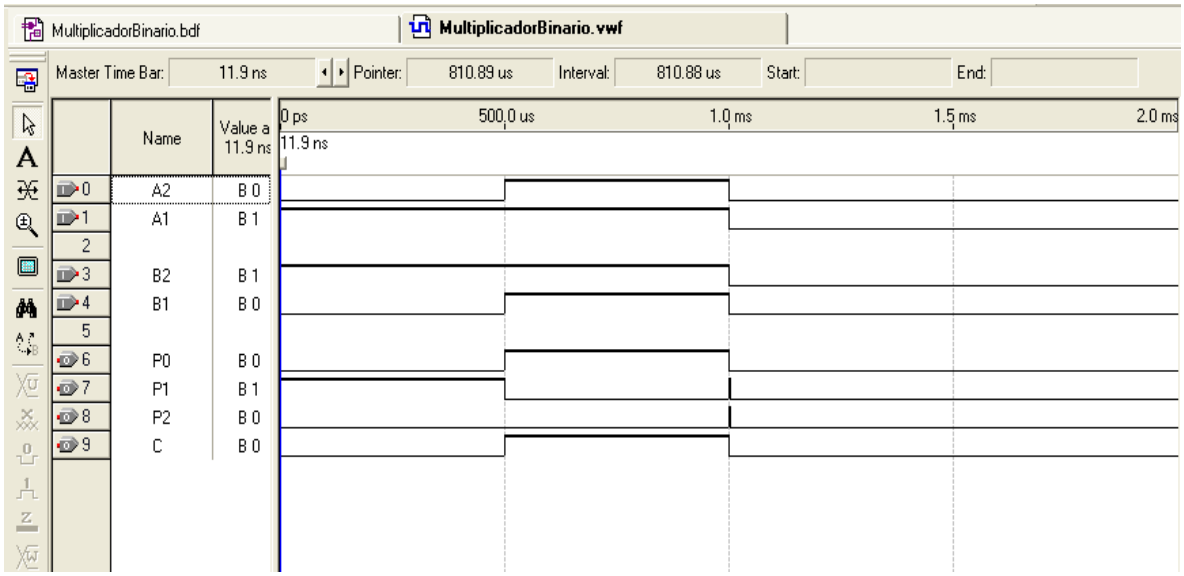


Figura 10. Resultados del multiplicador binario paralelo de dos bits

Como se puede observar, los resultados coinciden con lo esperado.

4.3- Sumador / Restador CA1

Este diseño está basado en el 'sumador binario paralelo de 4 bits' implementado en el apartado 3.1.3, el cual es un circuito que sólo es capaz de realizar sumas, aunque utilizándolo en el esquema representado en la Figura 11, se consigue un sumador / restador de dos números de cuatro bits.

Para sumar con este circuito, la entrada de control 'C' deberá estar a nivel bajo ('0'), de esta manera, esta entrada no añadirá nada a la suma y el conjunto de puertas XOR no alterarán el valor del número introducido en 'B'. Por lo tanto, la suma se realizará como si el circuito sumador binario paralelo de 4 bits estuviese solo.

En el caso de la resta, la entrada de control 'C' deberá ponerse a '1'. La resta es igual a la suma del minuendo más el opuesto del sustraendo. Para cambiar el signo del sustraendo se deberá hacer el CA1 del mismo, mientras que el minuendo permanecerá sin cambios. Para ello se utilizan cuatro puertas XOR, donde cada una de ellas contiene la señal 'C' y uno de los bits del sustraendo. De este modo se invertirán los bits del sustraendo.

También es importante tener en cuenta que en el caso de la resta, con el sustraendo en CA1, se puede generar un acarreo que debe sumarse al resultado.

El diseño se muestra en la Figura 11.

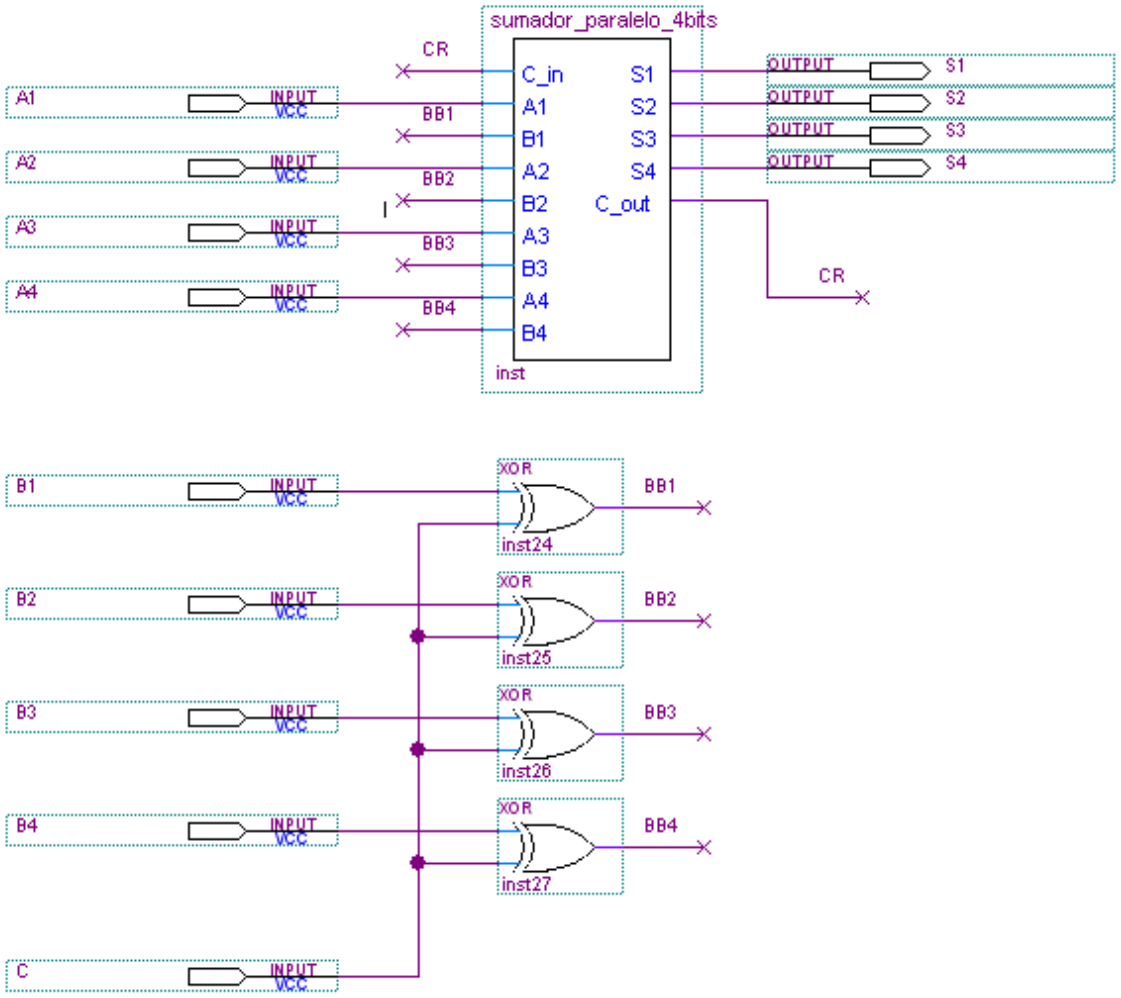


Figura 11. Circuito Sumador / Restador en CA1

Se comprobará el correcto funcionamiento del circuito mediante las siguientes operaciones:

a) 4 + 2 = 6 → 0010 + 0010 = 0110
b) 3 – 1 = 2 → 0011 + CA1(0001) = 0011 + 1110 = **1**0001 →0010
c) 1 – 5 = -4 → 0001 + CA1(0101) = 0001 + 1010 = 1011
-4 = CA1(0100) = 1011

En el apartado b) surge un valor de acarreo, que ha sido sumado al resultado.

Se debe tener cuidado a la hora de interpretar el resultado en casos como el del apartado c) ya que ante un resultado negativo, este viene expresado en CA1.

Para realizar la simulación, como en los anteriores casos, es necesario introducir a mano los datos de las operaciones, utilizando para ello los iconos que representan al '0' y al '1' situados en la barra de herramientas situada en el margen izquierdo del Editor de Señales.

La simulación de este diseño se muestra en la Figura 12.

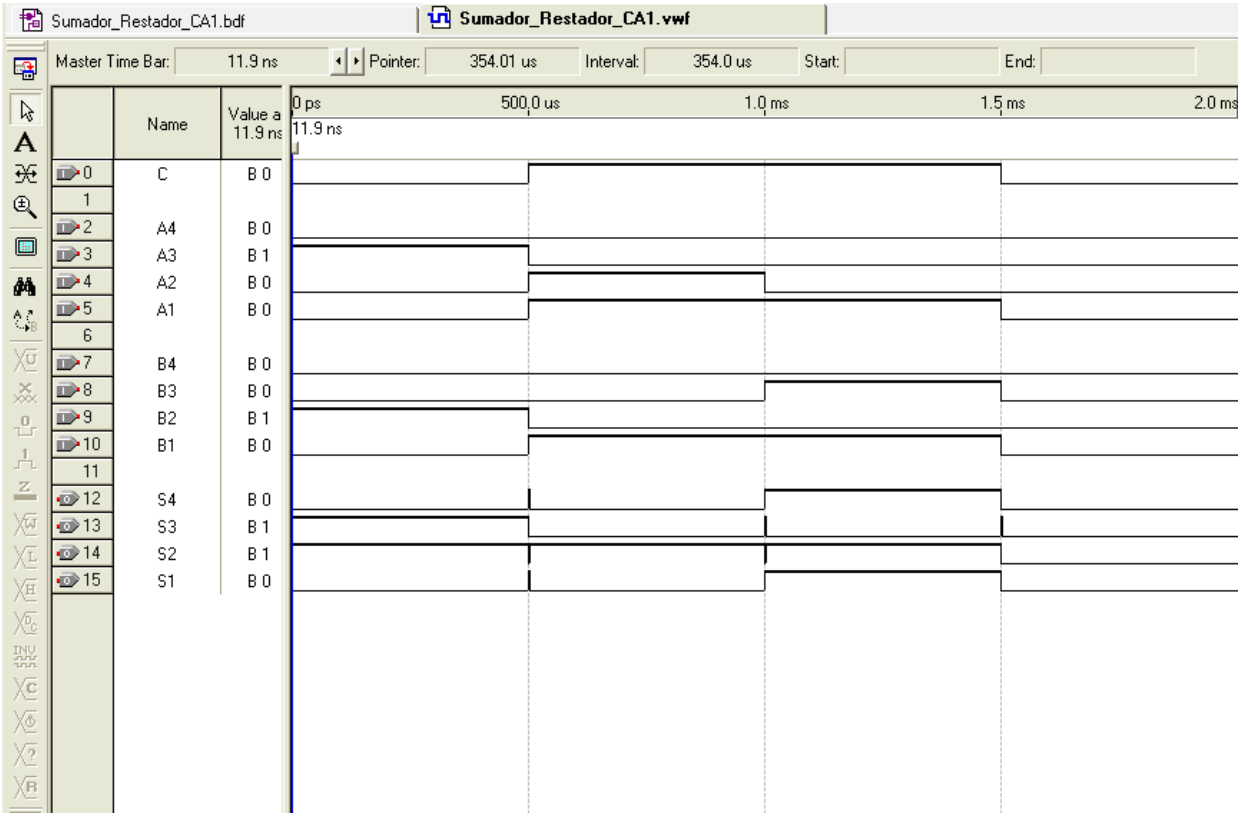


Figura 12. Resultados del sumador / restador CA1

Como se puede observar, los resultados coinciden con los obtenidos en las operaciones.

Un efecto destacable de esta simulación es que los picos que aparecen en las transiciones de las distintas operaciones son más exagerados que en los casos anteriores. Esto es debido a que la introducción de las señales se ha hecho a mano, por lo que los retardos son mayores.

4.4- Sumador / Restador en CA2

Al igual que en el caso anterior, el presente diseño está basado en el circuito 'sumador binario paralelo de 4 bits' implementado en el apartado 3.1.3, por lo tanto, en principio sólo capaz de realizar sumas, pero con la correspondiente modificación que también permite realizar la operación de restar.

Para sumar con este circuito, la entrada de control 'C' deberá estar a nivel bajo ('0'), de esta manera, esta entrada no añadirá nada a la suma y el conjunto de puertas XOR no alterarán el valor del número introducido en 'B'. Por lo tanto, la suma se realizará como si el circuito sumador binario paralelo de 4 bits estuviese solo.

En el caso de la resta, la entrada de control 'C' deberá ponerse a '1'. La resta es igual a la suma del minuendo más el opuesto del sustraendo. En este caso, para cambiar el signo del sustraendo se deberá hacer el CA2 del mismo, mientras que el minuendo permanecerá sin cambios.

En este caso, a diferencia de la resta con sustraendo en CA1 del apartado anterior, el posible acarreo de salida se ignora.

El proceso para obtener un número en CA2 es, en primer lugar, obtener dicho número en CA1 y posteriormente sumarle '1'. Por tanto, deberemos mantener el bloque compuesto de

cuatro puertas XOR utilizado en el apartado 4.2 que permiten invertir el valor de los bits, es decir, obtener el CA1 del sustraendo y conectar la entrada 'C', que en el caso de la resta tiene valor '1', a la entrada de acarreo del circuito 'sumador binario paralelo de 4 bits'. Esto permite sumar una unidad al sustraendo de modo que se obtiene la entrada 'B' en CA2.

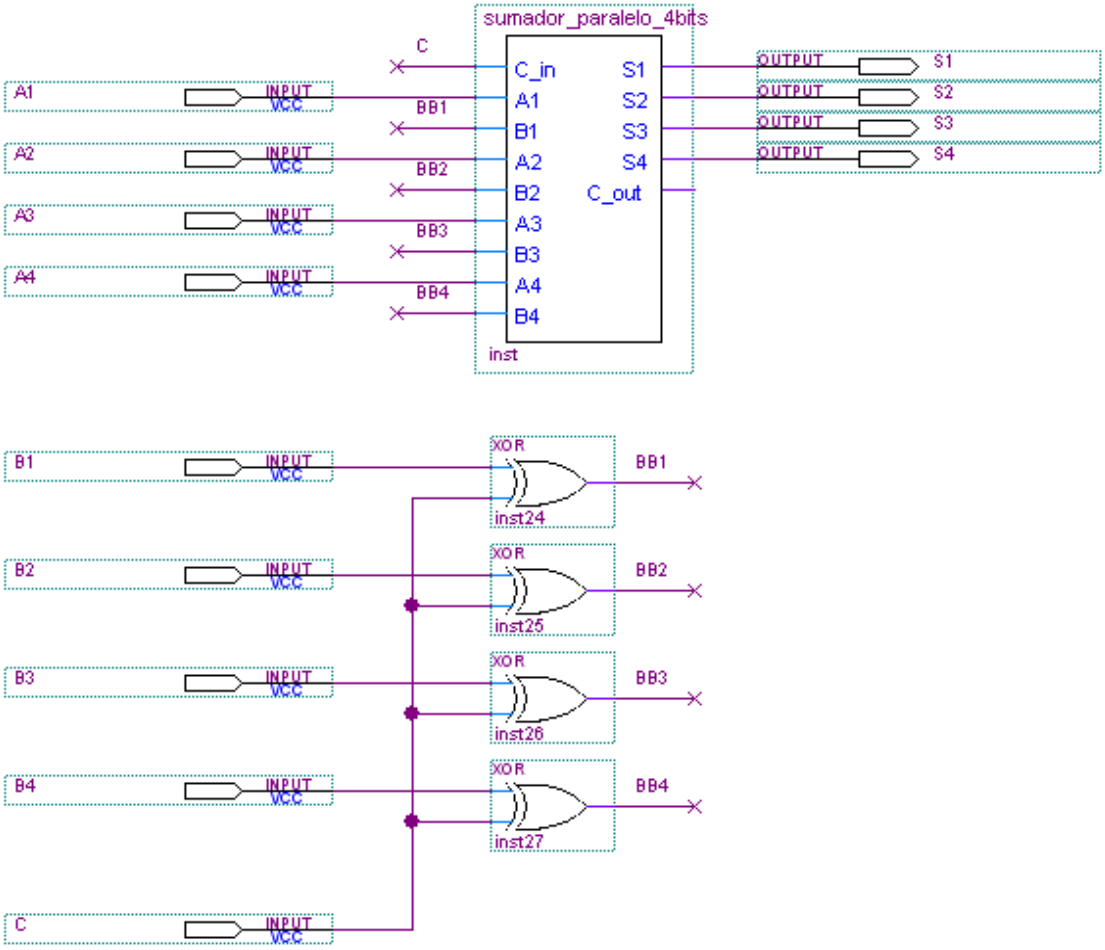


Figura 13. Circuito sumador / restador en CA2

Se comprobará el funcionamiento del circuito con las mismas operaciones del apartado anterior, de modo que se pueda ver que los resultados son los mismos.

a) 4 + 2 = 6 → 0100 + 0010 = 0110
b) 3 – 1 = 2 → 0011 + CA2(0001) = 0011 + 1111 = **1**0010 →0010
c) 1 – 5 = -4 → 0001 + CA2(0101) = 0001 + 1011 = 1100
-4 = CA2 (0100) = 1011 + 1 = 1100

A continuación se procede a realizar la simulación, introduciendo a mano los valores de las señales de los operandos

La simulación queda de la siguiente como se muestra en la Figura 14.

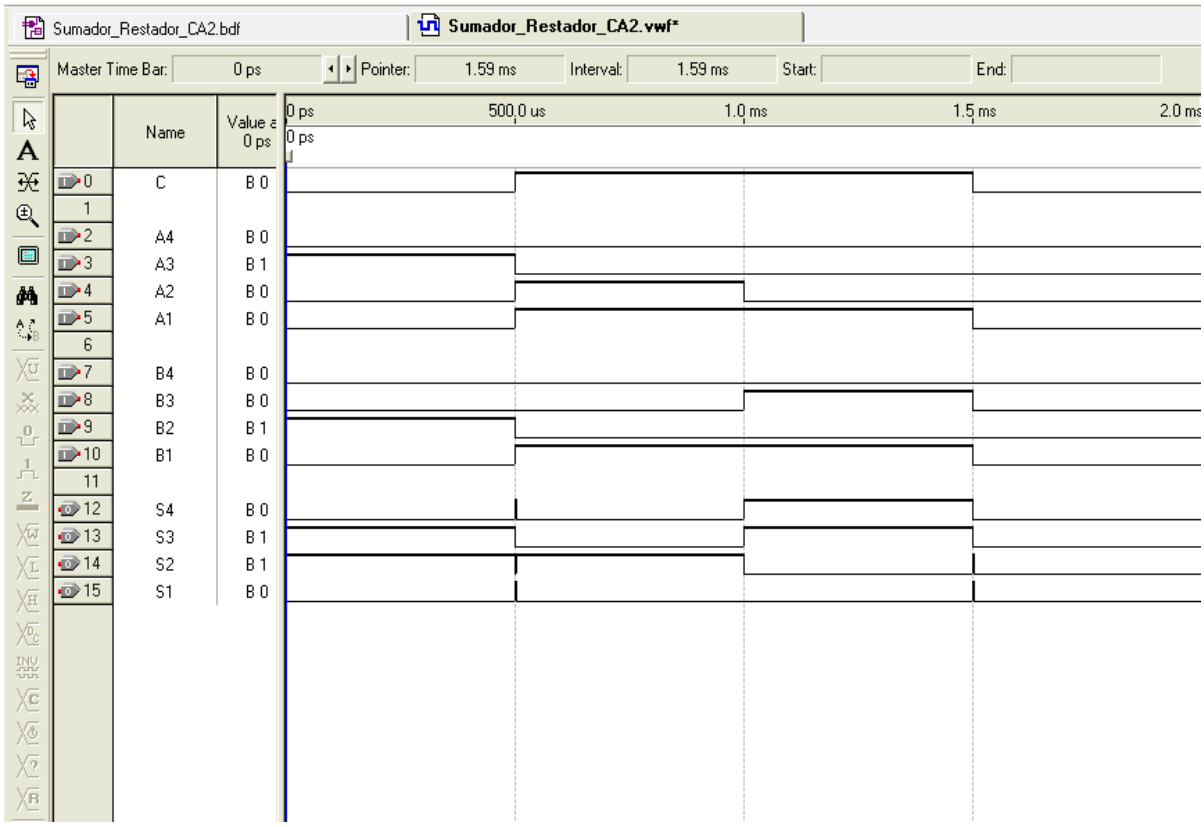


Figura 14. Resultados del circuito sumador / restador en CA2

Como se puede observar, el resultado de la simulación es el mismo que el obtenido en el apartado 4.2.

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

(1º Ingeniería Técnica en Informática de Gestión)

Práctica 3: Circuitos Lógicos Combinacionales

Un circuito lógico combinacional es aquel en el que sus salidas dependen exclusivamente del valor de sus entradas en un determinado momento, sin que intervengan en ningún caso estados anteriores de las entradas o salidas.

En electrónica digital la lógica combinacional está formada por ecuaciones simples a partir de las operaciones básicas del Álgebra de Boole. Entre los circuitos lógicos combinacionales clásicos tenemos:

- Multiplexor y Demultiplexor
- Codificador y Decodificador
- Comparadores

1.- OBJETIVOS

a) Dar a conocer tres elementos básicos de la electrónica digital en el campo de las comunicaciones: el Multiplexor, el Decodificador/Demultiplexor y el comparador.

Un multiplexor es un dispositivo lógico que recibe información por sus dos o más entradas y mediante una señal de control se decide cual de las entradas aparece reflejada en la salida, es decir, "un convertidor paralelo a serie". Si existe una señal de "enable", esta hace que el multiplexor esté habilitado o no.

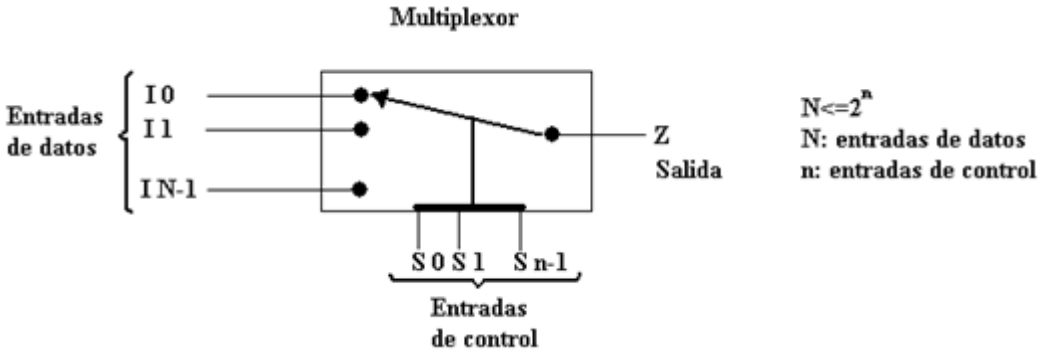


Figura 1. Multiplexor

Un Demultiplexor/Decodificador es un dispositivo en el cual se activará la salida que indique la codificación de la entrada.

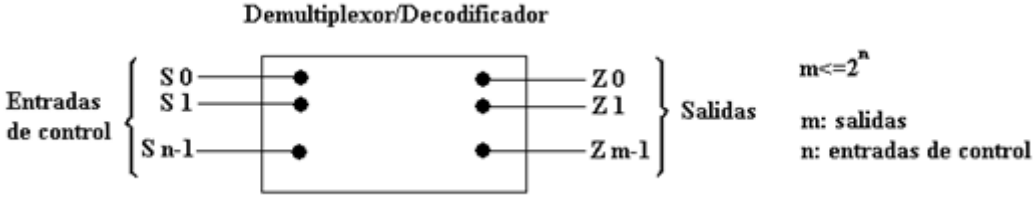


Figura 2. Demultiplexor/Decodificador

Un comparador es un dispositivo que compara dos entradas binarias (A y B de n bits) para indicar la relación de igualdad o desigualdad entre ellas.

Figura 3. Comparador

b) Utilizar multiplexores, decodificadores y comparadores para realizar funciones lógicas.

2.- MATERIAL

- Ordenador personal con el software QUARTUS II®.
- Manual introductorio al software.
- Guión de prácticas.

3.- DESARROLLO DE LA PRÁCTICA

En este apartado se pretende introducir y dar a conocer cómo se implementan los tres elementos básicos de la electrónica digital mencionados en el apartado de objetivos, los cuales normalmente nos los encontramos como bloques ya definidos donde no se aprecia su composición interna. Es importante entender cómo funcionan internamente, para que luego, cuando los mismos sean introducidos como bloque sepamos lo que deben hacer.

3.1.- MULTIPLEXOR DE CUATRO ENTRADAS DE DATOS

Introducir el diseño[1] mostrado en la Figura 4 en el Editor Gráfico, correspondiente a un multiplexor de cuatro entradas:

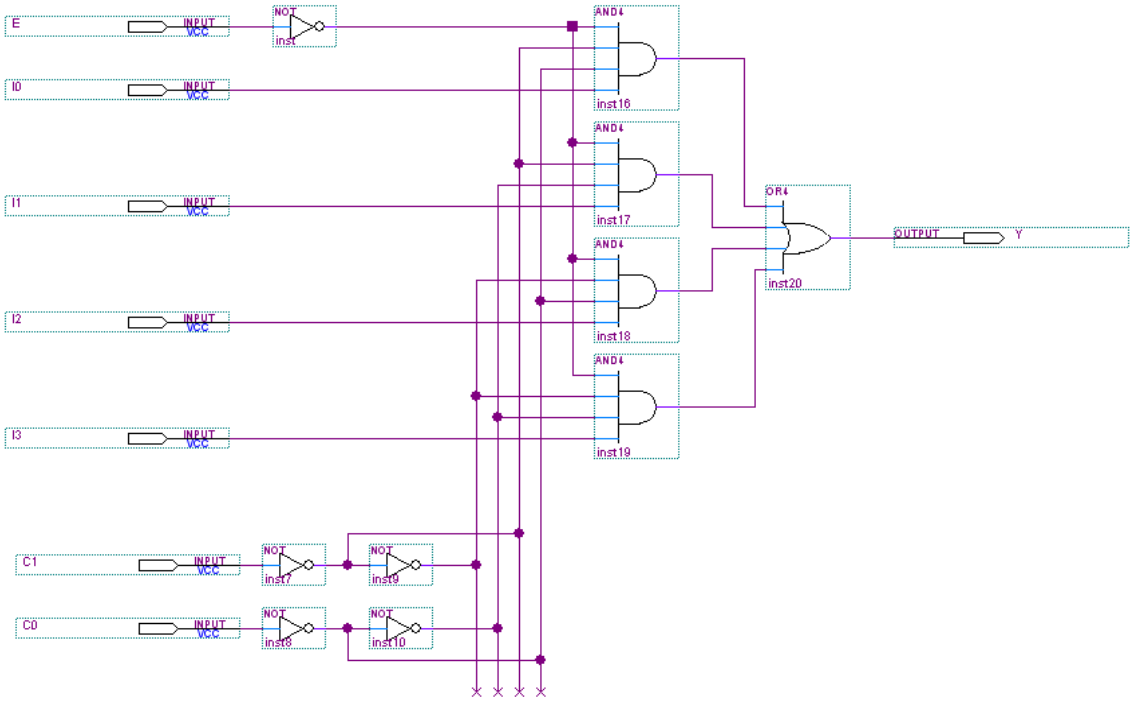


Figura 4. Circuito multiplexor de cuatro entradas de datos

Una vez introducido el circuito, se procederá a la compilación y simulación del diseño tal y como se hacía en la prácticas anteriores.

Realizar los siguientes ejercicios:

a) Obtener la tabla de verdad del multiplexor a partir de los resultados obtenidos en la simulación.

b) Representar el cronograma que representa la función del multiplexor.

c) Indicar la expresión de la salida en función de sus entradas, comprobando que corresponde a la función lógica de un multiplexor.

[1] *Nota: Este diseño corresponde a la mitad del circuito integrado 74153, el cual contiene dos multiplexores de cuatro entradas de datos.*

3.2.- DEMULTIPLEXOR/DECODIFICADOR DE CUATRO SALIDAS DE DATOS

Introducir el diseño mostrado en la Figura 5 en el Editor Gráfico, correspondiente a un demultiplexor/decodificador de cuatro salidas:

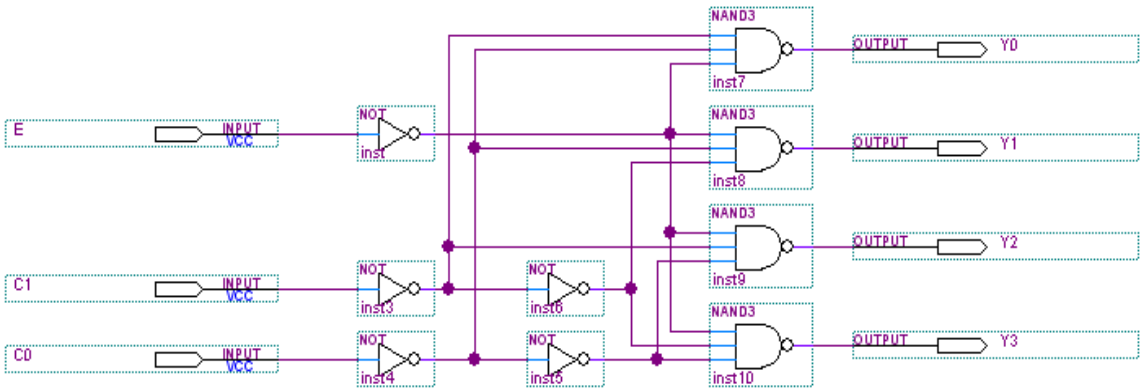


Figura 5. Circuito demultiplexor/decodificador de cuatro salidas

Al igual que en el ejercicio anterior, se procede a la compilación y simulación del diseño, de forma que se pueda ver si los resultados son correctos o no.

Realizar los siguientes ejercicios:

d) Obtener la tabla de verdad del demultiplexor/decodificador a partir de los resultados obtenidos en la simulación.

e) Representar el cronograma que representa la función del demultiplexor/decodificador.

f) Indicar las expresiones de las salidas en función de sus entradas, comprobando que corresponden a las funciones lógicas de un demultiplexor/decodificador.

3.3.- COMPARADOR BINARIO DE DOS BITS

Introducir el siguiente diseño en el Editor Gráfico, el cual nos indica si un bit es mayor, igual o menor que otro.

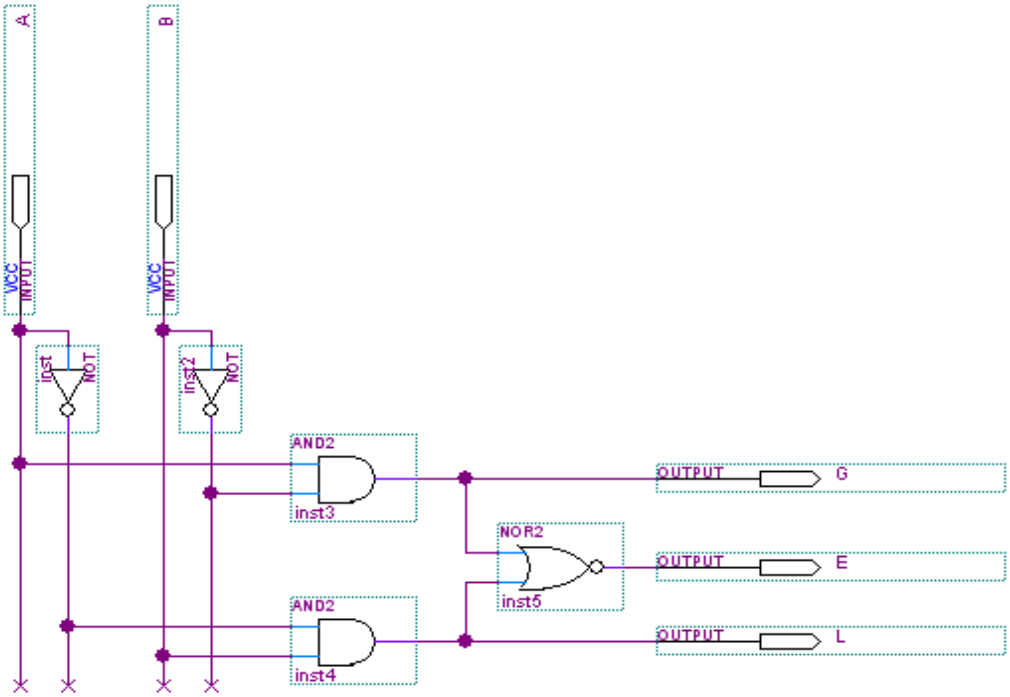


Figura 6. Circuito comparador binario de dos bits

El significado de las salidas que se pueden ver en la Figura 6 es el siguiente:

G → A > B
E → A = B
L → A < B

Proceder a la compilación y simulación del diseño para poder realizar los ejercicios que se presentan a continuación.

Realizar los siguientes ejercicios:

g) Obtener la tabla de verdad del comparador de dos bits a partir de los resultados obtenidos en la simulación.

h) Representar el cronograma que representa la función del comparador.

i) Indicar las expresiones de las salidas en función de las entradas, obteniéndolas de la forma más simplificada posible.

j) Obtener el correspondiente símbolo, tal y como se hacía en la práctica anterior, ya que será utilizado posteriormente.

4.- EJERCICIOS DE DISEÑO

4.1.- Generador de un bit de paridad

La salida de un sistema digital consiste en palabras de cuatro bits que han de transmitirse a otro sistema alejado físicamente. Para proteger la información enviada de errores introducidos en el camino se ha pensado en añadir un bit de paridad en cada palabra. (Criterio: el bit de paridad valdrá '1' cuando el número de unos en la palabra de información sea par; y valdrá '0' cuando el número de unos sea impar).

Diseñar un circuito combinacional capaz de generar dicho bit de paridad. Se dispone de un multiplexor 74151 y de puertas NAND2.

4.2.- Llave electrónica

Una empresa tiene ocho trabajadores, cada uno de los cuales dispone de una tarjeta digital que permite el acceso a las diferentes instalaciones. Los códigos de cada una de las tarjetas son los siguientes:

Trabajador	Código CBA
Iñaki	000
Emilio	001
Amaya	010
Aitor	011
Xabier	100
Ainara	101
Oihane	110
Mikel	111

Tabla 1

La función que permite el acceso al laboratorio de investigación es la siguiente:

$$F = \overline{A} \cdot B + \overline{B} \cdot \overline{C}$$

Si F = 1 acceso al laboratorio permitido
Si F = 0 acceso al laboratorio denegado

Decir qué trabajadores pueden acceder al laboratorio y diseñar un circuito combinacional empleando un demultiplexor/decodificador (por ejemplo el 74138) y una puerta NAND4, de manera que el resto de los trabajadores no pueda entrar.

4.3.- Detector de múltiplos de trece

Diseñar un circuito combinacional, utilizando un decodificador de tres entradas de código (74138) y un multiplexor de tres entradas de control (74151), de forma que el circuito reciba un número binario entre 0 y 63 y devuelva un '1' en la salida si se trata del número 13 o alguno de sus múltiplos.

En este ejercicio hay que tener especial cuidado con:

- Los pesos de las entradas de código del decodificador y de control del demultiplexor. En ambos casos la entrada 'C' es la de mayor peso.
- Realizando un doble clic sobre el símbolo del multiplexor, se accede a su correspondiente código de diseño, donde se puede apreciar como todas las entradas se encuentran conectadas a tierra (GND), lo que haría que las entradas que puedan quedar libres modificasen la función y, por tanto, la salida. Para que la solución no sea errónea, es necesario conectar todas las posibles entradas que puedan quedar libres a VCC.
- No conectar el enable a VCC, ya que éste se activa con un cero lógico.

4.4.- Comparador binario de palabras de 2 bits

Realizar, a partir del apartado 3.3, un comparador de palabras de dos bits, donde se obtenga si una de ellas es mayor, igual o menor que la otra.

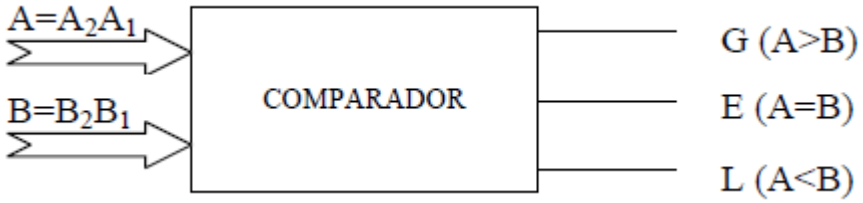


Figura 7. Modelo del comparador de palabras de dos bits

Para números de más de un bit, lo que se hace es ir comparando los bits de igual peso, desde el más significativo al menos significativo, hasta que se encuentre uno que sea mayor que el otro.

Las combinaciones posibles según las diferentes señales de entrada son $2^4 = 16$, pero para facilitar la situación sólo se estudiarán los siguientes casos:

A = 00	B = 00
A = 01	B = 11
A = 10	B = 00
A = 11	B = 11
A = 00	B = 01
A = 11	B = 10

Las señales a introducir en el Editor de Señales para realizar la simulación serán introducidas a mano.

RESOLUCIÓN PRÁCTICA 3

A continuación se presenta la resolución de los diferentes ejercicios que se presentan en este guión de prácticas.

3.1.- MULTIPLEXOR DE CUATRO ENTRADAS DE DATOS

El diseño a introducir es el siguiente:

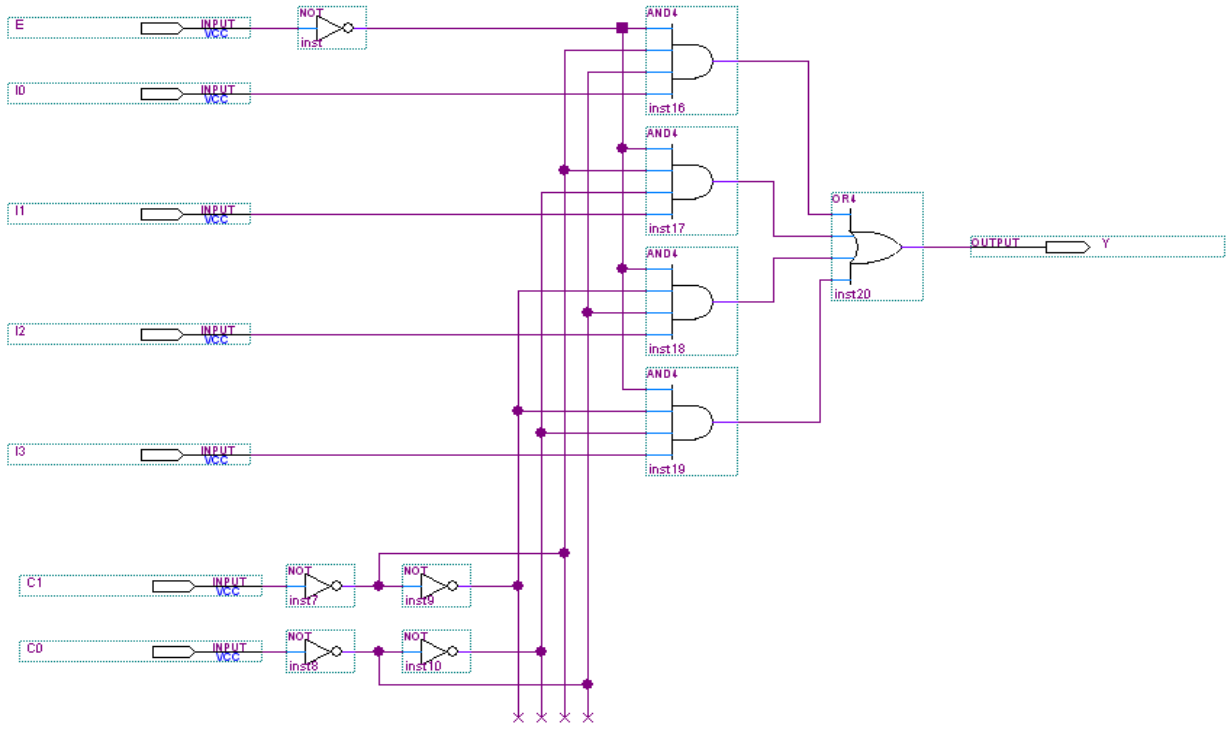


Figura 1. Circuito multiplexor de cuatro entradas de datos

El significado de cada una de las entradas y salidas es el siguiente:

- E: entrada de habilitación del circuito
- C0, C1: entradas de control
- I0, I1, I2, I3: entradas de datos
- Y: salida de datos.

Tras realizar la compilación del diseño, se procede a su simulación, obteniendo el resultado mostrado en la Figura 2.

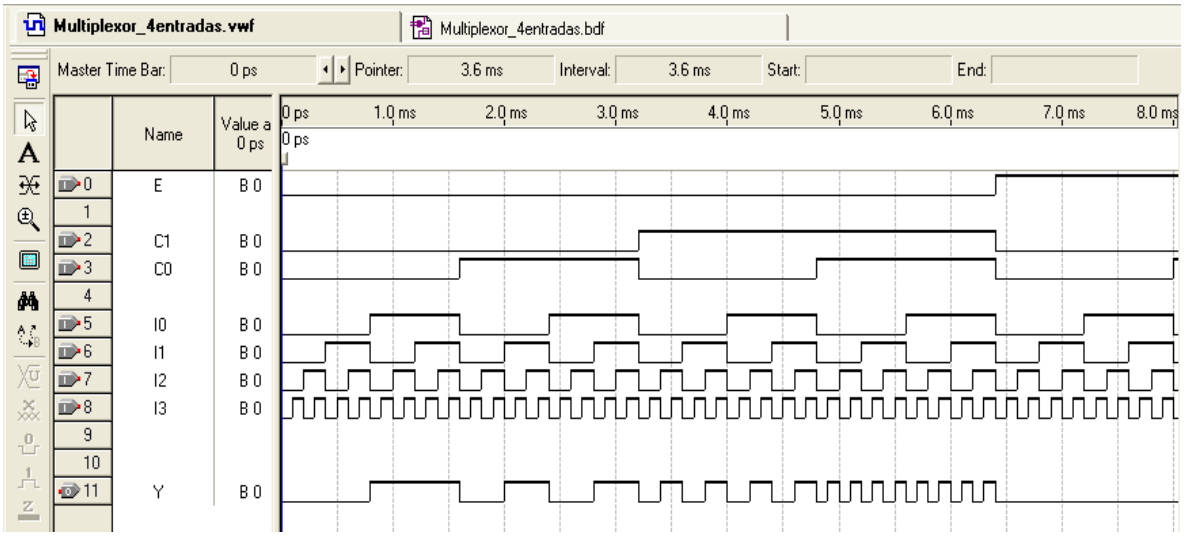


Figura 2. Resultados del multiplexor de cuatro entradas

Las frecuencias empleadas en la simulación son:

I3 → 5 KHz
I2 → 2,5 KHz
I1 → 1,25 KHz
I0 → 625 Hz
C0 → 312,5 Hz
C1 → 156,25 Hz
E → 78,125 Hz

En la simulación se puede ver como la señal de salida 'Y' se corresponde con la señal de entrada que está seleccionada mediante las entradas de control 'C1' y 'C0', mientras el circuito se encuentra habilitado teniendo un nivel bajo a la entrada 'E'.

La tabla de verdad de este multiplexor queda de la siguiente manera:

E	C1	C0	Y
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	X	X	0

Tabla 1

A partir de la tabla de la verdad se obtiene la expresión de la salida en función de las entradas:

$$Y = \overline{E} \cdot \overline{C1} \cdot \overline{C0} \cdot I0 + \overline{E} \cdot \overline{C1} \cdot C0 \cdot I1 + \overline{E} \cdot C1 \cdot \overline{C0} \cdot I2 + \overline{E} \cdot C1 \cdot C0 \cdot I3$$

Esta expresión se corresponde con la función lógica de un multiplexor genérico de dos entradas de control más una entrada de habilitación y cuatro salidas.

3.2.- DEMULTIPLEXOR/DECODIFICADOR DE CUATRO SALIDAS DE DATOS

El diseño introducido en el Editor Gráfico es el siguiente:

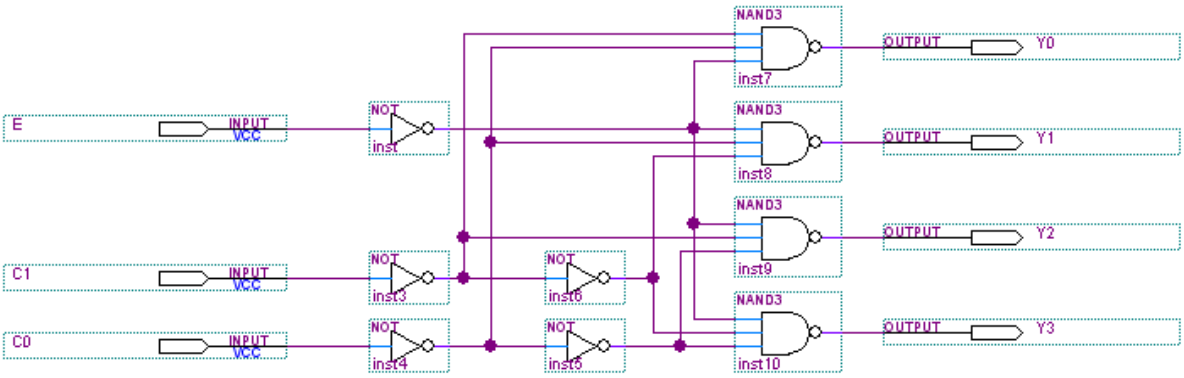


Figura 3. Circuito demultiplexor/decodificador de cuatro salidas

El significado de cada una de las entradas y salidas es el siguiente:

- E: entrada de habilitación del circuito.
- C0, C1: entradas de control/código.
- Y0, Y1, Y2, Y3: salidas de datos.

Realizando la simulación se obtiene el siguiente resultado:

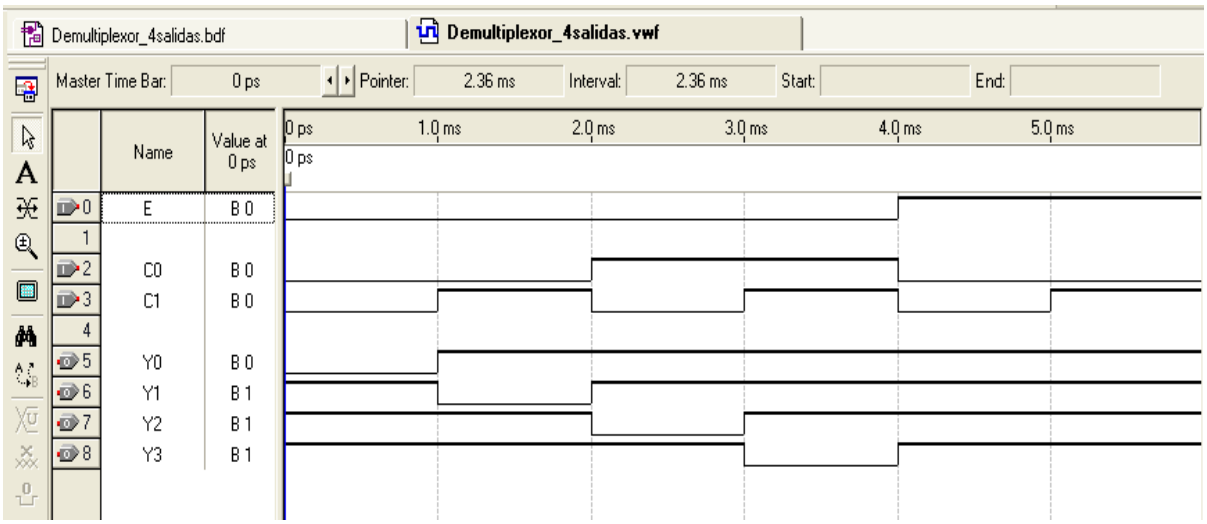


Figura 4. Resultado del demultiplexor de cuatro salidas

Como se puede observar en el resultado de la simulación, la salida queda activa a nivel bajo, según la combinación de las señales de las entradas de control.

Las frecuencias empleadas en la simulación son:

C1 → 500 Hz
C0 → 250 Hz
E → 125 Hz

La tabla de la verdad que se obtiene es la siguiente:

E	C1	C0	Y3	Y2	Y1	Y0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	X	X	1	1	1	1

Tabla 2

A partir de la tabla de la verdad se obtienen las expresiones de las salidas en función de las entradas, quedando de la siguiente manera:

$$Y0 = E + C1 + C0$$
$$Y1 = E + C1 + \overline{CO}$$
$$Y2 = E + \overline{C1} + CO$$
$$Y3 = E + \overline{C1} + \overline{C0}$$

Dichas expresiones se corresponden con las cuatro salidas de un decodificador de dos entradas de control/código (C0 y C1) más un entrada de habilitación del circuito (E).

3.3.- COMPARADOR BINARIO DE DOS BITS

El diseño introducido en este apartado es el siguiente:

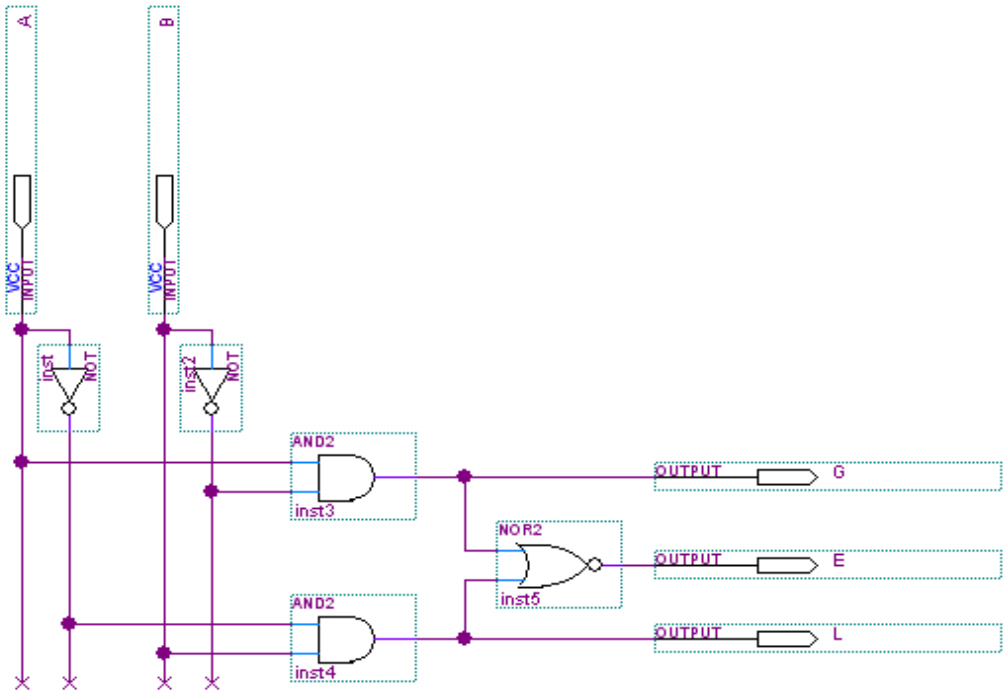


Figura 5. Circuito comparador binario de dos bits

El significado de las entradas y salidas es el siguiente:

- A, B: bits de entrada que van a ser comparados
- G: salida que indica A>B
- E : salida que indica A=B
- L : salida que indica A<B

El resultado de la simulación es el mostrado en la Figura 6:

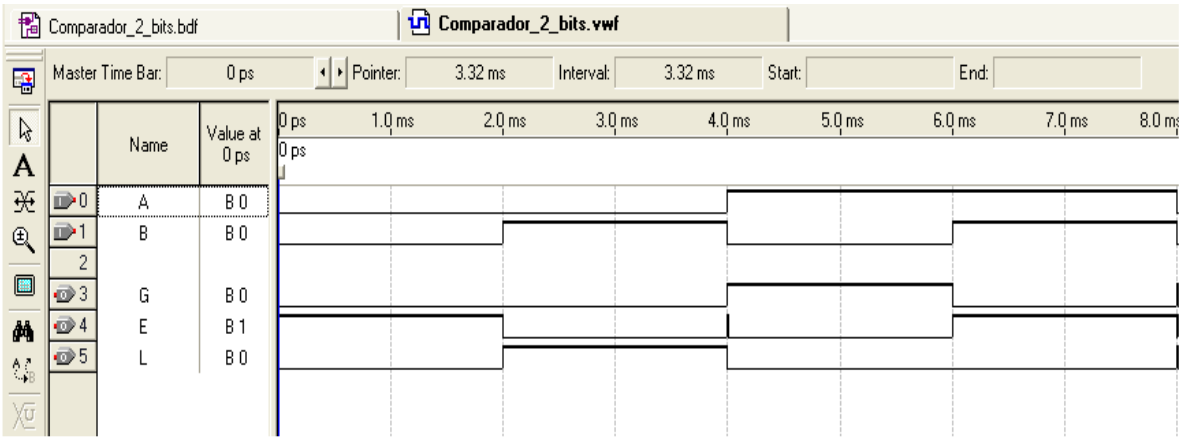


Figura 6. Resultados del comparador binario de dos bits

Las frecuencias empleadas son:

B → 250 Hz
A → 125 Hz

A partir del resultado de la simulación, se obtiene la siguiente tabla de la verdad:

A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Tabla 3

Las expresiones resultantes de las salidas y, que se corresponden con el comparador binario de 2 bits, son las siguientes:

$$G = A \cdot \overline{B}$$

$$L = \overline{A} \cdot B$$

$$E = \overline{A} \cdot \overline{B} + A \cdot B = \overline{A \oplus B} = \overline{\overline{A} \cdot B + A \cdot \overline{B}} = \overline{L + G}$$

Dichas expresiones, efectivamente permiten indicar la relación de igualdad o desigualdad de los bits de la entrada.

4.- EJERCICIOS DE DISEÑO

4.1.- Generador de un bit de paridad

Para comenzar con el diseño, se escribe la tabla de verdad según las especificaciones del ejercicio, es decir, una tabla con cuatro entradas y una salida a partir de la cual se obtendrá la función que representa el comportamiento del sistema.

El criterio exigido es que el bit de paridad sea '1' cuando el número de unos de la palabra sea par y '0' cuando éste sea impar.

Existe el caso particular en el cual la palabra está compuesta de todo ceros '0000', donde se toma que la palabra es par y, por tanto, su salida valdrá '1'.

La tabla de la verdad queda de la siguiente manera:

D	C	B	A	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Tabla 4

La función resultante es la siguiente:

$$Y = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}\,B\,A + \overline{D}\,C\,\overline{B}\,A + \overline{D}\,C\,B\,\overline{A} + D\,\overline{C}\,\overline{B}\,A + D\,\overline{C}\,B\,\overline{A} + D\,C\,\overline{B}\,\overline{A} + D\,C\,B\,A$$

Se dispone de un multiplexor 74151. Dicho multiplexor se compone de tres entradas de control, lo que supone ocho salidas de datos.

La función que representa el comportamiento del multiplexor 74151 es:

$$Y = \overline{D}\,0\,\overline{C}\,\overline{B}\,\overline{A} + D\,1\,\overline{C}\,\overline{B}\,A + D\,2\,\overline{C}\,B\,\overline{A} + \overline{D}\,3\,\overline{C}\,B\,A + D\,4\,C\,\overline{B}\,\overline{A} + \overline{D}\,5\,C\,\overline{B}\,A + \overline{D}\,6\,C\,B\,\overline{A} + D\,7\,C\,B\,A$$

Considerando las variables A, B y C como las entradas de control del multiplexor y teniendo D y $\overline{D}$ como las correspondientes entradas de datos se obtiene que:

D0 = D3 = D5 = D6 = $\overline{D}$

D1 = D2 = D4 = D7 = D

En circuito resultante en base al multiplexor 74151 y una puerta NAND2 es el mostrado en la Figura 7.

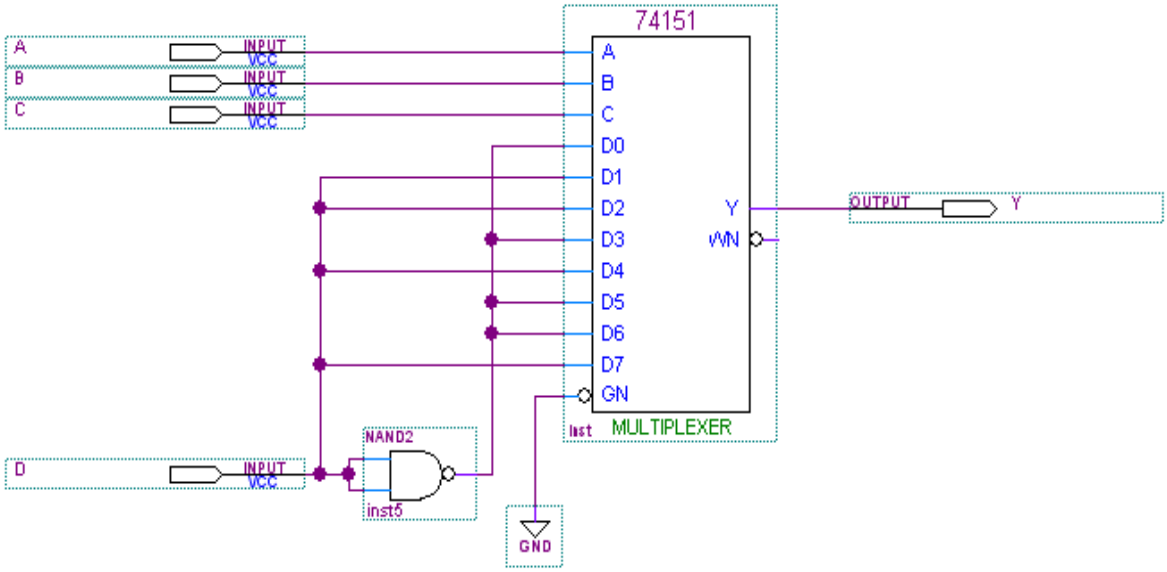


Figura 7. Circuito generador del bit de paridad

Realizando la simulación del diseño se puede comprobar como la señal resultante en la salda 'Y' se corresponde con sus respectivos valores en la tabla de la verdad previamente realizada, luego se considera que el diseño es correcto.

La Figura 8 muestra el resultado de la simulación

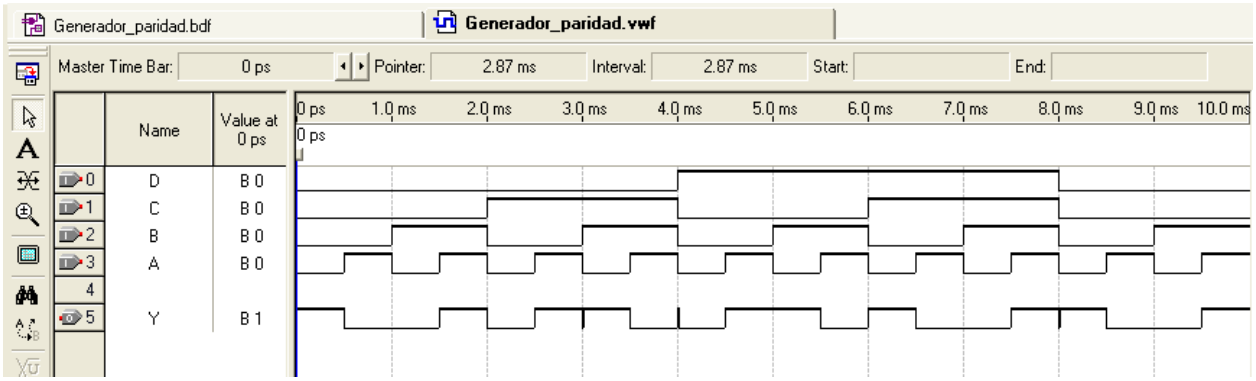


Figura 8. Resultados del generador del bit de paridad

Las frecuencias empleadas en la simulación son:

A → 1 Khz

B → 500 Hz

C → 250 Hz

4.2.- Llave electrónica

Tenemos la siguiente función:

$$F = \overline{A} \cdot B + \overline{B} \cdot \overline{C}$$

Si F = 1 acceso al laboratorio permitido
Si F = 0 acceso al laboratorio denegado

Como esta función consta de tres variables, es necesario emplear un demultiplexor de tres entradas de código, como, por ejemplo, el 74138.

Para comenzar con el diseño, es necesario calcular la expresión anterior en su forma canónica, quedando de la siguiente manera:

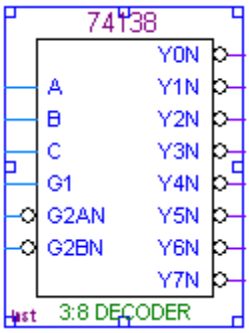
$$F = \overline{A}\,B(C+\overline{C}) + (A+\overline{A})\overline{B}\overline{C} = CB\overline{A} + \overline{C}B\overline{A} + \overline{C}\overline{B}A + \overline{C}\overline{B}\overline{A}$$

De esta función se obtiene la lista de trabajadores que pueden acceder al laboratorio:

		Código CBA	**Trabajador**
$CB\overline{A}$	→	110	Oihane
$\overline{C}B\overline{A}$	→	010	Amaya
$\overline{C}\overline{B}A$	→	001	Iñaki
$\overline{C}\overline{B}\overline{A}$	→	000	Emilio

Las expresiones de las respectivas salidas del decodificador 74138 en función de las entradas son:

$$Y0N = C + B + A \Rightarrow YO = \overline{C} \cdot \overline{B} \cdot \overline{A}$$
$$Y1N = C + B + \overline{A} \Rightarrow YO = \overline{C} \cdot \overline{B} \cdot A$$
$$Y2N = C + \overline{B} + A \Rightarrow YO = \overline{C} \cdot B \cdot \overline{A}$$
$$Y3N = C + \overline{B} + \overline{A} \Rightarrow YO = \overline{C} \cdot B \cdot A$$
$$Y4N = \overline{C} + B + A \Rightarrow YO = C \cdot \overline{B} \cdot \overline{A}$$
$$Y5N = \overline{C} + B + \overline{A} \Rightarrow YO = C \cdot \overline{B} \cdot A$$
$$Y6N = \overline{C} + \overline{B} + A \Rightarrow YO = C \cdot B \cdot \overline{A}$$
$$Y7N = \overline{C} + \overline{B} + \overline{A} \Rightarrow YO = C \cdot B \cdot A$$



Teniendo en cuenta las combinaciones que permiten el acceso al laboratorio, la función del diseño queda de la siguiente manera:

$$F = Y0 + Y1 + Y2 + Y6 = \overline{Y0N \cdot Y1N \cdot Y2N \cdot Y6N}$$

El diseño a partir del multiplexor 74138 y una puerta NAND4 es el mostrado en la Figura 9.

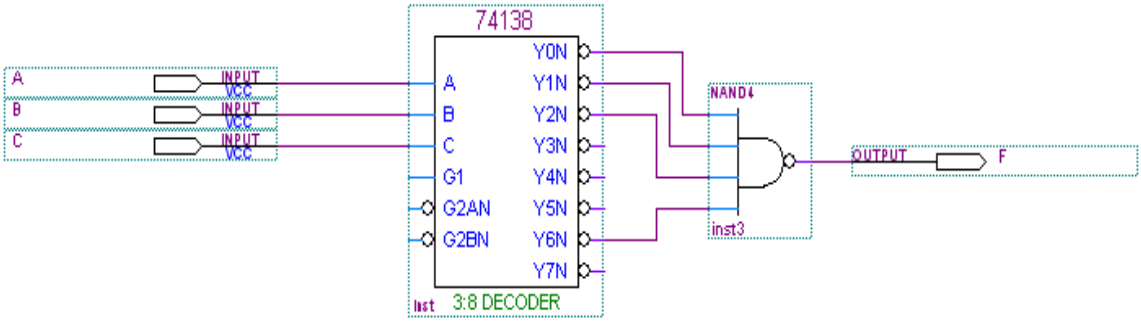


Figura 9. Circuito de la Llave electrónica

Al realizar la simulación, se visualiza como la señal 'F' toma valor '1' con las combinaciones de entrada que permiten el acceso al laboratorio, coincidiendo con la lista elaborada anteriormente. La simulación se visualiza en la Figura 10.

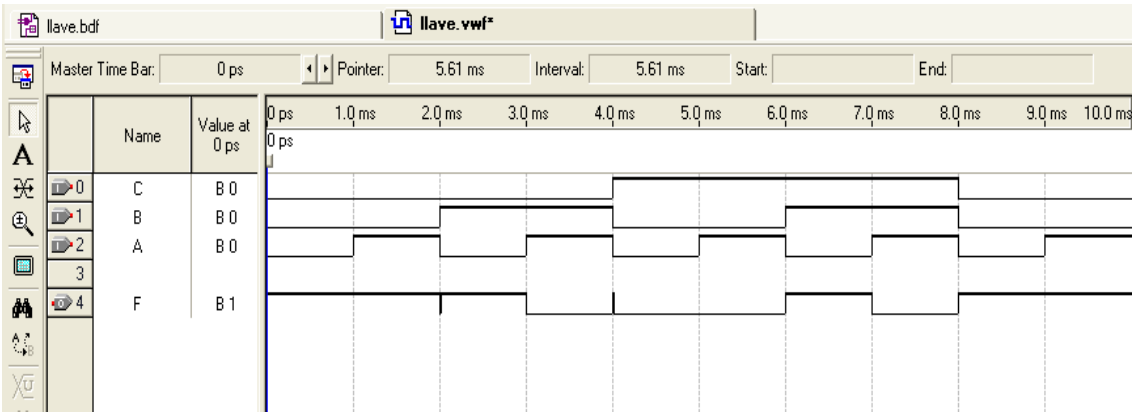


Figura 10. Resultados del diseño de la Llave electrónica

Las frecuencias empleadas en la simulación son:

A → 500 Hz
B → 250 Hz
C → 125 Hz

4.3.- Detector de múltiplos de trece

Los números múltiplos de 13 en el rango 0-63 son: 13, 26, 39 y 52.

Tenemos que $2^6 = 64$, luego existen 6 señales de entrada (I0, I1, I2, I3, I4, I5) con las que se representará el número existente a la entrada del circuito.

Tomando como bit más significativo la señal I0 y como bit menos significativo al bit I5, obtenemos que la codificación de los anteriores números queda de la siguiente manera:

Número	Codificación					
	I0	I1	I2	I3	I4	I5
13	0	0	1	1	0	1
26	0	1	1	0	1	0
39	1	0	0	1	1	1
52	1	1	0	1	0	0

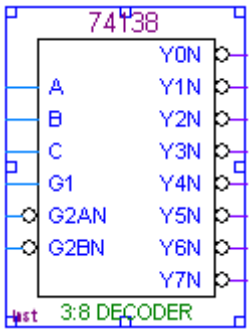
Tabla 5

Teniendo que F = 1 si el número es múltiplo de trece y F = 0 en caso contrario, se obtiene la siguiente función:

$$F = \overline{IO}\cdot\overline{I1}\cdot I2\cdot I3\cdot\overline{I4}\cdot I5 + \overline{IO}\cdot I1\cdot I2\cdot\overline{I3}\cdot I4\cdot\overline{I5} + IO\cdot\overline{I1}\cdot\overline{I2}\cdot I3\cdot I4\cdot I5 + IO\cdot I1\cdot\overline{I2}\cdot I3\cdot\overline{I4}\cdot\overline{I5}$$

El decodificador 74138 y el multiplexor 74151 disponen de tres entradas de código y control respectivamente, por lo que se toma como entradas del decodificador las señales I0, I1 e I2, y como entradas del multiplexor a las señales I3, I4, e I5.

Las ecuaciones de salida del decodificador son:



$$Y0N = C + B + A \Rightarrow YO = \overline{C}\cdot\overline{B}\cdot\overline{A}$$

$$Y1N = C + B + \overline{A} \Rightarrow YO = \overline{C}\cdot\overline{B}\cdot A$$

$$Y2N = C + \overline{B} + A \Rightarrow YO = \overline{C}\cdot B\cdot\overline{A}$$

$$Y3N = C + \overline{B} + \overline{A} \Rightarrow YO = \overline{C}\cdot B\cdot A$$

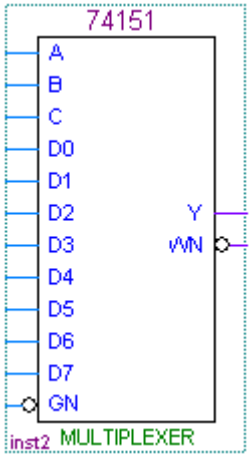
$$Y4N = \overline{C} + B + A \Rightarrow YO = C\cdot\overline{B}\cdot\overline{A}$$

$$Y5N = \overline{C} + B + \overline{A} \Rightarrow YO = C\cdot\overline{B}\cdot A$$

$$Y6N = \overline{C} + \overline{B} + A \Rightarrow YO = C\cdot B\cdot\overline{A}$$

$$Y7N = \overline{C} + \overline{B} + \overline{A} \Rightarrow YO = C\cdot B\cdot A$$

Para el caso del multiplexor se tiene lo siguiente:



$$Y = D0\cdot\overline{C}\,\overline{B}\,\overline{A} + D1\cdot\overline{C}\,\overline{B}A + D2\cdot\overline{C}B\overline{A} + D3\cdot\overline{C}BA + D4\cdot C\,\overline{B}\,\overline{A} + D5\cdot C\,\overline{B}A + D6\cdot CB\overline{A} + D7\cdot CBA$$

$$WN = \overline{D0}\cdot\overline{C}\,\overline{B}\,\overline{A} + \overline{D1}\cdot\overline{C}\,\overline{B}A + \overline{D2}\cdot\overline{C}B\overline{A} + \overline{D3}\cdot\overline{C}BA + \overline{D4}\cdot C\,\overline{B}\,\overline{A} + \overline{D5}\cdot C\,\overline{B}A + \overline{D6}\cdot CB\overline{A} + \overline{D7}\cdot CBA$$

Se asigna en el decodificador: C = I0; B = I1; A = I2;
Se asigna en el multiplexor: C = I3; B = I4; A = I5;

Tomando de nuevo la función 'F', la siguiente tabla muestra las conexiones entre decodificador y multiplexor:

$$F = \overline{IO}\cdot\overline{I1}\cdot I2\cdot I3\cdot\overline{I4}\cdot I5 + \overline{IO}\cdot I1\cdot I2\cdot\overline{I3}\cdot I4\cdot\overline{I5} + IO\cdot\overline{I1}\cdot\overline{I2}\cdot I3\cdot I4\cdot I5 + IO\cdot I1\cdot\overline{I2}\cdot I3\cdot\overline{I4}\cdot\overline{I5}$$

Número	Decodificador			Multiplexor		
13	$\overline{IO}\cdot\overline{I1}\cdot I2$	001	Y1N	$I3\cdot\overline{I4}\cdot I5$	101	D5
26	$\overline{IO}\cdot I1\cdot I2$	011	Y3N	$\overline{I3}\cdot I4\cdot\overline{I5}$	010	D2
39	$IO\cdot\overline{I1}\cdot\overline{I2}$	100	Y4N	$I3\cdot I4\cdot I5$	111	D7
52	$IO\cdot I1\cdot\overline{I2}$	110	Y6N	$I3\cdot\overline{I4}\cdot\overline{I5}$	100	D4

Tabla 6

El diseño del circuito se muestra en la Figura 11.

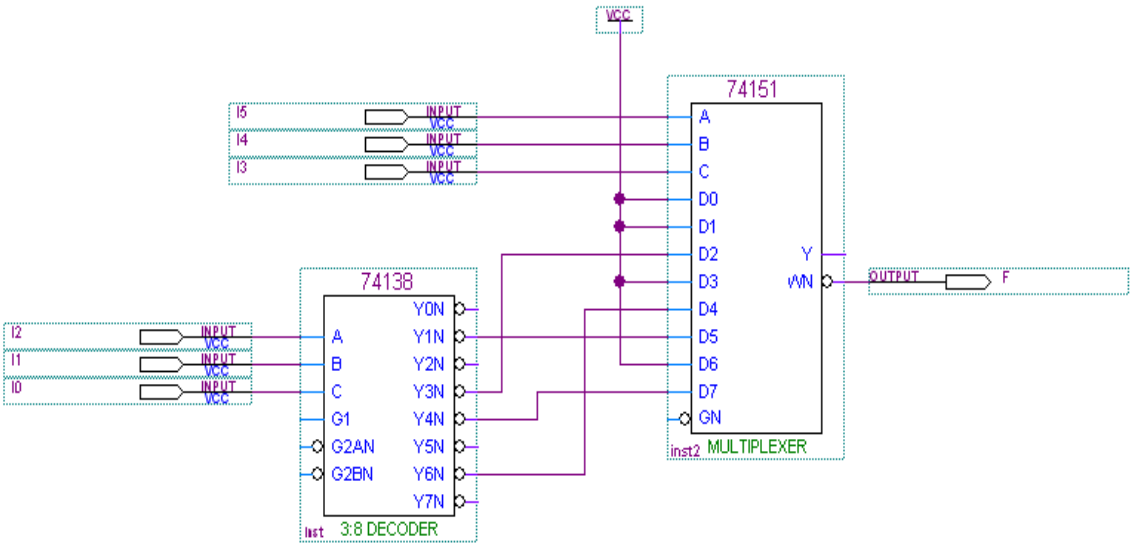


Figura 11. Circuito detector de múltiplos de trece

Tener en cuenta que hay que conectar las entradas libres del multiplexor a Vcc (excepto 'GN' que es activa a nivel bajo) con el fin de evitar resultados no deseados en la simulación.

Asimismo, hay que prestar atención a que las salidas del decodificador están invertidas, luego en el multiplexor se vuelve a invertir la señal, obteniendo el resultado a través de su salida negada.

La Figura 12 muestra el resultado de la simulación, donde se puede observar como la señal 'F' únicamente toma valor '1' para los múltiplos de trece.

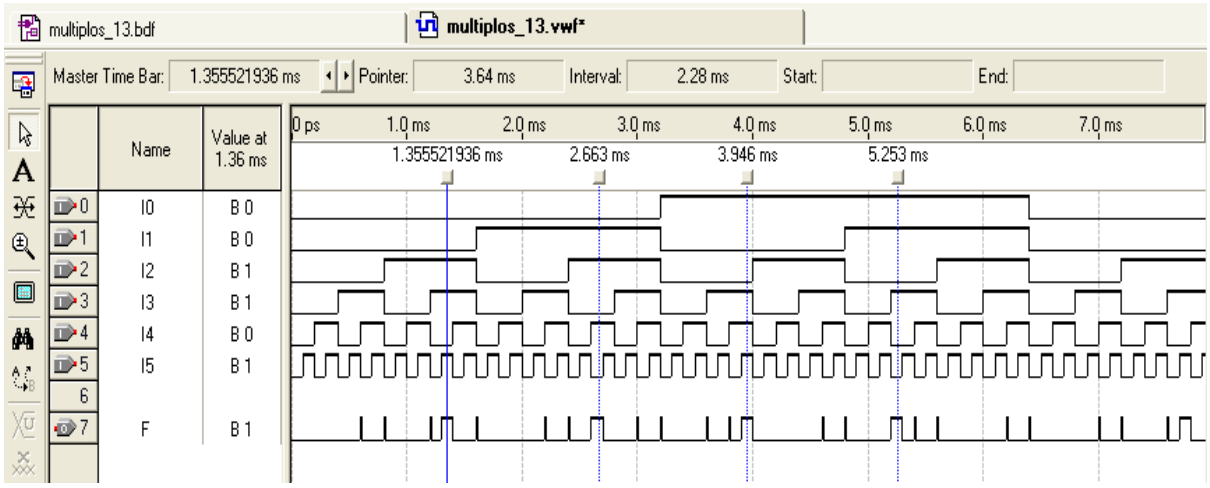


Figura 12. Resultados del circuito detector de múltiplos de trece

Las frecuencias de las señales de entrada son:

I0 → 312,5 Hz
I1 → 625 Hz
I2 → 1,25 KHz
I3 → 2,5 KHz
I4 → 5 KHz
I5 → 10 KHz

4.4.- Comparador binario de palabras de 2 bits

Para la realización del diseño hay que tener en cuenta los siguientes casos:

1º- Comparación del bit más significativo. En caso de desigualdad entre ambos ya estaremos en condiciones de saber los estados A>B y A<B.

2º.- En caso de igualdad en el bit de mayor peso, será el bit de menor peso el que nos de el resultado.

Esto se resume en las siguientes ecuaciones:

$$G(A>B) = 1 \text{ si } \{(A2 > B2) + ((A2 = B2) * (A1 > B1))\}$$
$$E(A=B) = 1 \text{ si } \{(A2 = B2) * (A1 = B1)\}$$
$$L(A<B) = 1 \text{ si } \{(A2 < B2) + ((A2 = B2) * (A1 < B1))\}$$

En base a las anteriores ecuaciones, se tiene que las expresiones de salida, utilizando el comparador del apartado 3.3, son las siguientes:

G = G2 + E2*G1
E = E2 * E1
L = L2 + E2*L1

El circuito resultante se muestra a continuación:

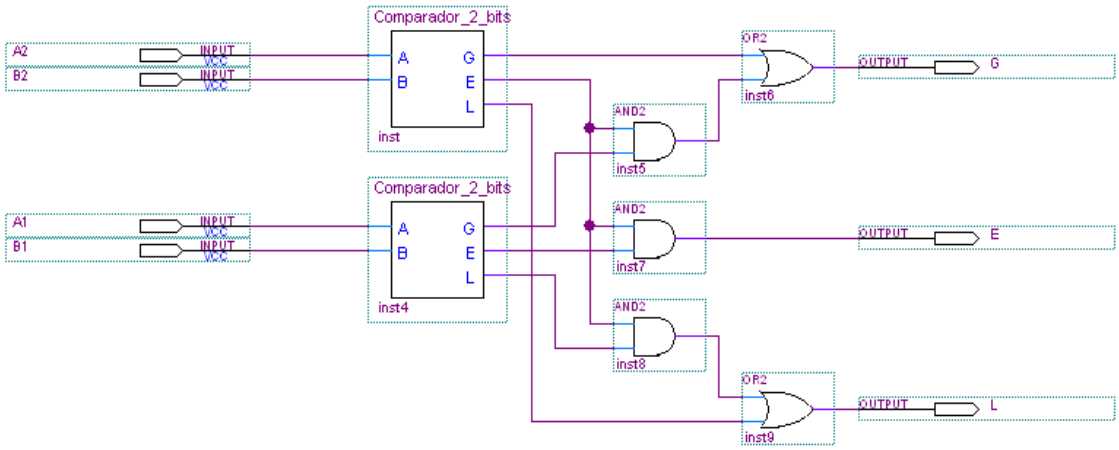


Figura 13. Circuito comparador binario de palabras de 2 bits

Para comprobar el correcto funcionamiento del diseño, se estudian los casos mostrados en la siguiente tabla, incluyendo sus respectivas salidas.

