Practica 1. Componentes generales del CPU

Paul Vazquez
Ingenieria en Sistemas Digitales y Robotica
Tecnologico de Monterrey
Laboratorio de Arquitectura de computadoras A00819877

Rodolfo Cuan
Ingenieria en Sistemas Digitales y Robotica
Tecnologico de Monterrey
Laboratorio de Arquitectura de computadoras A01233155

1. Introducción

El objetivo de esta practica es realizar seis distintos módulos: Adder, Shift Left 2 bits, Shift Left 32 bits, Program Counter, Sign Extender y un MUX. Estos módulos son sencillos y basicos, sin embargo tendrán una tarea importante en el desarrollo final del CPU MIPS.

Con esta practica también se refuerzan el dominio del lenguaje VHDL y la elaboracion de testbenches para probar los componentes.

2. Desarrollo

Los módulos fueron desarrollados en VHDL en la plataforma ISE design, integrada en la computadora virtual proporcionada. Igualmente los módulos fueron planeados siguiendo las especificaciones en el manual de practicas.

Para facilitar el desarrollo colaborativo de las practicas se opto por crear un repositorio git en el directorio del proyecto de ISE.

A continuación se presenta el desarrollo especifico de cada uno de los módulos.

2.1. Add

El modulo de Add tiene como funcionalidad realizar una suma sin signo. El modulo contiene como entrada 2 señales y 1 como salida (Fig. 1). Las 3 señales son de 32 bits. La librería de aritmética, STD_LOGIC_ARITH, se utilizo para sumar las dos entradas y la respuesta es puesta en la salida.

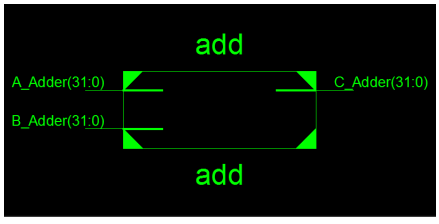

Figure 1: Esquemático RTL de Add.

2.2. Shift Left 2 bits para señales de 26 bits

Con este componente se recorre dos posiciones a la izquierda una entrada de 26 bits lo que da como resultado una salida de 28 bits. Se trata de una shift left logico, por lo que el signo de la entrada no se conserva en la salida. Para lograr esta funcionalidad se mapean directamente los 26 bits de la entrada (25:0) a los 26 bits mas significativos de la salida (27:2) y se agregan dos ceros en las primeras posiciones de la salida (1:0). Cuando se sintetiza la unidad queda el siguiente componente presentado en la Fig. 2.

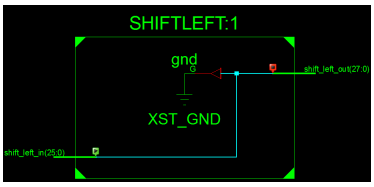

Figure 2: Esquematico RTL de Shift Left 26 bits.

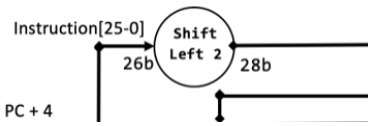

Figure 3: Diagrama de Shift Left en MIPS.

2.3. Shift Left 2 bit para señales de 32 bits

Este modulo contiene una entrada y una salida, su función es hacer un corrimiento de 2 bits hacia la derecha. La entrada y salida son del mismo tamaño, 32 bits (Fig. 4). El corrimiento fue creado concatenando 2 bits a la derecha de la entrada y guardar el resultado en una señal auxiliar de 34 bits. Después, los 32 bits menos significativos son mostrados como salida.

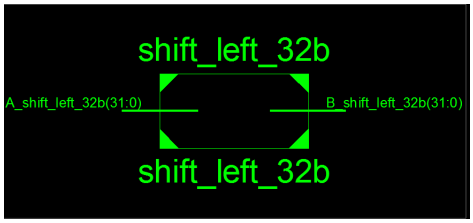

Figure 4: Esquematico RTL de Shift Left 2 bits para señales de 32 bits.

