UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO Y SIMULACIÓN DE UN MICROPROCESADOR DE PROPÓSITOS ESPECÍFICOS (ASIP) UTILIZANDO EL LENGUAJE DE PROGRAMACION VHDL

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

ULISES ABDON PISCOYA SILVA

PROMOCIÓN
2002 - II

LIMA – PERÚ
2006

DISEÑO Y SIMULACION DE UN MICROPROCESADOR DE PROPÓSITOS ESPECÍFICOS (ASIP) UTILIZANDO EL LENGUAJE DE PROGRAMACIÓN VHDL

Dedico este trabajo a mi familia , a mis compañeros de la Universidad Nacional de Ingeniería , a todas aquellas personas y empresas que me ayudaron a desarrollarme profesionalmente y a todos los docentes de la UNI-FIEE que hicieron posible que esta tesis se desarrolle de la mejor manera.

Además de ello le agradezco a la UNI por darme la oportunidad de ser un profesional de la Ingeniería Electrónica

SUMARIO

En el presente trabajo se muestra el diseño de un microprocesador de propósitos específicos (ASIP) utilizando programación en VHDL y tomando como referencia los dispositivos de la empresa ALTERA . El programador del ASIP podrá utilizar este microprocesador y sus recursos de la manera que estime conveniente.

- En el capitulo I se hacen los primeros planteamientos y descripciones del ASIP , se definen las instrucciones que se desean ejecutar y los ciclos de instrucción.
- En el capitulo II se diseña el ALU (Unidad Aritmética y Lógica) del ASIP , además de ello se realiza una simulación de cada uno de los subsistemas del ALU.
- En el capitulo III se desarrolla el diseño y simulación de los subsistemas de la unidad de control del ASIP.
- En el capitulo IV se desarrolla el diseño y simulación de los subsistemas que forman la unidad de procesamiento , se definen los registros del ASIP , se diseña el decoder de instrucciones , se define el bus de datos , el bus de direcciones y el contador de direcciones.
- En el capitulo V se hace el desarrollo de la arquitectura del ASIP .
- En el capitulo VI se realiza el diseño y la simulación de los subsistemas que producen los saltos e interrupciones con pines externos del ASIP.

En el capitulo VII se elabora un programa TEST para poder verificar y simular todas las instrucciones del ASIP.

- En el capitulo VIII se realiza un reporte acerca de la grabación del programa TEST dentro del FLEX de altera .

INDICE

PROLOGO

El microprocesador es uno de los desarrollos más sobresalientes del siglo XX ; con el paso del tiempo el microprocesador se acerca más al centro de nuestras vidas. Su presencia ha comenzado a cambiar la forma en que percibimos el mundo e incluso a nosotros mismos. Cada vez se hace más difícil pasar por alto el microprocesador como otro simple producto en una larga línea de innovaciones tecnológicas.

Ninguna otra invención en la historia se ha propagado tan rápidamente por todo el mundo o ha tocado tan profundamente tantos aspectos de la existencia humana. Hoy existen casi 15,000 millones de microprocesadores de alguna clase en uso. De cara a esa realidad, ¿quién puede dudar que el microprocesador no sólo está transformando los productos que usamos, sino también nuestra forma de vivir y, por último, la forma en que percibimos la realidad? No obstante que reconocemos la participación del microprocesador en nuestras vidas, ya estamos creciendo indiferentes a la presencia de esos miles de máquinas diminutas que nos encontramos sin saberlo todos los días. Así que, antes de que se integre de manera demasiado imperceptible en nuestra diaria existencia, he decidido en esta tesis estudiar al microprocesador y la revolución que ha originado en la electrónica . En esta tesis he realizado el diseño y simulación de un ASIP . Al diseñar un ASIP en VHDL daremos a conocer mas acerca de esta tecnología de alta integración en el Perú. Con el diseño de un microprocesador de propósitos específicos (ASIP) de 16 bits contribuiremos al desarrollo de la microelectrónica en nuestro país y al desarrollo de aplicaciones en nuestra industria nacional a bajo costo. Esto permitirá obtener una independencia parcial de los fabricantes clásicos de los microprocesadores, microcontroladores, DSP o similares. En esta tesis daremos a conocer una nueva herramienta de diseño electrónico la cual es la programación en VHDL y conoceremos las características técnicas del ASIP las cuales son las mismas que las del PLD o chip que se selecciona para grabar el programa del ASIP.

CAPITULO I:
DESCRIPCIÓN DEL MICROPROCESADOR DE PROPOSITOS ESPECIFICOS (ASIP)

En el presente capitulo realizaremos una breve descripción y los primeros planteamientos de las características técnicas y los subsistemas que forman el ASIP.

1.1 Descripción de las características y especificaciones del ASIP

El ASIP es diseñado en base a un hardware que le permite ejecutar las instrucciones mas importantes y necesarias de los microprocesadores comerciales , además de ello es posible implementar el ASIP en un chip PLD (Dispositivo Lógico Programable) desarrollando así un circuito altamente integrado , el cual puede trabajar a una frecuencia determinada por las características técnicas del chip donde se desea implementar el diseño . El ASIP a diseñar es de arquitectura tipo RISC , la memoria del ASIP es tipo ROM , es decir los datos solamente se almacenaran en los registros .Las especificaciones técnicas del hardware del ASIP es determinado parcialmente por las características técnicas del fabricante del chip , también es posible encontrar un chip de mayor capacidad de almacenamiento en el caso de implementarse en un PLD para realizar futuras ampliaciones ; con subsistemas que le permitan desarrollar otras propiedades al ASIP.

El ASIP posee las siguientes características:

a)_ Dos buses de datos para las señales de entrada A y B de 16 bits cada una .Con los buses de entrada A y B el ASIP opera las señales digitales mediante su arquitectura interna la cual estará gobernado por un programa interno del ASIP el cual se encuentra en su memoria.

b)_ Un bus de datos para la señal de salida C de 16 bits. Con este bus de datos el ASIP da ha conocer el resultado obtenido de ejecutar el programa que se encuentra en la memoria del ASIP.

c)_ Siete registros de propósitos generales , además cualquiera de estos 7 registros se puede comportar como un registro de propósitos específicos . El propósito de estos registros

es almacenar datos de 16 bits dentro del hardware del ASIP para que puedan ingresar mediante 2 subsistemas MUX_B a las entradas del subsistema ALU .

d)_ Un bus de datos interno de 16 bits. Con este bus de datos la memoria del ASIP se puede comunicar con los registros , además de ello los registros se pueden comunicar entre si para almacenar datos.

e)_ Dos tipos de saltos , un salto que ejecuta el programador del ASIP (salto por software) mediante instrucciones de salto propias del ASIP y un salto condicional el cual ocurrirá de acuerdo al nivel lógico de una bandera ó flag el cual toma el nombre DC (salto condicional) , la señal DC será 1 lógico cuando el subsistema ALU realice cualquier operación que de cómo resultado el valor de “0000000000000000” en caso contrario el valor de la señal DC será 0 lógico. Estos dos tipos de saltos solo se realizan si se habilita el bit H del subsistema C _LOGICO .

f)_ Cuatro INTERRUPCIONES por pines externos , los cuales tendrán su propio hardware y son independientes del bit H del subsistema C _LOGICO , estas interrupciones externas hacen que varié la dirección de la memoria del ASIP (subsistema ROM _R) hacia direcciones predefinidas .

G)_ Un bus de direcciones , en el cual se envían las direcciones a la memoria del ASIP , en dichas direcciones de memoria se encuentran almacenadas instrucciones las cuales se representan mediante un número en el sistema binario , la formación de este numero binario se explica en la sección 1.4 de esta tesis.

H)_ Dieciocho tipos de instrucciones del ASIP, Estos dieciocho tipos de instrucciones están compuestos de la siguiente manera :

- Instrucción de Movimiento de un dato del subsistema ROM _R (memoria del ASIP) hacia el subsistema REG_C (registro del ASIP). Debido a que existen 7 registros en el ASIP las instrucciones de este tipo son 7.
- Instrucción de Movimiento de un dato de la entrada del ASIP (Entrada A o B) hacia el subsistema REG_C . Debido a que existen 7 registros en el ASIP las instrucciones de este tipo son 7.
- Instrucciones AND , OR , XOR , NOT . Estas operación las genera el ALU , en este tipo de instrucción es posible realizar 81 instrucciones por cada operación lógica debido a existen 7 registros y 2 entradas en el ASIP, en total son 324 instrucciones de estos cuatro tipos
- Instrucción de TRANSFERENCIA . Este tipo de instrucción genera 9 instrucciones , una para cada registro y para las 2 entradas del ASIP.

- Instrucción de INCREMENTO . Este tipo de instrucción genera 9 instrucciones de este tipo , una para cada registro y para las 2 entradas del ASIP.
- Instrucción de Sustracción con préstamo y Operación sustracción . Estas dos operaciones las genera el subsistema ALU , en estos tipos de instrucciones es posible realizar 81 instrucciones por cada operación de sustracción debido a que existen 7 registros y dos entradas en el ASIP lo cual nos da 81 posibilidades para realizar la sustracción, en total podemos realizar 162 instrucciones de estos dos tipos de instrucciones.
- Instrucción de decremento . Este tipo de instrucción genera 9 instrucciones , una para cada registro y dos instrucciones para las 2 entradas del ASIP.
- Instrucción de salto condicional . Este tipo de instrucción genera 1 instrucción.
- Instrucción de salto por software . Este tipo de instrucción genera 2 instrucciones.
- Instrucción de adición . Esta operación la genera el subsistema ALU . Este tipo de instrucción genera 81 instrucciones .
- Instrucción de Rotación a la derecha y Rotación hacia la Izquierda .Estas operaciones las genera el subsistema ALU , en este tipo de instrucción es posible realizar 9 instrucciones , por cada operación de rotación con lo cual se pueden realizar en total 18 instrucciones de estos dos tipos .
- Instrucción de Interrupción mediante pines externos de prioridad . Debido a que existen 4 interrupciones , en esta tesis se ha considerado que son 4 instrucciones que dependen de el estado lógico de los pines de interrupción IA , IB , IC , ID.

De lo expuesto anteriormente concluimos que el ASIP posee 633 instrucciones los cuales lo he clasificado en 18 tipos de instrucciones.

1.2 Descripción en bloques del ASIP a diseñar

En la figura 1.1 se muestra el diagrama de bloques del ASIP el cual posee los siguientes subsistemas:

a)_ Un subsistema llamado DEMUX _C el cual posee un comportamiento similar a un demultiplexor .

b)_ Una subsistema ROM _R el cual posee un comportamiento similar a una memoria ROM .

c)_ Un subsistema ADAPTADOR el cual divide los datos que envía el subsistema ROM _R y los transfiere hacia todos los subsistemas del ASIP .

d)_ Un subsistema llamado Unidad de Procesamiento, el cual esta formado por otros subsistemas internos que se describirán en la sección 1.6

e)_ Un subsistema REG _H , el cual es un registro externo del subsistema Unidad de Procesamiento, que envía los datos del ciclo de instrucción actual hacia el bus de salida del ASIP o hace que el dato no sea detectado por las salidas del ASIP , este registro es usado por lo general para realizar operaciones intermediasen donde no se produce ninguna variación en la salida del ASIP.

f)_ Un subsistema CONTADOR que permite disponer del bus de direcciones de 64K , el cual permite definir la dirección del subsistema ROM _R.

g)_ Un subsistema llamado C_ LOGICO el cual define la ejecución de una instrucción de salto por software ó una instrucción de salto condicional , el subsistema C_ LOGICO habilitará o deshabilitará las instrucciones de estos dos tipos de salto del ASIP a través de su bit H .

h)_ Un subsistema llamado INTERRUPCIONES , este subsistema lo describiremos en la sección 1.9.

i)_ Los subsistemas DEMUX _C , REG _H, ADAPTADOR , los cuales conforman la Unidad de Control , el cual será descrito en la sección 1.8 .

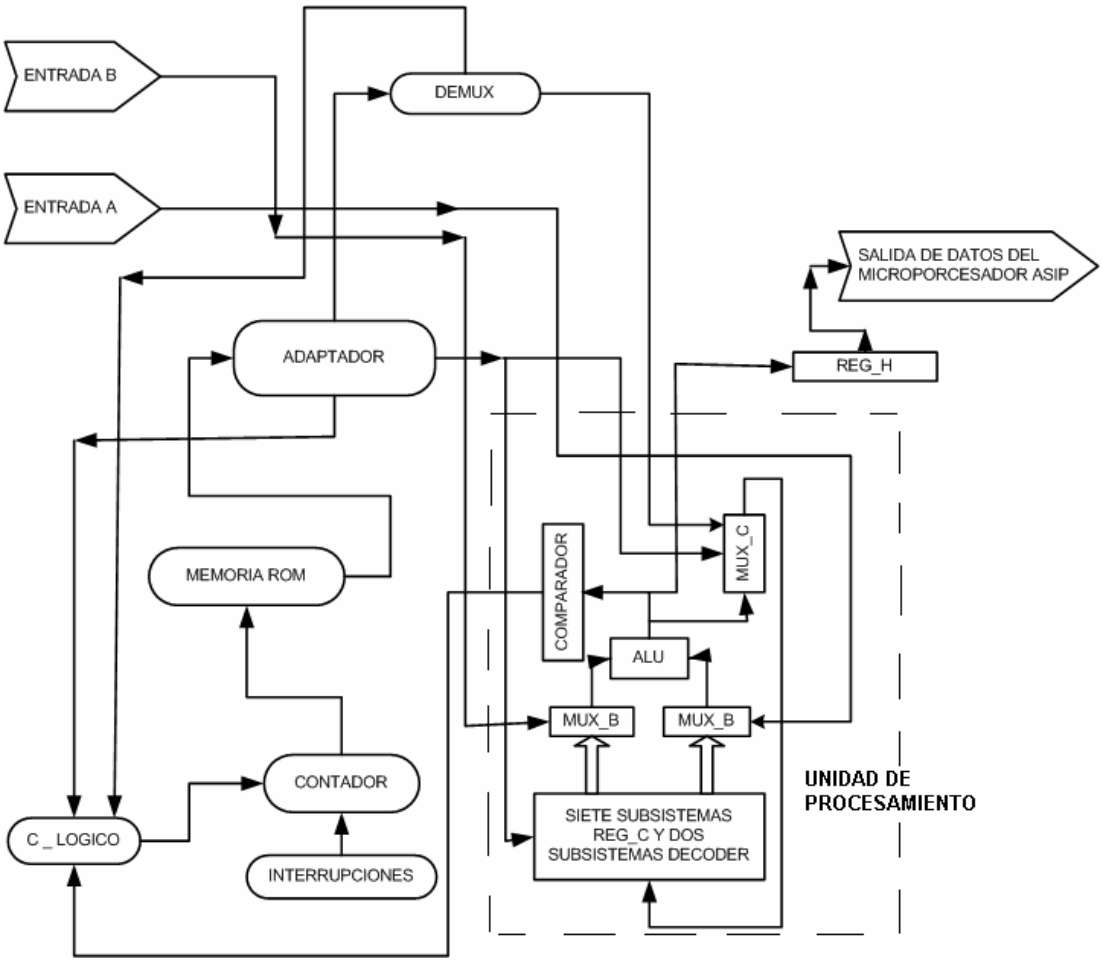


Fig.1.1 Diagrama de bloques del microprocesador de propósitos específicos (ASIP)

Dentro del subsistema Unidad de Procesamiento se encuentran otros subsistemas que han de diseñarse , estos subsistemas son los siguientes :

a)_ Siete subsistemas REG _C , estos subsistemas poseen un comportamiento similar al de los registros de propósitos generales de un microprocesador comercial . Los subsistemas REG _C están interconectados a través de un bus de datos el cual se encuentra en el interior del subsistema Unidad de Procesamiento y su función es enviar las señales de entrada a todos los subsistemas REG _C . Los subsistemas REG_C tienen sus salidas conectadas a los subsistemas MUX _B los cuales seleccionan que señales almacenadas en los subsistemas REG _C ingresarán al subsistema ALU .

b)_ Dos subsistemas DECODER , estos subsistemas poseen un comportamiento similar a los decoders decimales de propósitos generales y se utilizan para controlar los siete subsistemas REG _C antes mencionados.

c)_ Dos subsistemas MUX _B , estos subsistemas poseen un comportamiento similar a los multiplexores 8x1 de propósitos generales.

d)_ Un subsistema ALU , el cual esta formado por varios subsistemas que se describirán en la sección 1.7 .

e)_ Un subsistema COMPARADOR el cual realiza la comparación de la señal de salida del subsistema ALU .

f)_ Un subsistema MUX _C , el cual posee un comportamiento similar al de un multiplexor 2x1 de propósito general.

Todos estos subsistemas antes mencionados son estructurados en un solo subsistema llamado Unidad de Procesamiento. El diagrama de bloques de la Unidad de Procesamiento se muestra en la figura 1.2

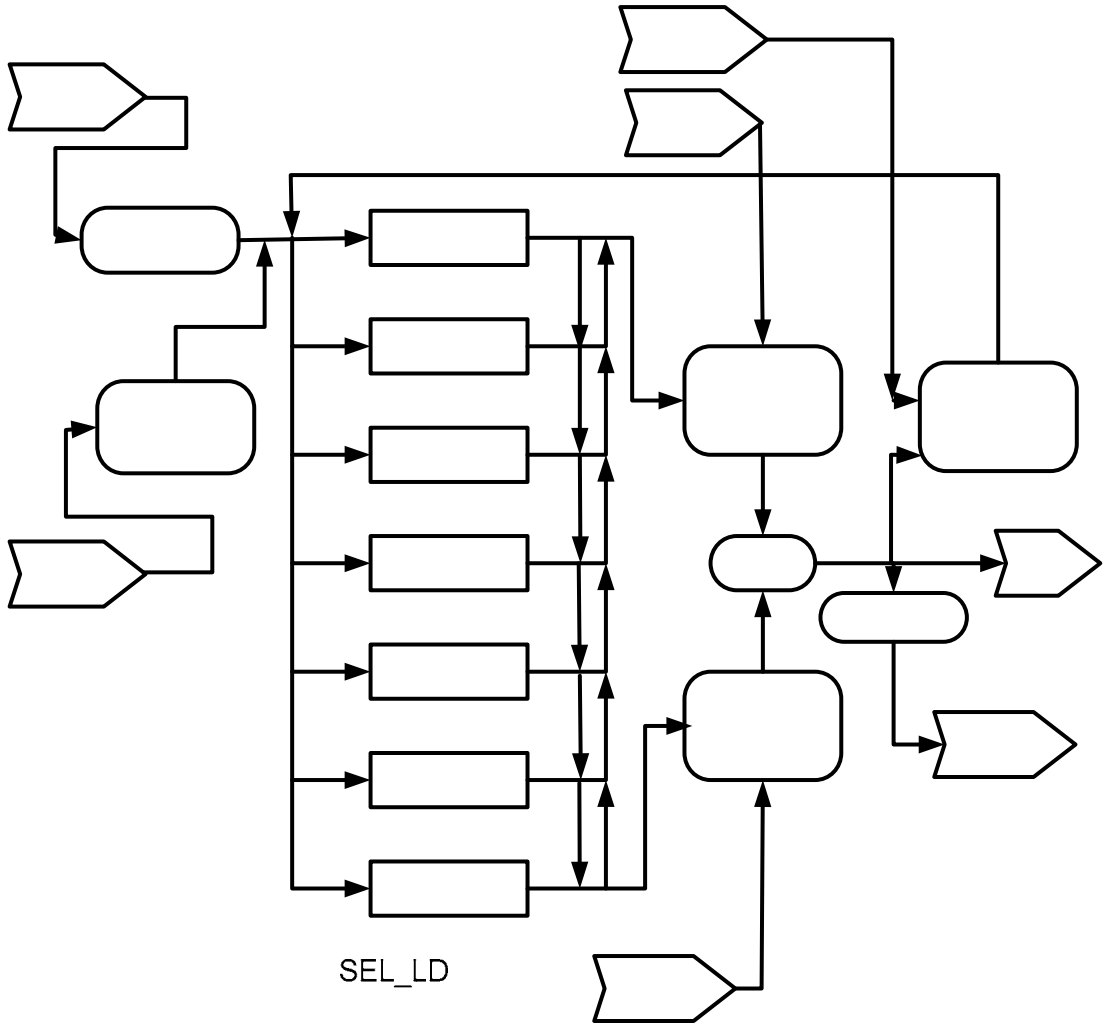


Fig.1.2 Diagrama de bloques del subsistema Unidad de Procesamiento

Dentro de la Unidad de Procesamiento encontramos el subsistema ALU el cual es diseñado conforme al diagrama de bloques de la figura 1.3 . Este subsistema estará compuesto por los siguientes subsistemas :

a)_ Un subsistema llamado ARITHMETIC .

b)_ Un subsistema llamado LOGIC .

c)_ Un subsistema llamado ASHL

d)_ Un subsistema llamado ASHR

e)_ Un subsistema llamado MUX _A .

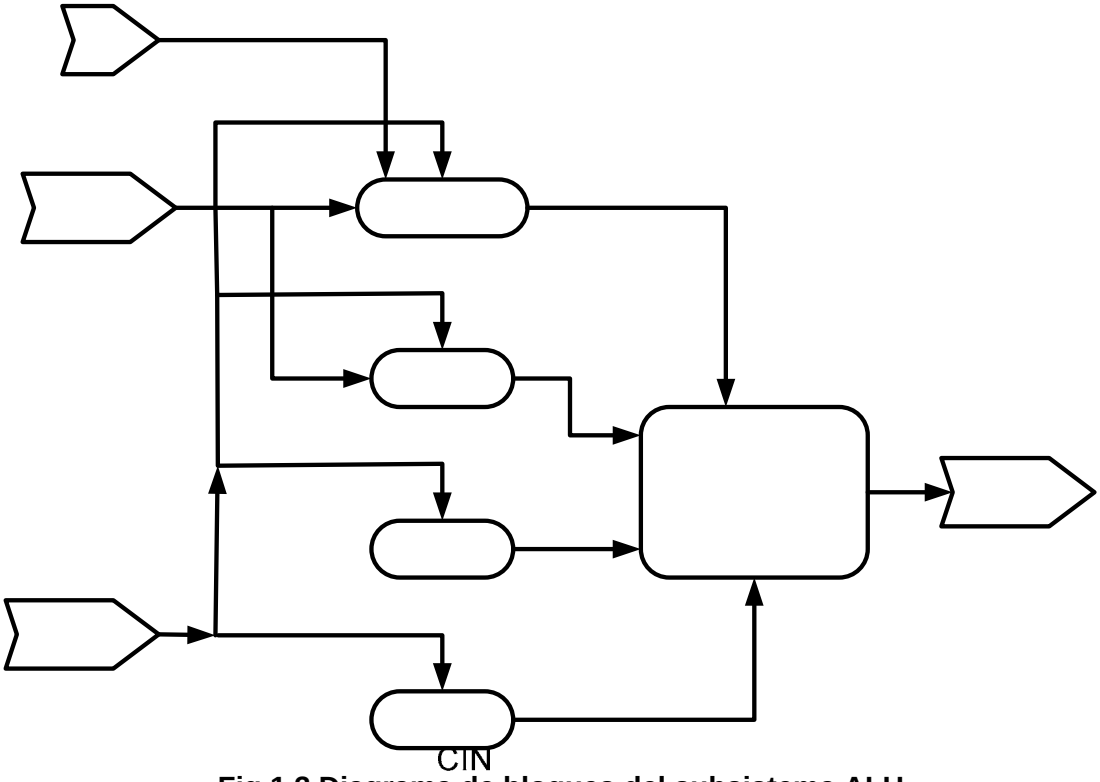


Fig.1.3 Diagrama de bloques del subsistema ALU

1.3 Análisis de los ciclos de instrucción del ASIP

El ciclo de instrucción del ASIP esta dado por el pin PC , el cual es la entrada CLOCK del subsistema CONTADOR .

La entrada CLOCK del ASIP nos da la frecuencia de oscilación del ASIP , en la entrada CLOCK del ASIP se han unido todas las señales de reloj de los subsistemas asíncronos del ASIP.

ENTRADA A ARITHMETIC

LOGIC

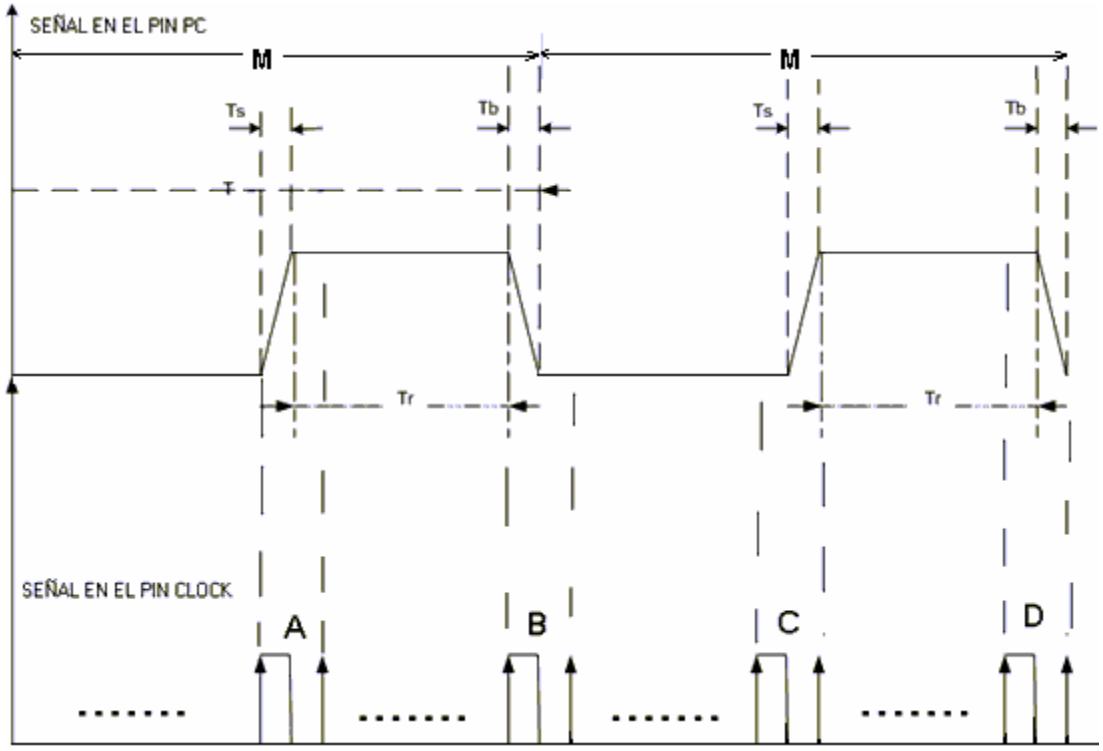


Fig.1.4: Esquema de los ciclos de instrucción del ASIP

En la figura 1.4 se observa el esquema de los ciclos de instrucción del ASIP a diseñar . Los dispositivos PLDs muestran en promedio las siguientes características técnicas:

a)_ tiempo de subida Ts < 4ns

b)_ tiempo de bajada Tb < 4ns

Con estas características técnicas definiré a continuación el ciclo de instrucción , el ciclo de maquina “M” y la frecuencia de reloj del ASIP.

La frecuencia de reloj lo he definido en 8ns debido a que es un valor que puede contener tanto al tiempo de subida (Ts) como al tiempo de bajada (Tb) considerando el caso mas desfavorable y las características técnicas de los PLDs.

Tomando como referencia que los errores permitidos en los sistemas electrónicos son menores del 10% , considero el error de captura de las señales de entrada como 8% lo cual nos da como resultado que el ciclo de instrucción es 100ns .

Además sabiendo que la memoria del ASIP envía las señales de control hacia los subsistemas del ASIP en forma paralela , diseño la frecuencia del ciclo de maquina “M” igual al ciclo de instrucción del ASIP , por lo tanto el ciclo de maquina M es 100ns .

Al realizar estas consideraciones observamos en la figura 1.4 que los ciclos de reloj A , B , C , D del ASIP capturan la señal de entrada PC en valores de tiempos definidos.

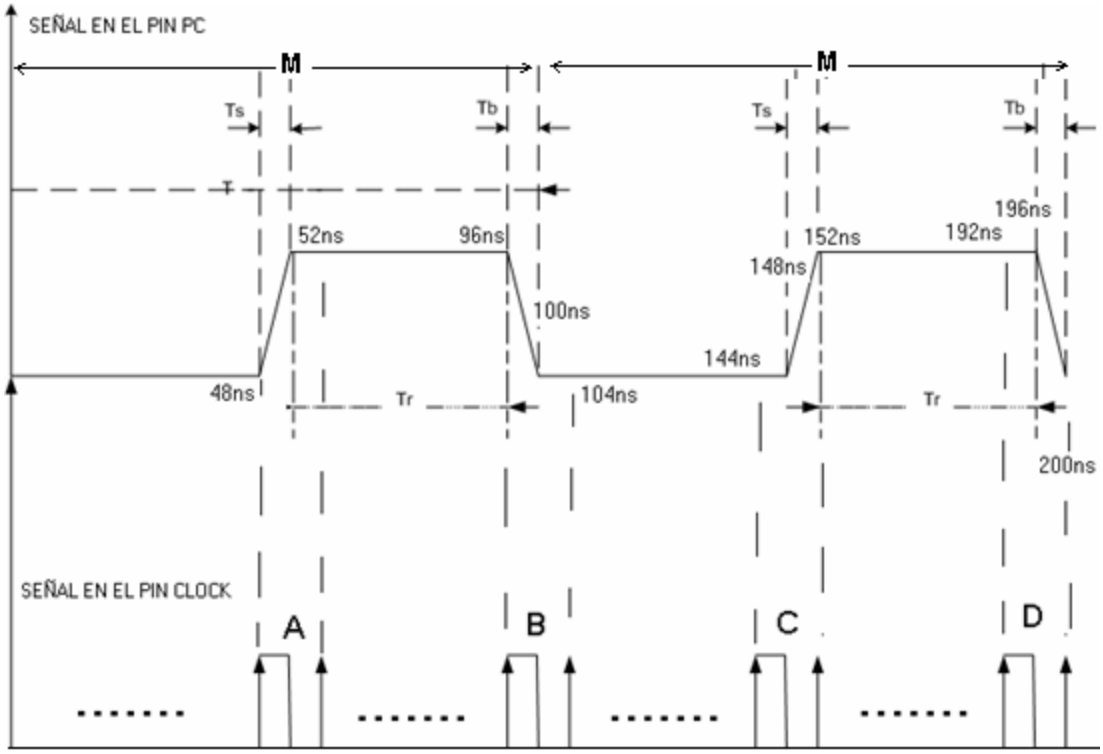


Fig.1.5 : Esquema de los ciclos de instrucción del ASIP considerando la frecuencia de oscilación en 8ns y el ciclo de instrucción en 100ns

De lo dicho anteriormente , en la figura 1.5 se realiza el esquema de los ciclos de instrucción , considerando la frecuencia de oscilación en 8ns (esta señal se encuentra en el pin CLOCK del ASIP) y el ciclo de instrucción de 100ns (esta señal se encuentra en el pin PC del ASIP) , además como se consideró el ciclo de maquina M igual al ciclo de instrucción concluyo que el ciclo de maquina es 100ns.

Considerando el caso mas desfavorable , es decir que Ts = Tb = 4ns se observa en la figura 1.5 que el primer flanco positivo del segmento A de la frecuencia de oscilación se encuentra en 48ns con lo cual el segmento A captura el primer flanco de subida del ciclo de instrucción , el cual es el que mas nos interesa debido a que este flanco es el que hace que se incremente o que se produzca una variación en la dirección de la memoria del ASIP (subsistema ROM _R).

Si ahora observamos el segmento B de la frecuencia de oscilación nos damos cuenta que el primer flanco de subida del segundo ciclo de instrucción se a desfasado 4ns pero eso no es un problema debido a que en el segmento D de la frecuencia de oscilación se observa que el

ultimo flanco de subida de la frecuencia de reloj en el segundo ciclo de instrucción se encuentra justo al final del segundo ciclo de instrucción , en consecuencia este desfasaje de 4ns se supera con pares de ciclos de instrucción los cuales se repiten cíclicamente durante el funcionamiento del ASIP. En el segmento C se observa que la frecuencia de oscilación del ASIP a capturado el segundo flanco positivo del ciclo de instrucción lo cual es nuestra mayor prioridad para que la memoria del ASIP se incremente o cambie de dirección.

Las instrucciones del ASIP se ejecutan en un ciclo de instrucción , con excepción de saltos e interrupciones en donde es necesario dos ciclos de instrucción.

La frecuencia de oscilación del ASIP es de 8ns , con lo cual se concluye que el ASIP necesita un oscilador de 125MHZ

1.4 Instrucciones del ASIP

Las instrucciones del ASIP son de 36 bits y nos permite gobernar todo el hardware del ASIP.