A2	A1	B2	B1	G	E	L
0	0	0	0	0	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	1	1	1	0	1	0
0	0	0	1	0	0	1
1	1	1	0	1	0	0

Tabla 7

A continuación se procede a realizar la simulación, para comprobar que los resultados coinciden con los expuestos en la *Tabla 7.*

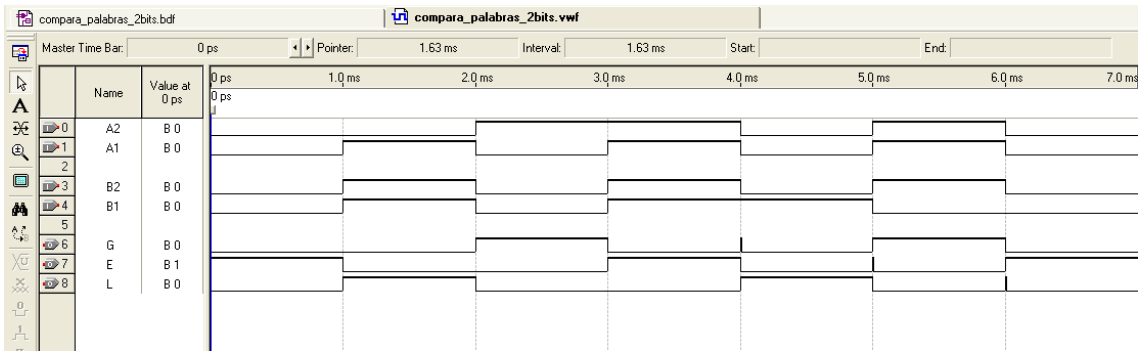


Figura 14. Resultados del comparador binario de 2 bits

Las señales de entrada para la simulación deben ser introducidas a mano.

Como se puede observar las señales de salida G, E y L coinciden con lo obtenido en la *Tabla 7.*

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

(1º Ingeniería Técnica en Informática de Gestión)

Práctica 4: Introducción a VHDL.

VHDL, viene de VHSIC (Very High Speed Integrated Circuit Hardware Description Language). VHDL es un lenguaje de descripción y modelado diseñado para describir (en una forma que los humanos y las máquinas puedan leer y entender) la funcionalidad y la organización de sistemas hardware digitales, placas de circuitos, y componentes.

1.- OBJETIVOS

Introducirse en el manejo y comprensión del lenguaje VHDL. Para ello se presentan una serie de ejercicios semi-guiados que deberán ser entendidos y realizados.

2.- MATERIAL

- Ordenador personal con el software QUARTUS II®.
- Manual introductorio al software.
- Guión de prácticas.

3.- INTRODUCCIÓN A VHDL

Antes de empezar, se va a hacer una breve introducción al lenguaje de programación VHDL. De esta manera, se simplificará la comprensión de los ejemplos que se presentarán posteriormente.

Los dos bloques que forman cualquier diseño basado en VHDL, son la entidad y la arquitectura.

La **entidad o entity** es la interfaz del dispositivo con el exterior. Tiene por objeto decir qué señales son visibles o accesibles desde el exterior, es decir, los puertos o ports del dispositivo.

La estructura más habitual de la entidad es la siguiente:

```
Entity nombre_entidad is port(
        puertos
);
  end entidad;
```

La **arquitectura o architecture** define la función del dispositivo, es decir, qué transformaciones se realizarán sobre los datos que entren por los puertos de entrada para producir la salida. Dentro de este apartado es donde se dota de operatividad al circuito.

La estructura más habitual para la arquitectura es:

```
architecture nombre_arquitectura of nombre_entidad is
    declaraciones
begin
    sentencias
end nombre;
```

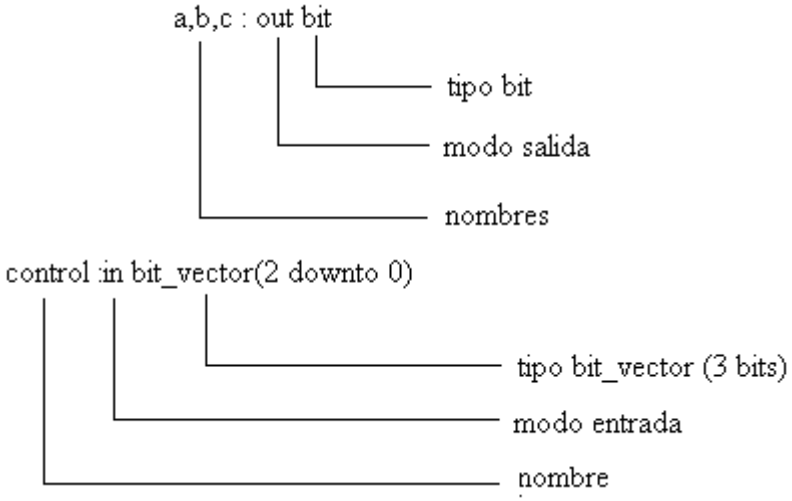
Cada señal I/O queda referida como un puerto (port). El conjunto de puertos definido para una entidad se denomina declaración del puerto.

Cada puerto declarado debe tener un nombre, una dirección o modo y un tipo de datos.

- **Modos**: existen cuatro tipos distintos para el campo nodo, de los cuales se van a utilizar dos:
  - In: los datos son sólo de entrada.
  - Out: los datos son sólo de salida. Las variables definidas de esta manera no se podrán utilizar en la arquitectura, pues se consideran inaccesibles a la entidad.

- **Tipos**: sirve para indicar el tipo de dato de cada puerto. Los más utilizados son: Boolean, bit, bit_vector e integer.

Ejemplo:



La sentencia **process** es una de las construcciones usadas para agrupar algoritmos. Se inicia (de forma opcional) con una etiqueta seguida de dos puntos, después la palabra reservada **process** y una lista de variables sensibles. Esta lista indica qué señales harán que se ejecute el proceso, es decir, que variable(s) debe(n) cambiar para que se ejecute el mismo. Dentro de un proceso se encuentran sentencias secuenciales. Esto hace que el orden dentro de un proceso sea importante, ya que se ejecuta una orden después de otra, y los posibles cambios que deba haber en las señales alteradas se producen después de evaluar todo el ciclo completo. Esta característica define una de las particularidades de VHDL.

La estructura de un proceso es la siguiente:

```
etiqueta: process (var1, var2, …)
  begin
  sentencias secuenciales
end process etiqueta;
```

Una forma de asignar valores a las distintas señales es mediante el símbolo “<=”. Esto se puede ver en los ejemplos que vienen a continuación.

4.- DESARROLLO DE LA PRÁCTICA

Todos los ejercicios que se muestran en este guión de prácticas han de realizarse mediante el Editor de Texto que dispone Quartus II. Por tanto, para abrir el Editor de Texto, basta con pinchar sobre el icono que representa un folio en blanco y seleccionar la opción 'VHDL File' ya que trabajamos con lenguaje VHDL. El archivo creado tendrá extensión '.vhd'. Anteriormente, tendremos que haber definido el proyecto tal y como se hacía en las anteriores prácticas.

4.1.- PUERTA AND CON DOS ENTRADAS

A continuación se presenta un sencillo ejercicio en VHDL que muestra lo visto hasta ahora. Dicho ejercicio emula el comportamiento de una puerta 'AND' con dos entradas.

```
entity P_and2 is port(
 a,b : in bit;                          --2 canales de entrada
 c : out bit                            --1 canal de salida
 );
 end entity P_and2;

 architecture arch_and of P_and2 is
  begin                                 --Cabecera del programa
                c<=a and b;             --Comienza el programa
  end architecture arch_and;            --Finaliza el programa
```

Como se puede observar, el programa se divide en las dos estructuras, entidad y arquitectura, mencionadas anteriormente. En la entidad se ve que las entradas de datos son 'a' y 'b' y la salida es 'c'. Todas tienen un bit de ancho.

- *Ejercicio:*

Analizar y comprender el código, introducirlo en el Editor de Texto y simular. Comprobar que el resultado es correcto.

***Nota**: a la hora de grabar, dos consideraciones a tener en cuenta:*
1º- El diseño tiene que ser guardado tal y como se ha nombrado a la entidad.
2º- No dar nombres de funciones que se encuentren en las librerías ya que tendremos problemas a la hora de compilar.

4.2.- PUERTAS AND, OR Y NOT

Introducir en el Editor de Texto el siguiente código, el cual representa el comportamiento de las puertas AND, OR y NOT.

```
entity and_or_not is port(
 a,b : in bit;                          --2 canales de entrada
 c_or, c_and, c_not : out bit           --3 canales de salida
);
end and_or_not;

architecture arch_aon of and_or_not is  --Cabecera del programa
 begin                                  --Comienza el programa
  a_o_x:process(a,b)                    --Cabecera de un proceso
   begin                                --Empieza el proceso
    c_or<=a or b;
    c_and<=a and b;
    c_not<=not a;
   end process a_o_x;                   --Acaba el proceso
  end arch_aon;                         --Finaliza el programa
```

- *Ejercicios:*

a) Introducir el código en el Editor de Texto, simular y comprobar el correcto funcionamiento del diseño.

b) Identifica cada una de las partes del proceso.

c) Escribe un programa similar para las puertas NAND y NOR. Comprueba los resultados.

*Nota 1: Como se puede comprobar, se le ha asignado una etiqueta a la sentencia **process**. Sin embargo, tal y como se ha comentado anteriormente, esta etiqueta es opcional y podría obviarse, como se verá en los posteriores ejemplos.*

4.3.- MULTIPLEXOR EN VHDL

A continuación se presenta el código de un multiplexor sencillo hecho con VHDL:

```
entity multiplexor is port(
 a,b:in bit;                                  --2 canales de entrada
 control:in bit;                              --1 señal de control
 enable:in bit;                               --señal enable
 c:out bit                                    --1 canal de salida
 );
end multiplexor;

architecture archmul of multiplexor is        --Cabecera del programa
 begin                                        --Comienza el programa
  process(a,b,control,enable)                 --Cabecera de un proceso
   begin                                      --Empieza el proceso
    if enable='1' then c<='0';                --Sentencia if
     elsif enable='0' then                    --Sentencia elsif
      case control is                         --Sentencia case
       when '0' =>c<=a;
       when '1' =>c<=b;
       when others =>c<='0';
      end case;                               --Acaba sentencia case
     end if;                                  --Acaba sentencia if y elsif
    end process;                              --Acaba el proceso
   end archmul;                               --Finaliza el programa
```

- *Ejercicios:*

a) ¿Qué tamaño tienen cada una de las entradas y salidas definidas en la entidad?

b) Observando el código, elaborar su correspondiente tabla de la verdad.

c) Introducir el ejemplo anterior en el Editor de Texto. Con él se creará un nuevo símbolo y tras abrirlo en el Editor Gráfico, se conectarán las entradas y salidas necesarias, y se verá si coincide con la tabla de verdad elaborada en el apartado anterior.

Nota 2: En el caso en que la señal 'control' tenga un valor que no sea ninguno de los especificados anteriormente, la salida vale cero. Esto se suele poner siempre por si no se contemplan todos los casos posibles al evaluar la señal, aunque en este caso, al ser tan sencillo, realmente no sería necesario.

4.4.- DEMULTIPLEXOR/DECODIFICADOR EN VHDL

A continuación se presenta un demultiplexor/decodificador sencillo hecho con VHDL:

```
entity demulti is port(
 control:in bit;                          --1 señal de control
 enable:in bit;                           --señal de enable
 a,b:out bit                              --2 canales de salida
 );
end demulti;

architecture archdemulti of demulti is    --Cabecera del programa
 begin                                    --Comienza el programa
  process (control,enable)                --Cabecera de un proceso
   begin                                  --Comienza el proceso
    if enable='0' then a<='0';            --Sentencia if
                       b<='0';
       elsif enable='1' then              --Sentencia elsif
       case control is                    --Sentencia case
        when '0'=>a<='1';b<='0';
        when '1'=>b<='1';a<='0';
       end case;                          --Acaba sentencia case
   end if;                                --Acaban sentencias if y elsif
  end process;                            --Acaba el proceso
 end archdemulti;                         --Finaliza el programa
```

- *Ejercicios:*

a) ¿Cómo se habilita el dispositivo, a nivel bajo o alto?

b) A partir del código anterior, elaborar la tabla de verdad del dispositivo.

c) Introducir el ejemplo en el Editor de Texto. Simular desde el Editor de Texto y comprobar los resultados con la tabla de la verdad del apartado anterior.

4.5.- MULTIPLEXOR CON OCHO SEÑALES DE ENTRADA

Introducir en el Editor de Texto el código que se presenta a continuación, correspondiente a un multiplexor de ocho entradas de datos y tres señales de control:

```
entity multi is port(
 a,b,c,d,e,f,g,h :in bit;                    --8 canales de entrada
 enable:in bit;                              --señal de enable
 control:in bit_vector(2 downto 0);          --3 canales de control
 s:out bit                                   --1 canal de salida
 );
end multi;

architecture archmul of multi is             --Cabecera del programa
 begin                                       --Comienza el programa
  process(a,b,c,d,e,f,g,h,control,enable)    --Cabecera del proceso
   begin                                     --Comienza el proceso
    if enable ='1' then s<='0';              --Sentencia if
     elsif enable='0' then                   --Sentencia elsif
      case control is                        --Sentencia case
       when "000"=>s<=a;
       when "001"=>s<=b;
       when "010"=>s<=c;
       when "011"=>s<=d;
       when "100"=>s<=e;
       when "101"=>s<=f;
       when "110"=>s<=g;
       when "111"=>s<=h;
      end case;                              --Acaba sentencia case
     end if;                                 --Acaba sentencia if y elsif
    end process;                             --Acaba el proceso
   end archmul;                              --Finaliza el programa
```

- Ejercicios:

a) Analizar la definición de las entradas. ¿Qué tamaño tiene cada una de ellas?

b) Introducir el ejemplo en el Editor de Texto, simular desde el mismo y ver el comportamiento del dispositivo.

4.6.- DECODIFICADOR BCD-7SEGMENTOS

A continuación se presenta el programa para la implementación de un decodificador de código BCD a 7 segmentos. Este programa será de gran utilidad a la hora de volcar diseños en la tarjeta educacional para los cuales sea preciso el uso de los displays 7 segmentos.

El código del programa es el siguiente:

```
entity dec_7seg is port(
   hex_digit:in bit_vector(3 downto 0);
   segment_a, segment_b, segment_c, segment_d, segment_e,
   segment_f,segment_g: out bit
 );
end dec_7seg;

architecture arch_dec of dec_7seg is
  signal segment_data : bit_vector(6 downto 0);

 begin
   process (hex_digit)

                    ...
                    ...

 end arch_dec;
```

En la parte del código que falta, hay que asignar, dentro de una estructura process, la salida correspondiente en el display 7-segmentos para cada uno de los casos posibles. Esto significa que, por ejemplo, cuando la entrada sea '0000', en el display aparezca el número cero. Se trabajará dentro de dicha estructura con lógica positiva, con lo que, por ejemplo, en el caso anterior, se tendrá que asignar a la señal auxiliar segment-data '11111110'.

Como los displays de la tarjeta educacional DE2 (o UP2 en su caso) trabajan con lógica negativa, una vez fuera del proceso, se asignará a cada una de las señales de salida el bit correspondiente de la señal auxiliar utilizada para negarlo.

Los displays con sus correspondientes segmentos son:

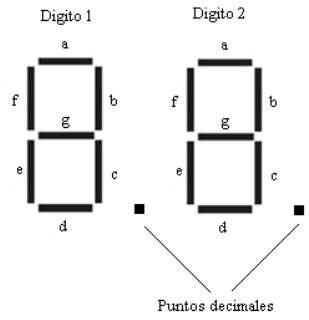


Figura 1. Displays de 7 segmentos

Nota 3: Una señal es un elemento auxiliar que puede ser modificado dentro de la arquitectura, pero que no puede ser utilizado fuera de ésta.

- *Ejercicios:*

a) Completar el código anterior e introducirlo utilizando el Editor de Texto.

b) Simularlo desde el Editor de Texto y ver su comportamiento.

c) Crear un símbolo para el decodificador, ya que será utilizado en prácticas posteriores.

4.7.- BLOQUE ELIMINA REBOTES

Si una señal que se va a utilizar como entrada se va a obtener mediante los pulsadores, hay que tener en cuenta que se producirán unos rebotes hasta que se estabilice la señal, lo que puede afectar al correcto funcionamiento del diseño. Por ello, se introduce un bloque denominado elimina-rebotes, el cual consigue esperar hasta que se estabilice la señal del pulsador.

El código es el siguiente:

```
entity elimina_rebotes is port(
 clock:in bit;
 con_rebotes:in bit;
 sin_rebotes:out bit
 );
end entity elimina_rebotes;

architecture archelimina_rebotes of elimina_rebotes is
 signal contador:integer range 0 to 15000;          --contador de 0 a 15000

  begin
   process (clock)
    begin
     if clock' event and clock='1' then
                    if contador = 10000 then
          sin_rebotes<=con_rebotes;                 --se asigna la entrada a la salida
          contador<=0;                              --se inicializa el contador
           else
            contador<=contador+1;                   --se incrementa en 1 el contador
                    end if;
          end if;
  end process;
 end archelimina_rebotes;
```

*Nota 4: La palabra clave **event** indica que se produce un flanco en la señal. En este caso si se produce un flanco de subida del reloj.*

- *Funcionamiento del programa elimina_rebotes*

Básicamente lo que hace el programa es recibir una señal con rebotes y sacar otra señal sin los mencionados rebotes. Para ello, cada vez que el contador alcanza el número 10000 (aproximadamente 393 μs), asigna a la señal de salida la de entrada y reinicializa el contador. Si no ha alcanzado este número, lo que hace es incrementar el contador en una unidad, sin obtener señal de salida.

- *Ejercicios:*

a) Introducir el ejemplo anterior en el Editor de Texto. No simular, debido a su complejidad.

b) Crear un símbolo para el bloque elimina-rebotes, ya que será utilizado en prácticas posteriores.

RESOLUCIÓN PRÁCTICA 4

4.1.- PUERTA AND DE DOS ENTRADAS

El código introducido ha sido el siguiente:

```
entity P_and2 is port(
 a,b : in bit;                          --2 canales de entrada
 c : out bit                            --1 canal de salida
 );
 end entity P_and2;

 architecture arch_and of P_and2 is
  begin                                 --Cabecera del programa
                c<=a and b;             --Comienza el programa
  end architecture arch_and;            --Finaliza el programa
```

Interpretando el código, se puede ver que se trata de un programa muy sencillo que representa el comportamiento de una puerta AND de dos entradas.

Tras incluir el código se procede a la compilación del mismo, proceso que detecta posibles errores de sintaxis, y a su correspondiente simulación, obteniendo los siguientes resultados:

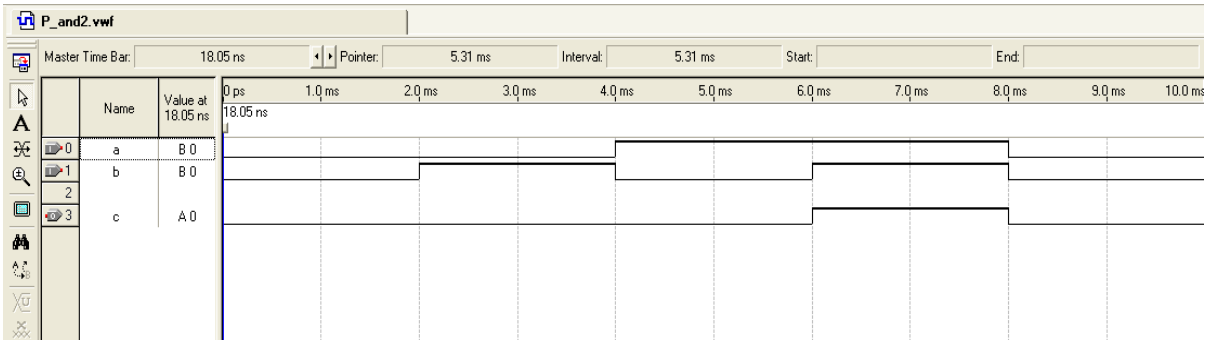


Figura 1. Puerta AND de 2 entradas

Como se puede apreciar en la figura, la señal de salida 'c' obtiene el resultado de una puerta AND de 2 entradas.

4.2.- PUERTAS AND, OR Y NOT

El código a introducir en este ejemplo es el mostrado a continuación:

```
entity and_or_not is port(
 a,b : in bit;                                  --2 canales de entrada
 c_or, c_and, c_not : out bit                   --3 canales de salida
);
end and_or_not;

architecture arch_aon of and_or_not is          --Cabecera del programa
 begin                                          --Comienza el programa
  a_o_x:process(a,b)                            --Cabecera de un proceso
   begin                                        --Empieza el proceso
    c_or<=a or b;
    c_and<=a and b;
    c_not<=not a;
   end process a_o_x;                           --Acaba el proceso
  end arch_aon;                                 --Finaliza el programa
```

- *Ejercicios:*

a) La simulación queda tal y como se muestra en la Figura 2.

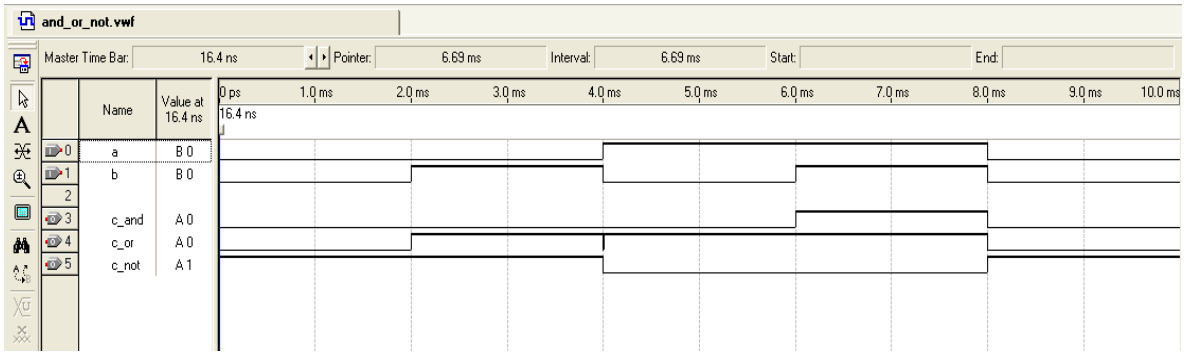
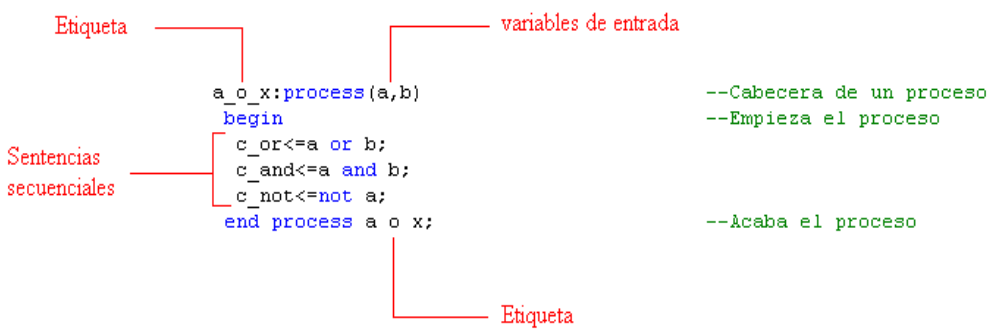


Figura 2. Resultados de puertas AND, OR y NOT

Si nos fijamos en las señales de salida, se comprueba fácilmente como éstas se corresponden con una puerta AND, OR y NOT respectivamente, ante las señales de entrada.

b)



c) El código para implementar una puerta NAND y una NOR es el siguiente:

```
entity nand_nor is port(
  a,b : in bit;                          --2 canales de entrada
  c_nand, c_nor : out bit                --2 canales de salida
 );
 end nand_nor;

 architecture arch_na_no of nand_nor is  --Cabecera del programa
  begin                                  --Comienza el programa
   na_no:process(a,b)                    --Cabecera de un proceso
    begin                                --Empieza el proceso
     c_nand<=a nand b;
     c_nor<=a nor b;
    end process na_no;                   --Acaba el proceso
   end arch_na_no;                       --Finaliza el programa
```

Tras realizar la simulación se obtienen los siguientes resultados:

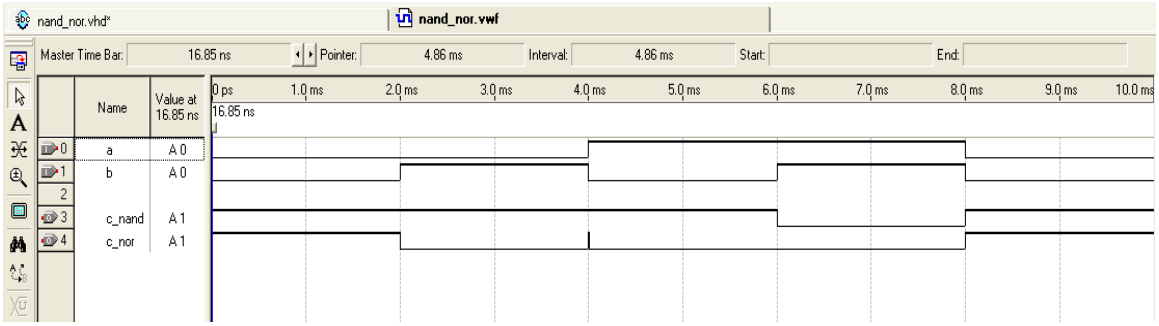


Figura 3. Resultados de las puertas NAND y NOR

Observando la Figura 3, se puede comprobar que los resultados son correctos.

4.3.- MULTIPLEXOR EN VHDL

El código introducido ha sido el siguiente:

```
entity multiplexor is port(
 a,b:in bit;                                  --2 canales de entrada
 control:in bit;                              --1 señal de control
 enable:in bit;                               --señal enable
 c:out bit                                    --1 canal de salida
 );
end multiplexor;

architecture archmul of multiplexor is        --Cabecera del programa
 begin                                        --Comienza el programa
  process(a,b,control,enable)                 --Cabecera de un proceso
   begin                                      --Empieza el proceso
    if enable='1' then c<='0';                --Sentencia if
     elsif enable='0' then                    --Sentencia elsif
      case control is                         --Sentencia case
       when '0' =>c<=a;
       when '1' =>c<=b;
       when others =>c<='0';
      end case;                               --Acaba sentencia case
     end if;                                  --Acaba sentencia if y elsif
    end process;                              --Acaba el proceso
   end archmul;                               --Finaliza el programa
```

- *Ejercicios:*

a) El tamaño de cada una de las señales de entrada y salida es de un bit.

b) La tabla de la verdad obtenida es la siguiente:

E	Control	C
1	X	0
0	0	a
0	1	b

Tabla 1

c) El circuito correspondiente al ejemplo, una vez realizado el símbolo, en el Editor Gráfico es el que se muestra en la Figura 4.

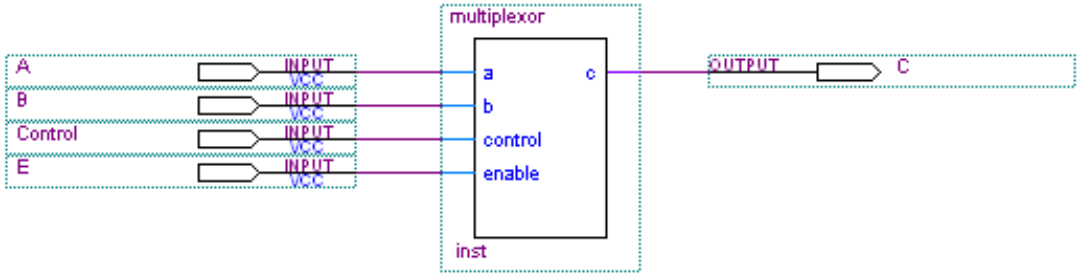


Figura 4. Multiplexor de dos entradas de datos.

A partir del circuito mostrado, la simulación queda de la siguiente manera:

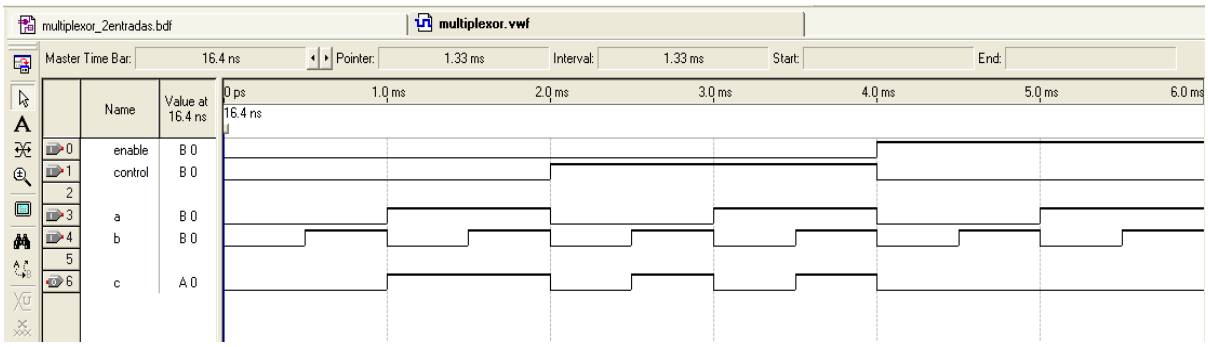


Figura 5. Resultados del multiplexor de dos entradas

Como se puede apreciar, mientras la señal de enable habilita el dispositivo, el sistema se comporta como un multiplexor de dos entradas y una señal de control.

4.4.- DEMULTIPLEXOR/DECODIFICADOR EN VHDL

El código introducido en el presente ejemplo es el siguiente:

```
entity demulti is port(
 control:in bit;                          --1 señal de control
 enable:in bit;                           --señal de enable
 a,b:out bit                              --2 canales de salida
 );
end demulti;

architecture archdemulti of demulti is    --Cabecera del programa
 begin                                    --Comienza el programa
  process (control,enable)                --Cabecera de un proceso
   begin                                  --Comienza el proceso
    if enable='0' then a<='0';            --Sentencia if
                       b<='0';
         elsif enable='1' then            --Sentencia elsif
         case control is                  --Sentencia case
          when '0'=>a<='1';b<='0';
          when '1'=>b<='1';a<='0';
         end case;                        --Acaba sentencia case
   end if;                                --Acaban sentencias if y elsif
  end process;                            --Acaba el proceso
 end archdemulti;                         --Finaliza el programa
```

- *Ejercicios:*

a) Si nos fijamos en las líneas correspondientes a la sentencia 'if' dentro de la arquitectura se puede comprobar como mediante la variable 'enable' el dispositivo queda activo mientras dicha variable esté a '1', es decir, a nivel alto.

b) La tabla de verdad de este demultiplexor/decodificador es la siguiente:

E	C	a	b
0	X	0	0
1	0	1	0
1	1	0	1

Tabla 2

c) El resultado de la simulación es el mostrado en la Figura 6.

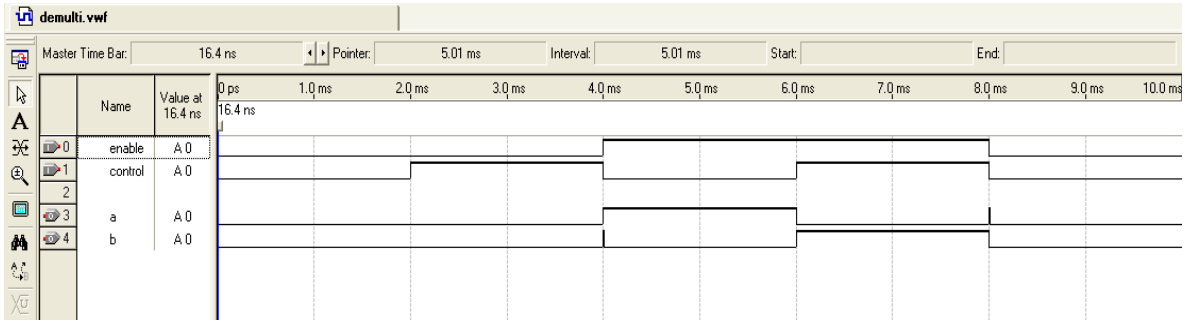


Figura 6. Resultados del demultiplexor/decodificador

Como se puede comprobar, el dispositivo se comporta de la manera esperada.

4.5.- MULTIPLEXOR CON OCHO SEÑALES DE ENTRADA

Este programa ofrece la posibilidad de ver un nuevo tipo de variable de entrada y cómo luego es empleada en la arquitectura del mismo.

El código empleado para el diseño es el siguiente:

```
entity multi is port(
 a,b,c,d,e,f,g,h :in bit;                    --8 canales de entrada
 enable:in bit;                              --señal de enable
 control:in bit_vector(2 downto 0);          --3 canales de control
 s:out bit                                   --1 canal de salida
 );
end multi;

architecture archmul of multi is             --Cabecera del programa
 begin                                       --Comienza el programa
  process(a,b,c,d,e,f,g,h,control,enable)    --Cabecera del proceso
   begin                                     --Comienza el proceso
    if enable ='1' then s<='0';              --Sentencia if
     elsif enable='0' then                   --Sentencia elsif
      case control is                        --Sentencia case
       when "000"=>s<=a;
       when "001"=>s<=b;
       when "010"=>s<=c;
       when "011"=>s<=d;
       when "100"=>s<=e;
       when "101"=>s<=f;
       when "110"=>s<=g;
       when "111"=>s<=h;
      end case;                              --Acaba sentencia case
     end if;                                 --Acaba sentencia if y elsif
    end process;                             --Acaba el proceso
   end archmul;                              --Finaliza el programa
```

- *Ejercicios:*

a) Las variables correspondientes a los 8 canales de entrada, a la señal de enable y a la salida son del tipo que hemos estado viendo hasta ahora, es decir, de tamaño igual a un bit. En cambio, la señal de control se trata de un vector de 3 bits, que luego es empleado en la arquitectura para realizar las combinaciones que van desde '000' hasta '111'.

b) A continuación se simula el diseño para comprobar su correcto funcionamiento, obteniendo los siguientes resultados:

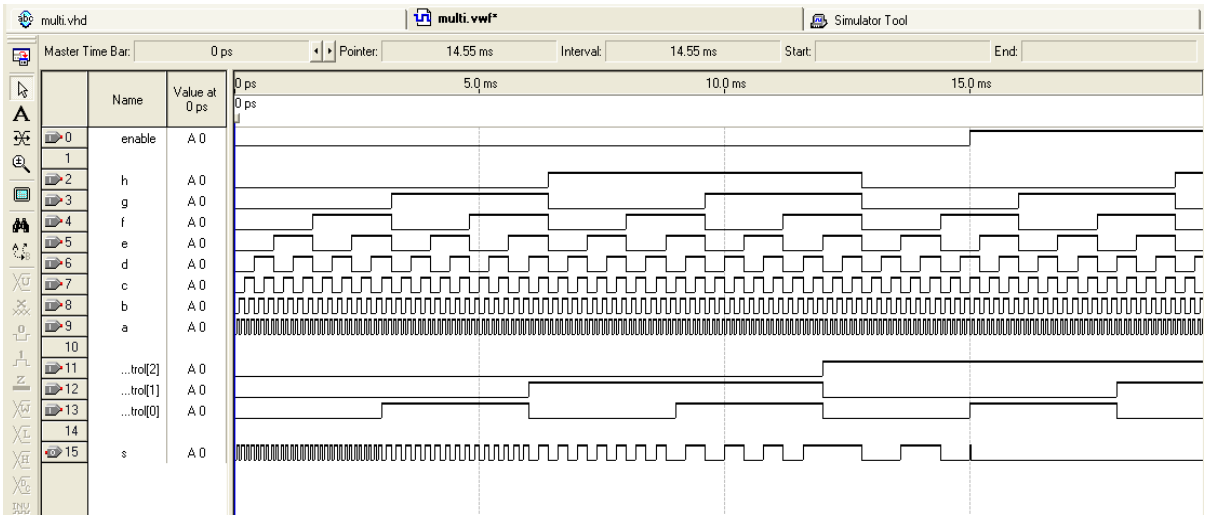


Figura 7. Resultados del multiplexor de 8 entradas

Como se puede comprobar en la Figura 7, el funcionamiento del programa es el correspondiente a un multiplexor de ocho canales de entrada y tres canales de control.

4.6.- DECODIFICADOR BCD-7SEGMENTOS

El código completo del programa es el siguiente:

```
entity dec_7seg is port(
  hex_digit:in bit_vector(3 downto 0);
  segment_a, segment_b, segment_c, segment_d, segment_e,
  segment_f,segment_g: out bit
 );
end dec_7seg;

architecture arch_dec of dec_7seg is
  signal segment_data : bit_vector(6 downto 0);

 begin
   process (hex_digit)
     begin
        case hex_digit is
          when "0000" =>segment_data<="1111110";
          when "0001" =>segment_data<="0110000";
          when "0010" =>segment_data<="1101101";
          when "0011" =>segment_data<="1111001";
          when "0100" =>segment_data<="0110011";
          when "0101" =>segment_data<="1011011";
          when "0110" =>segment_data<="1011111";
          when "0111" =>segment_data<="1110000";
          when "1000" =>segment_data<="1111111";
          when "1001" =>segment_data<="1111011";
          when others=>segment_data<="0000000";
        end case;
   end process;

                segment_a<=not segment_data(6);
                segment_b<=not segment_data(5);
                segment_c<=not segment_data(4);
                segment_d<=not segment_data(3);
                segment_e<=not segment_data(2);
                segment_f<=not segment_data(1);
                segment_g<=not segment_data(0);
end arch_dec;
```

Tras la simulación se obtienen los resultados mostrados en la Figura 8.

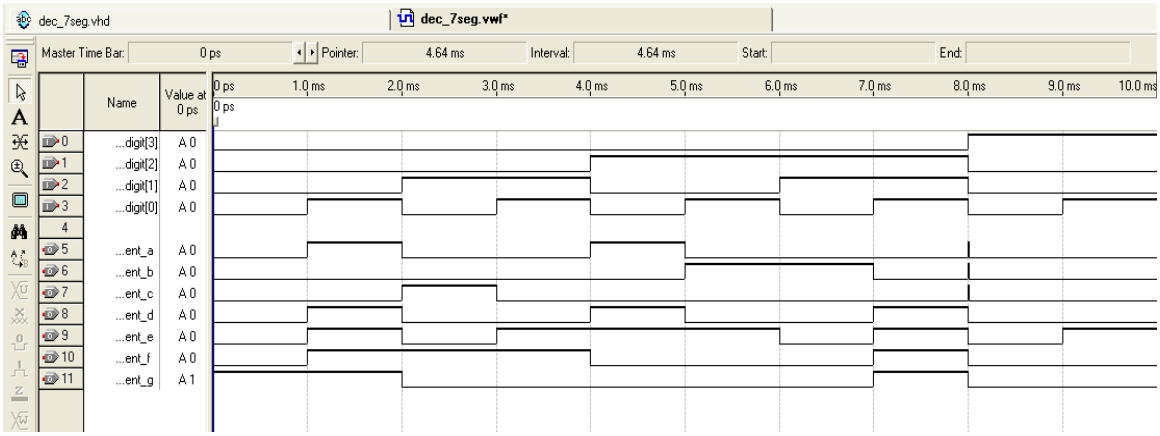


Figura 8. Resultados del decodificador BCD-7segmentos

En la siguiente figura se muestran los nombres de los distintos segmentos que componen el display.

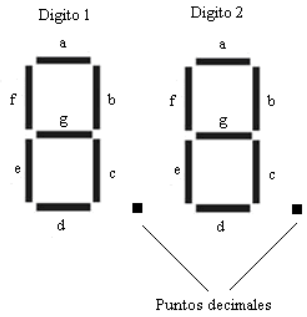


Figura 9. Display de 7 segmentos

A la hora de interpretar el resultado de la simulación, se debe tener en cuenta que los displays funcionan con lógica negativa (necesitan un cero lógico para que se iluminen). Por tanto, como se puede observar en la Figura 8, el resultado es correcto.

4.7.- BLOQUE ELIMINA – REBOTES

El código introducido para este ejercicio es el siguiente:

```
entity elimina_rebotes is port(
 clock:in bit;
 con_rebotes:in bit;
 sin_rebotes:out bit
 );
end entity elimina_rebotes;

architecture archelimina_rebotes of elimina_rebotes is
 signal contador:integer range 0 to 15000;          --contador de 0 a 15000

  begin
   process (clock)
    begin
     if clock' event and clock='1' then
                    if contador = 10000 then
            sin_rebotes<=con_rebotes;               --se asigna la entrada a la salida
            contador<=0;                            --se inicializa el contador
             else
              contador<=contador+1;                 --se incrementa en 1 el contador
                    end if;
            end if;
  end process;
 end archelimina_rebotes;
```

Por último se presenta el símbolo resultante del código anterior, y que será utilizado en posteriores prácticas:

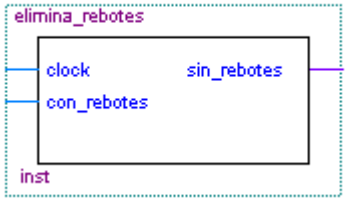


Figura 10. Símbolo Elimina-Rebotes

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

(1º Ingeniería Técnica en Informática de Gestión)

Práctica 5: Biestables y Contadores.

Los circuitos secuenciales se caracterizan porque las salidas están determinadas no sólo por las entradas existentes sino también por la secuencia de entradas que condujeron al estado existente, esto quiere decir que el circuito tiene memoria.

Uno de los tipos de circuitos secuenciales más común, y que va a ser estudiado en esta práctica, es el biestable.

1.- OBJETIVOS

a) Conocer el funcionamiento del elemento básico de los circuitos secuenciales, el biestable, y ver alguna de sus aplicaciones, tal como los contadores.
b) Conocer el funcionamiento de un biestable comercial, el 7476, que es un circuito biestable tipo JK disparado por flanco.
c) Simular y comprender el funcionamiento de dos contadores síncronos: uno ascendente y otro descendente.
d) Aprender a realizar el volcado del correspondiente diseño en la tarjeta educacional.

2.- MATERIAL

- Ordenador personal con el software Quartus II®.
- Manual introductorio al software.
- Guión de prácticas.
- Tarjeta educacional DE2 o UP2[1]

3.- DESARROLLO DE LA PRÁCTICA.

3.1.- BIESTABLE JK DISPARADO POR FLANCO DE SUBIDA:

En este ejercicio se va a realizar la simulación de un circuito biestable JK, con el objetivo de poder ver su funcionamiento, ya que es de especial importancia para el resto de los ejercicios.

Su símbolo se puede ver en la Figura 1.

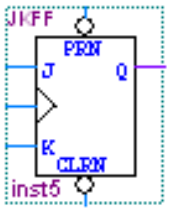


Figura 1. Biestable JK disparado por flanco de subida

[1] *En principio las prácticas están pensadas para realizarse sobre la tarjeta educacional DE2, si bien, por cualquier motivo, no hubiera un número suficiente de éstas, se emplearía la tarjeta educacional UP2. Por tanto, ambas tarjetas son tenidas en cuenta en este guión ya que presentan diferentes características.*

El biestable JK visto en la figura anterior forma parte de un circuito comercial como es el 7476, que se representa en la Figura 2.

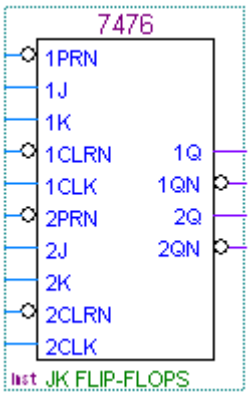


Figura 2. Circuito 7576.

Como se puede apreciar en la Figura 2, el circuito comercial 7476 está compuesto por dos biestables JK como el de la Figura 1. Por lo tanto, se pretende realizar la simulación del biestable JK, lo que a su vez servirá para conocer el funcionamiento del circuito 7476.

El circuito a introducir en el Editor Gráfico es mostrado en la Figura 3. Para la inclusión del biestable JK, introducir 'jkff' en el correspondiente apartado de la ventana de introducción de símbolos.

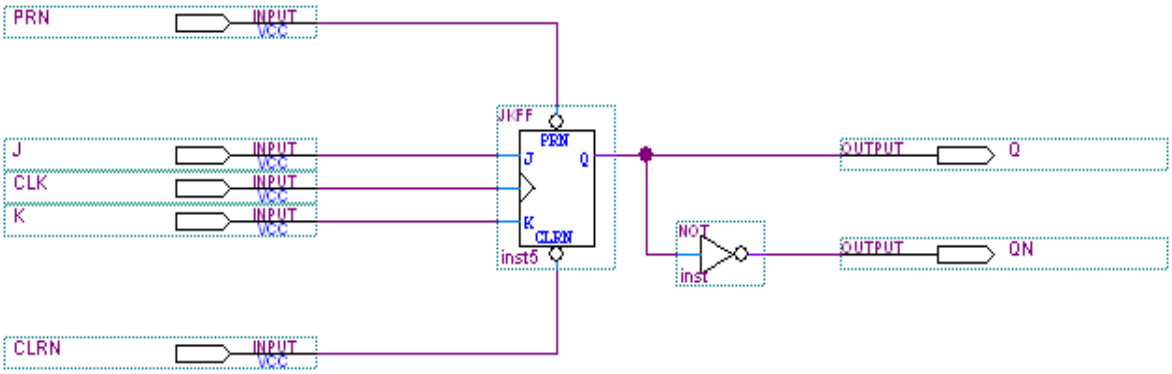


Figura 3. Circuito con biestable JK

Simular el diseño, eligiendo una frecuencia apropiada para la señal de reloj, comprender y explicar el funcionamiento en cada una de las distintas situaciones, ayudándose de la tabla de excitaciones del biestable JK, mostrada a continuación.

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabla 1

3.2.- CONTADOR SÍNCRONO ASCENDENTE DE MODULO 10.

En este ejercicio se pretende realizar un contador que cuente de cero a nueve, y una vez llegado a ese punto, inicialice la cuenta desde cero.

Introducir el diseño mostrado en la Figura 4 en el Editor Gráfico, utilizando para ello el símbolo 'jkff' también utilizado en el apartado anterior.

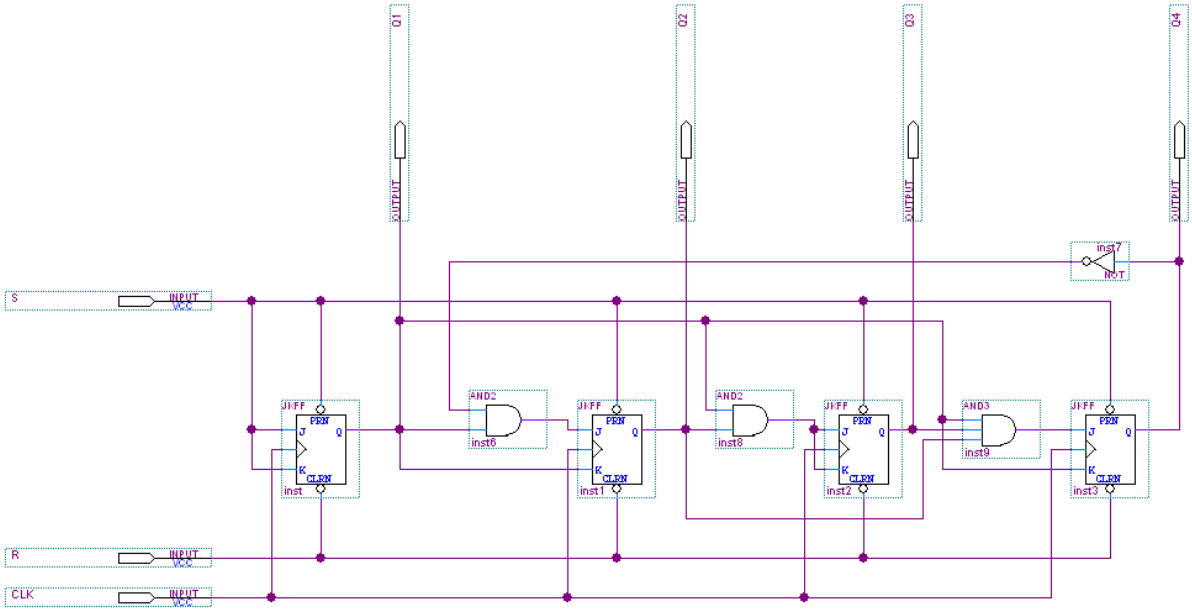


Figura 4. Circuito del contador ascendente síncrono módulo 10.

Realizar la simulación del diseño de la Figura 4 para poder ver el comportamiento del mismo. Para ello, elegir, por ejemplo, un tiempo de simulación de 25 ms y una frecuencia para la señal de reloj de 1 KHz (se pueden escoger otros valores siempre que permitan ver el resultado con claridad). Dar al inicio manualmente un valor lógico de '1' para la señal 'S' de entrada y un valor lógico de '0' a la señal 'R' durante, por ejemplo, un tiempo de 2 ms, para así inicializar el contador a Q1=Q2=Q3=Q4=0. A partir de los 2 ms, dar a la señal 'R' un valor lógico de '1' para que de esta manera las entradas asíncronas queden desactivadas.

- Observar que el circuito se comporta como un contador de módulo 10 ascendente.
- Crear un símbolo para el contador.

3.3.- VOLCADO DEL CONTADOR SÍNCRONO ASCENDENTE DE MÓDULO 10

El objetivo de este ejercicio es el de aprender a realizar volcados de los correspondientes diseños a la tarjeta educacional de forma que se pueda ver su funcionamiento en la misma. Para ello, hay a varios aspectos que hay que tener en cuenta ya que si se realiza el volcado del diseño tal y como está hecho hasta ahora no funcionará correctamente en la placa.

En primer lugar, para que resulte más sencillo de ver, se debe utilizar el decodificador BCD-7 segmentos visto en la práctica anterior, de forma que los números aparezcan en los displays y así no tener que utilizar para ello los diodos Led.

En segundo lugar, como la señal de reloj 'CLK' se va ha obtener mediante un pulsador de la tarjeta educacional, hay que tener en cuenta que se producirán unos rebotes hasta que se estabilice la señal, lo que puede afectar al correcto funcionamiento del diseño.

Utilizando la placa DE2 estos rebotes apenas son apreciables, si bien para asegurarnos del correcto funcionamiento del circuito es necesario emplear el bloque elimina-rebotes, diseñado en la práctica cuatro. Hay que incluir el archivo 'elimina-rebotes.vhd',y su correspondiente símbolo en cada uno de los proyectos en los que sea requerido.

En el caso de utilizar la tarjeta educacional UP2, es imprescindible la utilización del bloque elimina-rebotes para que el diseño funcione correctamente en la placa.

A partir de los símbolos del contador de módulo 10, decodificador BCD-7 segmentos y bloque elimina-rebotes, realizar el diseño mostrado en la Figura 5.

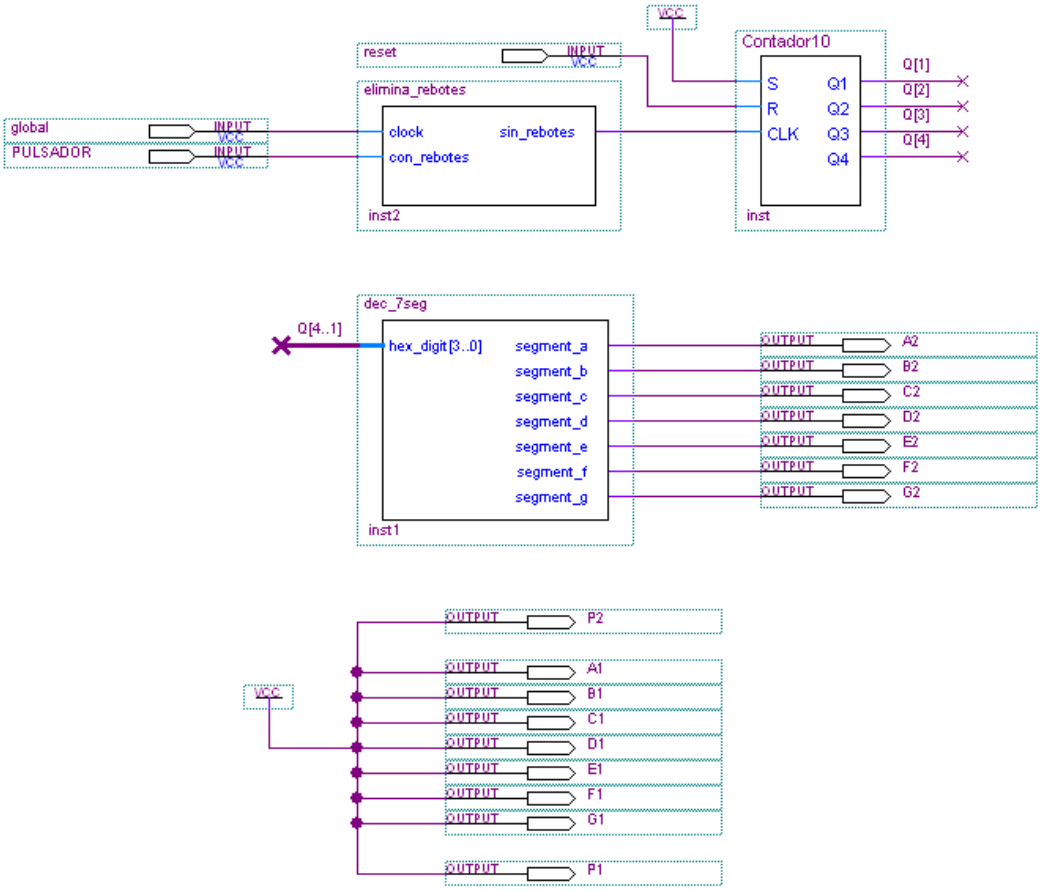


Figura 5. Contador síncrono ascendente de módulo 10

El bloque de conexiones A1-G1, se debe a todos los segmentos del display uno. Los Led de la placa trabajan con lógica negativa, por lo que si se quiere que éstos permanezcan apagados, el diseño debe incluir este bloque, de forma que dichos Led estén conectados a VCC.

Las entradas denominadas P2 y P1 que se pueden apreciar en el circuito de la Figura 5, han sido incluidas para el caso de querer implementar este circuito en la tarjeta educacional UP2 (para la tarjeta DE2 no son necesarias). Estas entradas se corresponden con los puntos

decimales de sus displays y, como en este caso también trabajan con lógica negativa, se conectan a VCC para que permanezcan apagados.

Realizar la simulación del diseño y volcado a la tarjeta educacional siguiendo los pasos que se explican a continuación. Observar el funcionamiento en la placa y comparar con lo obtenido en la simulación.

- Volcado en la tarjeta educacional DE2:

A continuación se presenta la serie de pasos que permiten la configuración y volcado de los respectivos diseños a la tarjeta educacional DE2.

Tener en cuenta que en el caso de no haber escogido el dispositivo EP2C35F672C6 de la familia Cyclone II a la hora de definir el proyecto, es necesario acceder a la ventana mostrada en la Figura 6, donde se escoge la familia y dispositivo a programar. Para ello acceder al menú 'Assignments' seguido de la opción 'Device'.

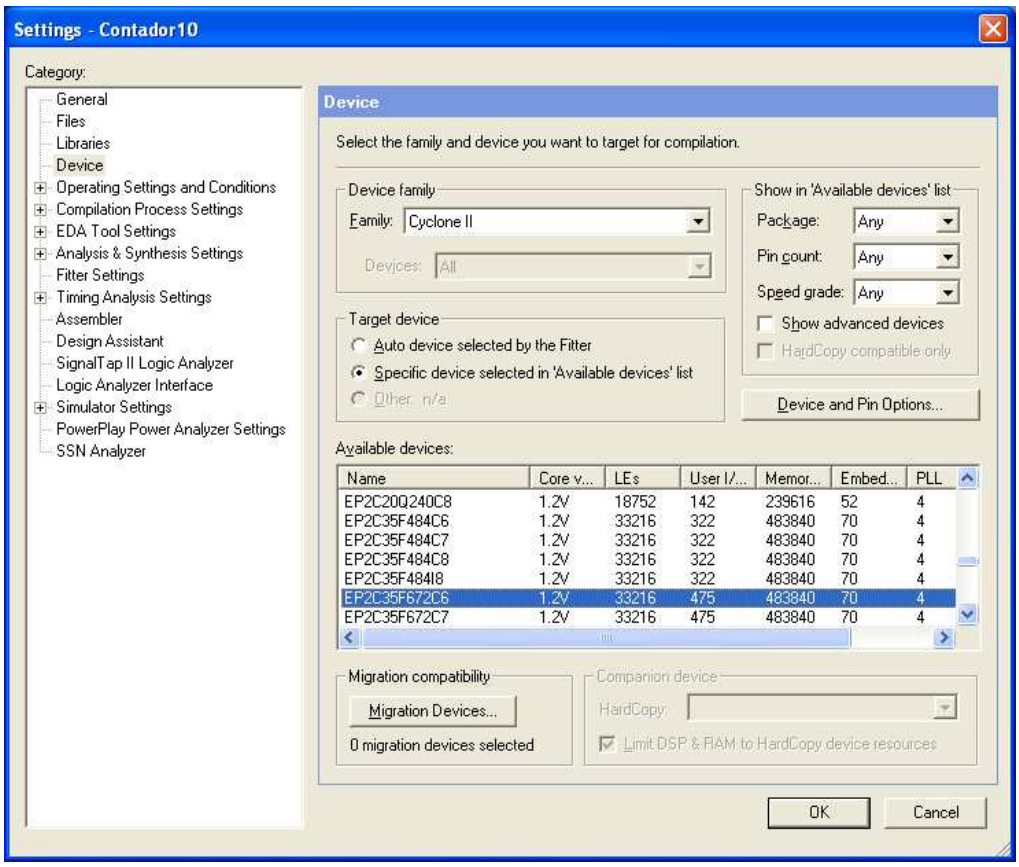


Figura 6. Selección del dispositivo a programar

1.- Asignación de pines:

Durante el proceso de compilación, el compilador de Quartus II es libre de elegir cualquier pin de la FPGA seleccionada que le sirvan de entradas y salidas. Sin embargo, la placa DE2 requiere especificar las conexiones entre la FPGA y el resto de componentes de la placa.

Para realizar la especificación de los pines es necesario utilizar la herramienta 'Assignment Editor' situada dentro del menú desplegable 'Assignments' de la barra de

herramientas principal. Siguiendo estos pasos se obtiene la ventana que se muestra a continuación:

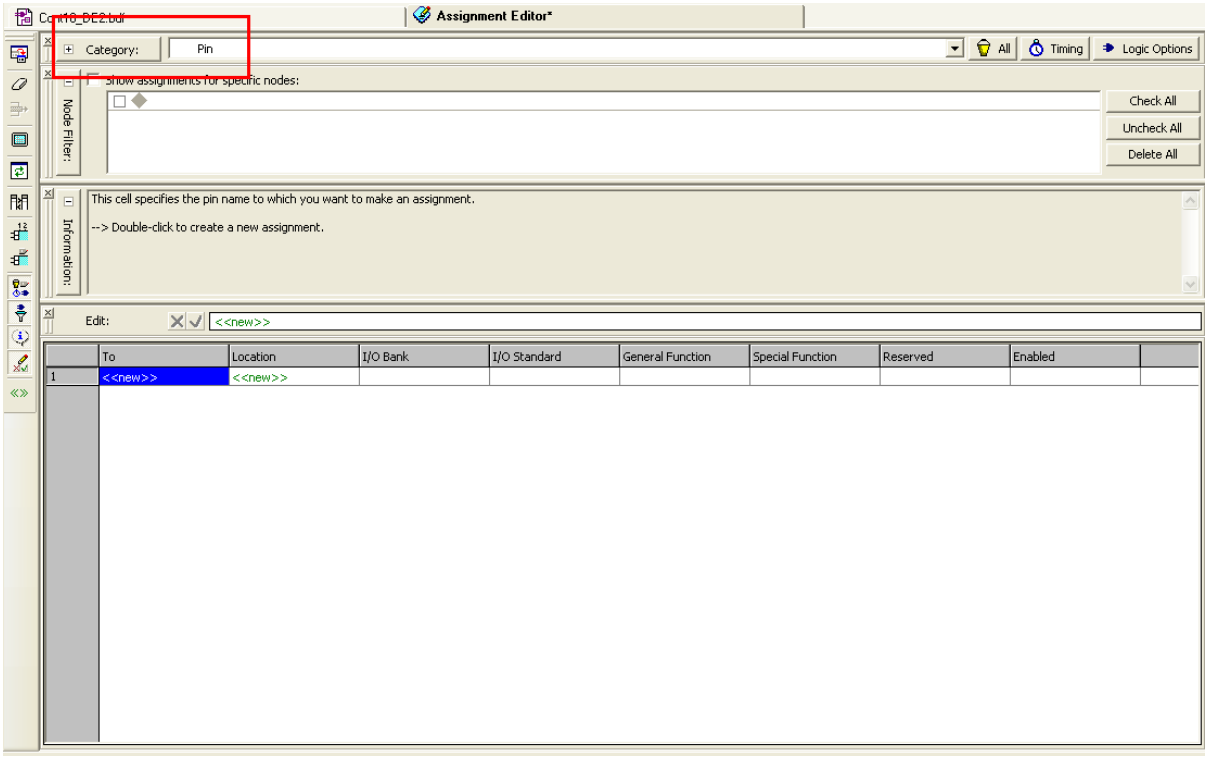


Figura 7. Ventana de la herramienta de asignación de pines

- Nota: es posible que al iniciar esta herramienta no tengamos todas las opciones que se pueden ver en la ventana de la Figura 7. Para solucionar esto basta con acceder al menú 'Window' de la barra de herramientas principal y seleccionar la opción 'Detach Window'. En este momento la herramienta pasa a estar en una ventana independiente, donde a través del menú 'View' se pueden cargar las opciones que faltan ('Category Bar', 'Node Filter Bar', 'Information Bar', 'Edit Bar').

Observando la figura anterior, para comenzar con la asignación de pines, seleccionar bajo el apartado categoría, recuadrado en rojo, la opción 'pin'. A continuación hacer doble clic sobre la entrada <<new>> sombreada en color azul, obteniendo un menú desplegable en el que aparecen cada uno de los puertos que se pretenden asignar, tal y como muestra la Figura 8.

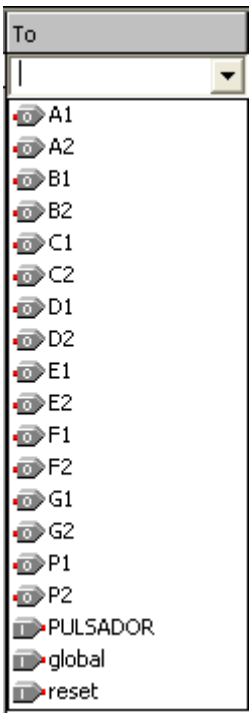


Figura 8. Menú desplegable con los nombres de las entradas y salidas

A partir de este menú, seleccionar el pin que va a ser asignado y, a continuación, acudir a la columna denominada 'Location' haciendo doble clic sobre su correspondiente celda para que aparezca el menú desplegable que permite asignar el pin con las características exigidas por cada entrada. Este menú es mostrado en la Figura 9.

Location	I/O Bank	I/O Standard	General Function	Special Fun
		3.3-V LVTTL		
PIN_M2	I/O Bank 2	Row I/O	LVDS27n	
PIN_M3	I/O Bank 2	Row I/O	LVDS27p, DPCLK0/DQS0L/CQ1L	
PIN_M4	I/O Bank 2	Row I/O	LVDS28p	
PIN_M5	I/O Bank 2	Row I/O	LVDS28n	
PIN_M19	I/O Bank 5	Row I/O	LVDS123p	
PIN_M20	I/O Bank 5	Row I/O	LVDS123n	
PIN_M21	I/O Bank 5	Row I/O	LVDS124n	
PIN_M22	I/O Bank 5	Row I/O	LVDS122p	
PIN_M23	I/O Bank 5	Row I/O	LVDS122n	
PIN_M24	I/O Bank 5	Row I/O	LVDS125p	
PIN_M25	I/O Bank 5	Row I/O	LVDS125n	
PIN_N1	I/O Bank 2	Dedicated Clock	CLK1, LVDSCLK0n, Input	
PIN_N2	I/O Bank 2	Dedicated Clock	CLK0, LVDSCLK0p, Input	
PIN_N9	I/O Bank 2	Row I/O	LVDS31p	
PIN_N18	I/O Bank 5	Row I/O	LVDS110p	
PIN_N20	I/O Bank 5	Row I/O	LVDS124p	
PIN_N23	I/O Bank 5	Row I/O	LVDS126p, DPCLK7/DQS0R/CQ1R	
PIN_N24	I/O Bank 5	Row I/O	LVDS126n	
PIN_N25	I/O Bank 5	Dedicated Clock	CLK4, LVDSCLK2p, Input	
PIN_N26	I/O Bank 5	Dedicated Clock	CLK5, LVDSCLK2n, Input	

Figura 9. Ventana de pines disponibles

Desde esta ventana se puede asignar la disposición de pines que interesa dada la configuración de la tarjeta educacional. La siguiente tabla muestra la disposición de pines que se va a utilizar:

Señal	Pin
PULSADOR	PIN_G26
Reset	PIN_P23
Global	PIN_D13
A2	PIN_R2
B2	PIN_P4
C2	PIN_P3
D2	PIN_M2
E2	PIN_M3
F2	PIN_M5
G2	PIN_M4
A1	PIN_L3
B1	PIN_L2
C1	PIN_L9
D1	PIN_L6
E1	PIN_L7
F1	PIN_P9
G1	PIN_N9

Tabla 2.

Mediante esta asignación, se están utilizando los dos pulsadores situados más a la derecha de la placa, una señal de reloj de 27 MHz, y la pareja de displays situados más a la izquierda de la placa.

La disposición de los pines se puede consultar en las tablas presentes en el Anexo 2 del manual.

Una vez que se ha terminado con la asignación de pines para este diseño, guardar la operación y volver a compilar para que el dispositivo se quede con su nueva configuración. Además, el usuario puede usar esta misma asignación de pines para posteriores diseños en la que sea requerida. Acudiendo al menú 'File' situado en la barra de herramientas principal mediante la opción 'Export' se crea un archivo con extensión '.csv' capaz de ser leído por el software Microsoft Excel. A continuación, en un nuevo proyecto, es posible importar esta asignación escogiendo el menú 'Assignments' seguido de la opción 'Import Assignments' seleccionando el archivo con extensión '.csv' que se desea importar.

2.- Programación:

El dispositivo FPGA debe ser programado y configurado si se quiere implementar el circuito diseñado. El archivo de configuración requerido para la operación ha sido creado previamente en el proceso de compilación.

La placa DE2 de Altera puede ser configurada de dos diferentes maneras: modo JTAG y modo AS, siendo el primero el que va a ser utilizado en las prácticas. Los datos de configuración son enviados desde el PC, con el software Quartus II activo, a través de un

puerto USB cualquiera, hasta la placa a través del puerto USB situado más a la izquierda de la misma, junto a la entrada de alimentación. Es fundamental, para realizar esta conexión tener el controlador USB-Blaster instalado.

- **Modo de programación JTAG:**

En primer lugar, la placa DE2 cuenta con un conmutador de dos posiciones RUN/PROG, el cual ha de ser situado en la posición RUN.

A continuación, en el software Quartus II, acceder al menú 'Tools' situado en la barra de herramientas principal y seleccionar la opción 'Programmer'. También es posible acceder directamente a su icono.

Siguiendo estos pasos, se obtiene la ventana mostrada a continuación:

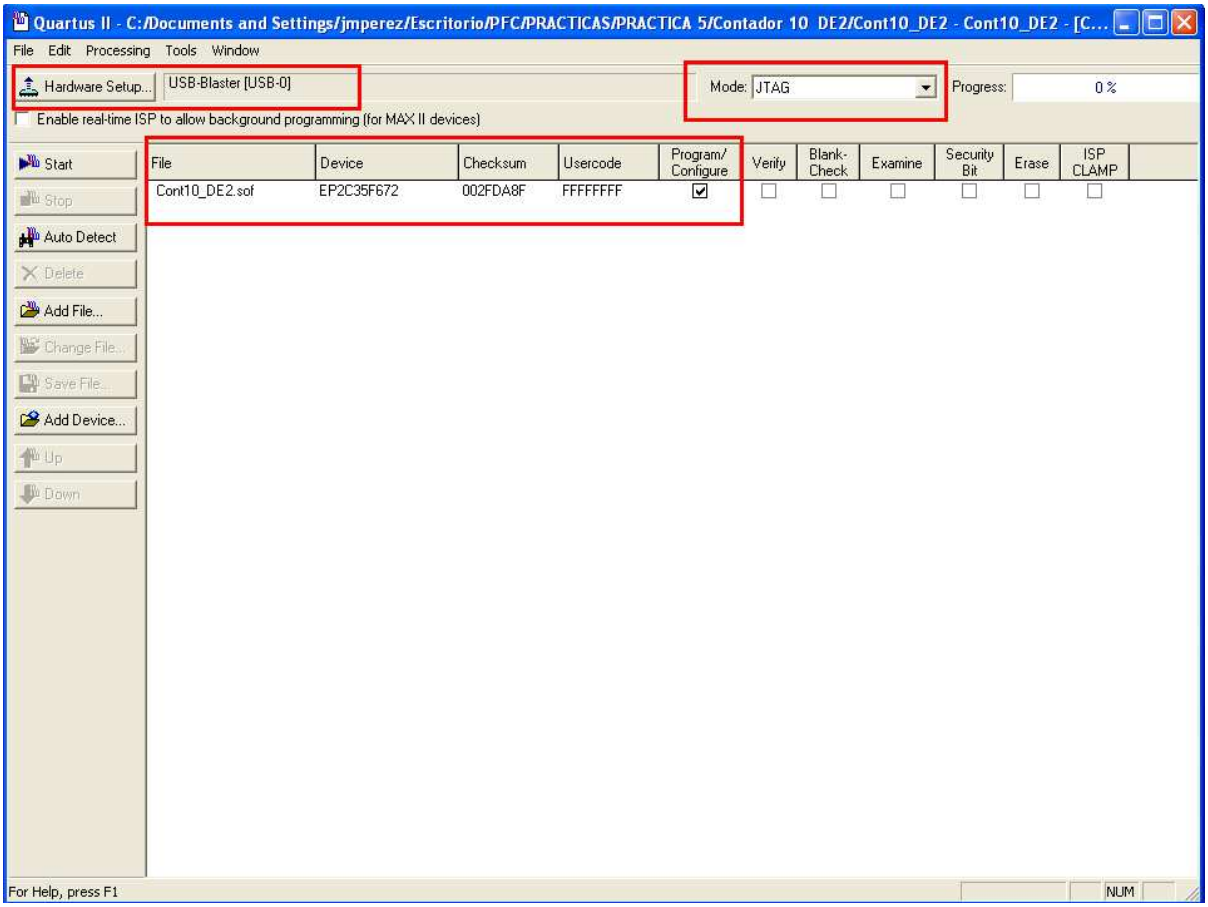


Figura 10. Ventana del Programador

En esta ventana es necesario especificar tanto el hardware como el modo de programación que va a ser utilizado. Si estos parámetros no vienen ya dados por defecto, seleccionar JTAG en la casilla 'Mode'. Asimismo, si el hardware USB-Blaster no ha sido el elegido, pulsar sobre el botón 'Hardware Setup' situado en la parte superior izquierda de la ventana y seleccionar la opción USB-Blaster en la nueva ventana, mostrada en la Figura 11.

Además se observa en la Figura 10, como el archivo de programación, con extensión '.sof', también ha sido cargado por defecto. En caso de no ser así, pulsar sobre el botón 'Add Files' de la barra de herramientas adjunta al programador. Éste es el archivo de configuración creado por el compilador y que incluye toda la información necesaria para la programación del dispositivo FPGA.

Comprobar también que el dispositivo que aparece en la misma línea es el EP2C35F672, que es el correspondiente a la placa DE2. Si todo es correcto, hacer clic sobre la casilla de la opción 'Program/Configure' y presionar el botón 'Start' de la barra de herramientas. Cuando la programación está siendo realizada, aparecen los correspondientes mensajes en la ventana de mensajes, así como la barra de progreso, que se encuentra en la parte superior derecha, muestra el estado de la programación. Un diodo LED de la placa se ilumina cuando la configuración del dispositivo ha concluido satisfactoriamente.

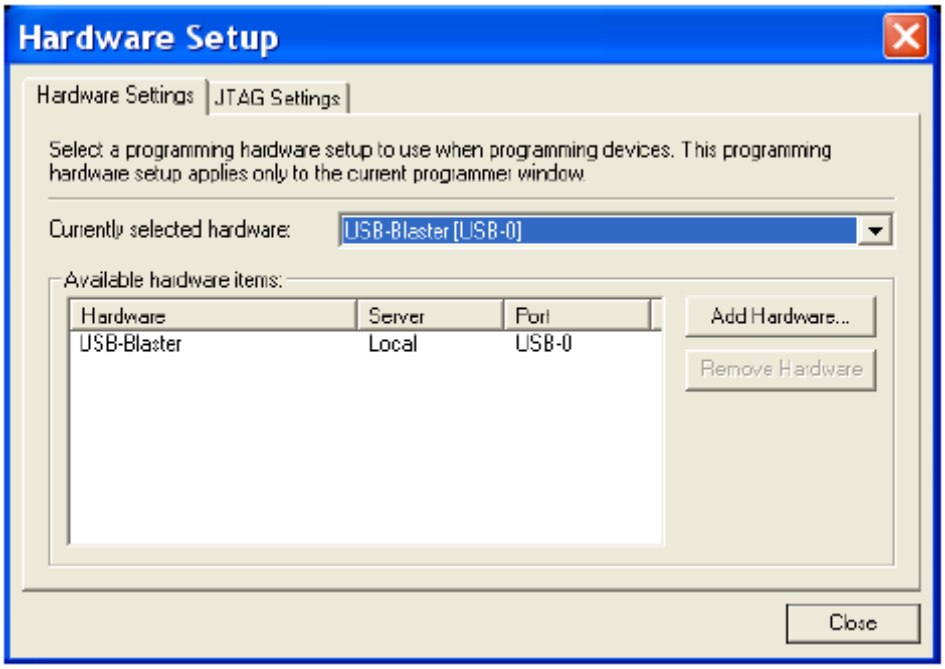


Figura 11. Ventana de selección de hardware

- Volcado en la tarjeta educacional UP2:

A continuación se presenta una breve descripción de cómo programar la tarjeta educacional UP2, es decir, cómo volcar diseños a la misma.

En primer lugar es necesario comprobar la configuración de los puentes sobre la propia tarjeta, optando por la configuración que se muestra en la Figura 12 como la adecuada para programar únicamente el dispositivo de la familia FLEX 10K.

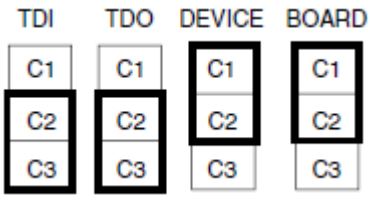


Figura 12. Disposición de los puentes de configuración de la placa

Asimismo, comprobar que el dispositivo elegido en la definición del proyecto se corresponde con la familia de dispositivos FLEX 10K dentro de la cual se encuentra el modelo EPF10K70RC240-4. Para ello acceder al menú 'Assignments' de la barra de herramientas principal y seleccionar la opción 'Device'.

A continuación, es necesario asignar a cada uno de los puertos de entrada y salida el correspondiente pin del dispositivo a programar. Mediante la ventana de asignación de pines, se asignan a los puertos de entrada y salida del diseño los pines correspondientes del dispositivo PLD que se está utilizando. El acceso a esta ventana se obtiene desde el menú 'Assignments' seguido de la opción 'Assignment Editor', obteniendo una ventana como la mostrada en la Figura 13.

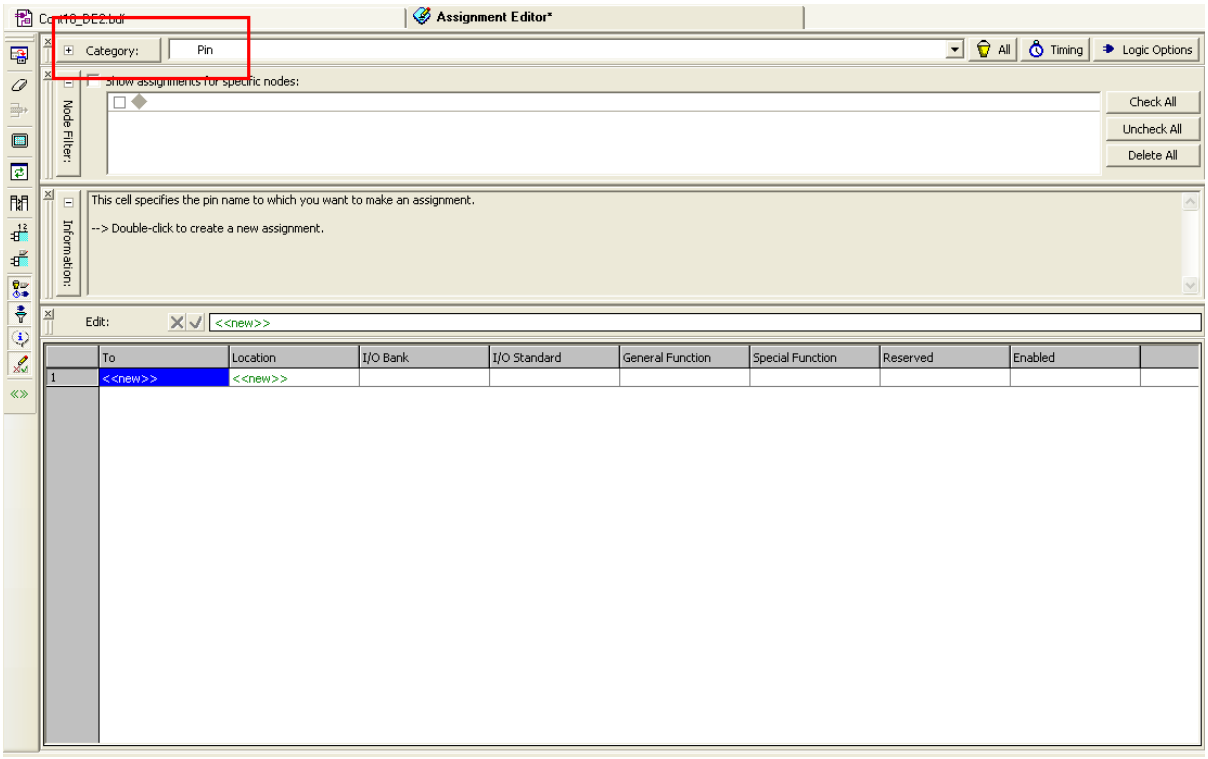


Figura13. Ventana de la herramienta de asignación de pines

Observando la figura anterior, para comenzar con la asignación de pines, seleccionar bajo el apartado categoría, recuadrado en rojo, la opción 'pin'. A continuación hacer doble click sobre la entrada <<new>> sombreada en color azul, obteniendo un menú desplegable en el que aparecen cada uno de los puertos que se pretenden asignar, tal y como muestra la Figura 14.

