Capítulo 5 Metodología

En este capítulo se muestra la descripción de todos los pasos que se siguieron para la realización total de este proyecto.

5.1 Descripción de circuitos lógicos

Los circuitos lógicos son los que contienen, en sí, el multiplicador rápido. Se decidió realizar las descripciones de los circuitos mediante la herramienta Quartus II, de la compañía Altera. Dicho *software* permite describir en VHDL circuitos lógicos para que después se puedan sintetizar en algún dispositivo específico. Asimismo, permite simular los circuitos generados y realizar algunas verificaciones de tiempo de respuesta y área utilizada (o bien, a utilizar) dentro del dispositivo entre otras opciones. Posteriormente, con el mismo *software* y alguna interfaz, es posible programar y/o configurar los circuitos descritos dentro del dispositivo que se desea emplear.

Después de haber realizado la investigación sobre arquitecturas y algoritmos de multiplicación rápida, se optó por realizar un par de ellos. El primero fue el multiplicador secuencial basado en sumas y desplazamientos, puesto que implicaba una relativa facilidad en la descripción en VHDL además de que es el algoritmo de sumas y desplazamientos sin un método que acelere la multiplicación por lo que era factible tomarlo como referencia a la hora de comparar tiempos con otras arquitecturas.

El otro algoritmo que se decidió implementar fue el de Booth, ya que dentro de lo recabado en la bibliografía, apuntaba a ser el algoritmo más rápido para el objetivo específico de este proyecto, debido a que la mayoría de los algoritmos o arquitecturas disminuyen el tiempo en el que se realizan las sumas mediante varios métodos que implican en su mayoría diferentes usos del carry generado por los elementos que realizan las adiciones; sin embargo el algoritmo de Booth,

debido a la recodificación antes vista, logra evitar la realización de varias sumas, que como sabemos es la operación dentro de la multiplicación que más tiempo consume.

Cabe mencionar que dentro de las descripciones de los circuitos lógicos no sólo se encuentran los multiplicadores, sino que se debieron hacer adecuaciones para que fuera posible realizar la comunicación posteriormente con el PIC. A lo que se refiere lo anterior es que debido a que el PIC no tiene suficientes puertos de entrada y salida como para albergar los 16 bits de los multiplicandos así como los 16 bits del resultado del multiplicador, aparte de la interfaz que se escoja para mostrar la multiplicación al usuario del sistema completo, se agregaron ciertos elementos a la descripción lógica tanto a la entrada como a la salida del sistema configurado en el FPGA.

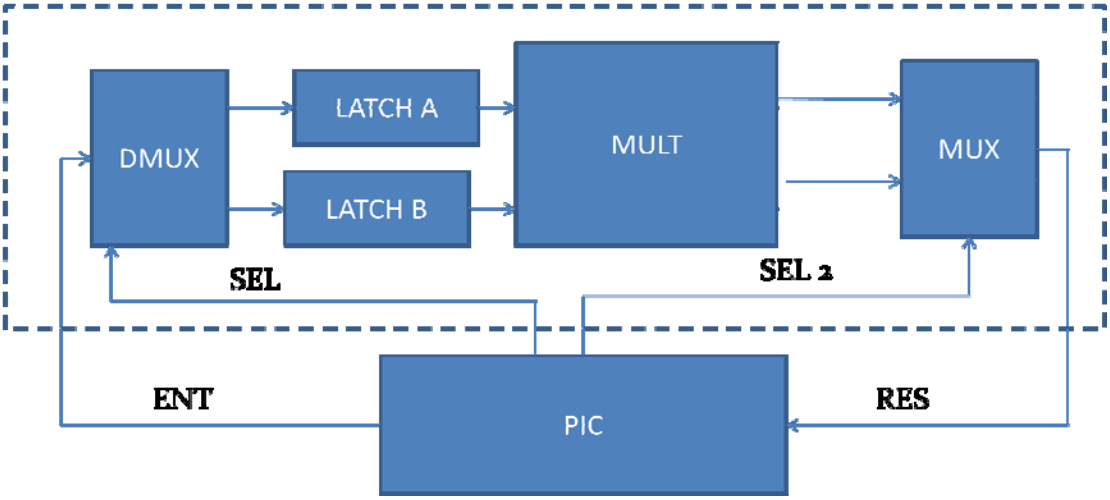


Figura 5.1 Diagrama de bloques del sistema completo.

La figura anterior muestra el diagrama de bloques del sistema FPGA-PIC. El sistema descrito en VHDL se encuentra encerrado por la línea punteada. Se observa que como elemento central tenemos el multiplicador y a su entrada y salida tenemos otros elementos lógicos. Para disminuir el número de pines

utilizados del PIC se decidió multiplexar tanto las entradas como las salidas al multiplicador, de esta forma, el PIC sólo manda 8 bits de entrada y recibe 8 bits del resultado. Con este propósito se crearon también dos señales de control, que seleccionan en la entrada si se trata del multiplicando A o B y dependiendo de esto lo registrado se guarda en el latch A o en el latch B de donde posteriormente el multiplicador obtendrá sus números a multiplicar. La segunda señal de control selecciona si se va a leer en el PIC la parte alta o la parte baja del resultado de la multiplicación.