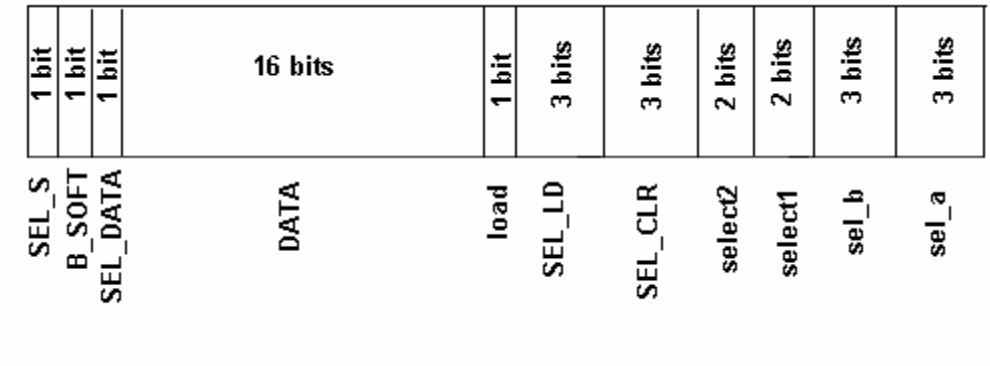


Fig.1.6 : Diseño de una instrucción

En la figura 1.6 se observa el contenido de una instrucción . En el subsistema ADAPTADOR se diseña con este orden la instrucción, en la sección 1.8 se explica la Unidad de Control , en este subsistema se encuentra el subsistema ADAPTADOR y en el capitulo 3 sección 3. 2 se simula el subsistema ADAPTADOR.

El ASIP posee los siguientes tipos de instrucciones:

1_ *Instrucción Movimiento de un dato del subsistema ROM _R hacia el subsistema REG*_C

El símbolo de esta instrucción es: **MOVML Rm , Bd** donde m es un numero del 0 al 6 que expresa en cual de los 7 registros a sido guardada la información, y d es un número binario de 16 bits.

La forma del dato que genera este instrucción es la siguiente:

001(data)0(sel_ld)1110000000000

donde :

sel_ld :es de 3 bits y indica en cual de los 7 registros a sido guardada la información.

data : es de 16 bits y indica el valor del dato binario que se va a almacenar.

2_ *Instrucción Movimiento de un dato de la entrada del ASIP hacia el subsistema REG_C:*

El símbolo de esta instrucción es **MOV A, Rm** donde m es un numero de 0 al 6 que expresa en cual de los 7 registros a sido guardada la información .

La forma del dato que genera este instrucción es la siguiente:

0001000000000000010(sel_ld)1110000000000

donde :

sel_ld :es de 3 bits y indica en cual de los 7 registros a sido guardada la información.

3_ *Instrucción AND* :

El símbolo de esta instrucción es **and Rm,Rn** donde m y n son números del 0 al 6 y expresan los registros que intervienen como operándos , el resultado de esta operación aparecerá en la salida del ASIP.

La forma del dato que genera este instrucción es la siguiente:

10110110001100000100101110100 (sel_b) (sel_a)

sel_a :es de 3 bits y indica cual de los 7 registros va a intervenir en el primer operando de la operación lógica, este instrucción también se puede usar con las entradas del ASIP, si sel_a es 0 entonces se referirá a la entrada A del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

sel_b :es de 3 bits y indica cual de los 7 registros va a intervenir en el segundo operando de la operación lógica , este instrucción también se puede usar con las entradas del ASIP, si sel_b es 0 entonces se referirá a la entrada B del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

4_ *Instrucción OR:*

El símbolo de esta instrucción es **or Rm,Rn** donde m y n son números del 0 al 6 y expresan los registros que intervienen como operándos , el resultado de esta operación aparecerá en la salida del ASIP

La forma del dato que genera este instrucción es la siguiente:

10110110001100000101111110110(sel_b) (sel_a)

donde:

sel_a :es de 3 bits y indica cual de los 7 registros va a intervenir en el primer operando de la operación lógica, este instrucción también se puede usar con las entradas del ASIP, si sel_a es 0 entonces se referirá a la entrada A del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

sel_b :es de 3 bits y indica cual de los 7 registros va a intervenir en el segundo operando de la operación lógica , esta instrucción también se puede usar con las entradas del ASIP, si sel_b es 0 entonces se referirá a la entrada B del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

5_ *Instrucción XOR :*

El símbolo de esta instrucción es **xor Rm,Rn** donde m y n son números del 0 al 6 y expresan los registros que intervienen como operándos , el resultado de esta operación aparecerá en la salida del ASIP

La forma del dato que genera esta instrucción es la siguiente:

10110110001100000010111110101(sel_b) (sel_a)

donde:

sel_a :es de 3 bits y indica cual de los 7 registros va a intervenir en el primer operando de la operación lógica, esta instrucción también se puede usar con las entradas del ASIP, si sel_a es 0 entonces se referirá a la entrada A del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

sel_b :es de 3 bits y indica cual de los 7 registros va a intervenir en el segundo operando de la operación lógica , esta instrucción también se puede usar con las entradas del ASIP, si sel_b es 0 entonces se referirá a la entrada B del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

6_ *Instrucción NOT:*

El símbolo de esta instrucción es **not Rm** donde m es un número del 0 al 6 hace referencia a algún registro del ASIP, el resultado de esta operación aparecerá en la salida del ASIP

La forma del dato que genera esta instrucción es la siguiente:

10110110001100000010111110111010(sel_a)

donde:

sel_a :es de 3 bits y indica cual de los 7 registros va a intervenir, esta instrucción también se puede usar con las entradas del ASIP, si sel_a es 0 entonces se referirá a la entrada A del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

7_ *Instrucción TRANSFERENCIA*

El símbolo de esta instrucción es **transf Rm,out** , donde m es un número del 0 al 6 hace referencia a algún registro del ASIP

La forma del dato que genera esta instrucción es la siguiente:

10110110001100000010111110000010(sel_a)

sel_a :es de 3 bits he indica cual de los 7 registros va a intervenir, esta instrucción también se puede usar con las entradas del ASIP, si sel_a es 0 entonces se referirá a la entrada A del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

8_ *Instrucción Incremento :*

Que en el ASIP es la misma instrucción **transf Rm,out** donde m es un numero del 0 al 6 y expresa el registro que interviene en la operación pero cuando CIN es 1 lógico.

9_ *Instrucción Sustracción con préstamo*

El símbolo de esta instrucción es :**resta Rm,Rn** donde m y n son números del 0 al 6 y expresan los registros que intervienen como operándos y se ejecuta con préstamo cuando CIN es 0 lógico.

La forma del dato que genera esta instrucción es la siguiente:

10110110001100000010010111000 01(sel_b) (sel_a)

donde:

sel_a :es de 3 bits y indica cual de los 7 registros va a intervenir en el primer operando de la operación lógica, esta instrucción también se puede usar con las entradas del ASIP, si sel_a es 0 entonces se referirá a la entrada A del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

sel_b :es de 3 bits y indica cual de los 7 registros va a intervenir en el segundo operando de la operación lógica , esta instrucción también se puede usar con las entradas del ASIP, si sel_b es 0 entonces se referirá a la entrada B del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

10_ *Instrucción sustracción*

El símbolo de esta instrucción es **resta Rm,Rn** donde m y n son números del 0 al 6 y expresan los registros que intervienen como operándos , pero se ejecuta cuando CIN es 1 lógico , esta es la resta que normalmente nosotros conocemos.

11_ *Instrucción Decremento*

El símbolo de esta instrucción es **tran Rn,out** , donde n es un número del 0 al 6 hace referencia a algún registro del ASIP y se ejecuta cuando CIN es 0 lógico , si CIN es 1 lógico realiza una transferencia.

La forma del dato que genera esta instrucción es la siguiente:

10110110001100000010111110001110(sel_a)

donde:

sel_a :es de 3 bits y indica cual de los 7 registros va a intervenir, esta instrucción también se puede usar con las entradas del ASIP, si sel_a es 0 entonces se referirá a la entrada A del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

12_ *Instrucción de Salto Condicional*

La cual se realiza con la instrucción COMPARADOR , esta instrucción hace un salto hacia cualquier dirección especificada por el programador del ASIP si el valor de la salida del subsistema ALU es cero, para lo cual el programador del ASIP debe además habilitar el flag o bandera H desde el subsistema ROM _R. esta instrucción se realiza en 2 ciclos de instrucción.

La forma de los dos datos que genera esta instrucción es la siguiente:

000(data)01111110000000000

000(data)01111110000000000

donde:

data : es de 16 bits y indica el valor de la dirección a la cual se va a saltar.

13_ *Salto por Software* :

Para el cual se **necesitan 2 instrucciones** , la instrucción “salta_n” y la instrucción “goto_n”, la primera de ellas es para que salte a una dirección especifica del subsistema ROM _R y la segunda es para que el hardware pueda darse cuenta que a habido un salto. Además n es un numero que indica la dirección de la memoria que se va tener acceso mediante el salto .

La forma de los el datos que genera la instrucción salta_n es:

010(data)11111110010010001

La forma de los el datos que genera la instrucción goto_n es:

000(data)01111110000000000

donde:

data : es de 16 bits y indica el valor de la dirección a la cual se va a tener acceso mediante el salto.

14_ *Instrucción Adición*

El símbolo de esta instrucción es **suma Rm,Rn** donde m y n son números del 0 al 6 y expresan los registros que intervienen como operándos.

La forma del dato que genera esta instrucción es la siguiente:

10110110001100000100101110010(sel_b) (sel_a)

Donde:

sel_a :es de 3 bits e indica cual de los 7 registros va a intervenir en el primer operando de la operación lógica, esta instrucción también se puede usar con las entradas del ASIP, si sel_a es 0 entonces se referirá a la entrada A del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

sel_b :es de 3 bits y indica cual de los 7 registros va a intervenir en el segundo operando de la operación lógica , esta instrucción también se puede usar con las entradas del ASIP, si sel_b es 0 entonces se referirá a la entrada B del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

15_ *Instrucción Rotación a la derecha :*

El símbolo de esta instrucción es : **ASHR Rn,out** donde n es un número del 0 al 6 hace referencia a algún registro del ASIP.

La forma del dato que genera esta instrucción es la siguiente:

10110110001100000101111111100(sel_b)001

donde:

sel_b :es de 3 bits y indica cual de los 7 registros va a intervenir en el segundo operando de la operación lógica , esta instrucción también se puede usar con las entradas del ASIP, si sel_b es 0 entonces se referirá a la entrada B del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

16_ *Instrucción Rotación a la izquierda*

el símbolo de esta instrucción es : **ASHL Rn,out** donde n es un número del 0 al 6 hace referencia a algún registro del ASIP.

La forma del dato que genera esta instrucción es la siguiente:

10110110001100000101111111000(sel_b)001

sel_b :es de 3 bits e indica cual de los 7 registros va a intervenir en el segundo operando de la operación lógica , esta instrucción también se puede usar con las entradas del ASIP, si sel_b es 0 entonces se referirá a la entrada B del ASIP , si es 1 se referirá al registro R0 y así sucesivamente hasta 7 en donde se referirá a R6 .

17_ *Interrupción mediante pines externos de prioridad*

Son 4 pines externos de prioridad los cuales realizan saltos hacia una dirección definida por el diseñador del ASIP . Este tipo de interrupciones nosotros lo consideramos como comandos del ASIP con lo cual se concluye que el microprocesador ASIP tiene 4 instrucciones de este tipo.

Realizando todas estas consideraciones se concluye que el ASIP tiene **18** tipos de instrucciones y de expuesto en la sección 1.1 de esta tesis el ASIP posee **633** instrucciones.

1.5 Descripción de la memoria del ASIP

El subsistema ROM _R[1] es un programa en VHDL que se comporta como la memoria del ASIP , dicha memoria es una ROM[2] .

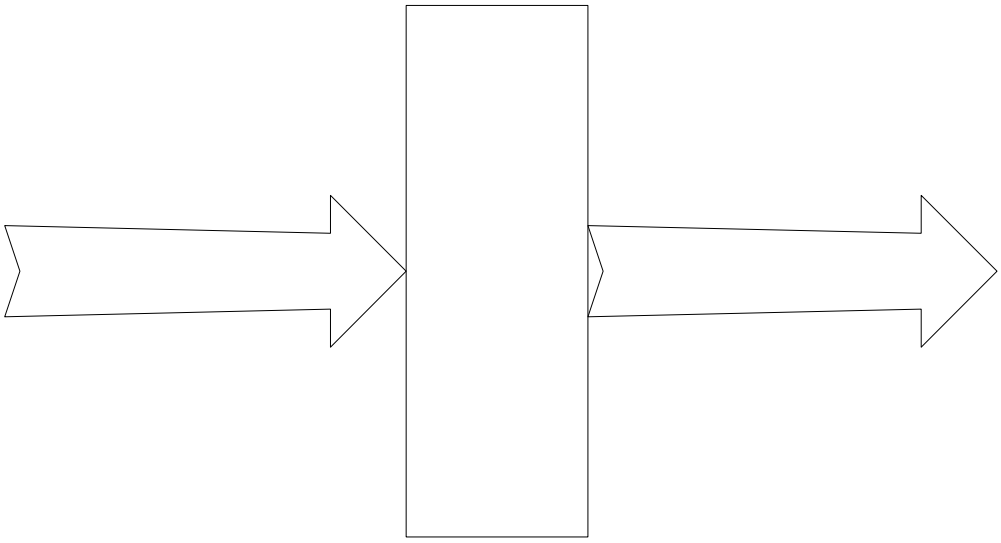


FIGURA 1.7 Esquema de la memoria del ASIP (subsistema ROM _R)

En la figura 1.7 se muestra el esquema de la memoria del ASIP el cual se a definido como subsistema ROM _R . De acuerdo a la figura 1.7 el subsistema ROM _R esta compuesto por

[1] En el apéndice H sección H.17 se encuentra el programa ROM _R. VHDL con el programa TEST el cual se explicará en el capitulo 9

[2] En el apéndice B se comenta la clasificación de las memorias

16 bits de dirección es decir posee 65536 direcciones y cada dirección almacena un dato respectivo el cual es de 36 bits.

El subsistema ROM _R no posee un reloj , debido a que este subsistema debe cambiar su señal de salida cada vez que cambie su señal de entrada ADDR .

En el subsistema ROM _R se guardarán los datos de 16 bits que generan las instrucciones del programa principal , para poder almacenar los datos en el subsistema ROM _R es necesario definir la dirección de dichos datos en el programa ROM _R .VHDL .

Debido a que los datos y las direcciones son definidas dentro del programa ROM _R .VHDL el subsistema ROM _R no necesita los pines de habilitación de escritura o de lectura que poseen las ROM de propósito general debido a que el subsistema ROM _R siempre esta en modo lectura y cuando se desea escribir los datos en el subsistema ROM _R se hace dentro del programa ROM _R .VHDL .

En el subsistema ROM _R el programador del ASIP escribirá el programa que opera la arquitectura del ASIP , el cual es llamado programa principal del ASIP , una vez finalizado la escritura del programa del ASIP el subsistema ROM _R genera una secuencia de bits en paralelo mediante el cual opera el hardware del ASIP , esta secuencia de bits representan que las instrucciones ASIP necesita para realizar sus operaciones internas. En la figura 1.8 se muestra el símbolo del subsistema ROM _R

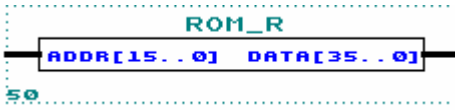


Fig.1.8 Símbolo del subsistema ROM _R. SYM

1.6 Descripción de la unidad de procesamiento del ASIP

El símbolo del subsistema Unidad de Procesamiento se muestra en la figura 1.9

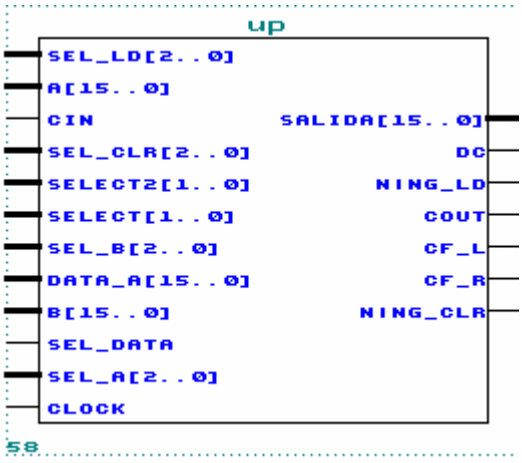


Fig.1.9 Símbolo del subsistema Unidad de Proceso

La Unidad de Procesamiento del ASIP esta formado por los siguientes subsistemas:

A)_ Siete subsistemas REG _C [3] que se comportan como registros[4] de propósito general . El subsistema REG _C almacena la información para que el subsistema ALU pueda efectuar sus operaciones . En la figura 1.10 observamos el símbolo del subsistema REG _ C .

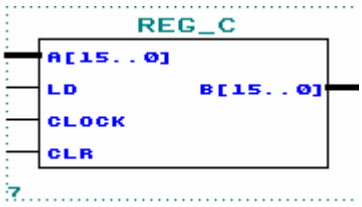


Fig.1.10 Símbolo del subsistema REG _C. SYM

B)_ Un subsistema ALU que recibe las señales de salida de dos subsistemas MUX _B . El subsistema ALU se diseñará en el capitulo 2.

C)_ Dos subsistemas llamados DECODER[5] que hacen que los subsistemas REG _C puedan estar en CLEAR o realizar la carga de los datos . Su señal de entrada AIN es de 3 bits y sus salidas son 8 bits . Los dos subsistemas DECODER gobiernan el almacenamiento de datos y el CLEAR de los subsistemas REG _C , el primero de estos subsistemas nos indica en que subsistema REG _C se va almacenar la información , el segundo subsistema

[3] El programa REG _ C .VHDL se encuentra en el apéndice H en la sección H.7
[4] En el apéndice D se explica de una manera mas detallada que es un registro y algunos diseños de registros
[5] El programa DECODER .VHDL se encuentra en el apéndice H en la sección H.6

DECODER lo uso en el diseño para realizar un reset en cualquiera de los subsistemas REG _C .

En la figura 1.11 muestro el símbolo del subsistema DECODER.

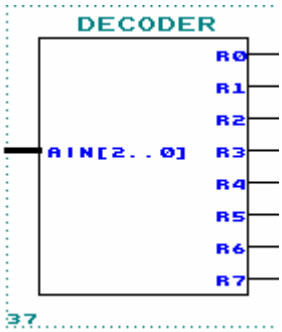
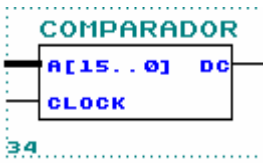


Fig.1.11 Símbolo del subsistema DECODE. SYM

D)_ Un subsistema llamado COMPARADOR[6] el cual compara si la salida del ALU es cero . Cada vez que la salida del subsistema ALU es cero , el subsistema COMPARADOR hace que su bit de salida DC sea igual a 1 lógico en caso contrario su valor será 0 lógico . El subsistema COMPARADOR brinda la información del bit DC[7] al subsistema C _LOGICO . En el caso de ser 1 lógico el bit DC , el ASIP puede realizar un salto por comparación[8] , en caso contrario no se realizará el salto por comparación debido a que la salida del subsistema ALU no es cero . Se define saltos por software como aquellos saltos que el programador del ASIP los ejecuta en cualquier parte del programa principal para poder realizar el algoritmo que controla el hardware del ASIP . Para poder realizar los saltos por software o por comparación es necesario que el bit H del subsistema C _LOGICO se encuentre en 1 lógico, si el bit H es 0 lógico no se realizaran los saltos por software o por comparación , los únicos saltos que son posibles de realizar sin importar el valor del bit H son los saltos por interrupción de los pines externos , a este tipo de saltos lo defino como interrupciones, estas interrupciones generan saltos hacia una dirección definida por el diseñador del ASIP. En la figura 1.12 se muestra el símbolo del subsistema COMPARADOR



[6] El programa COMPARADOR .VHDL se encuentra en el apéndice H en la sección H.9

[7] existen dos señales que la he denominado bit DC una esta en el subsistema COMPARADOR y la otra esta en el subsistema C _LOGICO

[8] No necesariamente cuando DC = 1 lógico el ASIP realiza un salto debido a que es necesario que el bit H sea 1 lógico

Fig.1.12 Símbolo del subsistema COMPARADOR. SYM

E)_ Dos subsistemas llamados MUX _B[9] los cuales tienen un comportamiento que será similar a un multiplexor 8x1 , el subsistema MUX_B es gobernado por un CLOCK , además los subsistemas MUX _B habilitan a dos señales de salida de dos subsistemas REG _C , los subsistemas REG _C enviarán a las entradas del subsistema ALU sus datos almacenados con los cuales se realizarán operaciones internas . El subsistema MUX _B puede también enviar a las entradas del subsistema ALU las señales de entrada A y B del ASIP para generar operaciones. El subsistema MUX_ B posee una entrada de selección SEL de 3 bits , 8 entradas de datos de 16 bits cada uno los cuales son: A , B , C , D , E , F , G , H y una señal de salida SAL de 16 bits .

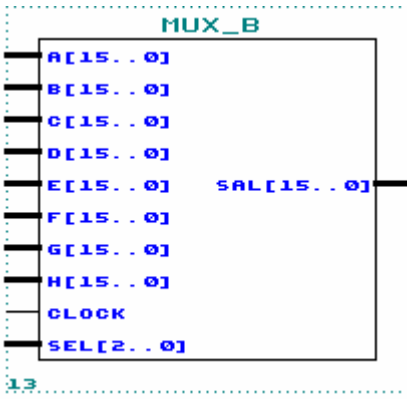


Fig.1.12 Símbolo del subsistema MUX _ B. SYM

F)_ Un subsistema MUX _C[10] el cual tiene un comportamiento que será similar a un multiplexor 2x1, el subsistema MUX _C es gobernado por un CLOCK y tiene la función dentro de la Unidad de Procesamiento de decidir si los datos que ingresan a los subsistemas REG _C provienen de la salida del subsistema ALU o del subsistema DEMUX _C . El subsistema MUX _C posee una señal de entrada de selección definida como SEL _DATA , dos señales de entrada de 16 bits , una de estas entradas esta unida a la salida del subsistema ALU y la otra esta unida a la salida B del subsistema DEMUX _C . La salida del subsistema MUX _C esta unida hacia el bus de datos que son las entradas de los subsistemas REG _C . En la figura 1.13 se muestra el símbolo del subsistema MUX _C .

[9] El programa MUX _ B .VHDL se encuentra en el apéndice H en la sección H.8
[10] El programa MUX _C .VHDL se encuentra en el apéndice H en la sección H.16

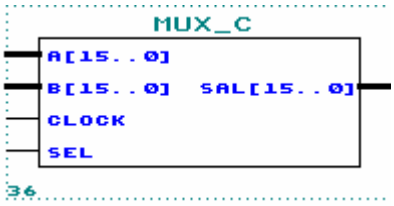


Fig.1.13 Símbolo del subsistema MUX _C. SYM

1.7 Descripción del subsistema ALU

La función del subsistema ALU es realizar las operaciones aritméticas , lógicas y corrimiento. En la figura 1.14 se muestra el símbolo del subsistema ALU.

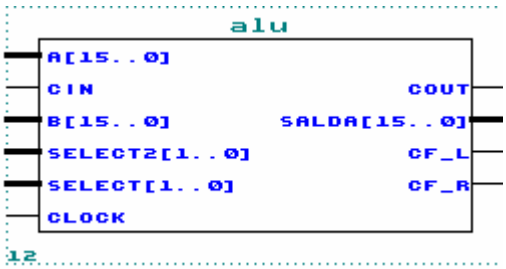


Fig.1.14 Símbolo del subsistema ALU .SYM diseñado

Para las operaciones aritméticas se diseña el subsistema ARITHMETIC[11] el cual posee un selector de 2 bits, una entrada CIN para el acarreo de entrada, una señal de entrada llamada CLOCK que es una entrada que proporciona el sincronismo de cada operación aritmética y dos señales de entrada llamadas A y B . El subsistema ARITHMETIC tiene una salida Q en la cual se encuentran los bits de la operación realizada y la salida COUT el cual es un bit de acarreo de la salida del subsistema ARITHMETIC. En la figura 1.15 se muestra el símbolo del subsistema ARITHMETIC.

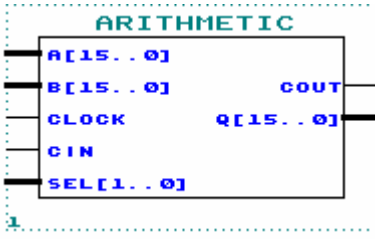


Fig.1.15 Símbolo del subsistema ARITHMETIC .SYM diseñado

[11] El programa ARITHMETIC .VHDL se encuentra en el apéndice H en la sección H.10

Para las operaciones lógicas se diseña el subsistema LOGIC [12] . En la figura 1.16 se muestra el símbolo del subsistema LOGIC .

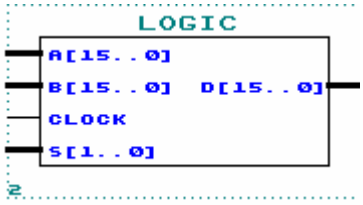


Fig.1.16 Símbolo del subsistema LOGIC .SYM diseñado

El subsistema LOGIC proporciona al subsistema ALU la capacidad de hacer operaciones lógicas que son seleccionadas por la entrada “S”, además este subsistema contiene una entrada de reloj de nombre CLOCK que nos da el sincronismo de cada operación lógica.

El subsistema ASHR[13] realiza un movimiento de bits hacia la derecha. En el subsistema ALU utilizaremos un subsistema de este tipo para realizar el corrimiento a la derecha de la entrada “B” del subsistema ALU. En la figura 1.17 se muestra el símbolo del subsistema ASHR.

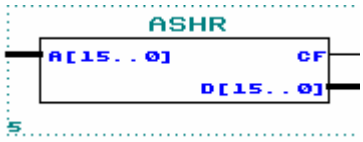


Fig.1.17 Símbolo del subsistema ASHR .SYM diseñado

El subsistema ASHL[14] realiza un movimiento de bits hacia la izquierda . En el subsistema ALU utilizo un subsistema ASHL para realizar el corrimiento a la izquierda de la entrada “B” del subsistema ALU. En la figura 1.18 se muestra el símbolo del subsistema ASHL.

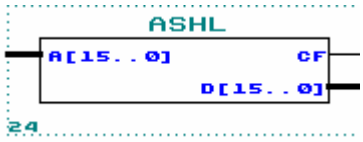


Fig.1.18 Símbolo del subsistema ASHL .SYM diseñado

Debemos dar hincapié además de que CF del subsistema ASHL no es la misma señal que el CF del subsistema ASHL pero ambas son señales internas del subsistema ALU.

---

[12] El programa LOGIC se encuentra en el apéndice H en la sección H.11

[13] El subsistema ASHR es conocido como subsistema de corrimiento a la derecha .El programa ASHR .VHDL se encuentra en el apéndice H en la sección H.13

[14] El subsistema ASHL es conocido como subsistema de corrimiento a la izquierda .El programa ASHL .VHDL se encuentra en el apéndice H en la sección H.12

Las señales de salida de los subsistemas ARITHMETIC ; LOGIC , ASHL , ASHR no son al mismo tiempo las señales de salida del subsistema ALU , de estos 4 subsistemas mencionados se escogerá una señal de salida para el subsistema ALU , el subsistema MUX_ A[15] elegirá cual de estos 4 subsistemas enviará su señal de salida hacia la salida del subsistema ALU de acuerdo a la señal de entrada SEL.

El subsistema MUX _A [16] realiza el control de la señal de salida del subsistema ALU. El funcionamiento del subsistema MUX _A es idéntico al de un multiplexor convencional 4x1 . Dentro del subsistema ALU utilizo un subsistema MUX _A para controlar las señales de salida del subsistema ALU . El subsistema MUX _A posee 1 entrada de selección SEL de 2 bits , 4 entradas de datos de 16 bits cada una denominadas A , B , C , D y una salida de 16 bits denominada SAL . En la figura 1.19 se muestra el símbolo del subsistema MUX _A

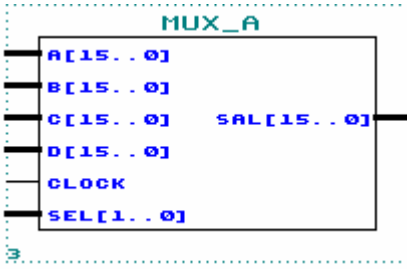


Fig.1.19 Símbolo del subsistema MUX_ A. SYM diseñado

1.8 Descripción de la Unidad de Control

La Unidad de Control esta compuesto por los siguientes subsistemas:

A)_ El subsistema ADAPTADOR[17] el cual divide la data del subsistema ROM_R para que puedan recibir la información todos los subsistemas del ASIP, además de ello define que bits del subsistema ROM _R se utilizan para controlar determinados subsistemas del ASIP .

El subsistema ADAPTADOR envía los datos que se encuentran almacenados en el subsistema ROM _R en forma paralela evitando posibles colisiones de datos y optimizando la velocidad del ASIP debido a que el subsistema ADAPTADOR posee un retardo despreciable .

El subsistema ADAPTADOR no tiene un pin de reloj , es decir apenas cambie el valor de la entrada A de este subsistema cambiará el valor de sus salidas .

En la figura 1.20 se muestra el símbolo del subsistema ADAPTADOR.

---

[15] El programa MUX _ A .VHDL se encuentra en el apéndice H en la sección H.14
[16] El programa MUX _A .VHDL se encuentra en el apéndice H en la sección H.14
[17] El programa ADAPTADOR .VHDL se encuentra en el apéndice H en la sección H.4

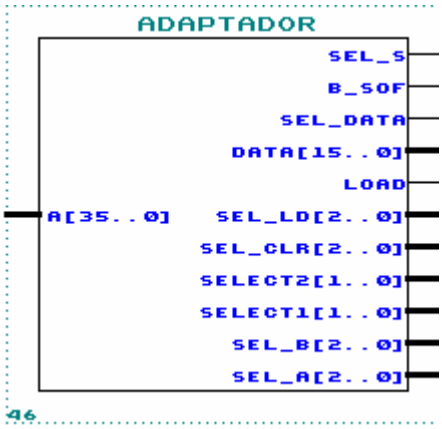


Fig.1.20 Símbolo del subsistema ADAPTADOR. SYM

B)_ El subsistema DEMUX _C[18] tiene una señal de entrada de 16 bits el cual lo podemos usar de dos formas , la primera forma es que esta señal nos sirva de dirección de salto por software o por comparación y la segunda forma es utilizar esta señal para cargar un valor a cualquiera de los 7 subsistemas REG _C , el modo de funcionamiento de este subsistema es igual al de un demultiplexor convencional , la señal de entrada SEL del subsistema DEMUX _C sirve para decidir hacia donde va la información de la señal de entrada I , si SEL = 0 la señal de entrada I se encontrará en la salida A del subsistema DEMUX _C y la enviará al subsistema CONTADOR con lo cual dicha señal dará la dirección del salto por software o comparación , si SEL = 1 la información de la señal de la entrada I se encontrará en la salida B del subsistema DEMUX _C y la enviará al subsistema Unidad de Procesamiento con lo cual la data se almacenará en alguno de los 7 subsistemas REG _C a través del bus de datos . En la figura 1.21 se muestra el símbolo del subsistema DEMUX_C.

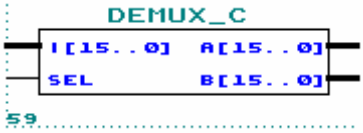


Fig.1.21 Símbolo del subsistema DEMUX. SYM

C)_ El subsistema REG _H es un registro de propósitos específicos, el cual deja pasar la información hacia los pines de salida del ASIP cuando es activado a 1 la señal de entrada LD del subsistema REG _H , en caso contrario la información no pasa hacia los

[18] El programa DEMUX _ C .VHDL se encuentra en el apéndice H en la sección H.3

pines de salida del ASIP y su salida no sufre ninguna variación respecto al ciclo de instrucción anterior . En la figura 1.22 se muestra el símbolo del subsistema REG _ H.

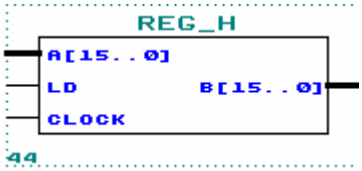


Fig.1.22 Símbolo del subsistema REG _H. SYM

Debo resaltar además que el subsistema REG _H[19] está gobernado por una señal de reloj cuya entrada es CLOCK la cual hace que se repita cíclicamente el programa principal .

1.9 Descripción del Decodificador de Interrupciones

El decodificador de interrupciones es un subsistema del ASIP llamado INTERRUPCIONES[20] , este subsistema nos permitirá realizar interrupciones al activar los pines IA ,IB, IC, ID del ASIP . Cuando se realiza una interrupción el ASIP solamente puede saltar hacia una dirección de memoria especificada por el diseñador del ASIP.

Si se activan a la vez uno o mas pines de interrupción[21] el ASIP ejecutará un solo salto correspondiente al pin de mayor prioridad. La salida del subsistema INTERRUPCIONES es una señal llamada SELECCIÓN cuya función es codificar los pines de prioridad y enviarlas al subsistema CONTADOR . En el subsistema CONTADOR he definido hacia que dirección va a saltar el subsistema ROM _R cuando se habiliten estos pines.