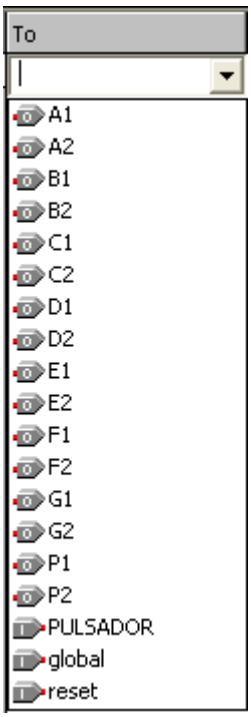


Figura 14. Menú desplegable con los nombres de las entradas y salidas

A partir de este menú, seleccionar el pin que va a ser asignado y, a continuación, acudir a la columna denominada 'Location' haciendo doble clic sobre su correspondiente celda para que aparezca el menú desplegable que permite asignar el pin con las características exigidas por cada entrada. Este menú es mostrado en la Figura 15.

Location	I/O Bank	I/O Standard	General Function	Special Fun
		3.3-V LVTTL		
PIN_M2	I/O Bank 2	Row I/O	LVDS27n	
PIN_M3	I/O Bank 2	Row I/O	LVDS27p, DPCLK0/DQS0L/CQ1L	
PIN_M4	I/O Bank 2	Row I/O	LVDS28p	
PIN_M5	I/O Bank 2	Row I/O	LVDS28n	
PIN_M19	I/O Bank 5	Row I/O	LVDS123p	
PIN_M20	I/O Bank 5	Row I/O	LVDS123n	
PIN_M21	I/O Bank 5	Row I/O	LVDS124n	
PIN_M22	I/O Bank 5	Row I/O	LVDS122p	
PIN_M23	I/O Bank 5	Row I/O	LVDS122n	
PIN_M24	I/O Bank 5	Row I/O	LVDS125p	
PIN_M25	I/O Bank 5	Row I/O	LVDS125n	
PIN_N1	I/O Bank 2	Dedicated Clock	CLK1, LVDSCLK0n, Input	
PIN_N2	I/O Bank 2	Dedicated Clock	CLK0, LVDSCLK0p, Input	
PIN_N9	I/O Bank 2	Row I/O	LVDS31p	
PIN_N18	I/O Bank 5	Row I/O	LVDS110p	
PIN_N20	I/O Bank 5	Row I/O	LVDS124p	
PIN_N23	I/O Bank 5	Row I/O	LVDS126p, DPCLK7/DQS0R/CQ1R	
PIN_N24	I/O Bank 5	Row I/O	LVDS126n	
PIN_N25	I/O Bank 5	Dedicated Clock	CLK4, LVDSCLK2p, Input	
PIN_N26	I/O Bank 5	Dedicated Clock	CLK5, LVDSCLK2n, Input	

Figura 15.Ventana de pines disponibles.

Desde esta ventana se puede asignar la disposición de pines que interesa dada la configuración de la tarjeta educacional. La siguiente tabla muestra la disposición de pines que se va a utilizar:

Señal	Pin	Señal	Pin
PULSADOR	Input Pin =28	P1	14 (1=LED OFF)
Reset	Input Pin = 29	P2	25 (1=LED OFF)
Global	Clock Pin = 91	A1	6 (1=LED OFF)
A2	Output Pin =17	B1	7 (1=LED OFF)
B2	Output Pin =18	C1	8 (1=LED OFF)
C2	Output Pin =19	D1	9 (1=LED OFF)
D2	Output Pin =20	E1	11 (1=LED OFF)
E2	Output Pin =21	F1	12 (1=LED OFF)
F2	Output Pin =23	G1	13 (1=LED OFF)
G2	Output Pin =24	-	-

Tabla 3.

Una vez que se ha terminado con la asignación de pines para este diseño, guardar la operación y volver a compilar para que el dispositivo se quede con su nueva configuración. Además, el usuario puede usar esta misma asignación de pines para posteriores diseños en la que sea requerida. Acudiendo al menú 'File' situado en la barra de herramientas principal mediante la opción 'Export' se crea un archivo con extensión '.csv' capaz de ser leído por el software Microsoft Excel. A continuación, en un nuevo proyecto, es posible importar esta asignación escogiendo el menú 'Assignments' seguido de la opción 'Import Assignments' seleccionando el archivo con extensión '.csv' que se desea importar.

La disposición de los pines se puede consultar en las tablas presentes en el Anexo 1 del manual.

Una vez que se tienen asignadas las entradas y salidas a los terminales correspondientes, se puede proceder a la programación mediante la herramienta que Quartus II tiene para ello.

Para comenzar la programación del dispositivo, acceder bien al menú 'Tools' y seleccionar la opción 'Programmer' o bien acudir directamente a su icono.

La ventana de esta herramienta es la mostrada en la Figura 16.

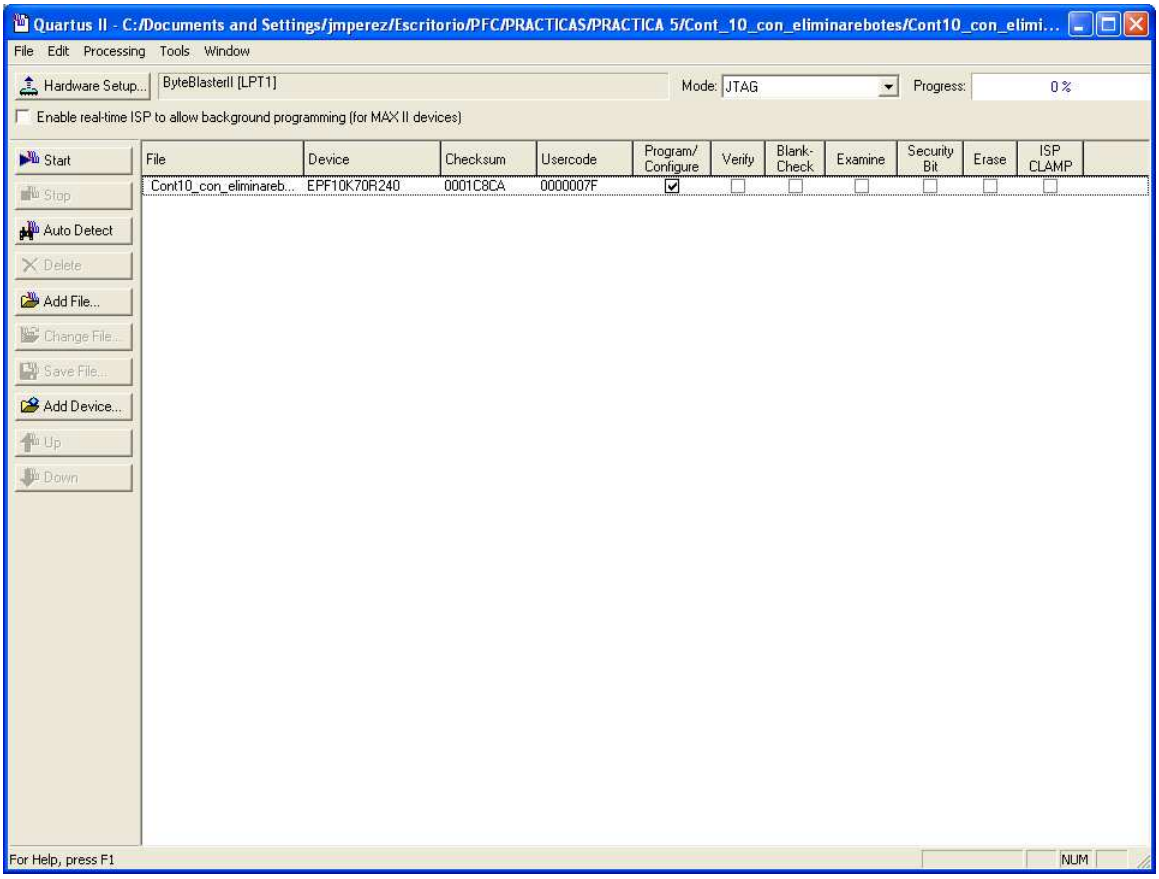


Figura 16. Ventana del programador.

En esta ventana es posible asignar el hardware usado para la programación del dispositivo, seleccionar el archivo de diseño que va a ser volcado (con extensión '.sof' en nuestro caso y generado por el compilador) y especificar el propio dispositivo que va a ser programado. En general, todo esto vendrá asignado por defecto en la ventana de configuración, de todas maneras asegurarse de que aparece lo siguiente:

- Hardware Setup: ByteBlaster II (LPTI)
- Mode: JTAG
- File: archivo de diseño a volcar con extensión '.sof'.
- Device: EPF10K70R240.
- Opción 'Program/Configure' seleccionada.

En el caso de que alguno de estos parámetros no aparezca o no sea el adecuado, hacer uso de la barra de herramientas asociada al programador situada en la parte izquierda de la ventana, la cual permite añadir o borrar archivos de programación y realizar la selección del dispositivo adecuado.

Además, es posible que la primera vez que se haga uso de la herramienta, no aparezca el hardware de configuración ByteBlaster II, por lo que pulsando sobre el botón 'Hardware Setup' surge una nueva ventana que permite añadir el mismo.

Una vez hecha la configuración en la ventana, pulsar el botón 'Start' que da comienzo a la simulación. Cuando la programación está siendo realizada, aparecen los correspondientes mensajes en la ventana de mensajes, así como la barra de progreso, que se encuentra en la parte superior derecha, muestra el estado de la programación.

Por último, una vez que la tarjeta se encuentre programada, comprobar el correcto funcionamiento del diseño.

4.- EJERCICIOS DE DISEÑO

4.1.- Contador de módulo 5

4.1.1.- Diseñar un contador síncrono ascendente de módulo 5. Para ello, utilizar biestables del tipo JK y las puertas lógicas que sean necesarias.

Comprobar el correcto funcionamiento del diseño realizando su simulación y correspondiente volcado a la placa educacional DE2 o UP2.

Crear un símbolo para el diseño, ya que será utilizado en prácticas posteriores.

4.1.2.- En un nuevo proyecto, modificar el diseño anterior añadiéndole un pulsador que permita elegir el sentido, con lo que se obtiene un contador síncrono reversible.

Comprobar el funcionamiento del diseño realizando su simulación y volcado a la tarjeta educacional DE2 o UP2.

RESOLUCIÓN PRÁCTICA 5

A continuación se presenta la resolución de los diferentes ejercicios que componen esta práctica.

3.- DESARROLLO DE LA PRÁCTICA

3.1.- BIESTABLE JK DISPARADO POR FLANCO DE SUBIDA

El diseño a introducir en el Editor Gráfico es el siguiente:

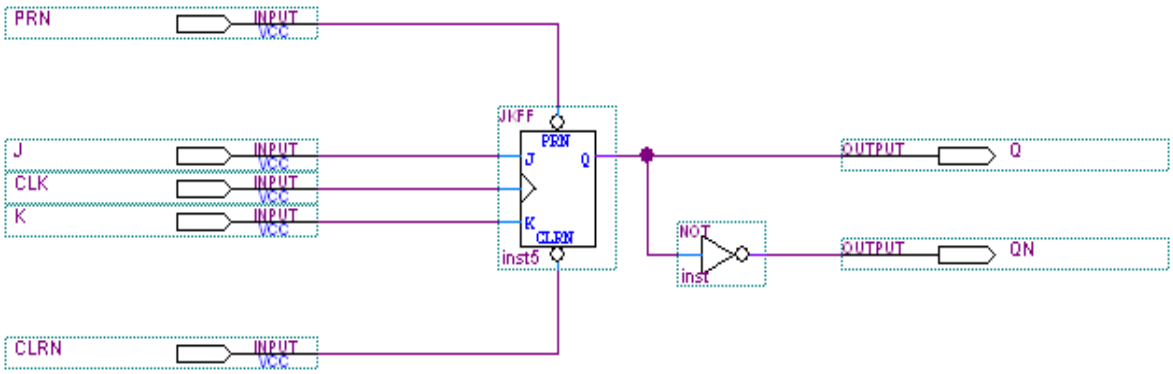


Figura 1. Circuito con biestable JK

A partir de este circuito, realizando su correspondiente simulación, se obtienen los siguientes resultados:

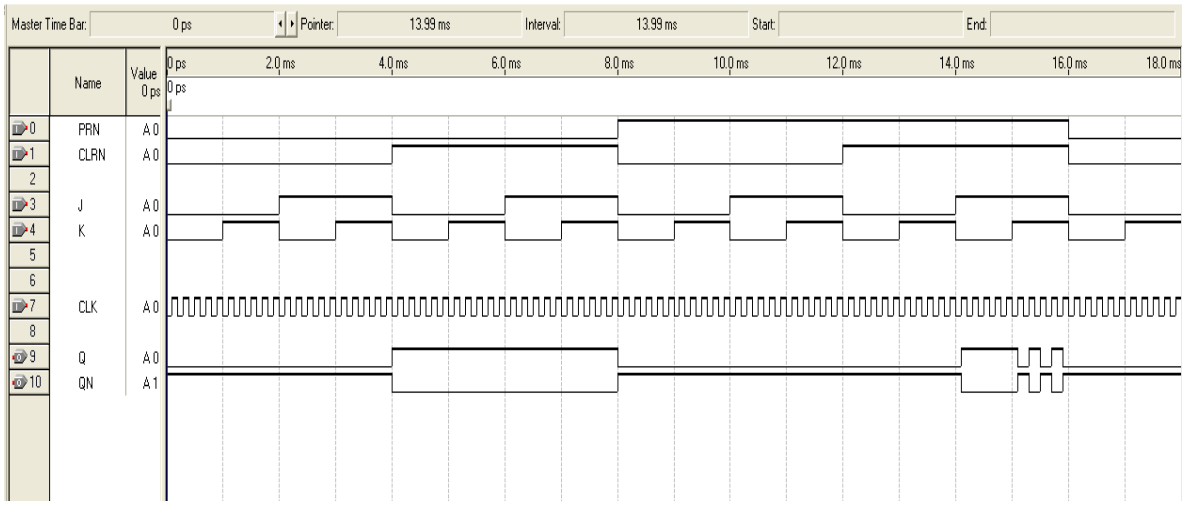


Figura 2 Resultados del biestable JK

Las frecuencias empleadas en la simulación son las siguientes:

- K → 500 Hz
- J → 250 Hz
- CLRN → 125 Hz
- PRN → 63 Hz
- CLK → 100 KHz

Como se puede apreciar en la Figura 2, el resultado es el correcto, pudiendo diferenciar en qué momento están actuando las entradas asíncronas (PRN y CLRN) y en qué momento están actuando las entradas síncronas disparadas por flanco de subida (J y K), viendo como las entradas asíncronas tienen prioridad sobre las síncronas.

En un primer momento, cuando ambas entradas asíncronas se encuentran a nivel bajo, no está claro qué pasa en la salida, ya que este estado produce una situación ilegal.

En la última zona, donde las señales asíncronas se encuentran a nivel alto, se observa un comportamiento anómalo: la salida está continuamente conmutando de un estado a otro (toggle). Esto se puede entender viendo la tabla de excitaciones del biestable JK:

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabla 1.

Como se puede apreciar, siempre que la entrada 'J' tenga un valor lógico de uno, la señal 'K' puede tomar indistintamente cualquier valor, y la salida actual toma un valor lógico de cero. De la misma manera, cuando la señal 'K' toma un valor lógico de uno la señal 'J' puede tomar indistintamente cualquier valor, y la salida actual toma un valor lógico de uno. Por tanto, se comprueba que al estar ambas entradas a un nivel lógico alto, la salida oscila entre cero y uno continuamente.

3.2.- CONTADOR SÍNCRONO ASCENDENTE DE MODULO 10.

El diseño a introducir en el Editor Gráfico es el siguiente:

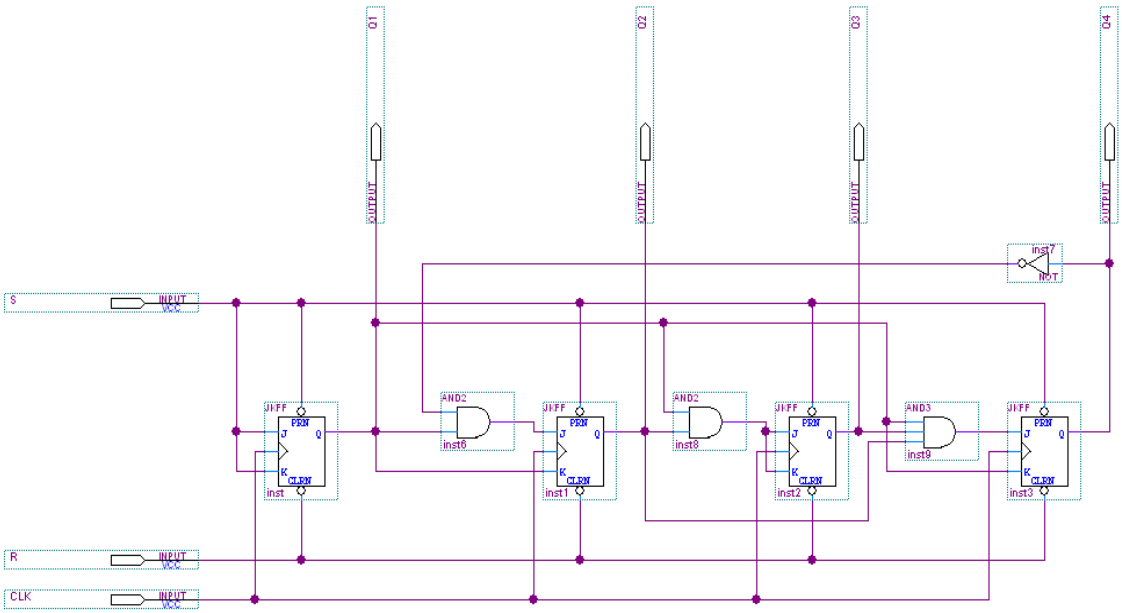


Figura 3 Circuito del contador ascendente síncrono módulo 10.

Observando el circuito, se tiene que la salida 'Q1' representa el bit menos significativo, mientras que la salida 'Q4' representa el bit más significativo.

El circuito realiza la cuenta ascendente de 0 a 9 siempre que las entradas asíncronas (S y R) se encuentren desactivadas (mediante un '1' lógico ya que trabajan a nivel bajo).

Tras realizar la simulación se obtiene el siguiente resultado:

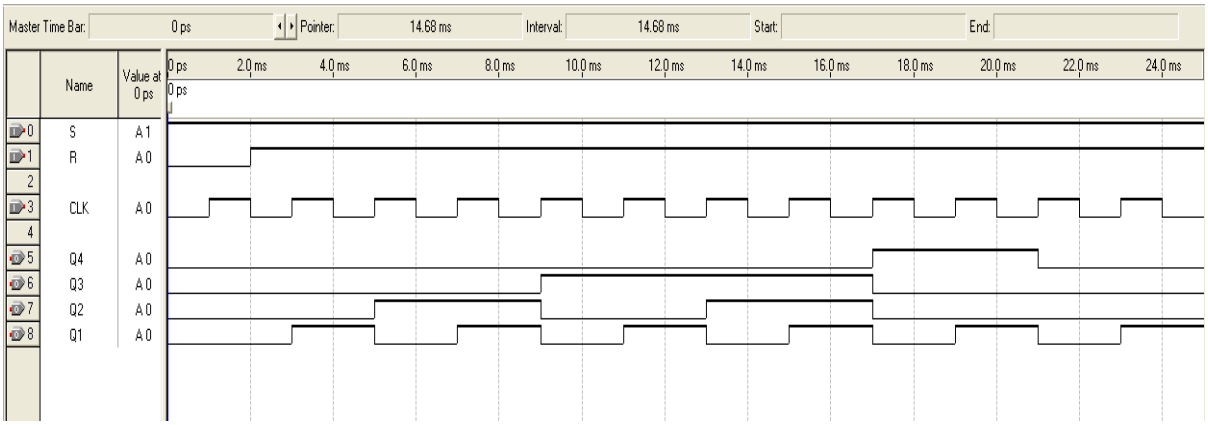


Figura 4. Resultado del contador síncrono ascendente de módulo 10

Como se puede comprobar en la Figura 4, el contador funciona correctamente, realizando la cuenta desde cero hasta nueve (incrementándose en los flancos de subida de la señal de reloj), para que una vez llegado a este punto comience de nuevo la cuenta desde cero.

Para realizar el volcado del diseño a la tarjeta educacional, es necesario introducir, en un nuevo proyecto, el diseño mostrado en la Figura 5, teniendo previamente que realizar el símbolo correspondiente al circuito del contador de módulo 10.

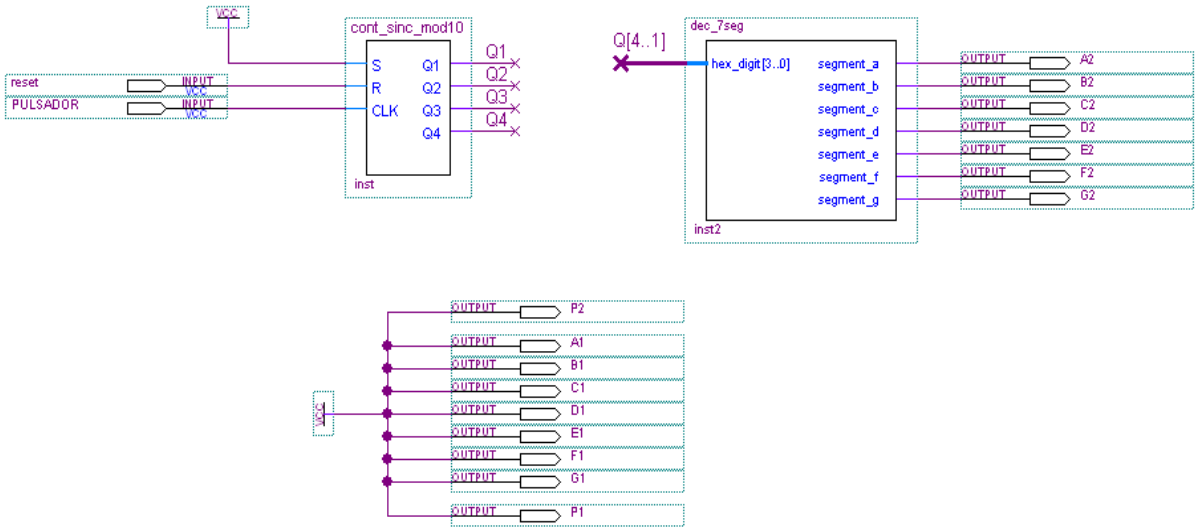


Figura 5. Contador de módulo 10 para volcar en la tarjeta educacional

Al realizar el volcado de este circuito a la tarjeta educacional UP2, el funcionamiento del diseño no es el correcto debido a la existencia de señales de rebote que interfieren en la señal original.

Si el volcado se realiza sobre la tarjeta educacional DE2, la influencia de las señales de rebote es mucho menor, la mayoría de veces no influye en el resultado y la cuenta se realiza correctamente, si bien no es fiable al cien por cien.

Debido a este motivo, de aquí en adelante, siempre que se haga el volcado de un diseño a cualquiera de las dos placas se incluirá el bloque 'Elimina-rebotes' diseñado en la práctica cuatro, tal y como se muestra en la siguiente figura:

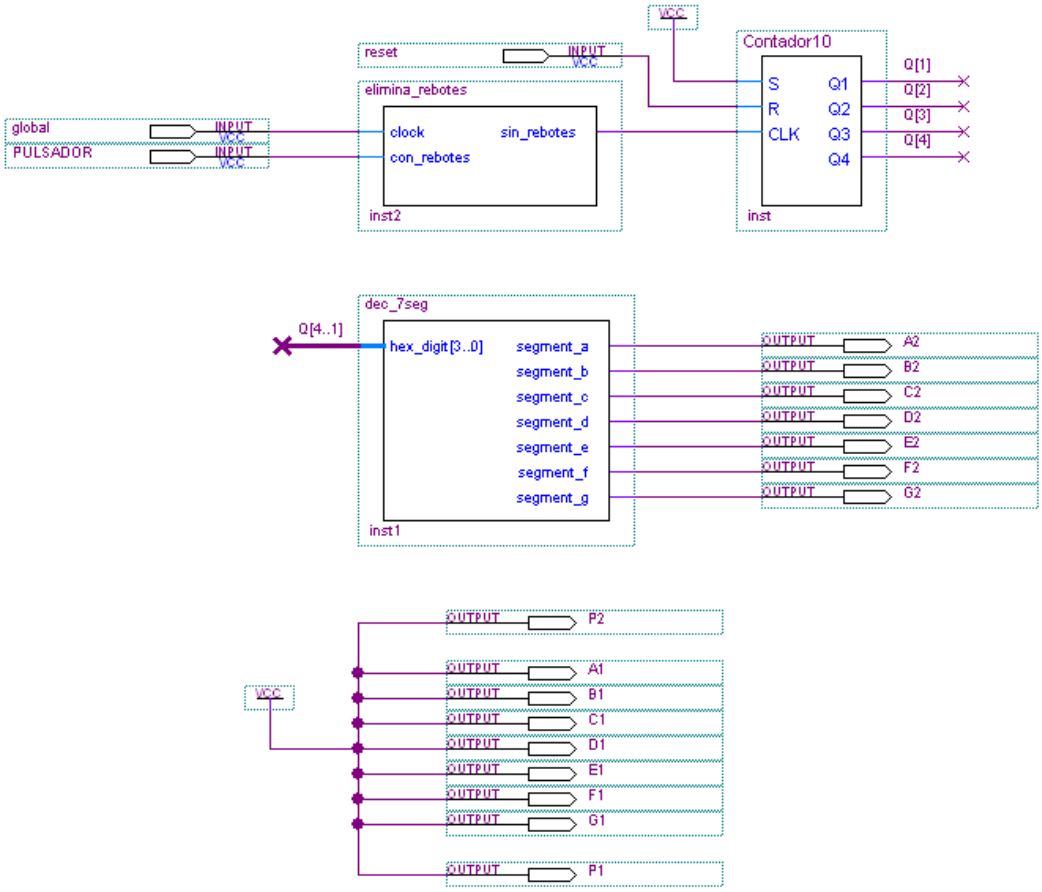


Figura 6. Contador módulo 10 con bloque elimina-rebotes para volcado en placa

La asignación de pines y programación del dispositivo se ha hecho siguiendo los pasos que aparecen en el guión de prácticas.

4.- EJERCICIOS DE DISEÑO

4.1.- Contador síncrono de módulo 5

4.1.1.- Contador síncrono ascendente de módulo 5:

Un contador síncrono ascendente de módulo 4 debe realizar la siguiente cuenta: 0, 1, 2, 3, 4, 0……

De esta manera el número máximo a representar es N = 4 (‘100’ en código binario), por lo que para la realización del circuito necesitamos una cantidad de tres biestables.

Se tiene que la tabla de excitaciones del biestable JK es la siguiente:

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabla 2.

Tomando como base la *Tabla 2* se construye la tabla que representa el comportamiento del contador.

Estado Actual Q(t)			Estado Siguiente Q(t+1)								
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X

Tabla 3.

Por medio del método de Karnaugh se obtienen las expresiones para las entradas de los distintos biestables, tomando como combinación de entrada el estado actual:

Función J_3:

Q_3/Q_2Q_1	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J3 = Q2 \cdot Q1$

Función K_3:

Q_3/Q_2Q_1	00	01	11	10
0	x	x	x	x
1	1	X	X	X

$K3 = 1$

Función J_2:

Q_3/Q_2Q_1	00	01	11	10
0	0	1	x	x
1	0	x	x	x

$J2 = Q1$

Función K_2:

Q_3/Q_2Q_1	00	01	11	10
0	x	x	1	0
1	x	x	x	x

$K2 = Q1$

Función J_1:

Q_3/Q_2Q_1	00	01	11	10
0	1	x	x	1
1	0	x	x	x

$J1 = \overline{Q3}$

Función K_1:

Q_3/Q_2Q_1	00	01	11	10
0	x	1	1	x
1	x	x	x	x

$K1 = 1$

En base a estas funciones, el circuito resultante queda de la siguiente manera:

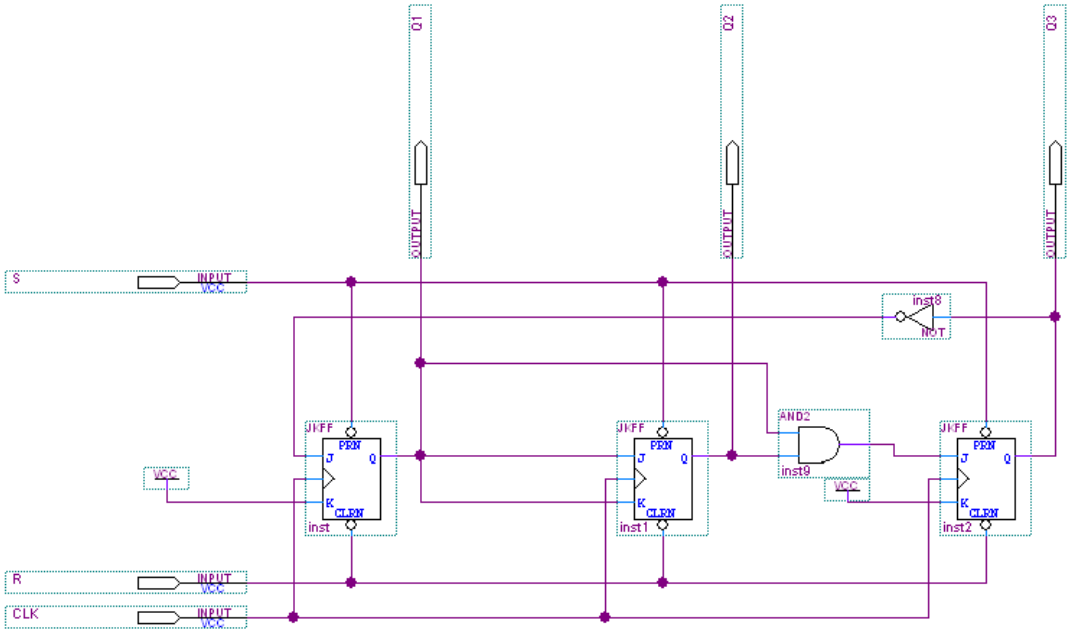


Figura 6. Contador síncrono ascendente de módulo 5

Realizando la simulación se obtiene el resultado mostrado en la Figura 7:

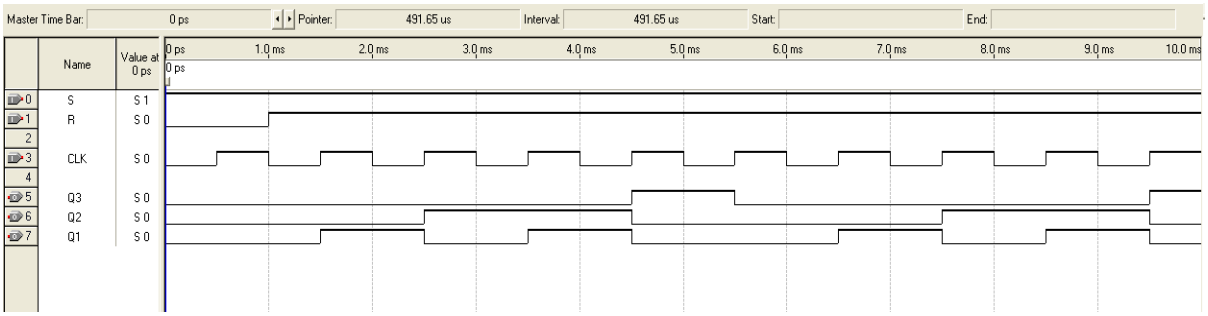


Figura 7. Resultado del contador ascendente de módulo 5

Se observa en la Figura 7 que el funcionamiento es el esperado, ya que el contador comienza a contar en el momento que las dos entradas asíncronas se encuentran desactivadas, realiza la cuenta desde cero hasta cuatro y vuelve a cero a continuación.

El circuito de volcado para el contador ascendente de módulo 5, una vez creado su correspondiente símbolo es el mostrado en la Figura 8.

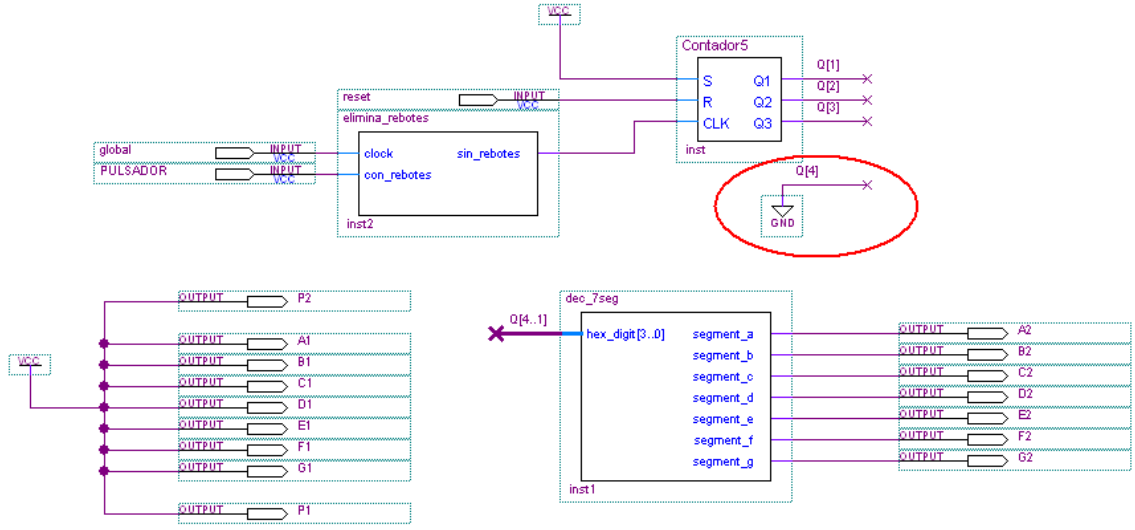


Figura 8. Circuito de volcado del contador ascendente de módulo 5

Un aspecto a tener en cuenta en el circuito de la Figura 8 es la parte redondeada en color rojo. El contador de módulo cinco consta de tres salidas, si bien a la entrada del decodificador BCD-7segmentos existe un bus que espera recibir cuatro señales. De ahí viene la conexión realizada en la parte redondeada en la que se incluye la cuarta señal que espera el decodificador, puesta a masa para que no afecte al resultado final.

4.1.2.- Contador síncrono reversible de módulo 5

La modificación de este circuito consiste en la inclusión de un pulsador que permita elegir el sentido de la cuenta. Por tanto, la forma de actuar del contador va a ser la siguiente:

X = 1 → 0, 1, 2, 3, 4, 0, ...
X = 0 → 0, 4, 3, 2, 1, 0, ...

Los pasos para el diseño del circuito son los mismos que los seguidos en el apartado anterior teniendo en cuenta la nueva variable.

Se construye la tabla que representa el comportamiento del circuito:

	Estado Actual Q(t)			Estado Siguiente Q(t+1)								
X	**Q3**	**Q2**	**Q1**	**Q3**	**Q2**	**Q1**	**J3**	**K3**	**J2**	**K2**	**J1**	**K1**
0	0	0	0	1	0	0	1	X	0	X	0	X
0	0	0	1	0	0	0	0	X	0	X	X	1
0	0	1	0	0	0	1	0	X	x	1	1	X
0	0	1	1	0	1	0	0	X	x	0	X	1
0	1	0	0	0	1	1	X	1	1	X	1	X
1	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
1	0	1	0	0	1	1	0	X	X	0	1	X
1	0	1	1	1	0	0	1	X	X	1	X	1
1	1	0	0	0	0	0	X	1	0	X	0	X

Tabla 4.

A través del método de las tablas de Karnaugh, se obtienen las expresiones para los diferentes biestables:

Función J_3:

xQ_3/Q_2Q_1	00	01	11	10
00	1	0	0	0
01	X	X	X	X
11	X	X	X	X
10	0	0	1	0

$$J3 = \overline{X} \cdot \overline{Q2} \cdot \overline{Q1} + X \cdot Q2 \cdot Q1$$

Función K_3:

xQ_3/Q_2Q_1	00	01	11	10
00	X	X	X	X
01	1	X	X	X
11	1	X	X	X
10	X	X	X	X

$$K3 = 1$$

Función J_2:

xQ_3/Q_2Q_1	00	01	11	10
00	0	0	X	X
01	1	X	X	X
11	0	X	X	X
10	0	1	X	X

$$J2 = \overline{X} \cdot Q3 + X \cdot Q1$$

Función K_2:

xQ_3/Q_2Q_1	00	01	11	10
00	X	X	0	1
01	X	X	X	X
11	X	X	X	X
10	X	X	1	0

$$K2 = X \cdot Q1 + \overline{X} \cdot \overline{Q1} = \overline{X \oplus Q1}$$

Función J_1:

xQ_3/Q_2Q_1	00	01	11	10
00	0	X	X	1
01	1	X	X	X
11	0	X	X	X
10	1	X	X	1

$$J1 = Q2 + \overline{X} \cdot Q3 + X \cdot \overline{Q3} = Q2 + X \oplus Q3$$

Función K_1:

xQ_3/Q_2Q_1	00	01	11	10
00	X	1	X	1
01	X	X	X	X
11	X	X	X	X
10	X	1	1	X

$$K1 = 1$$

De esta manera, el diseño del contador realizado en el Editor Gráfico es el mostrado en la Figura 9.

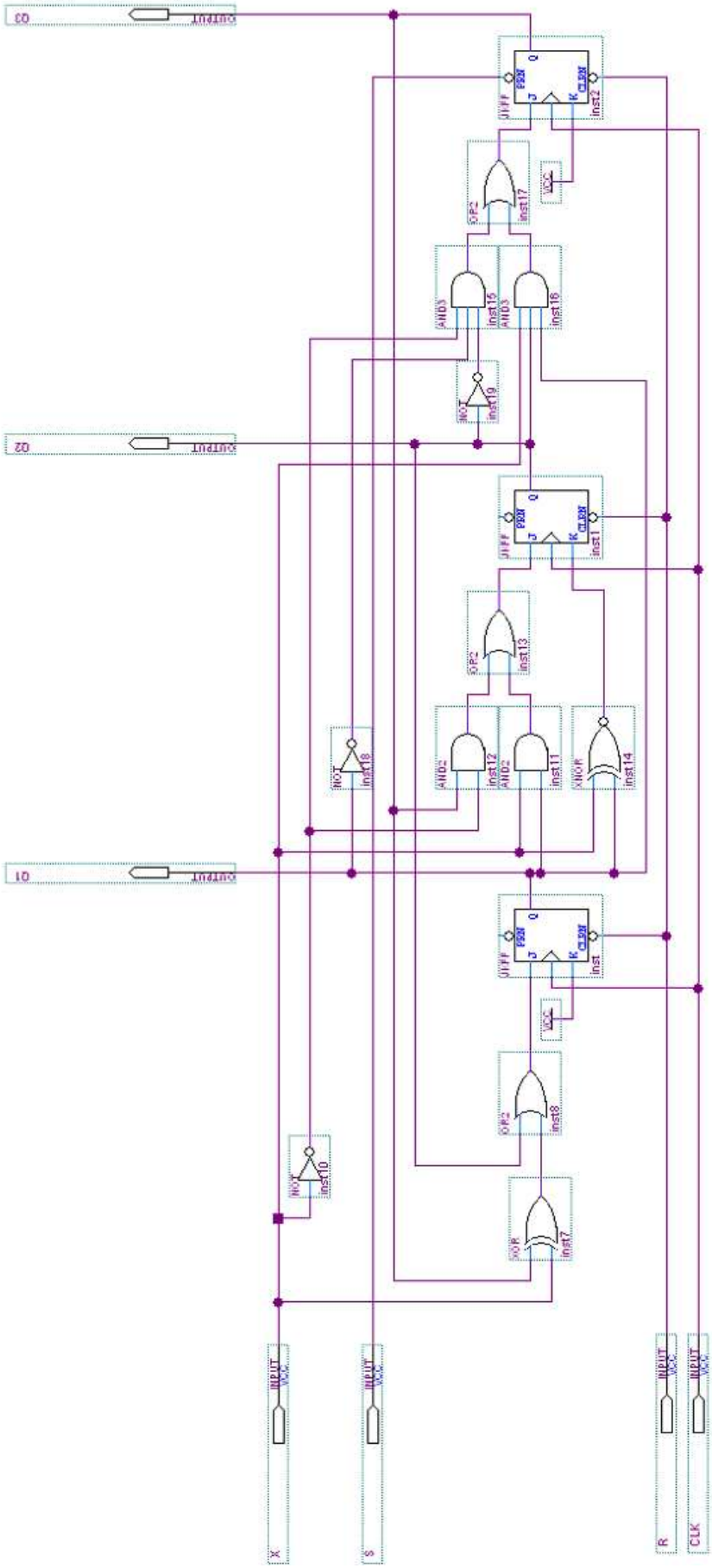


Figura 9. Contador reversible de módulo 5

La Figura 10 muestra el resultado de la simulación.

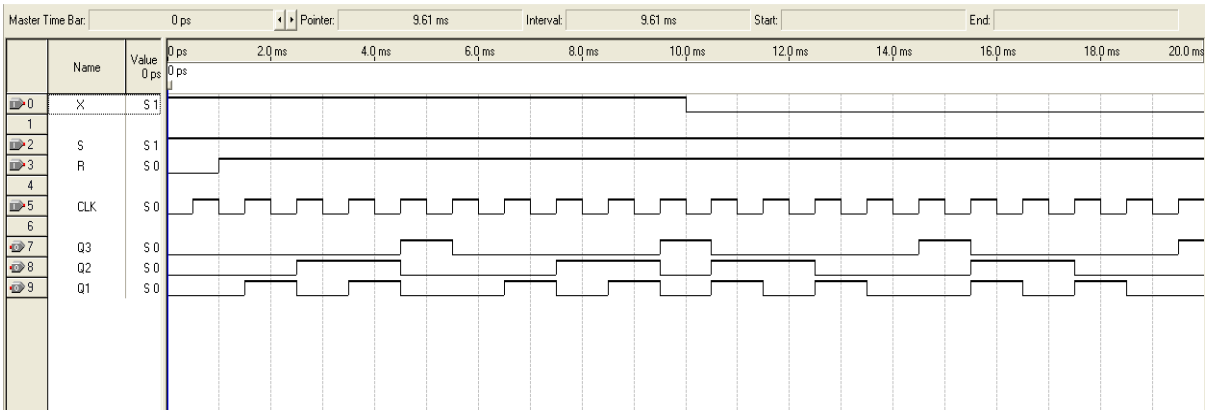


Figura 10. Resultado del contador síncrono reversible de módulo 5

En primer lugar, con la entrada 'X' a nivel alto, el contador realiza la cuenta en sentido ascendente para que, a continuación con 'X' a nivel bajo, el contador realice la cuenta en sentido descendente. Los valores de las salidas cambian según los flancos de subida de la señal de reloj, teniendo las entradas asíncronas desactivadas.

Los valores de las señales 'X', 'S' y 'R' son introducidos a mano, mientras que la señal de reloj tiene una frecuencia de 1 KHz.

El circuito de volcado a la tarjeta educacional, una vez creado el correspondiente símbolo es mostrado en la siguiente figura.

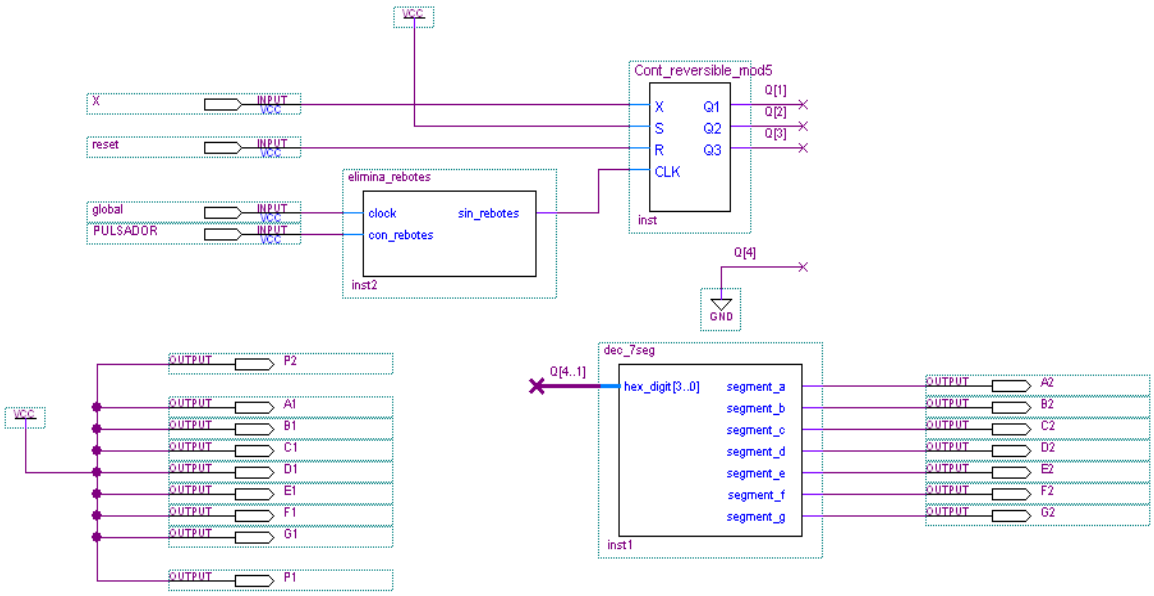


Figura 11. Contador síncrono reversible de módulo 5 para volcar

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

(1º Ingeniería Técnica en Informática de Gestión)

Práctica 6: Diseño y Volcado de un Cronómetro

1.- OBJETIVOS

a) Aprender y entender cómo a partir de la correcta combinación de contadores, se pueden conseguir aplicaciones como las que van a ser realizadas en esta práctica: Divisor de frecuencia y cronómetro.
b) El resultado a obtener en esta práctica es el diseño de un divisor de frecuencia que, a su salida, obtenga una señal de 1 Hz, partiendo de la señal de reloj de la tarjeta educacional DE2 o UP2, y emplear esta señal para diseñar un segundero de un reloj.

2.- MATERIAL

- Ordenador personal con el software QUARTUS II® de Altera®.
- Manual introductorio al software.
- Guión de prácticas.
- Tarjeta educacional DE2 ó UP2.

3.- DESARROLLO DE LA PRÁCTICA

3.1.- DIVISOR DE FRECUENCIA

Diseñar un divisor de frecuencia que permita obtener una señal de frecuencia 1 Hz a su salida, tomando como entrada la frecuencia de reloj de 50 MHz de la placa DE2[1].

Diseñar este divisor de frecuencia es lo mismo que diseñar un contador de módulo $50*10^6$. De esta manera, se deberá encontrar una combinación de contadores, los cuales conectados en cascada, formen el contador de módulo $50*10^6$. Los contadores a emplear en este diseño pueden ser encontrados en prácticas anteriores.

Debido al gran intervalo de tiempo necesario para simular este diseño y a la limitación de memoria que se dispone en los ordenadores, este diseño no se simulará. Para comprobar su correcto funcionamiento, se volcará directamente sobre la placa DE2. Para ello bastará con conectar a la entrada del diseño la señal global (terminal PIN_N2) y la salida a cualquiera de los Led de la placa, por ejemplo al denominado 'LEDR0' (PIN_AE23).

Se creará el símbolo correspondiente, para utilizarlo en el siguiente apartado.

[1] *En el caso de tener que utilizar la placa UP2, el procedimiento a seguir será el mismo, teniendo en cuenta que la señal de reloj de la placa es de 25,175 MHz ≈ 25 MHz, por tanto, el divisor de frecuencia se corresponde con un contador de módulo $25*10^6$.*
En este caso, la entrada global se corresponde con el terminal 91 y la salida a cualquiera de los Led de la placa.

- ***Nota:***

Tanto en este ejercicio como en el siguiente se trabaja con la señal de reloj de la tarjeta educacional DE2 de frecuencia 50 MHz. Tras realizar el diseño correspondiente en el Editor Gráfico, antes de realizar el proceso de compilación es necesario seguir las indicaciones que se exponen a continuación para que los resultados sean los esperados:

1º- Acudir al menú 'Assignments' situado en la barra de herramientas principal y seleccionar la opción 'Timing Analysis Settings…', obteniendo la ventana que se muestra en la Figura 1.

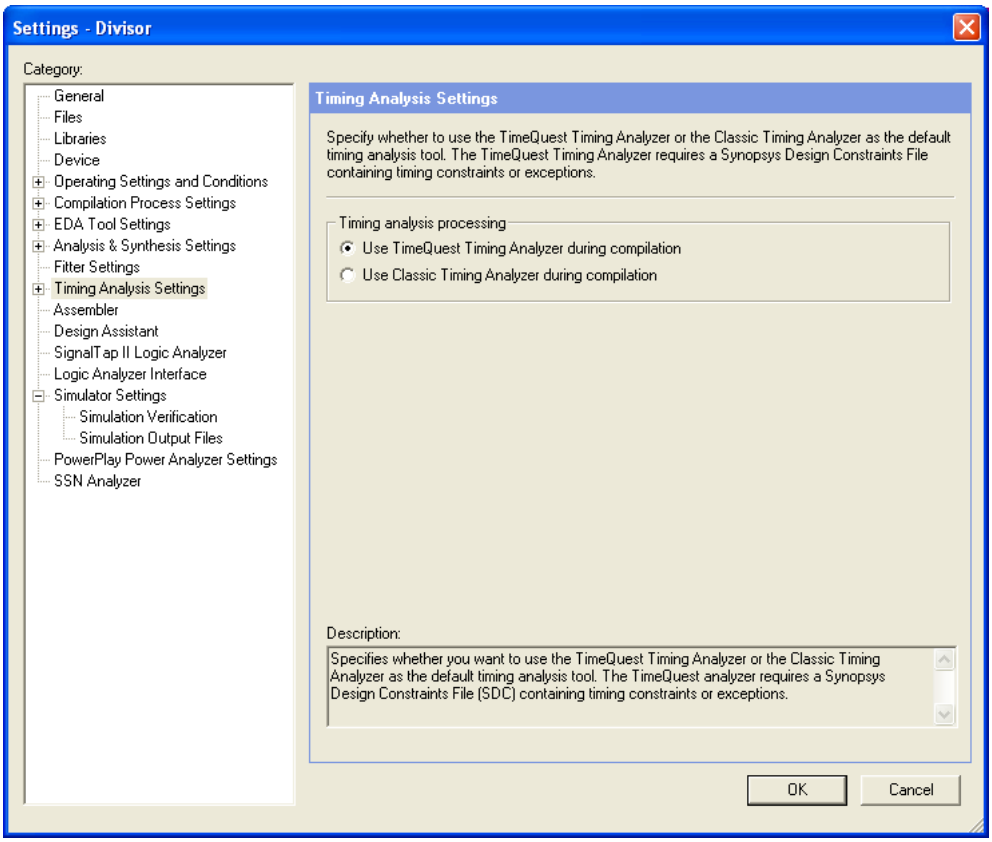


Figura 1. Timing Analysis Settings

2º- Desde esta ventana seleccionar la opción 'Use TimeQuest Timing Analyzer during compilation', ya que por defecto no viene seleccionada.

3º- Realizar los procesos de compilación y programación de la misma manera que en la práctica anterior.

3.2.- SEGUNDERO

Diseñar el segundero de un reloj digital. Para ello se utilizarán diseños ya realizados como el contador de módulo diez, divisor de frecuencia (1 Hz), decodificador BCD-7segmentos, etc. Este segundero deberá contar desde '0' hasta '59', dando la salida en la pareja de displays HEX6 y HEX7, situados más a la izquierda de la placa. Además, el diseño debe incluir un pulsador de puesta a cero del segundero.

De la misma forma que ocurría en diseños anteriores, existen varias soluciones posibles y al mismo tiempo válidas.

Por el mismo motivo del apartado anterior, este diseño se volcará directamente sobre la tarjeta educacional, siguiendo los pasos explicados en la anterior práctica y comprobando su correcto funcionamiento.

La distribución de pines a emplear es la mostrada en el guión de la práctica cinco.

RESOLUCIÓN PRÁCTICA 6

3.1.- DIVISOR DE FRECUENCIA

El divisor de frecuencia a diseñar permite obtener una frecuencia de 1 Hz, partiendo de la frecuencia de reloj de la placa.

En el caso de la placa DE2 existen dos señales de reloj que pueden ser utilizadas, a frecuencias de 27 MHz y 50 MHz.

En el caso de la tarjeta UP2 la frecuencia de la señal de reloj es de 25,175 MHz ≈ 25MHz.

- ***Divisor de frecuencia para tarjeta educacional DE2:***

Con el fin de poder aprovechar los diseños de contadores ya realizados, en concreto los contadores de módulo cinco y módulo 10, se toma como entrada al divisor de frecuencia la señal de reloj con frecuencia de 50 MHz. De esta manera, diseñar este divisor de frecuencia es lo mismo que diseñar un contador de módulo $50*10^6$.

Realizando la siguiente operación se ve el número y tipo de contadores que es necesario emplear:

$$\frac{50\cdot 10^6\ \text{Hz}}{5\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10} = 1\,\text{Hz}$$

En base a esta operación, se necesita un contador de módulo cinco y siete contadores de módulo diez, conectados en cascada, es decir, la salida de mayor peso de cada contador se conecta a la entrada de reloj del contador que se encuentra inmediatamente después, tal y como se muestra en la siguiente figura.

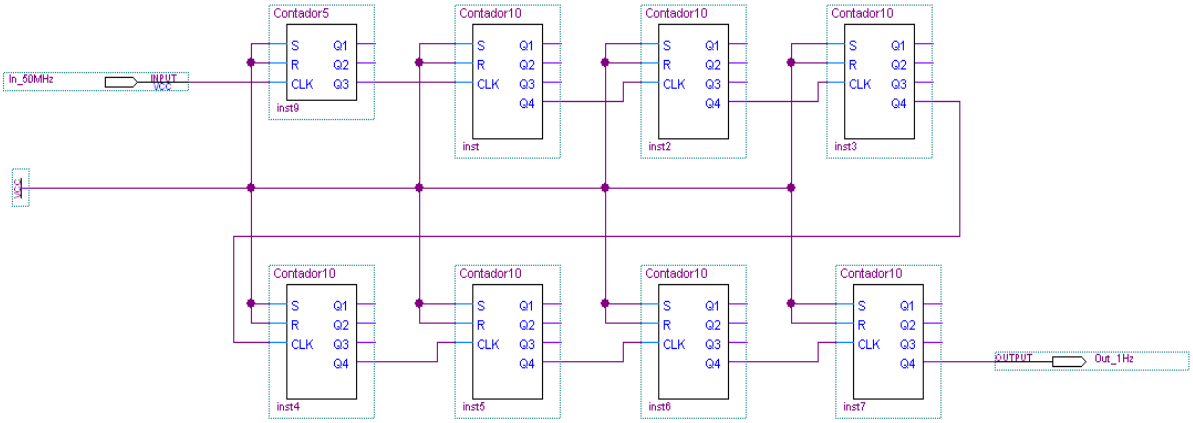


Figura 1. Divisor de frecuencia 50 MHz → 1Hz

En este diseño, observando la Figura 1, se ve como las entradas asíncronas de los contadores (S y R) quedan desactivadas al ser conectadas a nivel alto ya que, internamente, los contadores empleados trabajan a nivel bajo con estas entradas.

Este diseño no es simulado debido al gran intervalo de tiempo que requiere su simulación. Además se trabaja con símbolos que ya han sido simulados en la práctica anterior

por lo que no se plantea ningún problema para realizar directamente el volcado a la tarjeta educacional.

La asignación pines consiste en dar a la entrada de 50 MHz el pin 'PIN_N2' y utilizar cualquiera de los Led de la placa para ver la salida de 1 Hz, por ejemplo, escogiendo el pin 'PIN_AE23' será el Led de color rojo situado más cerca de los pulsadores el que funcione.

Por último se crea el símbolo correspondiente, ya que va a ser usado en el posterior apartado.

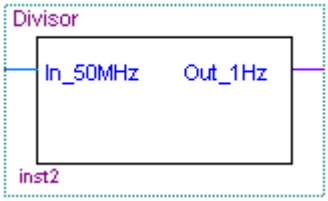


Figura 2. Símbolo divisor de frecuencia 50 MHz → 1Hz

- ***Divisor de frecuencia para tarjeta educacional UP2:***

Tal como se comenta anteriormente la placa UP2 presenta una señal de reloj de prácticamente 25 MHz. Por tanto, este divisor de frecuencia equivale a diseñar un contador de módulo $25*10^6$. Para realizar esto, una operación como la realizada en el apartado anterior servirá para determinar el número y tipo de contadores a utilizar:

$$\frac{25\cdot 10^6\ \text{Hz}}{5\cdot 5\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10} = 1\text{Hz}$$

Así, se necesitarán dos contadores de módulo cinco y seis contadores de módulo diez, conectados en cascada. El diseño es realizado utilizando los contadores de módulo cinco y módulo 10 implementados en la práctica anterior, tal como muestra la Figura 3.

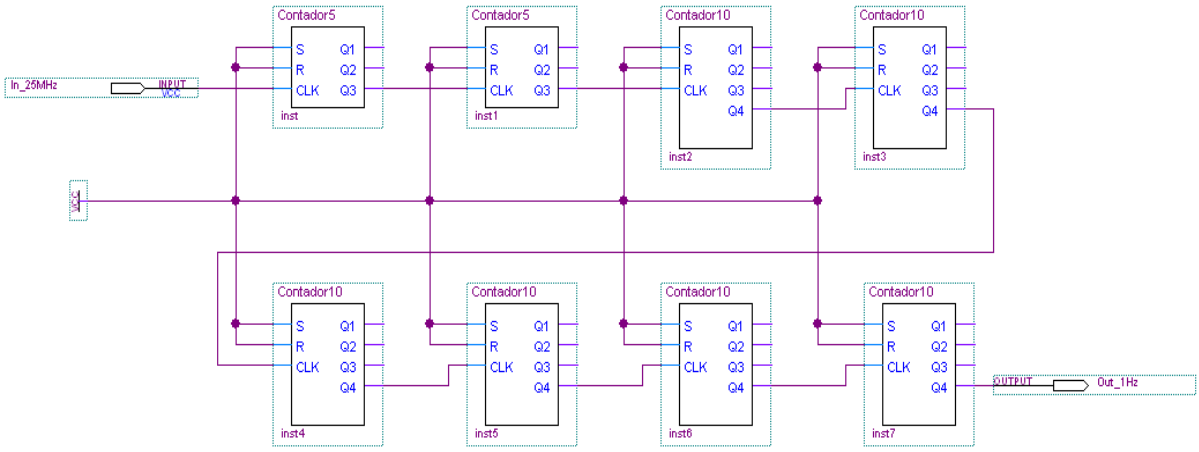


Figura 3. Divisor de frecuencia 25 MHz → 1 Hz

De la misma manera que en el anterior diseño, las señales asíncronas (S y R) son conectadas a 'vcc' para que permanezcan desactivadas.

Para este diseño no se realiza su simulación debido al gran intervalo de tiempo que conlleva, además de que se parte de símbolos ya simulados y probados.

En este caso, la señal de entrada queda conectada al terminal 91 y la salida a cualquiera de los Led que se encuentran en la placa.

A continuación se muestra el símbolo correspondiente, que será utilizado para realizar el segundero.

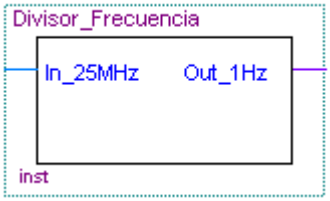


Figura 4. Símbolo del divisor de frecuencia 25 MHz → 1 Hz

3.2.- SEGUNDERO

El diseño a realizar debe ser capaz de imitar al segundero de un reloj, es decir, contar desde '0' hasta '59' y reiniciarse una vez llegado a este punto. Su salida será mostrada por los displays de la placa. Además, se incluye un pulsador que permite poner a cero el contador en cualquier momento.

En este guión se proponen dos soluciones diferentes, además de tener en cuenta la tarjeta educacional en la que va a ser volcado el diseño.

- **Tarjeta educacional DE2:**

La base para realizar el diseño del segundero es emplear el símbolo correspondiente al divisor de frecuencia realizado en el ejercicio anterior, el cual adapta la señal de reloj de 50 MHz procedente de la placa DE2 a una señal con frecuencia de 1 Hz. Una vez obtenida esta señal, a continuación se exponen dos posibles soluciones:

- **1ª Solución: Contador módulo 10 + Contador módulo 6**

La solución más intuitiva es la de emplear un contador de módulo 10 para la parte de las unidades, que cuente desde cero hasta nueve, y emplear un contador de módulo 6 para las decenas, es decir, realizar una cuenta desde cero hasta cinco.

El contador de módulo 10 a emplear será el diseñado en la práctica anterior, por tanto, en primer lugar se debe diseñar el contador de módulo 6 siguiendo los pasos que se presentan a continuación.

- *Contador de módulo 6:*

Un contador de módulo 6 realiza la cuenta desde 0 hasta 5 (101), por lo que son necesarios tres biestables JK para implementarlo.

La tabla que explica el funcionamiento del contador es la siguiente:

Estado actual Q(t)			Estado siguiente Q(t+1)								
Q3	Q2	Q1	Q3	Q2	Q1	J3	K3	J2	K2	J1	K1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	0	X	1	0	X	X	1

Tabla 1.

Mediante el método de las tablas de Karnaugh se obtienen las expresiones correspondientes a las entradas de cada biestable.

- Función J3:

Q_3/Q_2Q_1	00	01	11	10
0	0	0	1	0
1	x	x	x	x

$J3 = Q2 \cdot Q1$

- Función K3:

Q_3/Q_2Q_1	00	01	11	10
0	0	0	1	0
1	x	x	x	x

$K3 = Q1$

- Función J2:

Q_3/Q_2Q_1	00	01	11	10
0	X	X	1	0
1	X	X	X	X

$K2 = Q1$

- Función K2:

Q_3/Q_2Q_1	00	01	11	10
0	0	1	X	X
1	0	0	X	X

$J2 = \overline{Q3} \cdot Q1$

- Función J1:

Q_3/Q_2Q_1	00	01	11	10
0	X	1	1	X
1	X	1	X	X

$K1 = 1$

- Función K1:

Q_3/Q_2Q_1	00	01	11	10
0	1	X	X	1
1	1	X	X	X

$J1 = 1$

En base a estas funciones, el circuito resultante es el mostrado en la Figura 5.

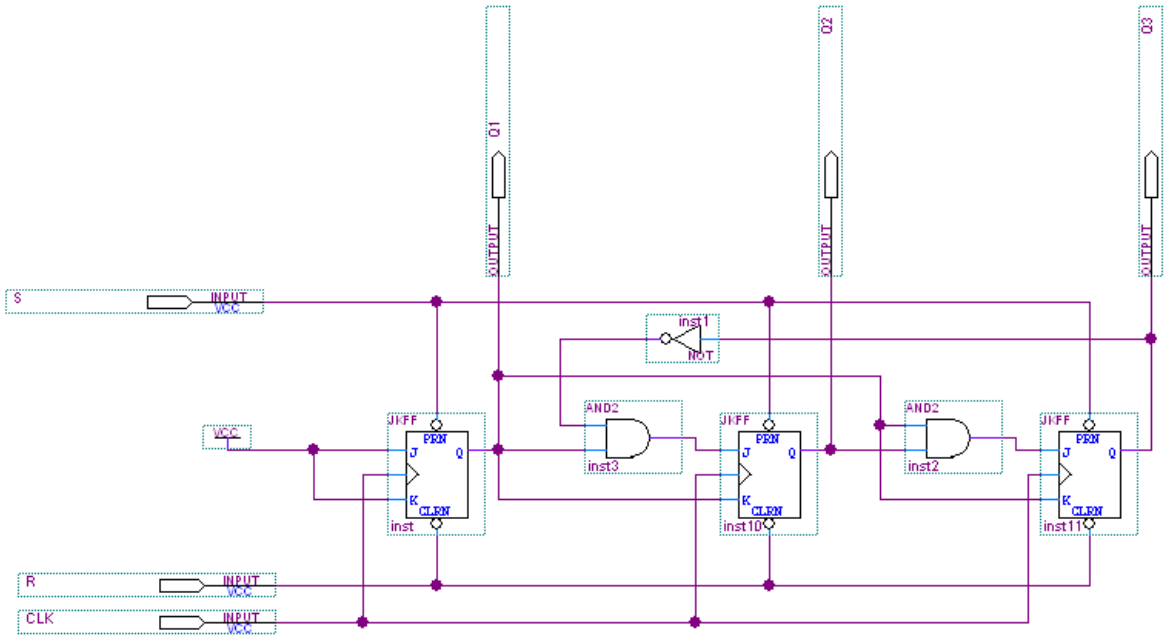


Figura 5. Contador de módulo 6

Realizando el correspondiente símbolo para el contador de módulo 6 y tomando los símbolos del contador de módulo 10 y del decodificador BCD-7 segmentos realizados en la práctica anterior, se realiza el diseño del segundero mostrado en la Figura 6:

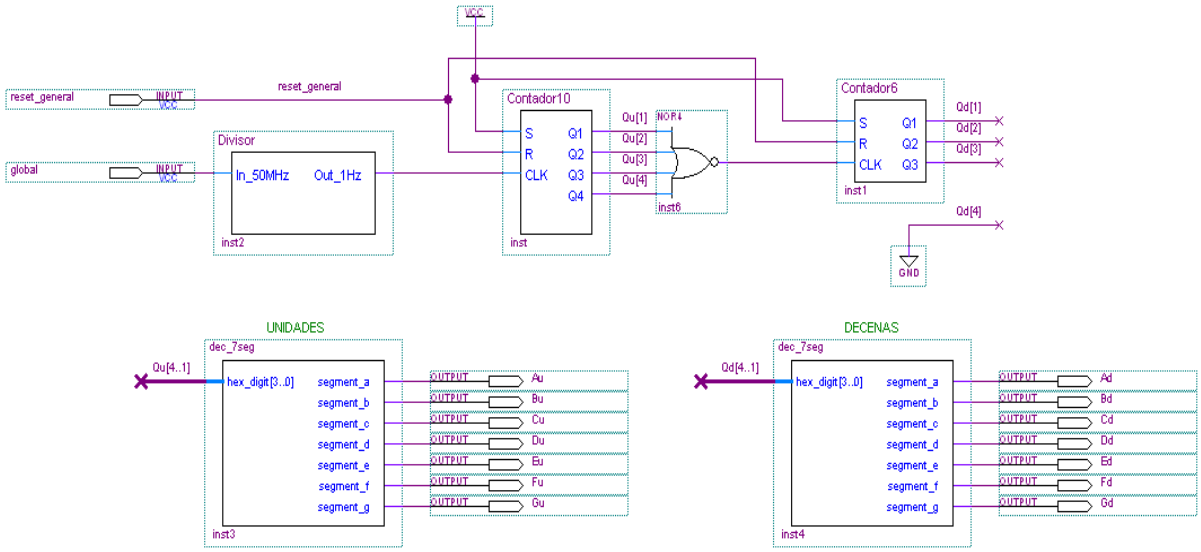


Figura 6. Circuito segundero

El contador de módulo 10 tiene como señal de reloj una señal de frecuencia 1Hz procedente del divisor de frecuencia. Este contador será el que indique las unidades y sus salidas servirán para controlar el funcionamiento del display asignado para las mismas a través de un decodificador BCD-7 segmentos.

El contador de módulo 6 será el que indique las decenas, por lo que deberá incrementarse cada vez que el anterior contador pase por cero. Para ello se emplea una puerta NOR4 cuyas entradas son las cuatro salidas del contador de módulo 10, por lo que cuando este contador pase por cero, todas sus salidas contarán con un '0' lógico y en la salida de la puerta NOR4 obtendremos un '1' lógico. Las salidas de este segundo contador irán a parar a otro decodificador BCD-7 segmentos, el cual controlará el display reservado para las decenas.

A su vez, existe una entrada para un pulsador (reset_general) que sirve para poner a cero la cuenta de ambos contadores al mismo tiempo.

Como se puede comprobar, esta solución para el diseño resulta bastante trabajosa ya que se utilizan otros diseños no realizados en prácticas anteriores, donde habrá que asegurarse que funcionan correctamente. Sin embargo, en la segunda solución mostrada a continuación, es posible aprovecharnos de diseños ya implementados y probados, por lo que conseguiremos ahorrar tiempo y asegurarnos de su funcionamiento.

- **2ª Solución: Contador módulo 10 + Contador módulo 10.**

La Figura 7 muestra el resultado del diseño obtenido, el cual se explica posteriormente.

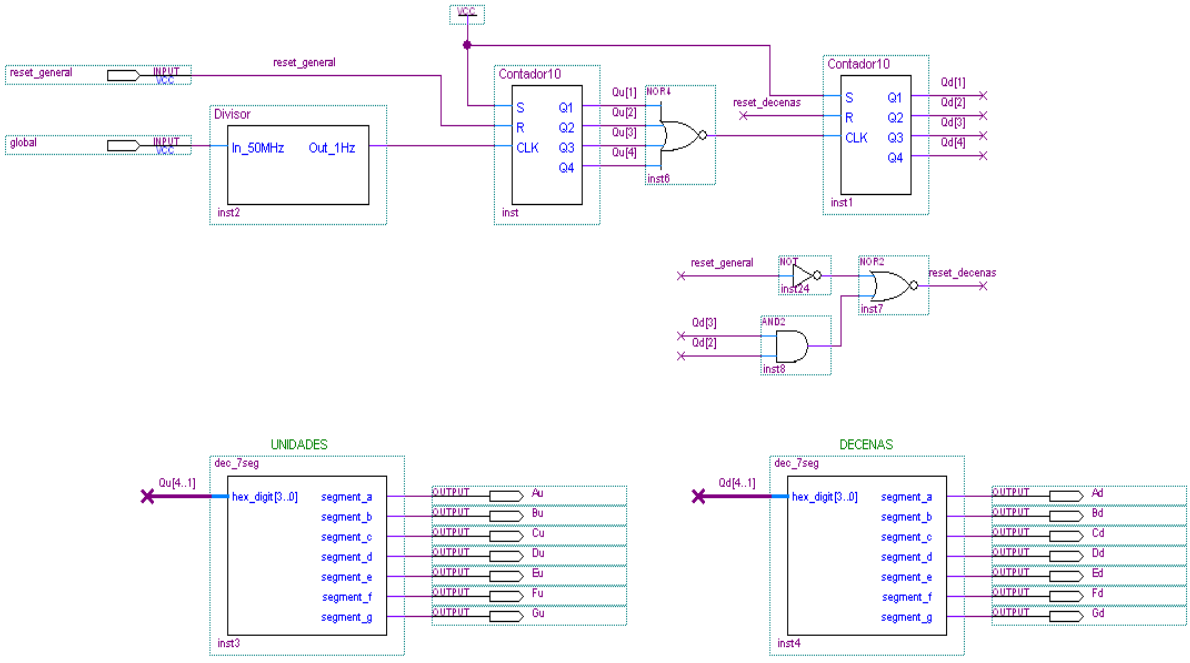


Figura 7. Circuito Segundero

En la Figura 7 se aprecia como, a diferencia del circuito anterior, se emplean dos contadores de módulo diez.

El primer contador tiene como entrada para la señal de reloj una señal de frecuencia 1 Hz procedente del divisor de frecuencia. Este contador servirá para indicar las unidades y sus salidas controlarán el display asignado para ello a través de un decodificador BCD-7segmentos.

El segundo contador será el que indique las decenas. Por tanto, éste deberá incrementarse cada vez que el primero pase por cero. Para ello se emplea una puerta NOR4 cuyas entradas son las cuatro salidas procedentes del contador de las unidades, por lo que cuando el primer contador pase por cero, todas sus salidas estarán a nivel bajo y la salida de dicha puerta NOR4 será de un '1' lógico. Las salidas de este contador servirán como entradas al decodificador BCD-7segmentos asignado para controlar el display de las decenas.

Tras este proceso se han conseguido las señales de reloj que excitan a ambos contadores pero todavía debemos tener en cuenta que el segundo contador sólo debe contar desde cero hasta cinco, por lo que habrá que realizar la pertinente modificación. Para ello basta con estudiar las salidas de este contador hasta que el mismo alcance el número seis. En este momento sus salidas presentarán los siguientes valores:

Qd[1] → '0'
Qd[2] → '1'
Qd[3] → '1'
Qd[4] → '0'

De esta manera, basta con unir las señales Qd[2] y Qd[3] mediante una puerta NAND de dos entradas y conectar su salida resultante a la entrada que resetea este segundo contador denominada 'reset_decenas', ya que esta entrada se activa a nivel bajo. Como también interesa poner a cero el contador cada vez que se activa el pulsador que permite poner a cero el segundero, es necesaria una segunda modificación, la cual se puede ver en la siguiente tabla de la verdad:

reset_general (pulsador)	Qd[2]	Qd[3]	reset_decenas
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Tabla 2.

Hay que tener en cuenta que el pulsador de la placa DE2, dará un '0' lógico al ser pulsado, y que los reset de los correspondientes contadores también se activan a nivel bajo.

En base a la Tabla 2 se emplea el método de Karnaugh:

reset/Qd[2]Qd[3]	00	01	11	10
0	0	0	0	0
1	1	1	0	1

La función resultante es la siguiente:

$$\overline{reset\_decenas} = \overline{reset\_general} + Qd[3] \cdot Qd[2]$$

$$\downarrow$$

$$reset\_decenas = \overline{\overline{reset\_general} + Qd[3] \cdot Qd[2]}$$

Por tanto, la forma de resetear el contador de las decenas se basa en el siguiente circuito combinacional:

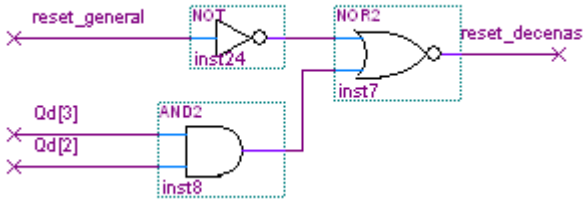


Figura 8. Reset decenas

Para resetear el contador de las unidades basta con conectar directamente su entrada de reset con la entrada de reset general procedente del pulsador de puesta a cero del segundero.

Para poner a cero el segundero se emplea un pulsador de la placa, el cual generará tantas señales de reset como rebotes se produzcan. Se podría haber incluido en el diseño el bloque elimina-rebotes visto en la práctica anterior, si bien no tiene mucho sentido ya que el efecto de varios reset es el mismo que si se realizara uno sólo, salvo con un mínimo retraso.

- **Tarjeta Educacional UP2**

El procedimiento a seguir para realizar el diseño mediante esta tarjeta educacional puede ser cualquiera de los dos vistos en el caso de la placa DE2, simplemente cambiando el divisor de frecuencia, que en este caso nos debe adaptar la señal de 25 MHz a 1 Hz. De esta manera, tomando la segunda solución vista anteriormente por considerarse más eficiente, el circuito resultante es el mostrado en la Figura 9.

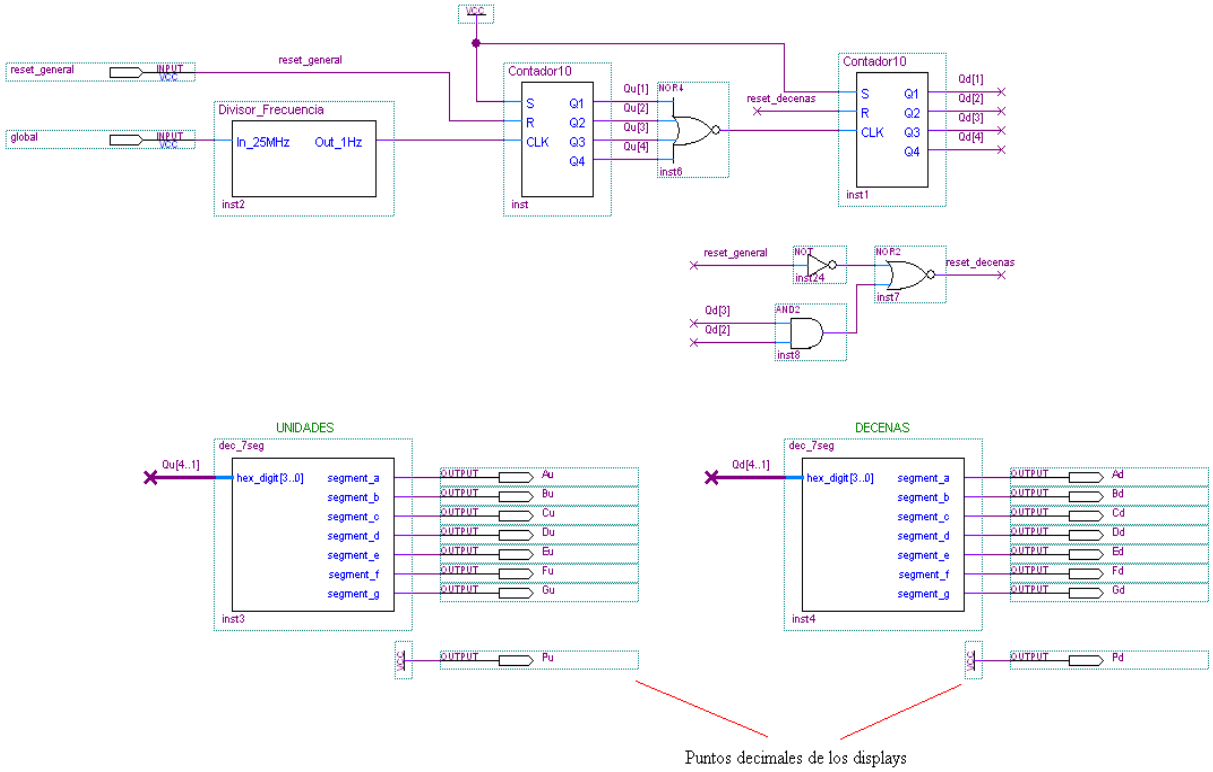


Figura 9. Circuito Segundero

Observando la figura, se aprecia como también son incluidas las salidas correspondientes a los diodos Led que representan los puntos decimales en los displays, los cuales son conectados a una entrada 'vcc' para que queden apagados, ya que funcionan a nivel bajo.