Por lo tanto, dentro de la descripción lógica se agregaron un demultiplexor junto con dos latches de 8 bits para determinar si la entrada se refiere al multiplicando A o B, y un multiplexor a la salida que envía la parte alta o la parte baja del resultado al PIC, dependiendo de la señal de selección. Se decidió que las señales de control las generara el PIC para mantener la parte de lectura y de muestra de los números totalmente sincronizada.

Las descripciones en VHDL del multiplicador secuencial basado en sumas y desplazamientos, así como del multiplicador basado en el algoritmo de Booth se encuentran en la sección de anexos de este trabajo.

Con las descripciones realizadas en VHDL Quartus traduce esto a circuitería que realiza lo descrito.

5.2 Simulación de los circuitos lógicos

Una vez que se realizaron las descripciones de los circuitos lógicos, se pasó a realizar las simulaciones de los mismos. Quartus tiene una herramienta que realiza simulaciones de los circuitos realizados. Con esta herramienta es posible manipular las señales de entrada al sistema y observar el comportamiento de las señales de salida e inclusive de señales internas, con el propósito de comprobar que el comportamiento del circuito es el deseado.

Primero se simularon los multiplicadores aislados, para determinar si realizaban correctamente las multiplicaciones y en tiempos correctos.

Una vez que se determinó que sí funcionaban los multiplicadores se procedió a agregarles los elementos de entrada y salida que se explicaron en la sección anterior y se simuló el sistema completo. Se comprobó que los nuevos elementos interactuaban de forma correcta con el multiplicador, de acuerdo a los valores de las señales de selección que posteriormente se generarían por medio del PIC.

5.3 Verificación de los circuitos generados

Quartus crea varios archivos de verificación cada vez que se utiliza la herramienta de compilación. Dentro de la compilación se ejecutan varias herramientas: análisis y síntesis, fitter, assembler y analizador de tiempos. Para poder hacer uso de estas herramientas es necesario determinar el circuito integrado para el que está destinado el circuito que se describió, ya que varios de los análisis se realizan con base en el comportamiento específico de cada dispositivo. Por lo tanto, se debió escoger el dispositivo con el cual se iba a trabajar.

El laboratorio de electrónica del CCADET cuenta con una tarjeta UP1 de Altera, en la cual se encuentran dos dispositivos, uno de ellos es un CPLD de la familia MAX 7000 que originalmente se planeaba usar para los fines de esta tesis. Sin embargo los análisis de verificación demostraron que no tenía suficiente área para programar el multiplicador dentro de él, mucho menos podría ser posible programar el sistema completo. Por lo tanto se optó por usar el otro dispositivo dentro de la tarjeta UP1: un FPGA FLEX10k20RC240-4.

Dicho dispositivo es un FPGA de 240 pines, de los cuales 189 están disponibles para el usuario como entrada/salida, en un encapsulado tipo RQFP. Los dispositivos EPF10k20 contienen 1152 LEs y 144 LABs y son capaces de trabajar a 5 volts.

Dentro de la tarjeta UP1 se tienen unos dispositivos externos como *dip switches*, LEDs, *displays* de siete segmentos y otras interfaces, las cuales ya tienen asignadas sus pines de entrada/salida en el FPGA. Los pines asignables por el usuario tienen terminales en lo que Altera llama líneas de expansión, las cuales son filas dobles de hoyos disponibles para el acceso a señales globales, tierra, Vcc y pines de entrada/salida.

En la etapa de verificación de los circuitos lógicos, habiendo escogido ya el dispositivo a usar, Quartus pasa la descripción por varias etapas. La primera es análisis y síntesis, en la cual analiza la descripción hecha, de tal forma que si existe algún error en sintaxis de acuerdo a estándares de VHDL detiene la compilación y señala en qué lugar se encuentra el error o discrepancia y por qué se considera como tal. Este módulo del compilador también construye una base de datos, optimiza el diseño para el dispositivo elegido y *mapea* la tecnología para el diseño lógico. El reporte que ofrece Quartus sobre esta parte de la compilación incluye un resumen sobre los ajustes hechos para lograr la síntesis del circuito dentro del dispositivo, como la versión de VHDL con la que se trabaja, los archivos necesarios para sintetizar el circuito (dentro de los cuales se incluye el .vhd que se creó para la descripción, y las funciones necesarias para crear el circuito descrito). El reporte habla también sobre los *latches* creados por el usuario o inferidos por el compilador y los recursos del dispositivo a usar, entre otras cosas.

La segunda etapa de la compilación de Quartus es la herramienta llamada *fitter*, la cual construye el circuito lógico sobre el dispositivo seleccionado para el proyecto y coloca y conecta el diseño. El *fitter* entrega datos como el número de elementos lógicos usados, el número de pines de entrada/salida usados así como la ubicación de los mismos en el dispositivo, la lista de todos los pines del encapsulado y las señales que se encuentran en cada uno y el *fan out* de las señales de entrada, entre otras cosas.