Cuando IA = 1 [22]sin importar el valor de los otros pines de interrupción , la dirección de la memoria será: 65518 que es un valor cualquiera que le he dado. Cuando IA = 0 y IB = 1[23] sin importar el valor de los otros pines de interrupción la dirección de la memoria será : 65516 . Cuando IA = 0 , IB = 0 y IC = 1[24] sin importar el valor de la otra patita de interrupción la dirección de la memoria será : 65514 y por ultimo . Cuando IA = 0 , IB = 0 y IC = 0 , ID = 1[25] , la dirección de la memoria será: 65512. El valor de la dirección del salto de los pines de

[19] El programa REG _ H .VHDL se encuentra en el apéndice H en la sección H.5
[20] El programa INTERRUPCIONES .VHDL se encuentra en el apéndice H en la sección H.16
[21] Los pines de interrupción son IA , IB , IC , ID y son llamados así porque estos pines al darles un pulso generan una interrupción
[22] IA es el pin de mayor prioridad o primera prioridad
[23] IB es el pin de segunda prioridad
[24] IC es el pin de tercera prioridad
[25] ID es el pin de cuarta prioridad o ultima prioridad

prioridad puede ser modificado , esto se realiza variando el programa del subsistema CONTADOR . En la figura 1.23 se muestra el símbolo del subsistema INTERRUPCIONES.

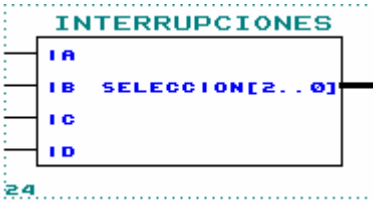


Fig.1.23 Símbolo del subsistema INTERRUPCIONES. SYM

1.10 Descripción del Contador de Direcciones

El subsistema encargado de enviar la dirección al subsistema ROM _R es llamado CONTADOR[26] , este subsistema proporciona la dirección del subsistema ROM _R .

La dirección del subsistema ROM _R durante su funcionamiento normal aumenta de uno en uno y su dato almacenado en dicha dirección genera una nueva instrucción , la cual esta dada por el programador del ASIP.

Cuando se realiza un salto o una interrupción, el subsistema CONTADOR deja de contar de uno en uno para automáticamente enviar al subsistema ROM _R una dirección . Si se realiza un salto por software o por comparación significa que la dirección a saltar esta especificada por el programador del ASIP, pero si se realiza una interrupción la dirección a saltar esta especificada por el diseñador del ASIP .

El subsistema CONTADOR se encarga de reconocer cuando se produce un salto por software o por comparación y si este salto es permitido. Si se produce un salto por software o comparación el subsistema CONTADOR le da al subsistema ROM _R una nueva dirección para que ejecute un nuevo grupo de instrucciones. Si el salto se produce por una interrupción el subsistema CONTADOR no pedirá autorización para saltar , el salto se realizará inmediatamente después de 2 ciclos de instrucción del ASIP. En la figura 1.24 se muestra el símbolo del subsistema CONTADOR.

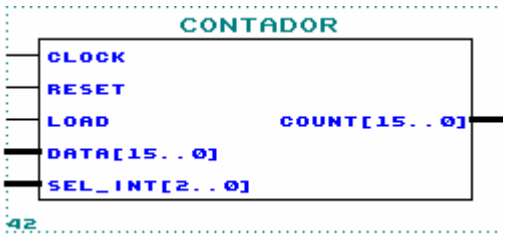


Fig.1.24 Símbolo del subsistema CONTADOR. SYM

[26] El programa CONTADOR .VHDL se encuentra en el apéndice H en la sección H.2

1.11 Descripción de los saltos e interrupciones del ASIP

El ASIP posee 2 tipos de saltos:

A)_ Salto condicional

B)_ Salto por software

Ambos saltos son habilitados o deshabilitados por el programador del ASIP a través de la entrada H del subsistema C _LOGICO.

Existe también un salto por activación de sus pines de interrupción , a estos tipos de saltos se le llamará interrupciones, debido a que estas interrupciones no son habilitados o deshabilitados por la señal de entrada H del subsistema C _LOGICO , las interrupciones poseen su propio subsistema llamado INTERRUPCIONES.

Para realizar los saltos por comparación y por software en el ASIP se necesita una lógica en la cual no existan dos saltos a la vez y además de ello el subsistema ROM _R pueda habilitar o deshabilitar dichos saltos cuando el programador del ASIP lo desee y que sea independiente de las interrupciones externas que también generan saltos.

En el caso de interrupciones externas que también generan saltos los subsistemas que están involucrados son INTERRUPCIONES y CONTADOR . En el caso de los saltos por software o comparación tenemos un subsistema especial que habilita o deshabilita dichos saltos llamado C_LOGICO.

El subsistema C _LOGICO[27] da al subsistema CONTADOR la orden de habilitar un salto solo cuando LOAD = 1 en caso contrario no se realiza un salto por software o comparación , para que ocurra un salto el subsistema ROM _R debe autorizarlo haciendo que la señal de entrada H del subsistema C _LOGICO se encuentre en 1 lógico , si esto no ocurre los saltos por software y por comparación están deshabilitados y no pueden ocurrir .

El salto condicional ocurre cuando la señal de salida DC del subsistema COMPARADOR es un 1 lógico , DC es una bandera o flag que se pone a 1 cuando el valor de la señal de salida del subsistema ALU es “0000000000000000” en caso contrario la señal DC estará en 0 , pero este salto se realizará solo si la señal H esta en 1 lógico , cabe resaltar que la señal DC no es manipulable por el programador del ASIP debido a que su valor depende de la señal de salida del subsistema ALU .

El salto por software ocurre cuando el subsistema ROM _R coloca la señal de entrada B _SOFT del subsistema C _LOGICO a 1 lógico , B _SOFT es una señal manipulable por el

[27] El programa C _LOGIC .VHDL se encuentra en el apéndice H en la sección H.1

programador del ASIP[28] , con esta señal el programador del ASIP realiza un salto siempre y cuando la señal H este en 1 lógico.

En el salto por software o en el salto condicional necesario definir hacia que dirección saltar , la dirección a la cual salta el ASIP se encuentra en la señal de salida DATA del subsistema ADAPTADOR . El subsistema ADAPTADOR envía dicha señal de dirección al subsistema DEMUX _C el cual puede hacer que dicha señal sea una dirección de salto o un dato el cual se almacena en cualquiera de los subsistemas REG _C .

Cabe resaltar de que para que se realice un salto condicional o un salto por software es necesario utilizar 2 ciclos de instrucción uno para ordenar el salto y otro para tomar la configuración del programa a realizar es decir colocar el flag o bandera de nuevo a cero y colocar los otros valores como H y B _SOFT a sus valores anteriores. En la figura 1.25 se muestra el símbolo del subsistema C _LOGICO .

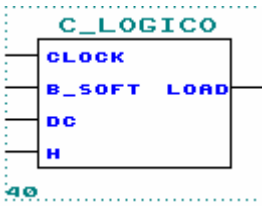


Fig.1.25 Símbolo del subsistema C _ LOGICO. SYM

[28] El bit B _SOFT es el que genera el salto por software.

CAPITULO II:
DISEÑO Y SIMULACIÓN DE LOS SUBSISTEMAS QUE CONFORMAN EL ALU

En este Capitulo se diseñará el ALU del microprocesador ASIP y se simulará sus operaciones aritméticas lógicas y de corrimiento de las entradas del ALU. Tomaremos en cuenta también los retardos de cada subsistema para así conocer el retardo de todo el ASIP.

2.1 Generalidades

En este capitulo realizo la simulación del funcionamiento de todas las operaciones aritméticas , lógicas y de desplazamiento que realiza el ASIP . Estas operaciones dependen del subsistema ALU (Unidad Aritmético Lógica). En el presente capitulo realizo la simulación de los subsistema que conforman el subsistema ALU.

El lenguaje de programación VHDL ofrece en su librería **use IEEE.std_logic_arith.all** todas las operaciones aritméticas directamente , es decir no es necesario realizar un algoritmo para realizar la suma y la resta , además de ello el ASIP posee un pin de entrada denominado CIN en donde se puede cuantificar un bit de acarreo de otro microprocesador o algún dispositivo o subsistema digital.

Para la parte lógica existen instrucciones especiales que tiene el lenguaje de programación VHDL , las operaciones lógicas que se pueden ejecutar con este lenguaje de programación son : AND , OR , NOT y XOR .

Para los desplazamientos hacia la derecha o hacia la izquierda se a realizado algoritmos que permitan al ASIP realizar los desplazamientos antes mencionados.

2.2 Subsistema Desplazamiento a la Derecha

El subsistema de desplazamiento hacia la derecha[1] es llamado subsistema ASHR , posee una entrada A y dos salidas D y CF . Este subsistema traslada los bits de la entrada B del subsistema ALU hacia la derecha. El bit A(15) de la entrada del subsistema ASHR aparece

[1] En el apéndice I se muestra un diseño de subsistema de corrimiento a base de multiplexores .

en la salida CF y la salida D ordena sus bits de la siguiente manera: D(0) = 0 , D(1) = A(0) , D(2) = A(1) y así sucesivamente hasta D(15) = A(14).

En el apéndice H sección H.13 se encuentra el programa del subsistema ASHR. En la figura 2.1 se observa el diagrama de flujo del subsistema ASHR

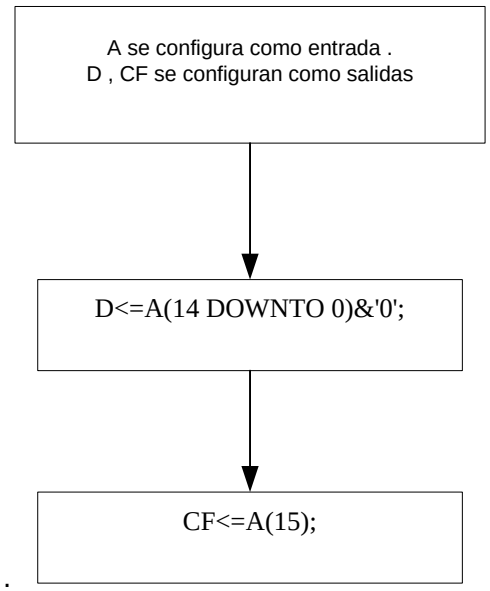


Fig.2.1: Diagrama de flujo del subsistema de Desplazamiento a la Derecha

El subsistema de desplazamiento a la derecha no depende de un CLOCK , es decir apenas las salidas sufran una variación , la salida D cambiará de acuerdo al valor de la entrada A , además se observa en la figura 2.2 que el subsistema ASHR tiene un retardo a la salida despreciable.

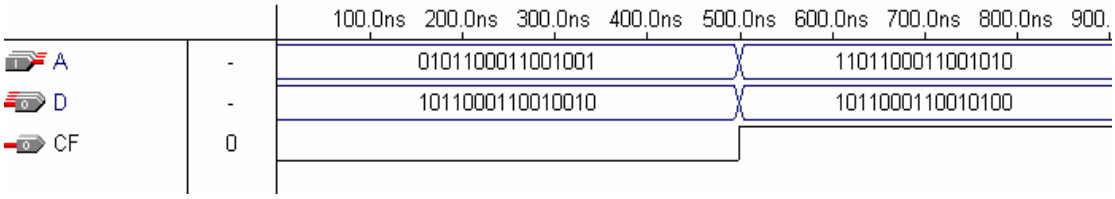


Fig.2.2: Simulación digital del subsistema de Desplazamiento a la Derecha

En la simulación de la figura 2.2 aparecen dos valores en la entrada . De 0ns a 500ns la entrada A = 0101100011001001 , la salida D = 1011000110010010 y CF = 0 . De 500ns a 1000ns la entrada A = 1101100011001010 , la salida D = 1011000110010100 y CF =1.

2.3 Subsistema Desplazamiento a la Izquierda

El subsistema de desplazamiento hacia la izquierda[2] es llamado subsistema ASHL , posee una entrada A y dos salidas D y CF . Este subsistema traslada los bits de la entrada B del subsistema ALU hacia la izquierda. El bit A(0) de la entrada del subsistema ASHL aparece en la salida CF y la salida D ordena los bits de la siguiente manera: D(16) = 0 , D(0) = A(1) , D(1) = A(2) y así sucesivamente hasta D(14) = A(15). En el apéndice H sección H.12 se encuentra el programa ASHL . En la figura 2.3 se observa el diagrama de flujo del subsistema ASHL

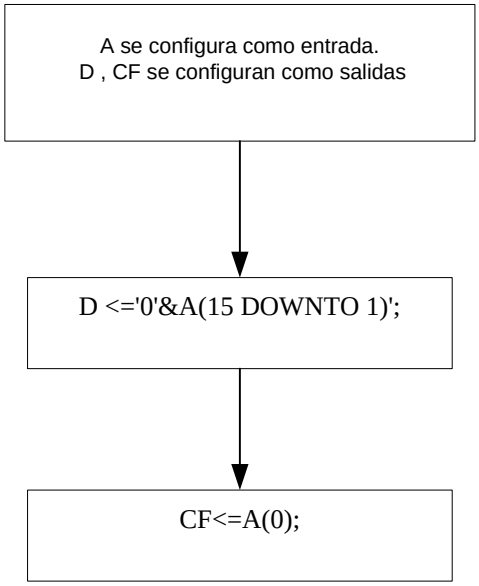


Fig.2.3: Diagrama de flujo del subsistema de Desplazamiento a la Izquierda

El subsistema de desplazamiento a la izquierda no depende de un CLOCK es decir apenas las salidas sufran una variación la salida D cambiará de acuerdo al valor de la entrada A , además se observa en la figura 2.4 que el subsistema ASHL tiene un retardo a la salida despreciable.

[2] En el apéndice I se muestra un diseño de subsistema de corrimiento a base de multiplexores .

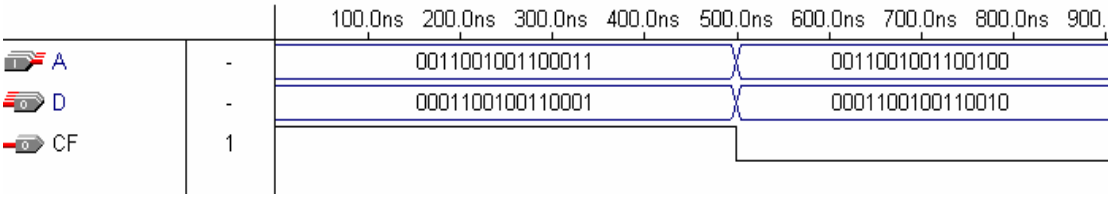


Fig.2.4: Simulación digital del subsistema de Desplazamiento a la Izquierda

En la simulación de la figura 2.4 aparecen dos valores en la entrada . De 0ns a 500ns la entrada A = 0011001001100011 , la salida D = 0001100100110001 y CF = 1 . De 500ns a 1000ns la entrada A = 0011001001100100 , la salida D = 0001100100110010 y CF = 0.

2.4 Subsistema Lógico

Las micro operaciones Lógicas listadas en la tabla 2.1 pueden implementarse en VHDL mediante forma algorítmica o mediante una estructura digital compuesta de compuertas y multiplexores.

TABLA N^0 2.1: Tabla de funciones del subsistema Lógico

S1	S0	D
0	0	A AND B
0	1	A OR B
1	0	A XOR B
1	1	NOT A

Las micro operaciones lógicas especifican operaciones binarias en los bits almacenados en los subsistemas REG _C o en las señales de entradas del ASIP. Estas operaciones consideran cada bit del subsistema REG _C en forma separada y los tratan como variables binarias . Para el ASIP de 16 bits diseñaremos un subsistema de nombre LOGIC el cual realizará todas las operaciones lógicas.

En el subsistema ALU usamos un solo subsistema LOGIC para que realice las operaciones lógicas entre las entradas A y B.

El subsistema LOGIC posee una entrada de reloj de nombre “CLOCK” además tiene 2 entradas A y B , cada una es de 16 bits, una entrada S de 2 bits que sirve para seleccionar la operación lógica a realizar y una salida D de 16 bits .

En el apéndice H sección H.11 se muestra el programa del subsistema LOGIC y en la figura 2.5 se observa el diagrama de flujo del subsistema LOGIC [3] . El algoritmo del subsistema LOGIC se repite cada vez que la entrada CLOCK de flanco positivo detecta una variación en las entradas A y B. En el diagrama de flujo observamos que la operación lógica a ejecutar depende de la entrada S . En las figuras 2.6 , 2.7 , 2.8 y 2.9 se realiza la simulación del subsistema LOGIC para los diferentes valores de la entrada de selección S , por cada valor de la entrada S se simulará una operación lógica del subsistema LOGIC.

[3] En el apéndice G se muestra el diseño de un subsistema lógico de 4 bits a base de compuertas y multiplexores que me a servido de base para el diseño del subsistema LOGIC .VHDL

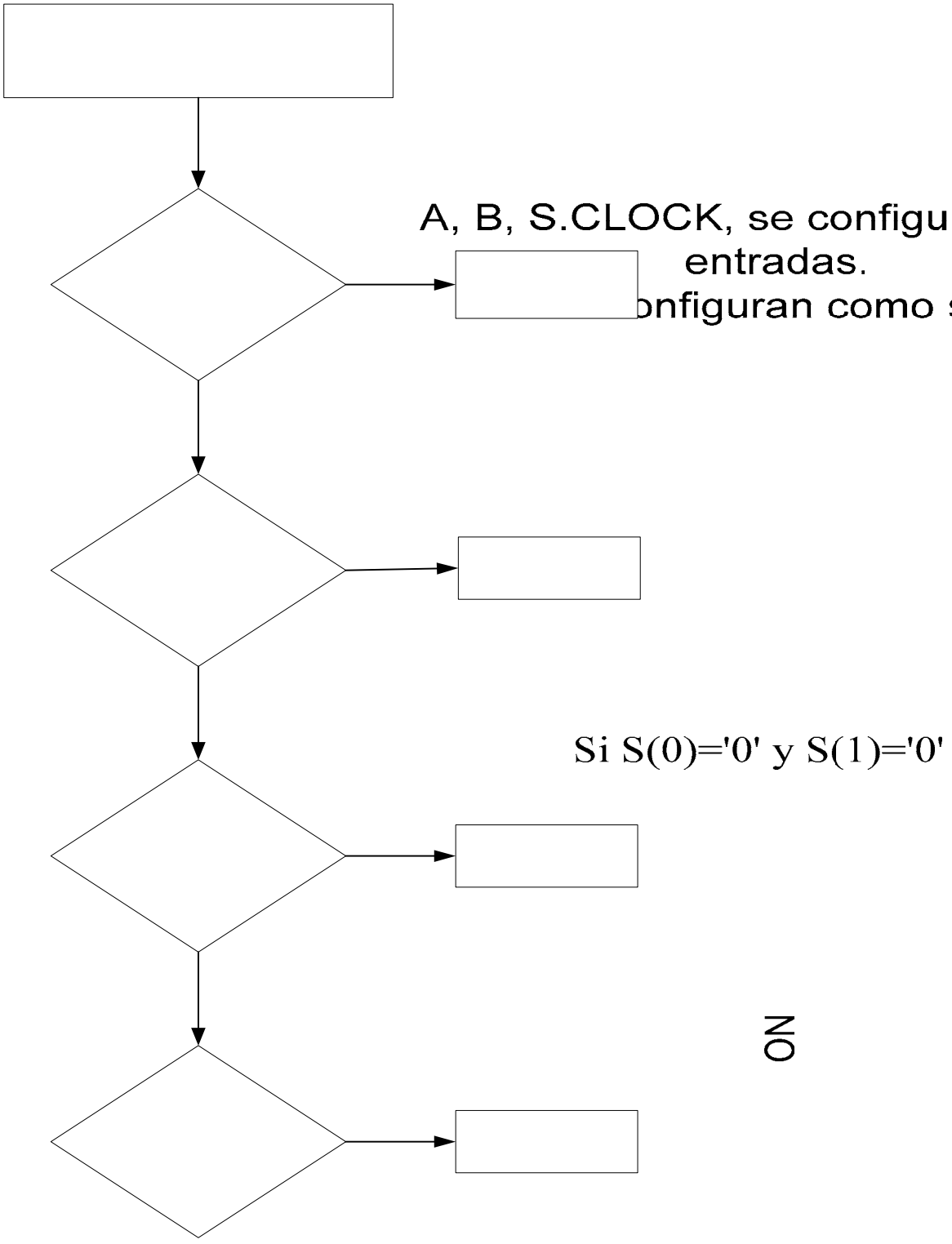


Fig.2.5: Diagrama de flujo del subsistema LOGIC.

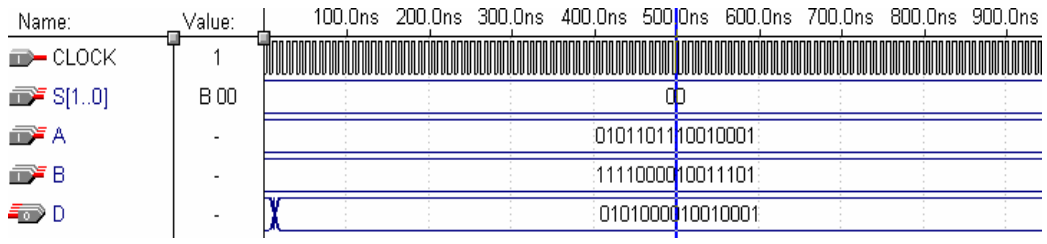


Fig.2.6 : Simulación digital del subsistema LOGIC cuando S = 00

En la figura 2.6 se muestra la simulación cuando la entrada S del subsistema LOGIC toma el valor lógico 00 y el valor de la entrada CLOCK es de 8ns entonces la salida D del subsistema LOGIC realiza la operación A AND B , En esta simulación el valor de A = 0101101110010001 y B = 111100001001101 dando como resultado la salida D = 0101000010010001, además se observa que la salida D a sufrido un retardo de 8.0ns.

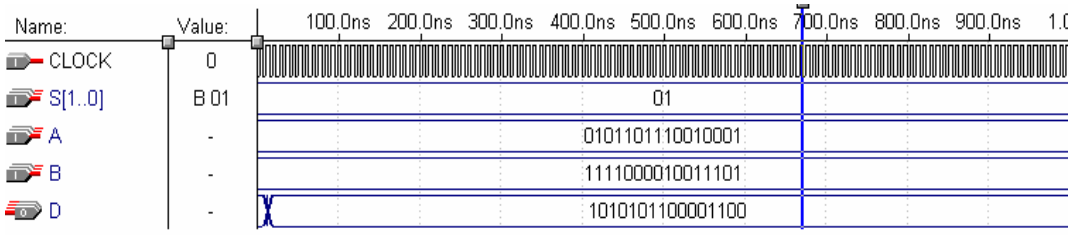


Fig.2.7: Simulación digital del subsistema LOGIC cuando S1 = 0 S0 = 1

En la figura 2.7 se muestra la simulación del subsistema LOGIC con la entrada s1 = 0 y s0 = 1 , entonces la salida D realiza la operación A XOR B . En esta simulación el valor de A = 0101101110010001 y B = 1111000010011101 dando como resultado la salida D = 1010101100001100 además se observa que la salida D a sufrido un retardo de 8ns .

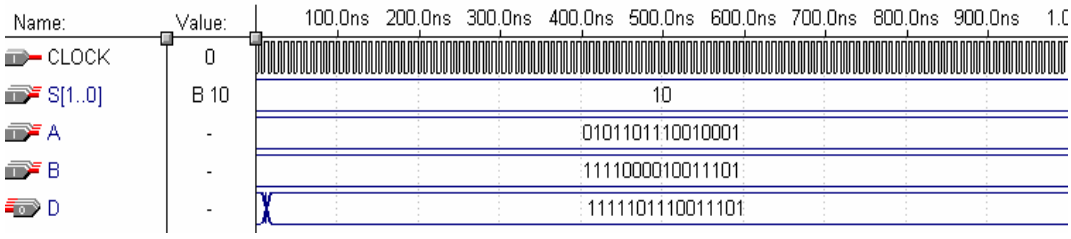


Fig.2.8: Simulación digital del subsistema LOGIC cuando S1 = 1 S0 = 0

En la figura 2.8 se muestra la simulación del subsistema LOGIC con la entrada s1 = 1 y s0 = 0 , además hacemos que el CLOCK trabaje a 8ns . La salida D realiza la operación A OR B . En esta simulación el valor de A = 0101101110010001 y B = 1111000010011101 dando

como resultado la salida Q = 1111101110011101 además en la simulación digital de la figura 2.8 se observa que la salida D a sufrido un retardo de 8ns .

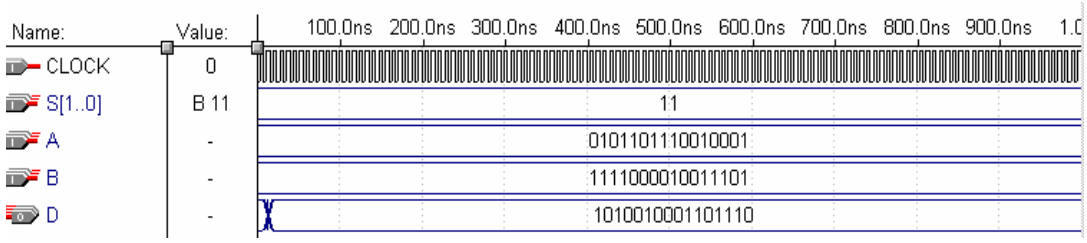


Fig.2.9: Simulación digital del subsistema LOGIC cuando S1 = 1 S0 = 1

En la figura 2.9 se muestra la simulación del subsistema LOGIC con la entrada s1 = 1 y s0 = 1 , además hacemos que el CLOCK trabaje a 8ns . La salida D realiza la operación NOT A . En esta simulación el valor de A = 0101101110010001 y B = 1111000010011101 dando como resultado la salida Q = 1010010001101110 . En esta operación lógica la salida Q no depende de la entrada B.

2.5 Subsistema Aritmético

Las micro operaciones ARITMETICAS listadas en la tabla 2.1 pueden implementarse en VHDL mediante forma algorítmica o mediante una estructura digital .

TABLA N^{0}2.2: Tabla de funciones del subsistema Aritmético

SEL1	SEL0	Q
0	0	A +CIN
0	1	A + NOT B +CIN
1	0	A +B + CIN
1	1	A – CIN

Las micro operaciones aritméticas especifican operaciones binarias para bits almacenados en los subsistemas REG _C o en las señales de entradas A y B del ASIP. El subsistema ARITHMETIC[4] esta gobernado por un reloj llamado CLOCK , la señal CLOCK es de flanco positivo. Cuando el CLOCK del subsistema ARITHMETIC detecta un cambio en alguna de sus entradas , el subsistema ARITHMRTIC ejecuta el programa principal , a este tipo de

[4] En el apéndice F se muestra un diseño de un circuito aritmético de 4 bits mediante compuertas el cual e tomado de base para diseñar el subsistema ARITHMETIC el cual a sido programado en vhdl

subsistemas se le conoce como asíncrono , esto se debe a que la arquitectura del subsistema ARITHMETIC esta gobernada por un CLOCK de flanco positivo .

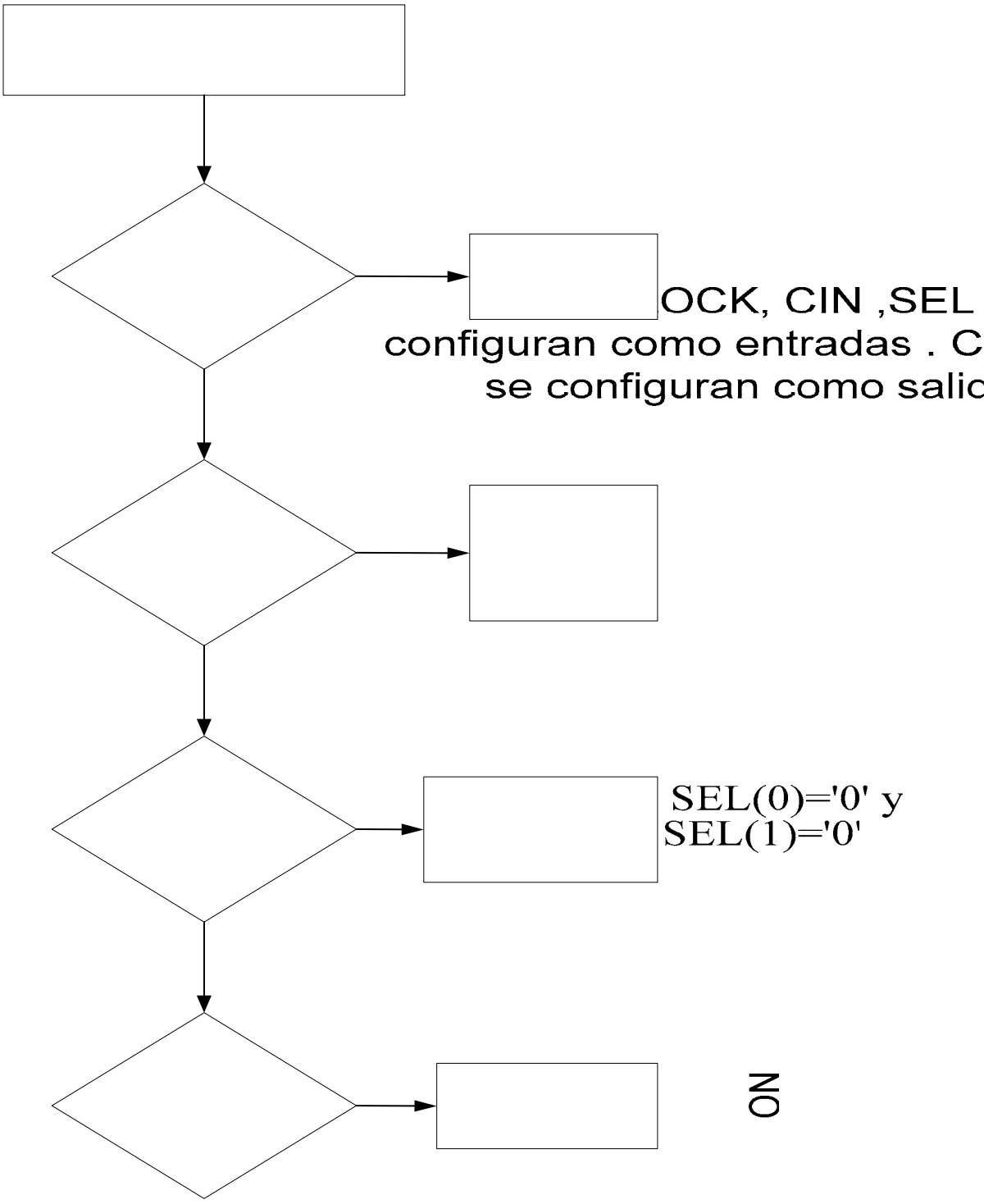


Fig.2.10: Diagrama de flujo del subsistema ARITHMETIC

Si SEL(0)='1' y
SEL(1)='0'

En el apéndice H sección H.10 se observa el programa del subsistema ARITHMETIC y en la figura 2.10 se observa el diagrama de flujo del subsistema ARITHMETIC, la entrada SEL es un selector que nos dirá que operación aritmética realiza la salida Q , CIN es otra entrada que puede ser el acarreo de un dispositivo , si no se usa la entrada CIN se recomienda colocarla a cero lógico. La salida COUT es el acarreo de las operaciones aritméticas realizadas.

En las figuras 2.11 , 2.12 , 2.13 , 2.14 , 2.15 , 2.16 , 2.17 mostramos las simulaciones de cada operación aritmética del subsistema ARITHMETIC , en estas simulaciones se a variado la entrada de selección SEL y CIN , los cuales nos proporcionan las diferentes operaciones aritméticas que realiza el subsistema ARITHMETIC .

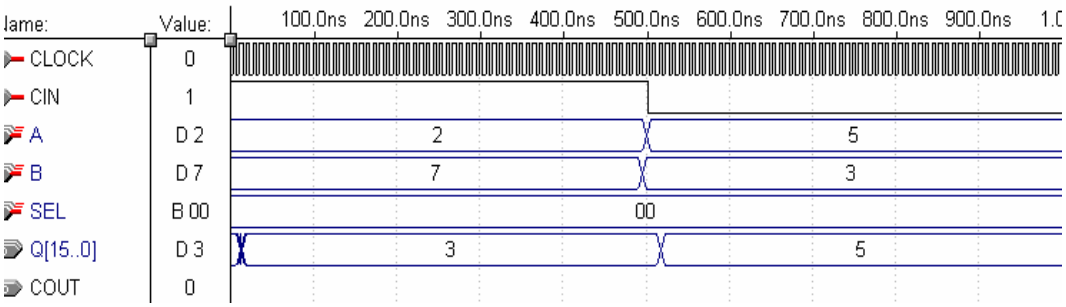


Fig.2.11: Simulación digital del subsistema ARITHMETIC cuando SEL(1) = 0 y SEL(0) =0.

En la figura 2.11 observamos que SEL(1) = 0 , SEL(0) = 0 y CIN tiene el valor de 1 lógico en el tiempo de 0 hasta 500ns , sin importar la entrada B la salida Q toma el valor de A +1 a esta operación se le llama INCREMENTO DE A , en la simulación se observa que cuando CIN =1 y A = 2 la salida Q = 3 . En el tiempo de 500ns a 1000ns se observa que CIN = 0 en ese caso A = Q a esta operación se le llama TRANSFERENCIA . Además en la simulación digital de la figura 2.9 se observa que la salida Q a sufrido un retardo de 16ns .