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

(1º Ingeniería Técnica en Informática de Gestión)

Práctica 7: Máquina de Estados

Se denomina ***máquina de estados*** a un modelo de comportamiento de un sistema con entradas y salidas, en donde las salidas dependen no sólo de las señales de entradas actuales sino también de las anteriores.

1.- OBJETIVOS

a) Entender el funcionamiento de circuitos secuenciales síncronos a través de los sistemas de la 'Máquina de Mealy' y 'Máquina de Moore'.
b) Profundizar más en el lenguaje de programación VHDL mediante la realización de dos diseños de máquinas de estados.

2.- MATERIAL

- Ordenador personal con el software QUARTUS II® de Altera®.
- Manual introductorio al software.
- Guión de prácticas.
- Tarjeta educacional DE2 ó UP2.

3.- DESARROLLO DE LA PRÁCTICA

En este apartado se va a realizar un diseño guiado, basado en la Máquina de Mealy con el fin de comprender su funcionamiento, y que sirva como base para la posterior realización del ejercicio de diseño que se presenta en el siguiente apartado del guión.

3.1.- CERRADURA DIGITAL

Se plantea realizar una cerradura digital cuyo funcionamiento, tanto para la apertura como para el cierre, se basa en una combinación de ceros y unos. La cerradura tiene las siguientes características:

- La combinación para abrir la cerradura es '0-1-1', donde el uno sería el primer valor de la secuencia que entra y el cero el último.
- Si la cadena introducida es la correcta, se obtendrá una señal de salida que provocará la apertura de la cerradura.
- Una vez que la cerradura está abierta, el introducir '0-0', hace que la cerradura se cierre. En este momento, el control se resetea a su estado inicial.

Posteriormente, en el presente guión, se muestra el código, correspondiente a una máquina de Mealy, que representa la cerradura digital, si bien, en primer lugar se desarrollan los pasos que nos llevan a ese código. Por tanto, la resolución del problema es la siguiente:

Tal y como dice el enunciado, la secuencia correcta para conseguir abrir la cerradura es '0-1-1', a continuación, estando la puerta abierta, para cerrarla hacen falta dos ceros consecutivos.

Denominando 'z' a la señal de salida, se considera que:

z = 0 → cerradura cerrada.
z = 1 → cerradura abierta.

Los posibles estados del sistema son los siguientes:

q0: Estado inicial. No ha llegado ningún uno.
q1: Introducción de '1'.
q2: Llega la secuencia '1-1'.
q3: Llega la secuencia '0-1-1'.
q4: Llega un '0' estando la cerradura abierta.

El diagrama de estados se presenta en la siguiente figura:

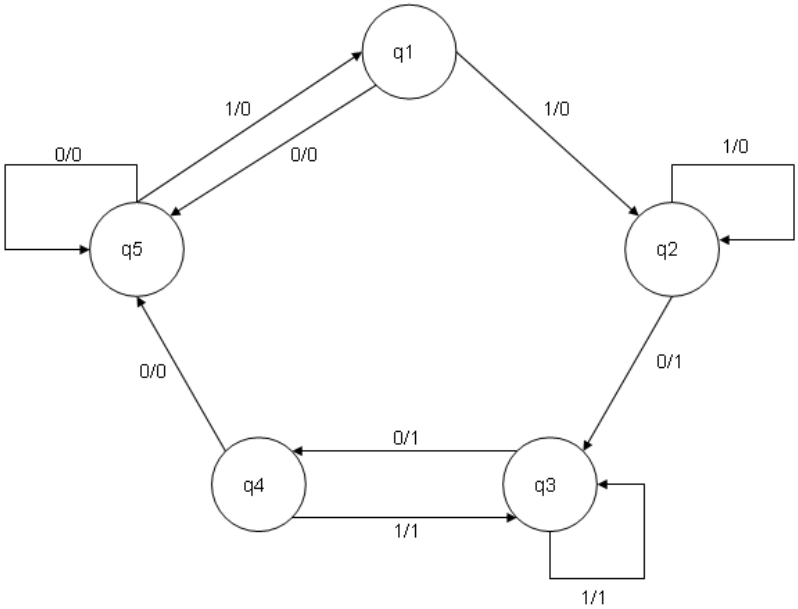


Figura 1. Diagrama de estados.

A continuación se representa la tabla de transiciones que muestra el comportamiento de los estados, en la que se obtiene el estado siguiente y la salida, dependiendo del estado actual y de la entrada.

Estado Actual Q(t)	Entrada X	Estado Siguiente Q(t+1)	Salida Z
q0	0	q0	0
	1	q1	0
q1	0	q0	0
	1	q2	0
q2	0	q3	1
	1	q2	0
q3	0	q4	1
	1	q3	1
q4	0	q0	0
	1	q3	1

Tabla 1.

Una vez visto el comportamiento del sistema según las especificaciones mencionadas en el enunciado, se pasa a traducir esto a código VHDL, quedando el siguiente programa:

```
-- CERRADURA DIGITAL

entity cerradura is port(
        x,clock : in bit;
        z : out bit
        );
end cerradura;

architecture arch_cerradura of cerradura is
  type tipo_estado is (q0,q1,q2,q3,q4);
  signal estado_actual,estado_siguiente:tipo_estado;


begin
  process (estado_actual,x)
    begin
            case estado_actual is
        when q0 =>
                if x='0' then
                  z<='0';
                  estado_siguiente<=q0;
                else
                  z<='0';
                  estado_siguiente<=q1;
                end if;
        when q1 =>
                if x='0' then
                  z<='0';
                  estado_siguiente<=q0;
                else
                  z<='0';
                  estado_siguiente<=q2;
                end if;
        when q2 =>
                if x='0' then
                  z<='1';
                  estado_siguiente<=q3;
                else
                  z<='0';
                  estado_siguiente<=q2;
                end if;
        when q3 =>
                if x='0' then
                  z<='1';
                  estado_siguiente<=q4;
                else
                  z<='1';
                  estado_siguiente<=q3;
                end if;
        when q4 =>
```

```
                if x='0' then
                  z<='0';
                  estado_siguiente<=q0;
                else
                  z<='1';
                  estado_siguiente<=q3;
                end if;
        end case;
end process;



process
   begin
        wait until clock' event and clock='1';
        estado_actual<=estado_siguiente;
 end process;
end arch_cerradura;
```

- *Ejercicio:*

Entender el contenido del código y realizar la simulación del mismo para comprobar su funcionamiento, el cual se basa en ver que pasa de un estado a otro correctamente y que la salida es la esperada en cada caso. Como simular todos los cambios de estado resulta bastante engorroso, bastará con simular los que se consideren más importantes y sean más representativos del diseño.

Es necesario tener en cuenta, para entender adecuadamente el comportamiento del diseño, que el cambio de estado se produce coincidiendo con un flanco de subida de la señal de reloj, mientras que la salida cambia instantáneamente al cambiar el estado de la entrada, si es que esto se produce. Por tanto, debemos tener en cuenta que dichas señales no tienen porqué cambiar a la vez.

El cambio de estado no se producirá en el momento de cambio de valor en la señal de entrada, sino que lo hará aproximadamente un periodo de la señal de reloj más tarde.

Para visualizar el resultado correctamente se puede introducir un valor de 10 ms como tiempo máximo y una frecuencia para la señal de reloj de 1 KHz (se pueden coger otros valores siempre que la visualización permita entender el comportamiento del diseño). Además, se designarán los valores de la señal de entrada a mano, eligiendo los cambios de valor de la misma justo en los flancos de subida de la señal de reloj, de forma se vayan obteniendo los distintos estados que interese estudiar, junto con su correspondiente salida. Para realizar esto correctamente, en el Editor de Señales, activar la opción 'Snap to Grid' situada en el menú 'View' de la barra de herramientas principal.

- Nota: En la ventana de selección de nodos a introducir es necesario desplegar todo tipo de señales (entradas, salidas, registros, etc.) que se producen como resultado de la compilación para poder escoger aquellas que muestren el funcionamiento del diseño.

4.- EJERCICIOS DE DISEÑO

4.1.- Panel de control de una puerta automática.

En este ejercicio se debe diseñar un sistema que se encargará de controlar el funcionamiento de una puerta de apertura automática, basado en la máquina de estados de Moore. El funcionamiento es sencillo, se trata de un panel que se desliza horizontalmente para la apertura o cerrado de la puerta. La señal que ordena la apertura o cierre de la misma se obtiene a partir de un conjunto de sensores. Esta señal indicará la presencia o no de algo o alguien en las cercanías de la puerta.

Se supondrá que la puerta no se abre de forma continua, sino que lo hace en tres etapas, es decir, empleará tanto en su apertura como en su cierre tres ciclos de la señal de reloj. La frecuencia de esta señal de reloj será de 1 Hz a la hora de volcar a la placa.

Las salidas del circuito que se encargarán de indicar la situación de la puerta son las siguientes:

L1: primera etapa de apertura de la puerta.
L2: segunda etapa de apretura de la puerta.
L3: tercera etapa de apertura de la puerta.
A: puerta abriéndose.
C: puerta cerrándose.

El esquema de la puerta y su secuencia de apertura o cierre sería la mostrada en la siguiente figura:

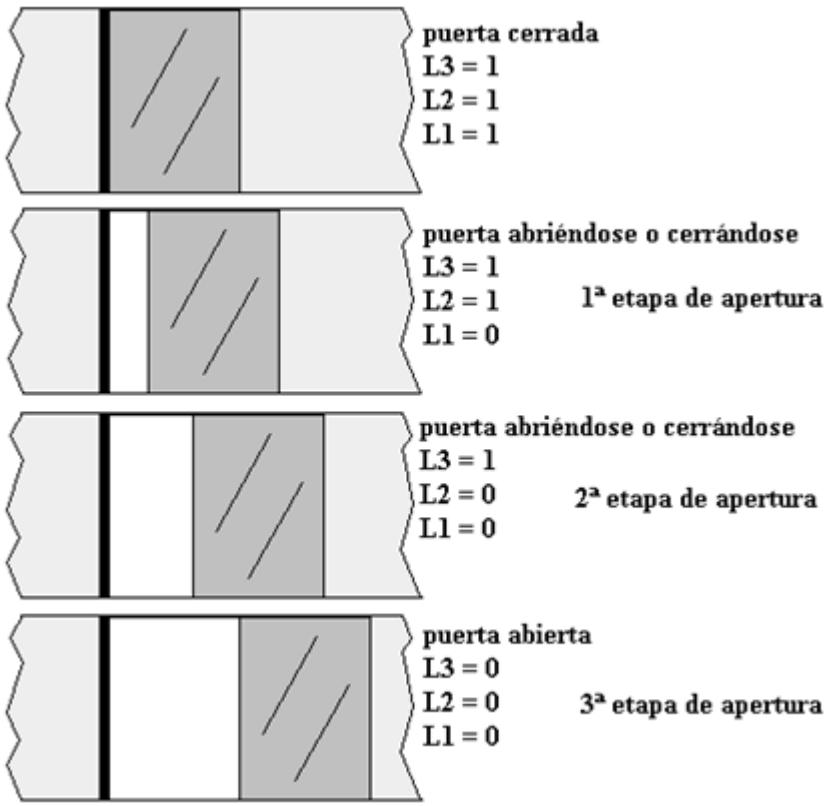


Figura 1. Etapas de apertura o cierre de la puerta

El circuito que representa este diseño se muestra a continuación, en la Figura 2:

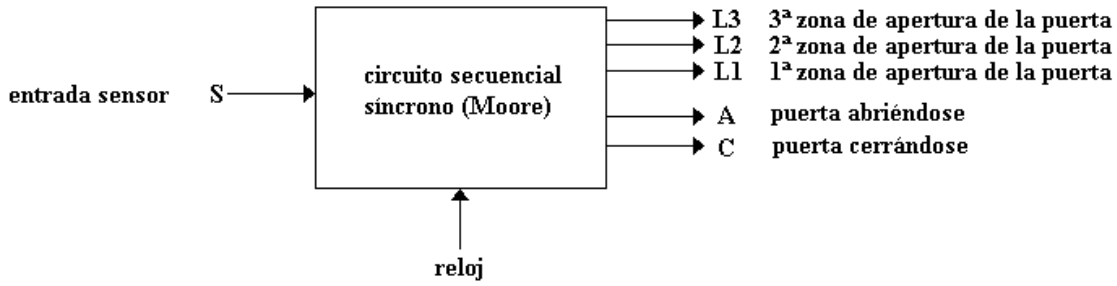


Figura 2. Circuito representativo del diseño

Cada vez que los sensores detecten la presencia de algo o alguien, se deberá abrir la puerta completamente, incluso en el caso de que esta presencia desaparezca antes de que la puerta hay llegado a abrirse del todo. Una vez que la puerta esté abierta, deberá permanecer así hasta que los sensores dejen de detectar la presencia de algo o alguien en sus cercanías.

En el diseño también se deberá tener en cuenta que si la puerta se encuentra en fase de cerrado, en el caso de que los sensores vuelvan a detectar presencia, se interrumpe el cerrado de la misma volviendo a abrir la puerta completamente. En esta interrupción de cerrado de la puerta se deberá jugar con los diferentes grados de apertura de la puerta, para que al interrumpirse el cierre de la misma, ésta comience a abrirse desde la posición en la que se encuentre.

La estructura de la máquina de Moore en código VHDL es casi idéntica a la de Mealy, teniendo en cuenta que el código a de sufrir una modificación para que las salidas actuales sean sólo función del estado actual.

Tomando como base el código del apartado 3.1, lo único que cambia es el bucle case, en el cual para cada estado se tiene un código como el que se muestra en el siguiente ejemplo:

```
case estado_actual is
        when q0 =>
                S1<='1';
                S2<='0';
                S3<='1';

                if X='0' then
                  estado_siguiente<=q0;
                else
                  estado_siguiente<=q1;
                end if;
```

En este ejemplo se ve como las salidas actuales sólo son función del estado actual.

Se deberá diseñar el código, correspondiente a una máquina de Moore, en VHDL, que represente el comportamiento del sistema explicado. A continuación se realizará su simulación y volcado a la tarjeta educacional DE2 (o UP2 en su caso), en la cual se emplearán tres diodos Led consecutivos para indicar el grado de apertura de la puerta y otros dos para

indicar si la misma se está abriendo o cerrando. Además, se simulará la señal de entrada de sensores mediante un pulsador de la placa.

Como simular todos los cambios de estado resulta bastante engorroso, bastará con simular los que se crean más convenientes o representativos del funcionamiento del diseño.

A la hora de volcar el diseño, habrá de tenerse en cuenta la lógica negativa con la que se trabaja en la placa.

4.2.- Cerradura digital con biestables

Realizar el ejercicio del apartado 3.1 mediante la utilización de biestables JK. Implementar el diseño y realizar su correspondiente simulación. Visualizar el cambio de estados y comprobar que el resultado coincide con el obtenido a través del diseño realizado en VHDL.

ANEXO

Tablas correspondientes a la asignación de pines de los diodos Led y pulsadores de la placa DE2.

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_AE23	LED Red[0]
LEDR[1]	PIN_AF23	LED Red[1]
LEDR[2]	PIN_AB21	LED Red[2]
LEDR[3]	PIN_AC22	LED Red[3]
LEDR[4]	PIN_AD22	LED Red[4]
LEDR[5]	PIN_AD23	LED Red[5]
LEDR[6]	PIN_AD21	LED Red[6]
LEDR[7]	PIN_AC21	LED Red[7]
LEDR[8]	PIN_AA14	LED Red[8]
LEDR[9]	PIN_Y13	LED Red[9]
LEDR[10]	PIN_AA13	LED Red[10]
LEDR[11]	PIN_AC14	LED Red[11]
LEDR[12]	PIN_AD15	LED Red[12]
LEDR[13]	PIN_AE15	LED Red[13]
LEDR[14]	PIN_AF13	LED Red[14]
LEDR[15]	PIN_AE13	LED Red[15]
LEDR[16]	PIN_AE12	LED Red[16]
LEDR[17]	PIN_AD12	LED Red[17]
LEDG[0]	PIN_AE22	LED Green[0]
LEDG[1]	PIN_AF22	LED Green[1]
LEDG[2]	PIN_W19	LED Green[2]
LEDG[3]	PIN_V18	LED Green[3]
LEDG[4]	PIN_U18	LED Green[4]
LEDG[5]	PIN_U17	LED Green[5]
LEDG[6]	PIN_AA20	LED Green[6]
LEDG[7]	PIN_Y18	LED Green[7]
LEDG[8]	PIN_Y12	LED Green[8]

Tabla 2. Asignación de pines para diodos LED.

Signal Name	FPGA Pin No.	Description
KEY[0]	PIN_G26	Pushbutton[0]
KEY[1]	PIN_N23	Pushbutton[1]
KEY[2]	PIN_P23	Pushbutton[2]
KEY[3]	PIN_W26	Pushbutton[3]

Tabla 3. Asignación de pines de los pulsadores

RESOLUCIÓN PRÁCTICA 7

A continuación se presentan las soluciones de los distintos ejercicios planteados en el guión de la práctica 7.

3.1.- CERRADURA DIGITAL

Una vez que se ha introducido el código del diseño, se procede a su simulación, tal y como se explica en el enunciado, para comprobar el correcto funcionamiento.

Figura 1. Resultados de funcionamiento de la cerradura digital

La Figura 1 muestra como la señal de salida 'Z' queda activa a nivel alto tras recibir la secuencia '0-1-1' por parte de la señal de entrada 'X'. A su vez, las salidas 'Q0-Q4' representan la transición de estados que se produce. En la Figura 1, un nivel alto en estas salidas, representa el estado en el que se encuentra el sistema según la señal de entrada y los flancos de subida de la señal de reloj.

- Nota: En la ventana de selección de nodos a introducir es necesario desplegar todo tipo de señales (entradas, salidas, registros, etc.) que se producen como resultado de la compilación para poder escoger aquellas que muestren el funcionamiento del diseño.

4.- EJERCICIOS DE DISEÑO

4.1.- Panel de control de una puerta automática

Teniendo en cuenta el enunciado del ejercicio se tiene que las salidas del circuito que se encargarán de indicar la situación de la puerta son las siguientes:

L1: primera etapa de apertura de la puerta.
L2: segunda etapa de apretura de la puerta.
L3: tercera etapa de apertura de la puerta.
A: puerta abriéndose.
C: puerta cerrándose.

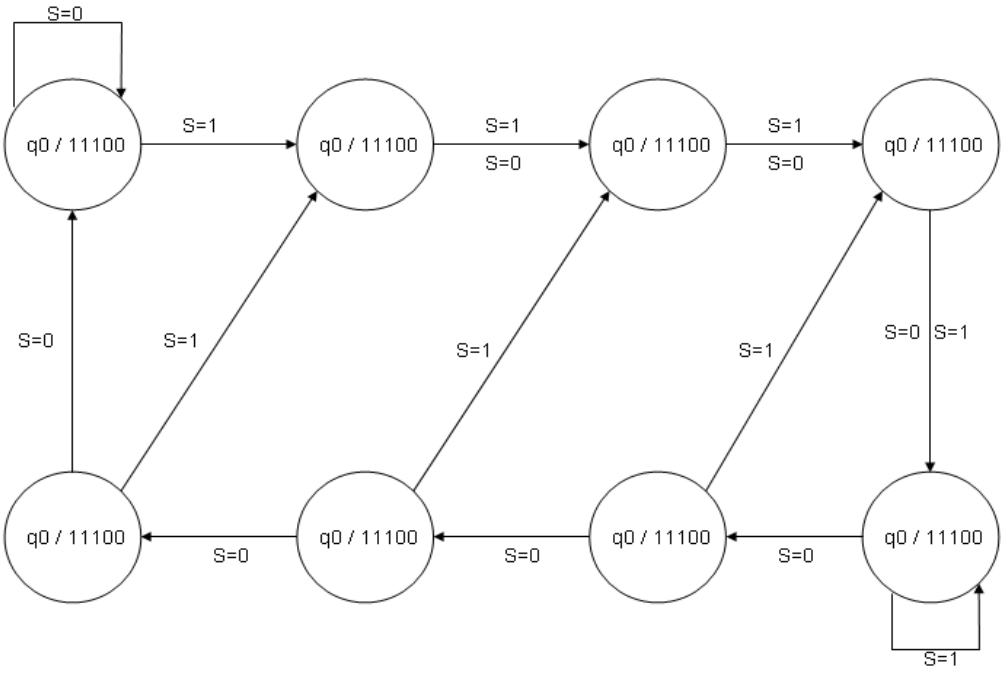
Por su parte, la entrada al circuito procedente de los sensores de detección de presencia funciona de la siguiente manera:

S = 0 → No se detecta presencia en las inmediaciones de la puerta.
S = 1 → Existe presencia en las inmediaciones de la puerta.

Los estados por los que atravesará el diseño son los expuestos a continuación:

q0: puerta cerrada
q1: puerta abriéndose, primer grado de apertura.
q2: puerta abriéndose, segundo grado de apertura.
q3: puerta abriéndose, tercer grado de apertura.
q4: puerta completamente abierta.
q5: puerta cerrándose, tercer grado de apertura.
q6: puerta cerrándose, segundo grado de apertura.
q7: puerta cerrándose, primer grado de apertura.

En base a los estados mencionados, se construye el correspondiente diagrama de estados, mostrado en la Figura 2.



- Interpretación: *Estado actual / L3 L2 L1 A C*

Figura 2. Diagrama de estados de la puerta automática

Además del diagrama de estados, se presenta la tabla de transiciones con sus correspondientes valores de las señales de salida, que permite entender completamente el comportamiento del diseño.

Entrada	Estado Actual	Estado Siguiente	Salidas				
S	Q(t)	Q(T+1)	L3	L2	L1	A	C
0	q0	q0	1	1	1	0	0
1	q0	q1	1	1	0	1	0
X	q1	q2	1	0	0	1	0
X	q2	q3	0	0	0	1	0
X	q3	q4	0	0	0	0	0
0	q4	q5	1	0	0	0	1
1	q4	q4	0	0	0	0	0
0	q5	q6	1	1	0	0	1
1	q5	q3	0	0	0	1	0
0	q6	q7	1	1	1	0	1
1	q6	q2	1	0	0	1	0
0	q7	q0	1	1	1	0	0
1	q7	q1	1	1	0	1	0

Tabla 1.

A partir de esta tabla ya se está en disposición de poder escribir el código VHDL que representa al diseño.

```
--SISTEMA DE CONTROL PARA PUERTA AUTOMÁTICA

entity puerta is port(
        S,clock : in bit;                    --entrada de señales de sensor y reloj
        L3,L2,L1,A,C : out bit               --salidas del sistema
        );
end puerta;

architecture arch_puerta of puerta is
 type tipo_estado is (q0,q1,q2,q3,q4,q5,q6,q7);          -- almacenamiento de estados
 signal estado_actual, estado_siguiente : tipo_estado;
    begin
        process (estado_actual,S)            --proceso que produce los cambios de estados
          begin
             case estado_actual is
        when q0 =>
                L3<='1';
                L2<='1';
                L1<='1';
                A<='0';
                C<='0';
        if S ='0' then
                estado_siguiente<=q0;
        else
                estado_siguiente<=q1;
        end if;
        when q1 =>
                L3<='1';
                L2<='1';
                L1<='0';
                A<='1';
                C<='0';
                estado_siguiente<=q2;          --caso donde el cambio de estado
                                               --no depende del valor de entrada
        when q2 =>
                L3<='1';
                L2<='0';
                L1<='0';
                A<='1';
                C<='0';
                estado_siguiente<=q3;
       when q3 =>
                L3<='0';
                L2<='0';
                L1<='0';
                A<='1';
                C<='0';
                estado_siguiente<=q4;

       when q4 =>
                L3<='0';
```

```
                L2<='0';
                L1<='0';
                A<='0';
                C<='0';
        if S ='0' then
                estado_siguiente<=q5;
        else
                estado_siguiente<=q4;
        end if;
        when q5 =>
                L3<='1';
                L2<='0';
                L1<='0';
                A<='0';
                C<='1';
        if S ='0' then
                estado_siguiente<=q6;
        else
                estado_siguiente<=q3;
        end if;
        when q6 =>
                L3<='1';
                L2<='1';
                L1<='0';
                A<='0';
                C<='1';
        if S ='0' then
                estado_siguiente<=q7;
        else
                estado_siguiente<=q2;
        end if;
        when q7 =>
                L3<='1';
                L2<='1';
                L1<='1';
                A<='0';
                C<='1';
        if S ='0' then
                estado_siguiente<=q0;
        else
                estado_siguiente<=q1;
        end if;
    end case;
end process;

        process
          begin
              wait until clock' event and clock='1';
                estado_actual<=estado_siguiente;
        end process;
end arch_puerta;
```

En base al código introducido, una posible simulación en la que se ven las diferentes situaciones es la siguiente:

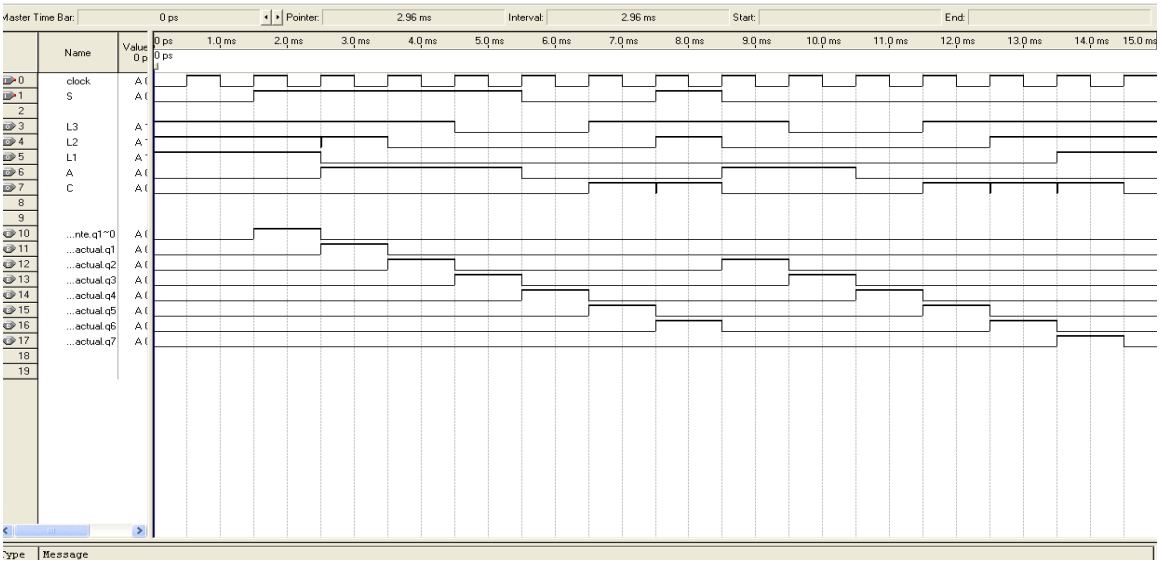


Figura 3. Resultados del sistema para la puerta automática

La Figura 3 muestra los resultados obtenidos tras realizar una simulación de apertura y cierre de la puerta. Las señales 'L1-L3', 'A' y 'C' muestran la combinación de cada estado, abriendo o cerrando la puerta, y las salidas 'Q0-Q7' muestran la correspondiente transición de estados, teniendo un nivel alto en el estado que se encuentra activo.

A continuación, se crea el símbolo correspondiente al diseño, para poder ser utilizado en el circuito[1] que será volcado a la tarjeta educacional, que se muestra en la Figura 4.

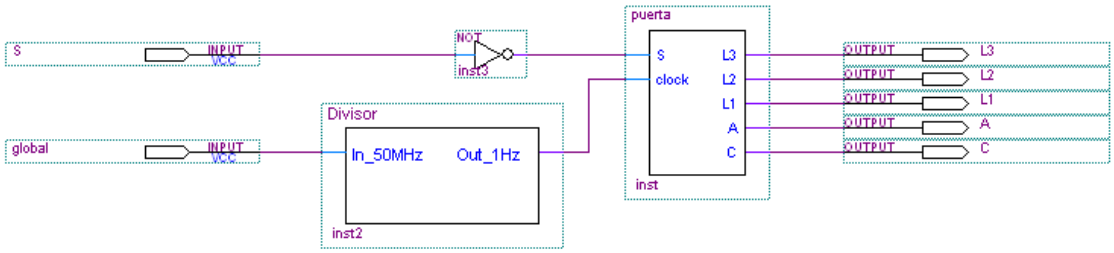


Figura 4. Circuito de volcado para la puerta automática

[1] *En principio, el circuito está pensado para ser volcado sobre la tarjeta educacional DE2, si bien, por cualquier motivo, se tendría que emplear la placa UP2, se deberá cambiar el divisor de frecuencia y las salidas han de ser negadas ya que los Led de la placa trabajan con lógica negativa. En este caso basta con emplear un divisor que adapte los 25 MHz de la señal de reloj de la placa a una señal de 1Hz.*

En este circuito cabe destacar como la entrada 'S' correspondiente al pulsador ha sido negada debido a que éstos trabajan con lógica negativa en la placa DE2. Por otra parte, a diferencia de los pulsadores, los diodos LED de la palca trabajan con lógica positiva, luego sus correspondientes salidas en el circuito no son negadas.

4.2.- Cerradura digital con biestables:

Observando el enunciado del apartado 3.1, tenemos que la secuencia 0-1-1 es la que permite la apertura de la cerradura mientras que la llegada de dos ceros consecutivos provoca su cierre.

Denominando 'z' a la señal de salida, se considera que:

z = 0 → cerradura cerrada.
z = 1 → cerradura abierta.

Los posibles estados del sistema son los siguientes:

q0: Estado inicial. No ha llegado ningún uno.
q1: Introducción de '1'.
q2 Llega la secuencia '1-1'.
q3: Llega la secuencia '0-1-1'.
q4: Llega un '0' estando la cerradura abierta.

El diagrama de estados se presenta en la siguiente figura:

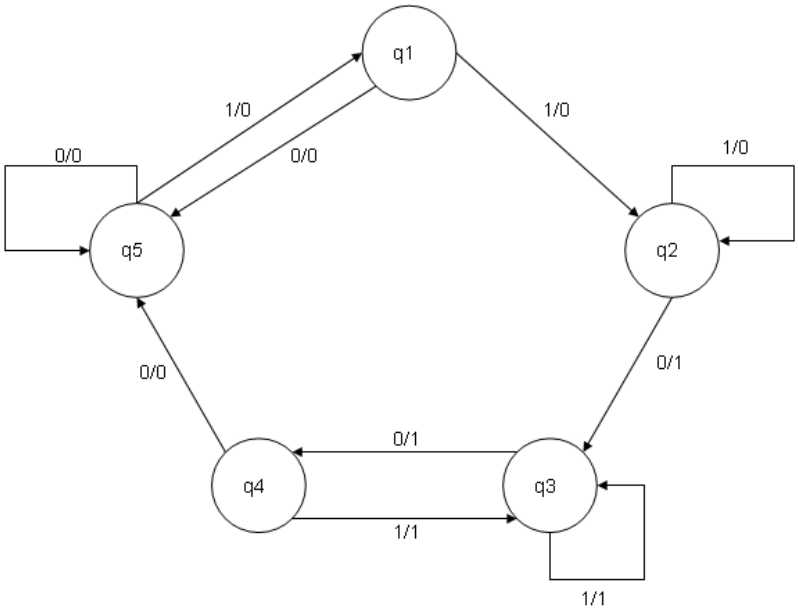


Figura 5. Diagrama de estados

A continuación se representa la tabla de transiciones que muestra el comportamiento de los estados, en la que se obtiene el estado siguiente y la salida, dependiendo del estado actual y de la entrada.

Estado Actual Q(t)	Entrada X	Estado Siguiente Q(t+1)	Salida Z
q0	0	q0	0
	1	q1	0
q1	0	q0	0
	1	q2	0
q2	0	q3	1
	1	q2	0
q3	0	q4	1
	1	q3	1
q4	0	q0	0
	1	q3	1

Tabla 2.

La asignación y codificación de estados queda como sigue:

N = 5 estados (q0, q1, q2, q3, q4) $\Rightarrow$ $2^{n-1} < N = 5 \leq 2^n$

Es necesario un número de tres biestables para implementar el circuito.

Por tanto, la tabla de asignación de estados es la siguiente:

Estados	Q3	Q2	Q1
q0	0	0	0
q1	0	0	1
q2	0	1	0
q3	0	1	1
q4	1	0	0

Tabla 3.

Se van a utilizar biestables JK, cuyo funcionamiento se muestra en la Tabla 4.

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabla 4.

A continuación se muestra la tabla de transiciones codificada:

Entrada x(t)	Estado actual Q(t)			Estado siguiente Q(t+1)			Excitaciones						Salida Z(t)
	Q3	Q2	Q1	Q3	Q2	Q1	J3	K3	J2	K2	J1	K1	
0	0	0	0	0	0	0	0	X	0	X	0	X	0
0	0	0	1	0	0	0	X	1	0	X	0	X	0
0	0	1	0	0	1	1	1	X	X	0	0	X	1
0	0	1	1	1	0	0	X	1	X	1	1	X	1
0	1	0	0	0	0	0	0	X	0	X	X	1	0
0	1	0	1	X	X	X	X	X	X	X	X	X	X
0	1	1	0	X	X	X	X	X	X	X	X	X	X
0	1	1	1	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	1	1	X	0	X	0	X	0
1	0	0	1	0	1	0	X	1	1	X	0	X	0
1	0	1	0	0	1	0	0	X	X	0	0	X	0
1	0	1	1	0	1	1	X	0	X	0	0	X	1
1	1	0	0	0	1	1	1	X	1	X	X	1	1
1	1	0	1	X	X	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X

Tabla 5.

A partir del método de Karnaugh, se obtienen las expresiones simplificadas para los diferentes biestables a partir del estado actual.

Función J1:

xQ_1/Q_2Q_3	00	01	11	10
00	0	0	1	0
01	X	X	X	X
11	X	X	X	X
10	0	0	0	0

$J1 = \overline{X} \cdot Q2 \cdot Q3$

Función K1:

xQ_1/Q_2Q_3	00	01	11	10
00	X	X	X	X
01	1	X	X	X
11	1	X	X	X
10	X	X	X	X

$K1 = 1$

Función J2:

xQ_1/Q_2Q_3	00	01	11	10
00	0	0	X	X
01	0	X	X	X
11	1	X	X	X
10	0	1	X	X

$J2 = X \cdot Q3 + X \cdot Q1$

Función K2:

xQ_1/Q_2Q_3	00	01	11	10
00	X	X	1	0
01	X	X	X	X
11	X	X	X	X
10	X	X	0	0

$K2 = \overline{X} \cdot Q3$

Función J3:

xQ_1/Q_2Q_3	00	01	11	10
00	0	X	X	1
01	0	X	X	X
11	1	X	X	0
10	1	X	X	X

$J3 = \overline{X} \cdot Q2 + X \cdot \overline{Q2} = X \oplus Q2$

Función K3:

xQ_1/Q_2Q_3	00	01	11	10
00	X	1	1	X
01	X	X	X	X
11	X	1	0	X
10	X	X	X	X

$K3 = \overline{X} + \overline{Q2}$

Función Z:

xQ_1/Q_2Q_3	00	01	11	10
00	0	0	1	1
01	0	X	X	X
11	1	X	X	X
10	0	0	1	0

$$Z = \overline{X} \cdot Q2 + X \cdot Q1 + Q2 \cdot Q3$$

El circuito introducido en el Editor Gráfico se muestra en la Figura 6.

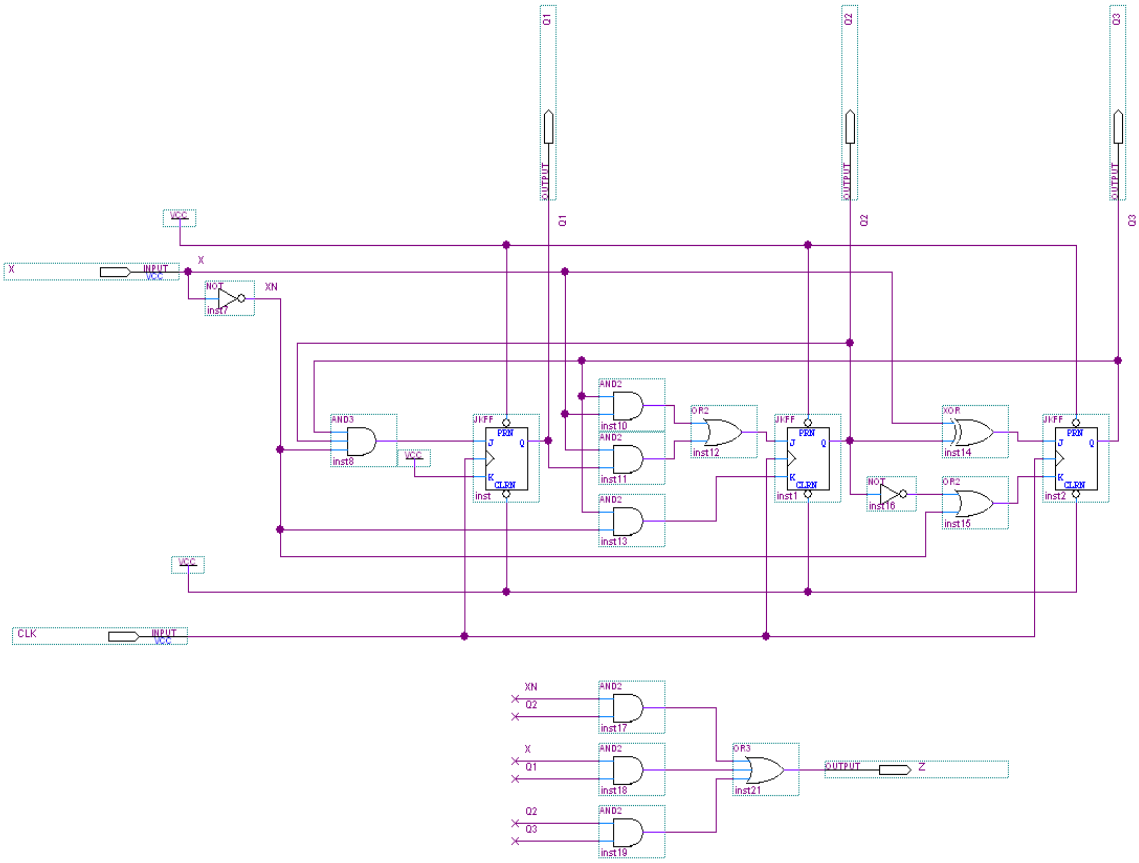


Figura 6. Cerradura digital mediante biestables

Realizando la correspondiente simulación, se obtienen los resultados mostrados en la Figura 7, los cuales habrán de ser comparados con los obtenidos en la Figura 1. Para ello se introducen los mismos parámetros para la señal de entrada y la señal de reloj.

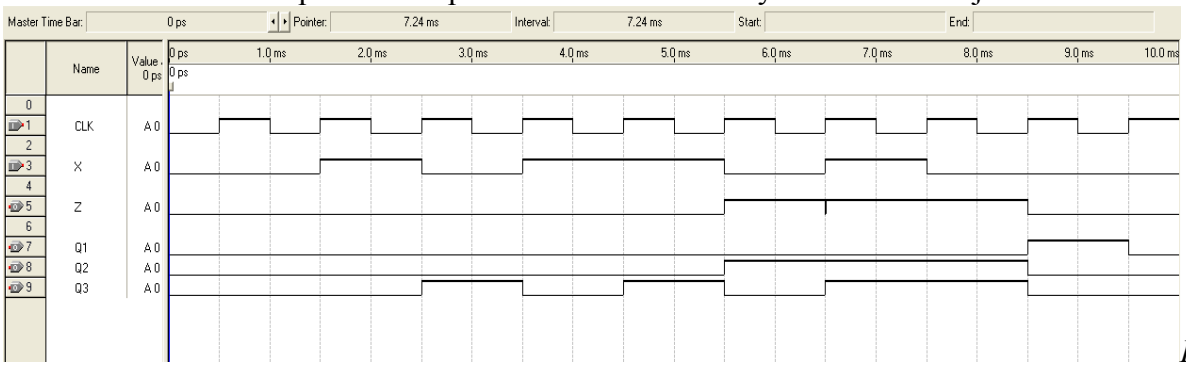


Figura 7. Resultados de funcionamiento de la cerradura digital

En la Figura 7 se aprecia el comportamiento del sistema frente a la señal de entrada 'X'. La salida 'Z' se activa tras recibir la secuencia '0-1-1' y vuelve a nivel bajo tras recibir dos ceros consecutivos. A su vez, las salidas 'Q1', 'Q2', y 'Q3' muestran la correspondiente transición de estados. Además, si se compara con el mismo diseño realizado en VHDL, se ve que los resultados son los mismos, lo que ratifica que el problema está bien resuelto.

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

(1º Ingeniería Técnica en Informática de Gestión)

Práctica 8: Diseño Libre

1.- OBJETIVOS

Aplicar los conocimientos adquiridos a lo largo de la realización de las prácticas de la asignatura.

2.- MATERIAL

- Ordenador personal con el software QUARTUS II® de Altera®.
- Manual introductorio al software.
- Guión de prácticas.
- Tarjeta educacional DE2 ó UP2.

3.- DISEÑO LIBRE

Realizar un diseño de libre elección por el alumno en el que se demuestre lo aprendido tras la realización de las prácticas. Dicho diseño deberá ser simulado y volcado en la tarjeta educacional para comprobar su correcto funcionamiento.

CAPÍTULO 9.

GUIONES DE PRÁCTICAS PARA LA TITULACIÓN

'INGENIERÍA DE TELECOMUNICACIÓN'

CIRCUITOS ELECTRÓNICOS DIGITALES

(3º Ingeniería de Telecomunicación)

Práctica 2: Entorno Software de Simulación.

- **Puertas Lógicas.**
- **Circuitos Aritméticos.**

PRIMERA PARTE. PUERTAS LÓGICAS

El haber estudiado el manual introductorio del software es muy importante y facilitará en gran medida el desarrollo de la práctica.

Es muy importante entender perfectamente esta práctica y obtener soltura en el manejo del software, ya que será el punto de partida para el resto de las prácticas.

1.1.- OBJETIVOS

c) Introducción al manejo del software Quartus II utilizado para la implementación de diseños lógicos empleando PLD`s, no llegando a programar en esta práctica las tarjetas educacionales DE2 ó UP2.

d) Se repetirán los ejercicios realizados en la práctica 1 con el software indicado anteriormente, realizando el primero de los ejercicios de manera guiada con el fin de observar que los resultados obtenidos en la práctica 1 coinciden con los simulados en la actual práctica.

1.2.- MATERIAL

- Ordenador personal con el software QUARTUS II® de Altera®.
- Manual introductorio al software.
- Guión de prácticas.

1.3.- DESARROLLO DE LA PRÁCTICA

1.3.1.- PUERTAS NOT, AND Y OR

El diseño a introducir será el siguiente:

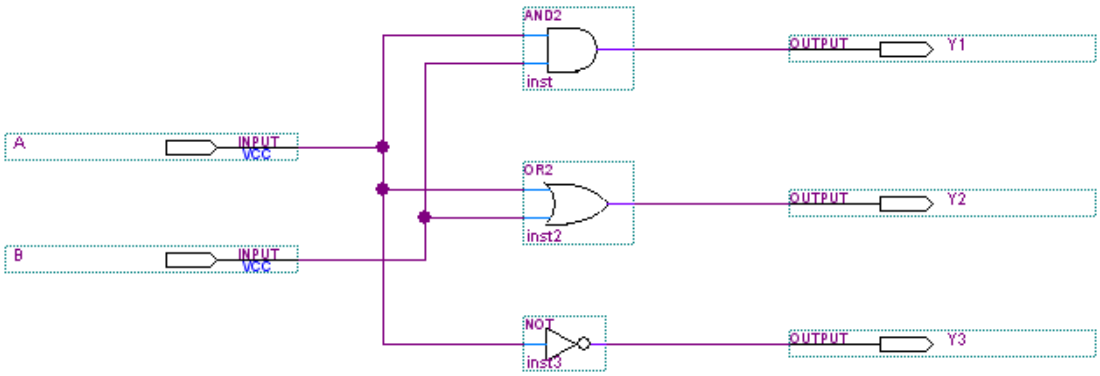


Figura 1. Diseño Ejercicio 1.3.1

Para la implementación del circuito y posterior comprobación de resultados se seguirán los pasos que se indican a continuación.

1º- Realización Gráfica del Diseño.

i) Inicializar el software Quartus II.

j) El primer paso a la hora de realizar un diseño con Quartus II consiste en la definición del proyecto en el que se va a operar. Un proyecto consiste en englobar una serie de archivos de diseño dentro de un subdirectorio dado.
Para definir el proyecto, es necesario acudir a la opción 'New Project Wizard' situada en el menú 'File', que es el asistente para la definición de proyectos. Este asistente ofrece una serie de ventanas que permiten realizar las siguientes operaciones:

I. Definición de nombre y directorio de almacenamiento del proyecto.

- **Importante: No se pueden crear dos proyectos en un mismo directorio de trabajo, ya que posteriormente, en el proceso de compilación, se producirá conflicto entre ambos.**

II. Inclusión de archivos anteriores que quieran ser utilizados en el presente proyecto. Para esta práctica no es necesario añadir ningún archivo.

III. Selección de la familia y dispositivo que vayamos a utilizar. Aunque en esta práctica los diseños no van a ser implementados sobre las tarjetas educacionales, es bueno acostumbrarnos desde ahora a seleccionar el dispositivo EPF10K70RC240-4 de la familia FLEX10K, en el caso de la tarjeta educacional UP2, o el dispositivo EP2C35F672C6 de la familia Cyclone II, en el caso de la placa DE2, para la compilación del diseño. Esto se muestra en la Figura 2.

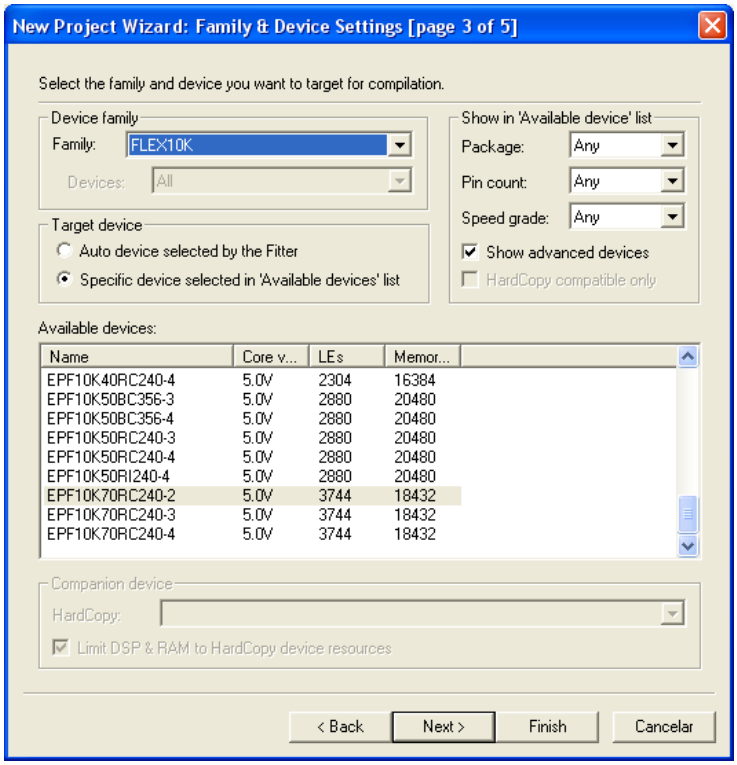


Figura 2. Selección de familia y dispositivo para la compilación

IV. Selección de otras herramientas EDA. No es necesario ninguna herramienta adicional luego pasamos directamente al siguiente paso.

V. Por último, aparece una ventana que muestra un resumen de lo incluido en los pasos anteriores. Pulsando sobre el botón 'Finish' quedará definido el nuevo proyecto y se estará en disposición de empezar a trabajar.

k) Abrir un nuevo archivo para el Editor Gráfico. Para ello se debe acceder al icono que representa un folio en blanco y, en la nueva ventana, seleccionar la opción 'Block Diagram/Schematic File'.

l) Introducir los elementos que forman el circuito a implementar, mostrado en la Figura 1. Esto se puede hacer pinchando con el botón derecho del ratón sobre la pantalla, eligiendo la opción 'Insert' seguido de 'Symbol…' en el menú que se despliega. Al realizar esto aparecerá la ventana que se muestra en la Figura 3 en la cual se puede escribir directamente el nombre del componente a utilizar o bien buscarlo en las diferentes librerías existentes. Los componentes que nosotros estamos buscando se encuentran en la carpeta de primitivas, opción 'logic', sin embargo al ser componentes conocidos y muy sencillos, se buscarán directamente por su nombre en la casilla 'Name'. Los componentes a utilizar son: input, and2 ('and' con dos entradas), or2 ('or' con dos entradas), not y output. Mediante la opción 'Repeat.insert mode' se incluye cada símbolo tantas veces como se quiera.

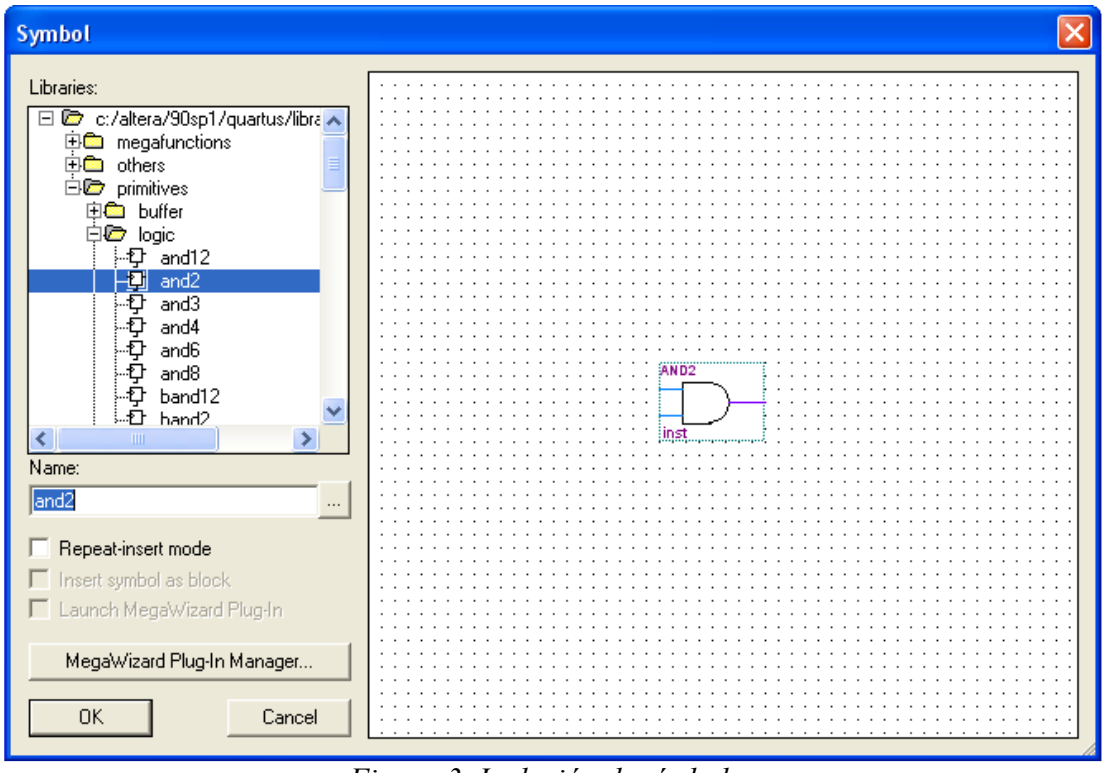


Figura 3. Inclusión de símbolos

m) Dibujar las conexiones. Para ello se pulsa sobre la herramienta 'Orthogonal Node Tool' y con el botón izquierdo del ratón pulsado se unen los elementos que interese.

n) Asignar nombres a las entradas y salidas. Para ello, hacer doble clic con el botón izquierdo o un clic con el botón derecho sobre cada una de las entradas y salidas. En el caso de utilizar el botón derecho, aparecerá un menú desplegable en el que se

debe seleccionar la opción 'Properties'. A continuación en la casilla 'Pin name (s)' introducir el nombre deseado.

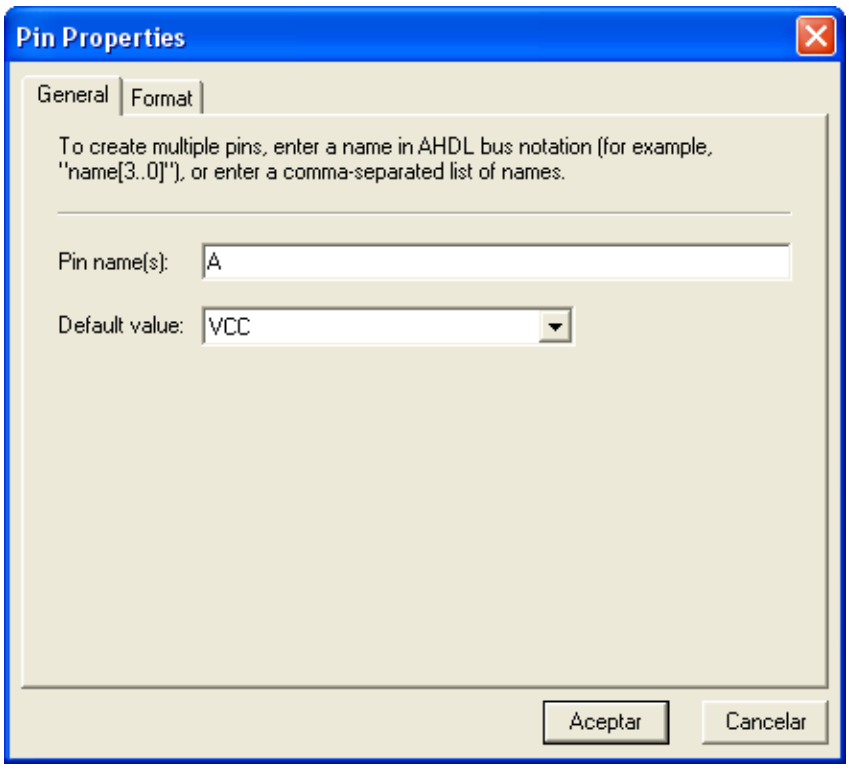


Figura 4. Introducción de nombres

Es imprescindible usar nombres únicos para los diferentes pines. Los pines y cables con el mismo nombre son conectados automáticamente sin que aparezca un cable visible en el diagrama esquemático.

o) Guardar el diseño en la carpeta definida para el proyecto. Para ello acceder a la orden 'Save' del menú 'File'.

Al llegar a este punto ya se ha introducido el diseño, por lo que, a continuación, se procederá a la compilación del mismo.

2º- Compilación del Diseño

La compilación del diseño conlleva la detección de posibles errores en la construcción o sintaxis del mismo, sintetiza el diseño lógico, produce información temporal para la simulación, ajusta el diseño al PLD seleccionado y genera el archivo necesario para el volcado del programa a la tarjeta de simulación.

Antes de iniciar la compilación es necesario que el diseño haya sido previamente guardado.

- **Importante: Tener en cuenta que en el caso de proyectos en los que se incluyen archivos de diseño de proyectos anteriores, el proceso de compilación se hace sobre el archivo jerárquicamente superior.**

El acceso a la herramienta de compilación de Quartus II puede realizarse de dos maneras diferentes:

- La primera de ellas consiste en acceder al menú desplegable de la pestaña 'Processing' de la barra de herramientas principal y seleccionar la opción 'Compiler Tool', como se ve en la Figura 5.
- La segunda opción consiste en iniciar directamente la compilación accediendo al icono asignado para ello, tal y como muestra la Figura 5. 

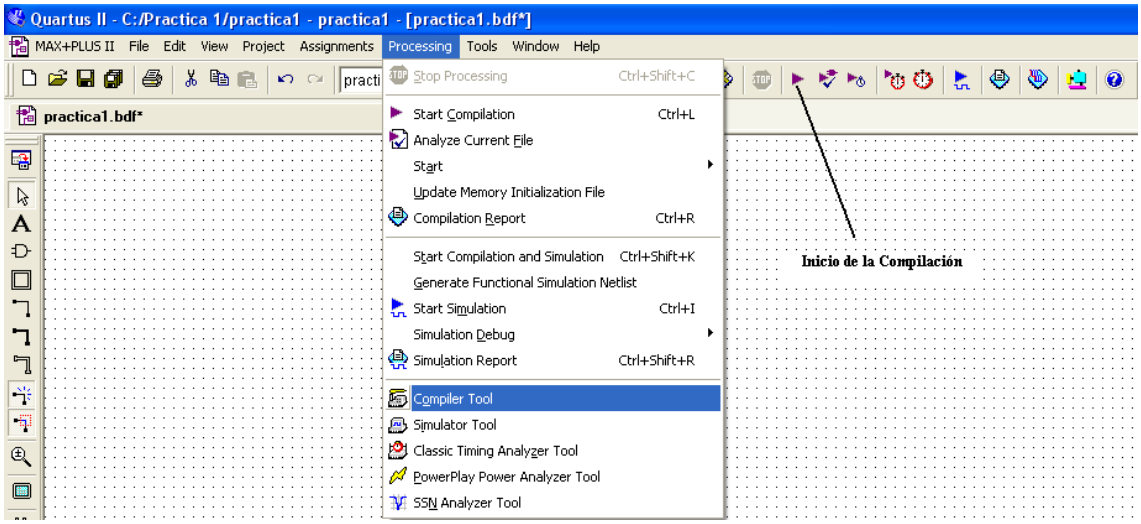


Figura 5. Apertura de la herramienta de compilación.

Se recomienda que el alumno escoja la primera de las opciones ya que de esta manera es posible ver la ventana de la herramienta de compilación y los procesos que se van dando en la misma.

La Figura 6 muestra la ventana del compilador.

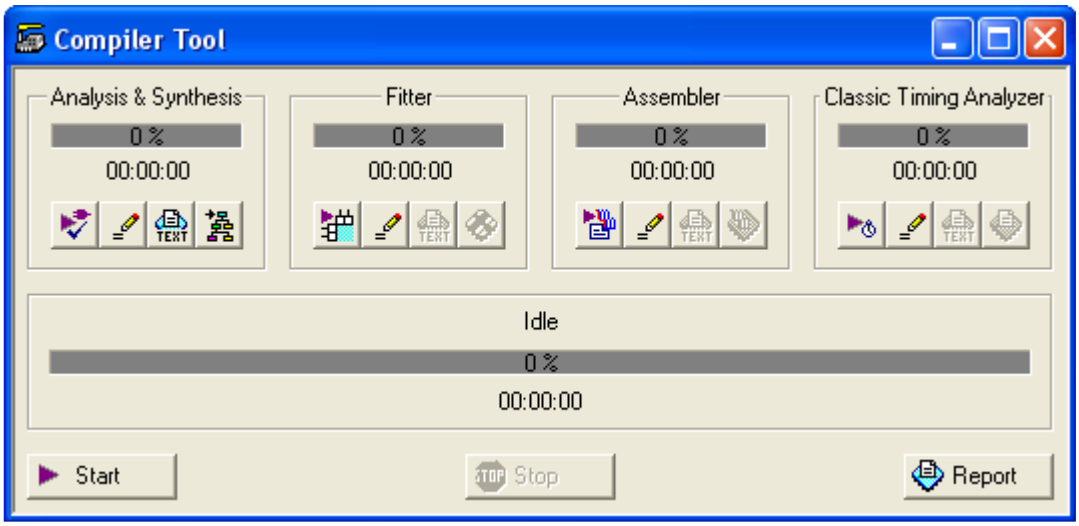


Figura 6. Ventana del compilador

El proceso de compilación completo comienza cuando se pulsa el botón 'Start', y éste puede ser interrumpido mediante el botón 'Stop'. Para ver las características del proceso es necesario recurrir al botón 'Report'.

Tras finalizar el proceso de compilación se presentará una ventana en la que se da información acerca de los posibles errores o advertencias que pudiera haber. Si no hay ningún error, se continuará con el proceso de simulación. Si por el contrario, hubiese algún error, éstos aparecerán en la ventana de mensajes del compilador en color rojo, y habrán de ser localizados y corregidos. Para localizar el error basta con seleccionarlo y pulsar el botón 'Locate'. Al hacer esto el programa regresa al editor en el que se diseñó el circuito y resalta el símbolo o el texto erróneo.

Un error habitual es no hacer caso a los mensajes de precaución (marcados en verde) que aparecen en la ventana de mensajes a la hora de compilar, los cuales permiten seguir, en el siguiente paso, con la simulación, pero puede que los resultados no sean los esperados.

Durante la compilación, el programa selecciona el dispositivo asignado previamente en la definición del proyecto. Si se hubiera obviado este paso el programa selecciona un dispositivo automáticamente. De todos modos siempre se podrá cambiar el PLD posteriormente.

3º- Edición de Señales

Para comprobar el correcto funcionamiento del diseño generado resulta necesario definir las señales de prueba que se van a introducir en el mismo. Esto se realiza mediante el Editor de Señales y los pasos a seguir son los siguientes:

g) Abrir un nuevo archivo del Editor de Señales. Para ello se pincha sobre el icono que representa una hoja en blanco y se selecciona la opción 'Vector Waveform File' desde la nueva ventana.
h) Fijar el tiempo de análisis desde el menú 'Edit' en la opción 'End Time…'. Fijar un valor alrededor de 5-10ms.
i) Introducir las señales que se quieren estudiar. Para ello se debe acceder al menú desplegable 'Edit' situado en la barra de herramientas principal y seleccionar la opción 'Insert' seguido de 'Insert Node or Bus…', con lo que se abrirá el siguiente cuadro.

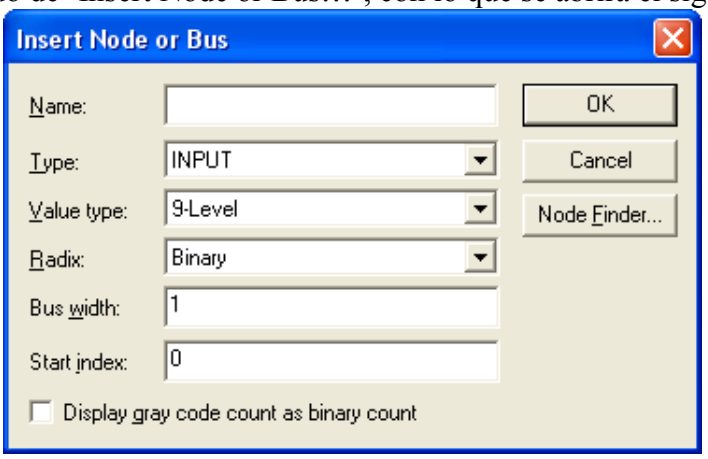


Figura 7. Tipo de señales

Esta ventana permite elegir el tipo de señales que interesa mostrar para la simulación del diseño. En nuestro caso dejaremos los parámetros que vienen por defecto y pulsaremos sobre el botón 'Node Finder…' para acceder a la siguiente ventana.

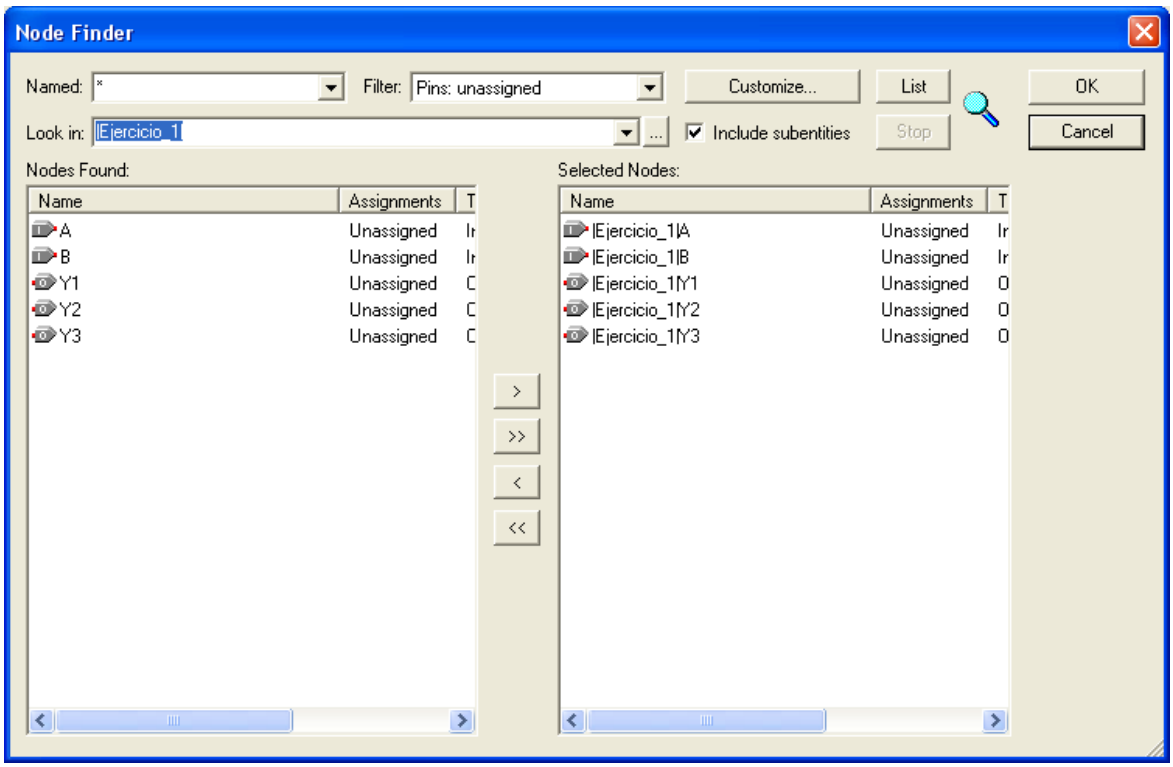


Figura 8. Selección de señales

En la ventana mostrada en la Figura 8 pincharemos sobre el botón 'List', con lo que nos aparecerán como señales disponibles todas las entradas y salidas correspondientes al diseño del proyecto en el que estamos trabajando. Aquí se han de seleccionar las señales con las que interesa trabajar, que en este caso particular son todas. A continuación se pulsará sobre el botón que indica la flecha hacia la derecha para incluir las señales en el Editor, por lo que ya están disponibles para la simulación. Para aceptar la operación pulsaremos sobre el botón 'OK'.

Aparecerá la siguiente ventana:

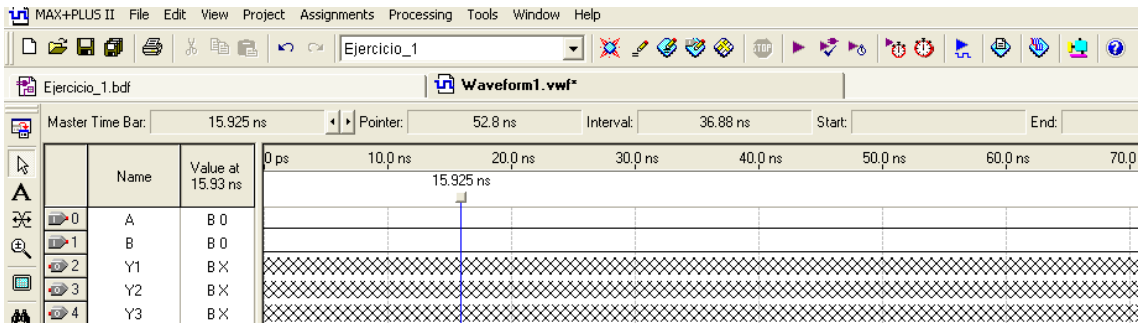


Figura 9. Señales para la simulación

En esta ventana se ve como las señales de entrada no tienen ningún valor asignado, por lo que se deberá elegir unos valores para las mismas de forma que representen los casos que interese estudiar.

j) Para simular los pulsadores utilizaremos señales cuadradas con una frecuencia de 1 KHz para la entrada A y otra de 500 Hz para la entrada B (esto se hace para que los retardos introducidos por las puertas no distorsionen los resultados que se quieren

obtener). De esta manera se obtendrán las distintas posibilidades de la tabla de verdad del circuito.

Lo primero que debemos hacer es seleccionar ambas señales de entrada y pinchar sobre ellas con el botón derecho del ratón, obteniendo la siguiente ventana.

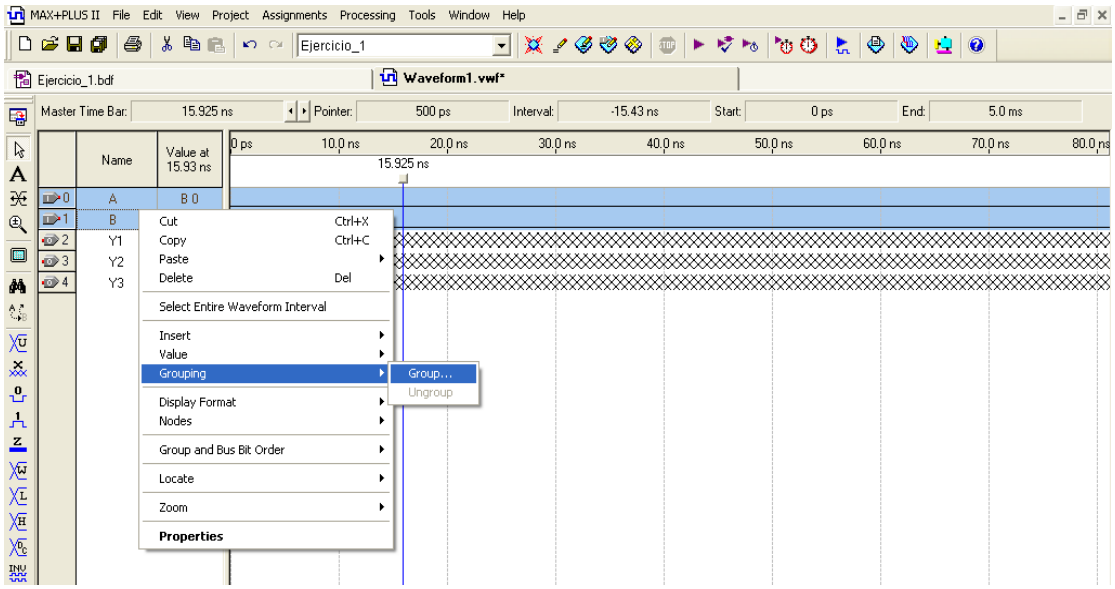


Figura 10. Agrupación de señales

En ella se elige la opción 'Grouping' seguido de 'Group…' con lo que se consigue agrupar las dos señales. Aparecerá una ventana en la que se le dará un nombre cualquiera al grupo. El siguiente paso será definir ese grupo. Para ello hay que pinchar con el botón derecho sobre el grupo creado y seleccionar la opción 'Value/Count Value…'. En la nueva ventana pinchar sobre la pestaña 'Timing' y fijar la opción 'Count Every' a la mitad del periodo más pequeño de interés. Esto se hace así porque esta opción indica el tiempo para el cual ocurre el cambio de estado de la señal. Si introducimos, por ejemplo, un valor de 0.5 ms, obtendremos una señal de 1 Khz.

A continuación, se deshace el grupo para ver claramente cada una de las señales por separado. Esto se hace pulsando con el botón derecho del ratón sobre el grupo y seleccionar la opción 'Grouping/Ungroup'.

**Nota: Asegurarse de que la opción 'Snap to grid' del menú 'View' se encuentre desactivada (por defecto suele estar activada), de lo contrario no se podrán fijar los semiperiodos que nosotros queramos*

k) Si es necesario, hacer un zoom in o zoom out, para ver claramente la señal. Las señales de entrada quedan como se muestra en la Figura 11.

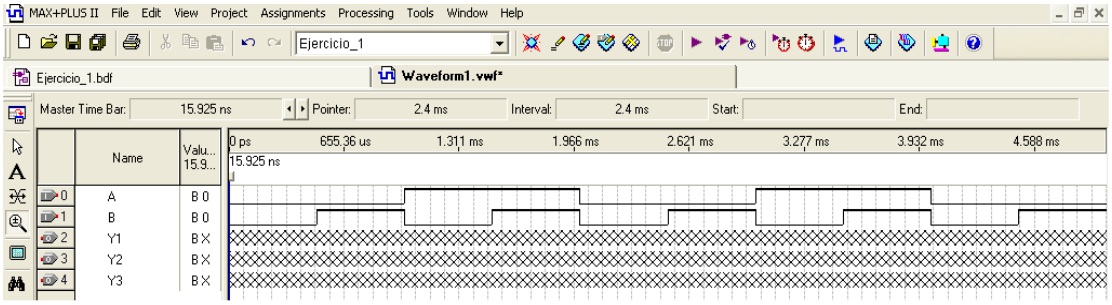


Figura 11. Señales de entrada

Como se puede observar, se obtienen todas las combinaciones posibles de una tabla de verdad de dos entradas.

l) Guardar el archivo, que al ser generado desde el Editor de Señales tendrá una extensión '.vwf'. Ya se han preparado las señales de entrada para proceder a la simulación y obtener los resultados correspondientes a las señales de salida.

4º- Simulación del Diseño.

La finalidad de este apartado es la comprobación del correcto funcionamiento del diseño realizado. Este paso es necesario hacerlo antes de realizar el volcado de los respectivos diseños a la tarjeta, debido a que podrían existir errores en los mismos que podrían estropearla.

Al igual que el resto de herramientas, tenemos varias opciones para acceder a la herramienta de simulación:

- La primera de ellas consiste en acceder al menú desplegable 'Processing' y elegir la opción 'Simulator Tool', como se ve en la Figura 12.

- La segunda opción consiste en iniciar directamente la simulación pulsando sobre el icono asignado para ello. 

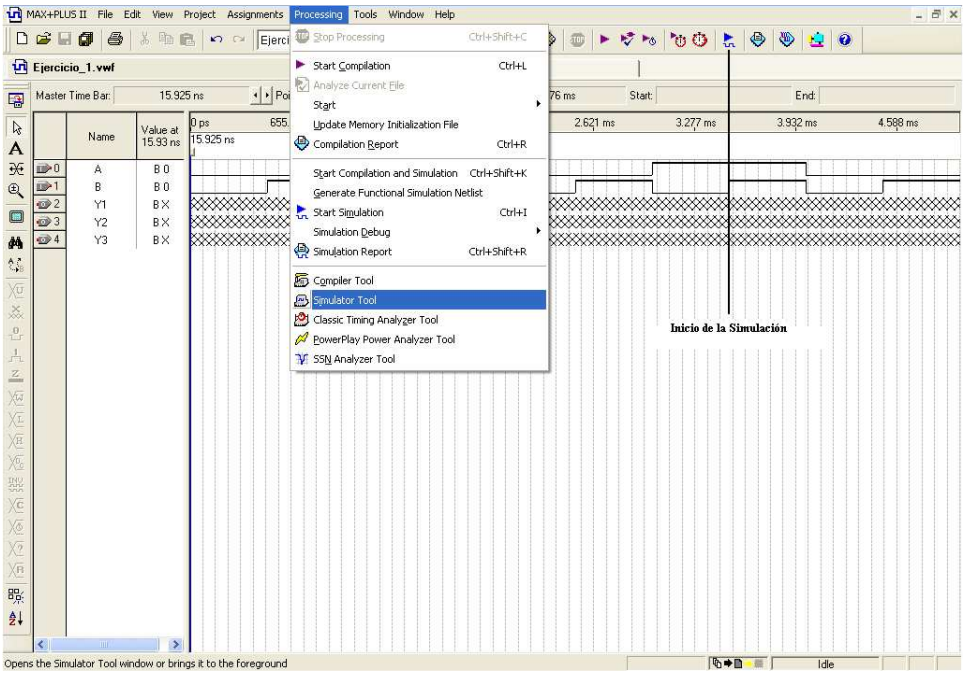


Figura 12. Acceso a la herramienta de simulación

Se recomienda realizar el acceso mediante la primera de las maneras ya que de esta forma tenemos acceso a la ventana de la herramienta, mostrada en la Figura 13.

Basta con pulsar el botón 'Start' para que comience la simulación.

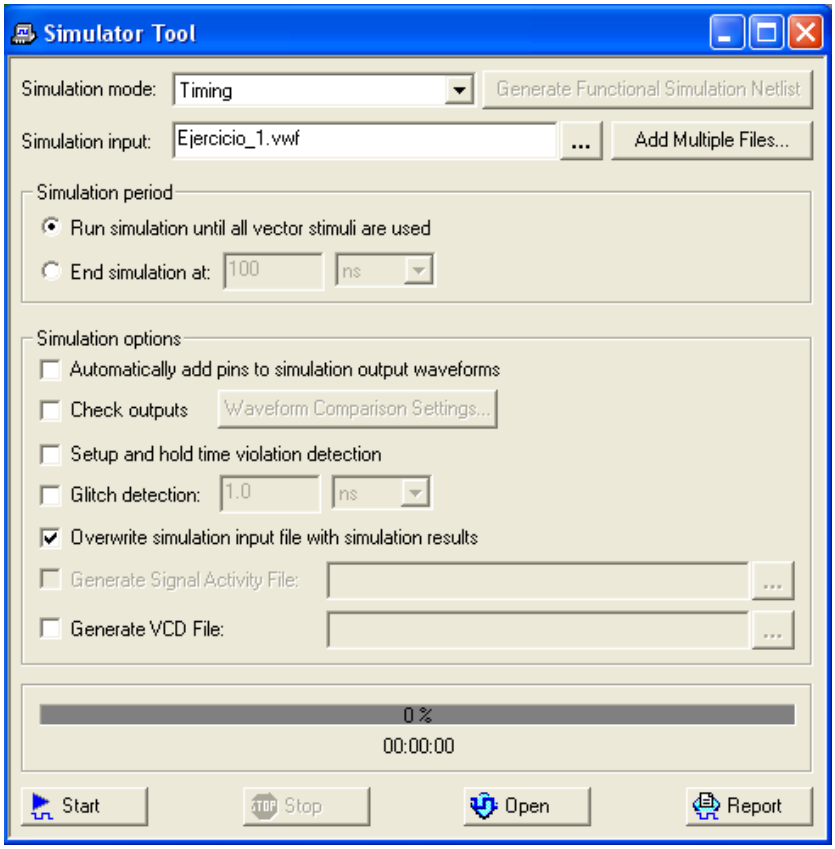


Figura 13. Ventana del simulador

Una vez terminada la simulación estaremos en disposición de poder ver los resultados. Para ello, en la misma ventana del simulador existen los botones 'Open' y 'Report'.

Si pulsamos sobre el botón 'Open' se modifica y se guarda el archivo original creado desde el Editor de Señales, pudiendo observar en el mismo tanto las entradas como las salidas resultado de la simulación.

Si pulsamos sobre el botón 'Report' podremos observar el resultado de la simulación (entradas y salidas), pero el archivo original creado con el Editor de Señales permanecerá sin cambios.

Comprobar que los resultados son los correctos.

