FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES
UNIVERSIDAD NACIONAL DE CÓRDOBA

Arquitectura de Computadoras

Trabajo Páctico Nº3 - BIP

Aagaard Martin - Navarro Matias

Universidad Nacional de Córdoba
20 de noviembre de 2019

Índice

1. Introducción

En el siguiente trabajo se realizó la implementación en Verilog de un procesador BIP I. Se desarrolló para la placa Basys 3, utilizando el software Vivado 2018.2. El procesador permite la ejecución de instrucciones simples en un ciclo. Es posible visualizar el resultado de las instrucciones ejecutadas desde la placa: ciclos de clock empleados, y el resultado final en un registro acumulador.

Las operaciones que se pueden realizar son:

Operation	Opcode	Instruction	Data Memory (DM) and Accumulator (ACC) Updating	Program Counter (PC) updating	Affected Flags	BIP Model
Halt	00000	**HLT**		PC ← PC		I, II
Store Variable	00001	**STO** operand	DM[operand] ← ACC	PC ← PC + 1		I, II
Load Variable	00010	**LD** operand	ACC ← DM[operand]	PC ← PC + 1		I, II
Load Immediate	00011	**LDI** operand	ACC ← operand	PC ← PC + 1		I, II
Add Variable	00100	**ADD** operand	ACC ← ACC + DM[operand]	PC ← PC + 1	Z,N	I, II
Add Immediate	00101	**ADDI** operand	ACC ← ACC + DM	PC ← PC + 1	Z,N	I, II
Subtract Variable	00110	**SUB** operand	ACC ← ACC - DM[operand]	PC ← PC + 1	Z,N	I, II
Subtract Immediate	00111	**SUBI** operand	ACC ← ACC - operand	PC ← PC + 1	Z,N	I, II

Figura 1: *Instrucciones del BIP I*

2. Desarrollo

Se definió un módulo Top, que contiene:

- Módulo CPU
- Módulo RAM de Instrucciones
- Módulo RAM de Datos

2.1. CPU

El módulo CPU contiene una implementación del procesador BIP I en Verilog. Se descompuso en los siguientes bloques funcionales: control y data-

path.