En la figura 2.12 se observa de que cuando SEL(1) = 0 , SEL(0) = 0 y CIN = 0 se esta realizando una diferencia entre A y B pero en la salida se le a aumentado una unidad debido a que el valor de CIN es 0 lógico. En esta figura se observa también que cuando se produce un resultado negativo COUT toma el valor de 1 lógico , esto se observa en el intervalo de 0 hasta 500ns pero si la diferencia es positiva el valor de COUT es 0 lógico , esto sucede en el intervalo de 500ns a 1000ns . En la figura 2.13 se realiza una simulación de la diferencia o resta que nosotros conocemos , esta operación se denomina resta con préstamo y a la operación simulada en la figura 2.12 se denomina resta.

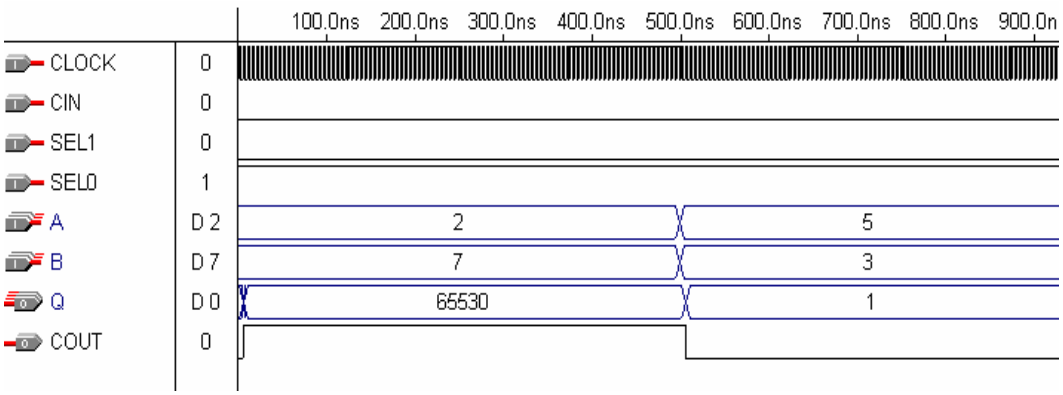


Fig.2.12: Simulación digital del subsistema ARITHMETIC cuando SEL(1) = 0 Y SEL(0) =1 además CIN = 0

En la simulación digital de la figura 2.12 se observa que la salida Q a sufrido un retardo de 16ns .

En la figura 2.12 observamos que de 500ns a 1000ns el subsistema ARITHMETIC realiza una resta, es decir la entrada A menos la entrada B , como la resta es un numero positivo la salida COUT es 0 lógico, solo si la diferencia entre A y B es negativa como en el caso del tiempo de 0ns a 500ns deberíamos observar un numero negativo (-5) pero esto no sucede porque cuando ocurre una diferencia negativa el subsistema ARITHMETIC realiza en la salida Q el complemento a 2 del valor absoluto de la diferencia de A y B y además COUT esta en 1 lógico

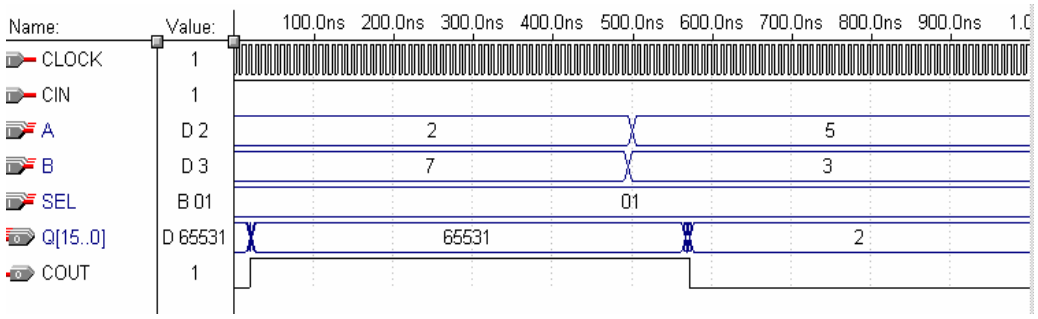


Fig.2.13: Simulación digital del subsistema ARITHMETIC cuando SEL(1) = 0 Y SEL(0) =1 además CIN = 1

En la simulación digital de la figura 2.13 se observa que la salida Q a sufrido un retardo que al medirlo en el simulador fue de 16ns .

La operación de la figura 2.14 es la suma de A con B y CIN pero el valor de CIN es 0 lógico , esta operación se realiza cuando el selector SEL esta en ‘10’ en binario.

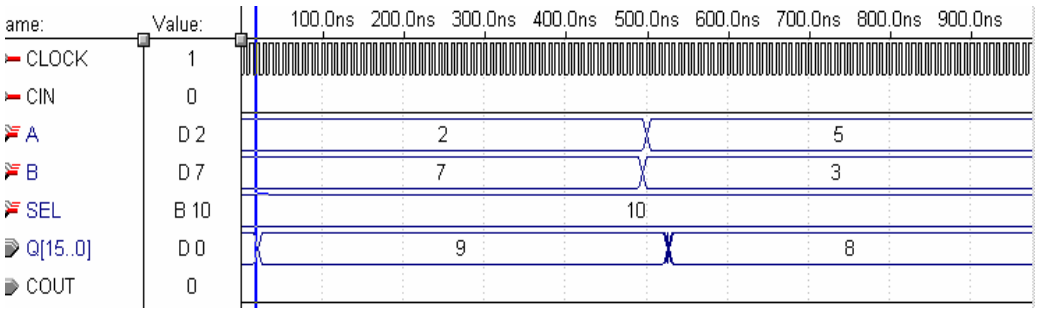


Fig.2.14: Simulación digital del subsistema ARITHMETIC cuando SEL(1) = 1 Y SEL(0) =0 además CIN = 0

En la simulación digital de la figura 2.14 se observa que la salida Q a sufrido un retardo de 16ns . Además en la simulación digital de la figura 2.13 el valor de la salida Q esta en el sistema decimal.

La operación de la figura 2.14 es la suma de A , B y CIN , pero el valor de CIN es 1 lógico , esto representa el acarreo de un subsistema externo, y se realiza cuando el selector SEL esta en '10' en binario. Además en la simulación digital de la figura 2.13 el valor de la salida Q esta en el sistema decimal.

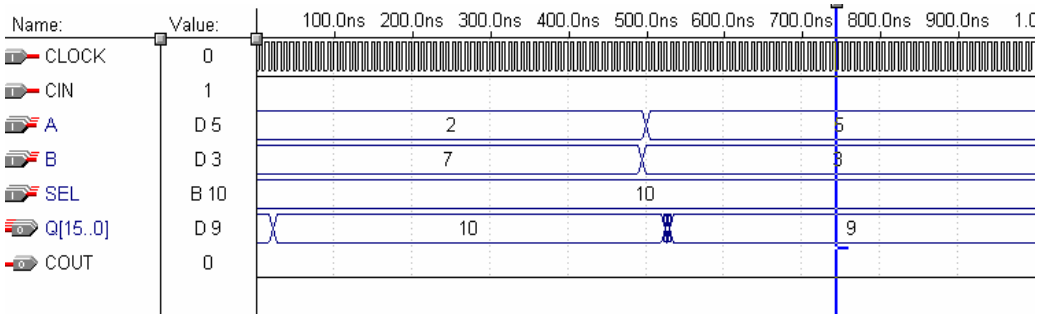


Fig.2.15: Simulación digital del subsistema ARITHMETIC cuando SEL(1) = 1 Y SEL(0) =0 ADEMAS CIN = 1

Además en la simulación digital de la figura 2.15 se observa que la salida Q a sufrido un retardo de 16ns .

La figura 2.16 muestra la operación decremento de A , es decir la salida Q es igual la entrada A disminuida en 1.

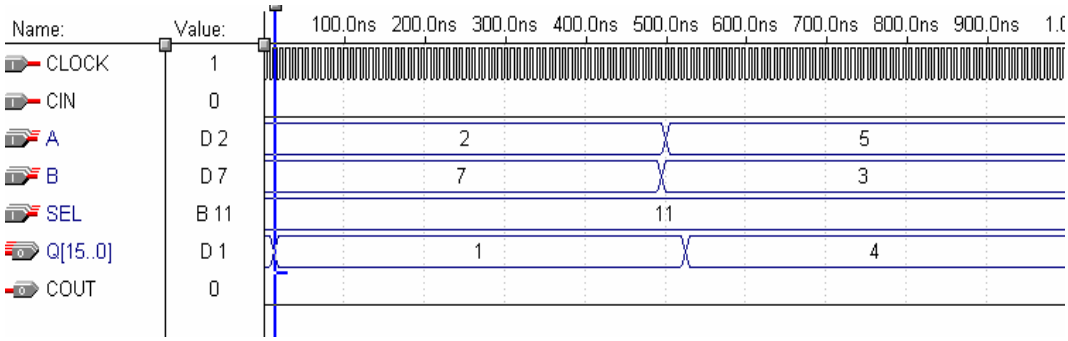


Fig.2.16: Simulación digital del subsistema ARITHMETIC cuando SEL(1) = 1 Y SEL(0) =1 además CIN = 0

En la simulación digital de la figura 2.16 se observa que la salida Q a sufrido un retardo de 16ns . Además en la simulación digital de la figura 2.16 el valor de la salida Q esta en el sistema decimal.

La figura 2.17 muestra la operación TRANSFERENCIA de A en la cual la salida Q es idéntica a la entrada A

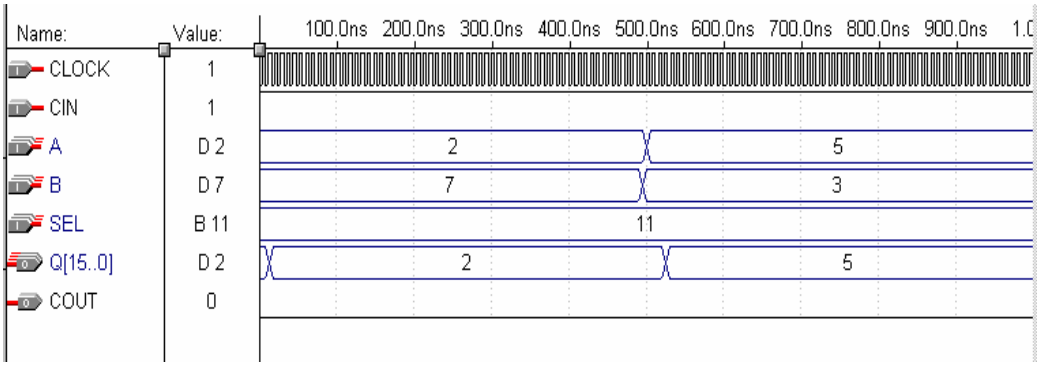


Fig.2.17: Simulación digital del subsistema ARITHMETIC cuando SEL(1) = 1 Y SEL(0) =0 además CIN = 1

En la simulación digital de la figura 2.18 se observa que la salida Q a sufrido un retardo de 16ns , además en la simulación digital de la figura 2.18 el valor de la salida Q esta en el sistema decimal.

2.6 Diseño del ALU

Habiendo simulado los subsistemas ARITHMETIC , LOGIC, ASHL, ASHR es necesario simular el subsistema MUX _A . En la figura 2.18 se muestra el diagrama de flujo del subsistema MUX _A .

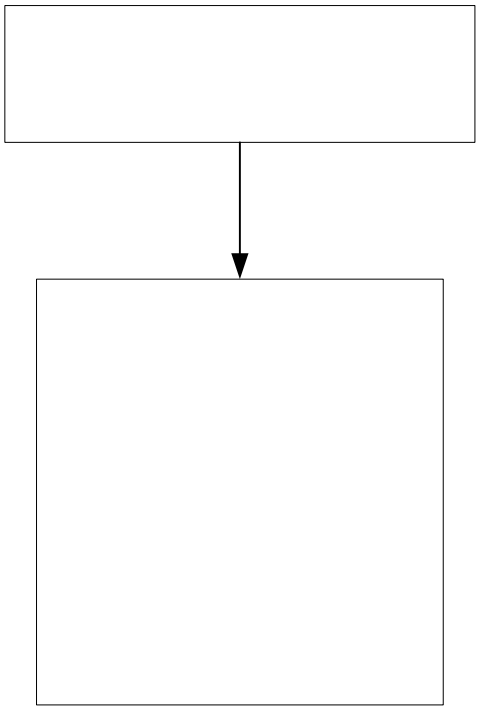


Fig.2.18 Diagrama de flujo del subsistema MUX _ A

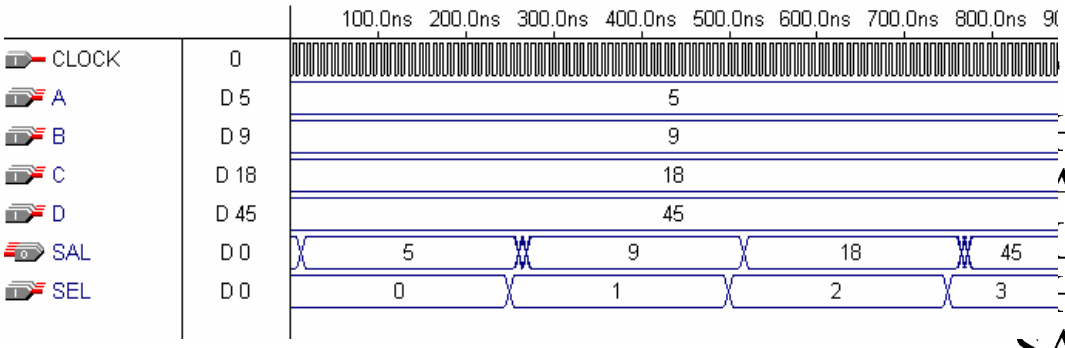


Fig.2.19: Simulación digital del subsistema MUX _A

En la simulación de la figura 2.19 se observa que usando un CLOCK de 8ns en el subsistema MUX _A tenemos un retardo en la salida SAL del subsistema MUX _A de 8ns .

La Arquitectura completa del subsistema ALU se muestra en la figura 2.20 en el cual se observa de que se a cortocircuitado la entrada “SEL” del subsistema ARITHMETIC con la entrada “S” del subsistema LOGIC y se a renombrado esta entrada como “SELECT” esta entrada va a definir una operación aritmética y una operación lógica pero ninguna de estas dos operaciones necesariamente van a aparecer en la salida del subsistema ALU porque el subsistema MUX_ A es el que va escoger cual será la salida del subsistema ALU.

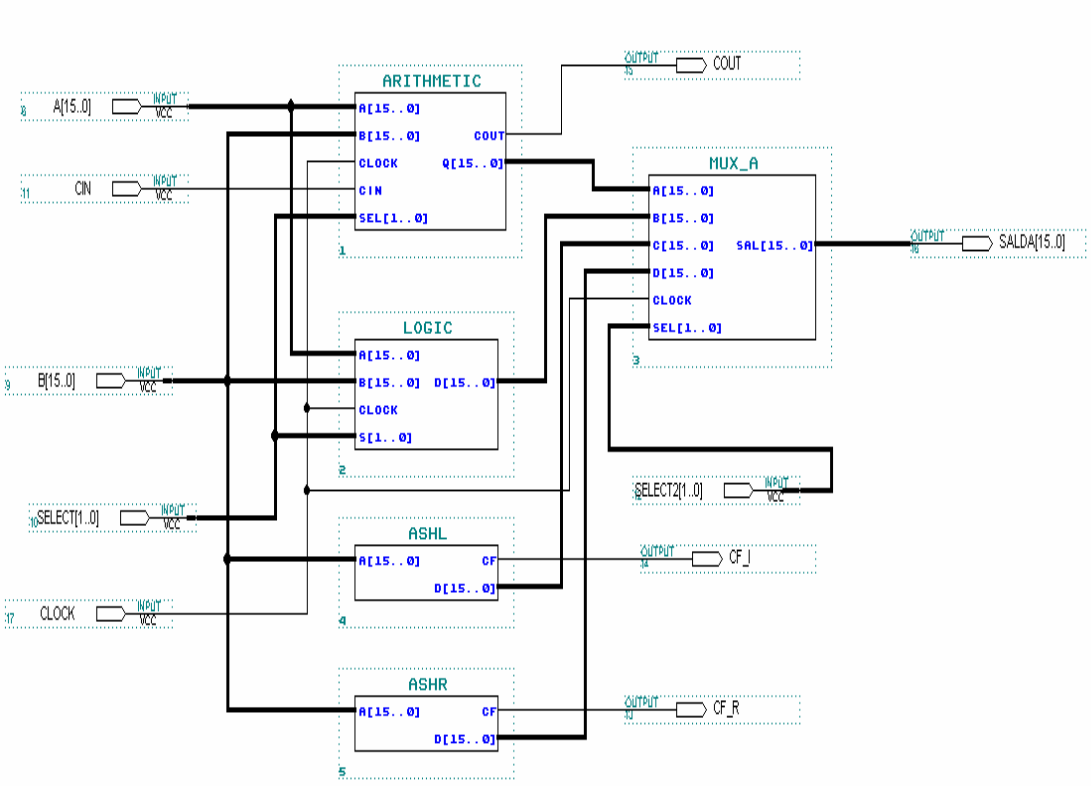


Fig. 2.20.: Arquitectura del subsistema ALU .GDF

Solo en la entrada B se va a realizar las operaciones de desplazamiento a la izquierda y a la derecha debido a que la entrada A no tiene conexión entre los subsistemas ASHR y ASHL . La entrada “SELECT2” es la que va a definir cual de las salidas de los subsistemas antes mencionados enviarán las señales de salida hacia del subsistema ALU, además todas las entradas CLOCK de los subsistemas se han cortocircuitado en una sola entrada de CLOCK.

Considerando un CLOCK de 8 ns y un dispositivo un FLEX de la familia 10K que es el dispositivo que pretendemos usar para grabar el ASIP , la salida del subsistema ALU tiene un retardo de 24ns cuando se realiza alguna operación aritmética , si se realiza una operación lógica el retardo del subsistema ALU es 16ns y si se realiza una operación de desplazamiento el retardo del subsistema ALU es 8ns, considerando el caso mas extremo , el retardo del subsistema ALU es de 24ns.

CAPITULO III:
DISEÑO Y SIMULACION DE LOS SUBSISTEMAS DE LA UNIDAD DE CONTROL DEL ASIP

En este capitulo se realizará el diseño y la simulación de todos los subsistemas que forman la Unidad de Control, tomaremos en cuenta también los retardos de cada subsistema para así conocer el retardo de todo el ASIP.

3.1 Generalidades

En la sección 1.8 del capitulo 1 encontramos la descripción de los componentes de la Unidad de Control , dichos componentes no están unidos electrónicamente y no lo he agrupado porque seria muy difícil realizar las uniones electrónicas con los otros subsistemas del ASIP , a continuación realizaremos la simulación de los subsistemas que conforman la unidad de control del ASIP.

3.2 Subsistema Adaptador

El subsistema ADAPTADOR tiene la función de dividir la señal que nos envía el subsistema ROM _R hacia los demás subsistemas del ASIP.

En el apéndice H sección H.4 se encuentra el programa del subsistema ADAPTADOR y en la figura 3.1 se muestra el diagrama de flujo del subsistema ADAPTADOR , en este diagrama de flujo se define con detalle hacia que señales internas del ASIP se divide la señal de entrada A del subsistema ADAPTADOR .

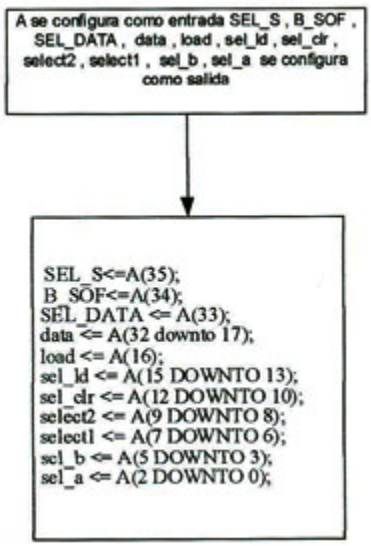


Fig.3.1:Diagrama de flujo del subsistema ADAPTADOR. VHDL

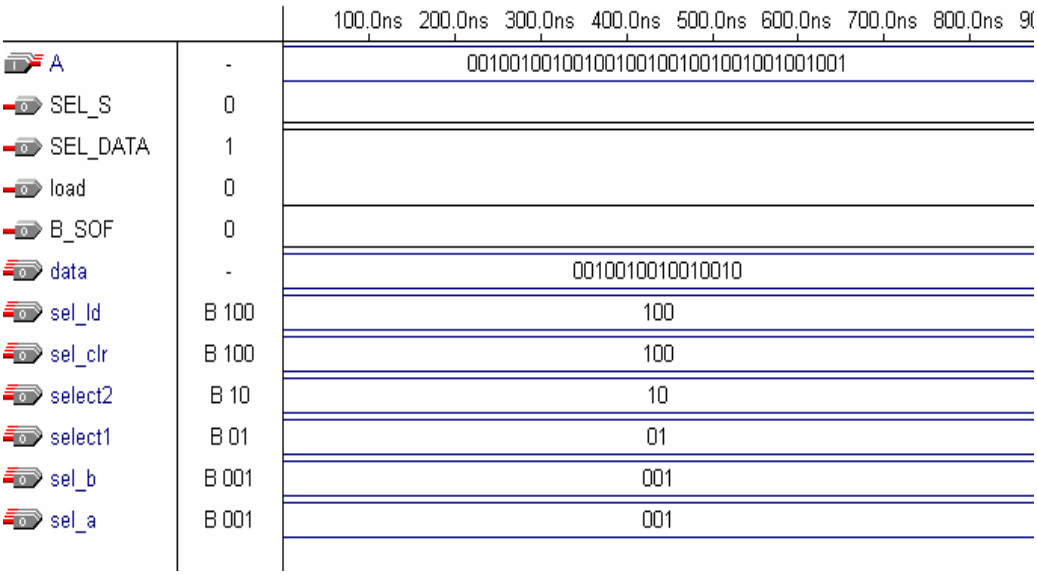


Fig.3.2: Simulación digital del subsistema ADAPTADOR

Además se observa en la simulación de la figura 3.2 que las salidas del subsistema ADAPTADOR tienen un retardo despreciable.

3.3 Subsistema DEMUX _C

En el apéndice H sección H.3 se encuentra el programa del subsistema DEMUX _C y en la figura 3.3 se muestra el diagrama de flujo del subsistema DEMUX _C .

Fig.3.3:Diagrama de flujo del subsistema DEMUX _ C. VHDL

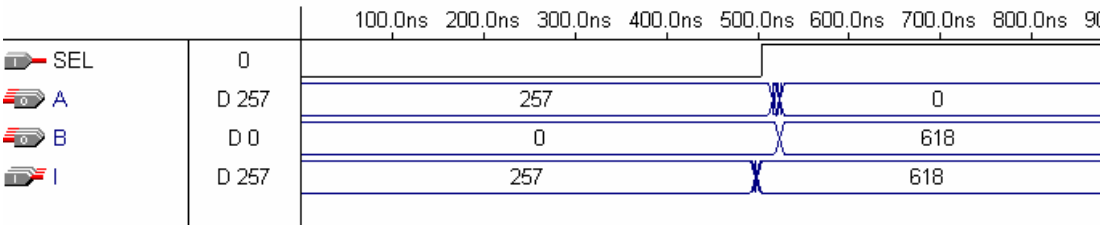


Fig.3.4: Simulación digital del subsistema DEMUX _C

Como se observa en la simulación digital de la figura 3.4 cuando la entrada SEL es 0 lógico , el subsistema DEMUX _C envía a su salida A la señal que se encuentra en su entrada I y en la salida B lo coloca a “00000000000000” . Si el valor de la entrada SEL es 1 lógico el subsistema DEMUX _C envía a su salida B la señal que se encuentra en su entrada I y en la salida A lo coloca a 0000000000000000. Al simular el subsistema DEMUX _C con una frecuencia de reloj de 8 ns en un dispositivo FLEX de 10K se observa que las salidas poseen un retardo despreciable.

3.4 Subsistema REG_H

En el apéndice H sección H.5 se encuentra el programa del subsistema REG_H y en la figura 3.5 se muestra el diagrama de flujo del subsistema REG _H . El subsistema REG _H se encuentra simulado en la figura 3.6 , y se observa que cuando la entrada LD del subsistema REG _H es 1 lógico , el subsistema REG _H carga el valor de su entrada A hacia su salida B , pero cuando LD es 0 lógico el subsistema REG _H no realiza ningún cambio en

se encuentra simulado en la figura 3.6 , y se observa que cuando la entrada LD del subsistema REG _H es 1 lógico , el subsistema REG _H carga el valor de su entrada A hacia su salida B , pero cuando LD es 0 lógico el subsistema REG _H no realiza ningún cambio en su salida B y mantiene su valor como estaba antes de estar LD en 0 lógico. El retardo a la salida de este subsistema es 8ns.

Fig.3.5: Diagrama de flujo del subsistema REG _H. VHDL

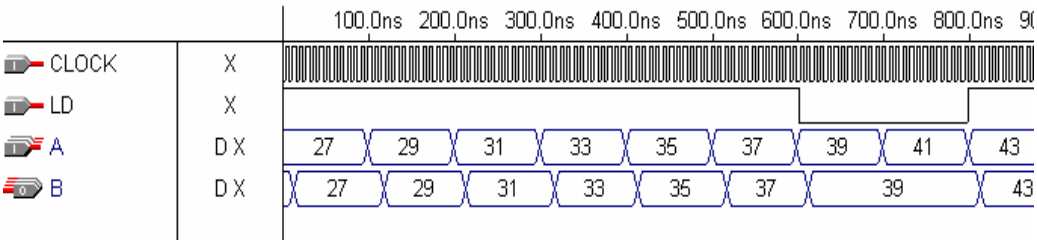


Fig.3.6: Simulación digital del subsistema REG _H

3.5 Subsistema MUX _C

En el apéndice H sección H.16 se encuentra el programa del subsistema MUX _C y en la figura 3.7 se muestra el diagrama de flujo del subsistema MUX _C . El subsistema MUX _C se ha simulado en la figura 3.8 y se observa que cuando la entrada SEL es 1 lógico , el subsistema MUX _C carga el valor de su entrada B a su salida SAL , pero cuando SEL es 0 lógico , el valor del la señal de salida SAL es la señal de la entrada A . El retardo a la salida del subsistema MUX _C es 8ns.

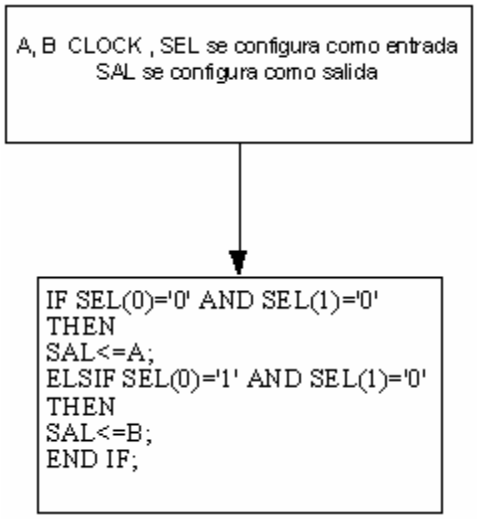


Fig.3.7: Diagrama de flujo del subsistema MUX _C. VHDL

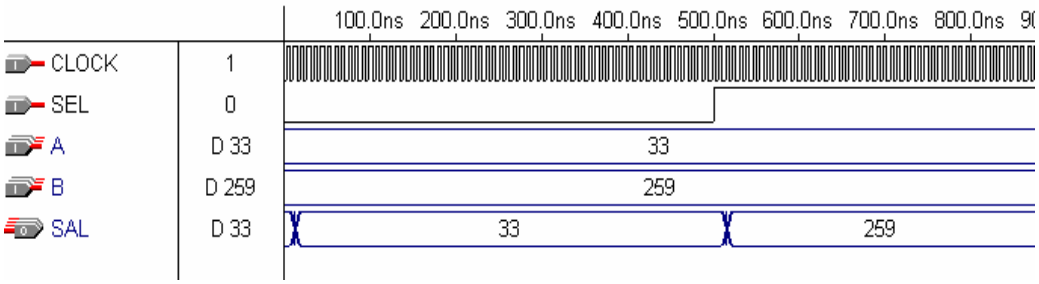


Fig.3.8: Simulación digital del subsistema MUX _C

CAPITULO IV:
DISEÑO Y SIMULACIÓN DE LOS SUBSISTEMAS QUE FORMAN LA UNIDAD DE PROCESAMIENTO Y EL ALMACENAMIENTO DE DATOS EN LOS REGISTROS (SUBSISTEMAS REG _C)

En este capitulo se simulará cada parte de la Unidad de Procesamiento y se explicará como esta constituido el bus de datos del ASIP. Tomaremos en cuenta también los retardos de cada subsistema para así conocer el retardo de todo el ASIP.

4.1 Generalidades

El Subsistema Unidad de Procesamiento nos ayudará a comprender como ingresan los datos al subsistema ALU, cuando los datos de los subsistemas REG _C ingresan a las entradas del subsistema ALU se realizará operaciones las cuales están explicadas y simuladas en el capitulo 2 .

4.2 Subsistema DECODER

En la figura 4.1 se muestra el diagrama de flujo del subsistema DECODER , el programa del subsistema DECODER se encuentra en el apéndice H sección H.6. La simulación del subsistema DECODER esta dado en la figura 4.2. La salidas del subsistema DECODER poseen un retardo despreciable.

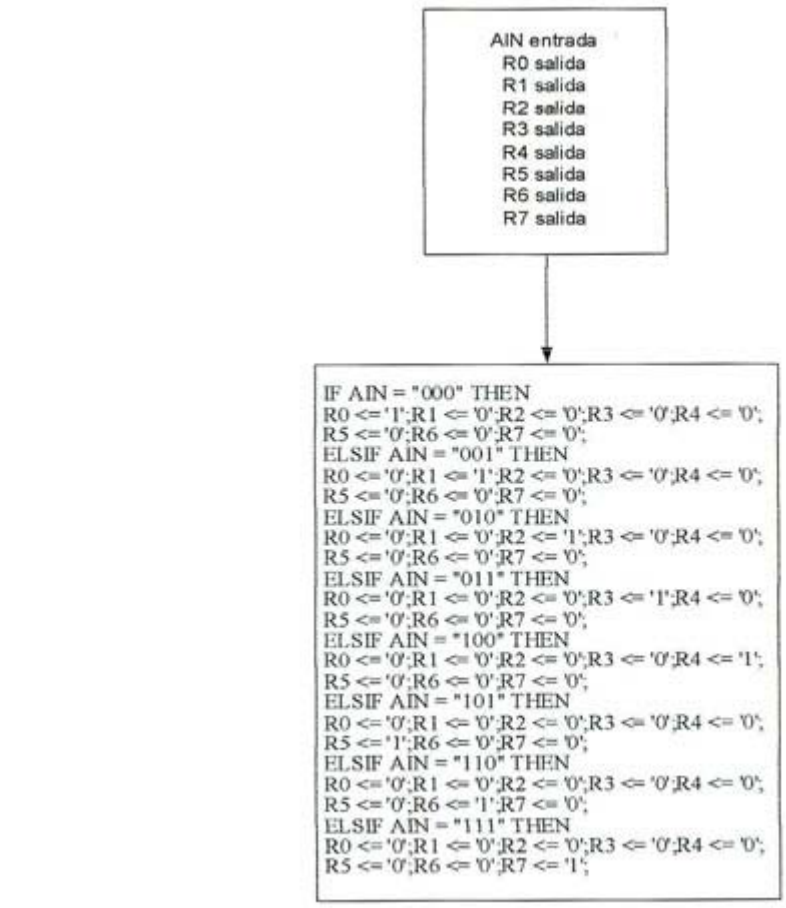


Fig.4.1: Diagrama de flujo del subsistema DECODER. VHDL

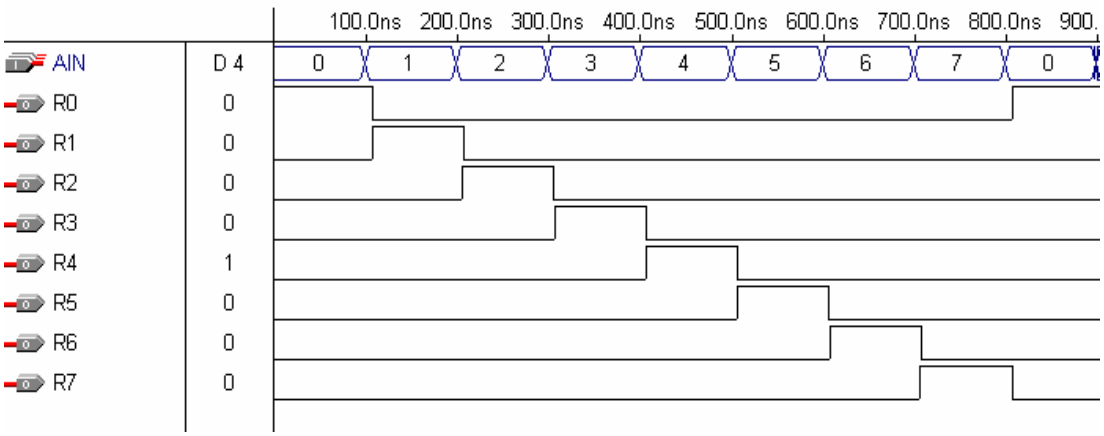


Fig.4.2: Simulación digital del subsistema DECODER

4.3 Subsistema REG _C

En la figura 4.3 se observa el diagrama de flujo del subsistema REG _C , el programa del subsistema REG _C se encuentra en el apéndice H sección H. 7 . La simulación del subsistema REG _C esta dado en la figura 4.4 a una frecuencia de reloj de 8ns y dentro de un dispositivo de altera FLEX de 10K . La salida del subsistema REG _C poseen un retardo de 8ns. Además la entrada A y la salida B están simulados en el sistema decimal.