3.2.- PUERTAS NAND, NOR Y XOR

En este segundo ejercicio el diseño a introducir será el siguiente:

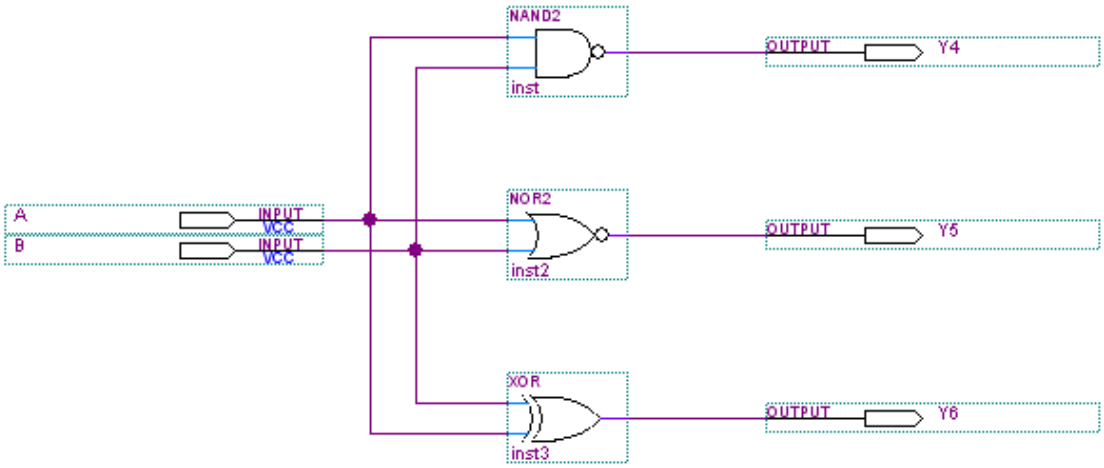


Figura 14. Diseño Ejercicio 3.2

Realizar los mismos pasos que en el apartado anterior y comprobar que los resultados son correctos y que coinciden con los obtenidos en la práctica cero.

SEGUNDA PARTE. CIRCUITOS ARITMÉTICOS

Las reglas para las operaciones aritméticas básicas son comunes a todos los sistemas de numeración. La aritmética binaria permite realizar cualquier tipo de operación, la diferencia está en que el sistema binario es el empleado por los microprocesadores y éstos, solo son capaces de sumar y restar.

Hay que tener en cuenta que la suma es la base de las operaciones aritméticas. La resta se puede sustituir por la suma empleando números negativos, el producto es una secesión de sumas, la división es una sucesión de restas, etc.

2.1.- OBJETIVOS

Mostrar algunos de los circuitos lógicos que realizan operaciones aritméticas básicas (suma, resta, multiplicación) con números binarios. Para ello se simularán distintos diseños realizados con el Editor Gráfico del software Quartus II.

2.2.- MATERIAL

- Ordenador personal con el software QUARTUS II® de Altera®.
- Manual introductorio al software.
- Guión de prácticas.

2.3.- DESARROLLO DE LA PRÁCTICA

2.3.1.- SUMADORES BINARIOS

Los circuitos sumadores parten de un circuito sencillo llamado semisumador. Este circuito es capaz de sumar un bit más otro bit pero no es capaz de tener en cuenta un acarreo anterior. Para corregir esta falta surge el sumador total que sí es capaz de emplear el acarreo anterior. Partiendo de un sumador total y mediante la combinación del número necesario de los mismos es posible realizar sumas de más de un bit.

Las reglas para la suma binaria de dos bits son las siguientes:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0 \rightarrow$ Acarreo o carry = 1

La suma de dos números binarios de longitud mayor a un bit se realiza por parejas de bits. Hay que tener en cuenta la situación de sumar 1+1 donde se lleva 1 de acarreo. Este dato se debe incluir al realizar la suma de la pareja siguiente.

En esta práctica se van a realizar tres diseños, donde el primero de ellos se utilizará en el segundo y éste en el tercero. Por esto, es necesario crear un símbolo para cada uno de los diseños, tal y como se explica en el punto 3.1.1.

2.3.1.1.- Semisumador

El semisumador es un circuito lógico que no es capaz de añadir a la suma un carry anterior. Este circuito suma dos bits, obteniendo a la salida la suma (S) de dichos bits y el posible acarreo (C).

Se comenzará este diseño definiendo un nuevo proyecto y abriendo un nuevo documento en el Editor Gráfico, diseñando el circuito semisumador mostrado en la Figura 15:

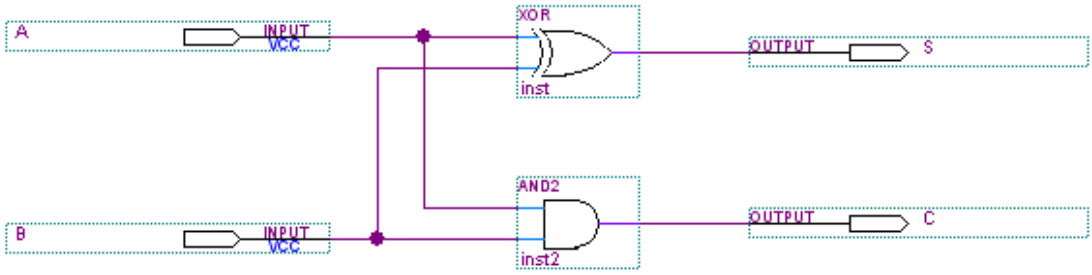


Figura 15. Circuito lógico del semisumador

En este diseño se emplean el símbolo XOR y el AND2 (puerta AND de dos entradas), los cuales están disponibles en la librería 'primitives'.

Una vez realizado el circuito, se grabará el diseño y se compilará para ver si hay posibles errores, tal y como se hizo en la anterior práctica. A continuación, se llevará a cabo la simulación para ver si los resultados son los esperados. Si no lo son, se debe tener en cuenta que el fallo debe estar en el diseño.

Se comprobarán los resultados obtenidos conforme a la tabla de la verdad del semisumador que se muestra a continuación:

B1	A1	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabla 1

Además, una vez compilado el diseño se procederá a la creación de su correspondiente símbolo, para poder ser utilizado en los siguientes diseños, tal y como se explica a continuación.

La creación de un símbolo bien puede ser en base a un diseño realizado con el Editor Gráfico o bien con el Editor de Texto. Para ello, una vez introducido, guardado y compilado el diseño, se accede al menú desplegable 'File' y se escoge la opción 'Create/Update' seguido de la opción 'Create Symbol Files for Current File', tal y como se puede ver en la siguiente ventana.

**Nota: El nombre elegido para cada uno de los diseños tiene que ser individual, sin poder repetirse, para que en el momento que interese crear un nuevo símbolo para un diseño no se tenga ningún problema. Asimismo, no se debe nombrar a los diseños con nombres ya ocupados por símbolos de las librerías, ni utilizar la letra 'ñ' en los mismos.*

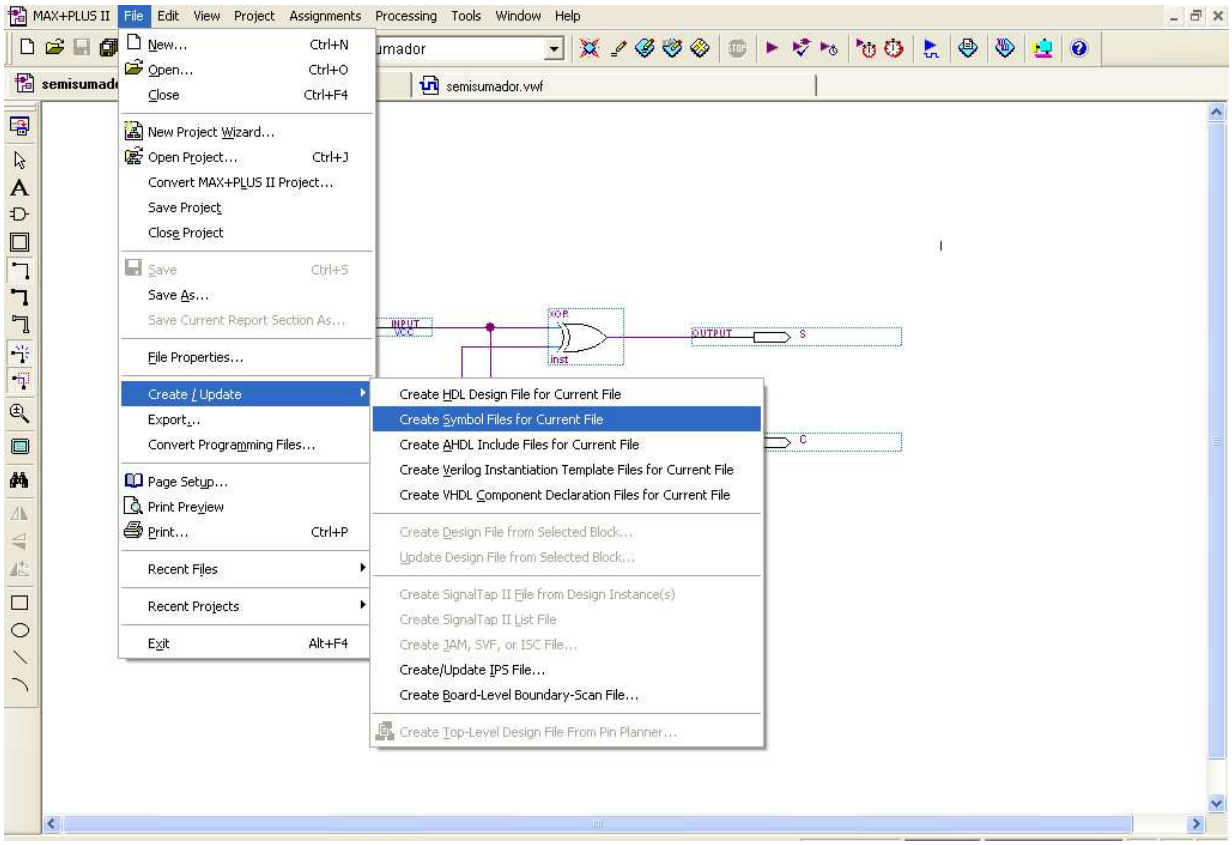


Figura 16. Acceso a la creación de símbolos

De esta manera se generará automáticamente un símbolo con las entradas y salidas del diseño, el cual se comportará de la misma forma, y tendrá el nombre del diseño con extensión '.bsf'. Este nuevo símbolo podrá ser utilizado en posteriores diseños de igual manera que los símbolos que ya existían en las librerías. Por ello, para la inclusión de este símbolo basta con acceder a la opción 'Insert Symbol' que aparece en el menú desplegable al pulsar con el botón derecho sobra la pantalla del Editor Gráfico.

- Importante: Para usar el símbolo creado en otro diseño es fundamental que éste se encuentre dentro del mismo proyecto. Si se crea otro proyecto se deberá incluir el archivo de diseño correspondiente al símbolo a la hora de definir este nuevo proyecto (por ejemplo, archivo con extensión '.bdf' correspondiente al Editor Gráfico).

El símbolo generado es el que se muestra en la Figura 17. Es posible cambiar las propiedades del símbolo (tamaño, nombres de las entradas y salidas, etc.) desde el Editor de Símbolos. Para profundizar en ello se puede consultar el manual introductorio al software.

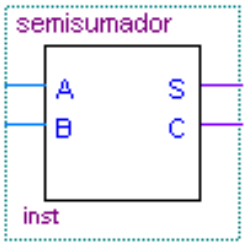
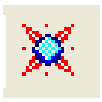


Figura 17. Símbolo del semisumador

Una forma de ver si el símbolo está creado o no, es pulsando sobre el símbolo de la derecha denominado 'Project Navigator' que representa jerárquicamente el diseño.



2.3.1.2.- Sumador completo

El semisumador sólo sirve para sumar dos bits. La suma de números binarios de más de un bit no se puede hacer sólo con semisumadores, ya que hay que tener en cuenta el posible acarreo de la pareja anterior. Para solucionar esto será necesario otro circuito lógico, el sumador completo.

El sumador completo realiza la suma de dos bits (A y B), teniendo en cuenta el acarreo (Ci) de la pareja anterior. A la salida se obtiene la suma (S) y el posible acarreo (Co).

Se puede realizar un sumador completo mediante dos semisumadores y una puerta OR. El diseño del circuito lógico es el que se muestra en la Figura 18.

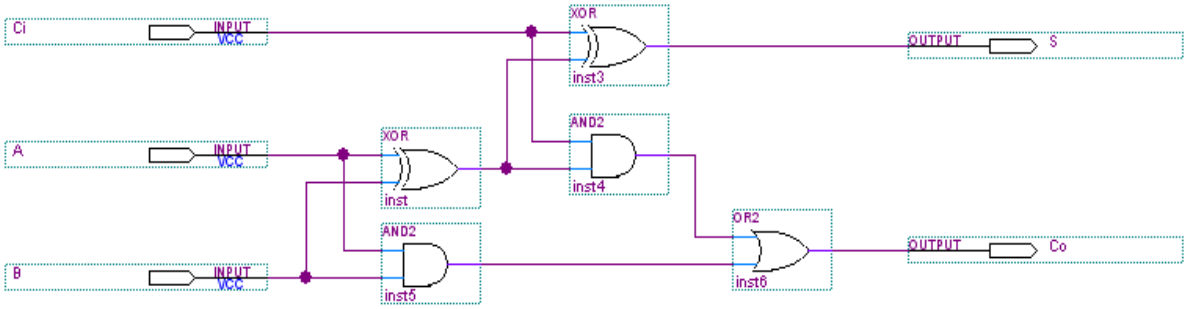


Figura 18. Circuito lógico del sumador completo

De cara a la implementación de circuitos más complejos, resulta más sencillo utilizar símbolos previamente creados. En este caso es posible utilizar el símbolo del semisumador generado en el apartado anterior. Para poder utilizar este símbolo, crearemos un nuevo archivo en el Editor Gráfico dentro de un nuevo proyecto (recordar guardar cada proyecto en un nuevo directorio) en el que ha sido incluido el semisumador. Utilizando el símbolo del semisumador el diseño queda como se muestra en la Figura 19.

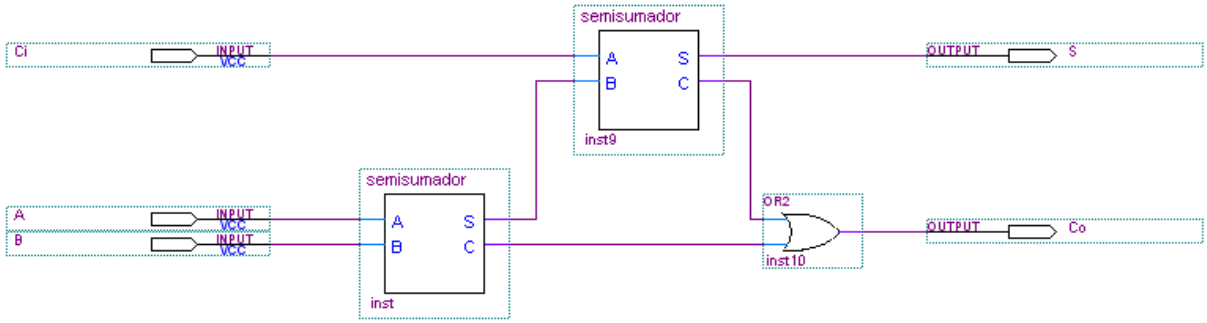


Figura 19. Sumador completo mediante la inclusión del símbolo semisumador

Una vez compilado, se simulará el sumador completo y se comprobarán que los resultados coinciden con los de su tabla de la verdad, mostrada a continuación:

Ci	B1	A1	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabla 2

En este caso, también se creará el símbolo que representará al sumador completo, que será usado en el siguiente apartado.

2.3.1.3.- Sumador binario paralelo de números de cuatro bits con entrada de acarreo

La suma de dos números de N bits se puede realizar con N-1 sumadores completos y un semisumador (para la pareja menos significativa) conectados en cascada. Sin embargo, si queremos un sumador para dos números de N bits con entrada de acarreo previo, son necesarios N sumadores completos.

```
                        C_in
      A4    A3    A2    A1
+
      B4    B3    B2    B1
_____________________________
C_out S4    S3    S2    S1
```

El circuito lógico que representa a este sumador se ve en la Figura 20.

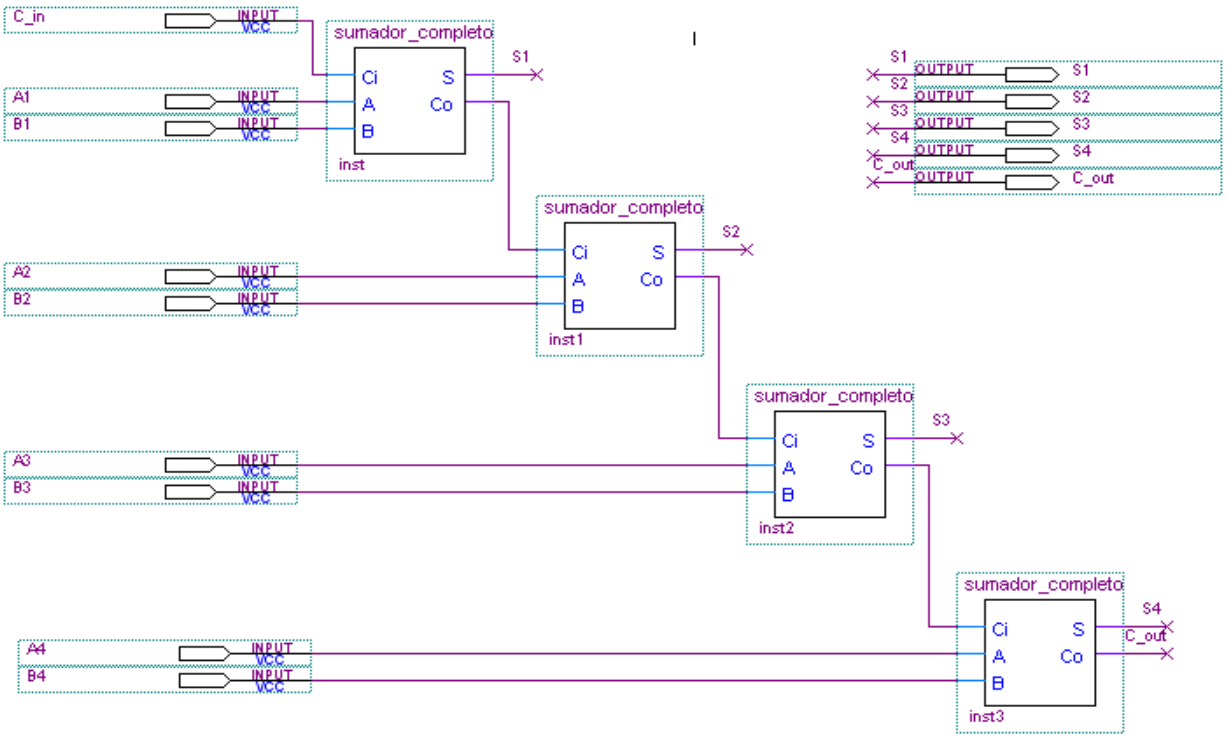


Figura 20. Circuito lógico del sumador paralelo de cuatro bits y entrada de acarreo

Como se puede observar en la Figura anterior, existen dos maneras de realizar las conexiones. En primer lugar uniendo los distintos elementos mediante hilos y en segundo lugar dando el mismo nombre al origen y destino de la conexión.

Se realizará este nuevo diseño generando un nuevo proyecto, donde se incluirán los archivos generados anteriormente (semisumador y sumador completo) para así poder incluir sus respectivos símbolos sin problemas.

Nuevamente se procederá a compilar el diseño y se utilizará el simulador para comprobar las siguientes operaciones:

a) C_in = 0; B = 0011; A = 0001;
b) C_in = 0; B = 1011; A = 0110;
c) C_in = 1; B = 0010; A = 0101;
d) C_in = 1; B = 0111; A = 1100;

Para realizar la simulación de las operaciones que se piden, es necesario introducir a mano los valores de las señales, utilizando para ello los iconos ‘0’ y ‘1’ de la barra de herramientas situada en el margen izquierdo del Editor de Señales, tal y como se ve en la Figura 21.

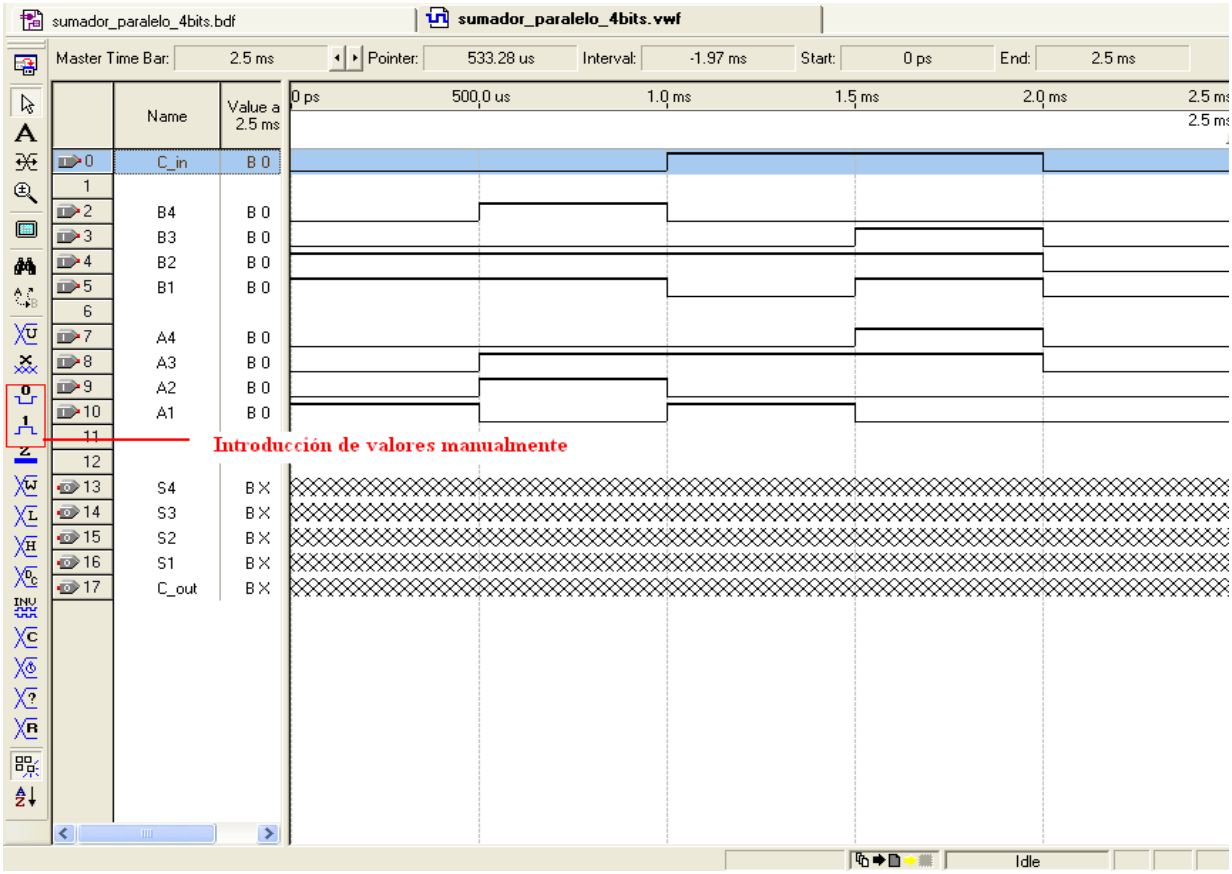


Figura 21. Introducción de las señales para las operaciones

A la hora de introducir valores manualmente es conveniente activar la opción 'Snap to Grid' situada en el menú desplegable 'View'. De esta manera se consigue forzar el valor introducido al tiempo del periodo.

También se deberá crear un símbolo de este diseño, el cual hará falta posteriormente.

En cuanto a los restadores, estos se realizarán mediante sumadores empleando bien el convenio CA1 o bien el convenio CA2.

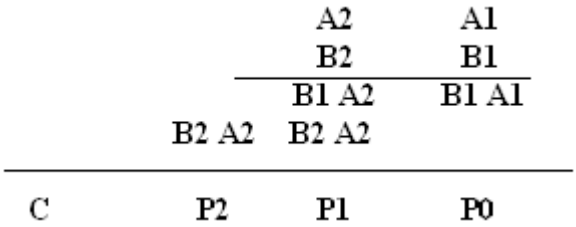
$$\text{RESTA} \rightarrow B - A = B + (-A) = \left\{ \begin{matrix} B + CA1(A); ConvenioCA1 \\ B + CA2(A); ConvenioCA2 \end{matrix} \right\}$$

4.- EJERCICIOS DE DISEÑO

4.1- Multiplicador binario paralelo de dos bits

Diseñar un circuito combinacional con el que se pueda obtener el producto (multiplicación) de dos números binarios de dos bits cada uno (A2 A1 y B2 B1) presentados en paralelo. Se utilizarán puertas AND y semisumadores, en concreto, seis puertas AND2 y dos puertas XOR.

La operación resultante para el diseño sería la siguiente:



Dibujar el circuito con el Editor Gráfico y comprobar su correcto funcionamiento.

4.2- Sumador / Restador (CA2)

Diseñar un Sumador / Restador para números binarios (A y B) de cuatro bits utilizando el complemento a dos (CA2) para la resta. El circuito tendrá una entrada de control C para seleccionar la suma o la resta (C=0 suma y C=1 resta). Para la realización del circuito se utilizará el símbolo del 'sumador binario paralelo de números de cuatro bits con entrada de acarreo' y cuatro puertas XOR.

Dibujar el circuito con el Editor Gráfico y comprobar su correcto funcionamiento con el simulador, realizando las siguientes operaciones:

g) 4 + 2
h) 3 – 1
i) 1 – 5

4.3- Comparadores binarios

4.3.1- Comparador binario de 2 bits

Realizar un circuito comparador que nos indique si un bit es mayor, igual o menor que otro.

Figura 22. Comparador binario de dos bits

Simular y comprobar su correcto funcionamiento. Crear un símbolo para dicho circuito, ya que será utilizado en el siguiente apartado.

4.3.2- Comparador binario de palabras de 2 bits

Realizar, a partir del diseño del apartado anterior, un comparador de palabras de dos bits, donde nos diga si una de ellas el mayor, igual o menor que la otra.

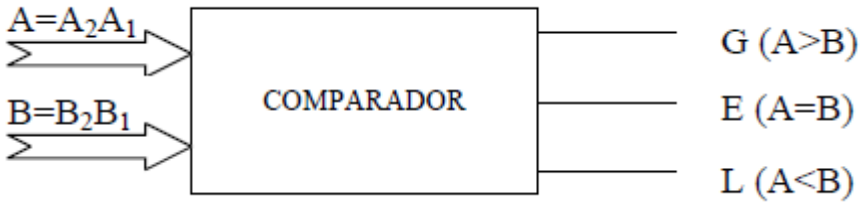


Figura 23. Modelo del comparador de palabras de dos bits

Para números de más de un bit, lo que se hace es ir comparando los bits de igual peso, desde el más significativo al menos significativo, hasta que se encuentre uno que sea mayor que el otro.

Las combinaciones posibles según las diferentes señales de entrada son $2^4 = 16$, pero para facilitar la situación sólo se estudiarán los siguientes casos:

A = 00	B = 00
A = 01	B = 11
A = 10	B = 00
A = 11	B = 11
A = 00	B = 01
A = 11	B = 10

Las señales a introducir en el Editor de Señales para realizar la simulación serán introducidas a mano.

RESOLUCIÓN PRÁCTICA 2

A continuación se presentan las soluciones de los diferentes ejercicios que componen esta práctica.

PRIMERA PARTE: PUERTAS LÓGICAS.

1.3.1.- PUERTAS NOT, AND Y OR

El diseño a introducir en el siguiente:

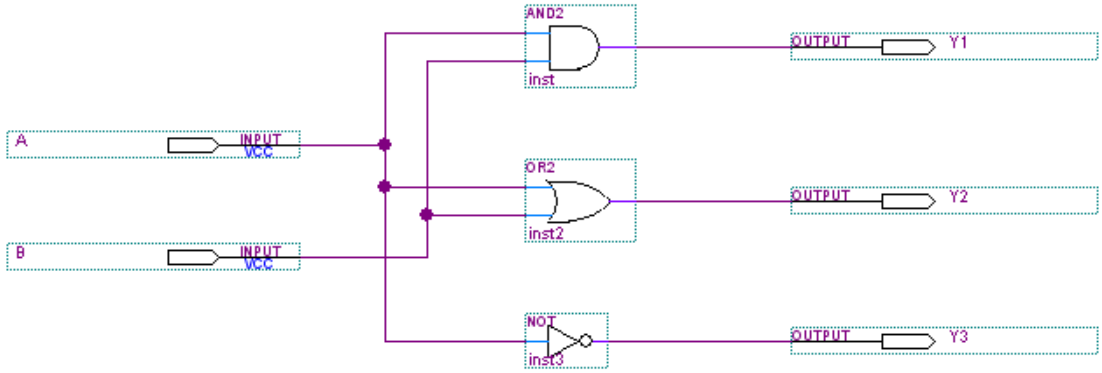


Figura 1. Circuito Apartado 1.3.1

Tras realizar la compilación del diseño, se procede a la simulación, obteniendo los siguientes resultados:

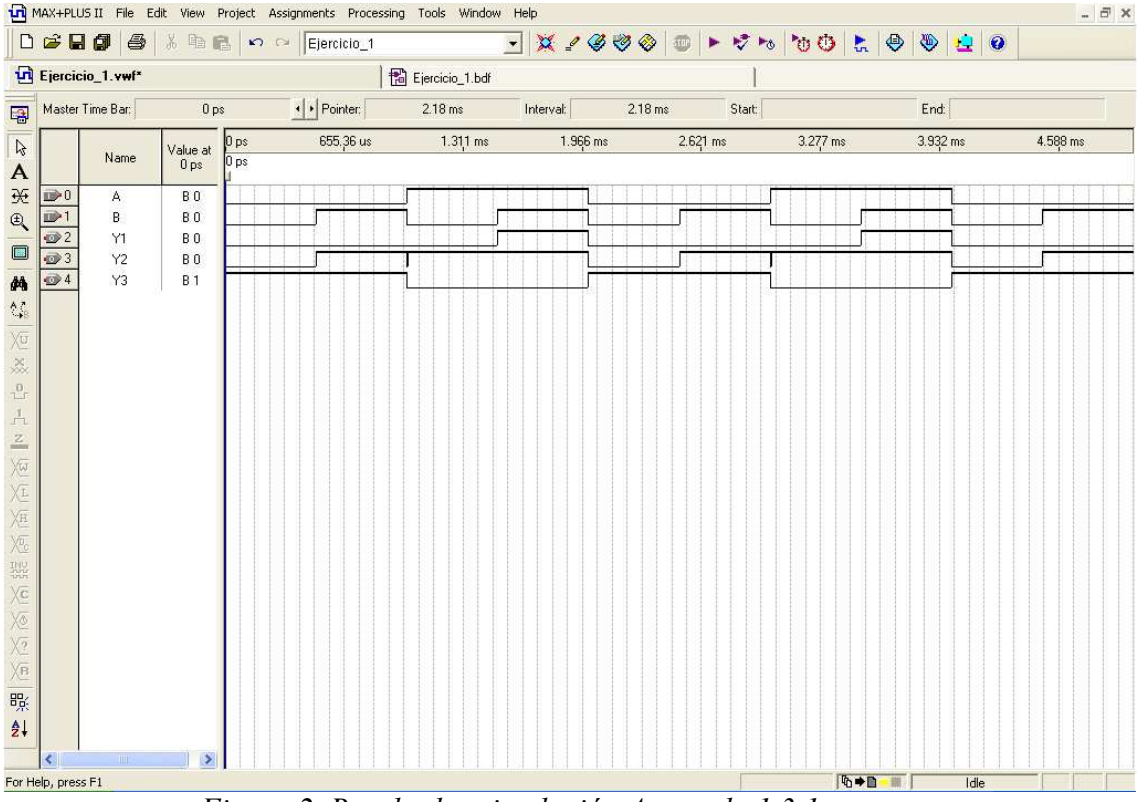


Figura 2. Resultados simulación Apartado 1.3.1.

A partir de los resultados de la simulación, se obtiene la tabla de verdad del circuito:

A	B	Y1	Y2	Y3
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Tabla 1

Si analizamos la tabla de verdad vemos que efectivamente, los resultados son los esperados:

- Y1 se corresponde con el producto (puerta AND) $A \cdot B$
- Y2 se corresponde con la suma (puerta OR) $A + B$
- Y3 se corresponde con $\overline{A}$ (puerta NOT)

Observaciones:

En la simulación se puede ver un pico de nivel bajo en la salida 'Y2', que justamente coincide con los flancos de subida y bajada de las señales 'A' y 'B' respectivamente.

La señal 'Y2' corresponde a la salida de la puerta OR y este pico es debido a que hay un instante en el que la señal 'B' ha cambiado a nivel bajo antes de que la señal 'A' haya cambiado a nivel alto, durante la transición de estados lógicos.

Este efecto se puede apreciar con detalle en la siguiente Figura.

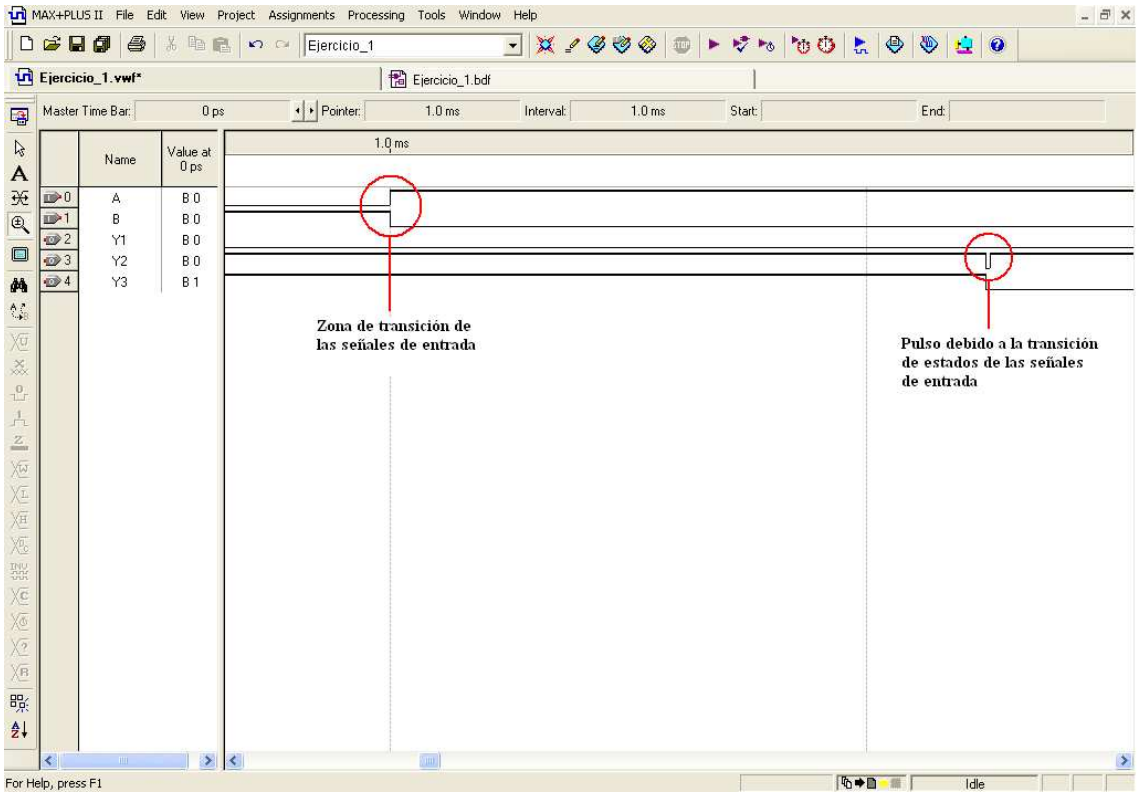


Figura 3. Efectos de la transición de estados

Como se puede apreciar en la Figura 3, el pulso aparece con el retardo que introduce la puerta OR.

1.3.2.- PUERTAS NAND, NOR Y XOR

El diseño a introducir para este ejercicio es el siguiente:

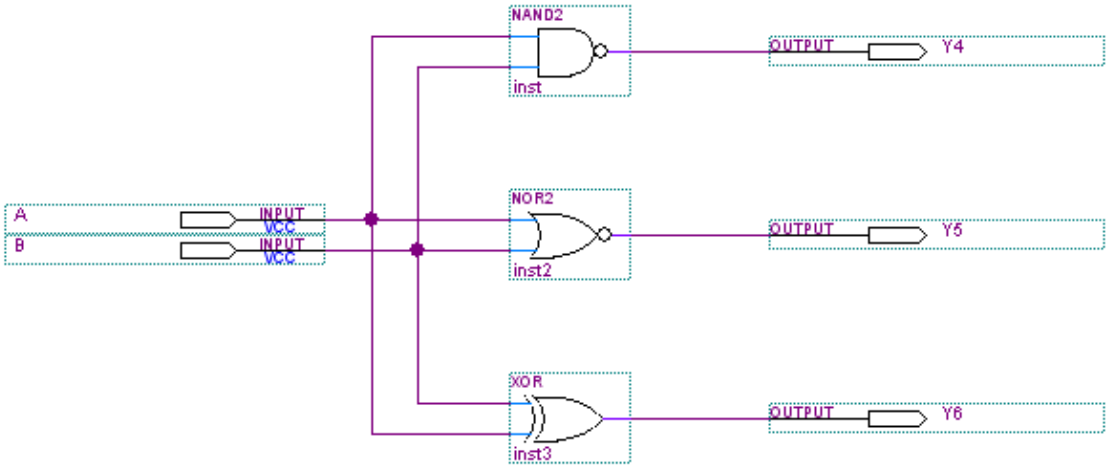


Figura 4. Circuito Ejercicio 1.3.2

Tras el proceso de simulación se obtienen los siguientes resultados:

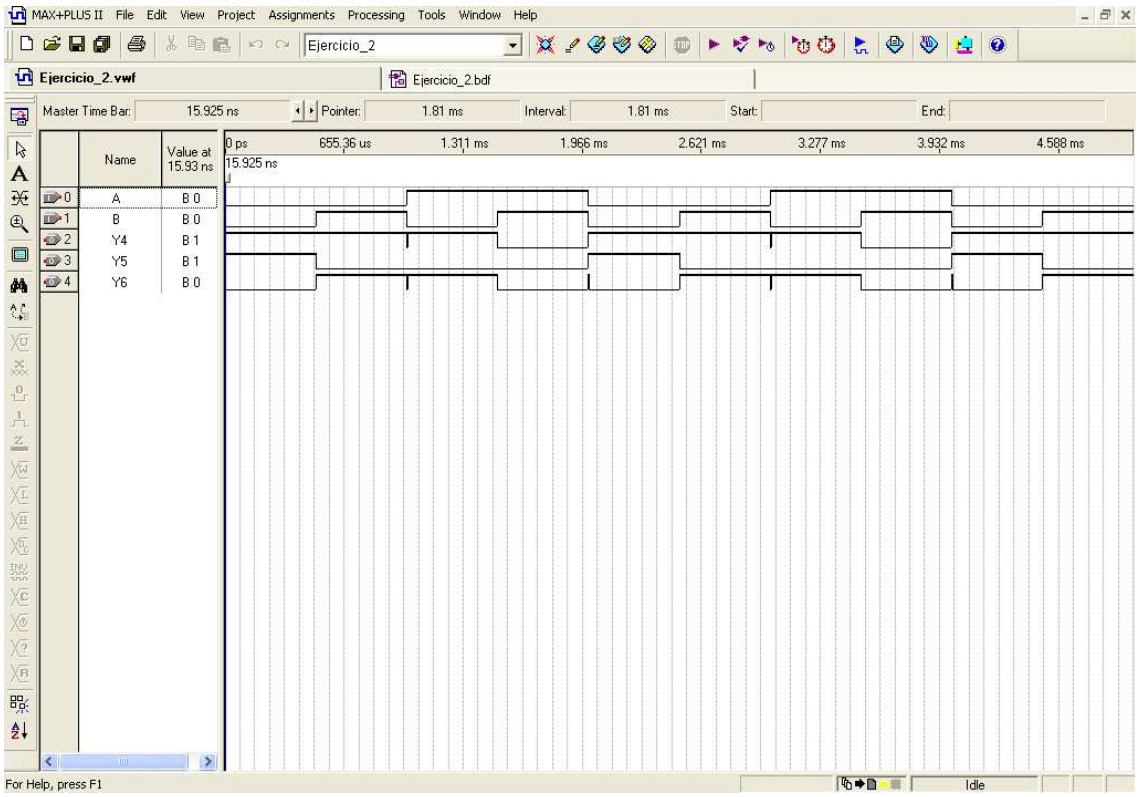


Figura 5. Resultado simulación Ejercicio 1.3.2

A partir de la simulación realizada se obtiene la siguiente tabla de la verdad:

A	B	Y1	Y2	Y3
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

Tabla 2

Viendo las combinaciones de la tabla de la verdad se observa que:

- Y4 corresponde al producto $\overline{A \bullet B}$ (puerta NAND)
- Y5 corresponde a la suma $\overline{A+B}$ (puerta NOR)
- Y6 corresponde a la función $A \oplus B$ (puerta XOR)

Observaciones:

En el resultado de esta simulación, el motivo de los pulsos que aparecen en las señales de salida 'Y4' e 'Y6' es el mismo que el comentado en el apartado anterior.

SEGUNDA PARTE: CIRCUITOS ARITMÉTICOS.

2.3.1.- SUMADORES BINARIOS

2.3.1.1.- Semisumador

El circuito a introducir es el que se muestra en la Figura 6.

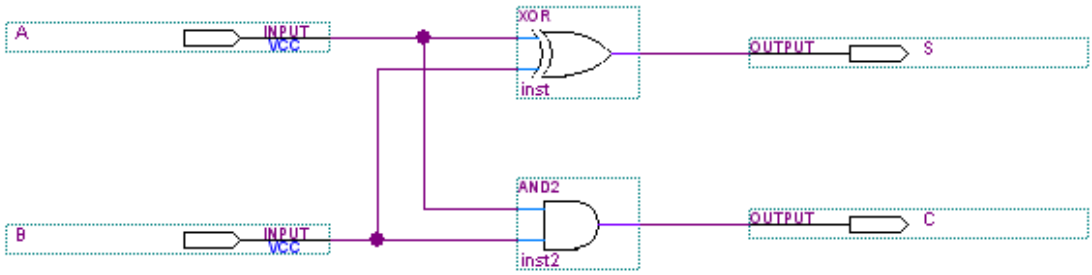


Figura 6. Circuito semisumador

Tras el proceso de simulación se obtienen los siguientes resultados:

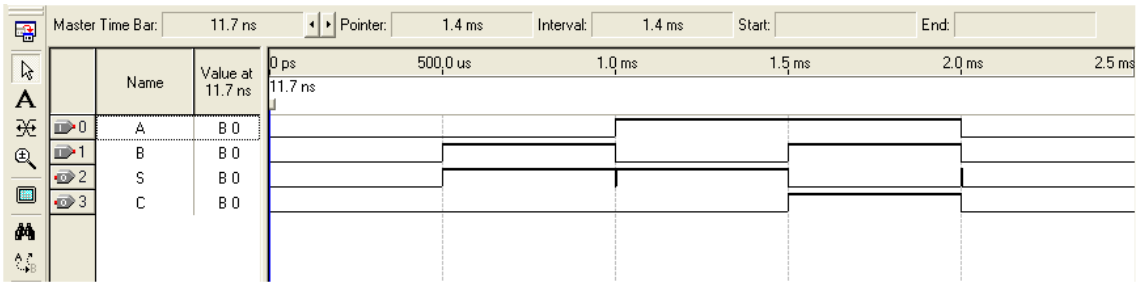


Figura 7. Resultados del semisumador

A partir de la simulación realizada se obtiene la siguiente tabla de la verdad, que coincide con la expuesta en el guión, correspondiente al semisumador:

B1	A1	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabla 3

2.3.1.2.- Sumador completo

El circuito lógico del sumador completo es el siguiente:

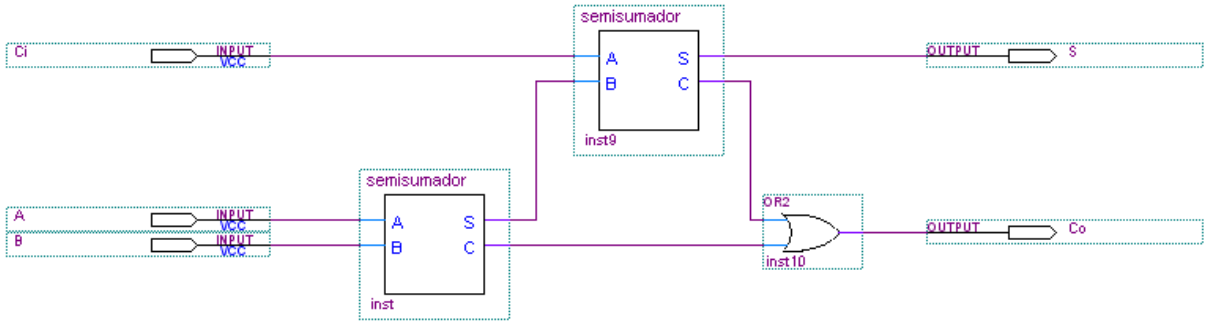


Figura 8. Circuito sumador completo

Los resultados obtenidos se muestran a continuación, en la Figura 9:

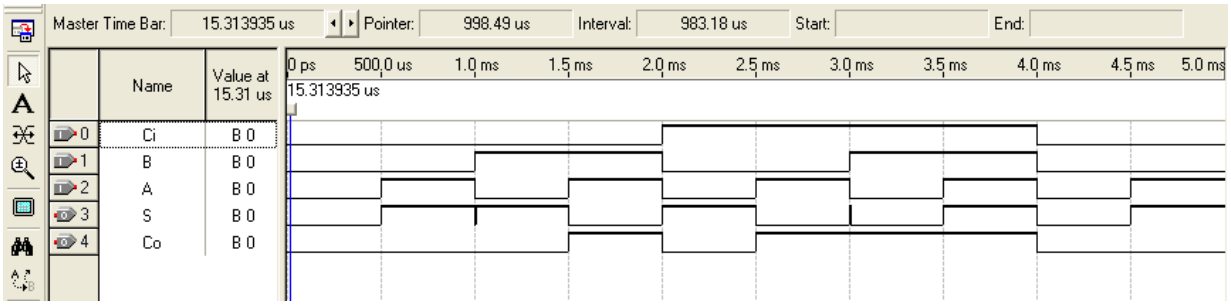


Figura 9. Resultados del sumador completo

Tras la simulación se obtiene la tabla de la verdad correspondiente a un sumador de dos bits con entrada de acarreo:

Ci	B1	A1	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabla 4

Como se puede comprobar, la tabla se corresponde con la del sumador completo.

2.3.1.3.- Sumador binario paralelo de números de cuatro bits con entrada de acarreo.

El diseño a introducir es el mostrado en la Figura 10.

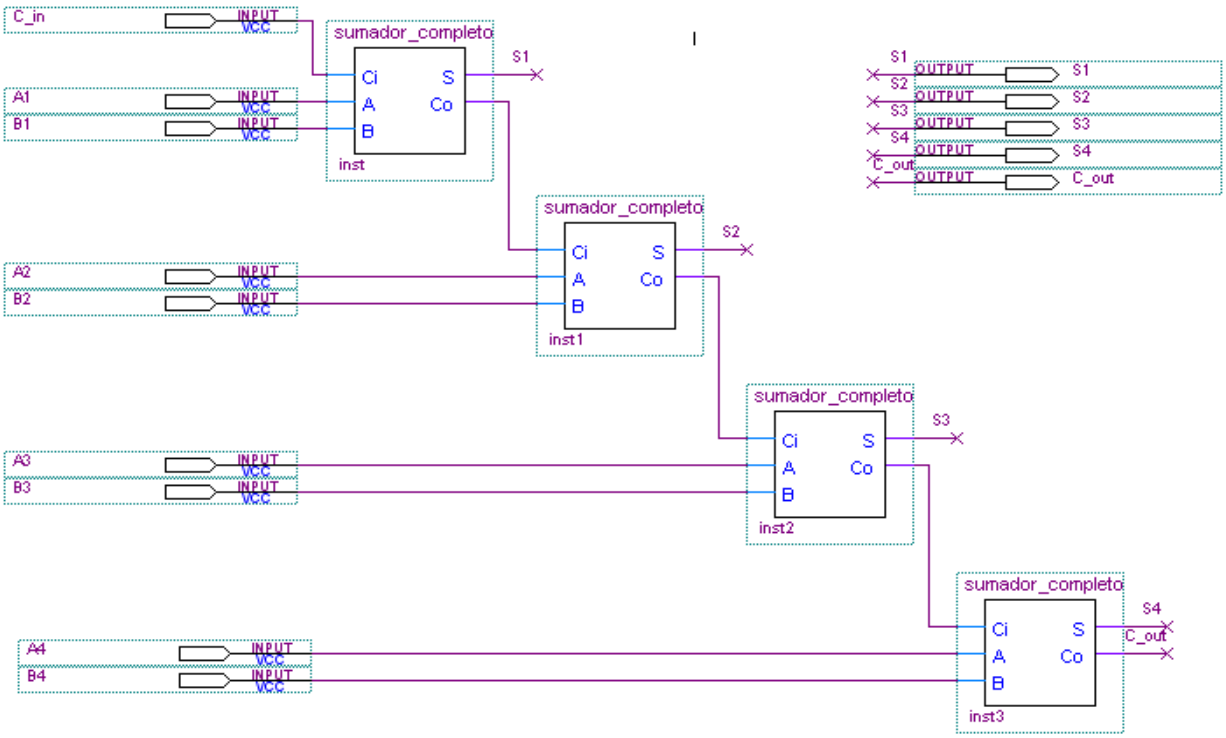


Figura 10. Circuito sumador paralelo de cuatro bits y entrada de acarreo

La suma que realiza este circuito es de la siguiente manera:

				C_in
	A4	A3	A2	A1
+				
	B4	B3	B2	B1
C_out	S4	S3	S2	S1

A continuación se muestran los resultados de los diferentes casos que se exigen como prueba:

e)	C_in	0
	B	0011
	A	0001
C_out=0		0100

f)	C_in	0
	B	1011
	A	0110
C_out=1		0001

g)	C_in	1
	B	0010
	A	0101
C_out=0		1000

h)	C_in	1
	B	0111
	A	1100
C_out=1		0100

A partir de estas operaciones, la simulación queda de la siguiente manera:

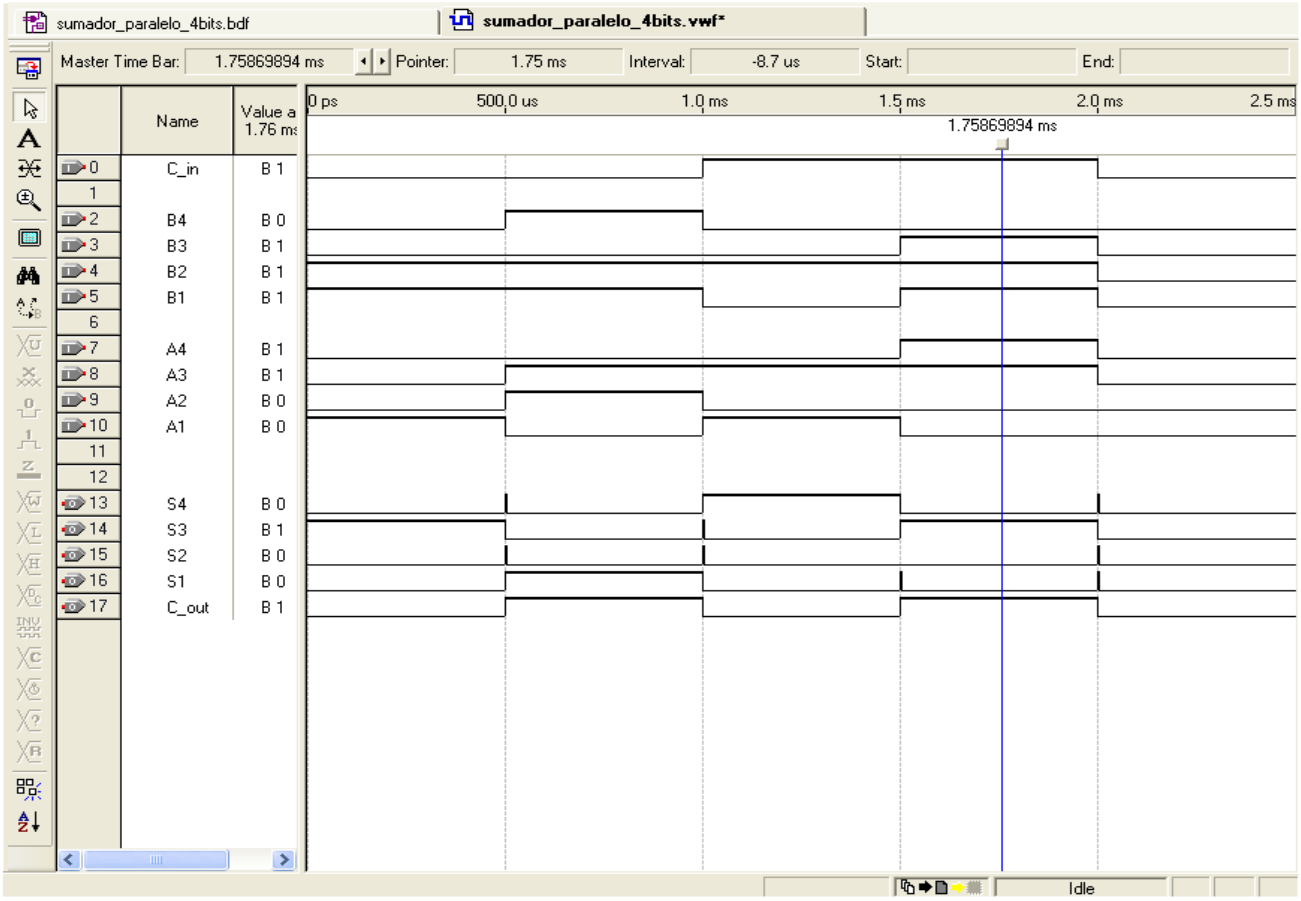


Figura 11. Resultados del sumador paralelo de 4 bits

Como se puede comprobar, los resultados son correctos.

4.- EJERCICIOS DE DISEÑO

4.1- Multiplicador binario paralelo de dos bits

El producto de dos números binarios, A y B se corresponde con:

		A2	A1
		B2	B1
		B1 A2	B1 A1
	B2 A2	B2 A2	
C	P2	P1	P0

De esto se deduce lo siguiente:

P0 = B1 A1
P1 = B1 A2 + B2 A1
P2 = B2 A2 (más un posible acarreo procedente de P1)
C = posible acarreo final

A partir de las expresiones anteriores, el circuito estará formado por cuatro puertas AND para realizar los productos de las parejas de bits y dos semisumadores, donde el segundo estará conectado con la salida de acarreo del primero.

El circuito lógico correspondiente al multiplicador binario queda de la siguiente manera:

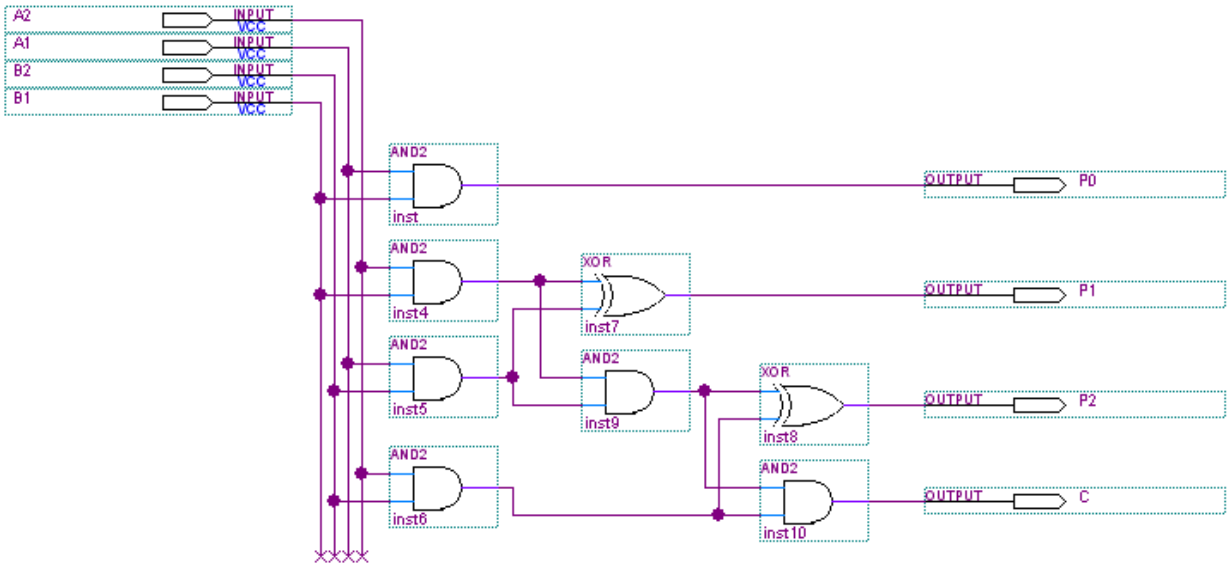


Figura 12. Circuito del multiplicador binario paralelo de dos bits

Como prueba se han tomado los siguientes ejemplos:

a)	B	1 0
	A	0 1
C=0		0 1 0

b)	B	1 1
	A	1 1
C=1		0 0 1

La simulación de estos ejemplos se muestra en la Figura 13.

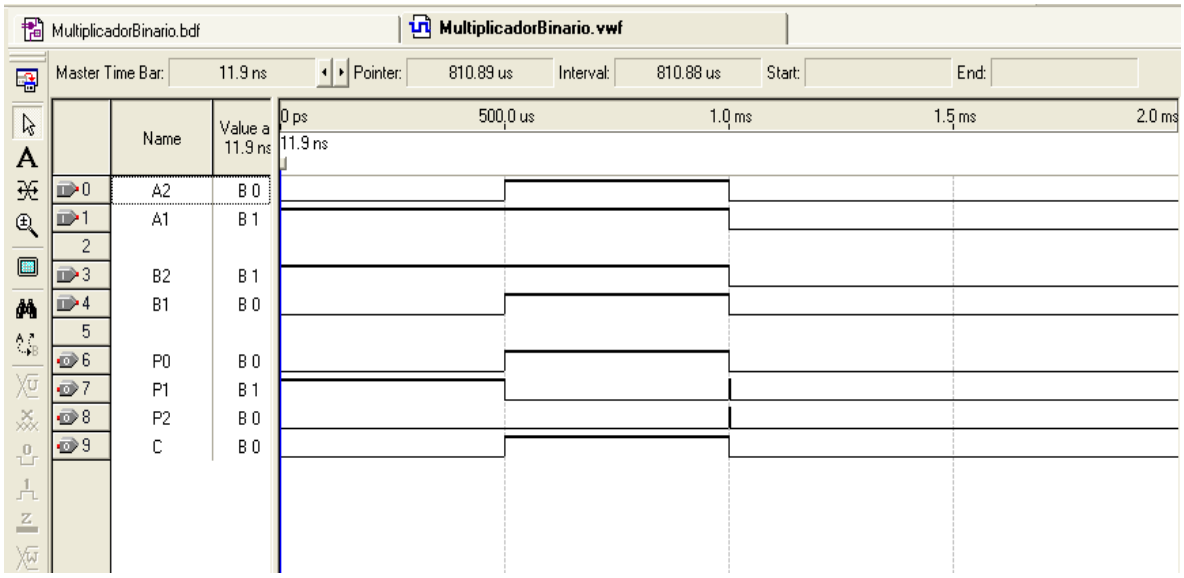


Figura 13. Resultados del multiplicador binario paralelo de dos bits

Como se puede observar, los resultados coinciden con lo esperado.

4.2- Sumador / Restador en CA2

El presente diseño está basado en el circuito 'sumador binario paralelo de 4 bits' implementado en el apartado 2.3.1.3, por lo tanto, en principio sólo capaz de realizar sumas, pero con la correspondiente modificación que también permite realizar la operación de restar.

Para sumar con este circuito, la entrada de control 'C' deberá estar a nivel bajo ('0'), de esta manera, esta entrada no añadirá nada a la suma y el conjunto de puertas XOR no alterarán el valor del número introducido en 'B'. Por lo tanto, la suma se realizará como si el circuito sumador binario paralelo de 4 bits estuviese solo.

En el caso de la resta la entrada de control 'C' deberá ponerse a '1'. La resta es igual a la suma del minuendo más el opuesto del sustraendo. En este caso, para cambiar el signo del sustraendo se deberá hacer el CA2 del mismo, mientras que el minuendo permanecerá sin cambios.

En este caso, el posible acarreo de salida se ignora. Este acarreo nos sería útil en el caso de que quisiéramos realizar la resta con el sustraendo en CA1.

El proceso para obtener un número en CA2 es, en primer lugar, obtener dicho número en CA1 y posteriormente sumarle '1'. Por tanto, mediante el bloque compuesto por cuatro puertas XOR es posible invertir el valor de los bits, es decir, obtener el CA1 del sustraendo y conectar la entrada C, que en el caso de la resta tiene valor '1', a la entrada de acarreo del

circuito ‘sumador binario paralelo de 4 bits’. Esto permite sumar una unidad al sustraendo de modo que se obtiene la entrada ‘B’ en CA2.

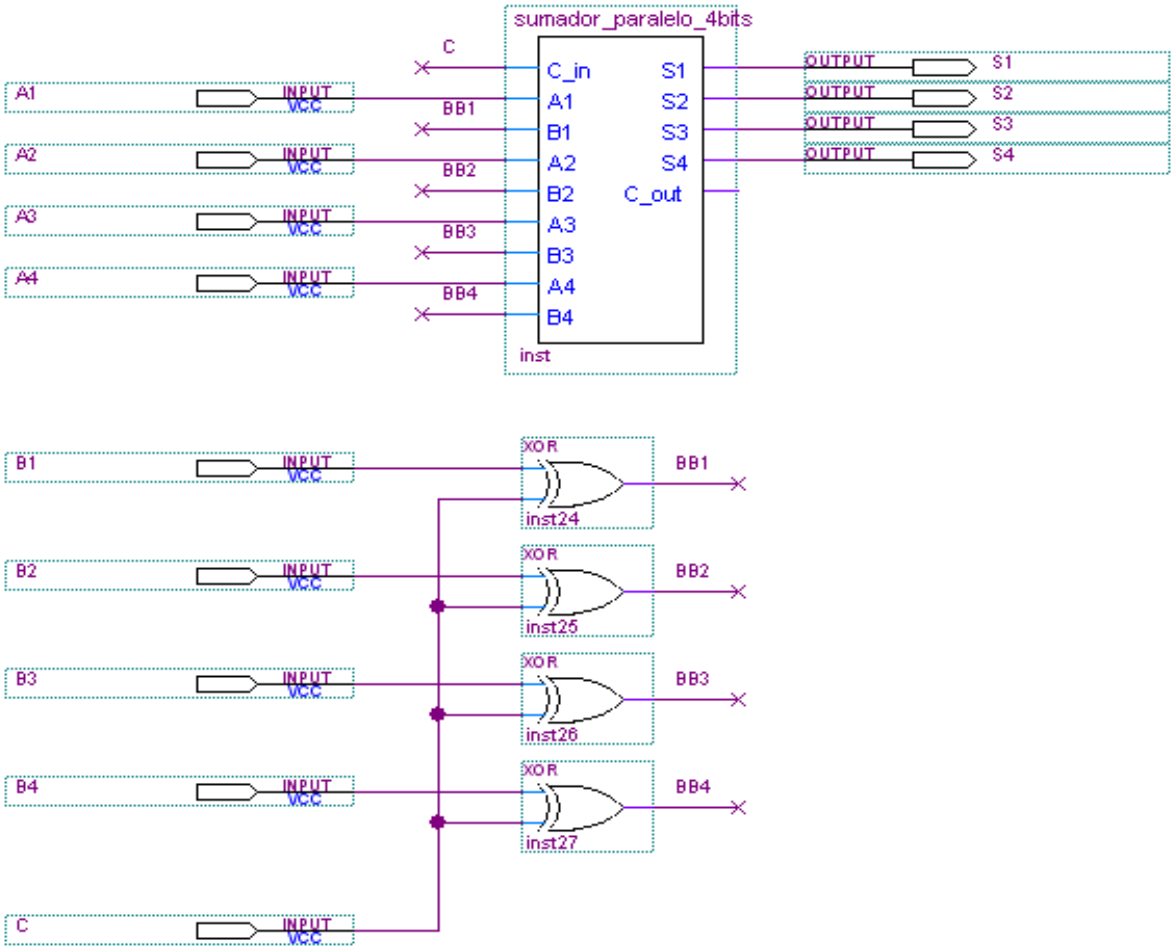


Figura 14. Circuito sumador / restador en CA2

Se comprobará el correcto funcionamiento del circuito mediante las siguientes operaciones:

a) 4 + 2 = 6 → 0100 + 0010 = 0110
b) 3 – 1 = 2 → 0011 + CA2(0001) = 0011 + 1111 = **1**0010 →0010
c) 1 – 5 = -4 → 0001 + CA2(0101) = 0001 + 1011 = 1100
-4 = CA2 (0100) = 1011 + 1 = 1100

A continuación se procede a realizar la simulación, introduciendo a mano los valores de las señales de los operandos.

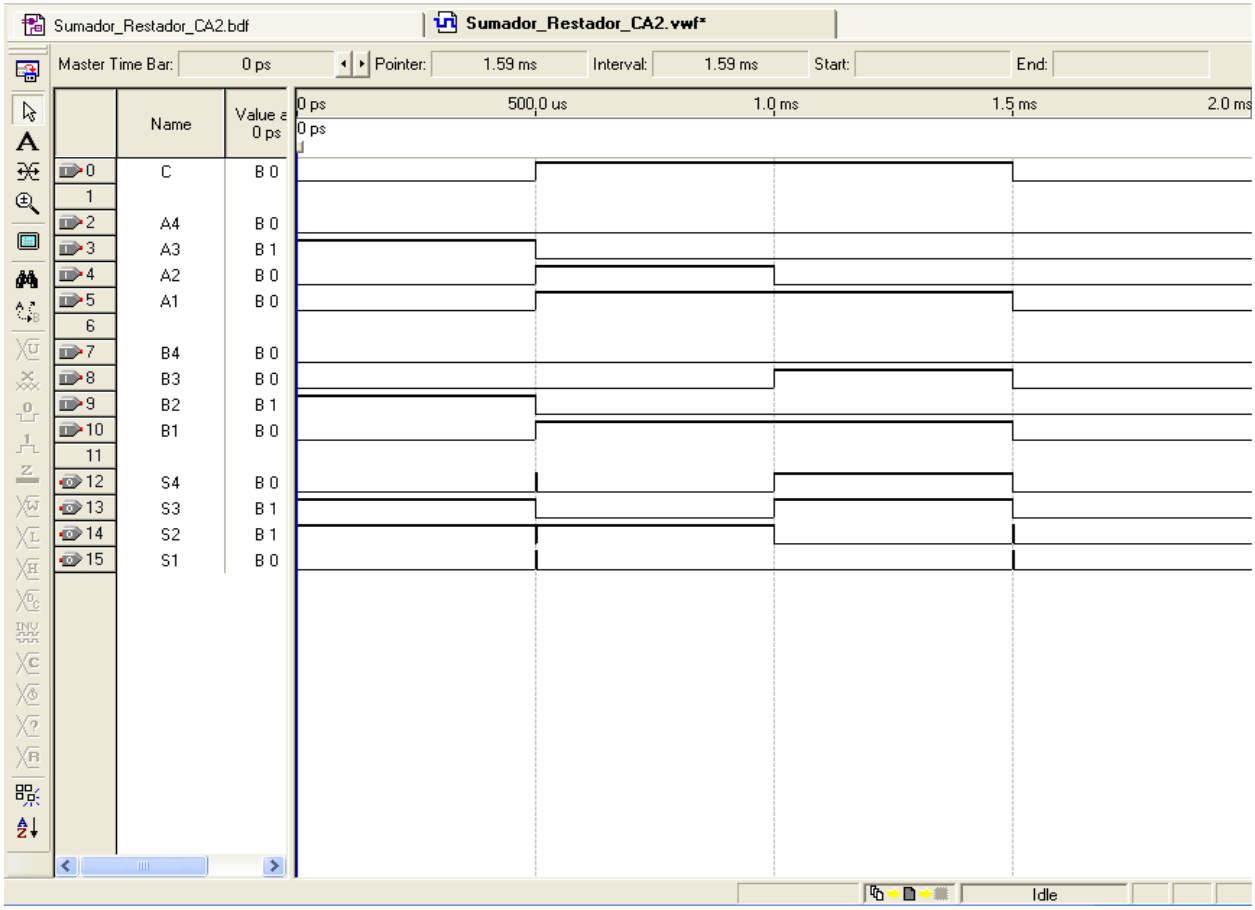


Figura 15. Resultados del circuito sumador / restador en CA2

Como se puede observar, los resultados coinciden con los obtenidos en las operaciones.

Un efecto destacable de esta simulación es que los picos que aparecen en las transiciones de las distintas operaciones son más exagerados que en los casos anteriores. Esto es debido a que la introducción de las señales se ha hecho a mano, por lo que los retardos son mayores.

4.3- Comparadores binarios

4.3.1- Comparador binario de 2 bits

La tabla de verdad que resume el comportamiento del circuito que se pretende diseñar es la siguiente:

A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Tabla 5.

A partir de esta tabla, se obtienen las funciones correspondientes a cada una de las salidas:

$$G = A \cdot \overline{B}$$
$$E = \overline{A} \cdot \overline{B} + A \cdot B = \overline{A \oplus B} = \overline{\overline{A} \cdot B + A \cdot \overline{B}} = \overline{G + L}$$
$$L = \overline{A} \cdot B$$

De esta manera, el diseño que se debe introducir en este apartado es el mostrado en la Figura 16:

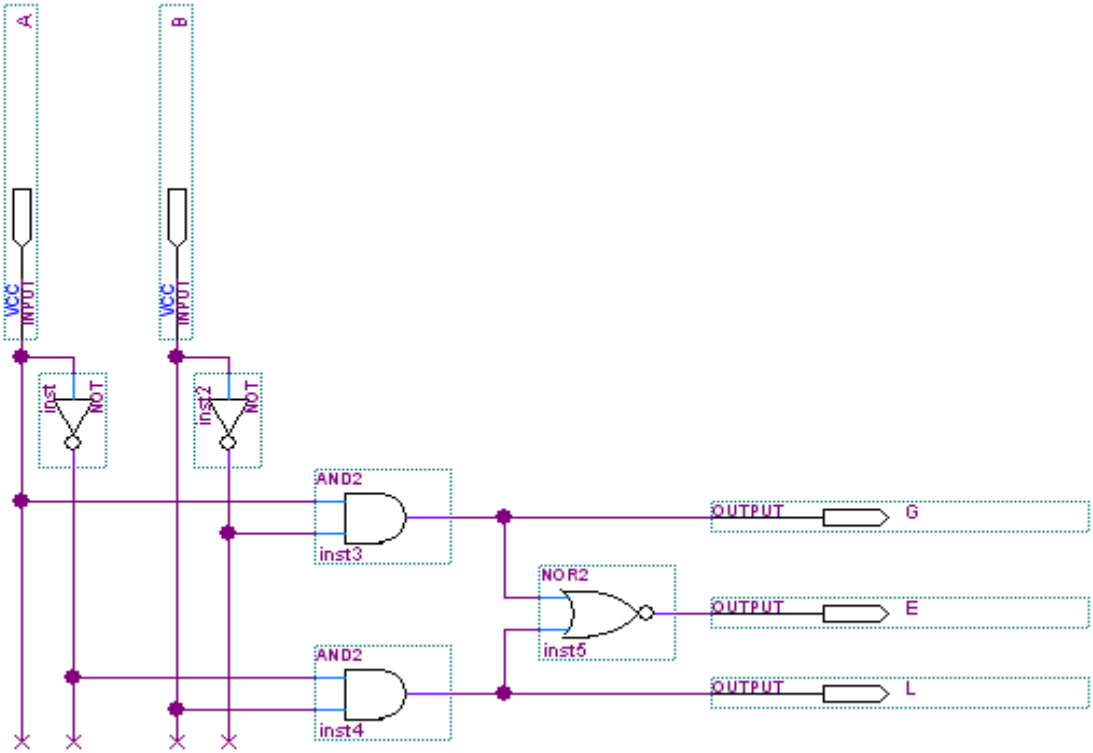


Figura 16. Circuito comparador binario de dos bits

El significado de las entradas y salidas es el siguiente:

- A, B: bits de entrada que van a ser comparados
- G: salida que indica A>B
- E : salida que indica A=B
- L : salida que indica A<B

El resultado de la simulación es el mostrado en la Figura 17:

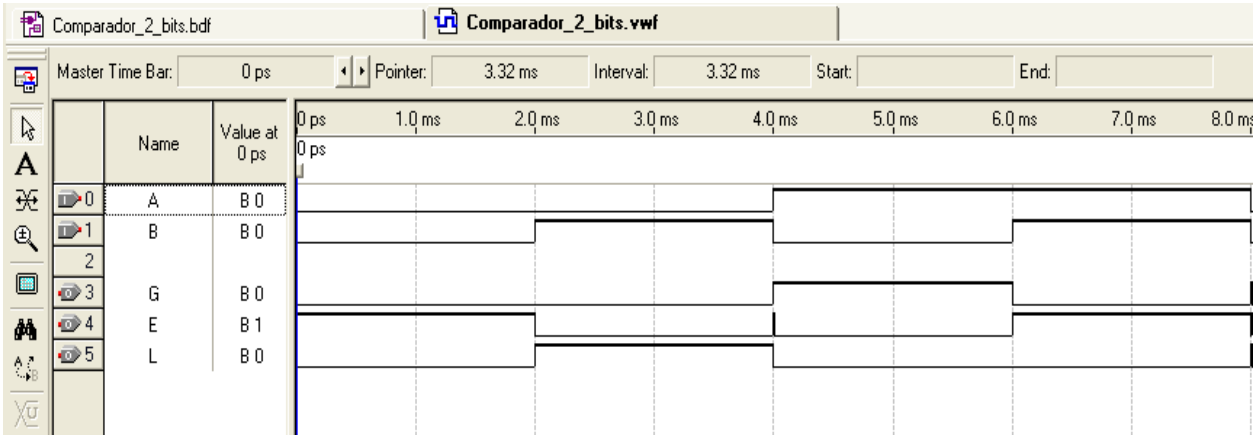


Figura 17. Resultados del comparador binario de dos bits

Las frecuencias empleadas son:
B → 250 Hz
A → 125 Hz

El símbolo creado, para ser utilizado en el siguiente ejercicio, es el siguiente:

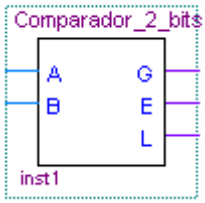


Figura 18. Símbolo del comparador de dos bits

4.3.2- Comparador binario de palabras de 2 bits

Para la realización del diseño hay que tener en cuenta los siguientes casos:

1º- Comparación del bit más significativo. En caso de desigualdad entre ambos ya estaremos en condiciones de saber los estados A>B y A<B.

2º.- En caso de igualdad en el bit de mayor peso, será el bit de menor peso el que nos de el resultado.

Esto se resume en las siguientes ecuaciones:

$$G(A>B) = 1 \text{ si } \{(A2 > B2) + ((A2 = B2) * (A1 > B1))\}$$
$$E(A=B) = 1 \text{ si } \{(A2 = B2) * (A1 = B1)\}$$
$$L(A<B) = 1 \text{ si } \{(A2 < B2) + ((A2 = B2) * (A1 < B1))\}$$

En base a las anteriores ecuaciones, se tiene que las expresiones de salida, utilizando el comparador del apartado 3.3, son las siguientes:

G = G2 + E2*G1
E = E2 * E1
L = L2 + E2*L1

El circuito resultante es el siguiente:

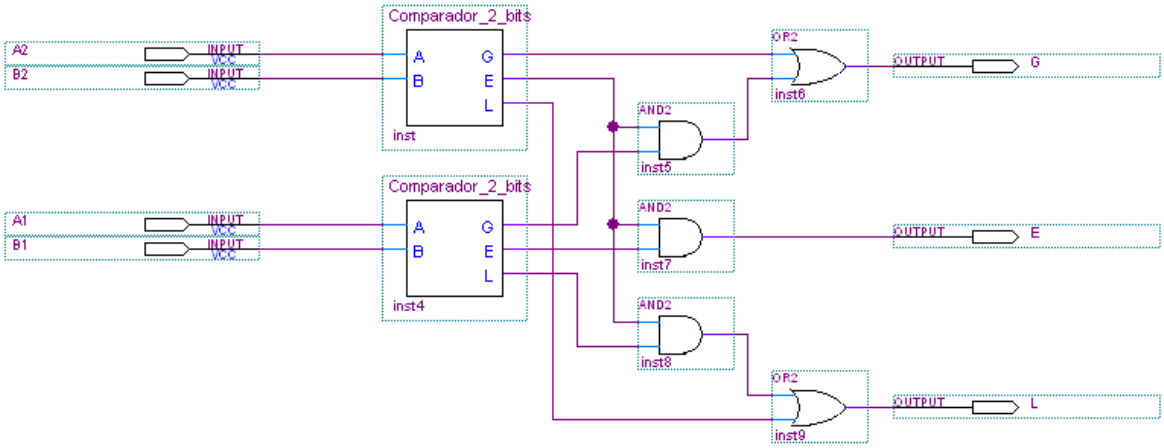


Figura 19. Circuito comparador binario de palabras de 2 bits

Para comprobar el correcto funcionamiento del diseño, se estudian los casos mostrados en la siguiente tabla, incluyendo sus respectivas salidas.