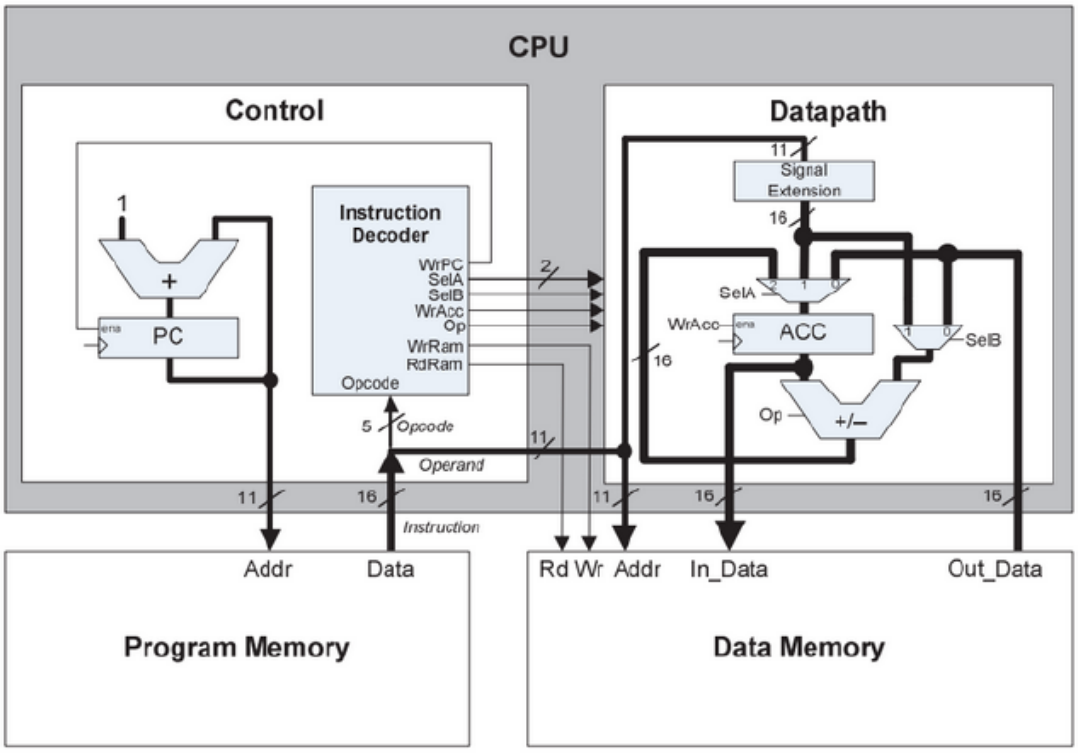

Figura 2: *Diagrama de bloques del procesador BIP I*

2.1.1. Control

El bloque de control del BIP se encarga de obtener la siguiente instrucción de la memoria de programa, la posición buscada indicada por el valor del registro PC. Además, posee un circuito combinacional que controla mediante diversas señales, el comportamiento del bloque de datapath, aquí se controlará exactamente que instrucción se realiza dependiendo las condiciones que se habilitan en el mismo. El único elemento controlado por el clock en este bloque es el registro PC.

Instruction Decoder

Este bloque habilita todas las señales necesarias para que se realiza la operación leída en el ciclo de clock. La parte llamada *opcode* (los 5 bits mas significativos) del la instrucción leída de la memoria de programa entran a

este bloque y, dependiendo de esto, el circuito combinacional modifica la salida. Las señales salientes son cables conectados al bloque datapath y a la memoria de datos (señales de escritura y lectura).

2.1.2. Datapath

Este bloque maneja el cálculo del resultado de la instrucción en curso. Trabaja accediendo a la memoria de datos según sea necesario, para almacenar un resultado, o leerlo y utilizarlo en una operación. Posee un único registro acumulador, una simple unidad aritmética, que permite realizar sumas y restas, y diversos multiplexores para seleccionar el dato con el que operar. Las señales de control mencionadas anteriormente se utilizan para controlar estos multiplexores, la escritura del registro acumulador, y el comportamiento de la memoria de datos (lectura o escritura).

2.2. RAM de Instrucciones y de Datos

Estos bloques se generaron mediante los templates que provee el entorno de desarrollo. Consisten en memorias de 16 bits, 2048 registros para las instrucciones y 1024 registros para los datos. La memoria de instrucciones fue inicializada con un archivo *.mem*, que contiene las instrucciones en binario utilizadas para probar el funcionamiento del BIP.

```
CODIGO          BINARIO             ACC

a(pos0)=10-2

ldi 10          0001100000001010    10
subi 2          0011100000000010    8
sto 0           0000100000000000    8

b(pos1)=a+1

ld 0            0001000000000000    8
addi 1          0010100000000001    9
sto 1           0000100000000001    9

c(pos2)=pos14

ld 14           0001000000001110    14
sto 2           0000100000000010    14

d(pos3)=a+b-c

ld 0            0001000000000000    8
add 1           0010000000000001    17
sub 2           0011000000000010    3
sto 3           0000100000000011    3

HALT            0000000000000000    3
```

Figura 3: *Programa cargado en la memoria de programa*

Un punto importante a destacar es el empleo del modo **Low Latency** para la memoria de datos. Esto fue necesario ya que, como las instrucciones deben completarse en un ciclo, se encontraron escenarios en los que esto no era posible, por ejemplo, si una instrucción solicita sumar el valor del acumulador con el guardado en una posición de la memoria de datos, la secuencia de operaciones a realizar es:

1. Leer el opcode de la memoria de programa.
2. Interpretarla y emitir las señales de control.
3. Leer el valor en memoria de datos a sumar.
4. Sumar el acumulador con el valor leído.
5. Guardar el resultado donde corresponda.

Todo el proceso debe realizarse en un ciclo, por requerimiento de diseño del procesador. Pero si se trabaja con un solo flanco no es posible, por lo que las operaciones con la memoria de programa se realizan en un flanco, y las de datos en el otro. Esto permite cumplir con los requisitos planteados sin comprometer el rendimiento del sistema.

2.3. ALU

El módulo ALU es el desarrollado en el primer práctico. Es puramente combinacional, con dos entradas para operandos, una entrada para el operador y una salida con el resultado de la operación.