2.4. Program Counter

Con este componente se establece una implementación de un conjunto de flip flops D de transición negativa (32 bits) con reset asincronico, que nos servirá en el MIPS para conocer que instrucción de un programa ejecutar después. De acuerdo al diagrama del MIPS, a este componente se le suman 4 unidades para avanzar el programa, o puede ser modificado por instrucciones de jump, branch, etc... En Fig. 5 se observa el resultado de la sintetizacion del flip flop en RTL.

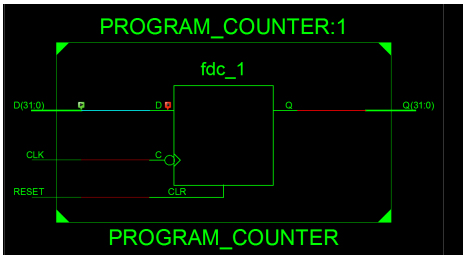

Figure 5: Esquemático RTL Program Counter.

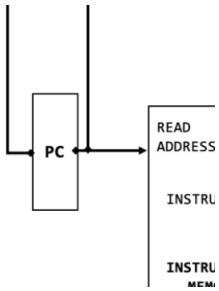

Figure 6: Diagrama Program Counter en MIPS.

2.5. Sign Extender

El modulo sign extender, como su nombre lo indica, tiene la funcionalidad de extender un numero respetado su signo. Como entrada se tiene una señal de 16 bits y se extiende a una de 32 bits (Fig. 7). Para respetar el signo se toma en consideración el bit mas significativo. Si este bit es 1 se tiene que concatenar a la izquierda 16 bits en 1, de lo contrario, 16 bits en 0. Esto ultimo es para respetar el signo en complemento a 2.

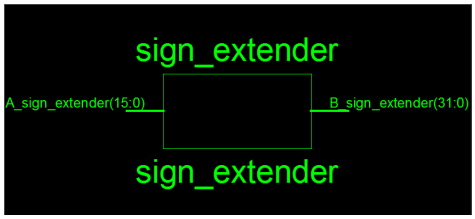

Figure 7: Esquemático RTL de Sign Extender.

2.6. Multiplexores 2 a 1

Se desarrollaron también dos multiplexores de dos entradas, con la particular diferencia que uno de 5 bits y el otro de 32 bits para cada entrada. Los multiplexores tienen la ventaja que nos permiten decidir entre dos posibles opciones con un solo bit de control, por lo que es conveniente para reducir la lógica de un circuito. Para generar ambos componentes se aprovecho el estatuto with select de vhdl (Listing 1), que internamente sintetiza multiplexores al usar esa instrucción, puramente concurrentes sin secciones secuenciales. Lo cual genero el esquemático presentado en la Fig. 8 Y 9.

```
with output select input <=
        "1000" when "00",
        "0100" when "01",
        "0010" when "10",
        "0001" when "11";
```

Listing 1: Mux VHDL code

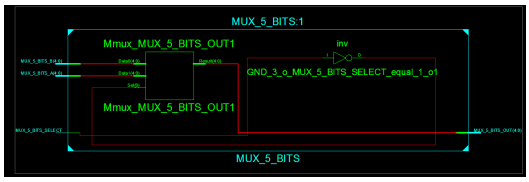

Figure 8: RTL Schematic MUX 5 bits.

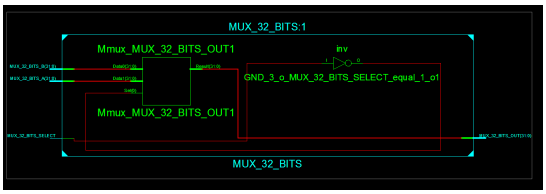

Figure 9: RTL Schematic MUX 32 bits.

3. Resultados y Discusión

Se utilizaron Test Benches para comprobar el funcionamiento correcto de cada uno de los módulos. Los Test Benches contienen casos extremos específicos para cada modulo. A continuación se presentan los resultados y las gráficas obtenidas.