A2	A1	B2	B1	G	E	L
0	0	0	0	0	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	1	1	1	0	1	0
0	0	0	1	0	0	1
1	1	1	0	1	0	0

Tabla 6

A continuación se procede a realizar la simulación, para comprobar que los resultados coinciden con los expuestos en la *Tabla 6.*

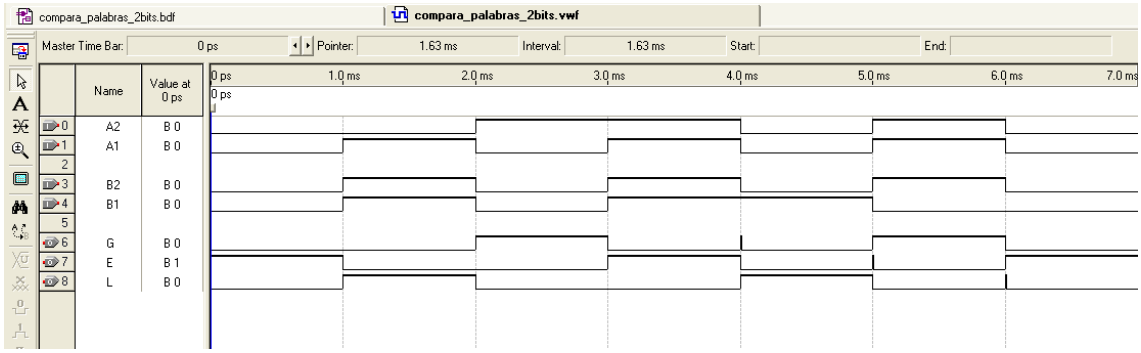


Figura 20. Resultados del comparador binario de 2 bits

Las señales de entrada para la simulación deben ser introducidas a mano.

Como se puede observar las señales de salida G, E y L coinciden con lo obtenido en la *Tabla 6.*

CIRCUITOS ELECTRÓNICOS DIGITALES

(3º Ingeniería de Telecomunicación)

Práctica 3: Biestables y Contadores

1.- OBJETIVOS

a) Conocer el funcionamiento del elemento básico de los circuitos secuenciales, el biestable, y ver alguna de sus aplicaciones, tal como los contadores.
b) Conocer el funcionamiento de un biestable comercial, el 7476, que es un circuito biestable tipo JK disparado por flanco.
c) Simular y comprender el funcionamiento de dos contadores síncronos: uno ascendente y otro descendente
d) Aprender a realizar el volcado del correspondiente diseño en la tarjeta educacional

.

2.- MATERIAL

- Ordenador personal con el software QUARTUS II® de Altera®.
- Manual introductorio al software.
- Guión de prácticas.
- Tarjeta educacional DE2 ó UP2[1].

3.- DESARROLLO DE LA PRÁCTICA.

3.1.- BIESTABLE JK DISPARADO POR FLANCO DE SUBIDA

En este ejercicio se va a realizar la simulación de un circuito biestable JK, con el objetivo de poder ver su funcionamiento, ya que es de especial importancia para el resto de los ejercicios.

Su símbolo se puede ver en la Figura 1.

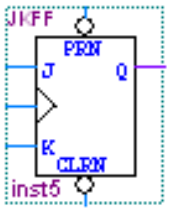


Figura 1. Biestable JK disparado por flanco de subida

El biestable JK visto en la figura anterior forma parte de un circuito comercial como es el 7476, que se representa en la Figura 2.

[1] *En principio las prácticas están pensadas para realizarse sobre la tarjeta educacional DE2, si bien, por cualquier motivo, no hubiera un número suficiente de éstas, se emplearía la tarjeta educacional UP2. Por tanto, ambas tarjetas son tenidas en cuenta en este guión ya que presentan diferentes características.*

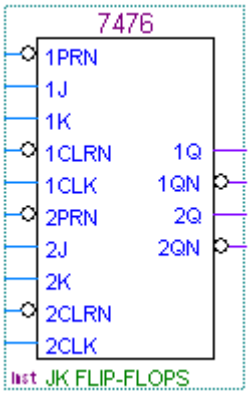


Figura 2. Circuito 7576.

Como se puede apreciar en la Figura 2, el circuito comercial 7476 está compuesto por dos biestables JK como el de la Figura 1. Por lo tanto, a través de la simulación del biestable JK, se conocerá el funcionamiento del circuito 7476.

El circuito a introducir en el Editor Gráfico es el mostrado en la Figura 3. Para la inclusión del biestable JK, introducir 'jkff' en el correspondiente apartado de la ventana de introducción de símbolos.

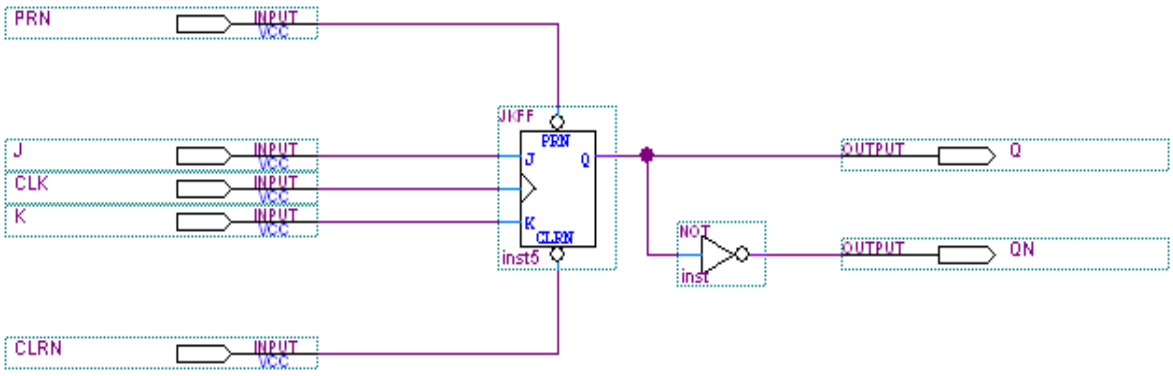


Figura 3. Circuito con biestable JK

Simular el diseño, eligiendo una frecuencia apropiada para la señal de reloj, comprender y explicar el funcionamiento en cada una de las distintas situaciones, ayudándose de la tabla de excitaciones del biestable JK, mostrada a continuación.

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabla 1.

3.2.- CONTADOR SÍNCRONO ASCENDENTE DE MODULO 10

En este ejercicio se pretende realizar un contador que cuente de cero a nueve, y una vez llegado a ese punto, inicialice la cuenta desde cero.

Introducir el diseño mostrado en la Figura 4 en el Editor Gráfico, utilizando para ello el símbolo 'jkff' también utilizado en el apartado anterior.

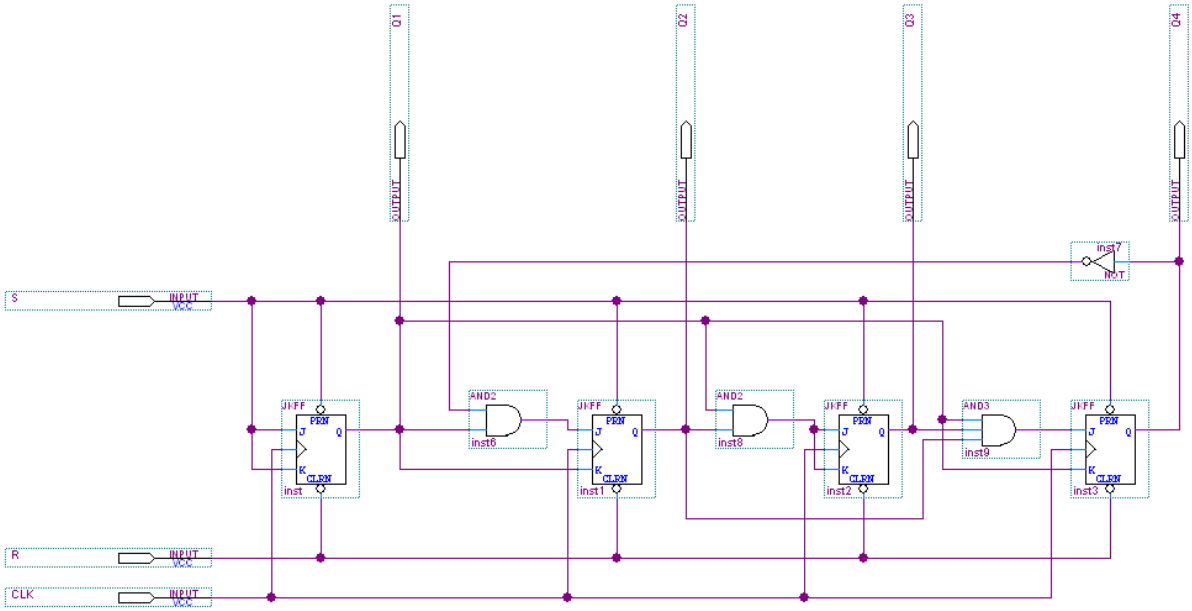


Figura 4. Circuito del contador ascendente síncrono módulo 10.

Realizar la simulación del diseño de la Figura 4 para poder ver el comportamiento del mismo. Para ello, elegir, por ejemplo, un tiempo de simulación de 25 ms y una frecuencia para la señal de reloj de 1 KHz (se pueden escoger otros valores siempre que permitan ver el resultado con claridad). Dar al inicio, manualmente, un valor lógico de '1' para la señal 'S' de entrada y un valor lógico de '0' a la señal 'R' durante, por ejemplo, un tiempo de 2 ms, para así inicializar el contador a Q1=Q2=Q3=Q4=0. A partir de los 2 ms, dar a la señal 'R' un valor lógico de '1' para que de esta manera las entradas asíncronas queden desactivadas.

- Observar que el circuito se comporta como un contador de módulo 10 ascendente.
- Crear un símbolo para el contador.

3.3.- VOLCADO DEL CONTADOR SÍNCRONO ASCENDENTE DE MÓDULO 10

El objetivo de este ejercicio es el de aprender a realizar volcados de los correspondientes diseños a la tarjeta educacional de forma que se pueda ver su funcionamiento en la misma. Para ello, hay varios aspectos que hay que tener en cuenta ya que si se realiza el volcado del diseño tal y como está hecho hasta ahora no funcionará correctamente en la placa.

En primer lugar, para que resulte más sencillo de ver, se debe utilizar el decodificador BCD-7 segmentos 7449, el cual se encuentra en la carpeta 'others/maxplus2', de forma que los números aparezcan en los displays y así no tener que utilizar para ello los diodos led.

En segundo lugar, como la señal de reloj 'CLK' se va ha obtener mediante un pulsador de la placa educacional, hay que tener en cuenta que se producirán unos rebotes hasta que se estabilice la señal, lo que puede afectar al correcto funcionamiento del diseño.

Utilizando la placa DE2 estos rebotes apenas son apreciables, si bien para asegurarnos del correcto funcionamiento del circuito es necesario emplear un bloque elimina-rebotes, proporcionado en la carpeta de trabajo. Hay que abrir el archivo 'elimina-rebotes.vhd', compilarlo y crear su correspondiente símbolo para poder ser insertado en los respectivos diseños.

En el caso de utilizar la tarjeta educacional UP2, es imprescindible la utilización del bloque elimina-rebotes para que el diseño funcione correctamente en la placa.

A partir de los símbolos del contador de módulo 10, decodificador BCD-7 segmentos y bloque elimina-rebotes, realizar el diseño mostrado en la Figura 5.

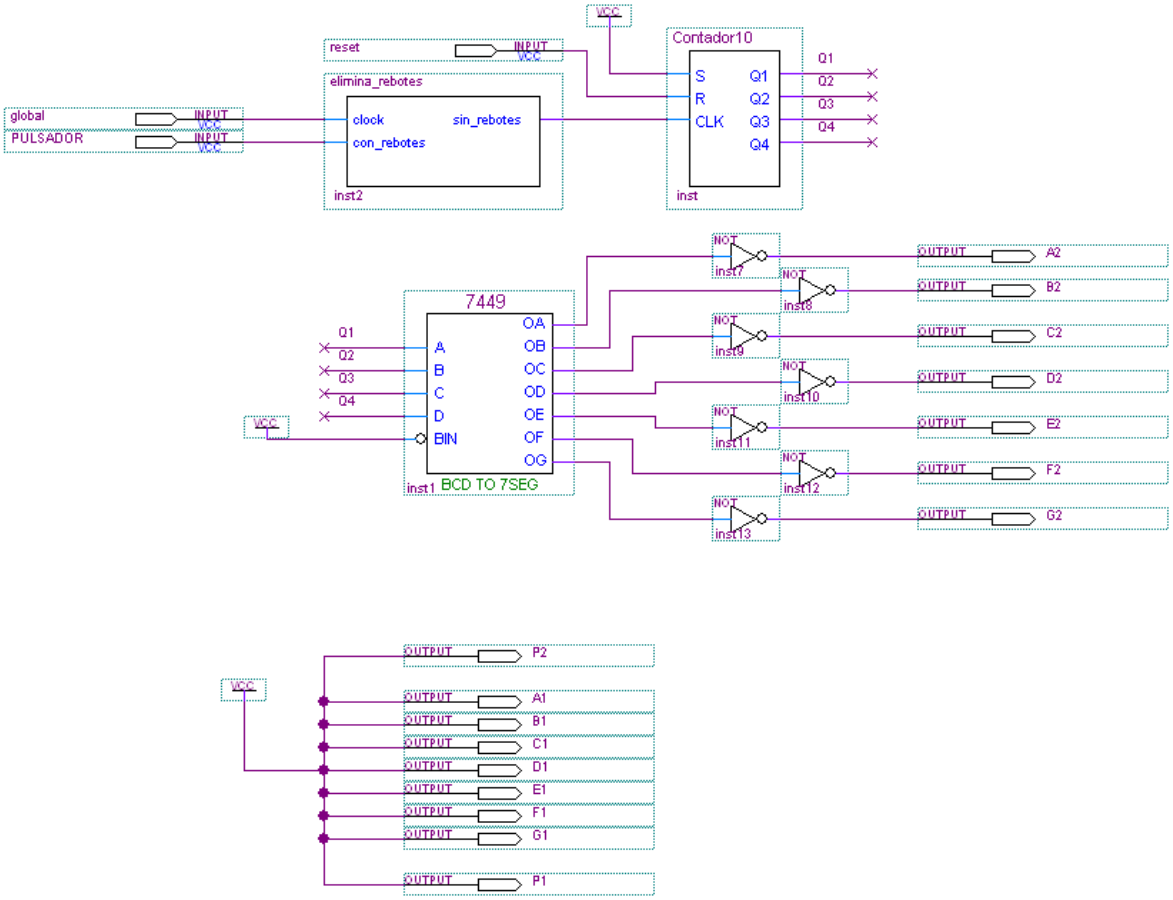


Figura 5. Circuito contador módulo 10 para volcado en placa.

El bloque de conexiones A1-G1, se debe a todos los segmentos del display uno. Los leds de la placa trabajan con lógica negativa, por lo que si se quiere que éstos permanezcan apagados, el diseño debe incluir este bloque de forma que dichos leds estén conectados a VCC.

Las entradas denominadas P2 y P1 que se pueden apreciar en el circuito de la Figura 5, han sido incluidas para el caso de querer implementar este circuito en la tarjeta educacional UP2 (para la tarjeta DE2 no son necesarias). Estas entradas se corresponden con los puntos decimales de sus displays y, como en este caso también trabajan con lógica negativa, se conectan a VCC para que permanezcan apagados.

Realizar la simulación del diseño y volcado a la tarjeta educacional siguiendo los pasos que se explican a continuación. Observar el funcionamiento en la placa y comparar con lo obtenido en la simulación.

- Volcado en la tarjeta educacional DE2:

A continuación se presenta la serie de pasos que permiten la configuración y volcado de los respectivos diseños a la tarjeta educacional DE2.

Tener en cuenta que en el caso de no haber escogido el dispositivo EP2C35F672C6 de la familia Cyclone II a la hora de definir el proyecto, es necesario acceder a la ventana mostrada en la Figura 6, donde se escoge la familia y dispositivo a programar. Para ello acceder al menú 'Assignments' seguido de la opción 'Device'.

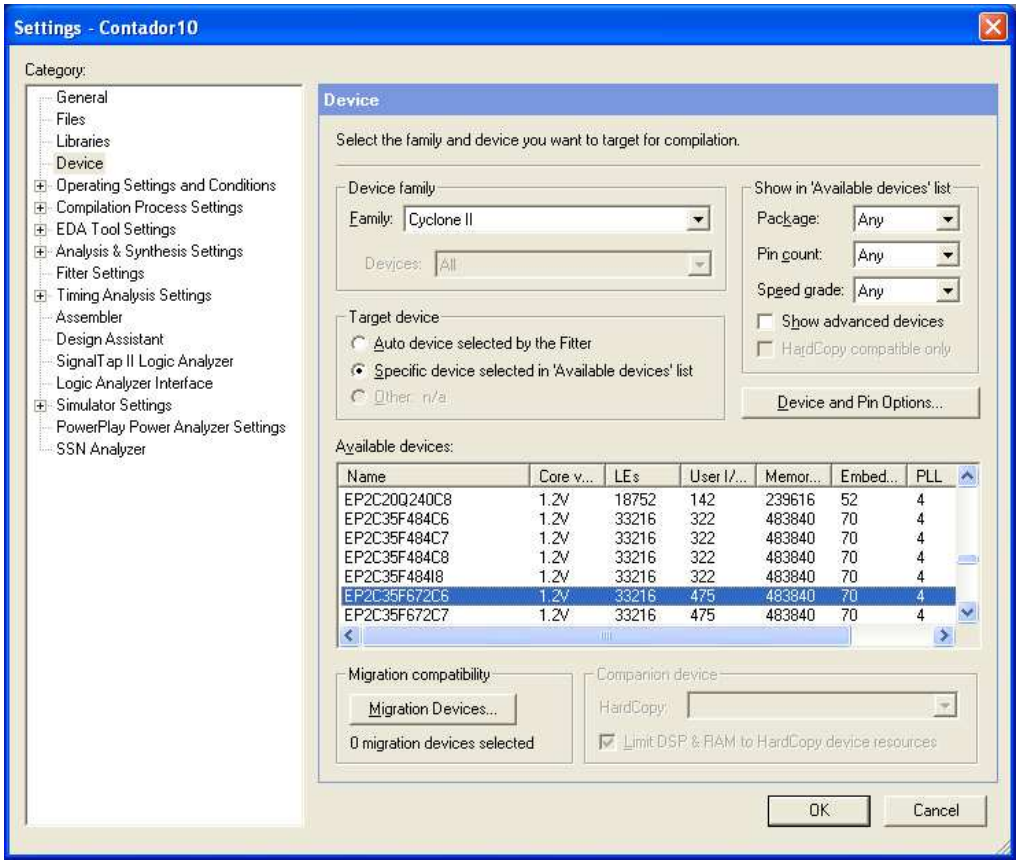


Figura 6. Selección del dispositivo a programar

1.- Asignación de pines:

Durante el proceso de compilación, el compilador de Quartus II es libre de elegir cualquier pin de la FPGA seleccionada que le sirvan de entradas y salidas. Sin embargo, la placa DE2 requiere especificar las conexiones entre la FPGA y el resto de componentes de la placa.

Para realizar la especificación de los pines es necesario utilizar la herramienta 'Assignment Editor' situada dentro del menú desplegable 'Assignments' de la barra de herramientas principal. Siguiendo estos pasos se obtiene la ventana que se mostrada en la Figura 7.

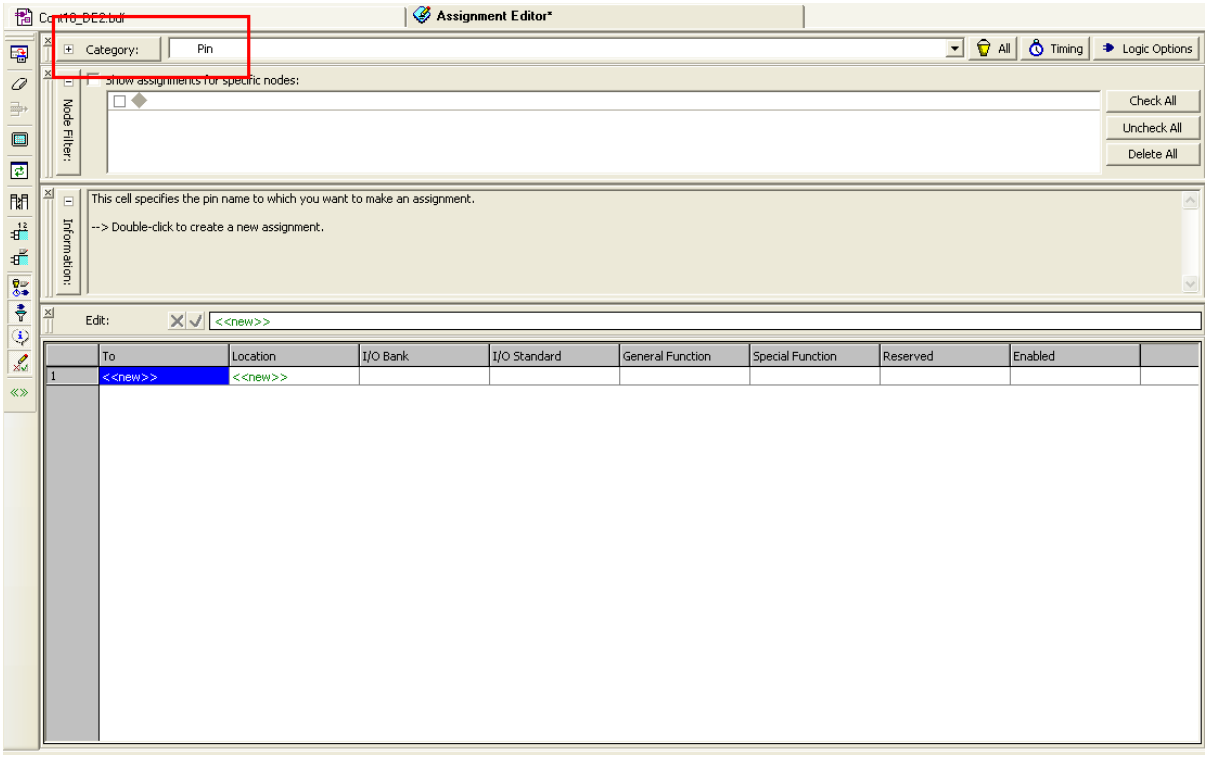


Figura 7. Ventana de la herramienta de asignación de pines

- Nota: es posible que al iniciar esta herramienta no tengamos todas las opciones que se pueden ver en la ventana de la Figura 7. Para solucionar esto basta con acceder al menú 'Window' de la barra de herramientas principal y seleccionar la opción 'Detach Window'. En este momento la herramienta pasa a estar en una ventana independiente, donde a través del menú 'View' se pueden cargar las opciones que faltan ('Category Bar', 'Node Filter Bar', 'Information Bar', 'Edit Bar').

Observando la figura anterior, para comenzar con la asignación de pines, seleccionar bajo el apartado categoría, recuadrado en rojo, la opción 'pin'. A continuación hacer doble clic sobre la entrada <<new>> sombreada en color azul, obteniendo un menú desplegable en el que aparece cada uno de los puertos que se pretenden asignar, tal y como muestra la Figura 8.

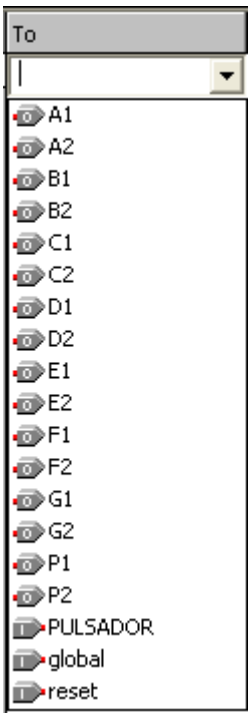
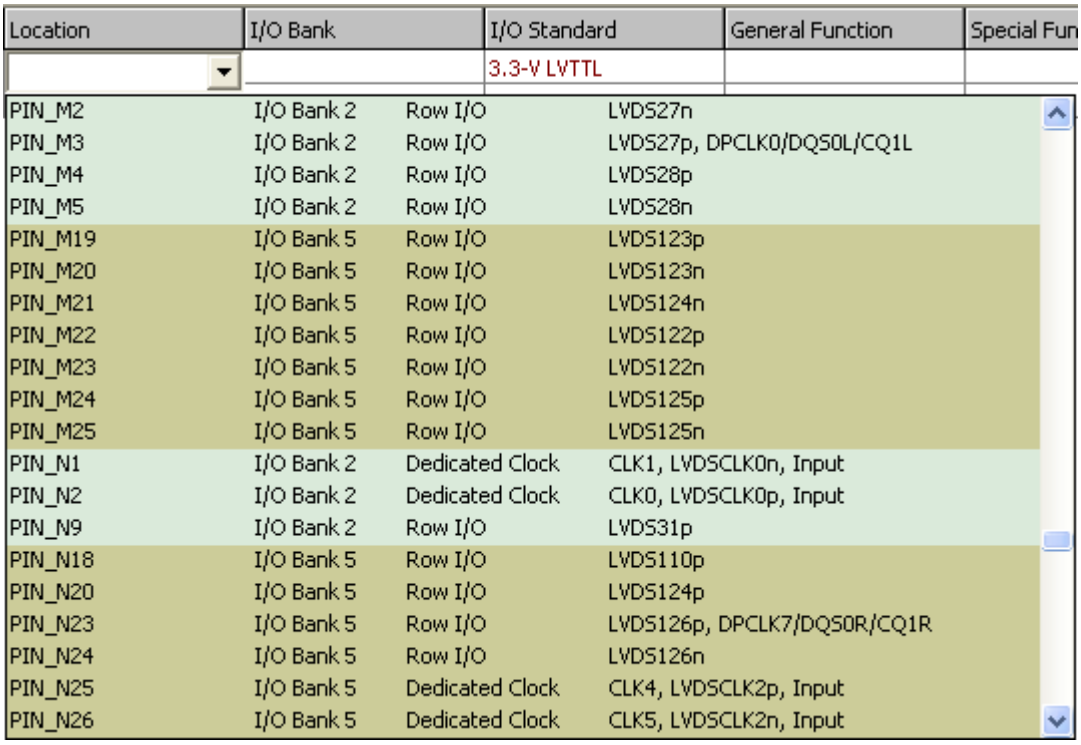


Figura 8. Menú desplegable con los nombres de las entradas y salidas

A partir de este menú, seleccionar el pin que va a ser asignado y, a continuación, acudir a la columna denominada 'Location' haciendo doble clic sobre su correspondiente celda para que aparezca el menú desplegable que permite asignar el pin con las características exigidas por cada entrada. Este menú es mostrado en la Figura 9.



Location	I/O Bank		I/O Standard	General Function	Special Fun
			3.3-V LVTTL		
PIN_M2	I/O Bank 2	Row I/O	LVDS27n		
PIN_M3	I/O Bank 2	Row I/O	LVDS27p, DPCLK0/DQS0L/CQ1L		
PIN_M4	I/O Bank 2	Row I/O	LVDS28p		
PIN_M5	I/O Bank 2	Row I/O	LVDS28n		
PIN_M19	I/O Bank 5	Row I/O	LVDS123p		
PIN_M20	I/O Bank 5	Row I/O	LVDS123n		
PIN_M21	I/O Bank 5	Row I/O	LVDS124n		
PIN_M22	I/O Bank 5	Row I/O	LVDS122p		
PIN_M23	I/O Bank 5	Row I/O	LVDS122n		
PIN_M24	I/O Bank 5	Row I/O	LVDS125p		
PIN_M25	I/O Bank 5	Row I/O	LVDS125n		
PIN_N1	I/O Bank 2	Dedicated Clock	CLK1, LVDSCLK0n, Input		
PIN_N2	I/O Bank 2	Dedicated Clock	CLK0, LVDSCLK0p, Input		
PIN_N9	I/O Bank 2	Row I/O	LVDS31p		
PIN_N18	I/O Bank 5	Row I/O	LVDS110p		
PIN_N20	I/O Bank 5	Row I/O	LVDS124p		
PIN_N23	I/O Bank 5	Row I/O	LVDS126p, DPCLK7/DQS0R/CQ1R		
PIN_N24	I/O Bank 5	Row I/O	LVDS126n		
PIN_N25	I/O Bank 5	Dedicated Clock	CLK4, LVDSCLK2p, Input		
PIN_N26	I/O Bank 5	Dedicated Clock	CLK5, LVDSCLK2n, Input		

Figura 9. Ventana de pines disponibles

Desde esta ventana se puede asignar la disposición de pines que interesa dada la configuración de la tarjeta educacional. La siguiente tabla muestra la disposición de pines que se va a utilizar:

Señal	Pin
PULSADOR	PIN_G26
Reset	PIN_P23
Global	PIN_D13
A2	PIN_R2
B2	PIN_P4
C2	PIN_P3
D2	PIN_M2
E2	PIN_M3
F2	PIN_M5
G2	PIN_M4
A1	PIN_L3
B1	PIN_L2
C1	PIN_L9
D1	PIN_L6
E1	PIN_L7
F1	PIN_P9
G1	PIN_N9

Tabla 2.

Mediante esta asignación, se están utilizando los dos pulsadores situados más a la derecha de la placa, una señal de reloj de 27 MHz, y la pareja de displays situados más a la izquierda de la placa.

La disposición de los pines se puede consultar en las tablas presentes en el Anexo 2 del manual.

Una vez que se ha terminado con la asignación de pines para este diseño, guardar la operación y volver a compilar para que el dispositivo se quede con su nueva configuración. Además, el usuario puede usar esta misma asignación de pines para posteriores diseños en la que sea requerida. Acudiendo al menú 'File' situado en la barra de herramientas principal mediante la opción 'Export' se crea un archivo con extensión '.csv' capaz de ser leído por el software Microsoft Excel. A continuación, en un nuevo proyecto, es posible importar esta asignación escogiendo el menú 'Assignments' seguido de la opción 'Import Assignments' seleccionando el archivo con extensión '.csv' que se desea importar.

2.- Programación:

El dispositivo FPGA debe ser programado y configurado si se quiere implementar el circuito diseñado. El archivo de configuración requerido para la operación ha sido creado previamente en el proceso de compilación.

La placa DE2 de Altera puede ser configurada de dos diferentes maneras: modo JTAG y modo AS, siendo el primero el que va a ser utilizado en las prácticas. Los datos de configuración son enviados desde el PC, con el software Quartus II activo, a través de un puerto USB cualquiera, hasta la placa a través del puerto USB situado más a la izquierda de la misma, junto a la entrada de alimentación. Es fundamental, para realizar esta conexión tener el controlador USB-Blaster instalado.

- **Modo de programación JTAG:**

En primer lugar, la placa DE2 cuenta con un conmutador de dos posiciones RUN/PROG, el cual ha de ser situado en la posición RUN.

A continuación, en el software Quartus II, acceder al menú 'Tools' situado en la barra de herramientas principal y seleccionar la opción 'Programmer'. También es posible acceder directamente a su icono.

Siguiendo estos pasos, se obtiene la ventana mostrada a continuación:

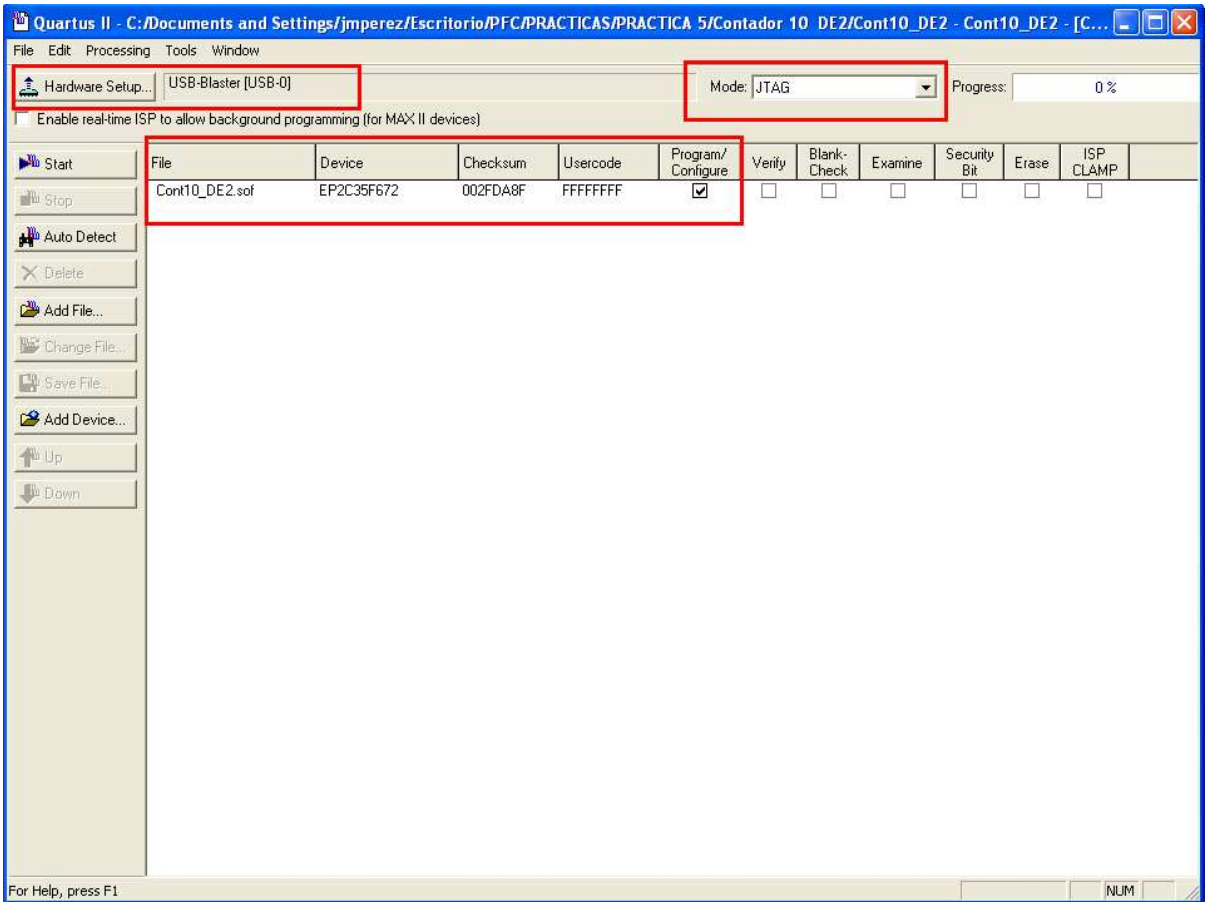


Figura 10. Ventana del Programador

En esta ventana es necesario especificar tanto el hardware como el modo de programación que va a ser utilizado. Si estos parámetros no vienen ya dados por defecto, seleccionar JTAG en la casilla 'Mode'. Asimismo, si el hardware USB-Blaster no ha sido el elegido, pulsar sobre el botón 'Hardware Setup' situado en la parte superior izquierda de la ventana y seleccionar la opción USB-Blaster en la nueva ventana, mostrada en la Figura 11.

Además se observa en la Figura 10, como el archivo de programación, con extensión '.sof', también ha sido cargado por defecto. En caso de no ser así, pulsar sobre el botón 'Add Files' de la barra de herramientas adjunta al programador. Éste es el archivo de configuración creado por el compilador y que incluye toda la información necesaria para la programación del dispositivo FPGA. Comprobar también que el dispositivo que aparece en la misma línea es el EP2C35F672, que es el correspondiente a la placa DE2. Si todo es correcto, hacer clic sobre la casilla de la opción 'Program/Configure' y presionar el botón 'Start' de la barra de herramientas. Cuando la programación está siendo realizada, aparecen los correspondientes

mensajes en la ventana de mensajes, así como la barra de progreso, que se encuentra en la parte superior derecha, muestra el estado de la programación. Un diodo LED de la placa se ilumina cuando la configuración del dispositivo ha concluido satisfactoriamente.

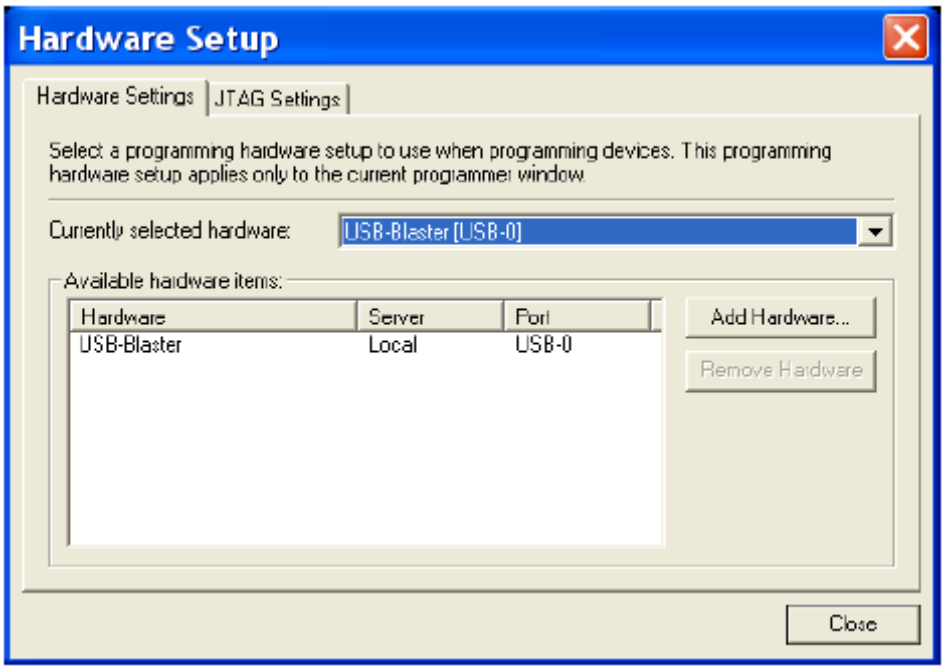


Figura 11. Ventana de selección de hardware

- Volcado en la tarjeta educacional UP2:

A continuación se presenta una breve descripción de cómo programar la tarjeta educacional UP2, es decir, cómo volcar diseños a la misma.

En primer lugar es necesario comprobar la configuración de los puentes sobre la propia tarjeta, optando por la configuración que se muestra en la Figura 12 como la adecuada para programar únicamente el dispositivo de la familia FLEX 10K.

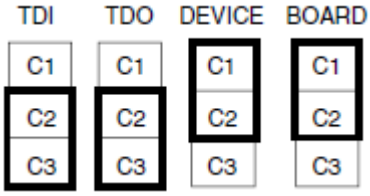


Figura 12. Disposición de los puentes de configuración de la placa

Asimismo, comprobar que el dispositivo elegido en la definición del proyecto se corresponde con la familia de dispositivos FLEX 10K dentro de la cual se encuentra el modelo EPF10K70RC240-4. Para ello acceder al menú 'Assignments' de la barra de herramientas principal y seleccionar la opción 'Device'.

A continuación, es necesario asignar a cada uno de los puertos de entrada y salida el correspondiente pin del dispositivo a programar. Mediante la ventana de asignación de pines,

se asignan a los puertos de entrada y salida del diseño los pines correspondientes del dispositivo PLD que se está utilizando. El acceso a esta ventana se obtiene desde el menú 'Assignments' seguido de la opción 'Assignment Editor', obteniendo una ventana como la mostrada en la Figura 13.

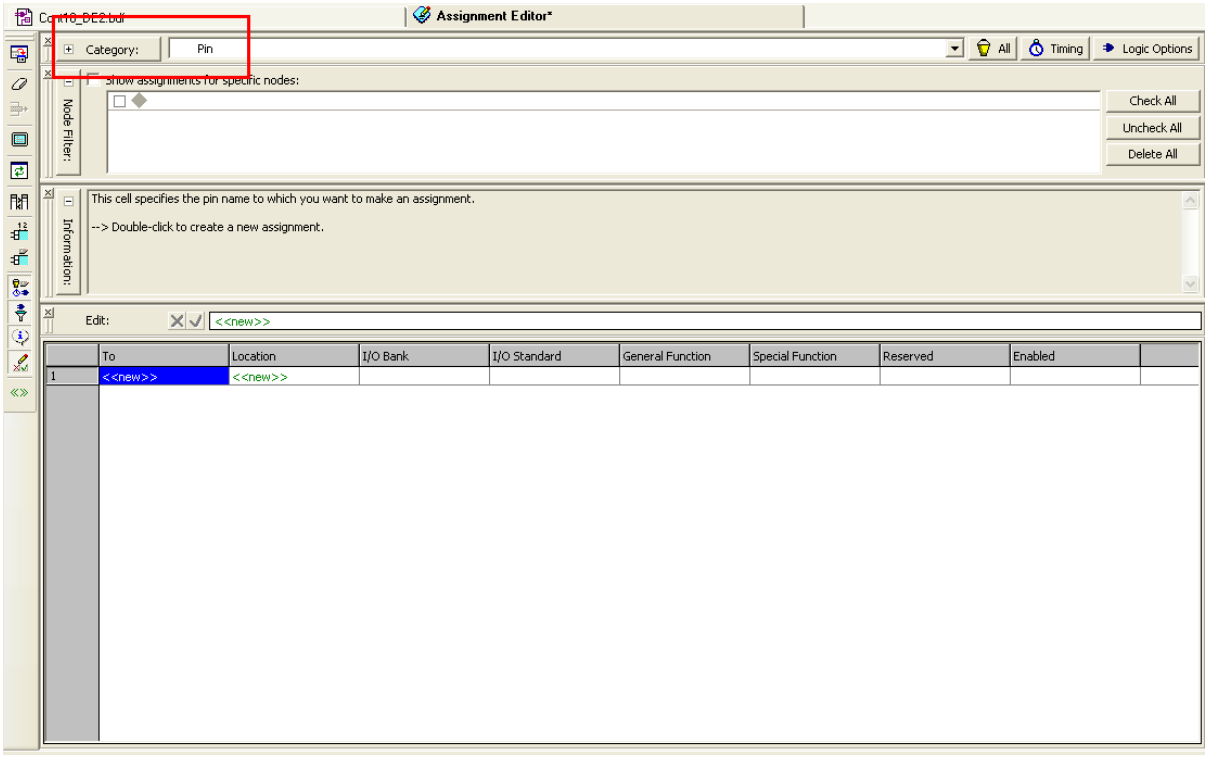


Figura 13. Ventana de la herramienta de asignación de pines

Observando la figura anterior, para comenzar con la asignación de pines, seleccionar bajo el apartado categoría, recuadrado en rojo, la opción 'pin'. A continuación hacer doble clic sobre la entrada <<new>> sombreada en color azul, obteniendo un menú desplegable en el que aparecen cada uno de los puertos que se pretenden asignar, tal y como muestra la Figura 14.

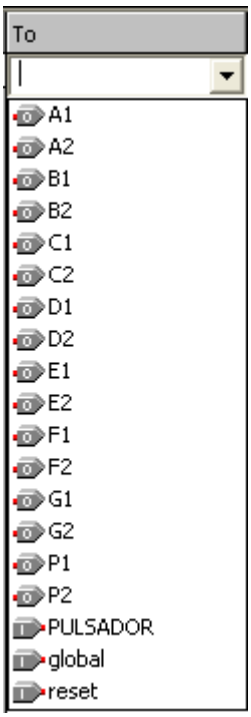


Figura 14. Menú desplegable con los nombres de las entradas y salidas

A partir de este menú, seleccionar el pin que va a ser asignado y, a continuación, acudir a la columna denominada 'Location' haciendo doble clic sobre su correspondiente celda para que aparezca el menú desplegable que permite asignar el pin con las características exigidas por cada entrada. Este menú es mostrado en la Figura 15.

Location	I/O Bank	I/O Standard	General Function	Special Fun
		3.3-V LVTTL		
PIN_M2	I/O Bank 2	Row I/O	LVDS27n	
PIN_M3	I/O Bank 2	Row I/O	LVDS27p, DPCLK0/DQS0L/CQ1L	
PIN_M4	I/O Bank 2	Row I/O	LVDS28p	
PIN_M5	I/O Bank 2	Row I/O	LVDS28n	
PIN_M19	I/O Bank 5	Row I/O	LVDS123p	
PIN_M20	I/O Bank 5	Row I/O	LVDS123n	
PIN_M21	I/O Bank 5	Row I/O	LVDS124n	
PIN_M22	I/O Bank 5	Row I/O	LVDS122p	
PIN_M23	I/O Bank 5	Row I/O	LVDS122n	
PIN_M24	I/O Bank 5	Row I/O	LVDS125p	
PIN_M25	I/O Bank 5	Row I/O	LVDS125n	
PIN_N1	I/O Bank 2	Dedicated Clock	CLK1, LVDSCLK0n, Input	
PIN_N2	I/O Bank 2	Dedicated Clock	CLK0, LVDSCLK0p, Input	
PIN_N9	I/O Bank 2	Row I/O	LVDS31p	
PIN_N18	I/O Bank 5	Row I/O	LVDS110p	
PIN_N20	I/O Bank 5	Row I/O	LVDS124p	
PIN_N23	I/O Bank 5	Row I/O	LVDS126p, DPCLK7/DQS0R/CQ1R	
PIN_N24	I/O Bank 5	Row I/O	LVDS126n	
PIN_N25	I/O Bank 5	Dedicated Clock	CLK4, LVDSCLK2p, Input	
PIN_N26	I/O Bank 5	Dedicated Clock	CLK5, LVDSCLK2n, Input	

Figura 15. Ventana de pines disponibles.

Desde esta ventana se puede asignar la disposición de pines que interesa dada la configuración de la tarjeta educacional. La siguiente tabla muestra la disposición de pines que se va a utilizar:

Señal	Pin	Señal	Pin
PULSADOR	Input Pin =28	P1	14 (1=LED OFF)
Reset	Input Pin = 29	P2	25 (1=LED OFF)
Global	Clock Pin = 91	A1	6 (1=LED OFF)
A2	Output Pin =17	B1	7 (1=LED OFF)
B2	Output Pin =18	C1	8 (1=LED OFF)
C2	Output Pin =19	D1	9 (1=LED OFF)
D2	Output Pin =20	E1	11 (1=LED OFF)
E2	Output Pin =21	F1	12 (1=LED OFF)
F2	Output Pin =23	G1	13 (1=LED OFF)
G2	Output Pin =24	-	-

Tabla 3.

Una vez que se ha terminado con la asignación de pines para este diseño, guardar la operación y volver a compilar para que el dispositivo se quede con su nueva configuración. Además, el usuario puede usar esta misma asignación de pines para posteriores diseños en la que sea requerida. Acudiendo al menú 'File' situado en la barra de herramientas principal mediante la opción 'Export' se crea un archivo con extensión '.csv' capaz de ser leído por el software Microsoft Excel. A continuación, en un nuevo proyecto, es posible importar esta asignación escogiendo el menú 'Assignments' seguido de la opción 'Import Assignments' seleccionando el archivo con extensión '.csv' que se desea importar.

La disposición de los pines se puede consultar en las tablas presentes en el Anexo 1 del manual.

Una vez que se tienen asignadas las entradas y salidas a los terminales correspondientes, se puede proceder a la programación mediante la herramienta que Quartus II tiene para ello.

Para comenzar la programación del dispositivo, acceder bien al menú 'Tools' y seleccionar la opción 'Programmer' o bien acudir directamente a su icono.

La ventana de esta herramienta es la mostrada en la Figura 16.

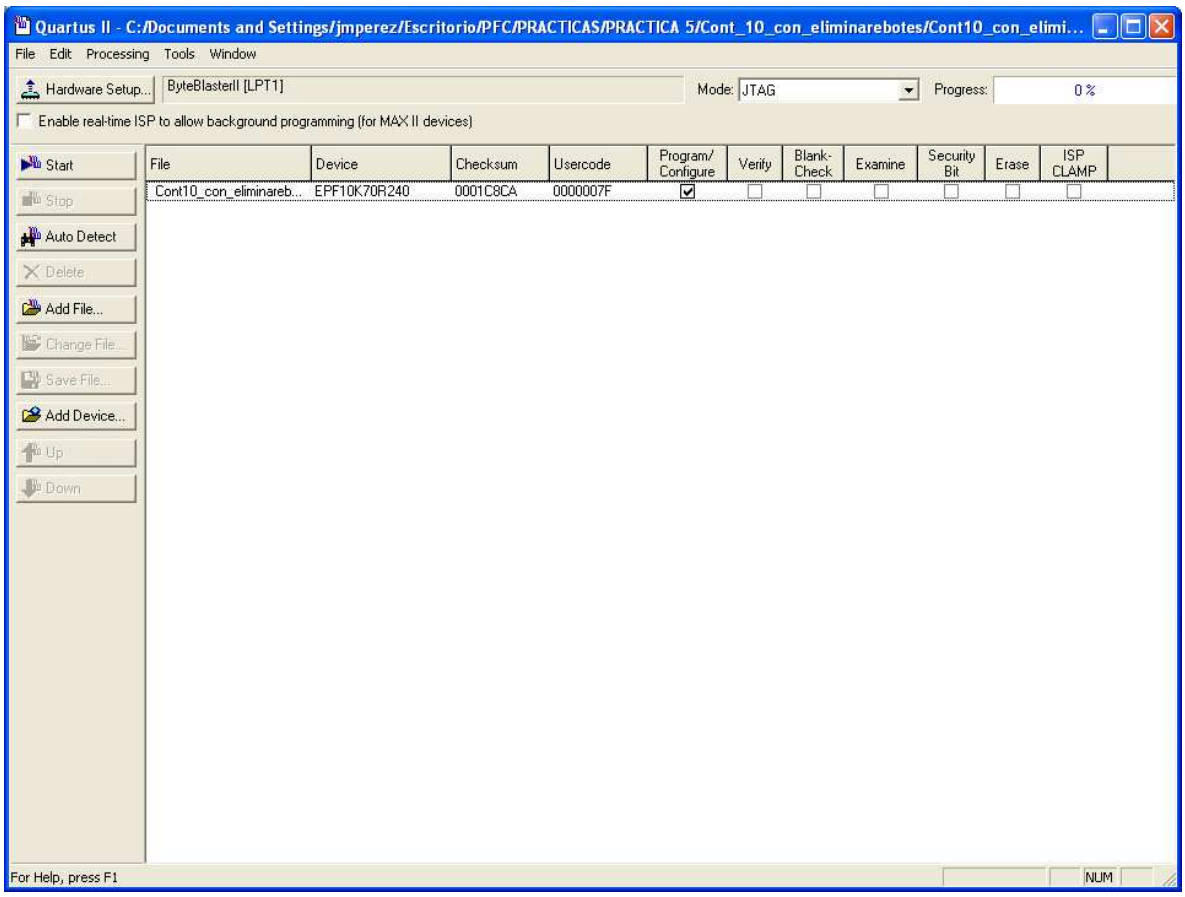


Figura 16. Ventana del programador

En esta ventana es posible asignar el hardware usado para la programación del dispositivo, seleccionar el archivo de diseño que va a ser volcado (con extensión '.sof' en nuestro caso y generado por el compilador) y especificar el propio dispositivo que va a ser programado. En general, todo esto vendrá asignado por defecto en la ventana de configuración, de todas maneras asegurarse de que aparece lo siguiente:

- Hardware Setup: ByteBlaster II (LPTI)
- Mode: JTAG
- File: archivo de diseño a volcar con extensión '.sof'.
- Device: EPF10K70R240.
- Opción 'Program/Configure' seleccionada.

En el caso de que alguno de estos parámetros no aparezca o no sea el adecuado, hacer uso de la barra de herramientas asociada al programador situada en la parte izquierda de la ventana, la cual permite añadir o borrar archivos de programación y realizar la selección del dispositivo adecuado.

Además, es posible que la primera vez que se haga uso de la herramienta, no aparezca el hardware de configuración ByteBlaster II, por lo que pulsando sobre el botón 'Hardware Setup' surge una nueva ventana que permite añadir el mismo.

Una vez hecha la configuración en la ventana, pulsar el botón 'Start' que da comienzo a la simulación. Cuando la programación está siendo realizada, aparecen los correspondientes mensajes en la ventana de mensajes, así como la barra de progreso, que se encuentra en la parte superior derecha, muestra el estado de la programación.

Por último, una vez que la tarjeta se encuentre programada, comprobar el correcto funcionamiento del diseño.

4.- EJERCICIOS DE DISEÑO

4.1.- Contador de módulo 5

4.1.1.- Diseñar un contador síncrono ascendente de módulo 5. Para ello, utilizar biestables del tipo JK y las puertas lógicas que sean necesarias.

Comprobar el correcto funcionamiento del diseño realizando su simulación y correspondiente volcado a la placa educacional DE2 o UP2.

Crear un símbolo para el diseño, ya que será utilizado en prácticas posteriores.

4.1.2.- En un nuevo proyecto, modificar el diseño anterior añadiéndole un pulsador que permita elegir el sentido, con lo que se obtiene un contador síncrono reversible.

Comprobar el funcionamiento del diseño realizando su simulación y volcado a la tarjeta educacional DE2 o UP2.

RESOLUCIÓN PRÁCTICA 3

A continuación se presenta la resolución de los diferentes ejercicios que componen esta práctica.

3.- DESARROLLO DE LA PRÁCTICA

3.1.- BIESTABLE JK DISPARADO POR FLANCO DE SUBIDA

El diseño a introducir en el Editor Gráfico es el siguiente:

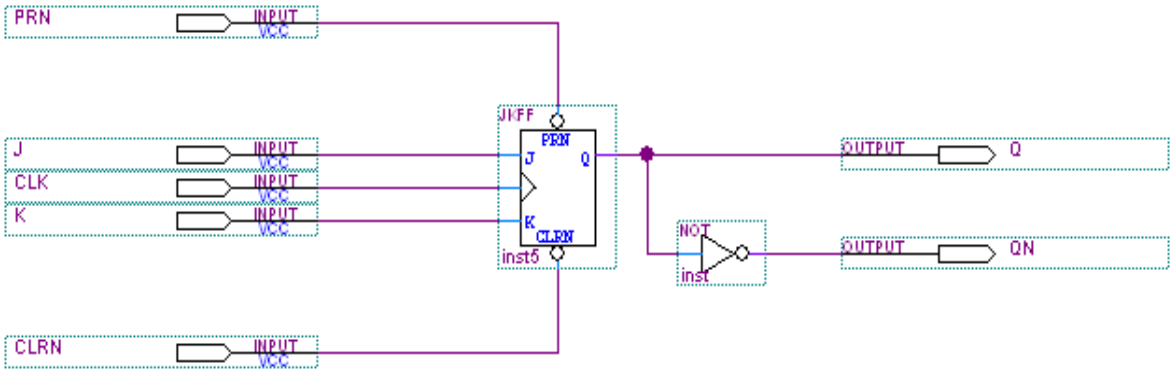


Figura 1. Circuito con biestable JK

A partir de este circuito, realizando su correspondiente simulación, se obtienen los siguientes resultados:

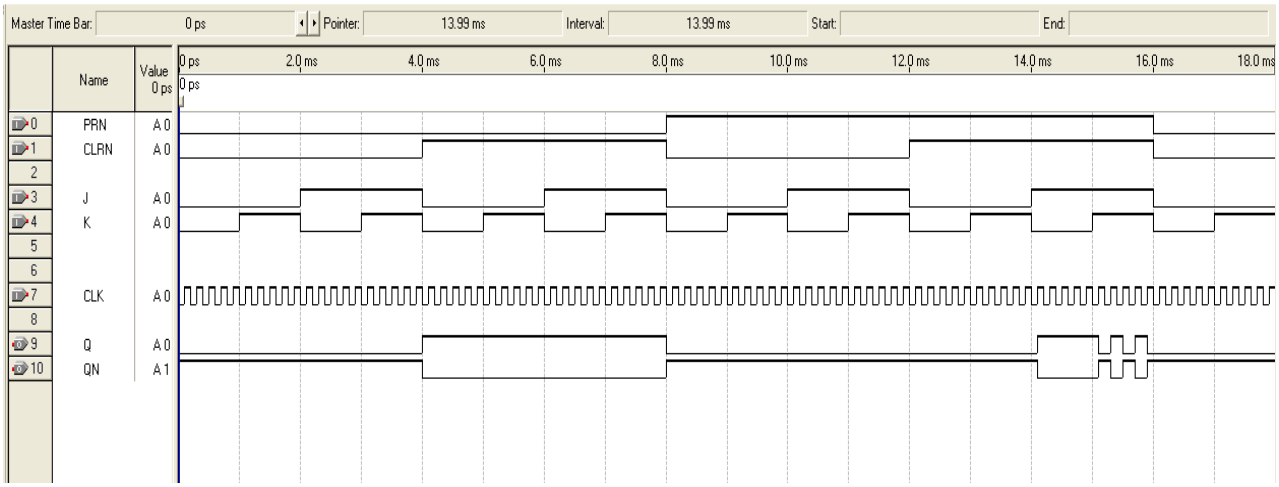


Figura 2. Resultados del biestable JK

Las frecuencias empleadas en la simulación son las siguientes:

- K → 500 Hz
- J → 250 Hz
- CLRN → 125 Hz
- PRN → 63 Hz
- CLK → 100 KHz

Como se puede apreciar en la Figura 2, el resultado es el correcto, pudiendo diferenciar en qué momento están actuando las entradas asíncronas (PRN y CLRN) y en qué momento están actuando las entradas síncronas disparadas por flanco de subida (J y K), viendo como las entradas asíncronas tienen prioridad sobre las síncronas.

En un primer momento, cuando ambas entradas asíncronas se encuentran a nivel bajo, no está claro qué pasa en la salida, ya que esto produce una situación ilegal.

En la última zona, donde las señales asíncronas se encuentran a nivel alto, se observa un comportamiento anómalo: la salida está continuamente conmutando de un estado a otro (toggle). Esto se puede entender viendo la tabla de excitaciones del biestable JK:

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabla 1.

Como se puede apreciar, siempre que la entrada 'J' tenga un valor lógico de uno, la señal 'K' puede tomar indistintamente cualquier valor, y la salida actual toma un valor lógico de cero. De la misma manera, cuando la señal 'K' toma un valor lógico de uno la señal 'J' puede tomar indistintamente cualquier valor, y la salida actual toma un valor lógico de uno. Por tanto, se comprueba que al estar ambas entradas a un nivel lógico alto, la salida oscila entre cero y uno continuamente.

3.2.- CONTADOR SÍNCRONO ASCENDENTE DE MODULO 10.

El diseño a introducir en el Editor Gráfico es el siguiente:

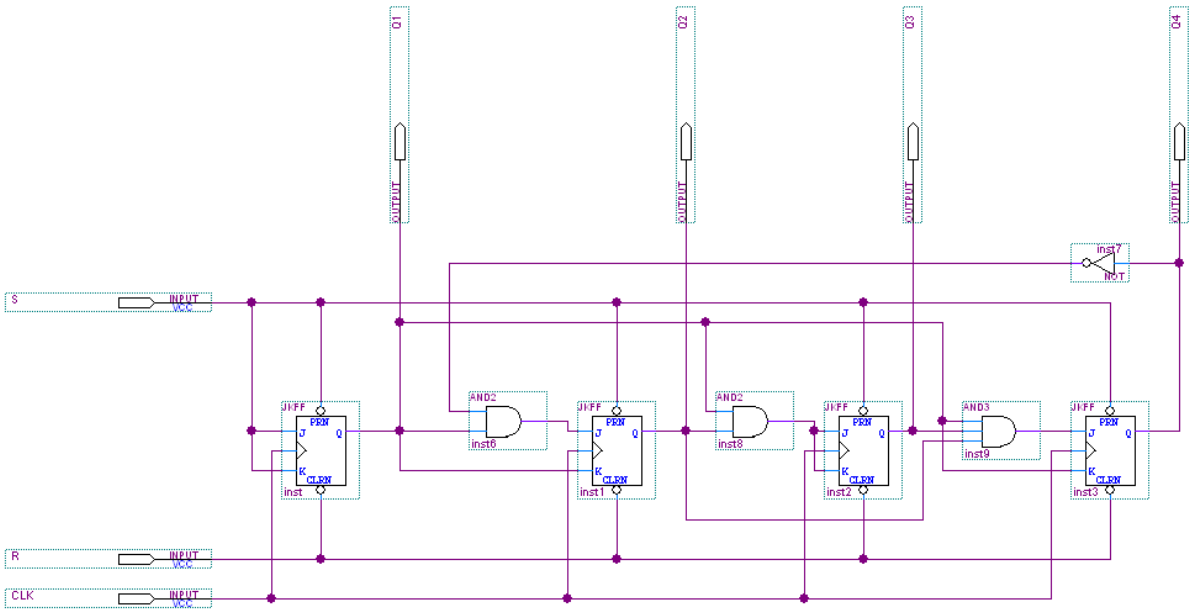


Figura 3. Circuito del contador ascendente síncrono módulo 10.

Observando el circuito, se tiene que la salida 'Q1' representa el bit menos significativo, mientras que la salida 'Q4' representa el bit más significativo.

El circuito realiza la cuenta ascendente de 0 a 9 siempre que las entradas asíncronas (S y R) se encuentren desactivadas (mediante un '1' lógico ya que trabajan a nivel bajo).

Tras realizar la simulación se obtiene el siguiente resultado:

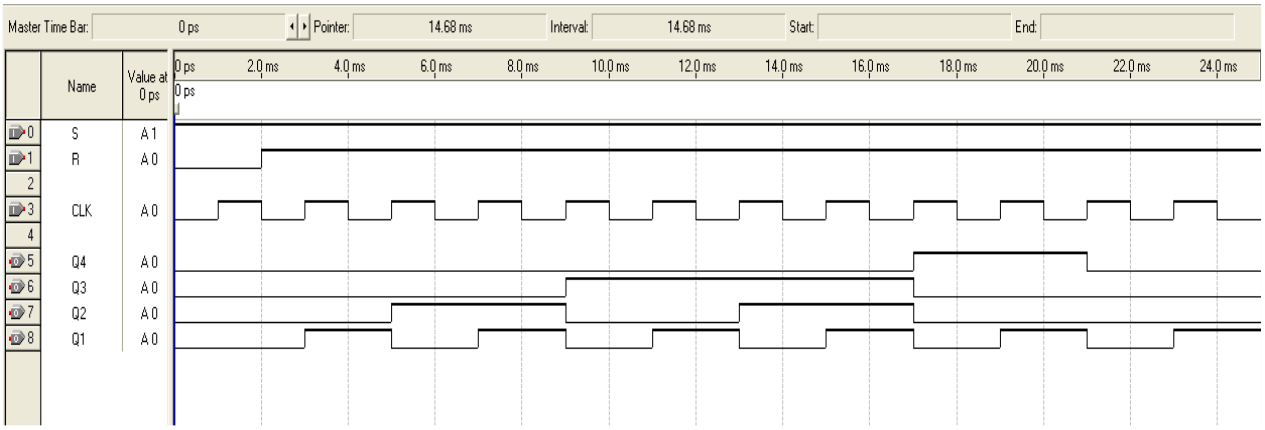


Figura 4. Resultado del contador síncrono ascendente de módulo 10

Como se puede comprobar en la Figura 4, el contador funciona correctamente, realizando la cuenta desde cero hasta nueve (incrementándose en los flancos de subida de la señal de reloj), para que una vez llegado a este punto comience de nuevo la cuenta desde cero.

Para realizar el volcado del diseño a la tarjeta educacional, es necesario introducir, en un nuevo proyecto, el diseño mostrado en la Figura 5, teniendo previamente que realizar el símbolo correspondiente al circuito del contador de módulo 10.

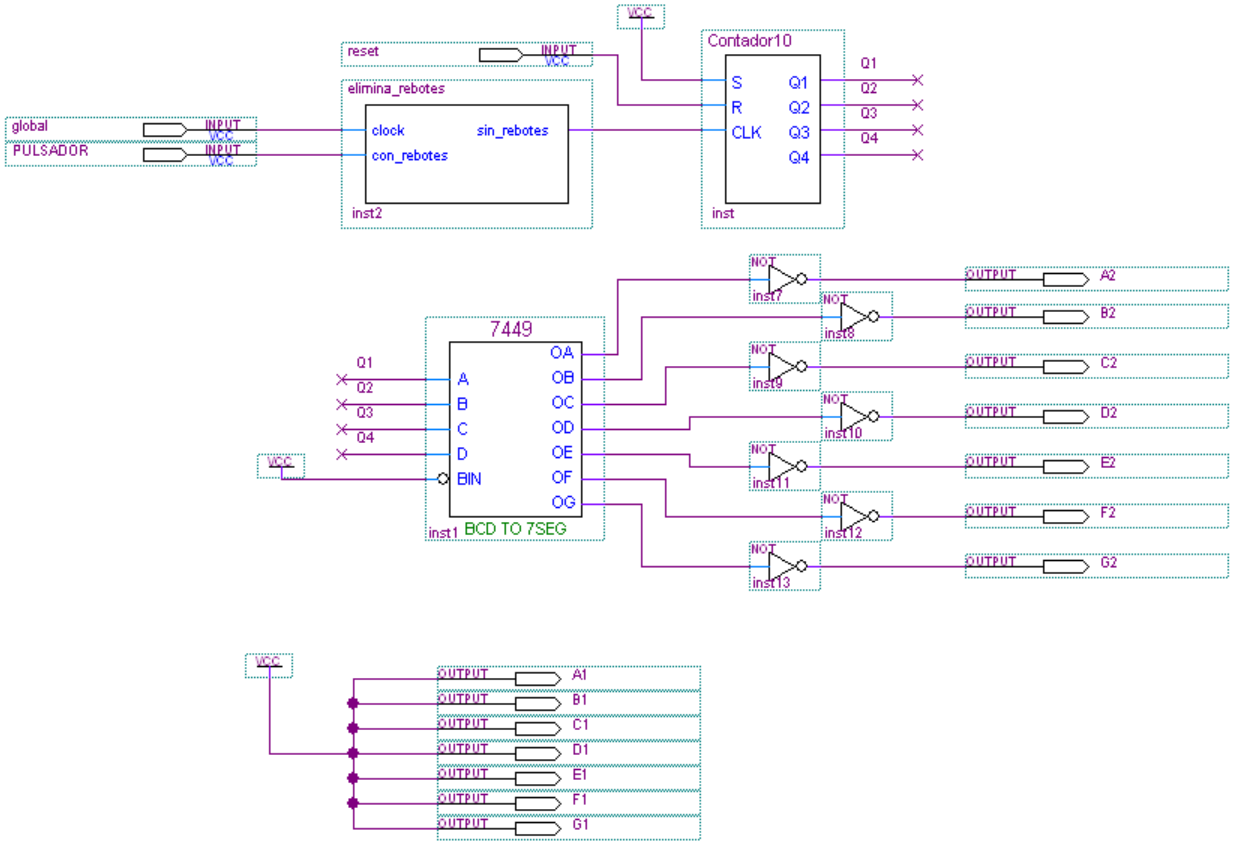


Figura 5. Contador de módulo 10 para volcar en la tarjeta educacional

La asignación de pines y programación del dispositivo se ha hecho siguiendo los pasos que aparecen en el guión de prácticas.

4.- EJERCICIOS DE DISEÑO

4.1.- Contador síncrono de módulo 5

4.1.1.- Contador síncrono ascendente de módulo 5:

Un contador síncrono ascendente de módulo 5 debe realizar la siguiente cuenta: 0, 1, 2, 3, 4, 0……

De esta manera el número máximo a representar es N = 4 (‘100’ en código binario), por lo que para la realización del circuito necesitamos una cantidad de tres biestables.

Se tiene que la tabla de excitaciones del biestable JK es la siguiente:

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabla 2.

Tomando como base la *Tabla 2* se construye la tabla que representa el comportamiento del contador.

Estado Actual Q(t)			Estado Siguiente Q(t+1)								
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X

Tabla 3.

Por medio del método de Karnaugh se obtienen las expresiones para las entradas de los distintos biestables, tomando como combinación de entrada el estado actual:

Función J_3:

Q_3/Q_2Q_1	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J3 = Q2 \cdot Q1$

Función K_3:

Q_3/Q_2Q_1	00	01	11	10
0	x	x	x	x
1	1	X	X	X

$K3 = 1$

Función J_2:

Q_3/Q_2Q_1	00	01	11	10
0	0	1	x	x
1	0	x	x	x

$J2 = Q1$

Función K_2:

Q_3/Q_2Q_1	00	01	11	10
0	x	x	1	0
1	x	x	x	x

$K2 = Q1$

Función J_1:

Q_3/Q_2Q_1	00	01	11	10
0	1	X	X	1
1	0	X	X	X

$$J1 = \overline{Q3}$$

Función K_1:

Q_3/Q_2Q_1	00	01	11	10
0	X	1	1	X
1	X	X	X	X

$$K1 = 1$$

En base a estas funciones, el circuito resultante queda de la siguiente manera:

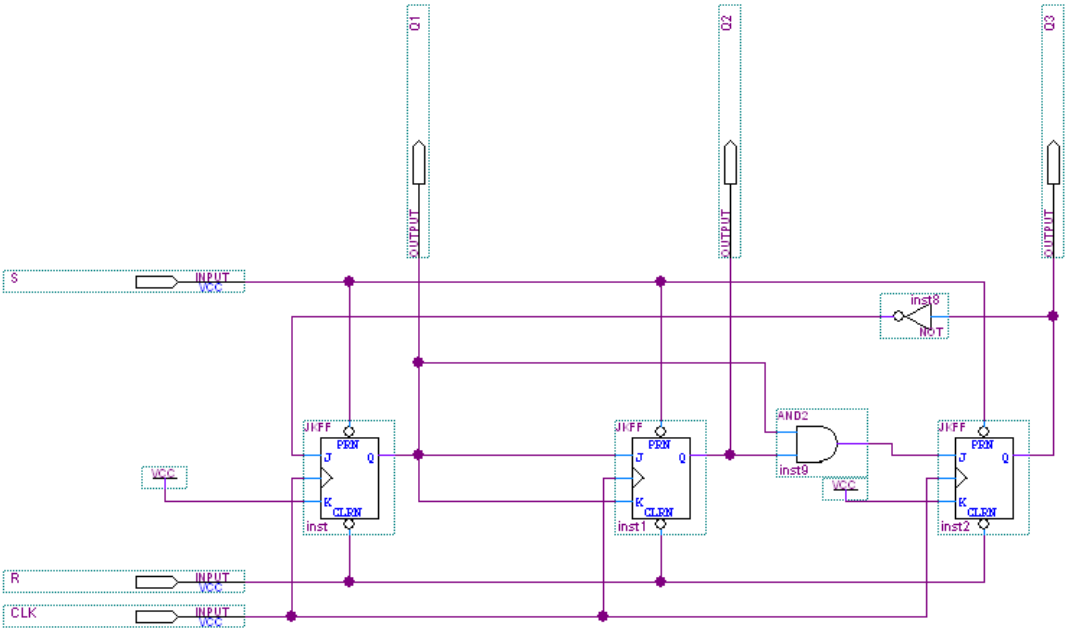


Figura 6. Contador síncrono ascendente de módulo 5

Realizando la simulación se obtiene el resultado mostrado en la Figura 7:

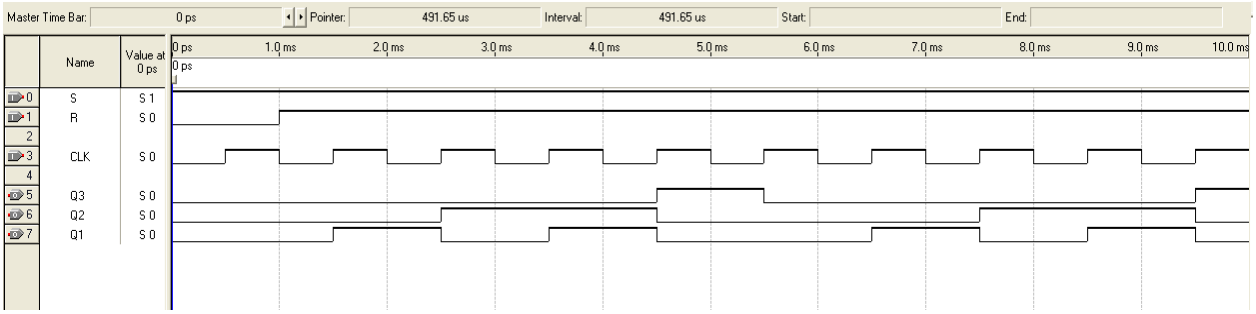


Figura 7. Resultado del contador ascendente de módulo 5

Se observa en la Figura 7 que el funcionamiento es el esperado, ya que el contador comienza a contar en el momento que las dos entradas asíncronas se encuentran desactivadas, realiza la cuenta desde cero hasta cuatro y vuelve a cero a continuación.

El circuito de volcado para el contador ascendente de módulo 5, una vez creado su correspondiente símbolo es el siguiente:

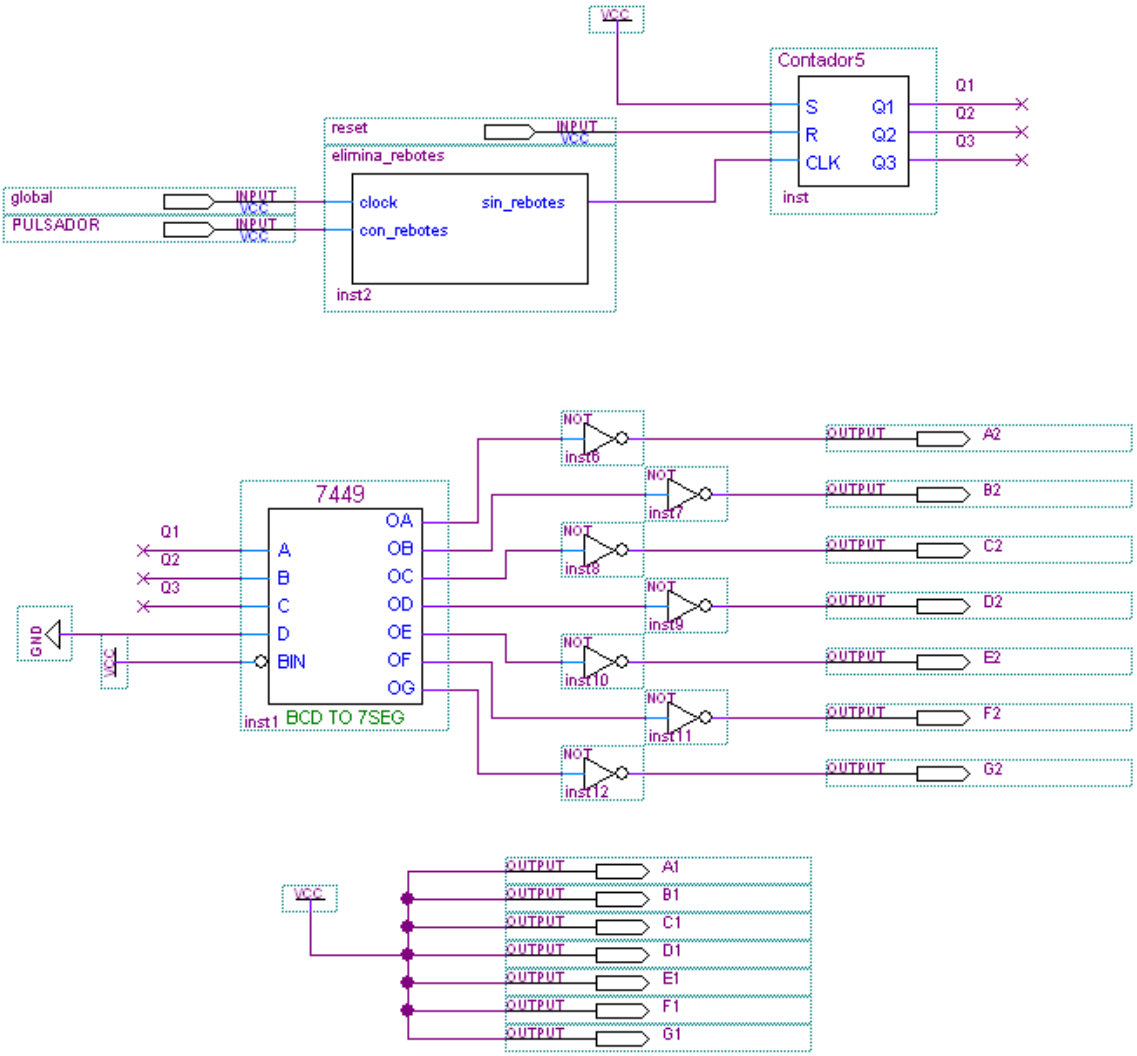


Figura 8. Circuito de volcado del contador ascendente de módulo 5

Un aspecto a tener en cuenta en el circuito de la Figura 8 es que el decodificador BCD-7 segmentos tiene cuatro entradas mientras que el contador de módulo 5 únicamente ofrece tres salidas. Por tanto, es necesario conectar la cuarta entrada del decodificador a masa para que ésta no afecte al resultado final.

4.1.2.- Contador síncrono reversible de módulo 5

La modificación de este circuito consiste en la inclusión de un pulsador que permita elegir el sentido de la cuenta. Por tanto, la forma de actuar del contador va a ser la siguiente:

X = 1 → 0, 1, 2, 3, 4, 0, ...
X = 0 → 0, 4, 3, 2, 1, 0, ...

Los pasos para el diseño del circuito son los mismos que los seguidos en el apartado anterior teniendo en cuenta la nueva variable.

Se construye la tabla que representa el comportamiento del circuito:

	Estado Actual Q(t)			Estado Siguiente Q(t+1)								
X	Q3	Q2	Q1	Q3	Q2	Q1	J3	K3	J2	K2	J1	K1
0	0	0	0	1	0	0	1	X	0	X	0	X
0	0	0	1	0	0	0	0	X	0	X	X	1
0	0	1	0	0	0	1	0	X	x	1	1	X
0	0	1	1	0	1	0	0	X	x	0	X	1
0	1	0	0	0	1	1	X	1	1	X	1	X
1	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
1	0	1	0	0	1	1	0	X	X	0	1	X
1	0	1	1	1	0	0	1	X	X	1	X	1
1	1	0	0	0	0	0	X	1	0	X	0	X

Tabla 4.