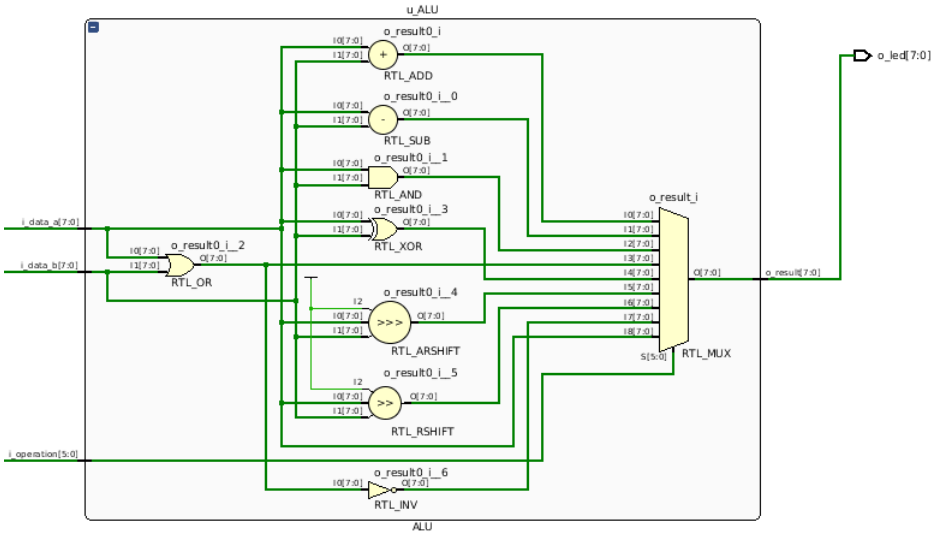

Figura 4: *Estructura RTL del módulo ALU obtenida de la herramienta Verilog*

3. Test Bench

Se realizaron las siguientes simulaciones para comprobar el correcto funcionamiento de cada uno de los bloques del sistema.

3.1. CPU

Permite verificar el correcto funcionamiento del procesador BIP en su totalidad, al interactuar con ambos bloques de memoria. Brinda información sobre el resultado en el acumulador, los valores escritos en memoria de datos, y el flujo de ejecución en general.

En la captura de pantalla 5 pueden verse los valores de las distintas señales de control luego de interpretadas diversas instrucciones.

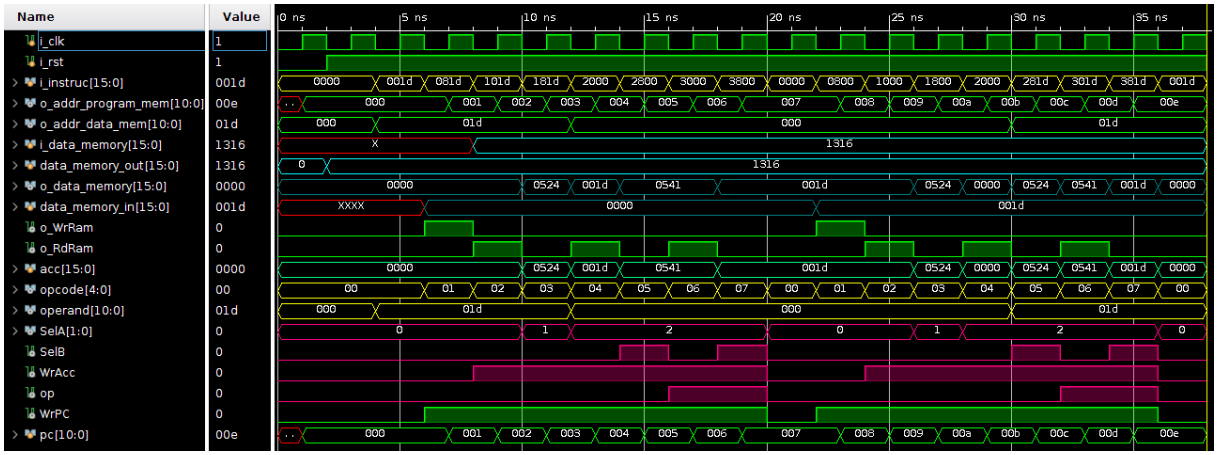

Figura 5: *Resultado del test bench del CPU*

3.2. Bloque de Control

Comprueba el correcto funcionamiento del bloque de control, dado un conjunto de instrucciones en la memoria de programa. Permite visualizar la salida del circuito combinacional que maneja el bloque de datapath.