Esta herramienta fue la que reportó que no era posible sintetizar el multiplicador dentro del CPLD de la tarjeta UP1 ya que no tenía el área lógica necesaria, por lo que hubo que decidir emplear el FPGA.

La tercera etapa que realiza Quartus se llama *Assembler*, la cual crea una imagen de programación del dispositivo para configura el dispositivo. Se generan los archivos para configurar el FPGA, ya sea mediante JTAG[1], en cuyo caso se le asigna un código de usuario para reconocer el dispositivo en la cadena y se genera el archivo .sof, o mediante un dispositivo de configuración, para lo cual se genera un archivo de tipo .pof que se emplea para programar el dispositivo que configurará el FPGA.

La cuarta y última etapa es el *timing analyzer*, que en el caso específico para FPGAs nos entrega cuatro tiempos específicos y una lista de señales involucradas en dichos tiempos. A continuación se describen los tiempos que entrega Quartus.

Tiempo t_{SU}

t_{SU} especifica la cantidad de tiempo que los datos necesitan para llegar y ser estables en un pin externo de entrada antes de de una transición de reloj en un pin de entrada/salida asociado a un reloj.

[1] Una interfaz JTAG (*Join Test Action Group*) es una interfaz de 5 pines diseñada para que múltiples circuitos integrados dentro de una tarjeta puedan ser conectados en cadena para ser probados, programados o configurados.

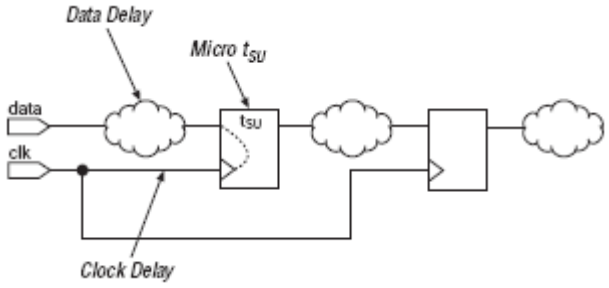


Figura 5.2 Tiempo de establecimiento de reloj.

El micro t_{SU} es el tiempo de establecimiento interno del registro. Es una característica del registro y no se afecta por las señales que alimentan al registro.

Tiempo t_H

t_H especifica la cantidad de tiempo que los datos necesitan mantenerse estables en un pin externo de entrada después de una transición de reloj en un pin de entrada/salida asociado al reloj.

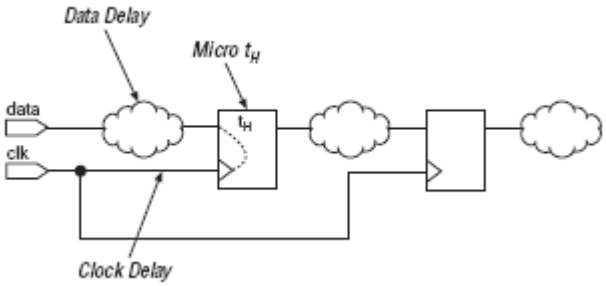


Figura 5.3 Tiempo de mantenimiento de reloj.

Tiempo t_{CO}

t_{CO} es el retraso denominado reloj a salida (*clock to output delay*), y es el máximo tiempo requerido para obtener una salida válida en un pin de salida alimentado por un registro, después de una transición de reloj en la entrada de reloj del registro. Micro t_{CO} es el retraso de reloj a salida interno del registro.

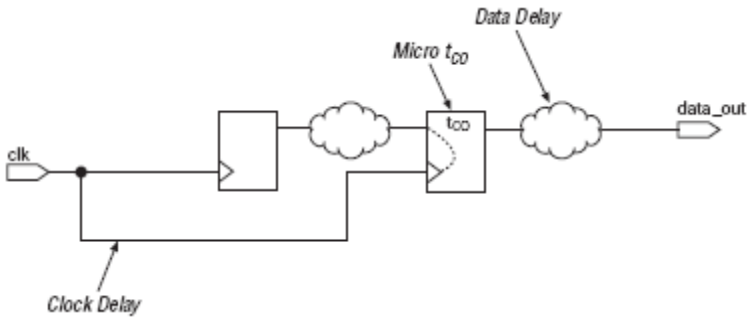


Figura 5.4 Retraso de reloj a salida.

Tiempo t_{PD}

El retraso de pin a pin (t_{PD}) es el tiempo requerido para que una señal de un pin de entrada se propague a través de lógica combinacional y aparezca en un pin de salida.

Puesto que no usamos reloj en los multiplicadores, el tiempo que nos concierne es el t_{PD}, ya que implica que los valores de entrada del multiplicador se propagan por toda la lógica generada y nos entrega el valor del producto en la salida. Es decir, debido a que el circuito descrito es combinacional, el único tiempo que nos concierne es el t_{PD}.