A través del método de las tablas de Karnaugh, se obtienen las expresiones para los diferentes biestables:

Función J_3:

xQ_3/Q_2Q_1	00	01	11	10
00	1	0	0	0
01	X	X	X	X
11	X	X	X	X
10	0	0	1	0

$$J3 = \overline{X} \cdot \overline{Q2} \cdot \overline{Q1} + X \cdot Q2 \cdot Q1$$

Función K_3:

xQ_3/Q_2Q_1	00	01	11	10
00	x	x	x	x
01	1	X	X	X
11	1	X	X	X
10	x	x	x	x

$$K3 = 1$$

Función J_2:

xQ_3/Q_2Q_1	00	01	11	10
00	0	0	X	X
01	1	X	X	X
11	0	X	X	X
10	0	1	X	X

$$J2 = \overline{X} \cdot Q3 + X \cdot Q1$$

Función K_2:

xQ_3/Q_2Q_1	00	01	11	10
00	X	X	0	1
01	X	X	X	X
11	X	X	X	X
10	X	X	1	0

$$K2 = X \cdot Q1 + \overline{X} \cdot \overline{Q1} = \overline{X \oplus Q1}$$

Función J_1:

xQ_3/Q_2Q_1	00	01	11	10
00	0	X	X	1
01	1	X	X	X
11	0	X	X	X
10	1	X	X	1

$$J1 = Q2 + \overline{X} \cdot Q3 + X \cdot \overline{Q3} = Q2 + X \oplus Q3$$

Función K_1:

xQ_3/Q_2Q_1	00	01	11	10
00	X	1	X	1
01	X	X	X	X
11	X	X	X	X
10	X	1	1	X

$$K1 = 1$$

De esta manera, el diseño del contador realizado en el Editor Gráfico es el siguiente:

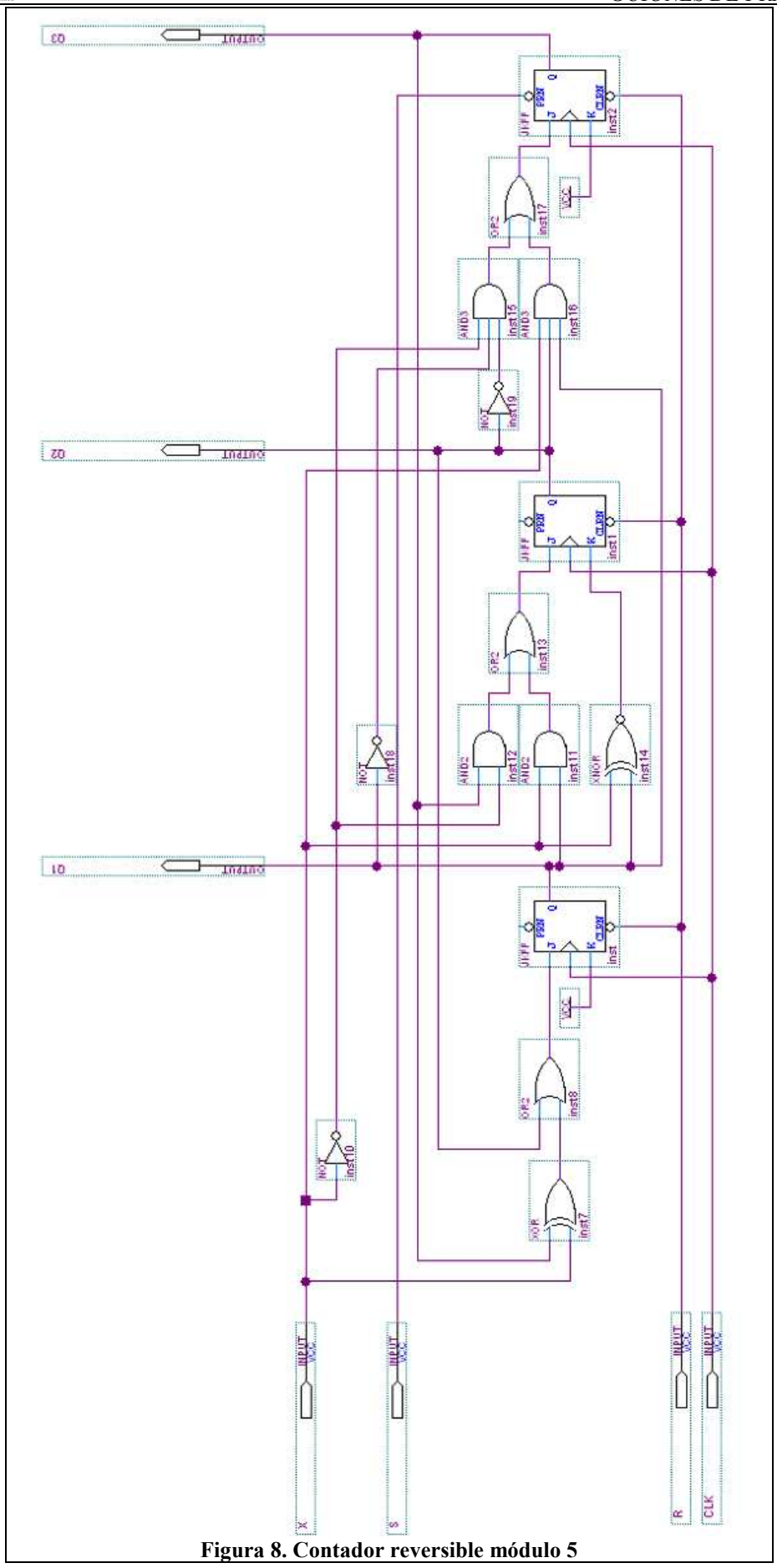


Figura 8. Contador reversible módulo 5

La Figura 9 muestra el resultado de la simulación:

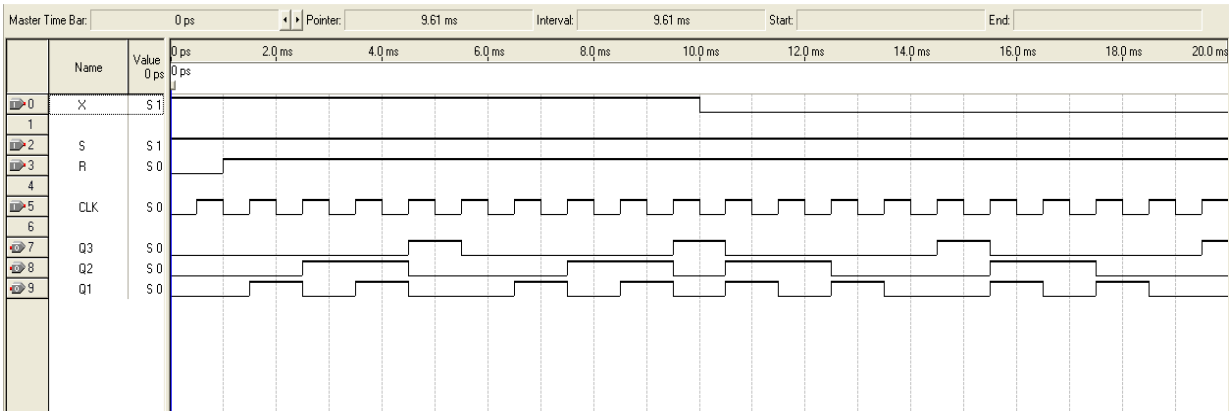


Figura 9. Resultado del contador síncrono reversible de módulo 5

En primer lugar, con la entrada 'X' a nivel alto, el contador realiza la cuenta en sentido ascendente para que, a continuación con 'X' a nivel bajo, el contador realice la cuenta en sentido descendente. Los valores de las salidas cambian según los flancos de subida de la señal de reloj, teniendo las entradas asíncronas desactivadas.

Los valores de las señales 'X', 'S' y 'R' son introducidos a mano, mientras que la señal de reloj tiene una frecuencia de 1 KHz.

El circuito de volcado a la tarjeta educacional, una vez creado el correspondiente símbolo es mostrado en la siguiente figura.

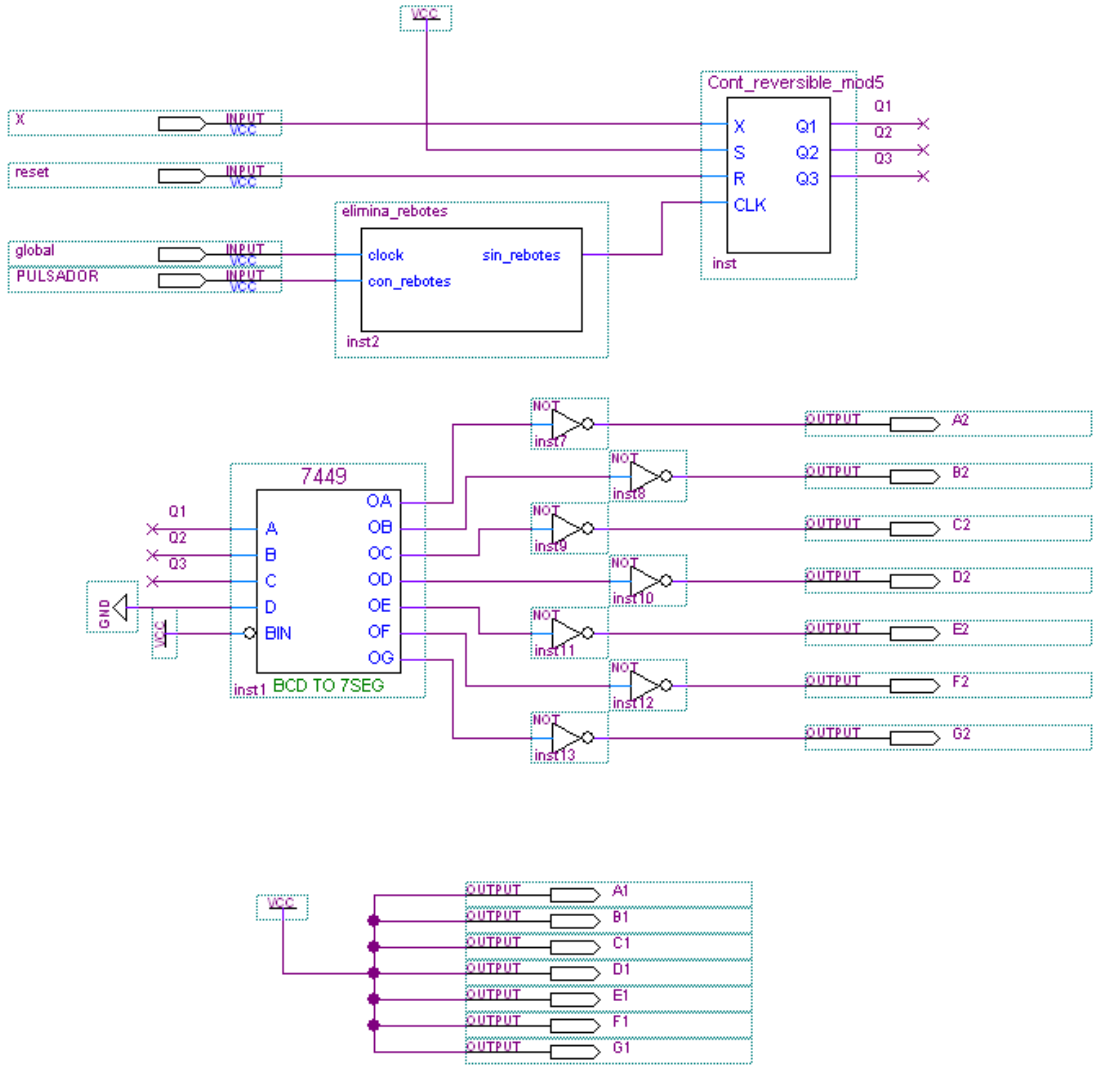


Figura 10. Contador síncrono reversible de módulo 5 para volcar

CAPÍTULO 10.

APLICACIÓN ESPECÍFICA

10.- APLICACIÓN ESPECÍFICA

10.1.- INTRODUCCIÓN

En este capítulo se expone una aplicación de un sistema de vídeo completo, desde su etapa de adquisición hasta su visualización final, implementado sobre un sistema basado en un dispositivo programable FPGA. Por tanto, se va a poder comprobar la diferencia, en cuanto a complejidad se refiere, entre los distintos diseños realizados en los respectivos guiones de prácticas y el diseño de esta aplicación. De igual manera, se trata de una aplicación que llama la atención visual de los usuarios, así como éstos tienen la posibilidad de interactuar con ella.

El diseño original en el que se basa esta aplicación puede descargarse desde la página Web de Altera: www.altera.com. Basta con acceder a la pestaña 'Products' seguido de la opción 'Development Kits/Cables'. En la nueva ventana, pulsar sobre el enlace 'All Development kits', donde se accede a todo el listado de kits de desarrollo que ofrece la compañía Altera. Dentro de este listado, acceder al enlace 'DE2-70 Digital Camera and Multimedia Development Platform'. A partir de aquí, se encuentra toda la información y recursos para la utilización del kit compuesto de la tarjeta educacional DE2-70, la cámara digital y el panel táctil.

La elección de esta aplicación para ser mostrada en la utilización de un entorno educativo basado en sistemas programables radica en conocer y aplicar los conocimientos adquiridos en distintas disciplinas, dado el amplio abanico de temas que se tocan.

El objetivo de su análisis es el de conocer la capacidad y forma de trabajar de un sistema basado en un dispositivo FPGA para trabajar con temas de vídeo, a la vez que se conoce cómo internamente funciona de forma general toda la cadena que permite implementar un sistema de vídeo completo.

En base a los objetivos mencionados, mediante la lectura de este capítulo, el lector tiene la oportunidad de visualizar conceptos teóricos sobre sistemas que trabajan con imágenes digitales, tales como una cámara digital o una pantalla LCD táctil. También se da una visión general al lenguaje de descripción hardware 'Verilog HDL' el cual es empleado en la presente aplicación.

Al final, es posible aplicar estos conceptos para la comprensión del diseño de la aplicación, y ésto sirva como base en el desarrollo de posteriores diseños.

10.2.- DESCRIPCIÓN DEL HARDWARE

10.2.1.-TARJETA EDUCACIONAL DE2-70

La placa DE2-70 representa la evolución de la tarjeta educacional DE2. En el Capítulo 5 del presente proyecto existe un manual de la placa DE2 en el que se explica el funcionamiento de la misma. Dicho contenido es igualmente válido para el caso de la tarjeta DE2-70, por lo que el apartado actual se centra en presentar y mostrar las características de la tarjeta educacional DE2-70.

10.2.1.1.- Diseño y componentes

La tarjeta educacional DE2-70 es una versión modificada de la placa DE2 cuyas diferencias radican en la existencia de un dispositivo FPGA de mayor capacidad, y en el aumento de la cantidad de memoria. Debido a estas modificaciones, esta tarjeta puede ser empleada para la realización de diseños más avanzados.

El elemento principal de la placa DE2-70 es el dispositivo 'Cyclone II 2C70 FPGA', al cual se conectan el resto de componentes importantes de la placa.

La Figura 1 muestra una imagen de la tarjeta DE2-70, la cual muestra el diseño de la misma e indica la localización de cada uno de sus conectores y componentes.

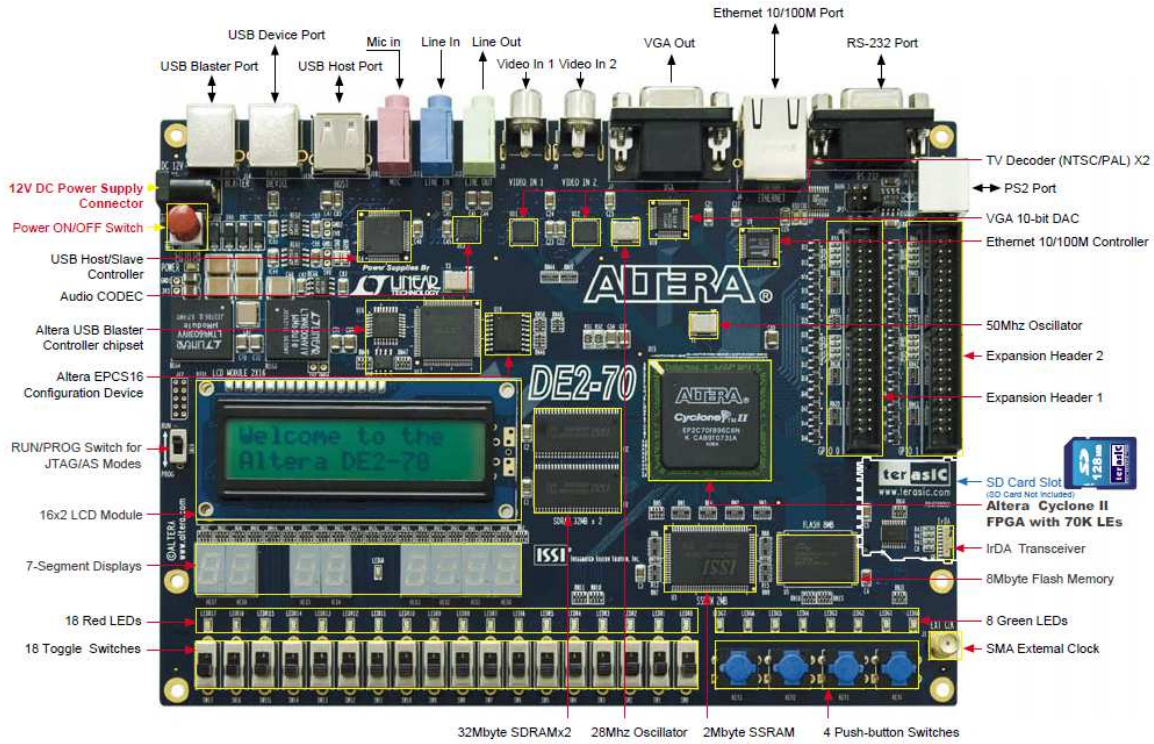


Figura 1. Placa DE2-70 de Altera 1.
(Imagen extraída de [12])

A continuación se presenta el listado de elementos integrados en la placa:

- Dispositivo Altera Cyclone II 2C70 FPGA.
- Elemento de configuración de dispositivos serie Altera - EPCS64.
- Dispositivo USB Blaster para la programación del dispositivo soportando los modos JTAG y Active Serial.
- 2-Mbyte de memoria SSRAM.
- 2 bloques de 32-Mbytes de SDRAM.
- 8-Mbytes de memoria Flash.
- Socket para memoria externa SD.
- 4 pulsadores.
- 18 interruptores.
- 9 diodos LED de color verde.
- 18 diodos LED de color rojo.
- Oscilador de 50 MHz y oscilador de 28,63 MHz como fuente de señales de reloj.
- CODEC de audio de 24 bits con entrada de línea, salida de línea y entrada de micrófono.
- VGA DAC (10-bit high-speed triple DACs) con conector de salida VGA.
- 2 decodificadores de TV (NTSC/PAL/SECAM) y conector de entrada de TV.
- Conector Ethernet 10/100.
- Controlador USB con conectores tipo A y tipo B.
- Conexión RS-232.
- Conector PS/2 para ratón ó teclado.
- Transceptor IrDA.
- 1 conector SMA.
- 2 conectores de expansión de 40 pines con diodo de protección.

10.2.1.2.- Diagrama de bloques

Como se ha comentado anteriormente, el dispositivo Cyclone II 2C70 es el elemento más importante de la tarjeta DE2-70. La Figura 2 muestra la disposición que el resto de elementos siguen alrededor de él.

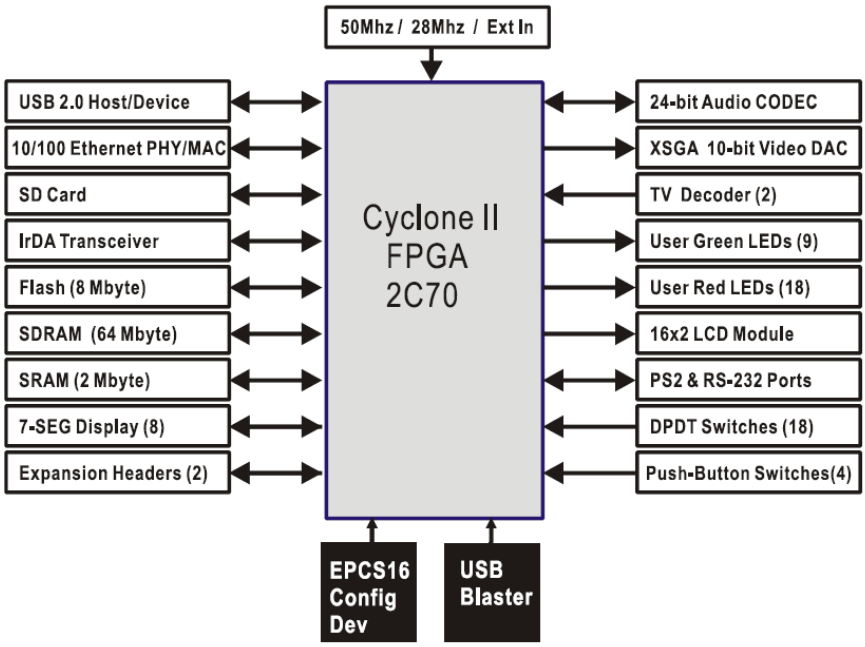


Figura 2. Diagrama de conexiones de la placa DE2-70
(Imagen extraída de [12])

Como se aprecia en la figura anterior, todas las conexiones pasan por el elemento central, lo que ofrece al usuario las máximas posibilidades en cuanto a la programación. De esta manera, el programador puede configurar el dispositivo FPGA para la realización de cualquier diseño.

10.2.1.3.- Dispositivo Cyclone II 2C70 FPGA

A continuación se presentan las principales características que dotan al dispositivo Cyclone II 2C70 FPGA:

- 68.416 Elementos Lógicos (LE's).
- 250 M4K bloques de RAM (bloques de 4 Kbits).
- 1.152.000 bits totales de RAM.
- 150 multiplicadores integrados.
- 4 PLL's.
- 622 pines de entrada/salida a disposición del usuario.

10.2.2.-KIT TRDB_D5M

10.2.2.1.- Contenido del kit

El kit TRDB_D5M trata de una tarjeta con una cámara digital integrada de 5 Megapíxeles que es posible conectar a las tarjetas de Altera DE1 / DE2 / DE2-70. Este kit permite tomar imágenes y guardar las mismas en un PC como archivo BMP ó JPEG (este último sólo en el caso de trabajar con la placa DE2-70).

La Figura 3 muestra una fotografía del kit TRDB_D5M:

Figura 3. Kit TRDB_D5M
(Imagen extraída de [13])

10.2.2.2.- Conexión del kit TRDB_D5M

La conexión entre el kit TRDB_D5M y la tarjeta educacional DE2-70 se realiza como muestra la siguiente figura:

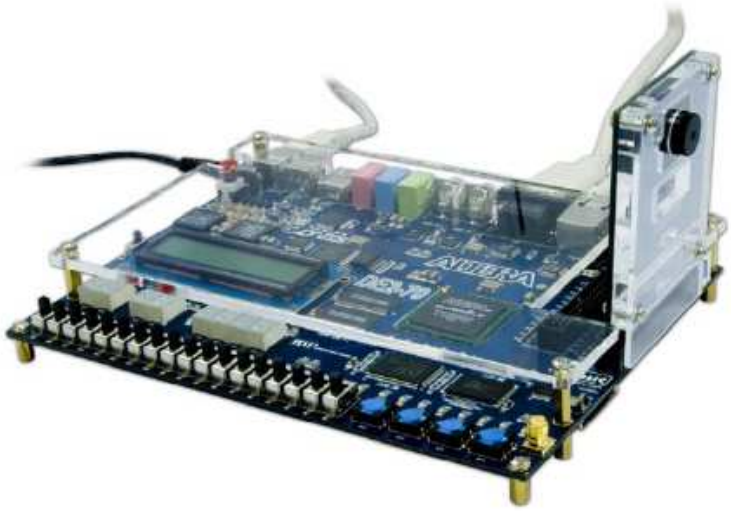


Figura 4. Conexión TRDB_D5M - DE2-70
(Imagen extraída de [13])

Además de la conexión mostrada en la Figura 4, también es posible, mediante un cable IDE conectado a cada uno de los extremos, separar la cámara de la placa DE2-70, con el objetivo de tener mayor flexibilidad a la hora de su manipulación.

10.2.2.3.- Características

El kit TRDB_D5M cuenta con un sensor CMOS como elemento principal. Las características fundamentales con las que cuenta la cámara se resumen en la siguiente tabla:

Parámetro		Valor
Nº de píxeles activos		2592H x 1944V
Tamaño del píxel		2,2µm x 2.2 µm
Matriz de filtro de color		Patrón RGB
Tipo de obturador		Global Reset Release (GRR)
Máxima velocidad de datos/reloj global		96 Mp/s - 96 MHz
Cantidad de imágenes	*Máxima resolución*	Programable con un máximo de 15fps
	VGA (640 x 480)	Programable con un máximo de 70fps
Resolución del Convertidor A/D		12-bit
Sensitividad		1,4 V/lux-sec (550nm)
Rango dinámico del píxel		70,1 dB
SNR MAX		38,1 dB
Voltaje suministrado	*Power*	3,3V
	I/O	1,7V~3,1V

Tabla 1. Parámetros de la cámara

10.2.2.4.- Estructura de la matriz de píxeles

La distribución de píxeles consiste en una matriz de 2752 columnas y 2004 filas, donde cada píxel está identificado según su número de fila y columna. La dirección (0,0) (columna 0, fila 0) representa la esquina superior derecha de la matriz, tal como se ve en la Figura 5.

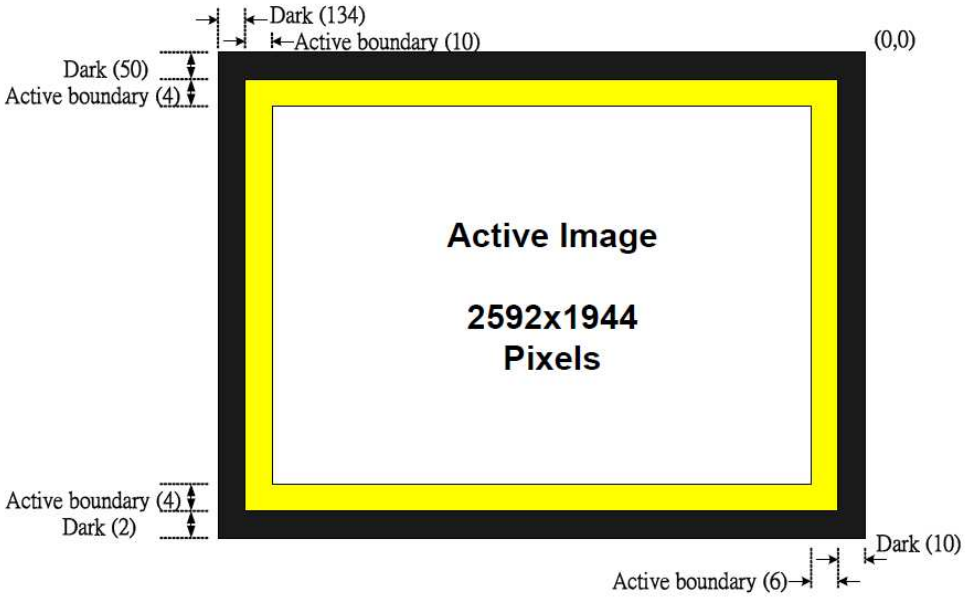


Figura 5. Distribución de la matriz de píxeles
(Imagen extraída de [14])

Como se aprecia en la Figura 5, se tiene una zona activa correspondiente a la matriz central, de tamaño 2592 x1944 (columnas x filas). Dicho matriz central será la imagen obtenida en la salida por defecto. La matriz central está rodeada de una serie de píxeles que limitan la zona activa (Boundary region), siendo esta región, a su vez, rodeada de un borde de píxeles oscuros. La zona límite de la zona activa puede ser usada para evitar los efectos de los bordes cuando se realiza el procesado de color para lograr la imagen resultado de tamaño 2592*1944. Las filas y columnas oscuras se emplean para establecer el nivel de negro en el monitor.

La imagen de salida ofrece una distribución de los píxeles siguiendo un formato basado en el patrón Bayer (conocido como filtro, máscara o mosaico de Bayer). Dicho patrón consiste en una malla cuadriculada de filtros rojos, verdes y azules situada sobre el sensor digital de imagen para hacer llegar a cada fotodiodo una tonalidad de los distintos colores primarios. Interpolando las muestras de varios fotodiodos se obtiene un píxel de color. El mosaico de Bayer se compone por un 50% de filtros verdes, un 25% de rojos y un 25% de azules. Por tanto, interpolando dos muestras verdes, una roja y una azul se consigue un píxel de color. La razón por la cual se utilizan más puntos verdes es debida a que el ojo humano es más sensible a este color. La disposición de los colores, sigue una alternancia entre verde y rojo (G1-R) en una fila, y azul y verde en la fila paralela (B-G2). Los píxeles G1 y G2 emplean el mismo filtro de color, pero son tratados como si fueran colores distintos.

Esta distribución es la mostrada en la Figura 6.

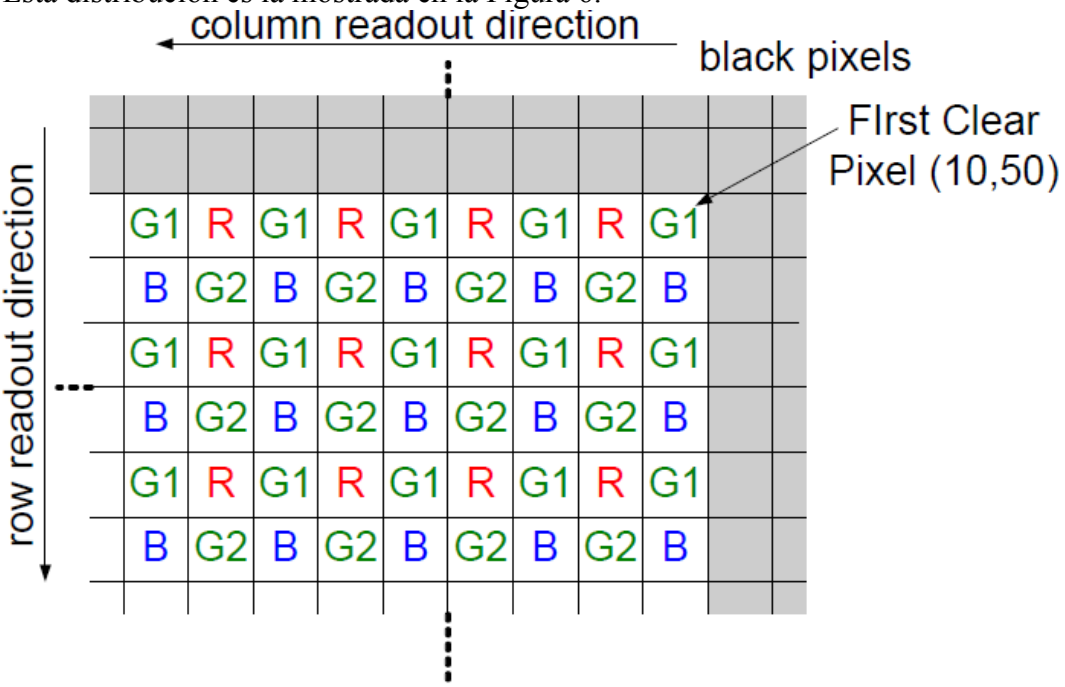


Figura 6. Distribución del patrón de color
(Imagen extraída de [14])

10.2.2.5.- Formato de salida de los datos (modo por defecto)

Para realizar la lectura de los datos de la imagen se hace un barrido progresivo de la misma. La Figura 7 muestra la distribución espacial de la imagen leída.

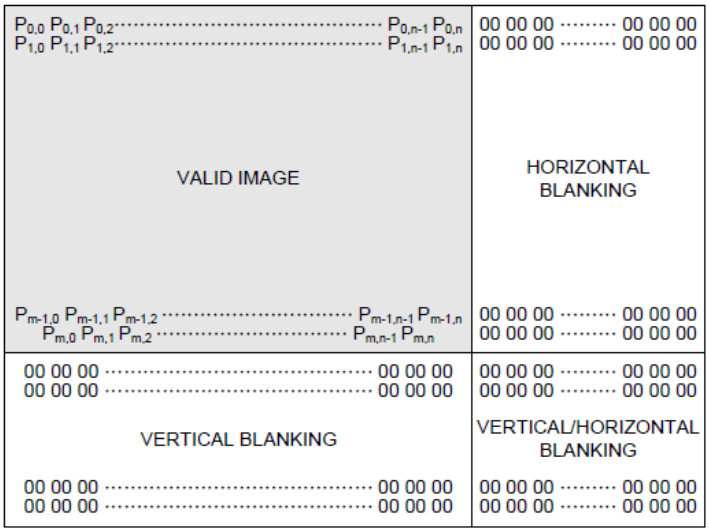


Figura 7. Distribución espacial de la imagen de lectura
(Imagen extraída de [14])

Típicamente, la ventana de lectura incluye sólo la región de píxeles activos. Sin embargo, el usuario tiene la opción de leer los píxeles oscuros que quedan fuera de la región activa. Si se da este caso, se deben dar las consideraciones de cómo interpreta el sensor las zonas oscuras para sus propios propósitos.

10.2.2.6.- Sincronización de los datos de salida

Las imágenes de salida están divididas en fotogramas, los cuales, a su vez, están divididos en líneas. Por defecto, el sensor produce 2592 columnas y 1944 filas por cada fotograma. Las señales FRAME_VALID y LINE_VALID indican los límites en cuanto a fotogramas y líneas respectivamente. Como se muestra en la Figura 8, cuando ambas señales quedan activas, el píxel es válido.

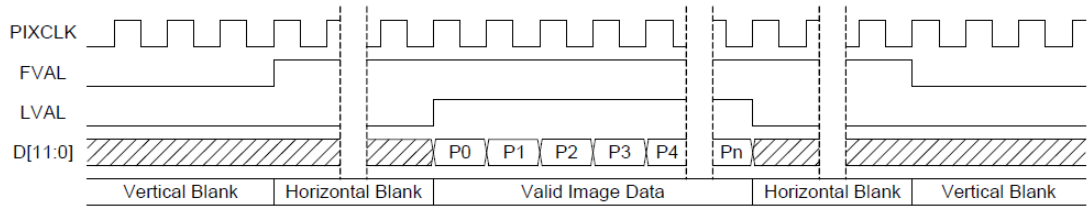


Figura 8. Sincronización de los datos de salida
(Imagen extraída de [14])

La señal 'PIXCLK' representa la señal de reloj de los píxeles y se corresponde con el tiempo necesario para muestrear un píxel del array de datos. Típicamente su valor es de 96 MHz.

10.2.2.7.- Modos de lectura

Por defecto, la resolución en las imágenes de salida es la máxima que puede ofrecer el sistema (2592x1944). Sin embargo, la resolución de salida puede ser minimizada siguiendo dos métodos: Skipping y Binning.

- **Skipping:**

Método que consiste en reducir la resolución realizando una selección de los píxeles procedentes del campo de visión (FOV) de la imagen de salida. La Figura 9 muestra el ejemplo de una reducción con factor 2X.

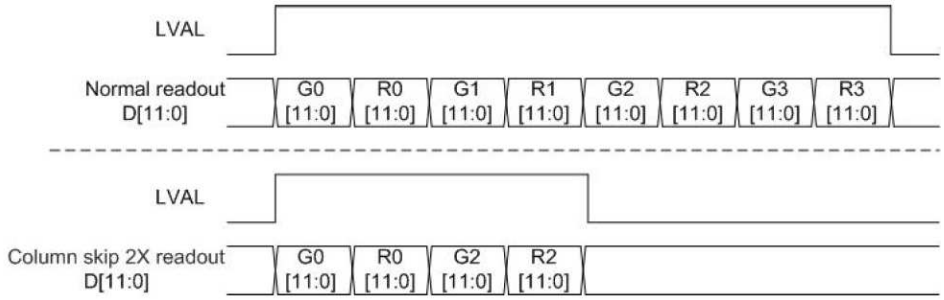


Figura 9. Skipping (2X)
(Imagen extraída de [14])

El método puede ser implementado trabajando con filas y columnas por separado, tal y como muestra la Figura 10.

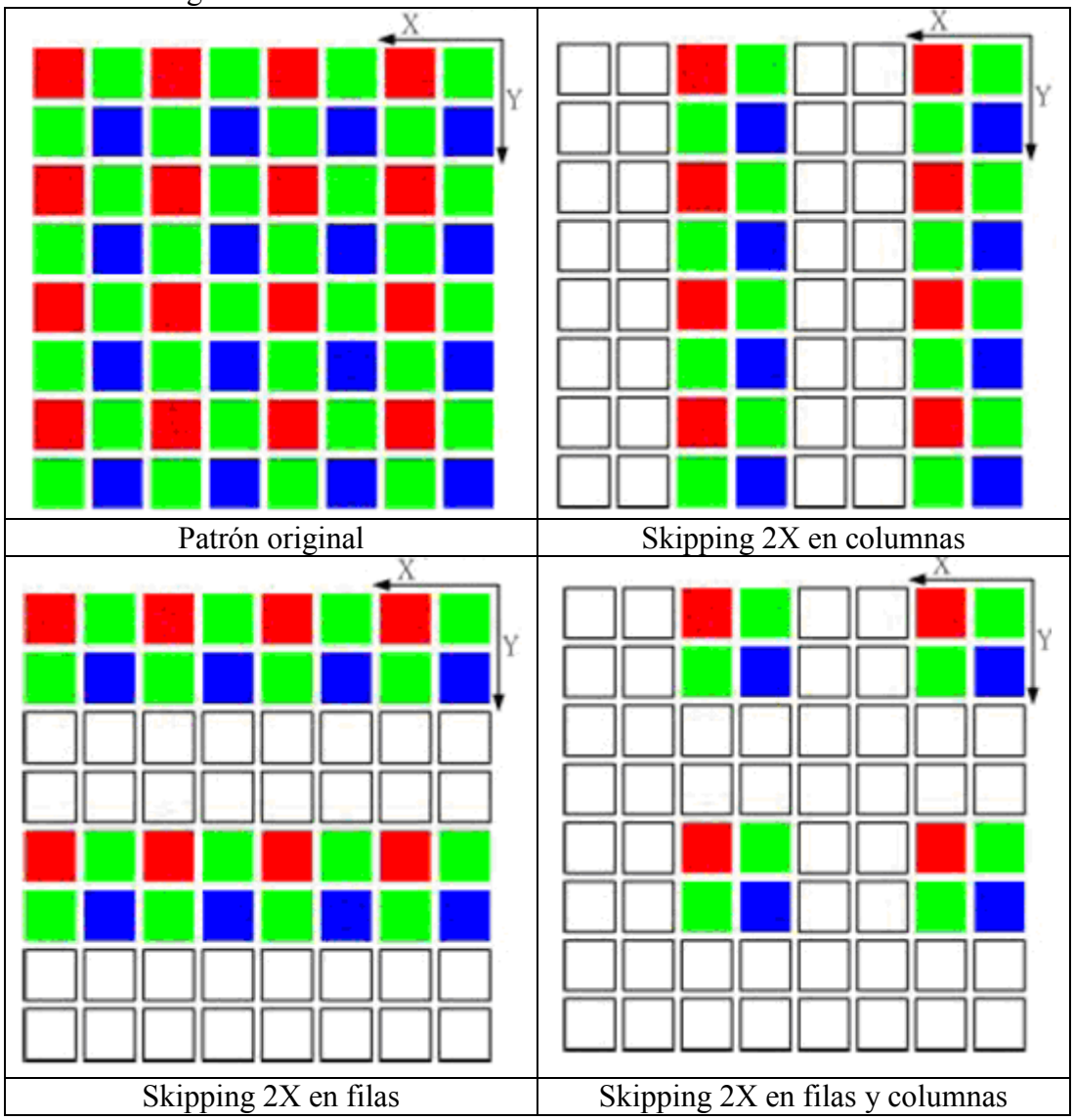
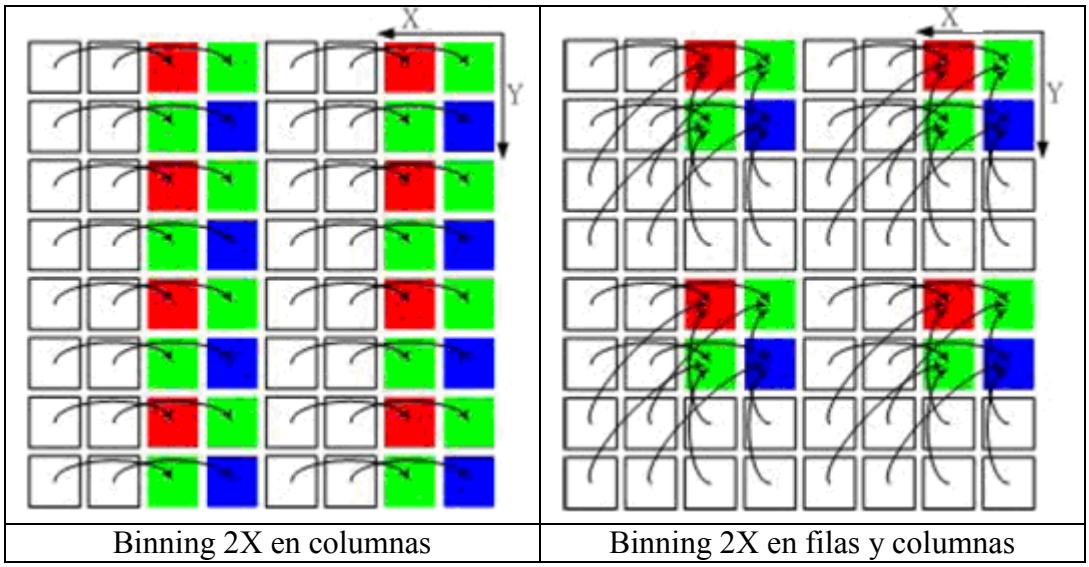


Figura 10. Skipping 2X en filas y columnas
(Imagenes extraída de [14])

- **Binning:**

El método consiste en reducir la resolución combinando píxeles adyacentes del mismo color para producir un único píxel de salida. En este caso, todos los píxeles existentes en el campo de visión contribuyen al resultado de la imagen final. Este método ofrece un mejor resultado en su salida consiguiendo disminuir la cantidad de píxeles. Además, este método permite mejorar el resultado final en condiciones de poca luz sobre la imagen. El píxel de salida puede ser el resultado de una media de los píxeles adyacentes ó una suma de dichos píxeles. Dependiendo de las condiciones de iluminación el proceso se hará de una manera u otra, es decir, en condiciones de baja iluminación, el píxel de salida será resultado de la suma de los píxeles adyacentes, consiguiendo un aumento de la ganancia de color en el mismo. La Figura 11 muestra un ejemplo de aplicación del método Binnig, tomando como base el patrón original mostrado en la Figura 10.



Binning 2X en columnas	Binning 2X en filas y columnas

Figura 11. Binning 2X en filas y columnas
(Imagenes extraída de [14])

- Modos de espejo:

Existen otros dos modos de lectura, que consisten en la realización de un espejo sobre las columnas ó sobre las filas de la imagen, es decir, invertir el orden de lectura bien de las columnas o bien de las filas.

10.2.2.8.-Circuito de procesado de las señales

La figura 12 muestra las etapas que siguen las señales en su procesado. Cada color es procesado independientemente, incluyendo las modificaciones de ganancia y 'offset' también por separado. Las muestras de voltaje correspondientes al array de píxeles existentes en la entrada del proceso pasan, en primer lugar por la etapa de ganancia analógica, la cual produce un factor de ganancia entre 1 y 8. Entonces, se aplica un Offset analógico, y la señal es enviada a un convertidor analógico-digital de 12 bits. A partir de aquí, se trabaja en formato digital, donde se aplica un factor de ganancia entre 1 y 16, y donde se añade un Offset digital con un valor entre -2048 y 2047. El resultado es un píxel formado de 12 bits el cual se envía a través de los puertos D[11:0].

El Offset analógico aplicado es determinado automáticamente por un algoritmo de calibración de nivel de negro o bien puede ser manipulado manualmente por el usuario. El Offset digital es un ajuste más fino, aplicado sobre el Offset analógico.

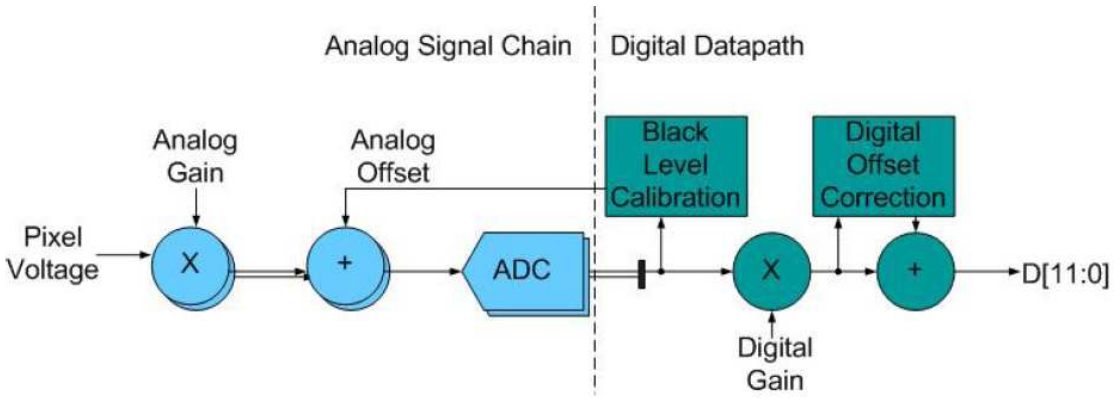


Figura 12. Circuito de procesado
(Imagen extraída de [14])

10.2.2.9.- Modos de adquisición de imágenes

La cámara TRDB_D5M soporta dos modos de adquisición de imágenes: 'electronic rolling shutter' y 'global reset release'.

- **Electronic Rolling Shutter (ERS):**

El modo ERS recoge las imágenes escaneando las filas del sensor en dos ocasiones, en el orden descrito anteriormente. En el primer escaneo, se activa cada una de las filas, iniciando la exposición. En el segundo escaneo, las filas son muestreadas y procesadas, para que, a continuación vuelvan a su estado inicial. Por tanto, la exposición es el tiempo transcurrido entre el primer y segundo escaneo. El tiempo de exposición de cada fila es el mismo, pero a su vez, la exposición de cada fila está ligeramente distanciada en el tiempo, lo que puede causar cortes (saltos) en escenas con movimiento.

Siempre que se cambia al modo ERS, antes de la adquisición del primer fotograma, existe una secuencia 'anti-blooming' la cual resetea todas la filas de la matriz.

- **Global Reset Release (GRR):**

El modo GRR se basa en iniciar la exposición de todas las filas al mismo tiempo. A diferencia del primer escaneo del modo ERS, en este caso todas las filas se resetean simultáneamente. El segundo escaneo se basa en lo mismo que en el caso anterior, así que el tiempo de exposición de cada fila toma diferentes puntos de inicio en el tiempo.

10.2.2.10.- Modos de operación

A continuación se definen los diferentes modos en los que es capaz de trabajar el sistema:

- ERS Continuous: Es el modo activado por defecto. En la salida se obtiene una secuencia continua de fotogramas, según la velocidad (frame rate) definida. El tiempo de exposición es controlado electrónicamente. Se emplea el modo de adquisición ERS.
- ERS Snapshot: Existe un único fotograma en la salida. En el momento en el que se efectúe el disparo del sistema se inicia la exposición. El tiempo de exposición y adquisición de la imagen es controlado electrónicamente. Se emplea el modo de adquisición ERS.

- ERS Bulb: Existe un único fotograma en la salida. Se inicia la adquisición con un disparo del sistema y se necesita un segundo disparo para realizar la lectura de la imagen. Se emplea el modo de adquisición ERS.
- GRR Snapshot: Similar al modo ERS Snapshot pero empleando el modo de adquisición GRR.
- GRR Bulb: Similar al modo ERS Bulb pero empleando el modo de adquisición GRR.

Todos los modos de operación siguen la siguiente secuencia de operaciones:

1.- Esperar al primer 'Trigger', y entonces iniciar la exposición.

2.- Esperar al segundo 'Trigger', y entonces iniciar la adquisición (lectura).

10.2.3.-KIT TRDB_LTM

10.2.3.1.- Contenido del kit

El kit TRDB_LTM provee de todo lo necesario para desarrollar aplicaciones utilizando un panel táctil digital conectado a uno de los modelos de tarjetas DE1/DE2/DE2-70 de Altera. La Figura 13 muestra los contenidos del kit: panel LCD y cable IDE (Integrated Device Electronic) de 40 pines.

Figura 13. Kit TRDB LTM
(Imagen extraída de [15])

10.2.3.2.- Conexión del panel LTM a la placa DE2-70

La conexión del panel LTM a la tarjeta DE2-70 de Altera se realiza de la siguiente manera:

1.- Conectar el cable IDE en la parte trasera de la tarjeta LTM mediante su correspondiente conector.

2.- Conectar el otro extremo del cable IDE en uno de los puertos de expansión de la placa DE2-70.
La conexión queda como se muestra en la Figura 14.

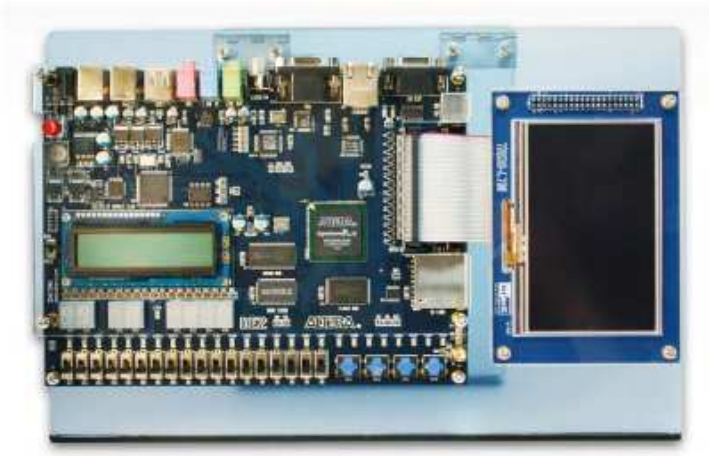


Figura 14. Conexión LTM - DE2-70
(Imagen extraída de [15])

10.2.3.3.- Características

En este apartado se describen las características técnicas fundamentales del panel LTM.

- Está equipado con una pantalla TFT-LCD (Thin Film Transistor-Liquid Crystal Display). Este tipo de pantalla es una variante de las pantallas de cristal líquido (LCD) de matriz activa que usan tecnología de transistor de película delgada (TFT) para mejorar la calidad de su imagen.
- Cuenta de un interfaz RGB de 24 bits en paralelo.
- Tiene capacidad de modificación de contraste, brillo y modulación gamma.
- Existencia de un convertidor analógico-digital para convertir los valores de coordenadas XY de cada punto del panel en un dato digital que es enviado al dispositivo controlador.

La siguiente tabla muestra las características generales de la pantalla que compone el kit TRDB-LTM:

Parámetro	Descripción	Unidades
Tamaño de display (diagonal)	4.3	pulgadas
Relación de aspecto	15:9	-
Área activa (HxV)	93.6 x 56.16	mm
Número de puntos (HxV)	800 x RGB x 480	puntos
Tamaño del punto (HxV)	0.039 x 0.117	mm
Número de colores	16 millones	-

Tabla 2. Características pantalla LCD

10.2.3.4.- Diagrama de bloques

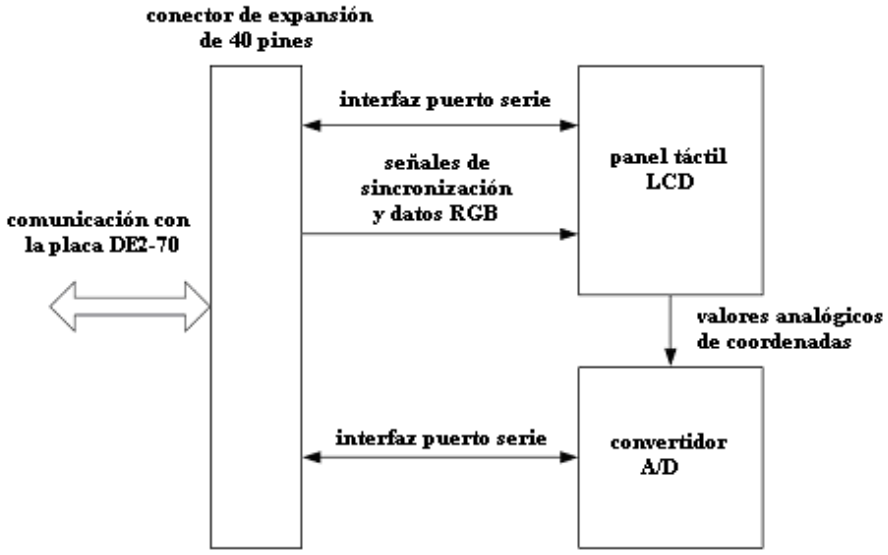


Figura 15. Diagrama de bloques del panel LTM

El panel LTM consta de tres componentes principales: panel LCD táctil, convertidor A/D y conector de expansión de 40 pines. A través del conector de 40 pines se establece la conexión y todas las comunicaciones con la placa DE2-70.

El panel LCD táctil recoge las señales de control directamente desde el dispositivo FPGA y muestra en pantalla las imágenes que le son enviadas. Finalmente, el convertidor A/D convierte las coordenadas XY de cada punto del panel táctil en su correspondiente señal digital, que es enviada a la FPGA a través del puerto de expansión.

10.2.3.5.- Interfaz del puerto serie

El panel LCD táctil está equipado de un driver capaz de controlar la resolución de pantalla, señales de sincronización y circuitos de suministro de alimentación. Para controlar estas funciones, el usuario puede usar la FPGA de la tarjeta y configurar los registros del dispositivo LTM, a través del interfaz del puerto serie.

Además, el convertidor A/D emplea el mismo puerto serie para enviar las señales digitales correspondientes a las coordenadas XY del panel táctil hasta la FPGA.

Como el número de pines de entrada o salida (I/O) del conector de expansión es limitado, es necesario compartir la misma señal de reloj (ADC_DCLK) y la misma señal de habilitación de dispositivos (SCEN) tanto para el controlador del panel LCD como para el convertidor A/D. Para evitar que el control de uno de los dispositivos pueda interferir en el funcionamiento del otro, la señal de habilitación (CS) que llega a la entrada del convertidor A/D ha sido invertida, de tal forma que este dispositivo trabaja con lógica negativa.

El controlador LCD soporta un reloj síncrono a través del interfaz serie para recibir las instrucciones procedentes de la FPGA. La Figura 16 muestra la secuencia de sincronización enviada a través del interfaz serie.

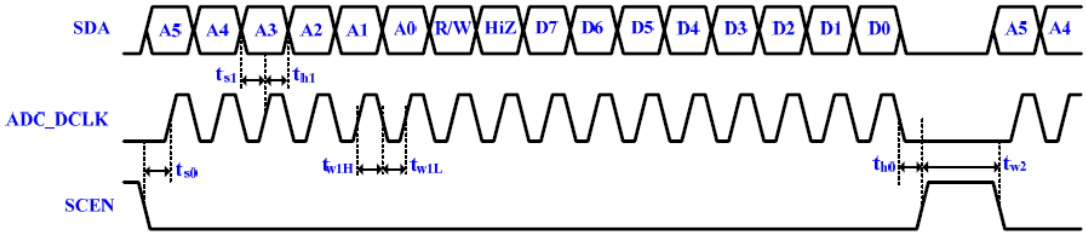


Figura 16. Secuencia de sincronización
(Imagen extraída de [15])

El controlador LCD reconoce el comienzo de la transmisión de datos cuando la entrada de habilitación (SCEN) pasa a nivel bajo. A través de la señal 'SDA' se envía la secuencia de datos que completa la instrucción.

Los primeros 6 bits (A5-A0) especifican la dirección del registro. El siguiente bit significa lectura/escritura, un '0' indica escritura y un '1' indica lectura. Finalmente, los 8 últimos bits (D7-D0) son los bits con la información de control.

10.2.3.6.- Interfaz serie para el convertidor A/D

El kit TRDB-LTM está equipado con el dispositivo AD7843, el cual es un convertidor analógico/digital (ADC) de 12 bits. Este convertidor convierte las coordenadas XY de cada punto del panel táctil en una secuencia de datos digitales. Los valores de las coordenadas son llevados hasta el dispositivo AD7843 a través del interfaz del puerto serie.

La Figura 17 muestra la secuencia de operación típica del interfaz serie sobre el convertidor A/D. La señal de reloj es la que controla la transferencia de información que entra y sale del ADC. Una conversión completa de coordenadas a datos digitales conlleva 24 ciclos de la señal de reloj (ADC_DCLK). Es necesario tener en cuenta que la señal de habilitación del dispositivo (SCEN) es compartida con el controlador del panel LCD, luego dicha señal llega al dispositivo ADC con lógica invertida y, por tanto, se debe realizar de nuevo la inversión de esta señal para que pueda ser usada por el ADC.

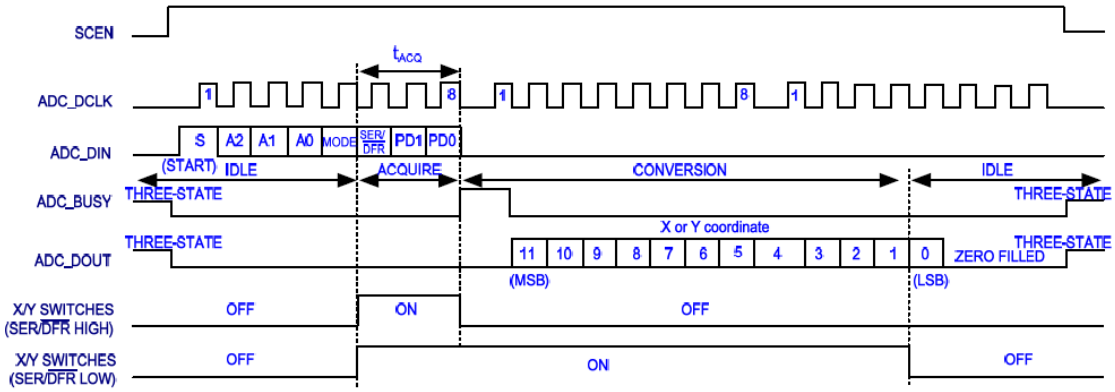


Figura 17. Parámetros temporales del interfaz del puerto serie
(Imagen extraída de [15])

10.2.3.7.- Señales de sincronismo de la pantalla LCD

La Figura 18 muestra los requerimientos básicos temporales de cada fila (horizontal) que es mostrada en pantalla.

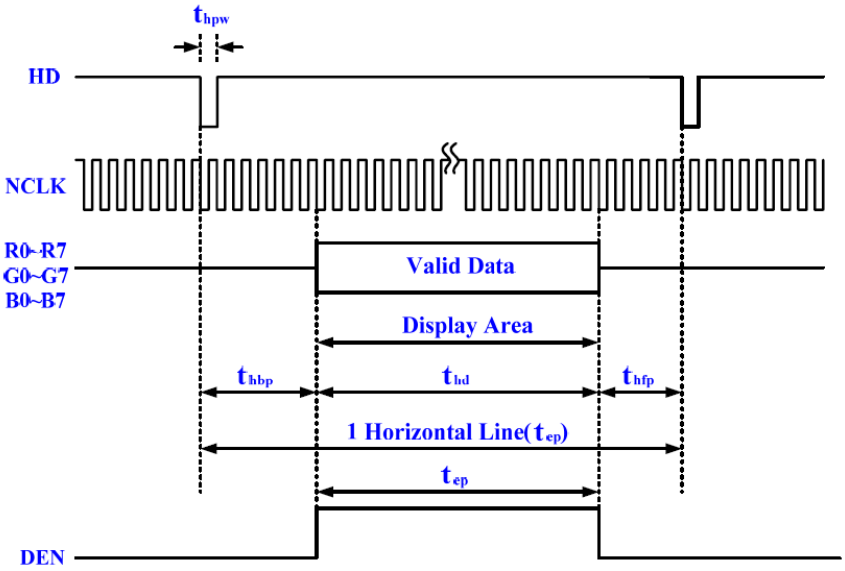


Figura 18. Especificaciones temporales del sincronismo horizontal
(Imagen extraída de [15])

Observando la figura, un pulso activo a nivel bajo de anchura t_{hpw} sirve para obtener el sincronismo horizontal (HD) de las líneas que componen la imagen en el panel. Dicho sincronismo marca el final de una línea y comienzo de la siguiente. Las señales de datos de entrada (RGB) para el panel LCD no son válidas durante un periodo de tiempo llamado

'hsync back porch (t_{hbp})', después de producirse el pulso de sincronismo horizontal. A continuación se encuentra la denominada 'Display Area' con una anchura temporal 't_{hd}', en la cual los datos RGB son llevados a cada píxel de la línea correspondiente. Además, durante el periodo 't_{hd}' la señal de habilitación de datos (DEN) debe realizar la transición a nivel alto. Finalmente, existe un periodo de tiempo llamado 'hsync front porch (t_{hfp})' donde de nuevo las señales RGB no son válidas, antes de producirse el nuevo pulso de sincronismo horizontal que marca el final de la línea actual y comienzo de la siguiente.

Las especificaciones temporales del sincronismo vertical (VD) siguen la misma forma que en el caso del sincronismo horizontal, como se puede ver en la Figura 19. En este caso el pulso de sincronismo vertical indica el final del fotograma y el comienzo del siguiente. Los datos se refieren a la serie de líneas que componen el fotograma.

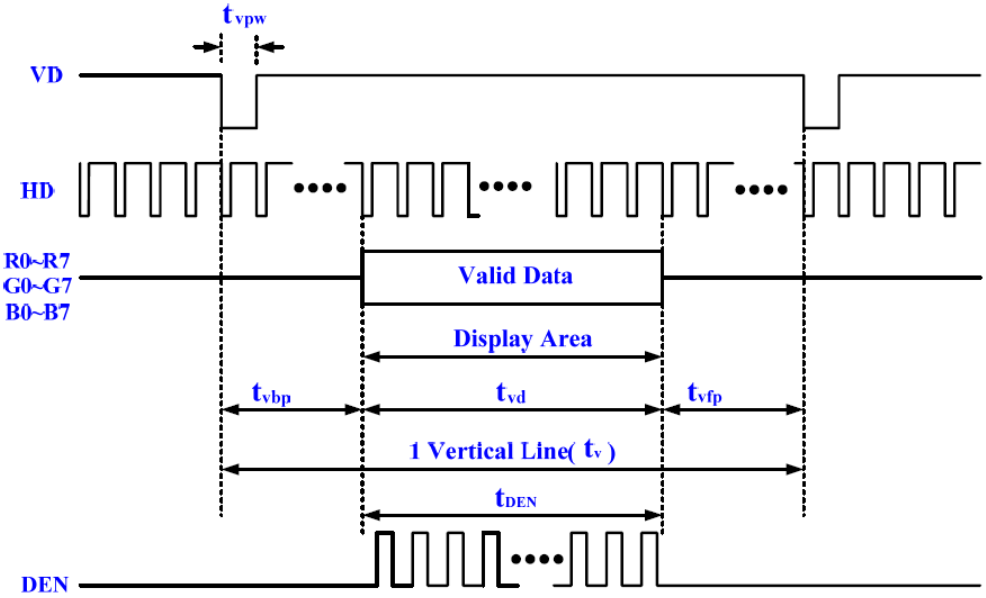


Figura 19. Especificaciones temporales del sincronismo vertical
(Imagen extraída de [15])

10.3.- LENGUAJE DE PROGRAMACIÓN VERILOG HDL

En este apartado no se pretende realizar un manual para el aprendizaje del lenguaje de programación Verilog HDL, sino dar una visión general del mismo, explicando sus características y particularidades respecto a otros lenguajes.

10.3.1.- INTRODUCCIÓN

Verilog es un lenguaje para la descripción de sistemas digitales (HDL: Hardware Description Language).

Existen multitud de lenguajes HDL en el mercado, de hecho inicialmente cada fabricante disponía de su propio lenguaje, sin embargo, la necesidad de unificación ha hecho que en la actualidad fundamentalmente existan dos grandes lenguajes: VHDL y Verilog. Ambos están acogidos a estándares IEEE.

Las herramientas CAD facilitan el diseño de circuitos digitales en base a esquemáticos, los cuales se forman a través de la conexión de componentes básicos o de bibliotecas. Sin embargo, a medida que el sistema aumenta en tamaño o complejidad, aumenta la dificultad en la descripción mediante estos métodos.

Los lenguajes descriptivos de hardware permiten diseñar, de forma abstracta, complejos sistemas digitales que luego pueden ser simulados e implementados en dispositivos programables como FPGA o CPLD.

Uno de los aspectos que llama la atención al visualizar un código en Verilog es la similitud de su sintaxis con la del lenguaje C. Sin embargo, una de las mayores diferencias que presenta este lenguaje es que permite modelar sistemas digitales reales, que funcionan de forma paralela, a diferencia de la ejecución secuencial, típica de un sistema computacional.

La representación de un sistema digital en Verilog se realiza definiendo una serie de módulos, donde cada uno de éstos realiza una tarea específica. Esto es similar al empleo de funciones durante el desarrollo de un programa mediante lenguaje C. Los argumentos de cada módulo son los nombres de las variables de entrada/salida del mismo.

Los diferentes lenguajes de programación son utilizados sobre herramientas (software) de desarrollo que permiten diseñar, verificar, simular y crear los códigos binarios de los respectivos circuitos digitales, que serán cargados en los dispositivos programables.

Los lenguajes HDL deben permitir la simulación, cuyo objetivo es el de verificar el correcto funcionamiento del diseño, es decir, verificar que se cumplen las especificaciones lógicas y temporales. Al mismo tiempo, los lenguajes deben realizar la síntesis lógica, lo que significa traducir el programa a una serie de compuertas y flip-flops permitiendo la formación de bloques que finalmente han de ser colocados y enrutados de forma conveniente en el dispositivo programable.

La Figura 20 resume las principales etapas del diseño digital.

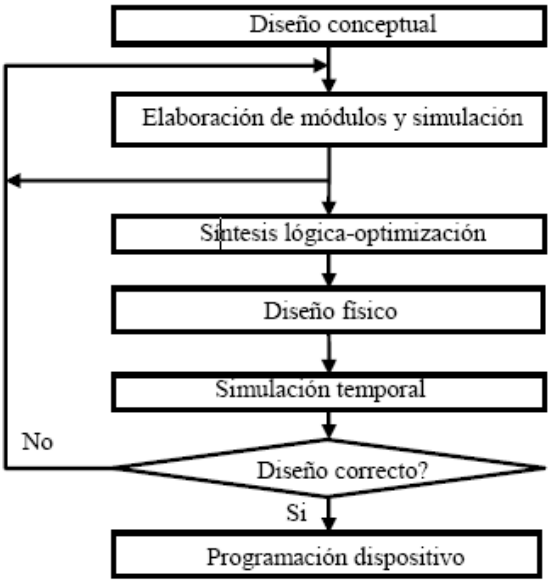


Figura 20. Etapas de diseño
Esquema Obtenido de [17]

10.3.2.- NIVELES DE DESCRIPCIÓN EN VERILOG

Verilog permite que en un mismo diseño se puedan usar diferentes niveles de descripción de sistemas digitales. Existe, en primer lugar, la descripción estructural del diseño en base a componentes básicas y, en segundo lugar, descripciones más abstractas que son enfocadas a describir el comportamiento del sistema.

- **Descripción estructural:**
  - **Nivel de puertas:** corresponde a una descripción de bajo nivel del diseño. El diseñador describe el diseño mediante el uso de primitivas lógicas (AND, OR, NOT, etc…) y conexiones lógicas. Además, se deben definir las propiedades temporales de las diferentes primitivas.
  - **Nivel de transistores:** Verilog permite descripciones a nivel de conmutación de transistores, lo cual permite el modelado de puertas lógicas.

- **Descripción del comportamiento:**

Cuando el diseño presenta cierta dificultad, puede ser muy laboriosa la descripción del diseño a nivel estructural.

Los lenguajes permiten una descripción más abstracta, y a la vez compacta, de la estructura del sistema, ya que puede describirse el comportamiento del mismo, es decir, se describe lo que debe efectuar el sistema, empleando sentencias de lenguaje.

Lo que el programa describe a este nivel son los registros y la transferencia de información entre ellos, debido a esto, este nivel también es denominado RTL (Register Transfer Level).

La descripción del comportamiento de un diseño es realizada mediante la definición de procesos o procedimientos.

10.3.3.- PROCESOS

El concepto de procesos que se ejecutan en paralelo es una de las características fundamentales del lenguaje, siendo este uno de los aspectos diferenciales con respecto a un lenguaje secuencial como el caso de C.

Existen dos tipos de procesos, también denominados bloques concurrentes.

- **Initial:** Este tipo de proceso se ejecuta una sola vez, comenzando su ejecución en tiempo cero. Se trata de un proceso no sintetizable, es decir no puede ser usado en una descripción RTL. Su uso está ligado a la inicialización de variables.
- **Always:** Este tipo de proceso se ejecuta continuamente a modo de bucle. Tal y como su nombre indica, se ejecuta siempre. Este proceso es totalmente sintetizable.

10.3.4.- MÓDULOS Y JERARQUÍAS

Tal y como se comentó en el apartado de introducción, la descripción de un sistema digital mediante lenguaje Verilog está basada en la definición de módulos. Debido a que los módulos pueden reutilizarse, éstos pueden ser usados para construir diseños de mayor complejidad, creando lo que se denomina diseño con jerarquía.

10.3.5.- SINTETIZABILIDAD

El lenguaje Verilog tiene una doble funcionalidad:

1) Realización de simulaciones de sistemas digitales.
2) Descripción de un sistema para su posterior síntesis e implementación en un dispositivo programable.

Si lo que se pretende es construir un diseño el cual ha de ser implementado en un dispositivo programable, se deben respetar ciertas normas, las cuales se enumeran a continuación:

1. No emplear retrasos en el diseño del sistema. Los sintetizadores suelen ignorar estos retrasos, pudiendo haber notables diferencias entre la simulación y la síntesis.
2. No modificar una misma variable en dos procesos diferentes. La modificación de la misma variable en procesos diferentes puede dar lugar a resultados diferentes entre la simulación y la síntesis, en caso de que se pueda sintetizar (problemas de compilación del diseño).
3. Si no se define completamente una variable (tamaño de la misma), el lenguaje supone que conserva el último valor asignado, sintetizando un biestable que almacena su estado.
4. Los sintetizadores no suelen admitir procesos Initial. Asimismo, tampoco suelen admitir los operadores división y resto.
5. Debe tenerse precaución cuando dos asignaciones se realizan en el mismo instante pero una depende de la otra y, por tanto, el orden de ejecución es importante.

10.4.- APLICACIÓN DE2_70_D5M_LTM

10.4.1.- INTRODUCCIÓN

En base al hardware descrito anteriormente, se implementa un sistema en el que la cámara está continuamente adquiriendo imágenes que, a través de la placa DE2-70, son adaptadas para ser mostradas en el panel LTM.

Además, también se realizan otras operaciones aprovechando los recursos que ofrece la placa. En este caso, a través de los displays de 7 segmentos se visualiza un contador del número de fotogramas que va adquiriendo la cámara, a la vez que los diodos LED de color verde están en constante parpadeo. También, cada LED de color rojo es iluminado si se activa su correspondiente interruptor.

A continuación se describe el funcionamiento general de la aplicación, describiendo los principales módulos que participan en el diseño, con el objetivo de comprender las distintas etapas de un sistema de vídeo basado en un sistema programable, desde su etapa inicial de adquisición de imágenes hasta su etapa final, donde dichas imágenes son visualizadas en una pantalla.

Para lograr comprender el funcionamiento de la aplicación se recomienda consultar el esquema lógico resultante del diseño en el capítulo de planos del presente proyecto (ANEXO1. Plano nº1. Esquema lógico de la aplicación DE2_70_D5M_LTM).

Nota: El código Verilog expuesto en los diferentes cuadros de texto a lo largo de este apartado no es el código completo que produce el funcionamiento del sistema, sino que son fragmentos del mismo que son incluidos con el objetivo de ayudar en la comprensión de la aplicación.

10.4.2.- DESARROLLO. ESTRUCTURA DEL DISEÑO

La Figura 1 muestra el diagrama de bloques donde se ven los respectivos pasos que se van cumpliendo con cada uno de los módulos explicados a continuación, relativos al proceso de adquisición de imágenes de la cámara digital, dentro del conjunto global de la aplicación.

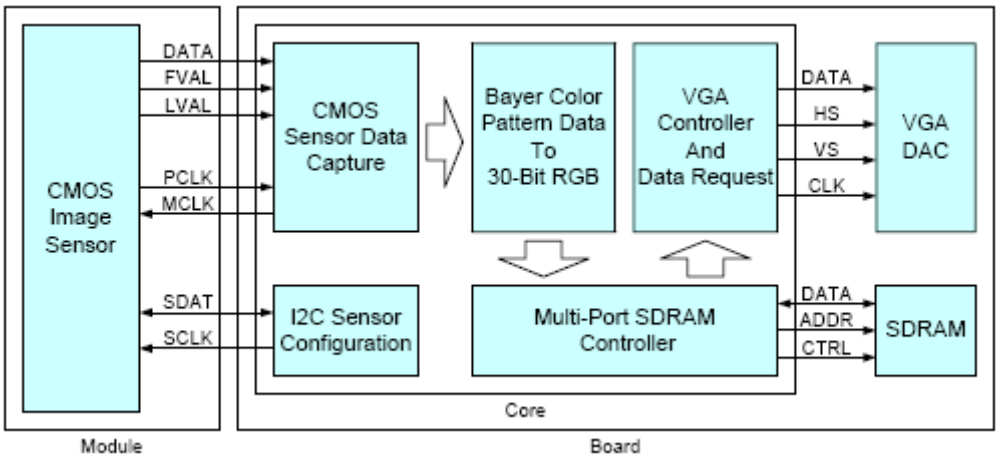


Figura 1. Diagrama de bloques para la adquisición de imágenes
(Imagen extraída de [13])

- **Módulo 'DE2_70'**

Es el módulo principal de la aplicación, el situado en primer lugar en la jerarquía de archivos, del cual dependen el resto de módulos del sistema.

El módulo comienza con la declaración de todos los puertos de comunicación con los que cuenta la tarjeta DE2-70. A través de estos puertos se realizarán las diferentes comunicaciones entre elementos hardware de la placa.

A continuación se presenta, a modo de ejemplo, la definición de puertos del interfaz de memoria flash, definidos como entradas, salidas ó líneas bidireccionales.

```
////////////////////////  Flash Interface  ////////////////////////
inout   [14:0]  FLASH_DQ;          // FLASH Data bus 15 Bits (0 to 14)
inout           FLASH_DQ15_AM1;    // FLASH Data bus Bit 15 or Address A-1
output  [21:0]  oFLASH_A;          // FLASH Address bus 22 Bits
output          oFLASH_WE_N;       // FLASH Write Enable
output          oFLASH_RST_N;      // FLASH Reset
output          oFLASH_WP_N;       // FLASH Write Protect
input           iFLASH_RY_N;       // FLASH Ready/Busy output
output          oFLASH_BYTE_N;     // FLASH Byte/Word Mode Configuration
output          oFLASH_OE_N;       // FLASH Output Enable
output          oFLASH_CE_N;       // FLASH Chip Enable
```

Cuadro de texto 1

Aparte de la definición de puertos mostrada, el objetivo de este módulo es el de realizar la interconexión del resto de módulos que dependen de él, logrando la arquitectura lógica del circuito que posteriormente será implementada en el dispositivo programable. Para ello se realiza la llamada al resto de módulos, definiendo las correspondientes entradas/salidas

internas de los mismos, incluyendo la variable con los datos soportados. A continuación se muestra un ejemplo de esto:

```
Reset_Delay     u1      (
                          .iCLK(iCLK_50),
                          .iRST(iKEY[0]),
                          .oRST_0(DLY_RST_0),
                          .oRST_1(DLY_RST_1),
                           oRST_2(DLY_RST_2)
                        );
```

Cuadro de texto 2

En el anterior cuadro de texto, se ve una llamada al módulo 'Reset_Delay', definiendo las líneas de entrada y salida que éste posee. Por ejemplo, en la primera línea de código, se define una entrada denominada 'iCLK', la cual contiene, entre paréntesis, la variable 'iCLK_50'. De esta manera se consigue mandar la señal de reloj de 50 MHz generada por la placa al módulo 'Reset_Delay'.

De la misma forma que se generan entradas de señales para los distintos módulos, desde el módulo principal se recogen las salidas que el resto producen, bien sea para que éstas actúen como entradas de otros módulos o bien para que sean salidas finales asignadas a un elemento hardware de la placa.

La asignación de una señal de salida a un elemento hardware de la placa se realiza de la siguiente manera:

```
assign  GPIO_1[14]      =       DLY_RST_1;
```

Cuadro de texto 3

El ejemplo mostrado hace que la señal del pulsador 1 de la placa (Key1) sea asignada al pin 14 del conector de expansión conectado al panel LTM.

En resumen, el módulo DE2_70 es el encargado de gestionar todas las comunicaciones del sistema para lograr el correcto funcionamiento de la aplicación.

- **Módulo 'Reset_Delay'**

Para completar la información sobre la interconexión de módulos iniciada en el Cuadro de texto 2, a continuación se muestra la forma de programar para que los módulos secundarios queden conectados con el principal y, por tanto, conectados entre ellos.

```
module  Reset_Delay (iCLK,iRST,oRST_0,oRST_1,oRST_2);
input           iCLK;
input           iRST;
output          oRST_0;
output          oRST_1;
output          oRST_2;
```

Cuadro de texto 4

En primer lugar, se ve como el nombre del módulo debe coincidir al ser llamado (Cuadro de texto 2) y al ser definido (Cuadro de texto 4).

En segundo lugar, en la primera línea de código del Cuadro de texto 4, se define la lista con los argumentos del módulo, realizando un conexionado por nombre con los puertos definidos en la llamada al módulo desde el programa principal.

Por último, en las siguientes líneas de código, sólo queda definir el sentido de las líneas, bien de entrada ó salida (también podría darse el caso de líneas bidireccionales).

En este módulo entra en funcionamiento el pulsador situado más a la derecha de la placa, el denominado 'Key0'. Hay que tener en cuenta que los pulsadores, por defecto, ofrecen una salida a nivel alto y, al ser pulsados, cambian su estado a nivel bajo.

Este pulsador controla tres señales que son conectadas al resto de módulos del sistema, con la función de realizar un reset del mismo.

El siguiente cuadro de texto muestra el código que resume el funcionamiento:

```
if (!iRST)
        begin
                oRST_0 <= 0;
                oRST_1 <= 0;
                oRST_2 <= 0;
        end
```

Cuadro de texto 5

El anterior código significa que al tener un nivel alto (el operador '!' produce una negación lógica) en la entrada 'iRST' las señales de salida 'oRST' quedan a nivel bajo, consiguiendo los siguientes efectos:

- oRST_0: realiza un reset del panel LTM y de la memoria RAM.
- oRST_1: realiza un reset en la configuración del lector CCD y en el módulo de adaptación de los datos en formato RAW a RGB.
- oRST_2: produce un reset en el módulo de inicio de adquisición de información por parte del CCD y en el módulo en el que se preparan las señales para ser mandadas al panel LTM.

- **Módulo 'CCD_Capture'**

Este módulo tiene el principal objetivo de iniciar la adquisición de imágenes por parte del lector CCD.

Consta de un primer proceso en el que se genera la señal que habilita o deshabilita el inicio de la adquisición. Dicha adquisición es controlada por los pulsadores Key3 y Key0. El pulsador Key3 es el que permite comenzar con la adquisición, mientras que una pulsación sobre Key0 produce la parada de la misma si ésta había sido iniciada, además de generar el reset del sistema explicado anteriormente.