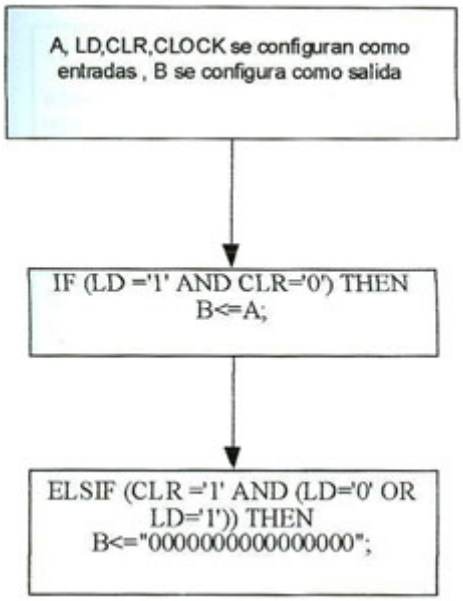


Fig.4.3: Diagrama de flujo del subsistema REG _ C. VHDL

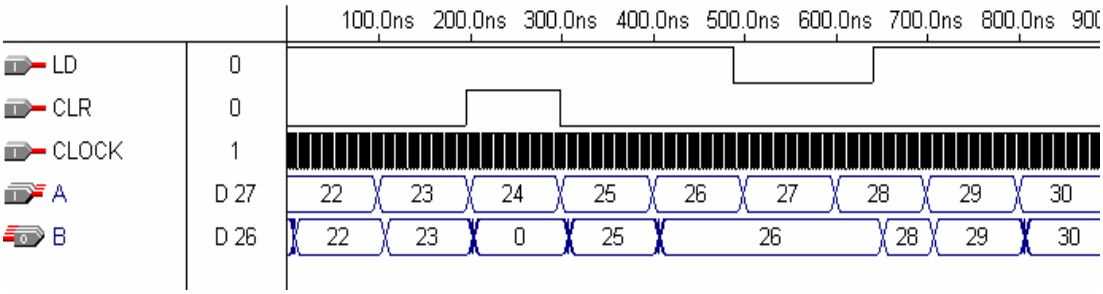


Fig.4.4: Simulación digital del subsistema REG _C

4.4 Subsistema MUX _B

En la figura 4.5 se observa el diagrama de flujo del subsistema MUX _B , el programa del subsistema MUX _B se encuentra en el apéndice H sección H. 8 . La simulación del subsistema MUX _B esta dado en la figura 4.6 a una frecuencia de reloj de 8ns y dentro de un dispositivo de Altera FLEX de 10K . La salida del subsistema MUX _B poseen un

retardo de 8ns , además todas las entradas y salidas se encuentran simulados en el sistema decimal.

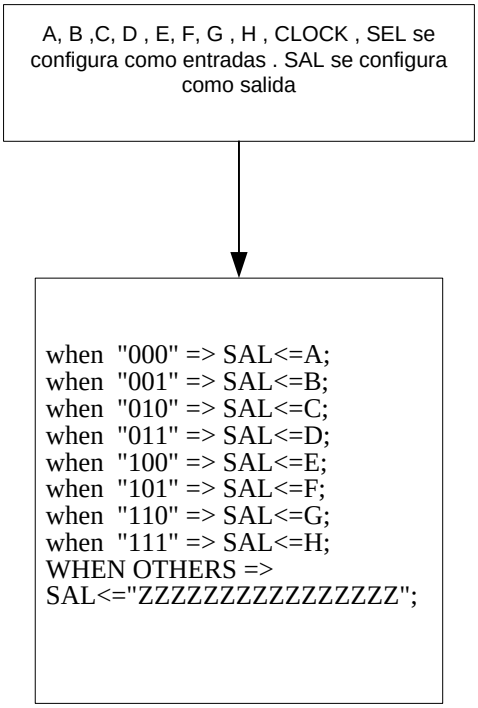


Fig.4.5: Diagrama de flujo del subsistema MUX _ B. VHDL

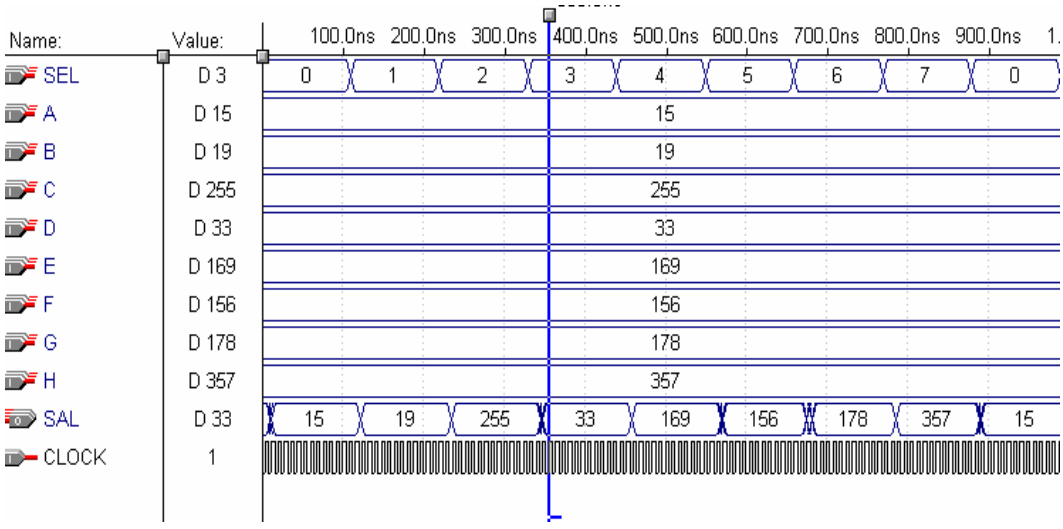


Fig.4.6: Simulación digital del subsistema MUX _B

4.5 Subsistema COMPARADOR

En la figura 4.7 se observa el diagrama de flujo del subsistema COMPARADOR , el programa del subsistema COMPARADOR se encuentra en el apéndice H sección H. 9 . La simulación de este subsistema esta dado en la figura 4.8 a una frecuencia de reloj de 8ns y en un dispositivo de altera de la familia de 10K . La salida del subsistema

COMPARADOR poseen un retardo despreciable , además la entrada A se encuentra simulado en el sistema decimal.

Fig.4.7: Diagrama de flujo del subsistema COMPARADOR. VHDL

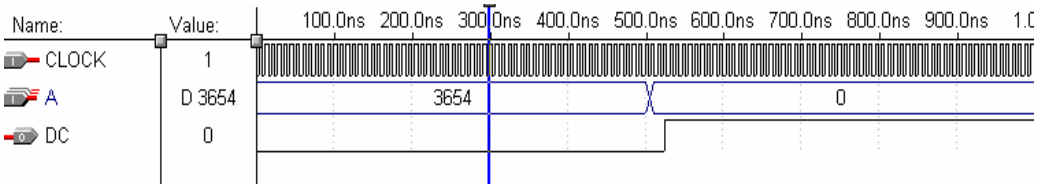


Fig.4.8: Simulación digital del subsistema COMPARADOR

4.5 Diseño de la Unidad de Procesamiento , Decoder de instrucciones y el Bus de Datos

En la sección 4.2 se simula el subsistema DECODER el cual realizará la función de un decoder de instrucción dentro de la Unidad de Procesamiento. En el subsistema Unidad de Procesamiento existen dos subsistemas DECODER , uno de estos subsistemas DECODER es el encargado de hacer que algún o ningún subsistemas REG _C esté en un estado de CLEAR y el otro subsistema DECODER es el encargado de que algún o ningún subsistema REG _C almacene la información de su entrada .

En el subsistema Unidad de Procesamiento los subsistemas REG _C realizan la función de los registros , el subsistema REG _C ha sido analizado en la sección 4.3.

En la figura 4.9 se observa la arquitectura final del subsistema Unidad de Procesamiento del ASIP. En esta sección hemos unido todos los subsistemas mencionados para formar el subsistema Unidad de Procesamiento.

En la figura 4.9 se muestra como se han unido los siete subsistemas REG_C con los dos subsistemas MUX _B y con el subsistema MUX _C , a esta unión electrónica de subsistemas se le denominan bus de datos debido a que por esa unión electrónica se

envían los datos que se almacenan en el subsistemas REG _C (registros) hacia el subsistema ALU . La salida del subsistema ALU o la salida B del subsistema DEMUX _C serán escogidos mediante el subsistema MUX _C , el cual enviará cualquiera de estos dos datos hacia el bus de datos . El subsistema MUX _C envía los datos hacia las entradas de los subsistemas REG _C y el subsistema DECODER se encarga de definir si este dato va a ser almacenado por los subsistemas REG _C y cual de estos subsistemas almacenará dicha información.

Existen muchos tipos de buses de datos , pero el bus de datos de nuestro ASIP es de un solo sentido y se comunica directamente con los subsistemas REG _C , MUX _B , MUX _C , ALU y ROM _R .

Hay que tener en cuenta que como el subsistema ALU posee un retardo a la salida de 24ns , el retardo del COMPARADOR y del subsistema DECODER es despreciable , el retardo de los subsistemas MUX _ B y REG _C son de 8ns cada uno , por lo tanto el subsistema Unidad de Procesamiento posee retardo de 40ns en el caso mas extremo , a una frecuencia de 8ns.

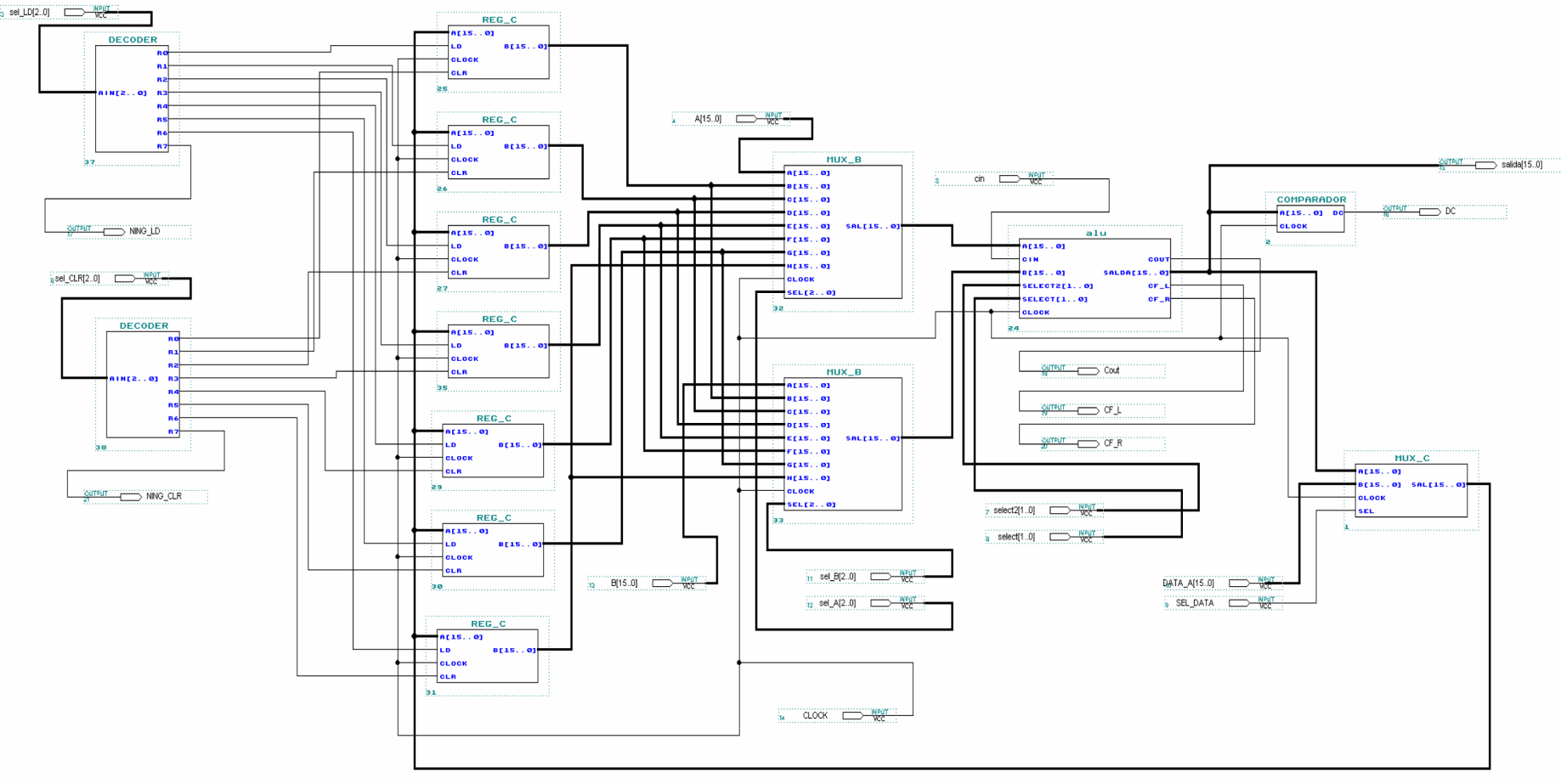


Fig 4.9: Arquitectura del subsistema Unidad de Procesamiento

4.7 Estructura de la memoria del ASIP en VHDL

El subsistema ROM _R es la memoria del ASIP , en este subsistema se encuentran las instrucciones del programa principal del ASIP , los cuales están representados mediante datos de 36 bits.

Los bits que forman las instrucciones del ASIP poseen 36 bits debido a que es el número de bits que se necesita para controlar todas las señales internas de selección de las operaciones de los subsistemas ARITHMETIC, LOGIC , la señal de almacenamiento de datos del subsistema REG _H , también es necesario poder controlar los subsistemas MUX _A , MUX _B , MUX _C , los subsistemas DECODER , la señal H la cual hablita o deshabilita a las interrupciones , la señal de selección del subsistema DEMUX _C , las señales que son entradas del subsistema DECODER las cuales rigen el comportamiento de los subsistemas REG _C , además de estos 36 bits , el subsistema DEMUX _C utiliza 16 bits como un dato que se puede almacenar en el subsistema REG _C o como 16 bits dirección de un salto , este salto puede ser por software o por comparación .

El programador del ASIP puede definir las instrucciones que forman parte del programa principal y colocar su correspondiente dato en binario para generar instrucciones en el ASIP.

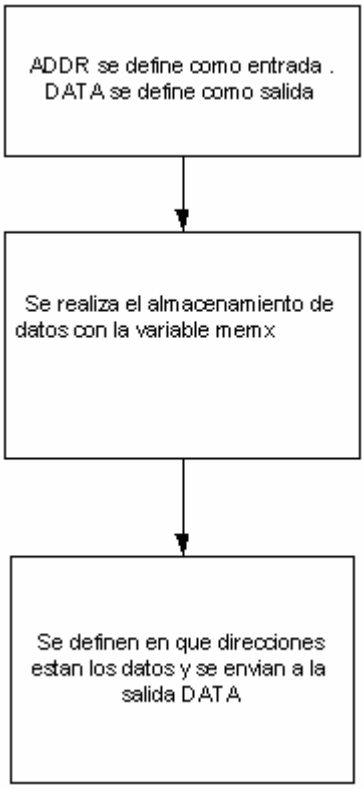


Fig.4.10: Diagrama de flujo del Subsistema ROM _R

En la figura 4.10 se muestra el diagrama de flujo del subsistema ROM _R , en el cual se describe el programa del apéndice H sección H.17 y también describe el programa del apéndice H sección H.18 , ambos programas son idénticos , pero en el primer programa en mención se han cargado todos los datos del programa TEST que explicaremos en el capitulo 7 y en el segundo programa en mención se han cargado 5 datos cualesquiera para poder simular el subsistema ROM _R.

El programa del subsistema ROM _R consta de las siguientes partes o fragmentos de programa:

- *Encabezado del programa :*

En este fragmento de programa se coloca las cabeceras de programación que se utilizan en el lenguaje VHDL , en el apéndice H sección H.17 se muestra el programa TEST el cual realiza una demostración de las instrucciones del ASIP , en este apéndice se puede observar que el primer fragmento del programa TEST define como se realiza la secuencia de las cabeceras del programa ROM _R .VHDL .

Estas cabeceras de programación se encuentran al inicio del texto del programa del ROM _R .VHDL y termina en el comando " end ROM_R ".

- *La Arquitectura del Programa Principal :*

En este fragmento de programa se definen las instrucciones del programa principal del ASIP. En el apéndice H sección H.17 se muestra como es La arquitectura del Programa Principal el cual es el segundo fragmento del programa ROM _R .VHDL , el cual empieza con el comando "architecture Funcionamiento of ROM_R is" y termina en el comando" Begin ".

Las instrucciones del programa principal del ASIP se van enumerando de la siguiente forma:

constant memx : std_logic_vector(35 downto 0) := mns…..mp;

Donde x es un numero que yo he hecho coincidir con la dirección de almacenamiento del subsistema ROM _R y "mns…..mp" es un número en binario de 36 bits , el cual esta dentro de dicha dirección y forma una instrucción.

- *Búsqueda de las instrucciones en el subsistema ROM _R :*

En este fragmento de programa se define la dirección del subsistema ROM _R en donde van las instrucciones del programa principal del ASIP . En el apéndice H sección H.17 se muestra el programa TEST completo , pero el tercer fragmento nos da la búsqueda de las instrucciones . En la señal de entrada ADDR se almacena la dirección para cada variable

memx . La búsqueda de las instrucciones en el subsistema ROM_R empieza en el comando ” with addr select ” y termina en el comando “ end Funcionamiento “ .

Cuando ejecutamos el comando : “ mem0 when "0000000000000000” “ implica que la señal almacenada en la dirección “0000000000000000” es mem0 , la señal men0 será la salida DATA del subsistema ROM _R la cual se observa en el apéndice H sección H.17.

Para poder simular el subsistema ROM _R es necesario darle un dato por cada dirección que se quiera simular , como no es posible simular 64K direcciones debido a que son demasiados datos , he decidido simular solamente 5 direcciones y observar sus retardos . Por este motivo simulo el programa del el apéndice H sección H.18 en el sistema decimal con el cual se observa el comportamiento del subsistema ROM _R , la simulación del programa del apéndice H sección H.18 se encuentra en la figura 4.11.

4.8 Simulación de la memoria del ASIP (Subsistema ROM_R)

En el apéndice H sección H.18 se muestra el programa ROM _R para 5 secuencias de datos , este programa lo he dividido en 3 partes , la primera parte es el encabezado del programa en el cual se encuentran todas las librerías del VHDL que se van a usar en este programa , la segunda parte es la arquitectura del programa principal en donde se definen los valores que se encuentran dentro de las direcciones del subsistema ROM _R estos valores son 5 datos de 36 bits cada uno , estos 5 datos el ASIP los entiende como 5 instrucciones , las cuales están dentro del subsistema ROM _R y en la tercera parte se definen las direcciones del subsistema ROM _R , las cuales son desde la dirección 0 hasta la dirección 4.

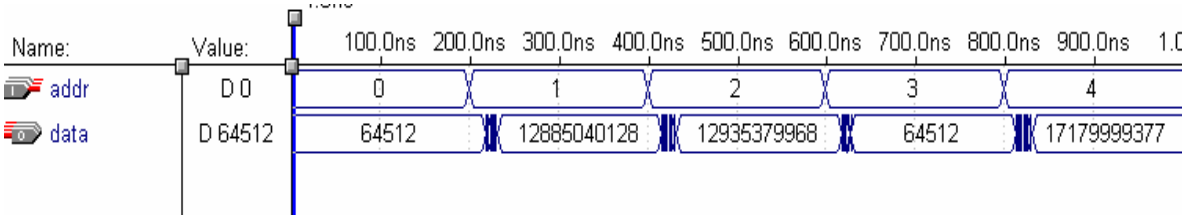


Fig.4.11: Simulación digital del subsistema ROM _R .GDF utilizando el programa del el anexo H sección H.18

En la figura 4.11 se observa que el subsistema ROM _R posee un retardo de 8ns . Al enviar las direcciones al subsistema ROM _R por su entrada ADDR se observa en la salida los datos almacenados en las direcciones del subsistema ROM _R, con lo cual se comprueba que el subsistema ROM _R tiene un comportamiento similar al de una memoria ROM en la

cual el usuario escribe sus datos en direcciones determinadas y estos son enviados cada vez que se envía dicha dirección a su entrada ADDR .

4.9 Estructura de la programación del subsistema CONTADOR y el Bus de Direcciones

El contador de direcciones es un subsistema del ASIP llamado CONTADOR el cual envía las direcciones al subsistema ROM _R . El tiempo en el cual el subsistema CONTADOR realiza un cambio en la dirección del subsistema ROM _R se denomina ciclo de instrucción .

Una instrucción se realiza en un ciclo de instrucción , pero un salto del programa principal se realiza en dos ciclos de instrucción , de los cuales un ciclo de instrucción se utiliza para definir a que dirección del subsistema ROM _R debe llegar el salto del programa principal y el otro ciclo de instrucción o programa se utiliza para efectuar el salto.

La Unión electrónica entre el subsistema CONTADOR y el subsistema ROM_R es llamado BUS DE DIRECCIONES debido a que por esa unión electrónica se trasmite las direcciones de memoria del ASIP.

Además de ello el subsistema CONTADOR posee un reset con el cual hace que el ASIP se dirija hacia la dirección cero , dicho reset se utiliza por lo general cuando el ASIP se encuentra colgado o su funcionamiento no es el adecuado y es necesario que realice otra vez todas sus instrucciones.

El subsistema CONTADOR posee una señal llamada LOAD la cual se comunica al subsistema C _LOGICO , cuando el subsistema C_ LOGICO le envía a la entrada LOAD del subsistema CONTADOR un 1 lógico se produce un salto de programa , es decir la dirección del subsistema ROM_R ya no va a ser la anterior aumentada en uno si no va a ser una dirección cualesquiera que elija el programador del ASIP.

El subsistema CONTADOR se comunica con el subsistema INTERRUPCIONES mediante la señal SEL _INT , el subsistema INTERRUPCIONES envía el estado de los pines IA , IB ; IC , ID, al subsistema CONTADOR para saber hacia que dirección se va a producir el salto.

Para definir hacia que dirección saltar he colocado direcciones aleatorias mediante el programa que se muestra en la figura 4.12 el cual se encuentra dentro del subsistema CONTADOR. El programador del ASIP no puede cambiar la dirección de los saltos de las interrupciones externas , pero para el diseñador del ASIP solo tendría que cambiar los valores que se cargan a la variable COUNT _I que están escrito con rojo dentro del programa de la figura 4.12 y colocarlo en el programa del subsistema CONTADOR. En la

figura 4.12 se muestra el programa del subsistema CONTADOR , y en la figura 4.13 se observa el diagrama de flujo del subsistema CONTADOR.

```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
entity CONTADOR is
port(CLOCK,RESET,LOAD: in std_logic;
     DATA: IN STD_logic_VECTOR(15 DOWNTO 0);
     SEL_INT: IN std_logic_VECTOR(2 DOWNTO 0);
     COUNT: out std_logic_VECTOR(15 DOWNTO 0));
     end CONTADOR;
architecture flujo of CONTADOR is
SIGNAL COUNT_I : std_logic_VECTOR(15 DOWNTO 0);
begin
PROCESS(CLOCK, RESET,SEL_INT)
begin
IF (RESET = '0') THEN
COUNT_I <= "0000000000000000";
ELSIF (CLOCK'EVENT AND CLOCK ='1') then
   IF LOAD ='1' THEN
   COUNT_I <= DATA;
   ELSE
      IF (SEL_INT = "000") THEN
      COUNT_I <= "1111111111111110";
      ELSIF (SEL_INT = "001") THEN
      COUNT_I <= "1111111111111100";
      ELSIF (SEL_INT = "010") THEN
      COUNT_I <= "1111111111111010";
      ELSIF (SEL_INT = "011") THEN
      COUNT_I <= "1111111111111000";
      ELSE
      COUNT_I <= COUNT_I + '1';
      END IF;
   END IF;
END IF;
END process;
COUNT <= COUNT_I;
END flujo;
```

FIGURA 4.12: Programa del subsistema CONTADOR con el cual se generan las direcciones de salto de las interrupciones externas

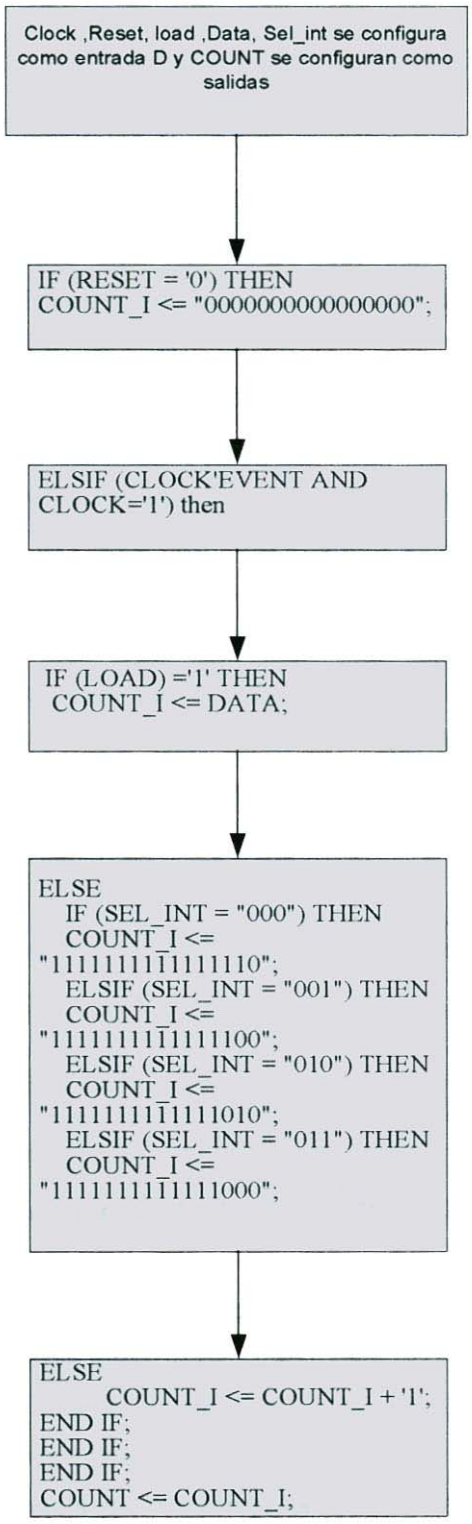


Fig.4.13: Diagrama de flujo del subsistema CONTADOR. VHDL

4.10 Simulación del contador de direcciones (Subsistema CONTADOR)

El subsistema INTERRUPCIONES interactúa directamente con el subsistema CONTADOR , el subsistema INTERRUPCIONES es el encargado de enviarle al subsistema CONTADOR el valor digital de las entradas IA ; IB ; IC ; ID .

El programa del subsistema INTERRUPCIONES se encuentra en el apéndice H sección H.15. El sistema CONTADOR esta unido electrónicamente con el subsistema INTERRUPCIONES los cuales tendrán el siguiente comportamiento:

Cuando IA = 1 el valor de SEL _INT será 000 , sin importar el valor de los otros pines de interrupción (pin de mayor prioridad) la dirección de la memoria a saltar será: 65518 que es un valor cualquiera que le he dado. Cuando IA = 0 y IB = 1 el valor de SEL _INT será 001 , sin importar el valor de los otros pines de interrupción la dirección de la memoria será : 65516 . Cuando IA = 0 , IB = 0 y IC = 1 el valor de SEL _INT será 010, sin importar el valor de el otro pin de interrupción la dirección de la memoria será : 65514 y por ultimo cuando IA = 0 , IB = 0 y IC = 0 , ID = 1 el valor de SEL _INT será 011 y la dirección de la memoria será: 65512.

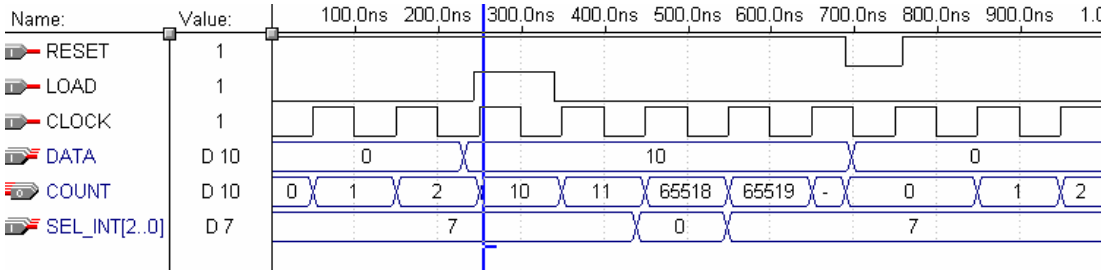


Fig.4.13: Simulación digital del subsistema CONTADOR

En la simulación de la figura 4.13 se muestra el conteo incrementándose en una unidad del subsistema CONTADOR , cuando la entrada SEL _INT esta en 7 en el sistema decimal . Si la entrada LOAD es 1 por un periodo de tiempo , el subsistema CONTADOR producirá un salto hacia la dirección que se encuentra en la entrada DATA , si cambia la entrada SEL _ INT de 7 a 0 realizando un salto hacia la dirección en el sistema decimal 65518 , este cambio de SEL _INT se produce cuando se realiza una interrupción , si SEL _INT a sido 0 durante un periodo de tiempo , significa que la interrupción ha sido de primer orden o de mayor prioridad, al poner a 0 lógico la señal de RESET el subsistema CONTADOR empezó a contar desde 0 otra vez .

Se observa además que el retardo a la salida de el subsistema CONTADOR es despreciable.

CAPITULO V:
DESARROLLO DE LA ARQUITECTURA DEL ASIP

En este capitulo definiremos el tipo de arquitectura del ASIP y mostraremos la distribución de cada subsistema antes mencionado dentro del ASIP.

5.1 Generalidades

Después de haber diseñado todas las partes del ASIP a llegado el momento de unir todos estos subsistemas mencionados en los capítulos anteriores y formar el ASIP . Para unir todos los subsistemas tomo como referencia el diagrama de bloques del ASIP que se encuentra en la figura 1.1 del capitulo 1 .

5.2 Arquitectura del ASIP

En la figura 5.1 se muestra la arquitectura del ASIP en la cual se han unido todos los subsistemas ya antes mencionados en los capítulos anteriores.

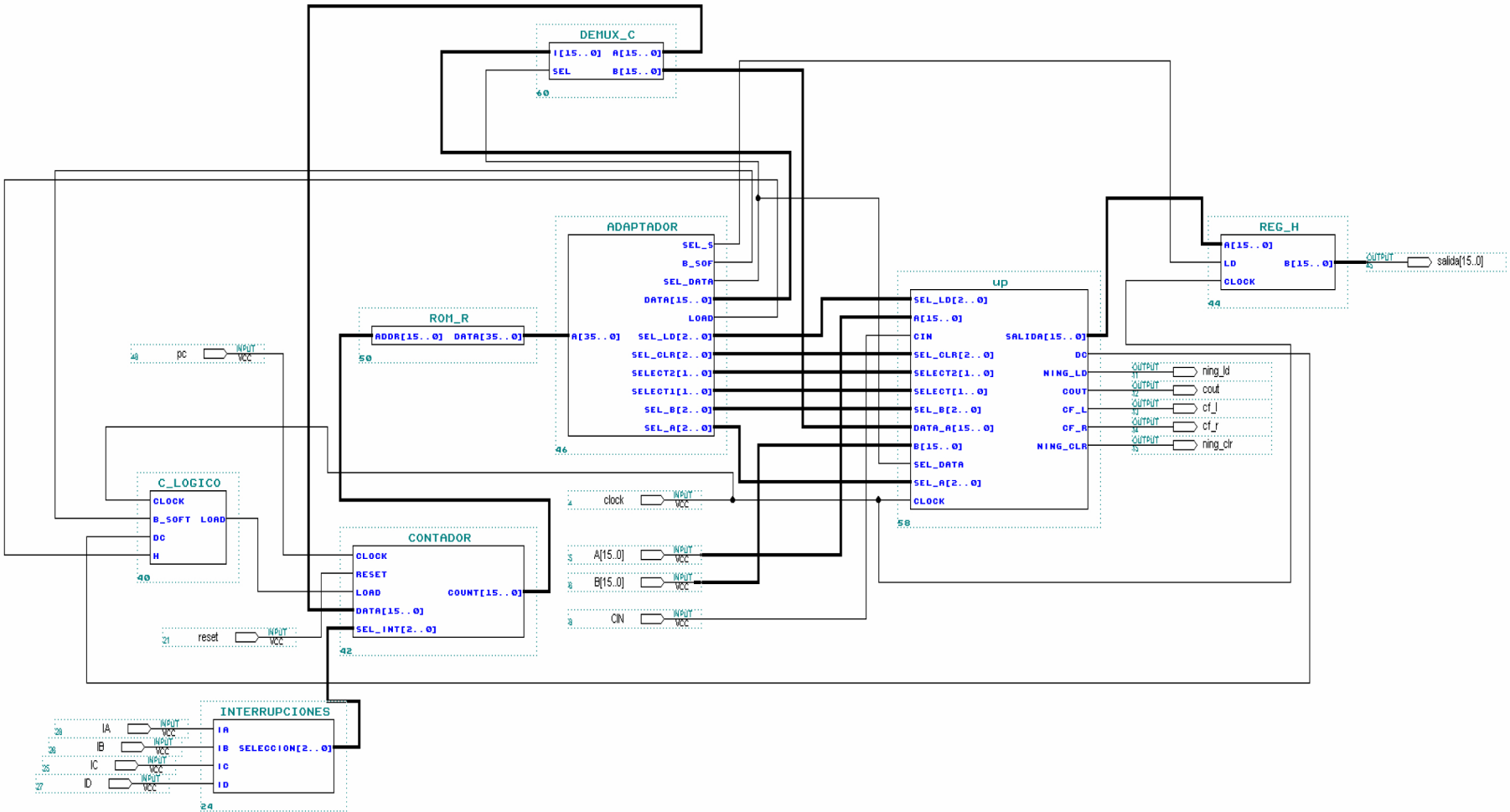


Fig 5.1: Arquitectura del ASIP

5.3 Desarrollo de la Arquitectura pipeline del ASIP

La arquitectura que muestra el ASIP es conocida como Arquitectura pipeline debido al tipo de funcionamiento del ASIP .