5.4 Implementación física

Una vez descritos los circuitos, habiéndolos verificado mediante Quartus y después de realizar las simulaciones correspondientes para constatar mediante el software su correcto funcionamiento, se realizó la implementación física. Como ya se mencionó se empleó el FPGA de la famila EPF10k20 incrustado en la tarjeta UP1 de Altera. Primero que nada se comprobó el funcionamiento correcto del

multiplicador únicamente, mediante 2 dip switches de ocho líneas cada uno actuando como entradas variables al FPGA y observando el producto obtenido a través de 16 LEDs conectados a las salidas correspondientes del dispositivo.

En esta etapa de las pruebas, el FPGA se configuró mediante JTAG empleando la interfaz ByteBlasterMV de Altera conectada al puerto paralelo de la PC.

Posteriormente para las pruebas finales con el sistema de prueba completo, se debió usar una memoria de configuración. Era necesario conseguir un dispositivo de configuración con encapsulado PDIP de 8 pines puesto que la tarjeta UP1 únicamente aceptaba ese tipo de circuitos integrados ya que tiene una base con esas características conectada al FPGA dispuesta de tal forma que logre configurar el FPGA cada que se alimente la tarjeta.

Por tal motivo, se optó por usar el EPC1PC8, fabricado también por Altera. Esta memoria es de tipo EPROM. Su forma de operar consiste en almacenar las interconexiones de la descripción realizada en VHDL y a la hora de que el sistema se enciende, este dispositivo manda dicha información al FPGA de forma serial. El proceso de configuración dura unos pocos milisegundos, después de los cuales el FPGA funciona de la manera en que fue especificado. El dispositivo contiene un oscilador interno que funciona como reloj tanto para él mismo como para el FPGA para que la etapa de configuración esté completamente sincronizada.

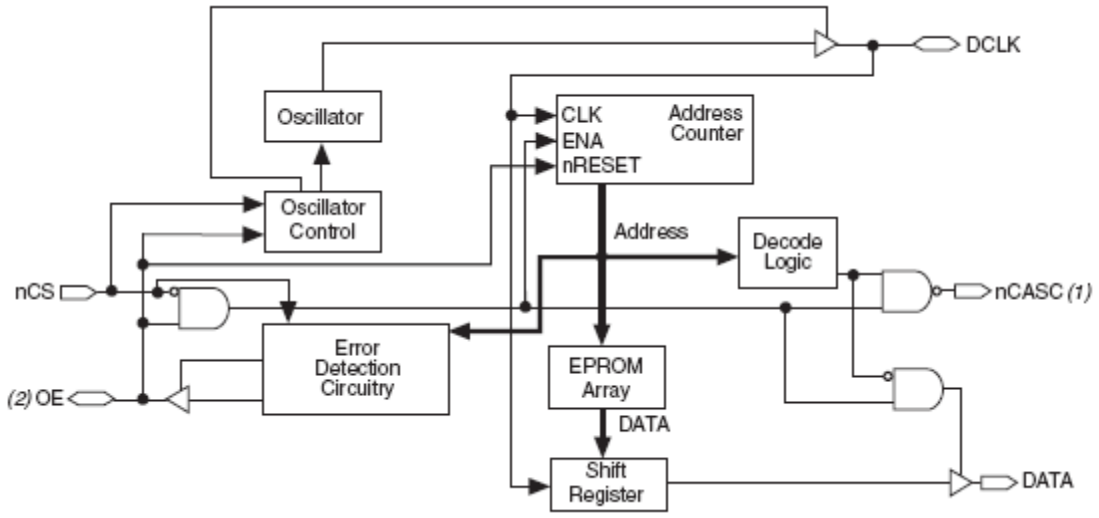


Figura 5.5 Diagrama de bloques del EPC1.

Las desventajas de este tipo de memorias de configuración es que sólo pueden ser programadas una sola vez, puesto que son de tipo EPROM, y no soportan JTAG, por lo que para ser programadas es necesario tener un programador con compatibilidad necesaria, o una interfaz de Altera, como el BitBlaster. En el caso de este proyecto, las memorias fueron programadas en el laboratorio abierto de la Facultad de Ingeniería.

Por otro lado, su costo es bajo, comparado con otras memorias de configuración e inclusive con otros métodos de configuración que pueden requerir un microcontrolador y/o memorias de otro tipo.

En la memoria de configuración se programó el sistema completo, de forma que ya se pudiera usar aunado con la segunda parte del proyecto, puesto que dichos dispositivos no son reprogramables ni muy fáciles de conseguir.

5.5 Programación del microcontrolador PIC

La segunda parte del proyecto trata de un PIC para que actuara como procesador principal del multiplicador. En este trabajo se usó simplemente para fines didácticos, de forma que sirviera para recibir los datos de entrada que se quieren multiplicar, así como el resultado proveniente del FPGA y de alguna manera mostrar toda la operación al usuario.

Se decidió usar un display de cristal líquido para mostrar tanto los multiplicandos como el resultado. Esto es importante ya que requiere de la programación de varias subrutinas específicas para el display dentro del programa principal.

