Arquitectura de Computadoras

Ingeniería en Computación FCEFyN - UNC

Trabajo Práctico 3 - BIP I

Izquierdo Agustina Mat: 37729473

Salvatierra Andrés Mat: 39611008

27 de Noviembre de 2019

1. Introducción

Para el Trabajo Práctico N° 3, se realizó la implementación en Verilog de un procesador BIP I (basic instruction-set processor I). Al mismo se le instanciaron los módulos necesarios del UART, el cual fue diseñado e implementado en el Trabajo Práctico N° 2. El sistema se completa con una PC, a partir de la cuál se inicializa el sistema enviando una señal por UART. Dicho procesador ejecuta las instrucciones que levanta desde un archivo .txt. Al ejecutar la instrucción HALT, el valor del acumulador (ACC) es enviado hacia la PC vía UART, mientras que los valores del PC y el Count Clock son mostrados por led.

TABLE I. INSTRUCTION SET

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

Figura 1: *Set de instrucciones*

2. Diseño de Modulo

El esquema propuesto para Bip_I es el siguiente:

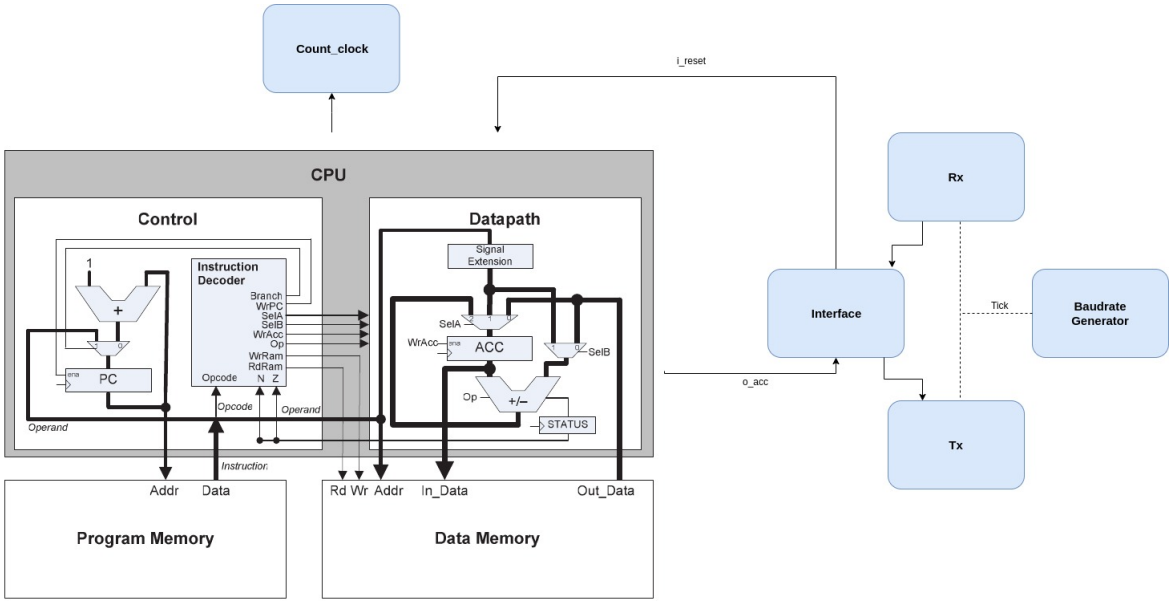

Figura 2: *Esquema general*

Como se puede ver en la figura 2, se crearán un modulo principal (**CPU**) en el cual estarán incorporados otros submodulos: **Control** y **Datapath**, sobre los cuales se ampliará a continuación.

3. Desarrollo del Bip_I

3.1. Control

Este módulo consiste en principalmente un decodificador de instrucciones, el cual recibe los comandos de la memoria de programa y los traduce en órdenes para la unidad de ejecución. Además,

en el mismo se encuentra el registro del contador de programa (**PC**), el cual se incrementa según las instrucciones.

3.2. Datapath

Es la unidad de de ejecución, consiste en su mayoria de elementos combinacionales, excepto por el registro acumulador (**ACC**). Dicho módulo cuenta con tres multiplexores y una extensión de signo.

3.3. Memoria de datos y programa

Para el desarrollo de las memorias se acudio a los templates de vivado donde explica como crear un módulo de memoria y cómo instanciarlo. Para disminuir los ciclos de procesamiento de instanciaron las memorias con el parametro de baja latencia (**LOW_LATENCY**). Además para ingresarle datos iniciales a la memoria de programa se utilizó un archivo externo (**instrucciones.txt**) con las instrucciones a ejecutar.

OPCODE	Instruction	OPERANDO	Explicacion
00011	Load Immediate	00000001010	ACC = 10
00001	Store Variable	00000000001	DM[1] = 10
00010	Load Variable	00000000010	ACC = 2
00001	Store Variable	00000000011	DM[3] = 2
00010	Load Variable	00000001010	ACC = 10
00100	Add Variable	00000000010	ACC = 10 + DM[2] es decir ACC= 12
00101	Add Immediate	00000000001	ACC = 12 + 1 es decir ACC= 13
00111	Subtract Immediate	00000000011	ACC = 13 -3 es decir ACC= 10
00001	Store Variable	00000001010	DM[10] = ACC

Figura 3: *Instrucciones implementadas*

En cuanto a la memoria de datos se inicializa con un (**index**).

```
// The following code either initializes the memory values to a specified file or to all zeros to match hardware
generate
  if (INIT_FILE != "") begin: use_init_file
    initial
      $readmemb(INIT_FILE, BRAM, 0, RAM_DEPTH - 1); //Inicializa con el archivo
  end
  else begin: init_bram_to_zero //Inicializa en cero en caso de no existir el archivo
    integer ram_index;
    initial
      for (ram_index = 0; ram_index < RAM_DEPTH; ram_index = ram_index + 1)
        BRAM[ram_index] = ram_index;
  end
endgenerate
```

Figura 4: *Inicializacion de memoria de datos*

3.4. Comunicación con el exterior

Para obtener datos del monoprocesador, se instació el modulo UART desarrollado durante el práctico anterior, diseñando una nueva interfaz de comunicación.