Para que funcione adecuadamente el ASIP es necesario que el subsistema ROM_R termine de enviar sus datos correctamente para que luego empiece a funcionar el subsistema ADAPTADOR , después de que el subsistema ADAPTADOR envié sus datos correctamente empezará a funcionar el subsistema Unidad de Procesamiento y cuando este termine de enviar sus datos correctamente empezará a funcionar el subsistema REG _H , este es el recorrido mas critico por lo cual se genera mas retardos , es posible de que solo se desee cargar un dato a un registro con lo cual el retardo será menor debido a que no se va a recorrer toda la arquitectura del ASIP si no solo hasta sus registros, pero para criterios de diseño es necesario considerar el caso mas desfavorable así que consideraremos como retardo el primer caso mencionado.

Si consideramos entonces la arquitectura del ASIP como pipeline , el retardo a la salida del ASIP es igual al retardo del subsistema ADAPTADOR , el cual posee un retardo que es despreciable mas el retardo del Unidad de Procesamiento que es 40ns mas el retardo del subsistema REG _H que es 8ns y esto hace un retardo de 48ns considerando además que la memoria del ASIP se demora 8ns una vez que recibe la información del subsistema CONTADOR . El retardo del ASIP se considera solo el tiempo que demora el ASIP en dar el valor de la salida una vez que sale la información del subsistema ROM _R y esto es 48ns.

Las líneas de carga o etapas del subsistema ALU son dos , por consecuencia las líneas de carga o etapas del Unidad de Procesamiento son cinco , con lo cual concluyo que las líneas de carga o etapas del ASIP son siete.

CAPITULO VI:
DISEÑO Y SIMULACIÓN DE LOS SUBSISTEMAS QUE PRODUCEN LOS SALTOS E INTERRUPCIONES CON PINES EXTERNOS DEL ASIP

En este capitulo se describe como se realizan los saltos y las interrupciones por pines externos dentro del ASIP, además de ello se detalla todos los subsistemas que intervienen en los saltos e interrupciones por pines externos.

6.1 Generalidades

En la secciones 1.9 y 1.11 se describe como deben funcionar los subsistemas INTERRUPCIONES y C _ LOGICO para poder realizar saltos e interrupciones con pines externos , estos dos subsistemas están unidos al subsistema CONTADOR el cual nos da la dirección del subsistema ROM _R del ASIP.

6.2 Simulación del subsistema C _LOGICO

Para decirle al subsistema CONTADOR cuando debe de saltar se ha creado el subsistema C _ LOGICO el cual posee una lógica para determinar en que momento habilitar la señal LOAD del subsistema CONTADOR. En el apéndice H sección H.1 se encuentra el programa del subsistema C _LOGICO y en la figura 6.1 se muestra el diagrama de flujo del subsistema C _LOGICO .

Fig.6.1:Diagrama de flujo del subsistema C _LOGICO. VHDL

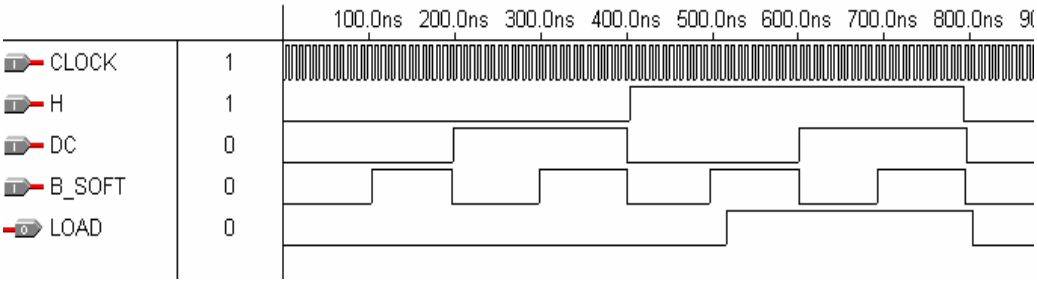


Fig.6.2: Simulación digital del subsistema C _LOGICO

En la figura 6.2 se observa que para que LOAD sea 1 lógico es necesario que H sea 1 lógico debido a que H habilita los saltos de memoria del ASIP.

Si DC es 1 lógico implica que la entrada del subsistema COMPARADOR es cero , si B _SOFT es 1 lógico implica que el programador del ASIP a decidido hacer una instrucción de salto por software hacia una dirección aleatoria del subsistema ROM _R .

6.3 Simulación del subsistema INTERRUPCIONES

En el apéndice H sección H.15 se encuentra el programa del subsistema INTERRUPCIONES y en la figura 6.3 se muestra el diagrama de flujo del subsistema INTERRUPCIONES.

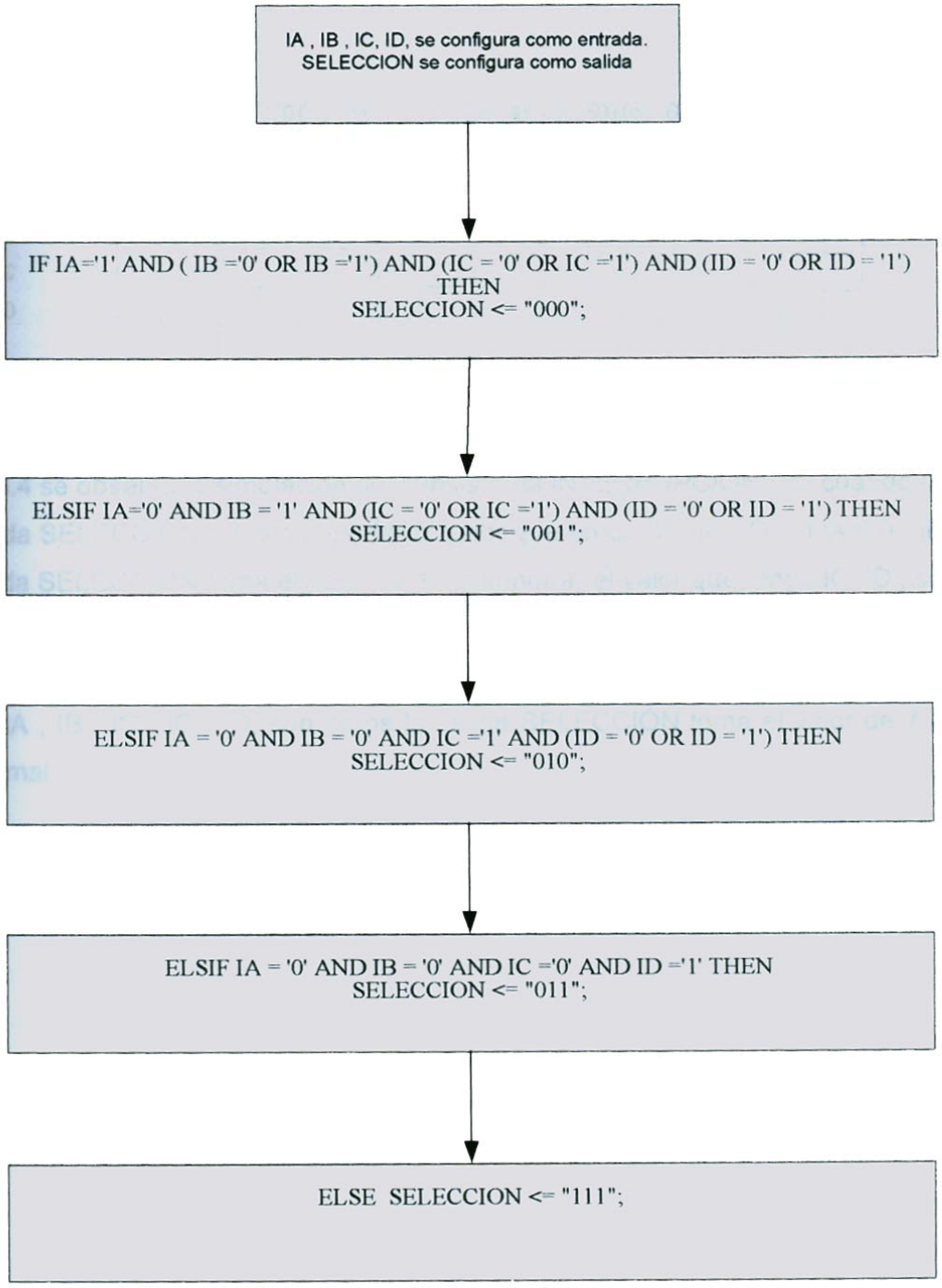


Fig.6.3: Diagrama de flujo del subsistema INTERRUPCIONES

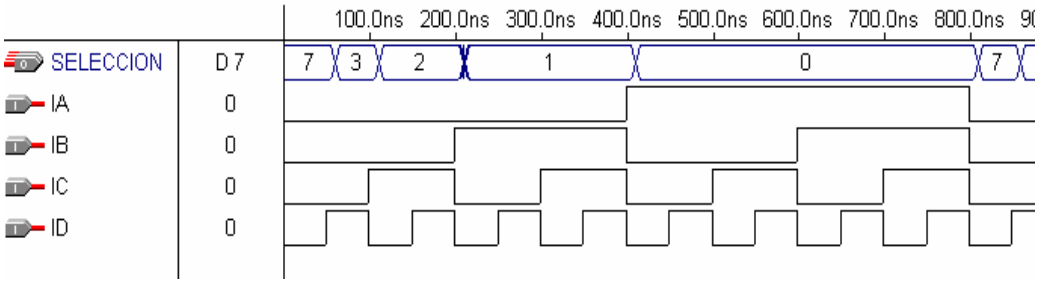


Fig.6.4: Simulación digital del subsistema INTERRUPCIONES

En la figura 6.4 se observa la simulación del subsistema INTERRUPCIONES , cuando IA = 1 lógico la salida SELECCIÓN = 0 sin importar el valor que tenga IB , IC, ID , si IA = 0 , IB = 1 lógico la salida SELECCIÓN toma el valor de 1 sin importar el valor que tenga IC, ID , si IA = 0 , IB = 0 , IC = 1 lógico la salida SELECCIÓN toma el valor de 2 en el sistema decimal y si IA = 0 , IB = 0 , IC = 0 ID = 1 lógico la salida SELECCIÓN toma el valor de 3 en el sistema decimal , si IA , IB , IC , IC , ID, son ceros la salida SELECCIÓN toma el valor de 7 en el sistema decimal.

CAPITULO VII:
ELABORACIÓN DEL PROGRAMA TEST PARA SIMULAR LAS INSTRUCCIONES DEL ASIP

He decidido realizar un programa para mostrar las instrucciones del ASIP y además mostrar la simulación de las interrupciones mediante pines externos, saltos por software y saltos por comparación.

7.1 Generalidades

En el apéndice H en la sección H.17 se encuentra escrito el programa TEST dentro del subsistema ROM _R el cual va a ser simulado y comentado en este capitulo, este programa es compilado en el software MAX + PLUS II de la empresa ALTERA y dentro del ASIP.

Cabe resaltar además que todas las figuras de este capitulo se refieren al programa TEST del apéndice H sección H.17

En la figura 7.1 se muestra como el simulador de la empresa ALTERA MAX +PLUS II compila el programa TEST diseñado.

Es necesario saber que el programa MAX +PLUS II no genera ningún error cuando compila el programa TEST.

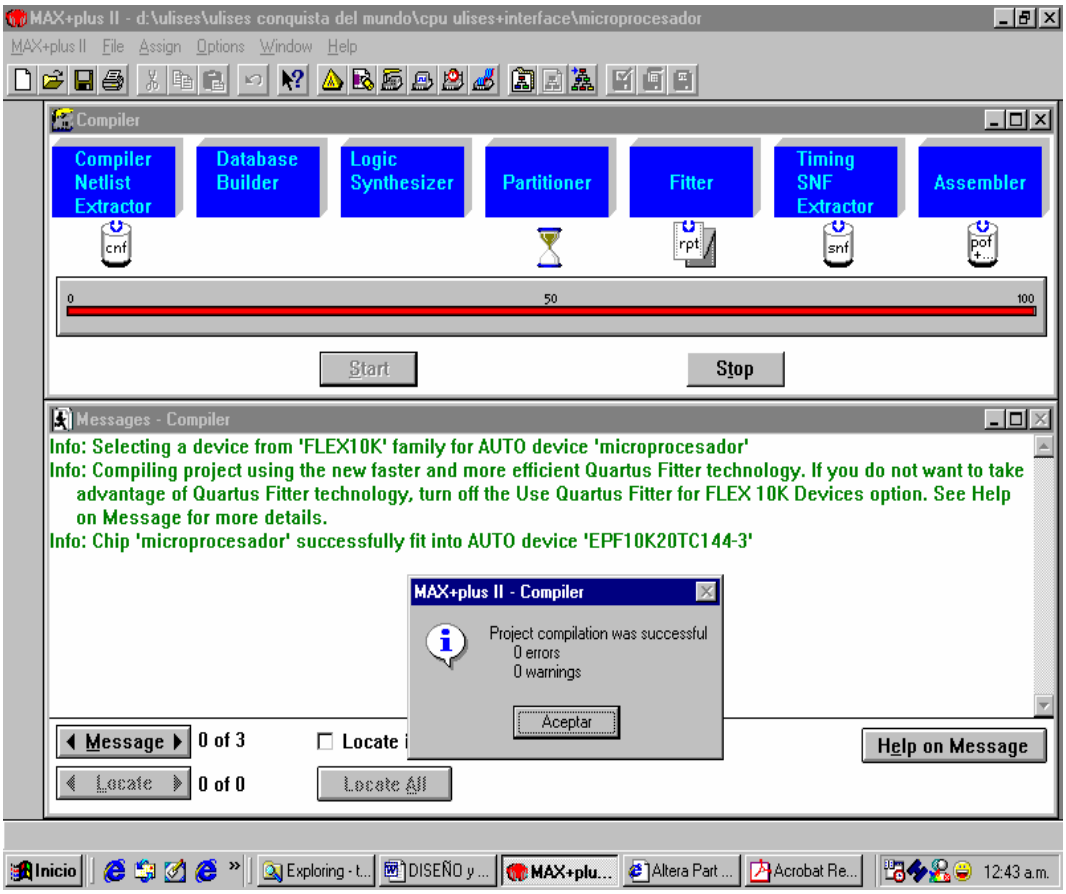


Fig.7.1: Compilación del ASIP con el programa test en el subsistema ROM _R

En la figura 7.1 se observa que el programa TEST no posee ningún error en la programación.

7.2 Simulación y comentarios del programa test para verificar los recursos del ASIP

De la compilación concluimos que el dispositivo de ALTERA que necesito para grabar mi programa es el dispositivo **EPF10K20TC144-3** el cual es un FLEX de 10K .

La primera simulación que realizaremos del programa TEST es sin interrupciones por pines externos , es decir ningún pulsador a generado un 1 lógico en las entradas IA , IB , IC o ID. Comúnmente a esta simulación se le denomina simulación del programa principal ya que no se va realizar ningún salto y este es el programa que rige el diseño electrónico.

En la figura 7.2 se encuentra la simulación en donde se observa que el contador va de 0 hasta 8 y luego empieza nuevamente a contar de 0 esto se debe a que el programa TEST posee un bucle que repite el programa indeterminadamente.

Cuando el subsistema CONTADOR esta en ‘0’ se encuentra en dicha dirección la instrucción ‘GOTO_0’ el cual se utiliza para inicializar al ASIP cuando ocurre algún salto ; como el programa TEST posee un bucle desde la dirección ‘0’ hasta la dirección ‘8’ , en la dirección ‘0’ se encuentra la instrucción para que retorne del salto que se realiza en la dirección ‘8’ este salto lo a realizado el programador del ASIP sin que ocurra ningún evento en particular por ello a este salto se le denomina salto por software.

Cuando el subsistema CONTADOR esta en ‘1’ se encuentra en dicha dirección la instrucción ‘MOV A, R0’ , como se observa en la simulación el valor en la entrada ‘A’ es de 18500 en el sistema decimal , pero justo cuando cambia el contador de direcciones de la memoria o subsistema CONTADOR a 1 el valor de la entrada ‘A’ se carga en uno de los subsistema REG _C enumerado como R0.

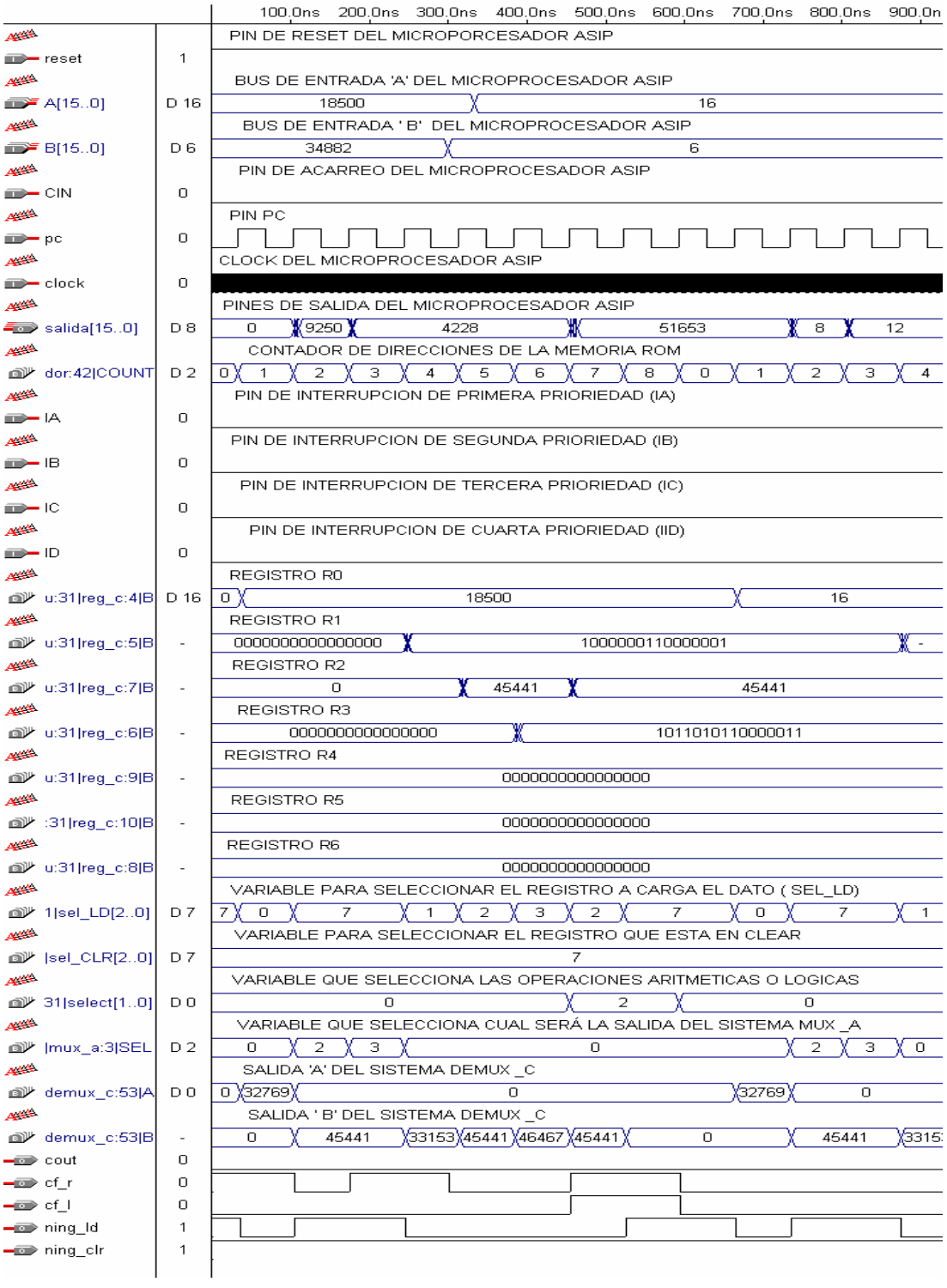


Fig.7.2: Simulación del programa test del ASIP cuando no esta habilitada la interrupción mediante pines externos[1]

[1] El programa TEST se encuentra en el apéndice H sección H.17

Cuando el subsistema CONTADOR esta en ‘2’ se encuentra en dicha dirección la instrucción ‘ASHL R0 , OUT ’ , esta instrucción realiza la operación desplazamiento a la izquierda del subsistema REG _C enumerado como R0 y envía el valor hacia los pines de salida del ASIP . En el subsistema REG _C enumerado como R0 se encuentra almacenado el valor 18500 en el sistema decimal que en el sistema binario es 0100100001000100 , al realizar el desplazamiento a la izquierda el dato se convertirá en 010010000100010 en binario que en el sistema decimal es el numero 9250 que es el valor que se encuentra en la salida del ASIP.
Cuando el subsistema CONTADOR esta en ‘3’ se encuentra en dicha dirección la instrucción ‘ASHL B , OUT ’ , esta instrucción realiza la operación desplazamiento a la derecha de la salida ‘B’ y envía dicho valor a la salida del ASIP . En la entrada ‘B’ se encuentra el valor 34882 en el sistema decimal cuyo valor en el sistema binario es 1000100001000010 al realizar el desplazamiento hacia la derecha la señal de entrada se convierte en el valor binario 0001000010000100 cuyo valor en el sistema decimal es 4228 el cual es el valor de salida del ASIP que se muestra en la simulación de la figura 7.2.
Cuando el subsistema CONTADOR esta en ‘4’ , ‘5’, ’6’ se realizan las instrucciones:
MOVML R1,B1000000110000001 ,
MOVML R2, B1011000110000001 ,
MOVML R3, B1011010110000011 .
La función de de estas instrucciones es guardar en los subsistemas REG _C enumerados como R1 , R2 , R3 los valores
1000000110000001 , 1011000110000001 , 1011010110000011 respectivamente.
En la simulación mostrada en la figura 7.2 se observa que cuando el subsistema CONTADOR es ‘4’ el valor que se muestra en el subsistema REG _C enumerado como R1 es 1000000110000001 , cabe resaltar que el subsistema REG _C enumerado como R1 lo he simulado en el sistema binario para tener una mejor comprensión del programa TEST y para hacer resaltar que es posible simular las entradas y salidas así como las variables en el sistema de numeración binaria o en el sistema de numeración decimal.
Cuando el contador de direcciones de la memoria ROM es ‘5’ el valor que se carga en el subsistema REG _C enumerado como R2 es 1011000110000001 en el sistema binario cuyo valor en el sistema decimal es 45441 , dicho valor se muestra en la simulación del subsistema REG _C enumerado como R2.

Cuando el subsistema CONTADOR es ‘6’ el valor que se carga en el subsistema REG _C enumerado como R3 es 1011010110000011 en el sistema binario , dicho valor se muestra en la simulación del subsistema REG _C enumerado como R3.

Cuando el subsistema CONTADOR es ‘7’ se realiza la instrucción SUMA R0 , R1,OUT , el cual suma los subsistemas REG _C enumerados como R0 , R1 y , dicho valor lo envía a la salida del ASIP, el valor de R1 cuando el subsistema CONTADOR es 7 es 1000000110000001 en el sistema binario cuyo valor en el sistema decimal es 33153 mientras que en R0 su valor en el sistema decimal es 18500 al sumarlos nos da el valor de 51653 que es la señal que se muestra en los pines de salida de la simulación del ASIP.

Cuando el subsistema CONTADOR es ‘8’ se realiza el la instrucción SALTA _0 el cual hace saltar al contador de direcciones de la memoria ROM a cero produciendo un bucle repetitivo como observamos en la simulación, cuando se repite este programa realiza las operaciones mencionadas antes con los valores actuales de las entradas del ASIP.

Mencionaré además de que existe una entrada CIN la cual en esta simulación se encuentra en 0 lógico , esto es debido a que esta entrada simula algún acarreo que nos pueda enviar algún dispositivo que trabaje junto con el ASIP .

Observamos además en la figura 7.2 que el pin de RESET , que hace que el subsistema CONTADOR envié al subsistema ROM _R la dirección cero cuando el pin RESET esta en 0 lógico , el pin RESET esta simulado en 1 lógico por ese motivo en esta simulación no se a realizado ningún evento de reset al programa principal.

En la figura 7.3 observamos un evento de RESET producido por el pin de RESET , vemos que después de que el pin de reset vuelve a 1 lógico el subsistema CONTADOR del ASIP empieza a contar normalmente.

El pin PC nos da el ciclo de instrucción es decir el tiempo en el cual el subsistema CONTADOR del ASIP esta enviando las direcciones del programa principal al subsistema ROM _R .

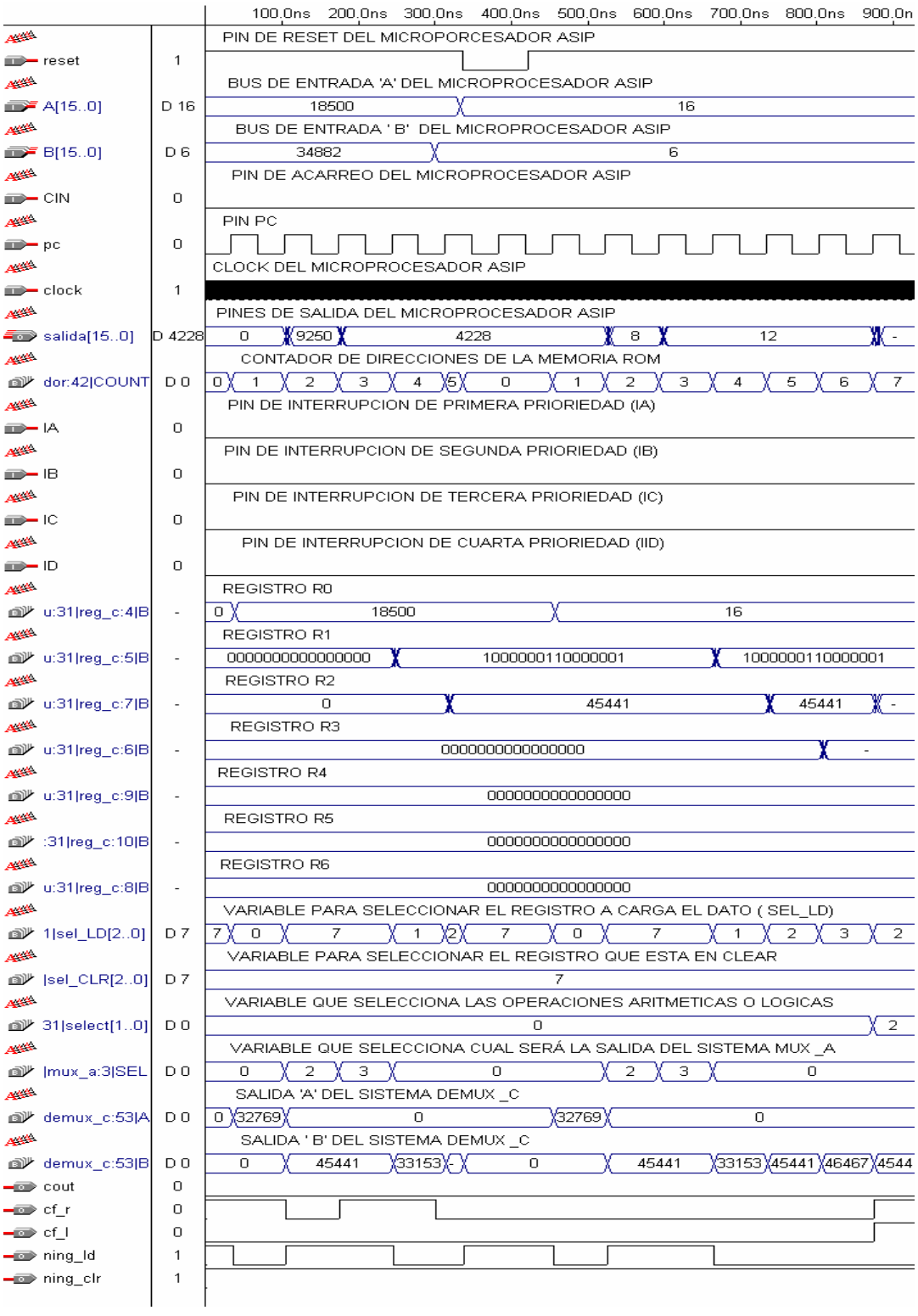


Fig.7.3: Simulación del programa test del ASIP cuando se produce un reset [2]

[2] El programa TEST se encuentra en el apéndice H sección H.17

Otra señal del ASIP a comentar es SEL _LD esta señal es interna en el ASIP pero la hemos simulado para hacer notar los momentos en los cuales se carga un subsistema REG _C , cuando la señal SEL _ LD toma el valor 0 implica que se ha cargado un dato en el subsistema REG _C enumerado como R0 , si toma los valores de 1 , 2 , 3 , 4 , 5 o 6 implica que se han cargado valores a los subsistemas REG _C enumerados como R1 , R2 , R3 ,R4 , R5 y R6 respectivamente , si SEL _LD toma el valor de 7 implica que no se ha realizado ninguna carga de datos a algún registro del bus de datos. Análogamente ocurre con la señal interna SEL_ CLR pero aquí se pone a cero algún subsistema REG _C, para nuestra simulación la señal SEL _CLR toma siempre el valor de 7 lo cual implica que no se ha puesto a cero o no se le a dado un reset a ningún registro del bus de datos.

Otra señal del ASIP a comentar es SELECT la cual nos dice que operación aritmética o lógica se a realizado en los subsistemas ARITHMETIC o LOGIC estas operaciones se realizan a la vez pero eso no implican que las dos operaciones van a pasar hacia la señal de salida del subsistema ALU , para ello la señal interna SEL selecciona cual de las salidas de los subsistemas ARITHMETIC , LOGIC , ASHL , ASHR pasaran a ser señales de salida del subsistema ALU del ASIP.

Observemos ahora la simulación del programa TEST cuando ocurre una interrupción de primera prioridad.

