Laboratorio de Electrónica Digital II
Práctica No. 2: CPU Básica - Parte 2

Profesor Teoría
Felipe Cabarcas Jaramillo. (felipe.cabarcas@udea.edu.co)
Profesores Laboratorio
Andrés Benavides Arévalo (bernardo.benavides@udea.edu.co)
John Byron Buitrago (john.buitrago@udea.edu.co)

Junio 13, 2022

Fecha de entrega:	Julio 12-15 de 2022
Medio de entrega:	Classroom
Sustentación:	Horario de Laboratorio
Valor Práctica:	8% del curso

1 Introducción

La unidad central de proceso o CPU (por sus siglas en inglés: Central Process Unit) de un computador, se encarga de manipular los datos dados de acuerdo con las instrucciones en la memoria. En su forma más básica, la CPU se compone de dos bloques funcionales esenciales: el *Datapath* (o camino de los datos) y la *Unidad de Control*. En esta práctica de laboratorio, el grupo de trabajo (máximo dos integrantes) llevará a cabo el diseño de la *Unidad de Control* para el *Datapath* diseñado en el laboratorio 1. El grupo de trabajo también diseñará una máquina de estados

FSM_UNIT que permita calcular la dirección de memoria que contiene la instrucción a ejecutar. Finalmente, el grupo diseñará un pequeño programa para la CPU básica. La descripción del hardware será realizada mediante el uso del lenguaje de descripción de hardware SystemVerilog, mientras que las pruebas de funcionamiento se llevarán a cabo mediante el uso de herramientas de simulación.

2 Objetivo de la Práctica

Diseñar la *Unidad de Control* de una CPU básica, empleando el lenguaje de descripción de hardware SystemVerilog y herramientas de simulación, capaz de ejecutar operaciones simples de 8-bits sobre los datos disponibles tanto en los registros como en la memoria de datos.

3 Procedimiento

Leer completamente esta guía para planear el diseño de la *Unidad de Control*, *FSM_UNIT* y el código del programa. Como primer paso, el grupo de estudiantes realizará una propuesta gráfica usando componentes digitales que describan cada componente de la unidad de control, diagrama de estados de la FSM y pseudocódigo. A continuación, procederán a codificar cada componente en SystemVerilog y a simular el código propuesto para verificar su correcto funcionamiento, llevando a cabo las correcciones que sean pertinentes. Posteriormente, conectarán los componentes individuales en un diseño final el cual será descrito en SystemVerilog y simulado con un test de prueba que permita verificar su correcto funcionamiento. Finalmente, el grupo de trabajo redactará un corto reporte y lo enviará junto con los archivos del proyecto en SystemVerilog. Más detalles sobre la entrega se darán a conocer al final de esta guía. La correspondiente sustentación se realizará en el horario de laboratorio correspondiente en la semana establecida para la entrega.

4 Especificaciones y Funcionamiento de la *unidad de control*

La Fig. 1 muestra una CPU básica junto con dos memorias: *Instruction Memory*, donde se almacenan las instrucciones a ejecutar y *Data Memory*, donde se almacenan los datos a procesar o ya procesados. En este modelo, la CPU tomará las instrucciones de la *Instruction Memory*, las decodificará dentro de su *Control Unit FSM* y, posteriormente, las ejecutará empleando la unidad de *Datapath* y la memoria de datos (Data Memory). En esta práctica, el grupo de trabajo diseñará la *Unidad de Control*, la cual trabajará en conjunto con el *Datapath* y la memoria de datos (*Data Memory*) diseñados en el laboratorio anterior, para completar la CPU básica. Posteriormente, el grupo de trabajo escribirá las instrucciones necesarias para el programa de software propuesto (ver Tab. 1). Finalmente, diseñarán una maquina de estados finitos (FSM) llamada *FSM_UNIT*, encargada de coordinar la ejecución del programa en la CPU básica.