Como se explicó en capítulos anteriores, se eligió el PIC18F452 para cumplir con los objetivos propuestos. Primero que nada, se realizó un diagrama de flujo de lo que debía realizar el PIC. El diagrama se muestra a continuación:

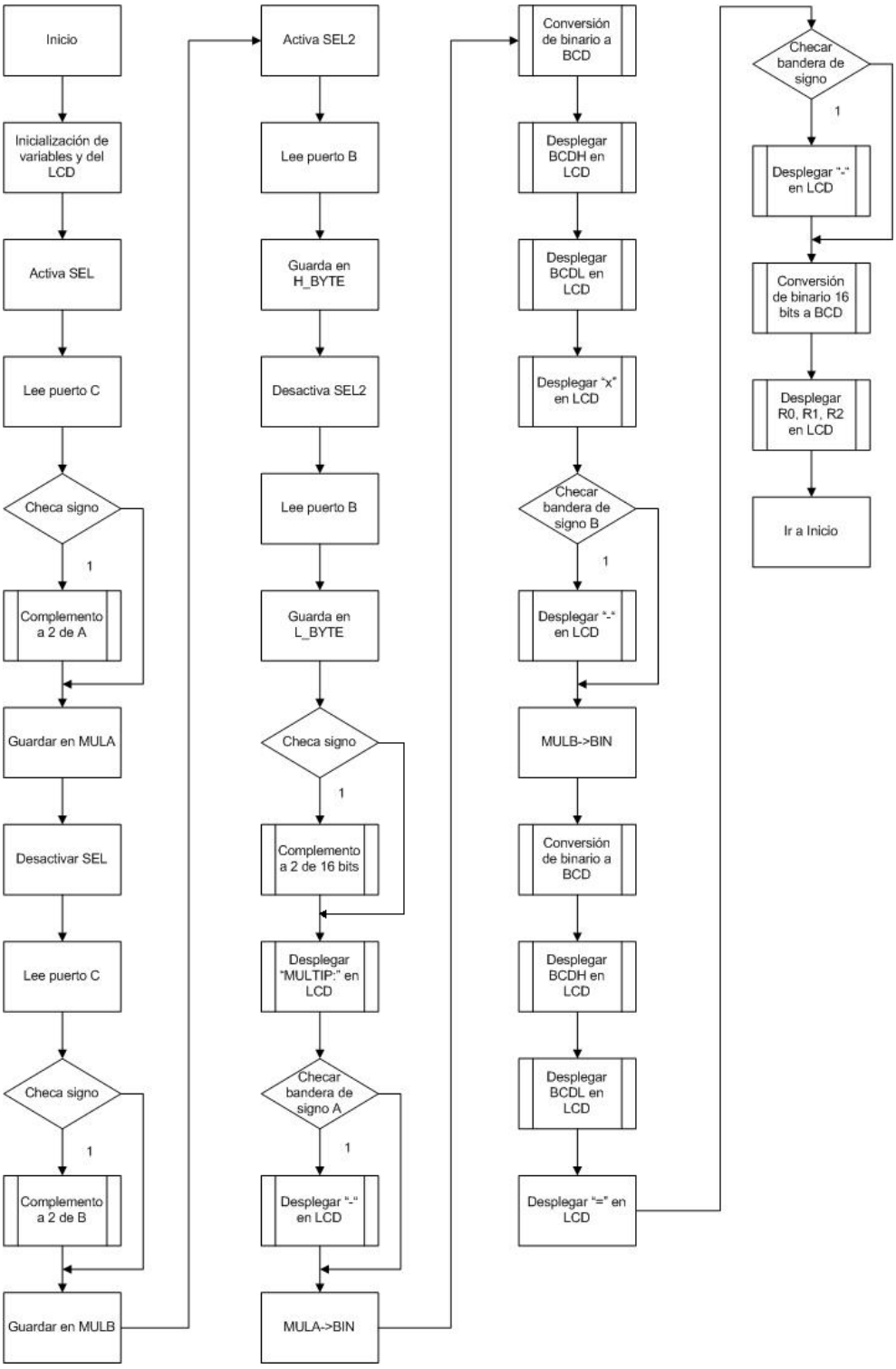


Figura 5.6 Diagrama de flujo del programa del PIC.

Como se puede observar, se tiene una etapa de inicialización del sistema, en el que se limpian variables y se inicializa el display, configurándolo de la forma en la que va a trabajar. Posteriormente se pasa a la etapa de lectura de datos. En esta etapa se manipulan las líneas de selección que se van a enviar al FPGA al mismo tiempo que se leen los multiplicandos, en primer lugar, y las partes alta y baja del resultado de la multiplicación. Durante esta misma etapa, se discrimina entre números positivos o negativos, de forma que si se trata de un número negativo, se le haga un complemento a 2 para poder mostrarlo como lo entendería el usuario. Si se trata de un número negativo, también se activa la respectiva bandera de signo para que pueda ser desplegado el signo "-" en el display, y de esta forma el usuario pueda corroborar que se trata de un número negativo.