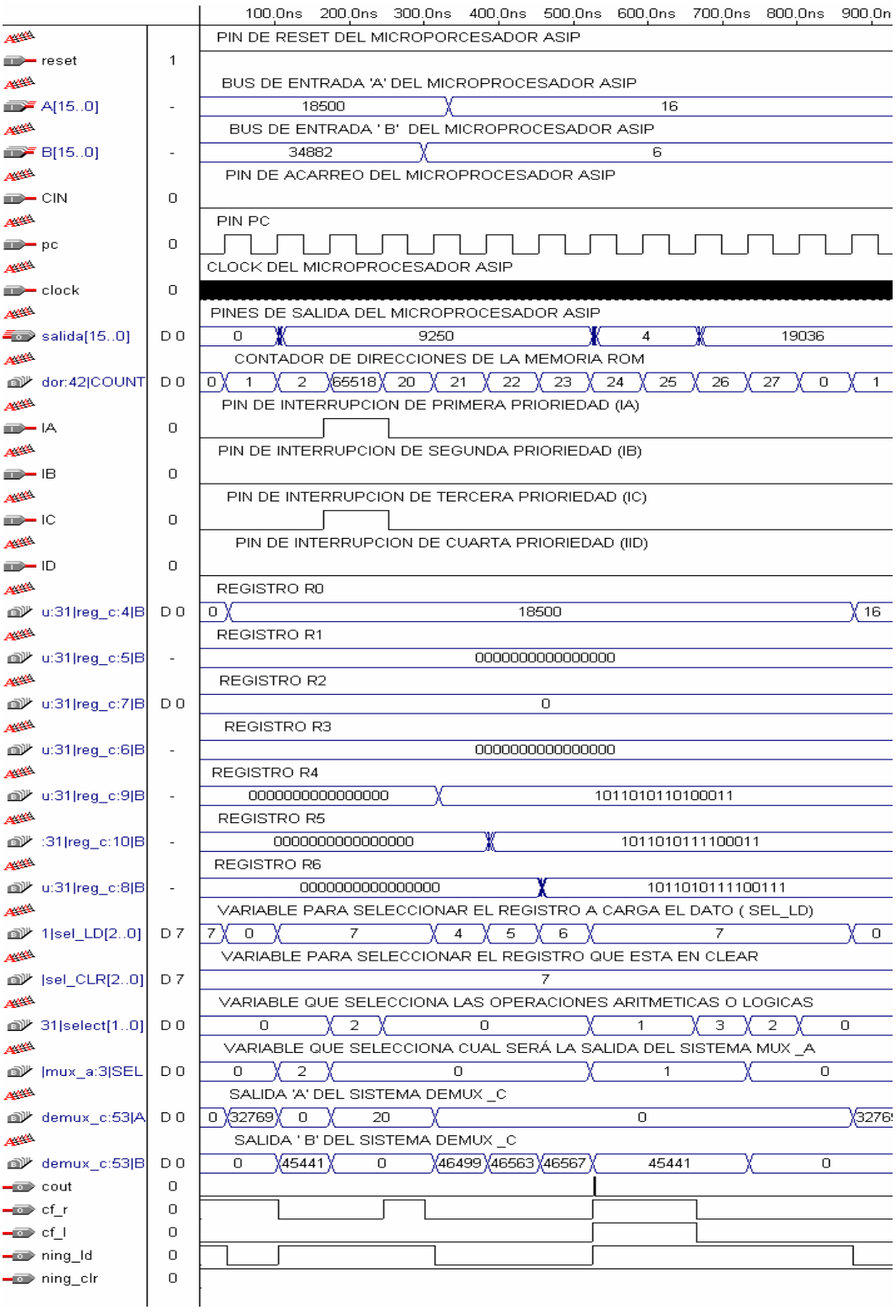


FIGURA 7.4: Simulación del programa test del ASIP cuando se produce una interrupción de primera prioridad[3]

[3] El programa TEST se encuentra en el apéndice H sección H.17

En la figura 7.4 se observa la simulación del programa TEST cuando el ASIP sufre una interrupción de primera prioridad , esto sucede porque el pin IA posee un pulso de 1 lógico, sin importar el valor de los otros pines de interrupción[4] se realizará una interrupción hacia la dirección 65518.

Cuando el subsistema CONTADOR es '65518' se coloca en dicha dirección del subsistema ROM _R la instrucción SALTA_20 el cual nos dice que el subsistema ROM _R saltará hacia la dirección 20 como se muestra en la figura 7.4.

Cuando subsistema CONTADOR es '20' coloco en dicha dirección la instrucción GOTO_20 con la cual el ASIP recibe todos los datos de la interrupción efectuada, es por ello que las interrupciones necesitan 2 ciclos de instrucción uno para saltar y otro para colocar al hardware dicha información en este caso las direcciones usadas son 65518 y 20 de la memoria ROM _R del ASIP.

Cuando el subsistema CONTADOR es '21' , '22' , '23' coloco en dichas direcciones las instrucciones MOVML R4,B1011010110100011 , MOVML R5,B1011010110100011 , MOVML R6,B1011010111100011 estas instrucciones se cargan en los subsistemas REG _C enumerados como R4 , R5 y R6 los datos 1011010110100011 , 1011010111100011 , 1011010111100111 respectivamente , esto se puede observar en la figura 7.4.

Cuando el subsistema CONTADOR es '24' , '25' se repite por 2 ciclos de reloj la instrucción xor R5,R6 esto se hace para hacer que en la salida del ASIP aparezca durante dos ciclos de instrucción dicho resultado.

Como los valores de los registros R5 y R6 son : 1011010110100011 , 1011010111100011 respectivamente al aplicar la operación xor obtendremos: 0000000000000100 que en el subsistema decimal es 4 lo cual se aprecia en la simulación de la figura 7.4.

Cuando el subsistema CONTADOR es '26' se ejecuta la instrucción not R4 , como en el registro R4 se encuentra la data 1011010110100011 entonces not R4 debe de ser igual a 0100101001011100 que en el subsistema decimal es el numero 19036 el cual se muestra en la simulación de la figura 7.4

Cuando el subsistema CONTADOR es '27' se ejecuta la instrucción SALTA_0 con lo cual el programa TEST hace que la dirección del subsistema ROM _R sea cero.

En la figura 7.5 se observa la simulación del ASIP cuando se produce una interrupción de segunda prioridad , esto sucede porque el pin IB posee un pulso de 1 lógico y además IA

[4] No importa que el pin IC posea un pulso de 1 lógico

esta en 0 lógico[5] , este evento que es en un momento cualesquiera en el cual se realizará una interrupción hacia la dirección 65516.

Cuando el subsistema CONTADOR es ‘65516’ y se envía dicha dirección al subsistema ROM _R , y además en esta dirección se encuentra la instrucción SALTA_100 , la dirección del subsistema ROM _R saltará hacia la dirección 100 como se muestra en la figura 7.5.

[5] En la interrupción de segunda prioridad el valor de los pines IA = 0 y además IB =1 sin importar el valor de IC y ID

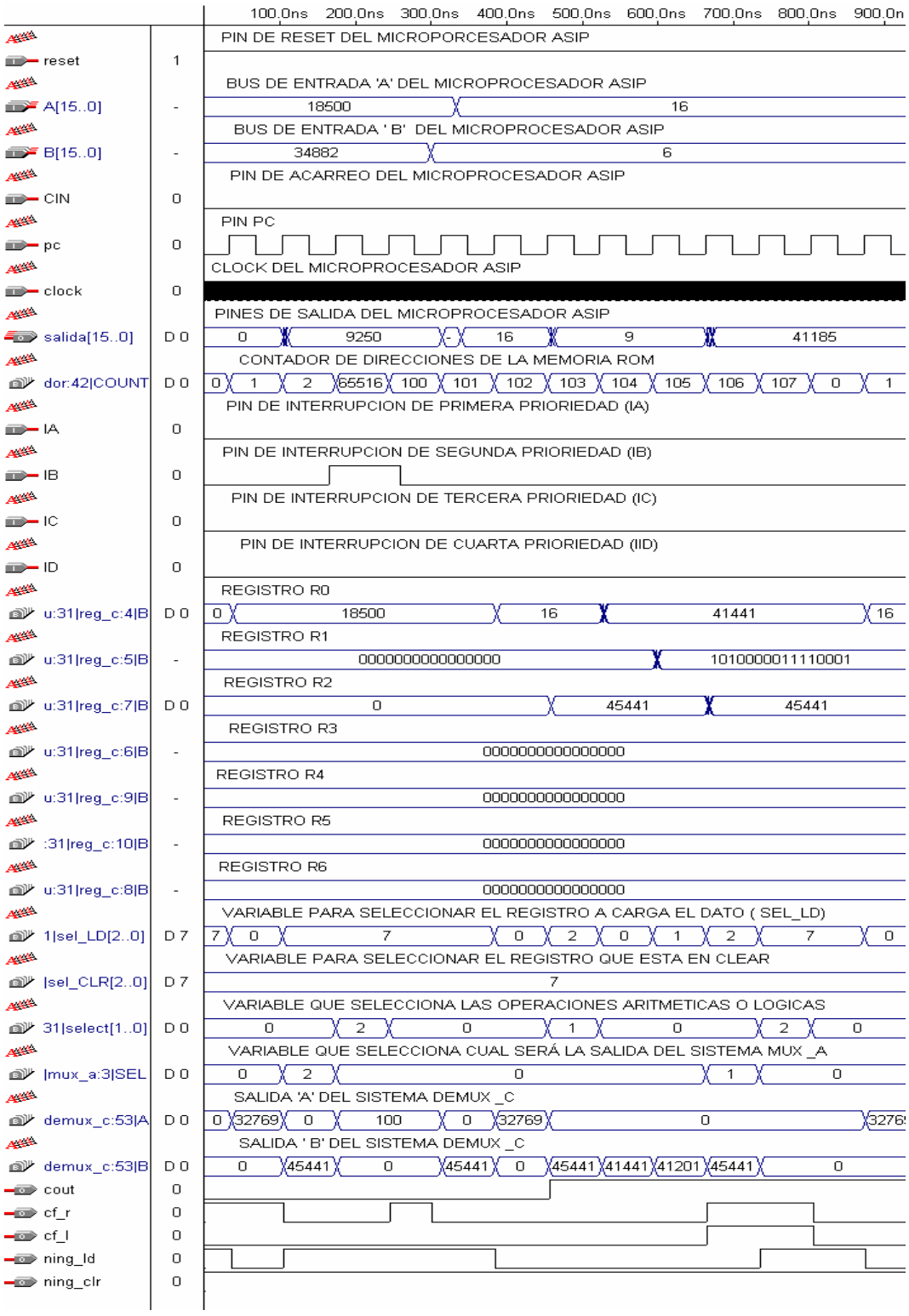


Fig.7.5: Simulación del programa test del ASIP cuando se produce una interrupción de segunda prioridad[6]

[6] El programa TEST se encuentra en el apéndice H sección H.17

Cuando el subsistema CONTADOR apunta hacia la dirección ‘100’ coloco en dicha dirección la instrucción GOTO_100 con el cual se recibe todos los datos de la interrupción efectuada.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘101’ el ASIP ejecuta la instrucción TRANSF A , OUT el cual transfiere el valor de la entrada ‘A’ hacia la salida del ASIP , para esta simulación el valor de la entrada A es 16 en el sistema decimal y el ASIP carga dicho valor a su salida , esto se muestra en la figura 7.5.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘102’ se ejecuta la instrucción MOV A, R0 con la cual la data de la entrada ‘A’ se carga en el subsistema REG _C el cual a sido enumerado como R0.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘103’ se ejecuta la instrucción RESTA A, B con la cual se realiza la resta de A – B – (not CIN) con lo cual la salida será 16 – 6 – not (0) = 9 a esto se llama resta sin préstamo y este valor se observa en la salida del ASIP.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘104’ y ‘105’ se ejecutan las instrucciones MOVML R0,B1010000111100001 y MOVML R1,B1010000011100001.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘106’ ejecuta la instrucción AND R0,R1 , al ejecutar la operación AND el resultado es 1010000011100001 en el sistema binario el cual en el sistema decimal es 41185 , el cual es el valor se muestra en la simulación.

Cuando el subsistema CONTADOR apunta hacia la dirección es ‘107’ ejecuta la instrucción SALTA_0 para regresar a la posición 0 de memoria.

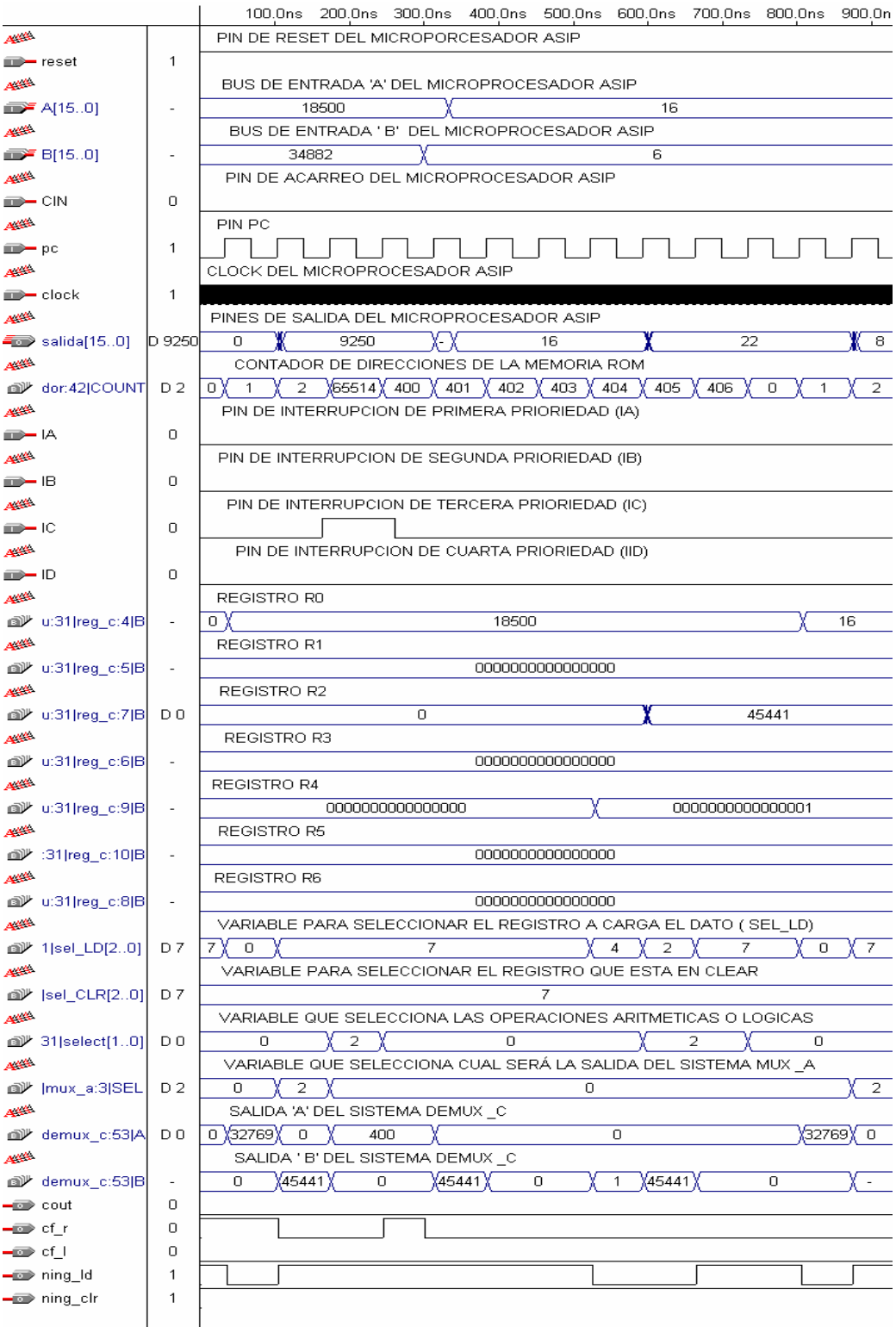


Fig.7.6: Simulación del programa test del ASIP cuando se produce una interrupción de tercera prioridad[7]

[7] El programa TEST se encuentra en el apéndice H sección H.17

En la figura 7.6 se observa la simulación del ASIP cuando se produce una interrupción de tercera prioridad , esto sucede porque el pin IC posee un pulso de 1 lógico , este evento que es en un momento cualesquiera y realizará una interrupción hacia la dirección 65514.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘65514’ se coloca en dicha dirección el subsistema ROM _R dentro del cual se encuentra la instrucción SALTA_400 el cual nos dice que el subsistema REG _C saltará hacia la dirección 400 como se muestra en la figura 7.6.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘400’ coloco en dicha dirección la instrucción GOTO_400 con el cual se recibe todos los datos de la interrupción efectuada.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘401’ ejecuto la instrucción TRANSF A , OUT con la cual la salida del ASIP tiene el valor de la entrada ‘A’, esto se observa en la simulación de la figura 7.6.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘402’ y ‘403’ ejecuto una sola instrucción de 2 ciclos de instrucción llamado COMPARADOR_0 si el valor de la salida del subsistema ALU es 0 el subsistema CONTADOR saltará a la dirección 0 , caso contrario el subsistema CONTADOR pasara a la siguiente dirección.

Como en la simulación de la figura 7.6 la salida del subsistema ALU es 16 en el sistema decimal no se realiza el salto por comparación.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘404’ ejecuto la instrucción MOVML R4 , B0000000000000001.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘405’ ejecuto la instrucción SUMA A , B , OUT .

Cuando el subsistema CONTADOR apunta hacia la dirección ‘406’ ejecuto la instrucción SALTA _ 0 el cual nos hace regresar a la dirección cero del subsistema ROM _R .

Cabe resaltar que si hubiera sucedido que en los ciclos de instrucción ‘402’ y ‘403’ el subsistema ALU hubiera encontrado el valor de cero a su salida , la salida DC del subsistema COMPARADOR seria 1 lógico y se hubiera producido un salto hacia la dirección cero.

En la figura 7.7 se observa que la entrada ‘A ’ ha sido cambiada de 16 a cero para obligar al ASIP a efectuar un salto por comparación, en los ciclos de instrucción ‘402’ y ‘403’ la señal de salida del subsistema ALU genera un cero lo cual hace que el ASIP realice un salto debido la instrucción COMPARADOR_0 , el numero cero implica que el subsistema

CONTADOR pasará a valer cero después de que se ejecuto las instrucciones de las direcciones ‘402’ y ‘403’ como se muestra en la simulación 7.7.

En la figura 7.8 se observa la simulación del ASIP cuando se produce una interrupción de cuarta o ultima prioridad , esto sucede porque el pin ID posee un pulso de 1 lógico , este evento es en un momento cualesquiera y realizará una interrupción hacia la dirección 65512.

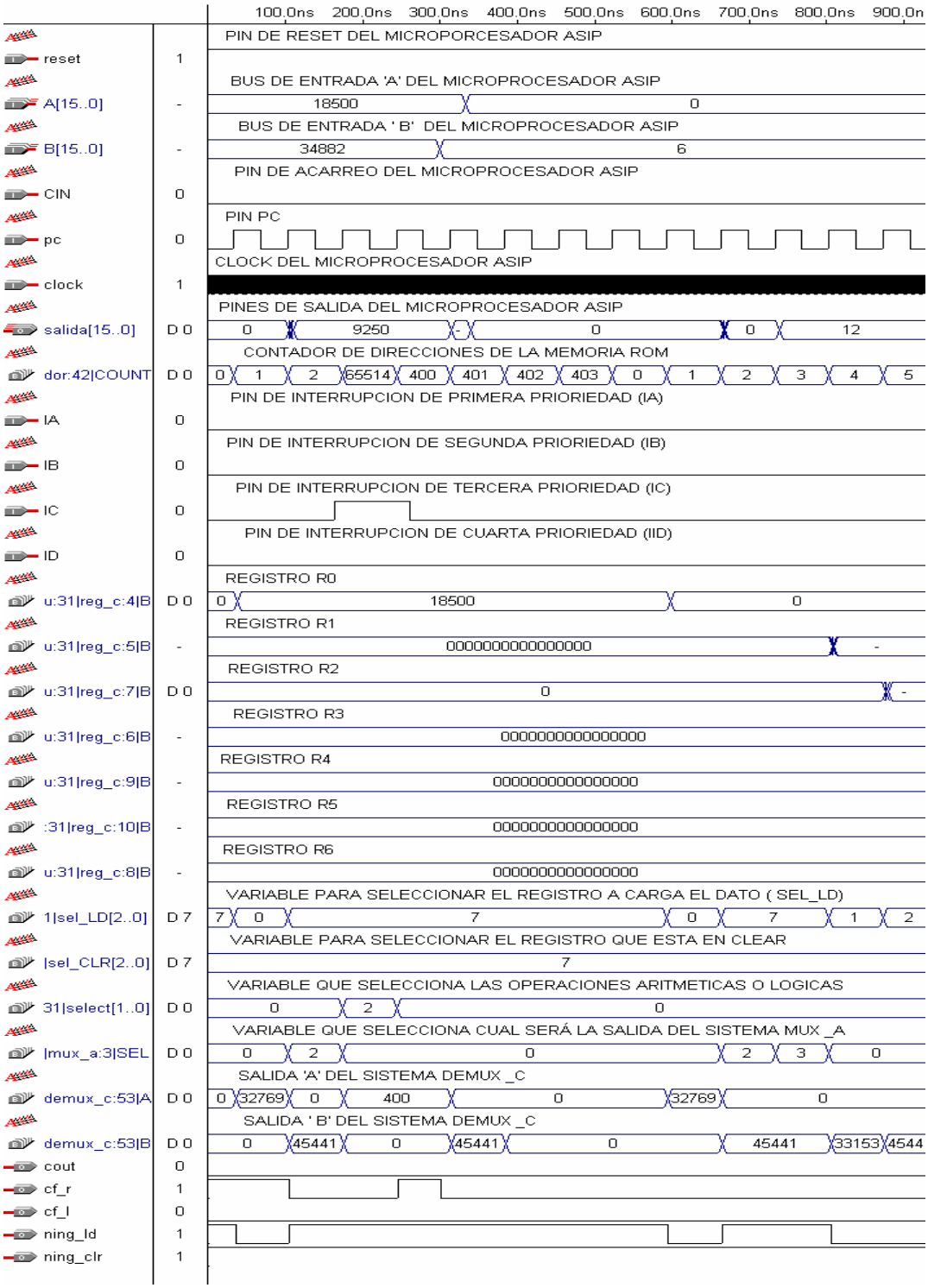


FIGURA 7.7: Simulación del programa test del ASIP cuando se produce una interrupción de tercera prioridad y aparece un salto por acción del subsistema COMPARADOR[8]

[8] El programa TEST se encuentra en el apéndice H sección H.17

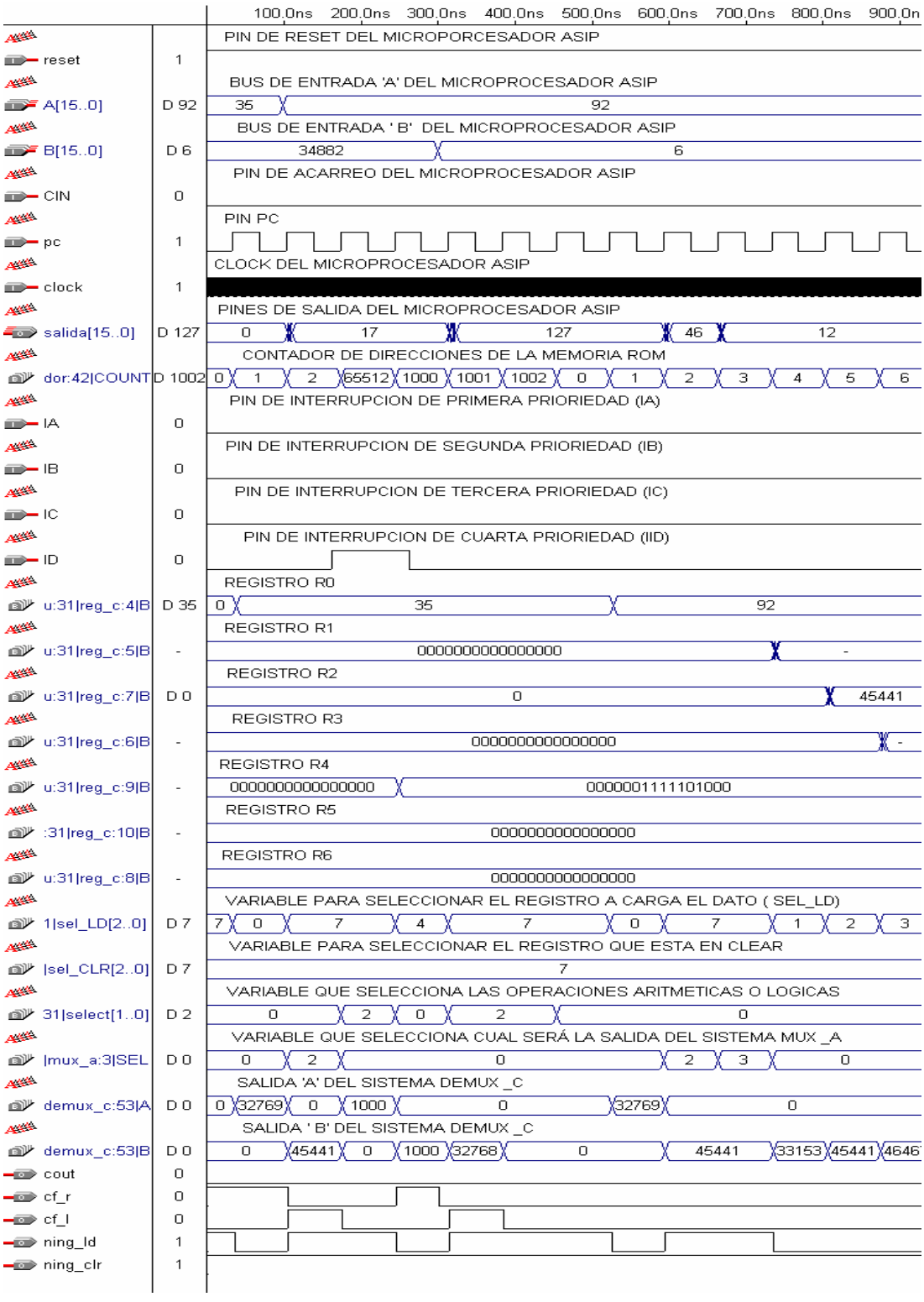


FIGURA 7.8: Simulación del programa test del ASIP cuando se produce una interrupción de cuarta o ultima prioridad[9]

[9] El programa TEST se encuentra en el apéndice H sección H.17

Cuando el subsistema CONTADOR apunta hacia la dirección ‘65512’ y se coloca en dicha dirección el subsistema ROM _R dentro del cual se encuentra la instrucción SALTA_1000 el cual nos dice que el subsistema ROM _R saltará hacia la dirección 1000 como se muestra en la figura 7.8.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘1000’ coloco en dicha dirección la instrucción GOTO_1000 con el cual se recibe todos los datos de la interrupción efectuada.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘1001’ ejecuto el instrucción SUMA A , R0 el cual efectúa la operación aritmética suma entre la entrada ‘A’ y el dato que contenga el subsistema REG _C enumerado como R0. En la simulación la entrada ‘A’ posee un valor de 92 y el subsistema REG _C enumerado como R0 tiene almacenado un valor de 35 lo cual hace que la salida tenga un valor de 127.

Cuando el subsistema CONTADOR apunta hacia la dirección ‘1001’ ejecuto el instrucción SALTA_0 con lo cual la dirección del subsistema ROM _R es cero .

CAPITULO VIII :
REPORTE DE LA GRABACION DEL PROGRAMA TEST DEL ASIP DENTRO DEL FLEX DE ALTERA

En este capitulo se muestra el reporte del dispositivo en donde se realiza la compilación y síntesis del programa del ASIP , en este reporte encontraremos como se definen los pines del dispositivo para el programa del ASIP expuesto a lo largo de esta tesis.

8.1 Definición de los pines del ASIP

Una vez compilado y sintetizado el programa del ASIP el software MAXPLUS II elige la mejor opción para la ubicación de las entradas y salidas en los pines del dispositivo (FLEX 10K). A continuación definiremos algunos parámetros del ASIP diseñado que nos da la compilación del programa TEST el cual esta dentro del subsistema ROM_R.

TABLA N^0 8.1 Definición de los pines del dispositivo EPF10K20TC144-3 (FLEX10K) cuando se graba el programa del ASIP

N.C. = No Connect. This pin has no internal connection to the device. VCCINT = Dedicated power pin, which MUST be connected to VCC (5.0 volts). VCCIO = Dedicated power pin, which MUST be connected to VCC (5.0 volts). GNDINT = Dedicated ground pin or unused dedicated input, which MUST be connected to GND. GNDIO = Dedicated ground pin, which MUST be connected to GND. RESERVED = Unused I/O pin, which MUST be left unconnected. CHIP "microprocesador" ASSIGNED TO AN EPF10K20TC144-3

TCK	: 1
CONF_DONE	: 2
nCEO	: 3
TDO	: 4
VCCIO	: 5
VCCINT	: 6
RESERVED	: 7

RESERVED	: 8
salida2	: 9
salida0	: 10
A1	: 11
cout	: 12
A14	: 13
B12	: 14
GNDIO	: 15
GNDINT	: 16
A3	: 17
A15	: 18
B15	: 19
B3	: 20
salida6	: 21
A8	: 22
A2	: 23
VCCIO	: 24
VCCINT	: 25
RESERVED	: 26
RESERVED	: 27
RESERVED	: 28
RESERVED	: 29
A13	: 30
B4	: 31
B10	: 32
B7	: 33
TMS	: 34
nSTATUS	: 35
B0	: 36
A10	: 37
A9	: 38
RESERVED	: 39

GNDIO	: 40
RESERVED	: 41
A11	: 42
RESERVED	: 43
A5	: 44
VCCIO	: 45
salida8	: 46
RESERVED	: 47
A7	: 48
RESERVED	: 49
GNDIO	: 50
RESERVED	: 51
VCCINT	: 52
VCCINT	: 53
reset	: 54
pc	: 55
CIN	: 56
GNDINT	: 57
GNDINT	: 58
RESERVED	: 59
RESERVED	: 60
VCCIO	: 61
RESERVED	: 62
B11	: 63
RESERVED	: 64
RESERVED	: 65
GNDIO	: 66
cf_r	: 67
salida15	: 68
RESERVED	: 69
RESERVED	: 70
VCCIO	: 71

A12	: 72
RESERVED	: 73
nCONFIG	: 74
VCCINT	: 75
MSEL1	: 76
MSEL0	: 77
B5	: 78
salida9	: 79
A6	: 80
A0	: 81
cf_l	: 82
RESERVED	: 83
GNDINT	: 84
GNDIO	: 85
RESERVED	: 86
RESERVED	: 87
B6	: 88
salida11	: 89
salida14	: 90
B1	: 91
salida12	: 92
VCCINT	: 93
VCCIO	: 94
salida10	: 95
salida4	: 96
salida5	: 97
salida13	: 98
salida1	: 99
IA	: 100
ning_clr	: 101
RESERVED	: 102
GNDINT	: 103

GNDIO	: 104
TDI	: 105
nCE	: 106
DCLK	: 107
DATA0	: 108
RESERVED	: 109
RESERVED	: 110
B14	: 111
A4	: 112
ning_ld	: 113
B2	: 114
VCCIO	: 115
RESERVED	: 116
salida3	: 117
RESERVED	: 118
RESERVED	: 119
RESERVED	: 120
RESERVED	: 121
RESERVED	: 122
VCCINT	: 123
IC	: 124
clock	: 125
ID	: 126
GNDINT	: 127
RESERVED	: 128
GNDIO	: 129
B8	: 130
B9	: 131
RESERVED	: 132
RESERVED	: 133
VCCIO	: 134
RESERVED	: 135

B13	: 136
RESERVED	: 137
RESERVED	: 138
GNDIO	: 139
RESERVED	: 140
RESERVED	: 141
RESERVED	: 142
salida7	: 143
IB	: 144

En la tabla 8.1 definimos los pines del dispositivo EPF10K20TC144-3 cuando se realiza la grabación del programa del ASIP y en la figura 8.1 se observa el esquema o grafico completo del dispositivo mencionado anteriormente y como quedan ordenados sus pines.