A continuación en 5se puede ver la máquina de estado de la interfaz implementada:

4. Simulaciones del módulo

A partir de una secuencia de intrucciones dadas, las cuales son cargadas desde el archivo antes mencionado, se realizó la verficación del funcionamiento del modulo principal a traves de un TestBech. El resultado para esa secuencia se puede ver a continuacion:

En el testbench de la figura 6 se puede observar que se llevaron a cabo la secuencias de operaciones y cuyo resultado final quedó almacenado en el contador (**i_interfaz_tx_data = 10**). Tambien se puede ver como las intrucciones ejecutadas varían según el aumento del (**PC**).

5. Implementación en FPGA

Para la implementación en la FPGA se utilizarón los diseños mostrados en la sección (**Diseño de módulo**). A su vez, se utilizaron los constrains de la placa correspondiente como el (**Clock**). y las (**Interfaces_Uart**).

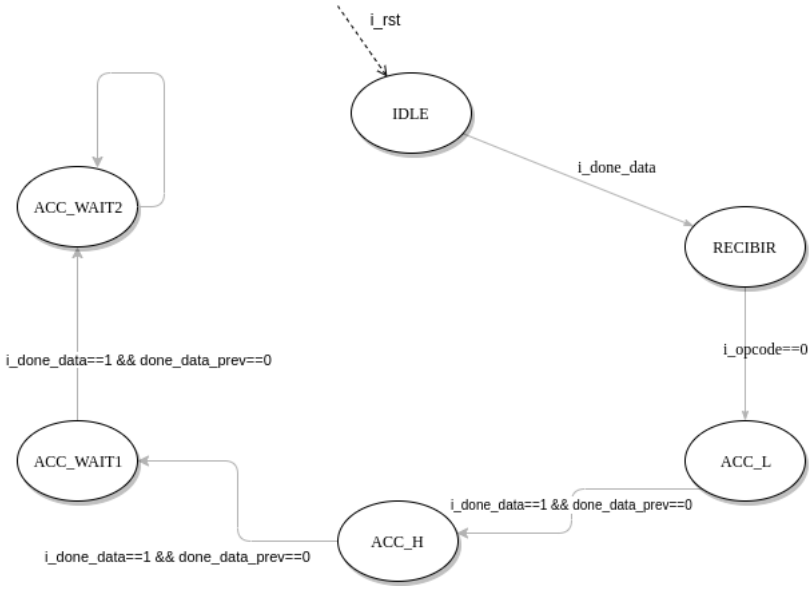

Figura 5: *Máquina de estados Interfaz Uart*

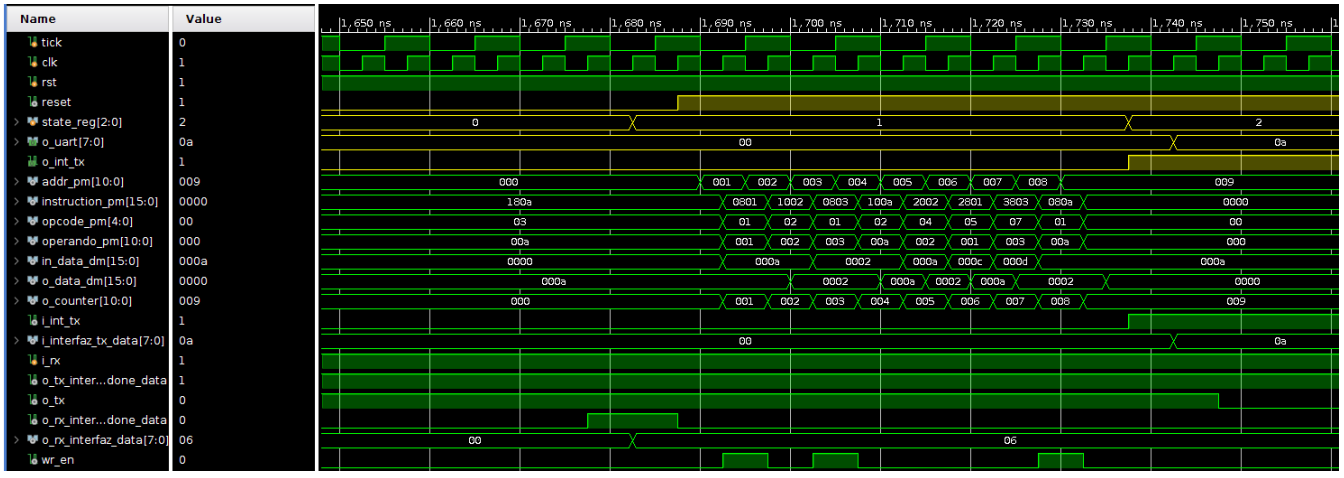

Figura 6: *Simulación Test Bench del sistema en su totalidad*