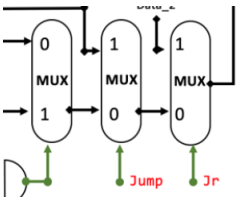

Figure 10: Diagrama multiplexores en MIPS.

3.1. Add

Tres casos diferentes fueron expuestos al modulo y son mostrados en Fig. 11. Como se puede observar en los primeros 2 casos (Fig. 11a y 11b) se obtienen resultados coherentes y correctos. En el tercer caso se obtiene una respuesta igualmente correcta, sin embargo, esta presenta overflow ya que la respuesta no cabe en el numero de bits.

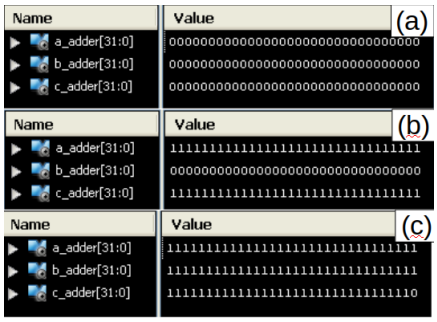

Figure 11: Resultados de modulo Add.

3.2. Shift Left para señales de 26 bits

Para comprobar el funcionamiento del modulo se probaron tres casos extremos:

1) Cuando se tiene un '1' ubicado en el bit mas significativo de la entrada y todos los demás bits en cero. De manera que la salida debe tener también el bit mas significativo encendido. (Fig. 12)
2) Cuando se tiene el bit menos significativo encendido en la entrada y todos los demás apagados. (Fig. 13)
3) Cuando se tiene el bit mas significativo y el bit menos significativo encendido en la entrada. (Fig. 14)

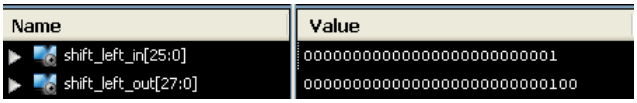

Figure 12: Resultados del Shift Left Case A.

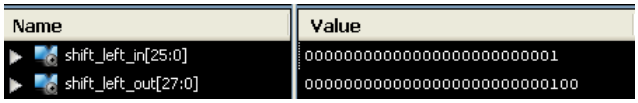

Figure 13: Resultados de Shift Left Case B.

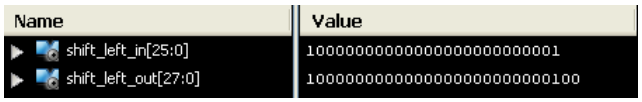

Figure 14: Resultados de Shift Left Case C.

3.3. Shift Left bit para señales de 32 bits

Cuatro diferentes numeros de 32 bits fueron probados por este modulo. En Fig. 15 se puede observar como en cada uno de los casos se hizo correctamente la corrida de dos bits hacia la izquierda.

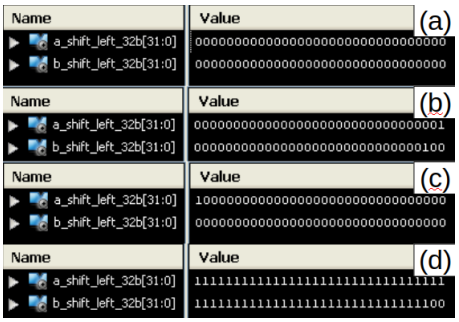

Figure 15: Resultados de modulo Shift Left de 32 bits.

3.4. Program Counter

para demostrar el funcionamiento de este componente se probo primero que respondiera correctamente a la entrada D en la transicion negativa del reloj, lo cual se puede observar en el siguiente TestBench, donde se ve que la salida refleja el valor de 562 en decimal en el momento adecuado (Fig. 16).

También se comprobó que funcionara correctamente el reset asincronico por lo que se observa que la salida refleja un 0 aun cuando la entrada cambio a 1365, debido a que esta en 1 el bit de reset (Fig. 17).