3.3. Instruction Decoder

Comprueba el correcto funcionamiento del bloque de decodificación de instrucciones, dado un *opcode* (parte de la instrucción). Permite visualizar la salida del circuito combinacional, observando los valores de las señales de control.

3.4. Datapath

Se testeó que el módulo se comporte adecuadamente frente a señales de control determinadas que representan una operación o instrucción.

3.5. RAM de Instrucciones y de Datos

Se comprobó la carga de datos, que sean síncronas con el clock y que el flanco en el cual devuelven el dato solicitado sea el correcto.

3.6. Top BIP_I

Comprueba el correcto funcionamiento del bloque que engloba y conecta todos los otros bloques. Permite visualizar algunos registros claves (PC y ACC) durante la ejecución del programa previamente cargado a la memoria de programa (ver sección 2.2).

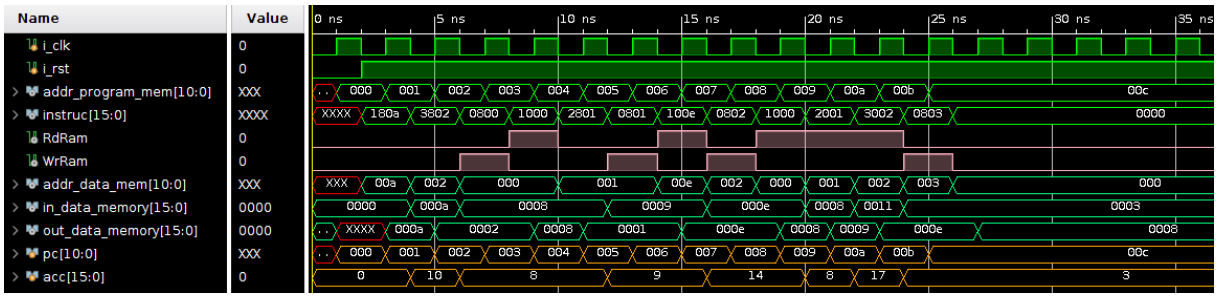

Figura 6: *Resultado del test bench del Top BIP_I*

4. Conclusión

Una vez terminado el práctico, a través de la herramienta Vivado se obtuvieron las especificaciones en la utilización de los recursos de la FPGA y en la potencia que consume dicho circuito instanciado.

Además, en este práctico se afianzaron los conceptos en cuanto al lenguaje de descripción de hardware Verilog, aprendimos sobre la utilización de bloques de memoria y sobre como utilizar un modulo de control.

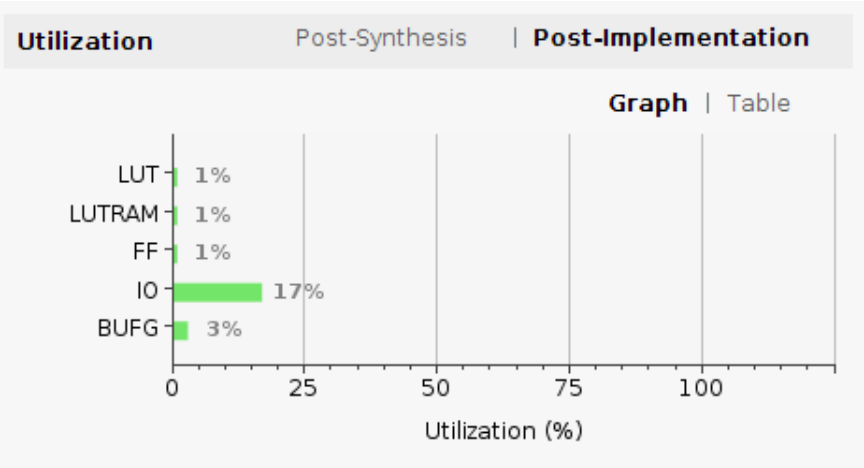

Figura 7: *Utilización de la FPGA*