A continuación se pasa a desplegar en el display la leyenda "MULTIP: " y enseguida el multiplicando A junto son su signo negativo si es que se trata de un número de esa índole, seguido del signo "x" y después el multiplicando B acompañado, si es necesario, de su signo negativo. Finalmente, se despliega el signo "=" y el resultado, con su signo si es que lo requiere. De esta forma, lo desplegado en el display tomaría la siguiente forma:

```
MULTIP:02x

-03=-0006
```

Para poder desplegar los números en el display, es necesario convertirlos a un formato que pueda entender dicho dispositivo. Por tal motivo, se manda llamar a las subrutinas BIN2BCD en el caso de los multiplicandos, la cual transforma el número binario en un número en código BCD, que más adelante empleará la subrutina BYTEDIS para finalmente enviar al display la información. En el caso del resultado, se debe usar otra subrutina, llamada B2_BCD, ya que ésta trabaja con números de 16 bits dividido en dos partes (alta y baja) que se colocan en dos

registros. Esta subrutina nos regresa tres valores que nuevamente con ayuda de la subrutina BYTEDIS se enviarán al display.

Finalizando la etapa del desplegado de la multiplicación en el display, se repite nuevamente todo el proceso.

Algunas otras subrutinas necesarias para el programa, son la de escritura de datos al display (LCDDAT), escritura de comandos al display (LCDCOM), la propia inicialización del display, las rutinas para realizar el complemento a dos, tanto de los números de 8 bits, como el resultado de 16 bits, y algunas subrutinas de retrasos, necesarias para la lectura de datos y el correcto funcionamiento del display.

El programa se realizó utilizando la herramienta MPLAB IDE de Microchip. El lenguaje empleado fue ensamblador. Se debieron crear todas las subrutinas que se han mencionado, así como el programa principal que realizara lo que se requería.

Para poder asegurar que las subrutinas funcionaban correctamente, previo a la programación del propio dispositivo, se empleó el simulador que se encuentra dentro de MPLAB, llamado MPLAB SIM, con el cual es posible correr el programa paso por paso y analizar el comportamiento del mismo. MPLAB permite observar los registros y unidades de memoria que se requiera, por lo que al correr el programa, se puede observar si se obtiene el resultado esperado en el registro en el que se quiere.

Posteriormente, cuando se finalizó el programa y habiendo probado las subrutinas, el siguiente paso fue programar el PIC y comprobar su funcionamiento.

El hardware de programación que se usó fue el PICSTART PLUS, que permite programar microcontroladores PIC, incluyendo la memoria de programa y bits de configuración entre otras cosas. El PICSTART PLUS permite ser operado

mediante el *software* MPLAB IDE. El *software* y el *hardware* de la programación del PIC se comunican a través de un cable RS232 estándar conectado a la PC.

Figura 5.7 Programador PICSTART PLUS.

5.6 Diseño del circuito de prueba

Para probar el funcionamiento del sistema completo se ideó un circuito de prueba que incluye al PIC, el display de cristal líquido para mostrar la multiplicación al usuario, un par de *dip switches* de ocho líneas cada uno, para modificar el valor de los multiplicandos, y las conexiones hacia el FPGA.

Como el diseño del sistema completo requirió que las entradas de los multiplicandos tanto para el PIC como para el FPGA fueran únicamente de 8 bits, y se fuera conmutando la lectura entre el multiplicando A y el B de acuerdo a la línea de selección generada por el PIC, el *hardware* para realizar las pruebas requería también responder a tal necesidad.

Se decidió utilizar dos circuitos integrados con ocho buffers digitales cada uno, controlados por la línea de selección y un circuito inversor de tal forma que cuando se encuentre en alto dicha señal de control, se active uno de los chips y habilite la lectura de uno de los multiplicandos. La misma señal de selección pasa por un circuito inversor para que cuando la señal se encuentre en bajo, se active el segundo chip y de esta forma leer el otro multiplicando.

El siguiente diagrama muestra lo anterior.