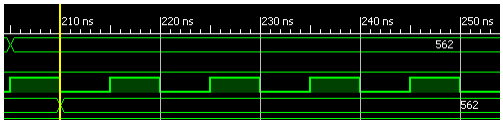

Figure 16: Program Counter Transicion negativa del reloj

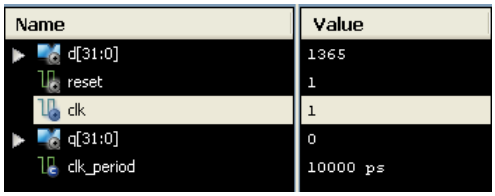

Figure 17: Program Counter Reset Asincronico

3.5. Sign Extender

Se puso aprueba el modulo ingresando 4 diferentes valores. Estos valores son presentados en la Fig. 18 y son los casos extremos. Como se puede observar el numero se extendió acorde a su bit mas significativo y su signo se respeto.

Figure 18: Resultados de modulo Sign Extender.

3.6. Multiplexores 2 a 1

Para ambos multiplexores se probaron ambos casos, la entrada A y la entrada B, con el bit de control el 0 o 1 respectivamente (Fig. 19 - 22). Lo que comprueba que su funcionamiento es el esperado, y se mantiene siempre el valor en la salida de la entrada que se selecciona con el bit de control de forma concurrente.

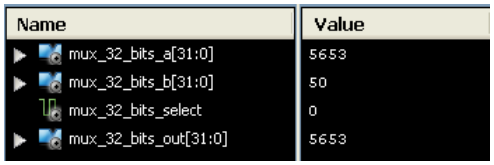

Figure 19: Caso Control 0 en MUX 32 bits.

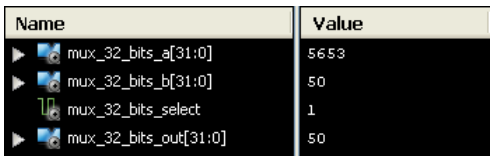

Figure 20: Caso Control 1 en MUX 32 bits.

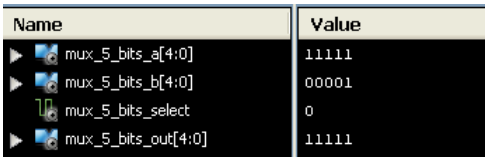

Figure 21: Caso Control 0 en MUX 5 bits.

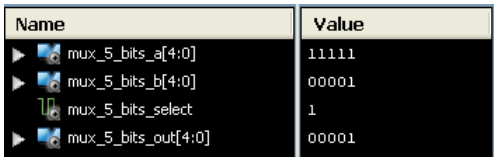

Figure 22: Caso Control 1 en MUX 5 bits.

4. Conclusión

4.1. Paul Vazquez

Con esta practica reforzamos exitosamente los conceptos básicos de diseño en VHDL. Aunque se trato de la elaboración de 5 módulos muy diferentes en comportamiento, cada uno nos aporto algo a nuestro aprendizaje de circuitos concurrentes y secuenciales. Es importante recalcar que los componentes que elaboramos en esta practica son parte del proyecto final del MIPS por lo que realizamos pruebas extensas para comprobar que funcionar bien en todos los casos. No hubo ninguna complicación mayor en la realización de esta practica o el reporte.

4.2. Rodolfo Cuan

Esta es la primera practica y los conceptos desarrollados fueron sencillos. Sin embargo, eran necesarios para recordar conceptos que habíamos visto en laboratorios anteriores, principalmente VHDL. Al desarrollar los distintos módulos, el funcionamiento de cada uno fue comprendido de una mayor manera; a comparación de verlos solo teóricamente en el salón de clases. Igualmente, se usaron plataformas las cuales eran familiares pero hace mucho no utilizábamos. Al subir el proyecto a GitHub facilito enormemente el desarrollo colaborativo, esta solucion sera igualmente implementada en las siguientes practicas.