El siguiente código es el que resume lo explicado:

CCD_Capture u2 (
.iSTART (!iKEY[3]),
.iEND (!iKEY[0]),
	if (iSTART)
	mSTART <= 1;
	if (iEND)
	mSTART <= 0;

Módulo DE2_70	*Módulo CCD_Capture*

Cuadro de texto 6

El anterior cuadro de texto muestra como las señales correspondientes a los pulsadores Key0 y Key3 son recogidas en sus respectivas variables y enviadas desde el módulo principal hasta el módulo 'CCD_Capture' con el objetivo de generar las señales de habilitación y parada de la adquisición de imágenes.

Un segundo proceso es el que permite definir el tamaño de la matriz en el CCD, es decir, definir el número de puntos de información para la formación de la imagen. Para ello,

se utilizan dos contadores (filas y columnas) que van incrementando su valor hasta llegar al valor máximo, previamente definido.

El último proceso de la adquisición de imágenes consiste en generar las señales de salida con los datos de información de la escena captada, que son enviadas al siguiente módulo para su correspondiente tratamiento.

Aparte del objetivo principal descrito anteriormente, una segunda aplicación que se inicia en este módulo es la de generar un contador que permita visualizar en los displays de 7 segmentos de la placa DE2-70 el número de frames que continuamente está adquiriendo el lector CCD.

El código mostrado a continuación es el que permite realizar esta operación:

```
begin
        if (!iRST)
        Frame_Cont  <=  0;
        else
        begin
                if ( ({Pre_FVAL,iFVAL}==2'b01) && mSTART )
                Frame_Cont <= Frame_Cont+1;
        end
end
```

Cuadro de texto 7

Según el código anterior, se genera la señal de salida que es conectada a la entrada del bloque que controla los displays de 7 segmentos.

- **Módulo 'RAW2RGB'**

El objetivo de este módulo es el de obtener las señales RGB (Red, Green, Blue) necesarias para formar cada píxel de color de la imagen.

Cada punto de la matriz del sensor CCD recoge la cantidad de luz equivalente para formar la imagen de la escena que está enfocando. Previamente, por delante del sensor CCD se coloca el filtro o mosaico de Bayer, explicado en el punto '2.2.2.- Estructura de la matriz de píxeles' el cual consigue que a cada fotodiodo del sensor le llegue luz con una tonalidad de los distintos colores primarios. Por tanto, el módulo anterior, 'CCD_Capture', manda la información de cada celda de la matriz, y es el módulo 'RAW2RGB' quien interpreta los datos recibidos formando las distintas señales de las componentes de color RGB (Red, Green, Blue).

Una vez que se tienen las tres componentes de color separadas, actuando en conjunto con los interruptores de la placa, es posible realizar diferentes combinaciones para lograr diferentes efectos de color de la imagen final. El Cuadro de texto 8 muestra lo explicado.

```
if (iSWITCH_0)
        begin
                oRed    = rRed;
                oGreen  = rGreen;
                oBlue   = rBlue;
        end
else if (iSWITCH_1)
        begin
                oRed    = rBlue;
                oGreen  = rRed;
                oBlue   = rGreen;
        end
else if (iSWITCH_2)
        begin
                oRed    = rGreen;
                oGreen  = rBlue;
                oBlue   = rRed;
        end
```

Cuadro de texto 8

En el cuadro de texto anterior, se ve como el efecto de tres interruptores de la placa alteran la información de las salidas correspondientes a las señales RGB. Si se cumple la primera condición, el aspecto de la imagen de salida es el natural de la escena real que este capturando la cámara. En las dos restantes condiciones, el efecto en la imagen final es el de producir alteraciones de color con respecto a la imagen original. Mediante estas dos últimas combinaciones, por ejemplo, los posibles objetos de color rojo pasan a ser de color azul ó verde, según el interruptor accionado.

- **Módulo 'I2C_CCD_Config'**

En este módulo se incluyen los parámetros de configuración del sensor de imagen para definir el uso del mismo.

De esta manera, a través del bus I2C, se configura el modo de lectura empleado, los parámetros del circuito de procesado de señales y el modo de adquisición de imágenes.

El funcionamiento del bus I2C consiste en realizar una transmisión serie, a través de la conexión realizada entre la tarjeta DE2-70 y el kit TRDB-D5M, por la cual se mandan todos los datos de configuración necesarios para el sensor de imagen.

- **Módulo 'SEG7_LUT_8'**

Este módulo es incluido para completar la aplicación iniciada en el módulo 'CCD_Capture', que consiste en implementar un contador que realice la cuenta del número de fotogramas que continuamente está adquiriendo la cámara.

Por tanto, este módulo realiza la configuración de los displays de 7 segmentos de la tarjeta DE2-70, consiguiendo mostrar el resultado del contador a través de ellos. A la salida del módulo se tienen las ocho señales que controlan cada uno de los ocho displays con los que cuenta la placa.

- **Módulo 'Sdram_Control_4Port'**

En esta aplicación existen dos bloques de memoria RAM, los cuales permiten almacenar de forma temporal los datos procedentes del conjunto de módulos encargados de realizar la adquisición de imágenes. El objetivo de almacenar temporalmente los datos de las señales RGB es el de conseguir formar el patrón de color mostrado en la Figura 6 del punto '2.2.2.- Estructura de la matriz de píxeles'.

El primer bloque de memoria RAM tiene como entradas las señales correspondientes a las componentes R y G, mientras que el segundo de los bloques tiene como entradas las señales de las componentes G y B. Además, estos bloques cuentan con las entradas necesarias para que, internamente cada uno de ellos, pueda gestionar la organización de un sistema de memoria RAM, en cuanto a las direcciones de las posiciones de memoria y a los datos que van a ser guardados en dichas posiciones.

La salida de cada uno de estos bloques está conectada a las respectivas entradas del módulo 'Touch_tcon' el cual controla el funcionamiento del kit TRDB_LTM.

Por tanto, el funcionamiento de los bloques de memoria RAM consiste en ir almacenando la sucesiva información que tiene a su entrada siguiendo el método FIFO (Fisrt In First Out), de forma que ésta es ofrecida al bloque de gestión del panel LTM de forma ordenada, logrando una correcta distribución espacial de la información en pantalla.

- **Módulo 'Touch_tcon'**

Tal como se comenta anteriormente, este modulo es el encargado de gestionar el kit TRDB_LTM generando las señales necesarias para reproducir las imágenes en pantalla.

Como señales de entrada, el módulo recibe las señales procedentes de cada uno de los bloques de memoria RAM con los datos de información de color que previamente han sido recogidos por la cámara.

Dentro del módulo 'Touch_tcon' se definen los parámetros que permiten iniciar el barrido de la pantalla que genera la imagen final. Para ello se define el número de líneas en pantalla, así como los parámetros necesarios para gestionar las señales de sincronismo horizontal y vertical.

Al igual que en el caso de la cámara, la imagen final es una matriz en la que el número de filas y columnas define su tamaño y, por tanto, el número de puntos de la imagen. El número de puntos en el panel del kit TRDB_LTM es 800xRGBx480. Esto quiere decir que se tiene una matriz de tres dimensiones, cuyo resultado en la imagen final es que cada punto del panel tiene su correspondiente contribución de las tres componentes de color RGB. Interpolando dichas contribuciones en la medida necesaria se obtiene cada píxel de color.

El Cuadro de texto 9 resume el funcionamiento de la formación de la imagen en el panel LTM.

```
if (x_cnt == (H_LINE-1))
            x_cnt <= 11'd0;
else
             x_cnt <= x_cnt + 11'd1;

if (y_cnt == (V_LINE-1))
              y_cnt <= 10'd0;
else
               y_cnt <= y_cnt + 10'd1;
begin
               oLCD_R <= read_red;
               oLCD_G <= read_green;
               oLCD_B <= read_blue;
end
```

Cuadro de texto 9

El funcionamiento mostrado en el anterior código se basa en tener un contador que constantemente se incrementa en una unidad, formando las filas que se corresponden con las líneas en sentido horizontal. Cuando dicho contador llega a la última línea de pantalla se reinicia, formando lo que se denomina sincronismo horizontal.

Lo mismo ocurre en el sentido vertical, existe un contador que va generando las columnas de la matriz y, al llegar a la última, se reinicia formando el denominado sincronismo vertical.

De esta manera, se generan todos los puntos de la matriz de la imagen, donde cada punto recibe las correspondientes señales de color, tal como se ve en la última parte del código, formando la imagen final.

- **Módulo 'lcd_3wire_config'**

Tiene la mima función que el módulo 'I2C_CCD_Config' para el caso del sensor de imagen, es decir, mediante este módulo se generan los datos de configuración del panel LTM.

CAPÍTULO 11.

CONCLUSIONES Y LÍNEAS FUTURAS

11.- CONCLUSIONES Y LÍNEAS FUTURAS

11.1.- CONCLUSIONES

Una vez llegado al final del proyecto parece clara la importancia de los sistemas digitales y, en concreto, los dispositivos de lógica programable. Se hace mención, durante el Capítulo 2 del presente proyecto, a su implicación en los diferentes campos de la industria, así como su constante evolución y ventajas respecto a otras tecnologías. Sin embargo, dicho capítulo no habla de las ventajas que la utilización de estos dispositivos supone en la etapa inicial de la ingeniería, es decir, la educación y formación de ingenieros.

De todas las características que un dispositivo programable tiene, es de vital importancia su capacidad de ser reprogramado un número ilimitado de veces. Esta característica dota al estudiante de la capacidad de aprender a base de errores, que muchas veces es como mejor se aprende. De esta forma, el estudiante tiene la capacidad de utilizar un sistema de gran complejidad tecnológica, con todas las ventajas y funcionalidades que esto conlleva, como si tuviera una hoja de papel "escrita a lápiz", es decir, si no le gusta lo que tiene, se puede borrar y volver a escribir.

Respecto al material de trabajo, es importante realizar como primer paso un detallado estudio de mercado que permita analizar los productos ofertados por las distintas compañías. Hay que tener en cuenta que el material elegido es el que pone las facilidades y, en ocasiones, los límites de los diseños que se pretendan implementar en el laboratorio.

En este proyecto, se trabaja con dispositivos programables de la compañía Altera®, si bien el trabajo con dispositivos de distintas compañías es similar. Aunque cada fabricante ofrece su propio entorno de desarrollo, la forma de proceder es parecida, de hecho, los distintos paquetes de software son utilizados bajo la utilización de herramientas estandarizadas, tales como los lenguajes de programación VHDL y Verilog.

Uno de los objetivos del proyecto era el de adaptar los guiones de prácticas de las respectivas asignaturas para la utilización del software Quartus II. Este software está dotado de todas las capacidades que caracterizan al software anterior Max+Plus II, con la diferencia de estar preparado para su utilización con los dispositivos actuales más potentes del mercado.

Los guiones de prácticas realizados para la titulación 'Ingeniería de Telecomunicación' ya han sido utilizados con éxito por los alumnos en las prácticas de la asignatura 'Circuitos Electrónicos Digitales' del tercer curso de la mencionada titulación.

El contenido de los guiones realizados para esta titulación es una recopilación del conjunto de guiones realizado para la asignatura 'Estructura y Tecnología de Computadores' de la titulación 'Ingeniería Técnica en Informática de Gestión'. Esto es debido a la diferencia de horas de prácticas de una titulación y otra.

La aplicación seleccionada para ser mostrada y analizada en el proyecto, mediante el correspondiente kit educativo, resulta de especial interés para su estudio en la titulación 'Ingeniería Técnica de Telecomunicación Especialidad Sonido e Imagen' debido a la cantidad de campos que cubre vistos a lo largo de la carrera. Mediante dicha aplicación, aparte de profundizar en los dispositivos de lógica programable, se realiza el estudio de las características de una cámara digital y de un panel táctil basado en una pantalla LCD.

Además, se estudia y aprende la programación en el lenguaje de descripción hardware Verilog HDL. Todo ello, permite analizar paso por paso un sistema completo de vídeo desde su etapa de adquisición hasta su posterior reproducción.

En este punto, se considera un defecto detectado en la página Web de la compañía Altera, desde donde es posible descargar el diseño utilizado en la aplicación y otros diseños de ejemplo, dado que dichos diseños únicamente se encuentran en leguaje de programación Verilog y no en VHDL, a pesar de la amplia cantidad de usuarios que emplea VHDL. Además, otro defecto importante, es que en dichos diseños prácticamente no se incluye ningún tipo de comentario explicativo que facilite el entendimiento y comprensión del código.

Después de la experiencia adquirida tras el trabajo realizado con la aplicación, cabe resaltar la dificultad encontrada a la hora de entender y trabajar con el lenguaje Verilog. Se trata de un lenguaje que trabaja de manera muy distinta a la forma de programar en un lenguaje de software tal como C ó MATLAB al que se está acostumbrado. Además, a pesar de que su sintaxis es similar a la de C, presenta un aspecto muy abstracto, de hecho, una frase que representa esto, encontrada al buscar información sobre este lenguaje es: 'Cuanto más abstracto sea el código, más rápido funciona el sistema'. Esto puede ser una ventaja para programadores avanzados que buscan la máxima efectividad del sistema, pero supone ser una herramienta dura y difícil de aprender para personas que se encuentran en su etapa de iniciación.

11.2.- LÍNEAS FUTURAS

Este proyecto tiene dos partes diferenciadas. En primer lugar está la serie de capítulos centrados en la realización de un laboratorio de prácticas, en el que se realiza la evolución en el trabajo con el software Max+Plus II al software Quartus II, así como también se pasa de utilizar la tarjeta educacional UP2 a la tarjeta DE2, todo ello de la compañía Altera. En segundo lugar, se muestra la utilización de elementos de lógica programable para la realización de tareas más complejas como es el tratamiento de vídeo.

En referencia a la primera parte, de cara al futuro, de la misma manera en la que se ha realizado una evolución en el software y en el hardware disponible en el laboratorio para la creación de un conjunto de prácticas, la tecnología y, en concreto, los fabricantes de dispositivos lógicos programables constantemente desarrollan nuevos productos que son lanzados al mercado. Debido a este motivo, mediante periódicos estudios de mercado se tendrá la capacidad de conocer las evoluciones que las respectivas compañías ofrezcan, con el objetivo de poder realizar sucesivas modernizaciones del material existente en el laboratorio.

En cuanto al material creado para que sea utilizado por los alumnos (manuales y guiones de prácticas) hay que tener en cuenta que se trata de iniciar a los alumnos en el manejo de dispositivos lógicos programables, sin que éstos requieran ningún conocimiento previo. Por ello, sus contenidos no han de ser excesivamente densos ni tendrán que abarcar todos los campos tratables. Por este motivo, y añadiendo que este material se ha creado siguiendo una particular manera de trabajar, dicho material quedará abierto a posibles mejoras que puedan surgir a los alumnos en el desarrollo de sus prácticas.

Como se ha comentado, el contenido de los guiones de prácticas es una introducción a los dispositivos lógicos programables, por lo que es posible profundizar mucho más en este campo. Para ello serían necesario asignaturas con temarios más avanzados en cursos posteriores o bien que el alumno siga sus propios medios de aprendizaje. Particularmente, es interesante para la persona que quiera ampliar sus conocimientos en este tema, profundizar en lenguajes de programación de descripción hardware tales como VHDL o Verilog, ya que son herramientas estandarizadas y muy potentes a la hora de adentrarse en el diseño de sistemas de mayor complejidad que los presentados en las respectivas prácticas.

Respecto a la segunda parte del proyecto, en este caso se ha presentado una aplicación que trata de un sistema de vídeo. Esto es sólo un ejemplo de implementación. Por tanto, se puede hablar de multitud de líneas futuras en cuanto a aplicaciones se refiere, ya que esto depende de las necesidades y funcionalidades que el usuario pretenda dar a su sistema, además de su propia imaginación.

En concreto para la aplicación presentada, a continuación se plantean posibles ampliaciones que doten al sistema de mayores funcionalidades:

- Permitir a la cámara del kit TRDB_D5M, a través de los pulsadores o interruptores de la tarjeta DE2-70, las funcionalidades que una cámara digital convencional posee, tales como controlar de forma manual el zoom aplicado sobre la escena.

- Mediante la combinación del diseño mostrado en la aplicación del presente proyecto con la de otros posibles diseños relacionados con temas de procesado digital de imagen, conseguir la realización de efectos tales como el control del contraste, emborronado de la imagen, obtención de bordes de objetos, etc.

- Al sistema de adquisición y reproducción de vídeo, se le puede adjuntar un sistema de adquisición y reproducción de audio, a través de los puertos de entrada/salida de audio que la tarjeta DE2-70 dispone, obteniendo un sistema audiovisual completo.

- Una curiosa aplicación de mayor complejidad que podría diseñarse tomando como base la aplicación presentada, sería la de implementar un puzzle digital. Esto consistiría en tomar una imagen digital por medio de la cámara que sea almacenada en la memoria flash de la tarjeta DE2-70 para que, posteriormente, esta imagen sea volcada en el panel LCD y sea dividida, por ejemplo, en una matriz 3x3 formando las piezas del puzzle. A continuación un proceso implementado sobre el dispositivo programable haría que se desordenaran las piezas aleatoriamente, para que el usuario, pulsando sobre el panel táctil, pueda volver a recomponer la imagen.

CAPÍTULO 12.

BIBLIOGRAFÍA

12.- BIBLIOGRAFÍA

12.1.-REFERENCIAS BIBLIOGRÁFICAS

[1] DIGITAL DESIGN AND IMPLEMENTATION WITH FIELD PROGRAMABLE DEVICES. Zainalabedin Navabi. Ed. Kluwer Academic Publishers Group (2005).

[2] RAPID PROTOTYPING OF DIGITAL SYSTEMS. James O. Hamblen and Michael D. Furman. Ed. Kluwer Academic Publishers Group (2001).

[3] Proyecto Final de Carrera: DESARROLLO DE UNA APLICACIÓN PARA DISPOSITIVOS LÓGICOS PROGRAMABLES ORIENTADA A LA TITULACIÓN DE INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN. Emilio Urruchi Celada (2002)

[4] Instituto Nacional de Tecnología Industrial. Comisión Europea en Argentina INTI. Proyecto "ESTADO DEL ARTE DE LA TECNOLOGÍA FPGA". Eduardo Boemo Scalvinoni (2005).

[5] Apuntes de la asignatura "DISPOSITIVOS LÓGICOS PROGRAMABLES". Universidad Politécnica de Cartagena. Andrés Iborra, Juan Suardiaz (2003).

[6] Apuntes de la asignatura "DISPOSITIVOS LÓGICOS PROGRAMABLES". Universidad de Oviedo. Juan Carlos Campo.

[7] Apuntes de la asignatura "MICROELECTRÓNICA". Universidad Francisco de Paula Santander. Gabriel Sánchez Suárez.

[8] QUARTUS II HANDBOOK, VOLUME 1. Altera Corporation (2007).

[9] DE2 DEVELOPMENT AND EDUCATION BOARD. USER MANUAL. Altera Corporation (2006).

[10] GETTING STARTED WITH ALTERA's DE2-70 BOARD. Altera Corporation (2007).

[11] DE2-70 APPLICATION NOTE. Terasic Technologies (2007).

[12] DE2-70 DEVELOPMENT AND EDUCATION BOARD. USER MANUAL. Terasic Technologies (2009).

[13] TRDB-D5M USER GUIDE. Terasic Technologies (2008).

[14] TRDB-D5M HARDWARE SPECIFICATIONS. Terasic Technologies (2008).

[15] TRDB-LTM USER GUIDE. Terasic Technologies (2007).

[16] MANUAL VERILOG. Jorge Chávez (1999).

[17] VERILOG LANGUAJE REFERENCE MANUAL. Open Verilog International (1996)

[18] MANUAL DE USUARIO DE MOWAY. Bizintek Innova (2007)

[19] MICROCHIP PIC16F87XA DATA SHEET. Microchip Technology (2003).

[20] MANUAL DE USUARIO DEL MÓDULO BZI-RF2GH4. Bizintek Innova (2007).

[21] NRF24L01 PRODUCT SPECIFICATION. Nordic Semiconductors (2007).

12.2.- REFERENCIAS EN LA RED

- ALTERA CORPORATION
  www.altera.com
- ACTEL
  www.actel.com
- ATMEL CORPORATION
  www.atmel.com
- CYPRESS SEMICONDUCTOR
  www.cypress.com
- LATTICE SEMICONDUCTOR
  www.latticesemi.com
- QUICKLOGIC CORPORATION
  www.quicklogic.com
- XILINX
  www.xilinx.com
- MICROCHIP TECHNOLOGY
  www.microchip.com
- MOWAY ROBOT
  www.moway-robot.com

CAPÍTULO 13.

PRESUPUESTO

13.- PRESUPUESTO

El presupuesto queda dividido en dos partes. En primer lugar se calcula la cantidad que supone la creación de los respectivos manuales y conjuntos de guiones de prácticas. En la segunda parte se calcula la cantidad necesaria para lograr la implementación de la aplicación presentada.

13.1.- LABORATORIO DE PRÁCTICAS

El coste total tasado para la confección del laboratorio de prácticas de las respectivas titulaciones se ha calculado sin tener en cuenta el hardware empleado. El gasto a realizar para la adquisición de las tarjetas educacionales DE2 de Altera corre por cuenta del interesado.

Por tanto, el coste total es el resultado del valor de la mano de obra más la cuantía de las obligaciones sociales.

- **Capítulo 1. Coste de mano de obra / salario**

CONCEPTO	Nº HORAS	SUELDO / HORA	TOTAL
1 Ingeniero Técnico	240	30 €	**7.200 €**

- **Capítulo 2. Obligaciones sociales**

CONCEPTO	PORCENTAJE (%)	VALOR (€)
Contingencias Generales	28,30	2038 €
Desempleo	8,30	598 €
Fondo de Garantía Salarial	0,20	14 €
Formación Profesional	0,70	50 €
Base de Accidentes de Trabajo	1,65	119 €
TOTAL	39,15 %	**2819 €**

- **Cálculo total de los respectivos capítulos:**

CAPÍTULO 1…… 7.200 €

CAPÍTULO 2…… 2.819 €

TOTAL CAPÍTULOS…….. **10.019 € ***

El total del presupuesto asciende a la cantidad de *Diez mil diecinueve Euros* (I.V.A. no incluido)

* Este valor no incluye el I.V.A

13.2.- APLICACIÓN ESPECÍFICA

El coste total tasado se divide en los siguientes capítulos:

- Capítulo 1: Coste de materiales
- Capítulo 2: Coste de mano de obra / salario
- Capítulo 3: Obligaciones sociales

- **Capítulo 1. Coste de Materiales**

CONCEPTO	CANTIDAD	PRECIO UNITARIO	TOTAL
Kit Tarjeta DE2-70 + Cámara Digital 5M + Panel LCD 4.3"	1	430 €	**430 €**

El precio desglosado de cada uno de los elementos que componen el kit es el siguiente:

CONCEPTO	CANTIDAD	PRECIO UNITARIO	TOTAL
Tarjeta DE2-70	1	242 €	242 €
Cámara Digital 5 Megapíxeles	1	63 €	63 €
Panel LCD 4.3 Pulgadas	1	125 €	125 €

- **Capítulo 2. Coste de Mano de Obra / Salario**

CONCEPTO	Nº HORAS	SUELDO / HORA	TOTAL
1 Ingeniero Técnico	80	30 €	**2.400 €**

- **Capítulo 3. Obligaciones Sociales**

CONCEPTO	PORCENTAJE (%)	VALOR (€)
Contingencias Generales	28,30	679
Desempleo	8,30	199
Fondo de Garantía Salarial	0,20	5
Formación Profesional	0,70	17
Base de Accidentes de Trabajo	1,65	40
TOTAL	39,15 %	**940 €**

- **Cálculo total de los respectivos capítulos:**

CAPÍTULO 1…… 430 €

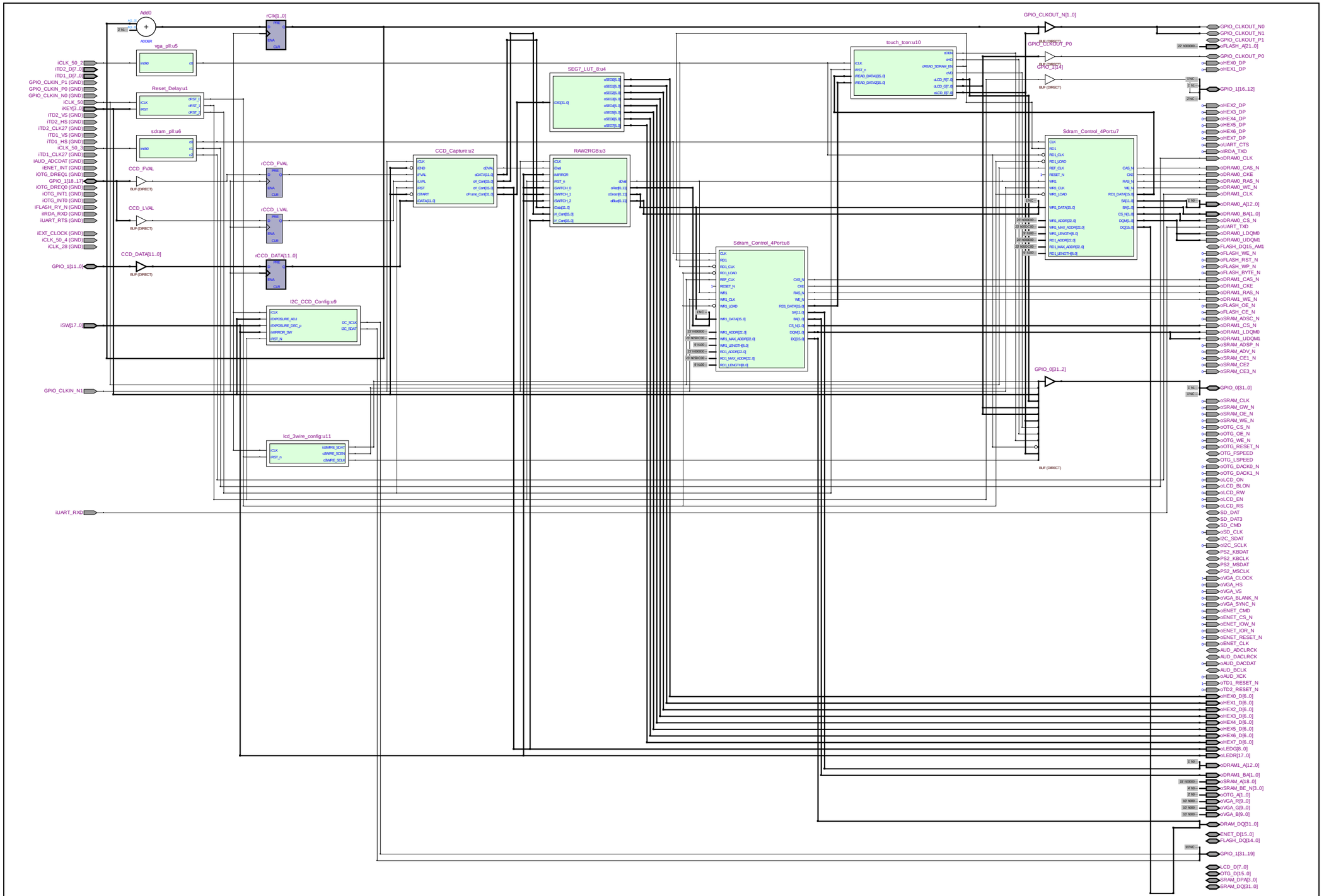
CAPÍTULO 2…… 2.400 €

CAPÍTULO 2…… 940 €

TOTAL CAPÍTULOS…….. **3.770 € ***

El total del presupuesto asciende a la cantidad de *Tres mil setecientos setenta Euros* (I.V.A. no incluido).

* Este valor no incluye el I.V.A



Plano nº 1	Circuito Lógico de la Aplicación DE2_70_D5M_LTM

ANEXO II
ROBOT MOWAY

ROBOT MOWAY

1.- INTRODUCCIÓN

El robot Moway es un pequeño robot programable, diseñado principalmente para realizar aplicaciones prácticas de robótica. En principio se trata de un robot autónomo, es decir funciona según la información del entorno que él mismo es capaz de detectar, sin embargo, como se verá en los posteriores apartados, también es posible controlarlo de forma remota, añadiendo los módulos de expansión que dotan al robot de esta funcionalidad.

El robot está dotado de una serie de sensores que le permiten desenvolverse en un entorno real. Asimismo, cuenta con un grupo de motores que le permite realizar desplazamientos sobre terreno liso comandado a través del bus de comunicaciones I2C. Cada uno de estos dos grupos de elementos está conectado a un microcontrolador que es el encargado de gestionar todas las operaciones del robot.

Además, el robot cuenta con opciones de ampliación a través de un bus de expansión por I2C/SPI. Entre las posibilidades de ampliación, destaca la posibilidad de conectar un módulo de comunicaciones por radiofrecuencia, del cual se habla posteriormente, o bien un kit de expansión en el que el usuario puede incorporar sus propios desarrollos electrónicos.

El diseño exterior del robot Moway es muy compacto, del tamaño de un teléfono móvil ó un ratón de ordenador. La Figura 1 muestra una vista del robot junto a su correspondiente base de control.

Figura 1. Vista general del robot Moway

2.- CARACTERÍSTICAS

A continuación, a lo largo de este apartado, se explican cada una de las partes que componen el robot Moway.

En términos globales, el interior del robot se compone de los siguientes elementos:
1.- Procesador
2.- Sistema Motriz
3.- Grupo de sensores e indicadores
4.- Sistema de alimentación
5.- Conector de expansión

2.1.- PROCESADOR

El elemento principal del robot es el microcontrolador PIC16F876A de 8 bits del fabricante Microchip Technology, el cual trabaja a una frecuencia de 4 Mhz. Este elemento es el encargado de controlar todos los periféricos distribuidos por el robot mediante sus puertos de entrada/salida. Algunos de estos puertos se controlan a través de señales digitales, otros a través de señales analógicas y otros, en cambio, se controlan a través de uno de los buses de comunicación I2C/SPI.

La grabación y programación del microcontrolador se realiza mediante la Base Moway, explicada posteriormente.

Las características técnicas del microcontrolador se explican detalladamente en el punto 2.4.

2.2.- SISTEMA MOTRIZ

El robot dispone de un grupo servo-motor doble para poder desplazarse. Dicho grupo consta de una parte electrónica y otra mecánica. Mediante la parte electrónica principalmente se controla la velocidad de los motores, mientras que la parte mecánica ofrece al robot la potencia suficiente para moverse sin problemas en diferentes entornos.
La Figura 2 muestra el sistema motriz.

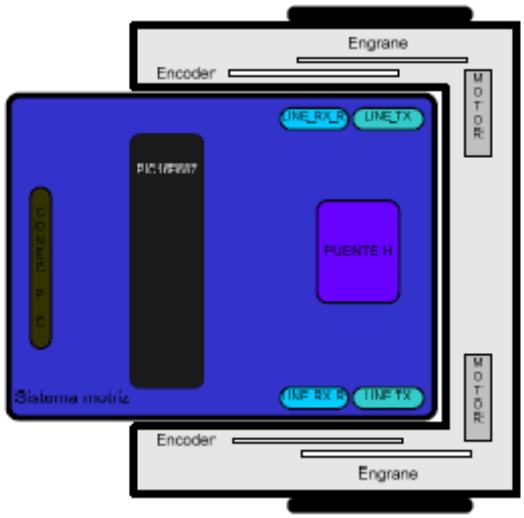


Figura 2. Sistema motriz: electrónica y mecánica

El grupo servo-motor presenta las siguientes funcionalidades:

1.- Control de velocidad: cada motor se controla por separado

2.- Control de tiempo: control de la duración de cada comando aplicado, con una precisión de 100 ms.

3.- Control de distancia recorrida: control de la distancia recorrida con cada comando, con una precisión de 1,7 mm.

4.- Cuentakilómetros general: cuenta la distancia recorrida desde el comienzo de los comandos.

5.- Control de ángulo: control del ángulo de rotación. Es posible realizar la rotación sobre su centro o sobre cada una de las dos ruedas.

El funcionamiento se basa en que el microcontrolador mande los comandos correspondientes mediante el protocolo I2C al sistema motriz, el cual está gobernado por un segundo procesador (PIC16F687). Este último controlador es el encargado de gestionar los motores, dejando libre de carga al microcontrolador principal, pudiendo éste dedicarse a otras tareas.

El control de velocidad se realiza mediante control proporcional con retroalimentación negativa de la señal de los encoders. En la Figura 3 se puede observar el sistema de control. El microcontrolador alimenta los motores mediante un puente H controlado por señales PWM (modulación de ancho de pulso). La rotación de la rueda es monitorizada por medio de una pegatina encoder y un sensor infrarrojo. Cuando la pegatina se encuentra en la parte negra, se tiene un 1 lógico en la salida, mientras que si se encuentra en la parte blanca su salida es 0. El microcontrolador (PIC16F687) analiza esta señal, realiza la medida del ancho del pulso y, de esta manera, conoce la velocidad exacta de la rueda, actuando en consecuencia sobre los motores. De esta manera, se mantendrá constante la velocidad del robot en cualquier superficie.

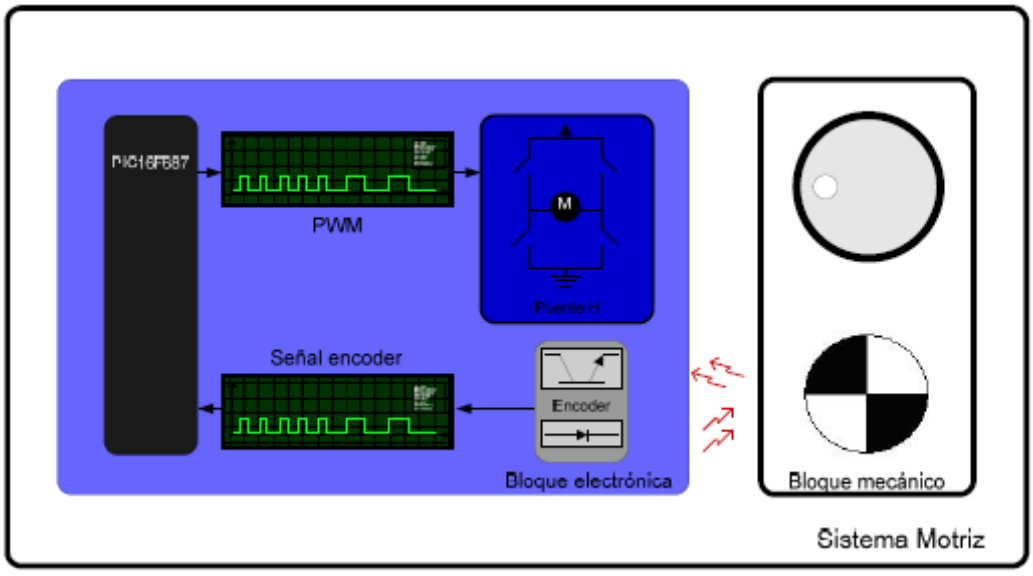


Figura 3. Control de motores

Por tanto, para realizar un desplazamiento del robot sólo es necesario recibir, desde el microcontrolador principal, un comando de movimiento con sus respectivos parámetros.

2.3.- GRUPO DE SENSORES E INDICADORES

El robot Moway cuenta con el siguiente listado de sensores e indicadores luminosos conectados al microcontrolador:

- Dos sensores de línea
- Dos sensores de infrarrojos detectores de obstáculos
- Un sensor de luz
- Un conector de expansión
- Cuatro diodos LED
- Un pad libre

La Figura 4 muestra el grupo de sensores e indicadores:

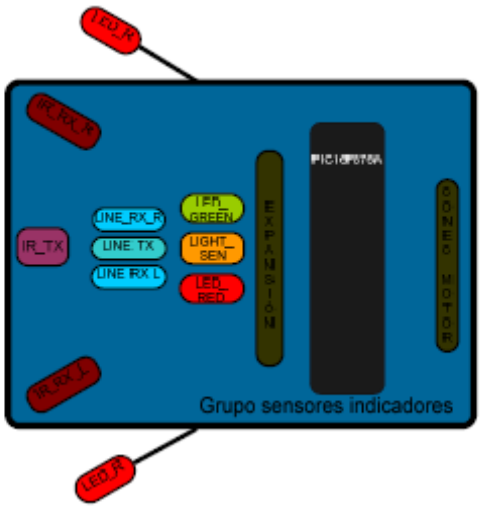


Figura 4. Grupo de sensores e indicadores

2.3.1.- Sensores de línea

Los sensores de línea son dos optoacloplladores de reflexión montados en la parte inferior delantera de la estructura del robot. Utilizan la reflexión de la luz infrarroja para detectar el tono del suelo en el punto en el que se encuentra el robot. Estos dos sensores están conectados a dos puertos analógicos del microprocesador por lo que es posible discernir entre diferentes tonos.

2.3.2.- Sensores detectores de obstáculos

Al igual que los sensores de línea, los sensores detectores de obstáculos emplean la luz infrarroja para detectar objetos situados en la parte delantera del robot.

El sensor está compuesto por una fuente de luz infrarroja, con una longitud de onda de 940 nm (KPA3010-F3C de Kingbright), y dos receptores (PT100F0MP de Sharp) colocados en ambos laterales de la estructura del robot.

La salida de los receptores está conectada a las entradas analógicas del microcontrolador, lo que permite detectar la presencia del algún objeto además de medir la distancia al mismo.

El funcionamiento del sensor consiste en que el emisor de luz emita un pulso de 70µs de duración que, si existe un obstáculo, el receptor capta utilizando una etapa de filtrado y

amplificación. Una vez procesada la señal electrónicamente, el microcontrolador puede medirla bien como entrada digital o bien mediante el convertidor analógico-digital. La distancia de alcance para la detección de obstáculos, en digital, es de unos 3 cm y se recomienda un entorno claro para aumentar la posibilidad de reflexión infrarroja.

La Figura 5 muestra el funcionamiento del sensor detector de obstáculos.

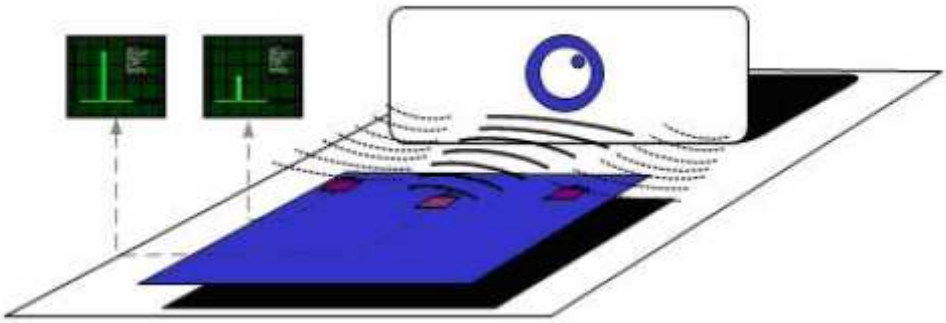


Figura 5. Sensor detector de obstáculos

2.3.3.- Sensor de luz

Este sensor permite al robot conocer la intensidad de luz que entra por una pequeña apertura orientada hacia delante situada en la parte superior del chasis. La salida del sensor (APDS-9002 de Avago Technologies) está conectada a un puerto analógico del microcontrolador, de manera que a través del convertidor analógico-digital, es capaz de conocer el nivel de intensidad de luz y ver si éste ha aumentado o disminuido respecto a la última lectura.

2.3.4.- Conector de expansión

Se trata de un conector situado en la parte superior de la estructura del robot, el cual permite la conexión de módulos comerciales o circuitos electrónicos que el usuario desee. Con el kit de Moway adquirido se dispone del módulo de radiofrecuencia BZI-RF2GH4, totalmente compatible con el robot Moway, el cual será explicado con mayor detalle en su correspondiente apartado.

2.3.5.- LEDs frontales

Existen dos diodos LED en la parte frontal del robot situados detrás del filtro infrarrojo. Únicamente tener en cuenta que estos LEDs deben estar apagados cuando se usan los detectores de obstáculos.

2.3.6.- LED superior bicolor

Se trata de un indicador doble (LED rojo + LED verde) situado en la misma apertura que el sensor de luz. Destacar que el caso de querer utilizar el sensor de luz, este indicador debe permanecer apagado.

2.3.7.- Pad libre

El PCB del robot Moway consta de un Pad, accesible abriendo el robot, para poder conectar circuitos electrónicos adicionales.

2.3.8.- Sistema de alimentación.

El robot cuenta con una batería situada en el interior de la estructura del mismo, la cual se trata de una pequeña célula de LiPo recargable. La recarga de la batería se realiza por el puerto USB de cualquier ordenador mediante la Base Moway.

La duración de la batería depende en gran medida de los sensores activos y del tiempo de utilización de los motores, de todas formas, el tiempo de carga aproximado es de 2h.

2.4.- MICROCONTROLADOR PIC16F876A

En este apartado se describen las características técnicas relevantes del microcontrolador PIC16F876A de Microchip.

2.4.1.- Principales características

La Tabla 1 muestra las principales características del dispositivo:

Características	PIC16F876A
Frecuencia de operación	DC – 20 MHz
Memoria de programa Flash (14-bit-words)	8K
Memoria de datos (bytes)	368
Memoria de datos EEPROM (bytes)	256
Interrupciones	14
Puertos I/O	Ports A,B,C
Temporizadores	3
Módulos Captura/Comparación/PWM	2
Comunicaciones serie	MSSP, USART
Comunicaciones paralelo	-
Módulo Analógico – Digital 10-bit	5 input channels
Comparadores analógicos	2
Set de instrucciones	35 instructions

Tabla 1. Características dispositivo PIC16F876A

2.4.2.- Diagrama de bloques

La Figura 6 muestra el diagrama de bloques del dispositivo PIC16F876A:

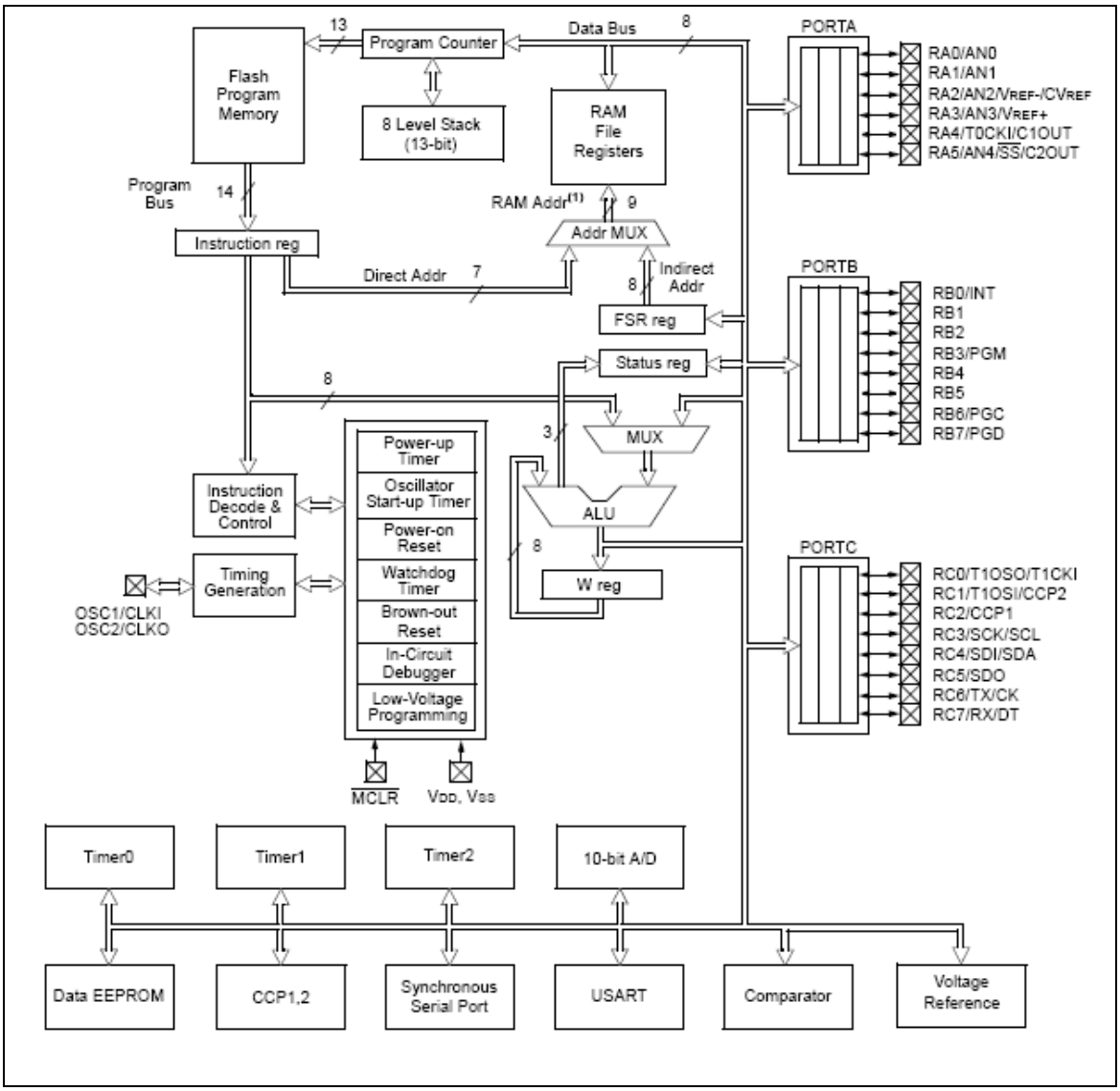


Figura 6. Diagrama de bloques PIC16F876A

2.4.3.- Organización de la memoria del dispositivo

Existen tres bloques de memoria implementados sobre el dispositivo:

- Bloque de memoria de programa.
- Bloque de memoria de datos.
- Bloques Data EEPROM/Flash Program Memory.

- Organización de la memoria de programa:

El dispositivo PIC16F876A cuenta con una capacidad de memoria de programa Flash de 8K words x 14 bits. La Figura 7 muestra la correspondiente distribución del mapa de memoria.

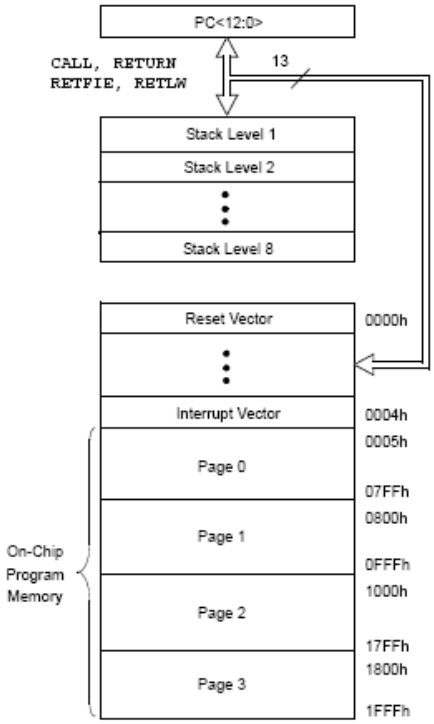


Figura 7. Memoria de Programa

- Organización de la memoria de datos:

La memoria de datos está dividida en distintos bancos, los cuales contienen registros de propósito general y registros para funciones especiales.

Cada uno de los bancos se extiende hasta un tamaño de 128 bytes, donde las direcciones más bajas están reservadas para los registros de funciones especiales. Por encima, se encuentran los registros de propósito general, implementados como memoria RAM estática.

Los registros para funciones especiales son usados por la CPU y por módulos periféricos para controlar el modo de funcionamiento del dispositivo. Estos registros también están implementados sobre memoria RAM estática.

La Figura 8 muestra el mapa de registros que compone la memoria de datos.

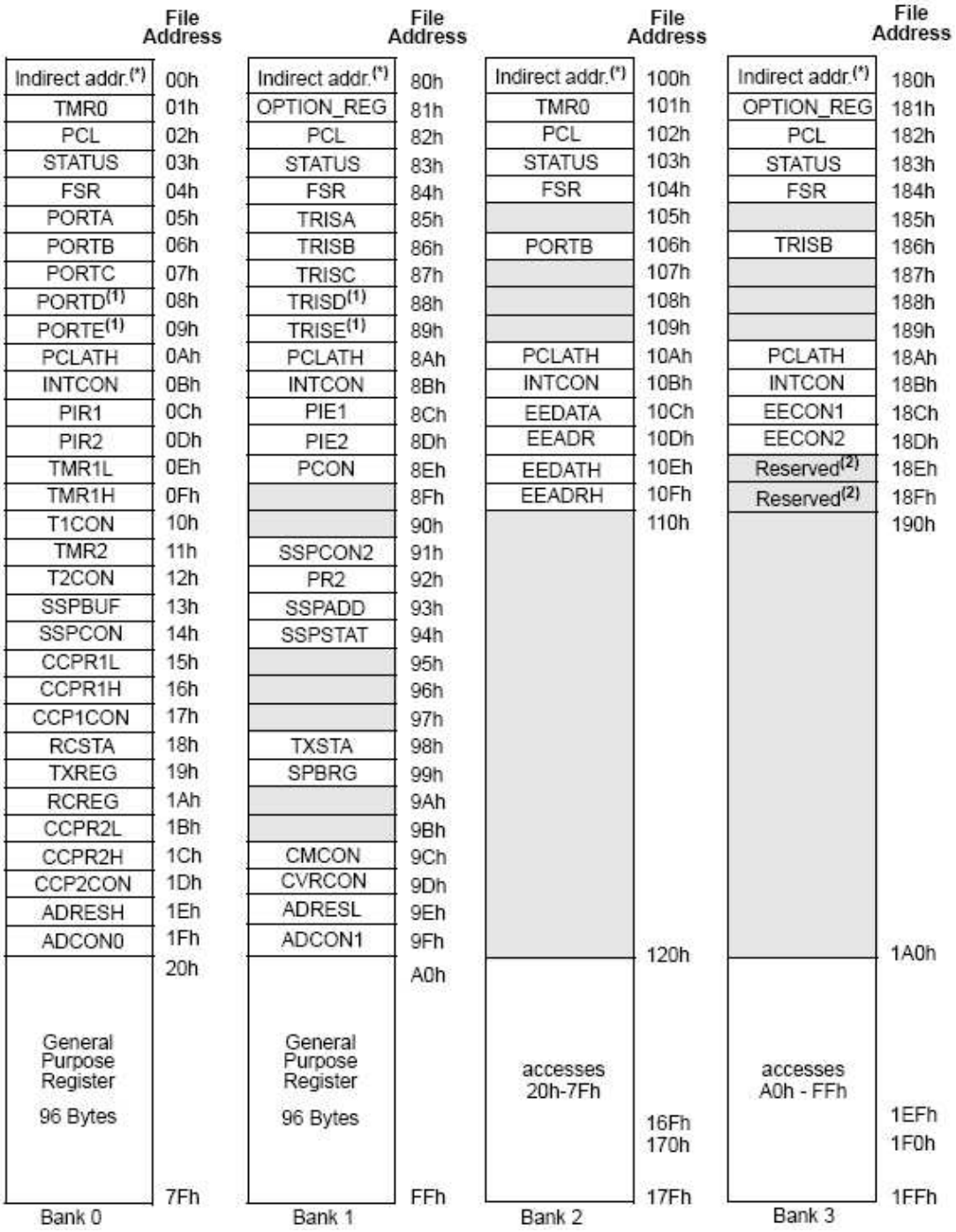


Figura 8. Mapa de Memoria de Datos

- Memoria de datos EEPROM/Memoria de programa Flash:

Esta memoria no tiene una distribución específica en el mapa de registros, sino que es indirectamente empleada por los registros de funciones especiales.

2.4.4.- Puertos de Entrada/Salida

2.4.4.1.- PORTA y registro TRISA.

PORTA es un puerto bidireccional de 6 bits. Su correspondiente registro de datos es el denominado TRISA. Configurando los bits del registro con valor '1', los pines del puerto actúan como entradas, mientras que si son configurados con valor '0', los pines se comportan como salidas.

2.4.4.2.- PORTB y registro TRISB.

PORTB es un puerto bidireccional de 8 bits. El registro TRISB es su correspondiente registro de datos. Configurando los bits del registro con valor '1', los pines del puerto actúan como entradas, mientras que si están configurados con valor '0' actúan como salidas. Este puerto es usado para recibir las estradas de interrupción del dispositivo.

2.4.4.3.- PORTC y registro TRISC.

PORTC es un puerto bidireccional de 8 bits. Su correspondiente registro de datos es el TRISC. Configurando los bits del registro como '1', los pines del puerto actúan como entradas, mientras que si se configuran con valor '0' actúan como salidas. Este puerto es usado para realizar la comunicación con módulos periféricos del dispositivo.

2.4.5.- Módulo MSSP (Master Synchronous Serial Port)

El modulo MSSP es un interfaz serie, útil para realizar comunicaciones con otros periféricos ó microcontroladores externos. El módulo puede operar de dos maneras:

- Serial Peripheral Interface (SPI): permite que 8 bits de datos sean transmitidos y recibidos simultáneamente, de forma sincronizada.
- Inter-Integrated Circuit (I^2C): Este modo de comunicación es el empleado en el funcionamiento del robot Moway para realizar la transferencia de comandos entre el microcontrolador general (PIC16F876A) y el módulo encargado de gestionar el grupo servo-motor.

  I^2C es un bus de comunicaciones en serie. La transmisión bidireccional serie (8 bits) de datos puede realizarse a 100 Kbits/s en el modo estándar ó 400 Kbits/s en el modo rápido, permitiendo en algunas ocasiones llegar a velocidades de hasta 3.4 Mbit/s.

  La principal característica de I^2C es que utiliza dos líneas para transmitir la información: línea de datos (SDA), y señal de reloj (SCL).

2.4.6.- Características especiales de la CPU

Todos los dispositivos de la familia PIC16F87XA tienen una serie de características dirigidas a maximizar el rendimiento del dispositivo, minimizar costes, ofrecer protección para los datos, etc. De todas ellas, de especial importancia es la selección del oscilador, el cual determina la frecuencia de trabajo del dispositivo. Los tipos de oscilador disponibles son los siguientes:

- LP Low-Power Crystal
- XT Crystal/Resonator
- HS High-Speed Crystal/Resonator
- RC Resistor/Capacitor

La siguiente tabla muestra los valores de frecuencia capaces de obtener con cada uno de ellos.

Modo	Frecuencia
LP	32 KHz 200 KHz
XT	200 KHz 1 MHz 4MHz
HS	4 MHz 8 MHz 20 MHz
RC	Frecuencia variable según: - Voltaje de entrada - Valor de la resistencia - Valor del condensador

Tabla 2. Frecuencia oscilador

3.- BASE MOWAY

Se trata de una tarjeta electrónica que se encarga de gestionar tanto la descarga del programa para el robot como la carga de la batería y la comunicación por radiofrecuencia. Consta del PIC18F2550 como elemento principal encargado de controlar todas las operaciones mencionadas anteriormente.

La Base se conecta con el robot a través de una pequeña apertura que éste tiene debajo del filtro infrarrojo delantero, mientras que en el otro extremo la conexión con el ordenador se realiza mediante USB.

4.- MÓDULO DE RF BZI-RF2GH4

4.1.- DESCRIPCIÓN GENERAL

El módulo BZI-RF2GH4 de comunicación por radiofrecuencia, está basado en el transceptor nRF24L01 fabricado por la compañía "Nordic Semiconductors". Dicho módulo incorpora la lógica necesaria para establecer una comunicación inalámbrica bidireccional con acuse de recibo. La comunicación con el microcontrolador se realiza mediante un bus SPI (Serial Peripheral Interface).

Las principales características del módulo BZI-RF2GH4 son las siguientes:

- Frecuencia de trabajo de 2.4GHz.
- Potencia de emisión entre -18 y 0 dBm.
- Velocidad de transmisión entre 1 y 2 Mbps.
- 128 canales de transmisión seleccionables por el bus SPI.

Además del circuito integrado nRF24L01, el módulo incorpora la electrónica anexa necesaria para su correcto funcionamiento y una antena microstrip en la misma placa con la red de adaptación de impedancias.

Básicamente, este tipo de antena se diseña a partir de líneas de transmisión o resonadores sobre substrato dieléctrico. Las dimensiones de la antena se eligen de forma que la estructura disipe la potencia en forma de radiación. Como ventajas más importantes de este tipo de antena se tiene su fácil adaptación a la forma de la estructura (plana o curvada), robustez, combinables con circuitos integrados de microondas, y capacidad de trabajo a distintas frecuencias y con distintas polarizaciones.

Como interfaz para la comunicación con el microcontrolador, el módulo dispone de cuatro pines accesibles para el bus SPI, dos pines para el control del módulo y otros dos pines para alimentación.

La Figura 9 muestra una imagen del módulo BZI-RF2GH4:

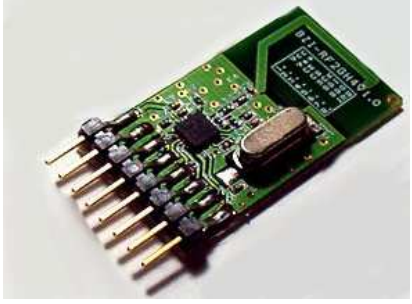


Figura 9. Módulo BZI-RF2GH4 1

4.2.- ESPECIFICACIONES TÉCNICAS

Las tablas expuestas a continuación resumen las características técnicas del módulo.

Parámetro	Valor	Unidad
Tensión mínima de alimentación	1.9	V
Tensión máxima de alimentación	3.6	V
Potencia máxima de salida	0	dBm
Velocidad máxima de transmisión	2000	Kbps
Corriente en modo de transmisión @ 0 dBm potencia de salida	11.3	mA
Corriente en modo recepción @ 2000 kbps	12.3	mA
Corriente en modo Power Down	900	nA
Frecuencia máxima del bus SPI	8	MHz
Rango de temperatura	-40 a +85	ºC

Tabla 3. Parámetros principales del módulo BZI-RF2GH4

Pines	Nº	Descripción
Vcc	1	Tensión de alimentación del módulo
Vss	2	GND
CE	3	Chip Enable
CSN	4	Chip Select del SPI (Negado)
SCK	5	Reloj del bus SPI
SDI	6	Entrada de datos del bus SPI
SDO	7	Salida de datos del bus SPI
IRQ	8	Salida de interrupción (negado)

Tabla 4. Distribución de pines del módulo BZI-RF2GH4

4.2.1.- TRANSCEPTOR nRF24L01.

4.2.1.1.- Introducción

El módulo nRF24L01 es un transceptor a 2,4 GHz diseñado para aplicaciones inalámbricas de muy baja potencia.

Dicho módulo permite ser configurado a través del interfaz SPI (Serial Peripheral Interface). Mediante este interfaz se accede al mapa de registros del módulo permitiendo la configuración del modo de operación deseado. El usuario puede configurar parámetros tales como: modo de operación (Tx/Rx), frecuencia del canal de RF, potencia de salida ó velocidad de transmisión de los datos.

4.2.1.2.- Diagrama de Bloques:

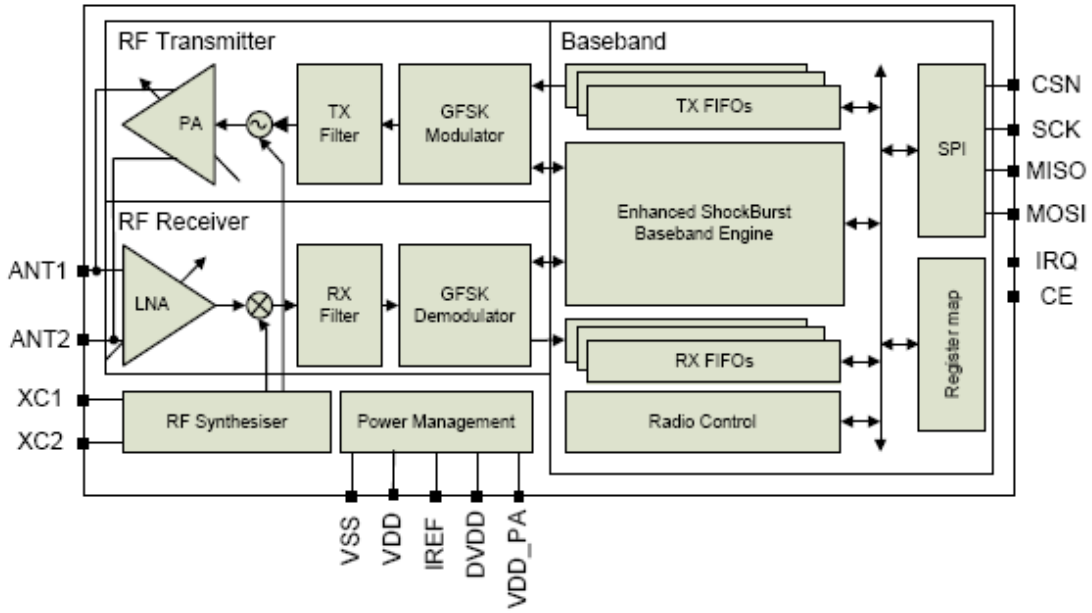


Figura 10. Diagrama de bloques chip nRF24L01

La siguiente tabla especifica la función de cada uno de los pines del chip:

Pin	Función	Descripción
CE	Entrada digital	Habilitación del chip. Selección del modo Rx ó Tx
CSN	Entrada digital	Chip select del SPI
SCK	Entrada digital	Reloj del SPI
MOSI	Entrada digital	Entrada de datos del bus SPI
MISO	Salida digital	Salida de datos del bus SPI
IRQ	Salida digital	Salida de interrupciones. Activa a nivel bajo
VDD	Alimentación	Suministro de potencia (+1,9V / +3,6V DC)
VSS	Alimentación	Tierra (0V)
XC2	Salida analógica	Pin 2 sintetizador RF
XC1	Entrada analógica	Pin 1 sintetizador RF
VDD_PA	Salida de potencia	Salida de suministro de potencia (1,8V) para el amplificador interno del chip
ANT1	RF	Interfaz de antena 1
ANT2	RF	Interfaz de antena 2
IREF	Entrada analógica	Corriente de referencia
DVDD	Salida de potencia	Suministro interno para propósitos de demodulación

Tabla 5. Función pines nRF24L01 1

4.2.1.3.- Control RF. Configuración.

En este apartado se describen los diferentes modos en los que el transceptor nRF24L01 es capaz de operar. Asimismo se indican los parámetros a configurar para ello.

El chip puede ser configurado de tres modos diferentes: standby, Rx (recepción) y Tx (transmisión).

a) Modo Standby:

Dentro de este modo se distinguen dos estados: standby-1 y standby-2. El modo standby-1 es usado para minimizar el consumo medio de corriente. El modo queda activo procedente de los modos Rx ó Tx cuando la entrada CE contiene un nivel bajo.

En el estado standby-2 se consume más corriente que en el estado anterior, y éste entra en funcionamiento cuando la pila (FIFO) de transmisión está vacía.

b) Modo Rx (Recepción):

Es el modo activo cuando el transceptor actúa como receptor. En este modo, el receptor demodula las señales procedentes del canal de RF, pasando a trabajar en banda base. El protocolo en banda base continuamente busca paquetes válidos de información, dejando los que encuentra en la pila de recepción (FIFO). Tener en cuenta que si esta pila está llena, el paquete se descarta.

c) Modo Tx (Transmisión):

Este modo queda activo a la hora de querer transmitir uno o varios paquetes de información. El transceptor permanece en modo Tx hasta que finaliza la transmisión del paquete actual. La siguiente acción depende del estado de la pila de transmisión. Si ésta no se encuentra vacía se transmitirá el siguiente paquete, si por el contrario la pila se encuentra vacía se pasa al modo standby-2. Es importante no mantener el transductor en modo Tx en más de 4 ms seguidos (aunque el modo de retransmisión esté activado).

La siguiente tabla describe cómo configurar cada uno de los modos de operación expuestos anteriormente:

Modo	Registro PWR_UP	Registro PRIM_RX	CE	Estado de la pila de memoria (FIFO)
Modo Rx	1	1	1	-
Modo Tx[1]	1	0	1	Entrada de dato en pila de Tx
Modo Tx[2]	1	0	pulso a nivel alto, mínimo de 10µs.	Entrada de dato en pila de Tx
Standby-2	1	0	1	Pila de Tx vacía
Standby-1	1	-	0	No hay nuevo dato a transmitir

Tabla 6. Configuración de los modos de operación.

[1].- En este modo de operación, si la entrada CE contiene un nivel alto la pila de transmisión es vaciada, por lo que todas las posibles retransmisiones pueden ser llevadas a cabo. Si la pila estaba vacía cuando llega un nivel alto en la entrada CE, el transceptor queda en modo standby-2. En este caso, se volverá a transmitir un

paquete en el momento en el que la entrada CSN contenga un nivel alto y exista un paquete en la pila de transmisión.

[2].- En este modo de operación es necesario un pulso de al menos 10 µs en la entrada CE. En cada pulso se transmite un paquete de información. Este es el modo normal de operación. Después de que el paquete es transmitido, el transductor pasa al modo standby-1.

4.2.1.4.- Ratio de transmisión/recepción de datos.

El ratio puede ser de 1Mbps ó 2Mbps. A 1Mbps se tiene una sensibilidad en el receptor 3dB mayor que el caso de 2Mbps. Por su parte, un ratio mayor significa un menor consumo medio de corriente y la reducción de posibles colisiones en la comunicación.

Es importante tener en cuenta que tanto el emisor como el receptor deben ser configurados de igual manera para realizar una correcta comunicación.

4.2.1.5.- Frecuencia del canal de RF.

El transceptor nRF24L01 opera en el rango de frecuencias de 2,400GHz a 2,525GHz. Un canal ocupa un ancho de banda de 1MHz a 1Mbps y de 2MHz a 2Mbps.

La frecuencia del canal se selecciona mediante el registro RF_CH siguiendo la siguiente fórmula:

$$F_0 = 2400 + RF\_CH\ [MHz]$$

Tanto el emisor como el receptor deben ser programados con el mismo canal de RF para poder lograr la comunicación entre ambos.

4.2.1.6.- Control de potencia de salida

Este control tiene cuatro posibilidades, tal como se resume en la siguiente tabla:

SPI RF_SETUP	Potencia de salida RF	Consumo de corriente
11	0 dBm	11,3 mA
10	-6 dBm	9,0 mA
01	-12 dBm	7,5 mA
00	-18 dBm	7,0 mA

Tabla 7. Control de potencia de salida.

4.2.1.7.- Ganancia LNA

El transceptor cuenta con un amplificador de baja figura de ruido (LNA) en la etapa de recepción, controlado por un control de ganancia. Es posible reducir la ganancia por defecto, minimizando el consumo de energía (0,8 mA), con la penalización de una disminución de 1,5 dB de sensibilidad en el receptor.

4.2.1.8.- Pila de memoria FIFO

La pila de memoria FIFO es empleada para almacenar la información que va a ser transmitida o bien para almacenar temporalmente los datos recibidos antes de analizar si los mismos son válidos.

La pila se estructura de la siguiente manera:

- Tres niveles de 32 bytes para Tx.
- Tres niveles de 32 bytes para Rx.

La siguiente figura muestra el correspondiente diagrama de bloques:

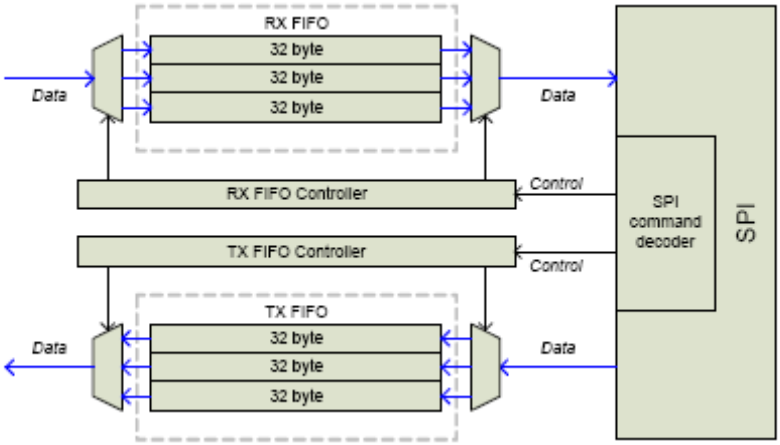


Figura11. Diagrama de bloques FIFO Tx/Rx

5.- PROGRAMACIÓN DE MOWAY EMPLEANDO MowayGUI

MowayGUI es el software para la programación del robot Moway de forma gráfica para la realización de sencillas aplicaciones. Dicho software está incluido en el CD-ROM suministrado con el kit de Moway.

5.1.- INTERFAZ DE USUARIO

La figura 12 muestra la pantalla principal del programa, en la que se pueden distinguir las siguientes partes:

1. Rutina principal: Es el espacio donde el usuario crea el programa principal. Para ello ofrece un área de trabajo en la que se incluyen los bloques o herramientas que el usuario decida para el diseño del proyecto. Esta rutina principal es reutilizable y puede ser usada como subrutina en otros proyectos.
2. Inicio de programa: dentro de la rutina principal existe el elemento de inicio, del cual colgará el resto del programa.
3. Herramientas: Mediante el empleo de las distintas herramientas el usuario deberá diseñar la aplicación deseada. La inclusión de módulos y elementos condicionales unidos con flechas forma el flujo de programa.
4. Variables: Usadas para configurar, guardar y realizar operaciones matemáticas.

Una vez realizado el diseño, el usuario debe conectar Moway a la base y pulsar el botón de programación para cargar el programa en el robot.

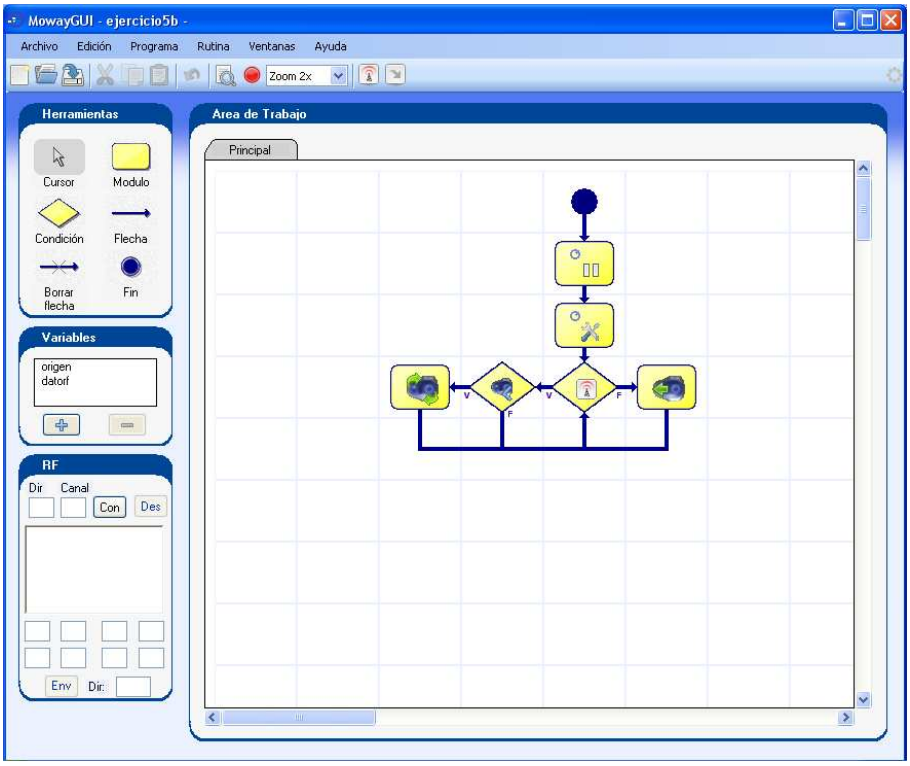


Figura 12. Entorno gráfico MowayGUI

5.2.- MOWAY RC Center

Es el programa que controla el robot Moway desde el PC por radio control. Además mediante este programa se monitoriza el estado de los diferentes sensores, con lo que puede ser una importante herramienta para explorar el terreno.

La siguiente figura muestra la ventana de la aplicación:

Figura 13. Entorno Moway RC Center

6.- PROGRAMACIÓN DE MOWAY EMPLEANDO MPLAB

MPLAB IDE de Microchip es el entorno de programación empleado a la hora de trabajar con microcontroladores PIC (Microchip también es el fabricante de dichos microcontroladores). En principio el lenguaje que utiliza es ensamblador, si bien se le pueden añadir otros lenguajes, junto con su correspondiente compilador. Este software se puede descargar gratuitamente de la página web de Microchip (www.microchip.com).

Dicho compilador es el encargado de generar ficheros en formato hexadecimal (.hex), los cuales contienen la información de configuración del dispositivo.

Para el caso de la programación en lenguaje C, los compiladores compatibles con el PIC de Moway son los de la familia CCS, los cuales están orientados, facilitando directivas y funciones propias, al trabajo con microcontroladores PIC. Dicho compilador queda integrado dentro del entorno de desarrollo de MPLAB, lo cual permite desarrollar todas y cada una de las fases que componen un proyecto.

Tanto para la programación en lenguaje ensamblador como para la programación en lenguaje C existen librerías disponibles en la página Web de Moway (www.moway.es), que permiten el manejo de los sensores, motores y módulo RF.

7.- PRESUPUESTO

En el presente apartado se muestra el coste del Kit educativo basado en el robot Moway.

CONCEPTO	CANTIDAD	PRECIO UNITARIO	TOTAL
Kit Moway Deluxe	1	265 €	**265 €***

El kit incluye los siguientes elementos:

- 2 Robots Moway
- Base Moway USB
- Cable USB tipo AB
- CD de instrucciones y software de programación
- 3 módulos de radiiofrecuencia
- Maletín Moway

* Este valor no incluye el I.V.A

	Tarjetas educacionales de Altera						***Tarjetas educacionales de Xilinx***				
	UP2	**DE0**	**DE1**	**DE2**	**DE2-70**	**DE3**	**Basys**	**Nexys 2**	**Spartan 3E Starter**	**XUPV5-LX110T**	**XUPV2P**
Software	MAX+PLUS II - QUARTUS II	QUARTUS II	QUARTUS II	QUARTUS II	QUARTUS II	QUARTUS II	ISE WebPack	ISE WebPack	ISE WebPack	ISE WebPack	ISE Foundation software
Dispositivo FPGA	FLEX 10K EPF10K70	Cyclone III EP3C16F484C 6 with EPCS4	Cyclone II EP2C20F484C7 with EPCS4	Cyclone II EP2C35F672C6 with EPCS16	Cyclone II EP2C70F896C6 with EPCS64	Stratix III EP3S150F1152C2, EP3S260F1152C2 or EP3S340F1152C2 with EPCS128	Xilinx Spartan 3E, 250K gates	Xilinx Spartan 3E, 500K or 1200K gates	Xilinx Spartan 3E XC3S500E	Xilinx Virtex-5 XC5VLX110T	Virtex-II Pro XC2VP30
RISC	-	-	NIOS II	NIOS II	NIOS II	-	-	32-bit MicroBlaze	32-Bit MicroBlaze - PicoBlaze	-	2 x PowerPC processor
Cable descarga	ByteBlaster II	USB Blaster	USB Blaster	USB Blaster	USB Blaster	USB Blaster	USB	USB	USB	USB	USB
Alimentación	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si
Memoria	18 Kbits Socket for optional EPROM	8 MB SDRAM, 4 MB Flash, SD Memory Card Slot	8 MB SDRAM, 4 MB FLASH, 512 KB SRAM, SD Card socket	8 MB SDRAM, 4 MB FLASH, 512 KN SRAM, SD Card socket	2 MB SSRAM, 64 MB SDRAM, 8 MB FLASH, SD card socket	DDR2 SO-DIMM socket, SD card socket	72 Kbits dual-port RAM, Platform FLASH ROM	16MB PSDRAM, 16MB FLASH, Platform Flash ROM	32 MB DDR SDRAM, 16MB FLASH, 2MB Serial Flash	2 x 32MB Flash PROM, 1GB Compact Flash card, 256 MB SODIMM module.	DDR SDRAM DIMM that can accept up to 2Gbytes of RAM
Puerto Comunicación	JTAG, VGA, PS/2	Video Out (VGA 4-bit DAC), PS/2	Line In/Out, Microphone In (24-bit Audio Codec), VGA 4-bit DAC, RS-232, PS/2	Line In/Out, Microphone In (24-bit Audio Codec), Video out (VGA 10-bit DAC), Video in (NTSC/PAL), 10/100 Ethernet, USB 2.0 (type A&B), Infrared Port, RS-232, PS/2	Line In/Out, Microphone In (24-bit Audio Codec), Video out (VGA 10-bit DAC), Video in (NTSC/PAL), 10/100 Ethernet, USB 2.0 (type A&B), Infrared Port, RS-232, PS/2	USB 2.0 (mini-AB+ 2 type A)	USB port, VGA, PS/2	USB2 Port, VGA, PS/2, serial ports	DB15HD, VGA, PS/2, 2 x DB9 RS-232, RJ-45 Ethernet	2 x USB, 2 x PS/2, RJ-45, RS-232, 2 x Audio-In, 2 x Audio_Out, Video Input, Video (DVI/VGA) Output	10/100 Ethernet, USB2, XSGA video port, PS/2, RS-232, Line-in, Microphone-in, Line-out, AMP Out, SATA connector
Puerto Expansión	3 x 42 pines	2 x 40 pin	2 x 40 pin	2 x 40 pin	2 x 40 pin	2 x 40 pin, 4 HSMC high-speed headers	4 x 6-pin Pmod connectors	4 x 12-pin Pmod connectors, Hirose FX2	100-pin Hirose FX2 connector. 3 x 6-pin Pmod connectors, 16 pin header	Single-Ended and Differential I/O Expansion	High and low speed expansion connectors
LEDs	16	10 green LEDs	8 green LEDs 10 red LEDs	9 green LEDs 18 red LEDs	9 green LEDs 18 red LEDs	8 RGB LEDs	8	8	-	-	4
Displays 7 segmentos	4	4	4	8	8	2	4	4	-	-	-
LCD Display	-	-	-	16x2 LCD Display	16x2 LCD Display	-	-	-	Opcional	16x2 LCD Display	-
Pulsadores	4	3	4	4	4	4	4	4	-	-	5
Interruptores	4 DIP switch	10	10	18	18	1 DIP switch, 4 slide switches	8 slide switches	8 slide switches	4 slide switches	-	4
Precio	99$	Academic Customers:79$ (quantities >30: 59$) Commercial Customers: 119$	Academic Customers:125$ (quantities >30: 99$) Commercial Customers: 199$	Academic Customers:269$ Commercial Customers: 495$	Academic Customers:329$ Commercial Customers: 599$	Academic Customers:contact Altera Commercial Customers: 2695$ (Price for DE3-150)	Academic Customers:89$ Commercial Customers: 109$	Academic Customers:99$ Commercial Customers: 129$	Academic Customers:149$ Commercial Customers: 169$	Academic Customers:750$ Commercial Customers: 1999$	Academic Customers:299$ Commercial Customers: contact Digilent

Tabla 1. Oferta y características de productos de las compañías Altera y Xilinx