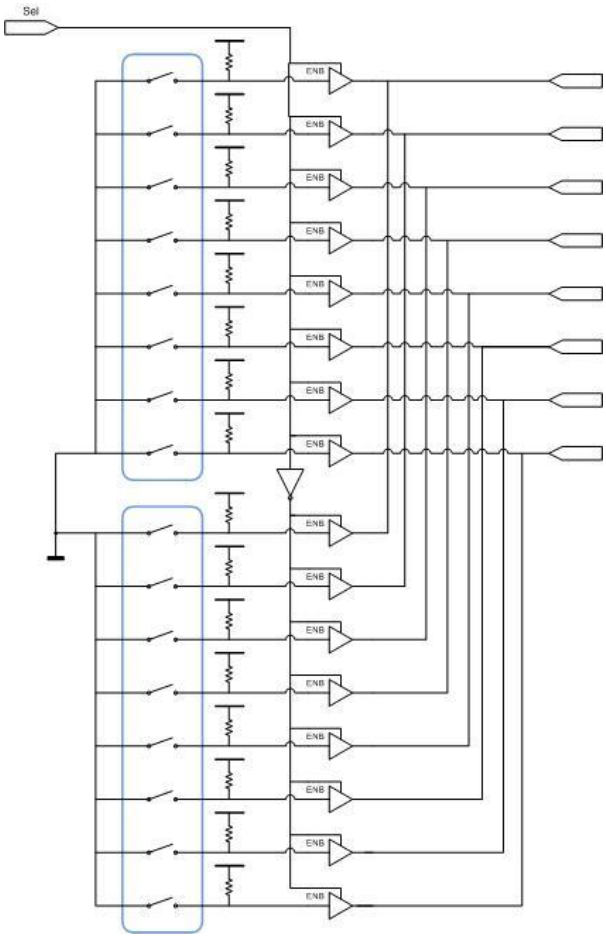
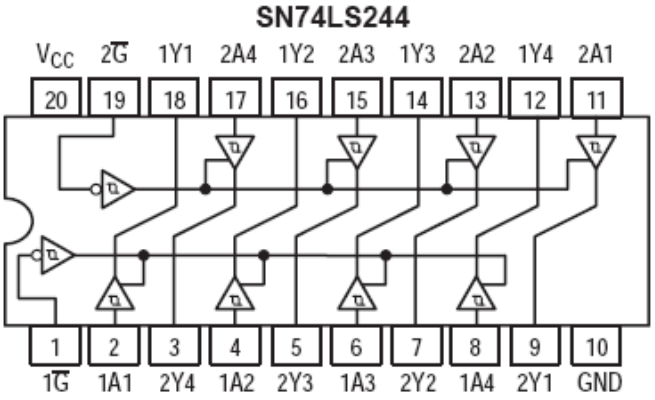


Figura 5.8 Diagrama del circuito de selección de multiplicandos.

Los *buffers* empleados son 74LS244, cada encapsulado incluye 8 *buffers* con su línea de activación. En el circuito integrado se tienen únicamente 2 líneas de selección.

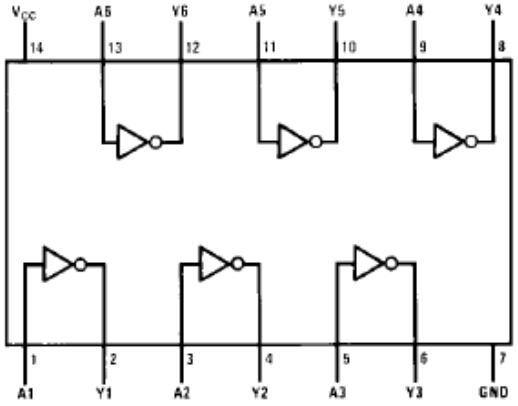


SN74LS244

INPUTS		OUTPUT
1G, 2G	D	
L	L	L
L	H	H
H	X	(Z)

Figura 5.9 Configuración interna y tabla de verdad del 74LS244.

El circuito inversor empleado para conmutar la línea de selección y diferenciar la entrada entre un multiplicando y otro fue el tradicional 74LS04.



$Y = \overline{A}$

Input	Output
A	Y
L	H
H	L

H = HIGH Logic Level
L = LOW Logic Level

Figura 5.10 Configuración interna y tabla de verdad del 74LS04.

De esta forma el circuito de prueba, está compuesto por el PIC, con los componentes externos que éste necesita (cristal, capacitores, etc), los *dip switches* con sus correspondientes buffers y pull ups, así como el circuito inversor para conmutar entre un *dip switch* y otro, el LCD para mostrar la multiplicación, y las bases para lograr la conexión entre el circuito de prueba y la tarjeta UP1 de Altera que contiene el FPGA usado.

.

5.7 Implementación del sistema

La parte final del proyecto fue crear el sistema físico completo. Con este fin, se realizó el diseño del circuito impreso a partir del diagrama esquemático del circuito de prueba. La paquetería PCAD se utilizó para realizar el diagrama de conexiones y corroborar la implementación correcta de los dispositivos electrónicos. A partir de dicho diagrama, se generaron las *netlists* para trasladar las conexiones hacia la herramienta de diseño de PCBs. Dentro de ésta, se acomodaron los componentes de la manera más conveniente posible y se realizaron las conexiones pertinentes.

Finalmente, se realizaron una serie de análisis dentro de la misma paquetería para asegurar que las conexiones realizadas en el diseño del circuito impreso correspondieran íntegramente a las que se encuentran en el diagrama esquemático, así como la verificación de parámetros preestablecidos de diseño de PCBs, como puede ser el de *clearance*, que precisa tener un mínimo de distancia entre líneas de cobre.

El diseño del circuito impreso se realizó con doble capa, puesto que en algunas partes del circuito, especialmente en el área de los buffers, se cruzaban las líneas de conexión. A continuación se muestra el diseño de las caras superior e inferior del PCB.