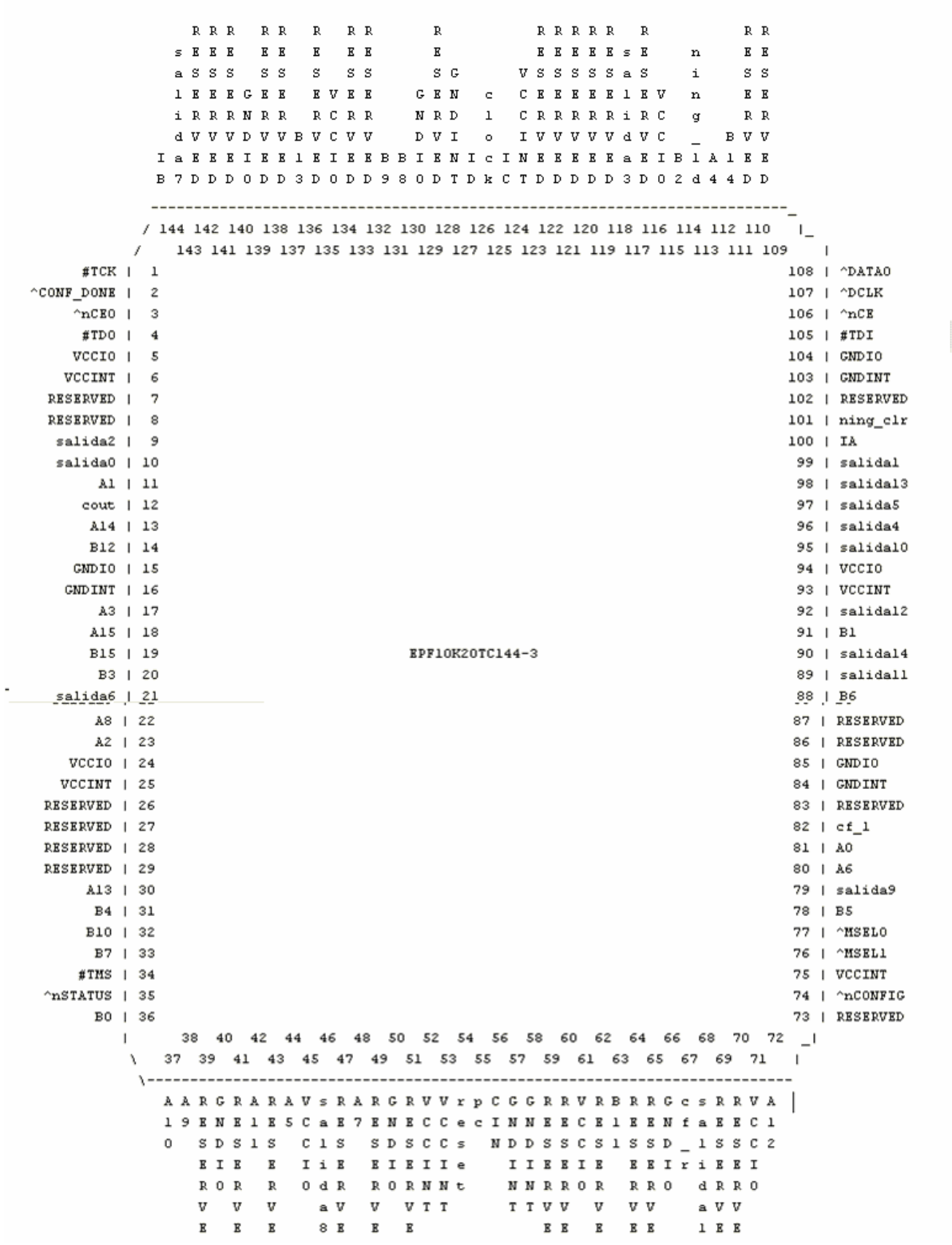


Fig.8.1 Definición de los pines del dispositivo EPF10K20TC144-3 (FLEX 10K)

8.2 Recursos usados en el dispositivo de Altera

En la tabla 8.2 veremos como se encuentra la arquitectura interna del dispositivo EPF10K20TC144-3 (FLEX 10K) de ALTERA

TABLA N^0 8.2 Recursos usados en el dispositivo de Altera EPF10K20TC144-3 (FLEX 10K) para grabar el programa del ASIP

LOGIC	COLUMN		ROW			
ARRAY	Interconnect		Interconnect			
BLOCK	LOGIC CELL	DRIVEN	DRIVEN	CLOCKS	PRESETS	INTERCONNECT
A1	8/ 8(100%)	2/ 8(25%)	7/ 8(87%)	0/2	0/2	9/22(40%)
A2	8/ 8(100%)	1/ 8(12%)	3/ 8(37%)	0/2	0/2	19/22(86%)
A3	8/ 8(100%)	3/ 8(37%)	1/ 8(12%)	0/2	0/2	15/22(68%)
A4	8/ 8(100%)	2/ 8(25%)	4/ 8(50%)	0/2	0/2	12/22(54%)
A5	8/ 8(100%)	1/ 8(12%)	6/ 8(75%)	0/2	0/2	11/22(50%)
A6	8/ 8(100%)	3/ 8(37%)	4/ 8(50%)	1/2	1/2	14/22(63%)
A7	8/ 8(100%)	0/ 8(0%)	5/ 8(62%)	0/2	0/2	13/22(59%)
A8	8/ 8(100%)	2/ 8(25%)	2/ 8(25%)	2/2	1/2	16/22(72%)
A9	8/ 8(100%)	3/ 8(37%)	1/ 8(12%)	0/2	0/2	17/22(77%)
A10	8/ 8(100%)	1/ 8(12%)	7/ 8(87%)	0/2	0/2	10/22(45%)
A11	8/ 8(100%)	3/ 8(37%)	0/ 8(0%)	0/2	0/2	11/22(50%)
A12	8/ 8(100%)	2/ 8(25%)	3/ 8(37%)	1/2	0/2	16/22(72%)
A13	7/ 8(87%)	2/ 8(25%)	2/ 8(25%)	2/2	1/2	14/22(63%)
A14	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	1/2	1/2	7/22(31%)
A15	7/ 8(87%)	1/ 8(12%)	4/ 8(50%)	2/2	1/2	10/22(45%)
A16	8/ 8(100%)	3/ 8(37%)	3/ 8(37%)	1/2	1/2	8/22(36%)
A17	8/ 8(100%)	3/ 8(37%)	3/ 8(37%)	2/2	1/2	18/22(81%)
A18	7/ 8(87%)	2/ 8(25%)	2/ 8(25%)	1/2	0/2	13/22(59%)
A19	8/ 8(100%)	1/ 8(12%)	4/ 8(50%)	1/2	1/2	9/22(40%)
A20	8/ 8(100%)	1/ 8(12%)	6/ 8(75%)	0/2	0/2	10/22(45%)
A21	7/ 8(87%)	5/ 8(62%)	7/ 8(87%)	0/2	0/2	5/22(22%)
A22	8/ 8(100%)	1/ 8(12%)	6/ 8(75%)	0/2	0/2	10/22(45%)
A23	8/ 8(100%)	2/ 8(25%)	6/ 8(75%)	0/2	0/2	9/22(40%)
A24	5/ 8(62%)	2/ 8(25%)	3/ 8(37%)	2/2	1/2	13/22(59%)
B1	8/ 8(100%	7/ 8(87%)	0/ 8(0%)	0/2	0/2	7/22(31%)
B2	7/ 8(87%)	1/ 8(12%)	2/ 8(25%)	1/2	0/2	16/22(72%)
B3	8/ 8(100%	4/ 8(50%)	1/ 8(12%)	1/2	0/2	5/22(22%)
B4	7/ 8(87%)	1/ 8(12%)	1/ 8(12%)	1/2	0/2	14/22(63%)
B6	8/ 8(100%)	4/ 8(50%)	1/ 8(12%)	1/2	0/2	16/22(72%)

B7	7/ 8(87%)	0/ 8(0%)	3/ 8(37%)	0/2	0/2	14/22(63%)
B8	8/ 8(100%)	7/ 8(87%)	0/ 8(0%)	1/2	0/2	13/22(59%)
B10	8/ 8(100%)	1/ 8(12%)	1/ 8(12%)	1/2	0/2	16/22(72%)
B11	8/ 8(100%)	9/8(112%)	0/ 8(0%)	1/2	0/2	5/22(22%)
B12	8/ 8(100%)	1/ 8(12%)	6/ 8(75%)	1/2	0/2	16/22(72%)
B13	8/ 8(100%)	3/ 8(37%)	4/ 8(50%)	0/2	0/2	14/22(63%)
B14	8/ 8(100%)	3/ 8(37%)	7/ 8(87%)	0/2	0/2	7/22(31%)
B15	8/ 8(100%)	4/ 8(50%)	6/ 8(75%)	0/2	0/2	13/22(59%)
B16	8/ 8(100%)	4/ 8(50%)	4/ 8(50%)	0/2	0/2	11/22(50%)
B17	8/ 8(100%)	4/ 8(50%)	0/ 8(0%)	1/2	0/2	11/22(50%)
B18	8/ 8(100%)	3/ 8(37%)	8/ 8(100%)	0/2	0/2	8/22(36%)
B19	8/ 8(100%)	4/ 8(50%)	7/ 8(87%)	0/2	0/2	11/22(50%)
B20	8/ 8(100%)	2/ 8(25%)	5/ 8(62%)	0/2	0/2	14/22(63%)
B21	8/ 8(100%)	4/ 8(50%)	3/ 8(37%)	2/2	1/2	12/22(54%)
B22	7/ 8(87%)	4/ 8(50%)	5/ 8(62%)	0/2	0/2	11/22(50%)
B23	8/ 8(100%)	4/ 8(50%)	7/ 8(87%)	0/2	0/2	13/22(59%)
B24	8/ 8(100%)	1/ 8(12%)	6/ 8(75%)	0/2	0/2	7/22(31%)
C1	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	1/2	0/2	11/22(50%)
C2	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	1/2	0/2	11/22(50%)
C3	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	1/2	0/2	12/22(54%)
C4	8/ 8(100%)	2/ 8(25%)	5/ 8(62%)	1/2	0/2	12/22(54%)
C5	8/ 8(100%)	2/ 8(25%)	1/ 8(12%)	1/2	0/2	15/22(68%)
C6	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	1/2	0/2	14/22(63%)
C7	8/ 8(100%)	2/ 8(25%)	1/ 8(12%)	1/2	0/2	15/22(68%)
C8	8/ 8(100%)	2/ 8(25%)	2/ 8(25%)	1/2	0/2	14/22(63%)
C9	7/ 8(87%)	0/ 8(0%)	0/ 8(0%)	1/2	0/2	13/22(59%)
C10	8/ 8(100%)	1/ 8(12%)	1/ 8(12%)	1/2	0/2	12/22(54%)
C12	8/ 8(100%)	2/ 8(25%)	2/ 8(25%)	1/2	0/2	17/22(77%)
C13	8/ 8(100%)	0/ 8(0%)	0/ 8(0%)	0/2	0/2	8/22(36%)
C14	7/ 8(87%)	0/ 8(0%)	0/ 8(0%)	1/2	0/2	11/22(50%)
C15	8/ 8(100%)	0/ 8(0%)	0/ 8(0%)	1/2	0/2	13/22(59%)
C16	8/ 8(100%)	0/ 8(0%)	0/ 8(0%)	1/2	0/2	15/22(68%)
C17	8/ 8(100%)	0/ 8(0%)	0/ 8(0%)	1/2	0/2	13/22(59%)
C18	8/ 8(100%)	1/ 8(12%)	1/ 8(12%)	1/2	0/2	13/22(59%)
C19	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	1/2	0/2	11/22(50%)
C20	8/ 8(100%)	0/ 8(0%)	1/ 8(12%)	½	0/2	14/22(63%)
C21	8/ 8(100%)	1/ 8(12%)	3/ 8(37%)	½	0/2	15/22(68%)
C22	8/ 8(100%)	0/ 8(0%)	4/ 8(50%)	0/2	0/2	11/22(50%)
C23	6/ 8(75%)	0/ 8(0%)	3/ 8(37%)	½	0/2	12/22(54%)
C24	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	½	0/2	13/22(59%)
D1	8/ 8(100%)	2/ 8(25%)	0/ 8(0%)	½	0/2	14/22(63%)

D3	8/ 8(100%)	2/ 8(25%)	2/ 8(25%)	½	0/2	7/22(31%)
D4	6/ 8(75%)	1/ 8(12%)	2/ 8(25%)	½	0/2	10/22(45%)
D5	7/ 8(87%)	0/ 8(0%)	2/ 8(25%)	½	0/2	9/22(40%)
D6	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	½	0/2	12/22(54%)
D7	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	½	0/2	14/22(63%)
D8	8/ 8(100%)	2/ 8(25%)	0/ 8(0%)	½	0/2	14/22(63%)
D9	8/ 8(100%)	1/ 8(12%)	3/ 8(37%)	½	0/2	13/22(59%)
D10	6/ 8(75%)	0/ 8(0%)	2/ 8(25%)	½	0/2	9/22(40%)
D13	6/ 8(75%)	0/ 8(0%)	4/ 8(50%)	½	0/2	11/22(50%)
D14	3/ 8(37%)	1/ 8(12%)	2/ 8(25%)	½	0/2	4/22(18%)
D15	8/ 8(100%)	2/ 8(25%)	1/ 8(12%)	½	0/2	14/22(63%)
D16	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	½	0/2	12/22(54%)
D17	8/ 8(100%)	3/ 8(37%)	1/ 8(12%)	2/2	0/2	14/22(63%)
D18	8/ 8(100%)	0/ 8(0%)	4/ 8(50%)	½	0/2	7/22(31%)
D19	8/ 8(100%)	2/ 8(25%)	2/ 8(25%)	½	0/2	14/22(63%)
D21	8/ 8(100%)	1/ 8(12%)	3/ 8(37%)	½	0/2	15/22(68%)
D22	8/ 8(100%)	1/ 8(12%)	1/ 8(12%)	½	0/2	16/22(72%)
D23	8/ 8(100%)	1/ 8(12%)	4/ 8(50%)	½	0/2	11/22(50%)
D24	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	½	0/2	12/22(54%)
E1	8/ 8(100%)	1/ 8(12%)	0/ 8(0%)	½	0/2	13/22(59%)
E2	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	½	0/2	13/22(59%)
E3	8/ 8(100%)	0/ 8(0%)	6/ 8(75%)	0/2	0/2	10/22(45%)
E4	7/ 8(87%)	1/ 8(12%)	1/ 8(12%)	½	0/2	13/22(59%)
E5	8/ 8(100%)	1/ 8(12%)	1/ 8(12%)	½	0/2	12/22(54%)
E6	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	½	0/2	14/22(63%)
E7	8/ 8(100%)	1/ 8(12%)	1/ 8(12%)	½	0/2	12/22(54%)
E8	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	½	0/2	11/22(50%)
E9	8/ 8(100%)	0/ 8(0%)	4/ 8(50%)	½	0/2	12/22(54%)
E10	8/ 8(100%)	0/ 8(0%)	4/ 8(50%)	½	0/2	10/22(45%)
E11	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	½	0/2	8/22(36%)
E12	5/ 8(62%)	0/ 8(0%)	3/ 8(37%)	½	0/2	8/22(36%)
E13	8/ 8(100%)	0/ 8(0%)	4/ 8(50%)	0/2	0/2	10/22(45%)
E14	8/ 8(100%)	1/ 8(12%)	3/ 8(37%)	0/2	0/2	10/22(45%)
E15	8/ 8(100%)	2/ 8(25%)	3/ 8(37%)	½	0/2	13/22(59%)
E16	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	½	0/2	13/22(59%)
E17	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	½	0/2	13/22(59%)
E18	6/ 8(75%)	2/ 8(25%)	1/ 8(12%)	½	0/2	12/22(54%)
E19	8/ 8(100%)	2/ 8(25%)	2/ 8(25%)	0/2	0/2	10/22(45%)
E20	8/ 8(100%)	1/ 8(12%)	0/ 8(0%)	½	0/2	13/22(59%)
E21	2/ 8(25%)	1/ 8(12%)	1/ 8(12%)	0/2	0/2	3/22(13%)
E22	8/ 8(100%)	1/ 8(12%)	3/ 8(37%)	0/2	0/2	10/22(45%)

E23	8/ 8(100%)	2/ 8(25%)	2/ 8(25%)	0/2	0/2	10/22(45%)
E24	8/ 8(100%)	0/ 8(0%)	4/ 8(50%)	0/2	0/2	10/22(45%)
F1	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	½	0/2	11/22(50%)
F2	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	½	0/2	11/22(50%)
F3	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	½	0/2	12/22(54%)
F4	8/ 8(100%)	1/ 8(12%)	1/ 8(12%)	½	0/2	14/22(63%)
F5	8/ 8(100%)	2/ 8(25%)	0/ 8(0%)	½	0/2	14/22(63%)
F6	8/ 8(100%)	2/ 8(25%)	0/ 8(0%)	½	0/2	9/22(40%)
F8	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	½	0/2	11/22(50%)
F9	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	½	0/2	11/22(50%)
F10	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	½	0/2	11/22(50%)
F11	8/ 8(100%)	0/ 8(0%)	4/ 8(50%)	½	0/2	11/22(50%)
F13	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	½	0/2	11/22(50%)
F14	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	1/3	0/2	11/22(50%)
F15	7/ 8(87%)	0/ 8(0%)	2/ 8(25%)	¼	0/2	11/22(50%)
F16	7/ 8(87%)	3/ 8(37%)	0/ 8(0%)	1/5	0/2	14/22(63%)
F17	8/ 8(100%)	0/ 8(0%)	2/ 8(25%)	1/6	0/2	12/22(54%)
F18	3/ 8(37%)	3/ 8(37%)	0/ 8(0%)	1/7	0/2	7/22(31%)
F19	8/ 8(100%)	2/ 8(25%)	0/ 8(0%)	1/8	0/2	14/22(63%)
F20	8/ 8(100%)	0/ 8(0%)	3/ 8(37%)	1/9	0/2	12/22(54%)
F22	8/ 8(100%)	1/ 8(12%)	2/ 8(25%)	1/10	0/2	15/22(68%)
F23	8/ 8(100%)	2/ 8(25%)	2/ 8(25%)	1/11	0/2	14/22(63%)
F24	8/ 8(100%)	3/ 8(37%)	2/ 8(25%)	1/12	0/2	13/22(59%)

Total dedicated input pins used:	6/6	(100%)
Total I/O pins used:	55/96	(57%)
Total logic cells used:	1024/1152	(88%)
Total embedded cells used:	0/48	(0%)
Total EABs used:	0/6	(0%)
Average fan-in:	3.28/4	(82%)
Total fan-in:	3365/4608	(73%)
Total input pins required:		40
Total input I/O cell registers required:		0
Total output pins required:		21
Total output I/O cell registers required:		0
Total buried I/O cell registers required:		0
Total bidirectional pins required:		0

Total reserved pins required	0
Total logic cells required:	1024
Total flipflops required:	295
Total packed registers required:	0
Total logic cells in carry chains:	0
Total number of carry chains:	0
Total logic cells in cascade chains:	0
Total number of cascade chains:	0
Total single-pin Clock Enables required:	0
Total single-pin Output Enables required:	0
Synthesized logic cells:	112/1152 (9%)

TABLA N^0 8.3 Celdas lógicas usadas en el dispositivo de altera EPF10K20TC144-3 (FLEX 10K) para grabar el programa del ASIP

Column:	01	02	03	04	05	06	07	08	09	10	11	12	EA	13	14	15	16	17	18	19	20	21	22	23	24	Total(LC/EC)
A:	8	8	8	8	8	8	8	8	8	8	8	8	0	7	8	7	8	8	7	8	8	7	8	8	5	185/0
B:	8	7	8	7	0	8	7	8	0	8	8	8	0	8	8	8	8	8	8	8	8	8	7	8	8	172/0
C:	8	8	8	8	8	8	8	8	7	8	0	8	0	8	7	8	8	8	8	8	8	8	8	6	8	180/0
D:	8	0	8	6	7	8	8	8	8	6	0	0	0	6	3	8	8	8	8	8	0	8	8	8	8	148/0
E:	8	8	8	7	8	8	8	8	8	8	8	5	0	8	8	8	8	8	6	8	8	2	8	8	8	180/0
F:	8	8	8	8	8	6	0	8	8	8	8	0	0	8	8	7	7	8	3	8	8	0	8	8	8	159/0
Total:	48	39	48	44	39	46	39	48	39	46	32	29	0	45	42	46	47	48	40	48	40	33	47	46	45	1024/0

8.3 Entradas del ASIP en el dispositivo de Altera

En la tabla 8.4 se observa todas las entradas registradas por el software MAX + PLUS II en el dispositivo **EPF10K20TC144-3 (FLEX)** cuando se graba el programa del ASIP.

TABLA N^0 8.4:Entradas usadas en el dispositivo de altera EPF10K20TC144-3 (FLEX 10K) para grabar el programa del ASIP

							Fan -In		Fan - Out	
PIN	LC	EC	ROW	Col	Primitive	INP	FBK	OUT	FBK	NAME
81	-	-	F	--	INPUT	0	0	0	1	A0
11	-	-	C	--	INPUT	0	0	0	1	A1
23	-	-	D	--	INPUT	0	0	0	1	A2
17	-	-	D	--	INPUT	0	0	0	1	A3
112	-	-	-	2	INPUT	0	0	0	1	A4
44	-	-	-	18	INPUT	0	0	0	1	A5
80	-	-	F	--	INPUT	0	0	0	1	A6
48	-	-	-	15	INPUT	0	0	0	1	A7
22	-	-	D	--	INPUT	0	0	0	1	A8
38	-	-	-	22	INPUT	0	0	0	1	A9
37	-	-	-	23	INPUT	0	0	0	1	A10
42	-	-	-	19	INPUT	0	0	0	1	A11
72	-	-	-	3	INPUT	0	0	0	1	A12
30	-	-	F	--	INPUT	0	0	0	1	A13
13	-	-	C	--	INPUT	0	0	0	1	A14
18	-	-	D	--	INPUT	0	0	0	1	A15
36	-	-	-	24	INPUT	0	0	0	1	B0
91	-	-	C	--	INPUT	0	0	0	1	B1
144	-	-	-	4	INPUT	0	0	0	1	B2
20	-	-	D	--	INPUT	0	0	0	1	B3
31	-	-	F	--	INPUT	0	0	0	1	B4
78	-	-	F	--	INPUT	0	0	0	1	B5
88	-	-	D	--	INPUT	0	0	0	1	B6
33	-	-	F	--	INPUT	0	0	0	1	B7
130	-	-	-	15	INPUT	0	0	0	1	B8
131	-	-	-	16	INPUT	0	0	0	1	B9
32	-	-	F	--	INPUT	0	0	0	1	B10

63	-	-	-	10	INPUT	0	0	0	1	B11
14	-	-	C	--	INPUT	0	0	0	1	B12
136	-	-	-	20	INPUT	0	0	0	1	B13
111	-	-	-	1	INPUT	0	0	0	1	B14
19	-	-	D	--	INPUT	0	0	0	1	B15
56	-	-	-	--	INPUT	0	0	0	11	CIN
125	-	-	-	--	INPUT G	0	0	0	0	clock
100	-	-	A	--	INPUT	0	0	0	5	IA
144	-	-	A	--	INPUT	0	0	0	5	IB
124	-	-	-	--	INPUT	0	0	0	3	IC
126	-	-	-	--	INPUT	0	0	0	2	ID
55	-	-	-	--	INPUT G	0	0	0	0	PC
54	-	-	-	--	INPUT G	0	0	0	0	RESET

Code:

s = Synthesized pin or logic cell

+ = Synchronous flipflop

/ = Slow slew-rate output

! = NOT gate push-back

r = Fitter-inserted logic cell

@ = Uses single-pin Clock Enable

& = Uses single-pin Output Enable

G = Global Source. Fan-out destinations counted here do not include destinations that are driven using global routing resources. Refer to the Auto Global Signals, Clock Signals, Clear Signals, Synchronous Load Signals, and Synchronous Clear Signals Sections of this Report File for information on which signals' fan-outs are used as Clock, Clear, Preset, Output Enable, and synchronous Load signals.

8.4 Salidas del ASIP en el dispositivo de Altera

En la tabla 8.5 se observa todas las salidas registradas por el software MAX + PLUS II en el dispositivo **EPF10K20TC144-3 (FLEX)** cuando se graba el programa del ASIP.

TABLA N^0 8.5:Salidas usadas en el dispositivo de Altera EPF10K20TC144-3 (FLEX 10K) para grabar el programa del ASIP

						Fan- in		Fan out		
	Fed By	Fed By				Code		code		
PIN	LC	EC	ROW	COL	Primitive	INP	FBK	OUT	FBK	NAME
82	-	-	E	--	OUTPUT	0	1	0	0	cf_l
67	-	-	-	8	OUTPUT	0	1	0	0	cf_r
12	-	-	C	--	OUTPUT	0	1	0	0	cout
101	-	-	A	--	OUTPUT	0	1	0	0	ning_clr
113	-	-	-	3	OUTPUT	0	1	0	0	ning_ld
10	-	-	B	--	OUTPUT	0	1	0	0	salida0
99	-	-	B	--	OUTPUT	0	1	0	0	salida1
9	-	-	B	--	OUTPUT	0	1	0	0	salida2
117	-	-	-	5	OUTPUT	0	1	0	0	salida3
96	-	-	B	--	OUTPUT	0	1	0	0	salida4
97	-	-	B	--	OUTPUT	0	1	0	0	salida5
21	-	-	D	--	OUTPUT	0	1	0	0	salida6
143	-	-	A	--	OUTPUT	0	1	0	0	salida7
46	-	-	-	17	OUTPUT	0	1	0	0	salida8
79	-	-	F	--	OUTPUT	0	1	0	0	salida9
95	-	-	B	--	OUTPUT	0	1	0	0	salida10
89	-	-	C	--	OUTPUT	0	1	0	0	salida11
92	-	-	C	--	OUTPUT	0	1	0	0	salida12
98	-	-	B	--	OUTPUT	0	1	0	0	salida13
90	-	-	C	--	OUTPUT	0	1	0	0	salida14
68	-	-	-	7	OUTPUT	0	1	0	0	salida15

Code:

s = Synthesized pin or logic cell

+ = Synchronous flipflop

/ = Slow slew-rate output

! = NOT gate push-back

r = Fitter-inserted logic cell

@ = Uses single-pin Clock Enable

& = Uses single-pin Output Enable

8.5 Utilización de la interconexión Fasttrack del ASIP dentro del dispositivo de Altera

En la tabla 8.6 y 8.7 se observa todas las interconexiones entre subsistemas , y en que celdas del dispositivo han sido guardadas dicha información.

TABLA N^0 8.6:Filas del Fastrack usado en el dispositivo de Altera EPF10K20TC144-3 (FLEX 10K) para grabar el programa del ASIP

	Global FastTrack	Left Half-FastTrack	Right Half-FastTrack			
Row	Interconnect	Interconnect	Interconnect	Input Pins	Output Pins	Bidir Pins
A:	78/ 96(81%)	27/ 48(56%)	29/ 48(60%)	2/16(12%)	2/16(12%)	0/16(0%)
B:	83/ 96(86%)	24/ 48(50%)	26/ 48(54%)	0/16(0%)	7/16(43%)	0/16(0%)
C:	65/ 96(67%)	34/ 48(70%)	33/ 48(68%)	4/16(25%)	4/16(25%)	0/16(0%)
D:	53/ 96(55%)	18/ 48(37%)	36/ 48(75%)	7/16(43%)	1/16(6%)	0/16(0%)
E:	39/ 96(40%)	30/ 48(62%)	35/ 48(72%)	0/16(0%)	1/16(6%)	0/16(0%)
F:	50/ 96(52%)	23/ 48(47%)	25/ 48(52%)	7/16(43%)	1/16(6%)	0/16(0%)

TABLA N^0 8.7:Columnas del Fastrack usado en el dispositivo de Altera EPF10K20TC144-3 (FLEX 10K) para grabar el programa del ASIP

Column	Interconnect	Input Pins	Output Pins	Bidir Pins
1	13/24(54%)	¼(25%)	0/4(0%)	0/4(0%)
2	5/24(20%)	¼(25%)	0/4(0%)	0/4(0%)
3	10/24(41%)	¼(25%)	1/4(0%)	0/4(0%)
4	9/24(37%)	¼(25%)	0/4(0%)	0/4(0%)
5	10/24(41%)	¼(25%)	1/4(25%)	0/4(0%)
6	6/24(25%)	0/4(0%)	0/4(0%)	0/4(0%)
7	8/24(33%)	0/4(0%)	1/4(25%)	0/4(0%)
8	10/24(41%)	0/4(0%)	1/4(25%)	0/4(0%)

9	4/24(16%)	0/4(0%)	0/4(0%)	0/4(0%)
10	4/24(16%)	¼(25%)	0/4(0%)	0/4(0%)
11	9/24(37%)	0/4(0%)	0/4(0%)	0/4(0%)
12	8/24(33%)	0/4(0%)	0/4(0%)	0/4(0%)
13	5/24(20%)	0/4(0%)	0/4(0%)	0/4(0%)
14	6/24(25%)	0/4(0%)	0/4(0%)	0/4(0%)
15	9/24(37%)	2/4(50%)	0/4(0%)	0/4(0%)
16	14/24(58%)	¼(25%)	0/4(0%)	0/4(0%)
17	12/24(50%)	0/4(0%)	1/4(25%)	0/4(0%)
18	11/24(45%)	¼(25%)	0/4(0%)	0/4(0%)
19	8/24(33%)	¼(25%)	0/4(0%)	0/4(0%)
20	9/24(37%)	¼(25%)	0/4(0%)	0/4(0%)
21	9/24(37%)	0/4(0%)	0/4(0%)	0/4(0%)
22	12/24(50%)	¼(25%)	0/4(0%)	0/4(0%)
23	9/24(37%)	¼(25%)	0/4(0%)	0/4(0%)
24	10/24(41%)	¼(25%)	0/4(0%)	0/4(0%)
EA	0/24(0%)	0/4(0%)	0/4(0%)	0/4(0%)

8.6 Señales de clock del ASIP dentro del dispositivo de Altera

En la tabla 8.8 se definen todas los pines de CLOCK del ASIP y en que pines del dispositivo se encuentran .

TABLA N^0 8.8:Definición de los pines de Clock usado en el dispositivo de Altera EPF10K20TC144-3 (FLEX 10K) para grabar el programa del ASIP

Type	Fan-out	Name
INPUT	279	clock
INPUT	16	pc

8.7 Señales de Reset del ASIP dentro del dispositivo de Altera

En la tabla 8.9 se definen todas los pines de Reset del ASIP y en que pines del dispositivo se encuentran .

TABLA N^{0}8.9:Definición del Fan-Out del Clear usado en el dispositivo de Altera EPF10K20TC144-3 (FLEX 10K) para grabar el programa del ASIP

Type	Fan-out	Name
INPUT	16	reset

CONCLUSIONES

1._ El ASIP diseñado posee un pin CLOCK de 8ns , el cual posee una entrada PC de 100ns , este pin de entrada PC hace que el sistema contador varié la dirección de la memoria del ASIP (subsistema ROM _R) . En cada dirección de la memoria del ASIP se encuentra una instrucción , el cual controla todo el hardware del ASIP.

2._ El microprocesador ASIP se ha diseñado con 633 instrucciones las cuales están clasificadas en 18 tipos . Al simular el ASIP en el software MAX + PLUS II de la empresa ALTERA con un CLOCK de 8ns concluyo que el ASIP funciona adecuadamente y su respuesta electrónica es optima.

3._ El ASIP esta formado por los siguientes subsistemas:

- La memoria del ROM del ASIP (subsistema ROM _R).
- Un subsistema ADAPTADOR
- Un subsistema CONTADOR
- Un subsistema DEMUX
- Un subsistema INTERRUPCIONES
- Un subsistema C _LOGICO
- Un subsistema Unidad de Procesamiento
- Un subsistema REG _H

4._ Dentro del subsistema Unidad de Procesamiento se encuentra los siguientes subsistemas:

- Dos subsistemas DECODER
- Siete subsistemas REG _C

- Dos subsistemas MUX _B
- Un subsistema ALU
- Un subsistema COMPARADOR
- Un subsistema MUX _C

5._ Dentro del subsistema ALU se encuentran los siguientes subsistemas:

- Un subsistema ARITHMETIC
- Un subsistema LOGIC
- Un subsistema ASHL
- Un subsistema ASHL
- Un subsistema MUX _A

6._Todos los subsistemas mencionados en 3 , 4 , 5 han sido simulados con el software MAX + PLUS II de la empresa ALTERA y muestran una respuesta electrónica optima y satisfactoria.

7._Cada uno de estos subsistemas han sido descritos en el capitulo I y han sido diseñados y simulados en los capítulos II , III , IV , V , VI . En dichos capítulos también se encuentran el diagrama de flujo y la simulación de cada subsistema mencionado. Los programas en VHDL de cada uno de los subsistemas del ASIP se muestran en el apéndice H.

8._ Después de realizar el diseño del ASIP , realizamos un programa TEST para el ASIP . En el programa TEST simulamos todos los recursos del ASIP y muestro los 18 tipos de instrucciones que puede realizar el ASIP, el programa TEST para el ASIP se encuentra en el subsistema ROM _R . En el capitulo VII explicamos cada ciclo de instrucción del programa TEST y lo simulamos en el software MAX + PLUS II de la empresa ALTERA.

9._ En el capitulo VIII de esta tesis se realiza el reporte de la grabación del programa en el ASIP , en este capitulo se observa el espacio de celdas del chip que se utilizaran para guardar el programa TEST dentro del hardware del ASIP , para compilar el programa TEST del ASIP utilizo un FLEX 10K , el cual nos proporciona el espacio suficiente para grabar el diseño. En este capitulo también muestro la distribución de pines del ASIP dentro del chip de ALTERA.

10._ Al realizar la simulación del programa TEST del ASIP observo que el programa compila sin ningún error y sin ninguna advertencia de sintaxis , con lo cual se concluye que es posible llevar este programa a un CHIP de la empresa ALTERA y poder utilizar el ASIP en un diseño particular.

ANEXOS

Usted puede acceder a los anexos consultando el formato impreso de la tesis.

BIBLIOGRAFIA

1. Jose A. Boluda Grau, “VHDL lenguaje para síntesis y modelado de circuitos” Dpto informática y electrónica de la universidad de Valencia
2. “Handbook de Actel HDL coding Style Guide”
3. M. Morris Mano, “Arquitectura de computadoras” universidad estatal de California en los Ángeles - tercera edición
4. GBarry B Brey ,”Los microprocesadores INTEL arquitectura y programación” - tercera edición
5. T.L. Floyd ,”Fundamentos de sistemas digitales” - sexta edición
6. Microprocessor Intel Corporation, “Microprocessor and Peripheral handbook volume 1”
7. Peter J. Ashenden ,”The VHDL Cookbook” - first edition
8. V. Carl Hamacher , Zvonko G. Vranesic, Safwat G Zaky , “Organización de Computadoras”- Segunda edición
9. Datasheet ByteBlasterMV Parallel Port Download Cable
10. Cypress Programmable Logic Data Book
11. Cypress Programmable Training for PLDs , CPLDs and FPGAs 1999
12. Skahill , K ,” VHDL for Programmable logic ”- 1998
13. Sudhakary ,”VHDL Starter’s Guide” - 1998
14. Messmer , Hans Peter “ the indispensable PC hardware book “
15. VHDL Manual de la Universidad Politécnica de Valencia
16. F. Pardo, J.A. Boluda. "VHDL: Lenguaje para síntesis y diseño de circuitos digitales". Ed. Rama -1999.
17. S. Olloz, E. Villar, Y. Torroja, L. Teres: "VHDL, lenguaje estándar de diseño electrónico" Ed. McGraw-Hill.

18._ www.altera.com
19._ www.cypress.com
20._ www.xilinx.com
21._ www.actel.com
22._ www.synopsis.com