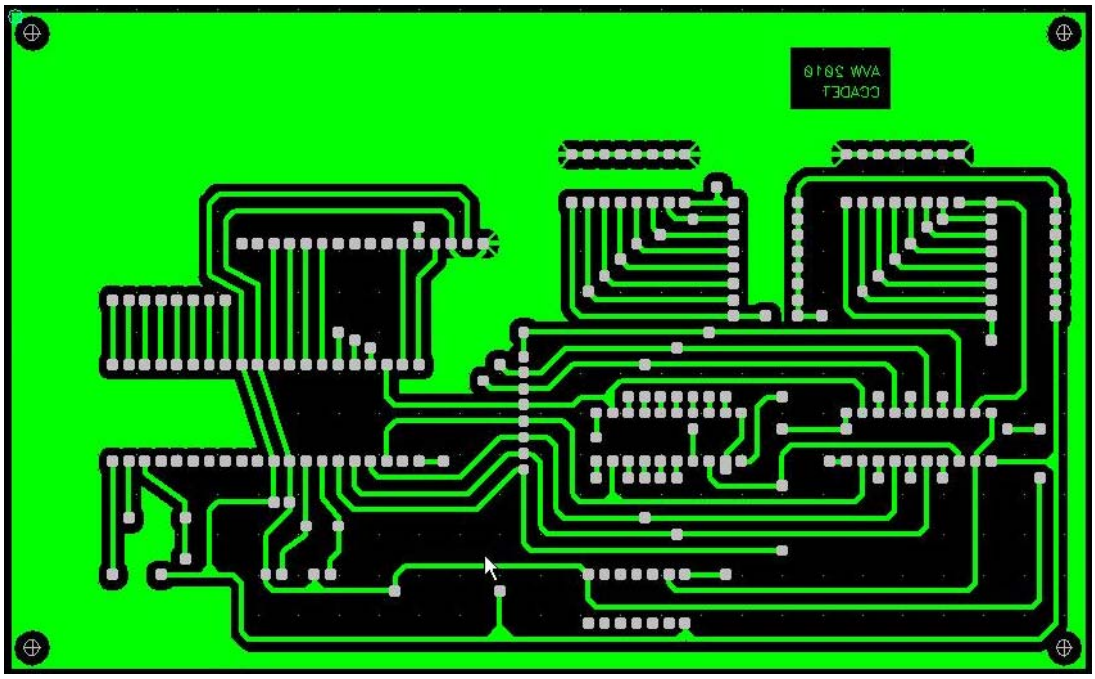


Figura 5.11 Cara inferior del circuito impreso.

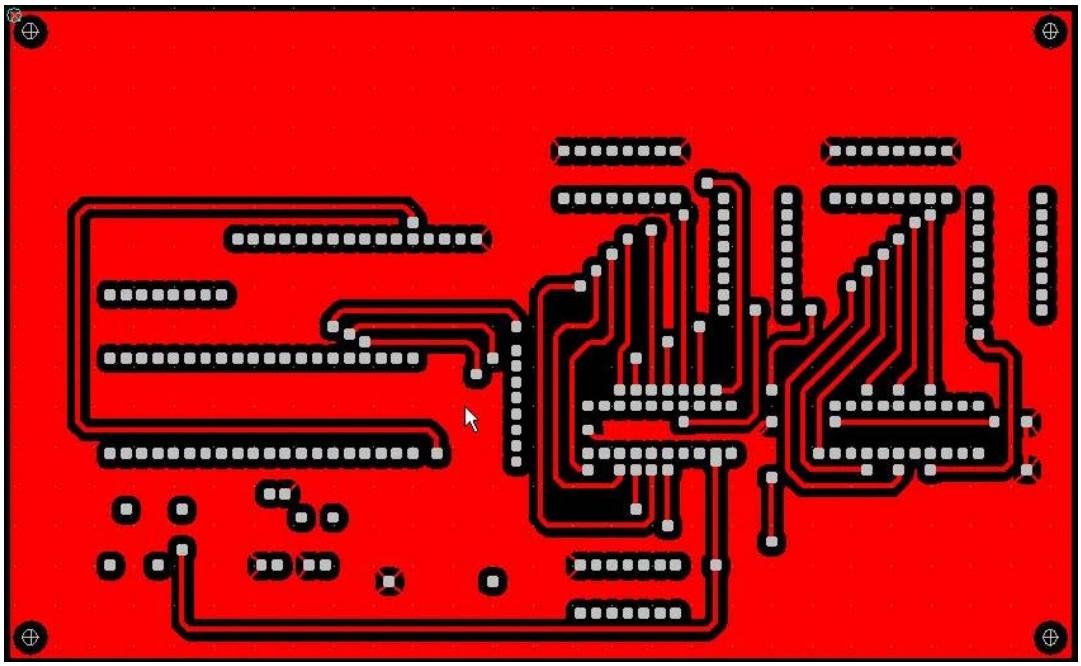


Figura 5.12 Cara superior del circuito impreso.

Una vez diseñado el circuito impreso, se mandó a hacer en el departamento de publicaciones del CCADET, se le soldaron los componentes, y se hicieron los cables de conexión entre el circuito de prueba y la tarjeta UP1 de Altera.

La etapa final de este proyecto consistió en realizar pruebas del funcionamiento del sistema completo, introduciendo números mediante los dip switches y obteniendo los resultados correctos de la multiplicación en el display incrustado en el circuito impreso que se hizo.