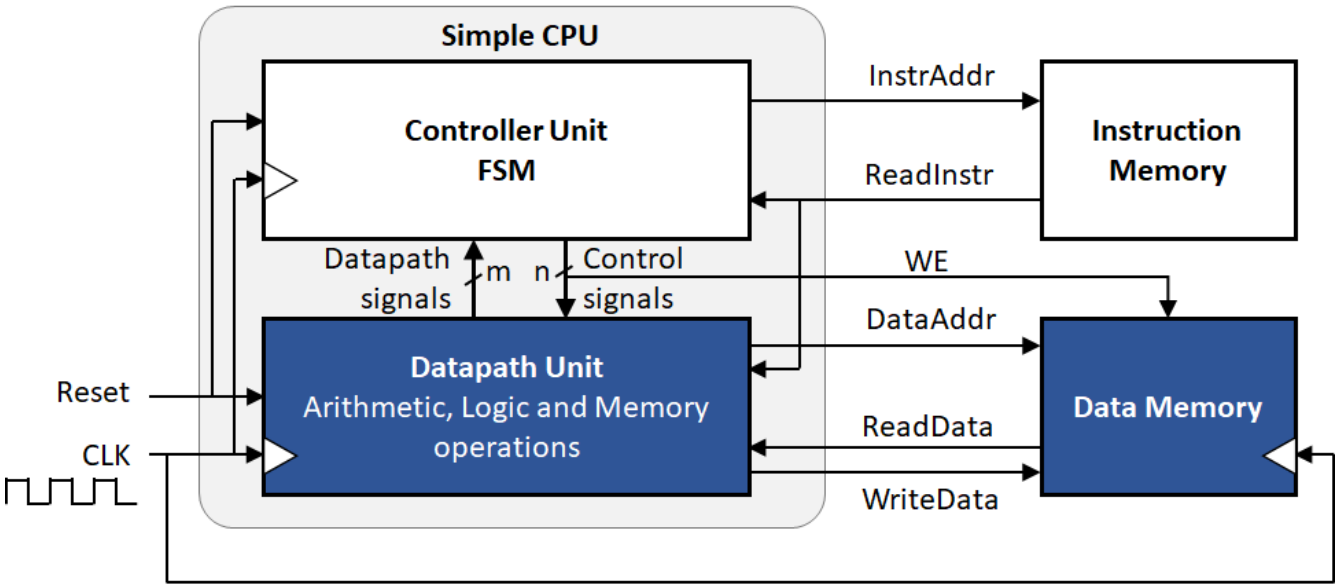

Fig. 1: CPU Simple + Memoria de datos e instrucciones

Tab. 1: Programa a diseñar. El número de grupo será asignado por el profesor

Grupo	Programa
1	f(x,y,z)=22x+13y+14z+6
2	f(x,y,z)=13x-12y+19z+8
3	f(x,y,z)=19x+14y-23z+7
4	f(x,y,z)=20x-12y+23z+6
5	f(x,y,z)=13x-23y+13z+8
6	f(x,y,z)=23x-14y+15z+7
7	f(x,y,z)=24x-14y+15z+6
8	f(x,y,z)=11x-23y+12z+5
9	f(x,y,z)=21x-22y+13z+4
10	f(x,y,z)=13x-23y+14z+6
11	f(x,y,z)=23x-10y+24z+6
12	f(x,y,z)=11x+13y-14z+9
13	f(x,y,z)=13x-7y+15z+7
14	f(x,y,z)=12x-33y+22z+4
15	f(x,y,z)=23x+13y+24z+3

Los módulos a diseñar junto con sus características se describen a continuación y se muestran en la Fig. 2 :

- **FSM_UNIT** (Máquina de estados): Calcula la dirección de memoria que contiene la instrucción a ejecutar.
- **ControllerUnit** (Unidad de Control): Decodifica la instrucción a ejecutar y genera las

señales de control para el *Datapath*.

- **Instruction Memory** (Memoria de Instrucciones): bloque de 256 registros o direcciones de memoria, de 16-bits cada uno, que conforman una memoria de solo lectura. Cualquiera de los registros o direcciones de memoria se puede seleccionar mediante la entrada InstrAddr. El valor del registro seleccionado será llevado a la salida ReadInstr. Cada campo de memoria contiene la codificación binaria de la instrucción a ejecutar. El resumen de las operaciones que puede realizar se muestran en la Fig. 3.

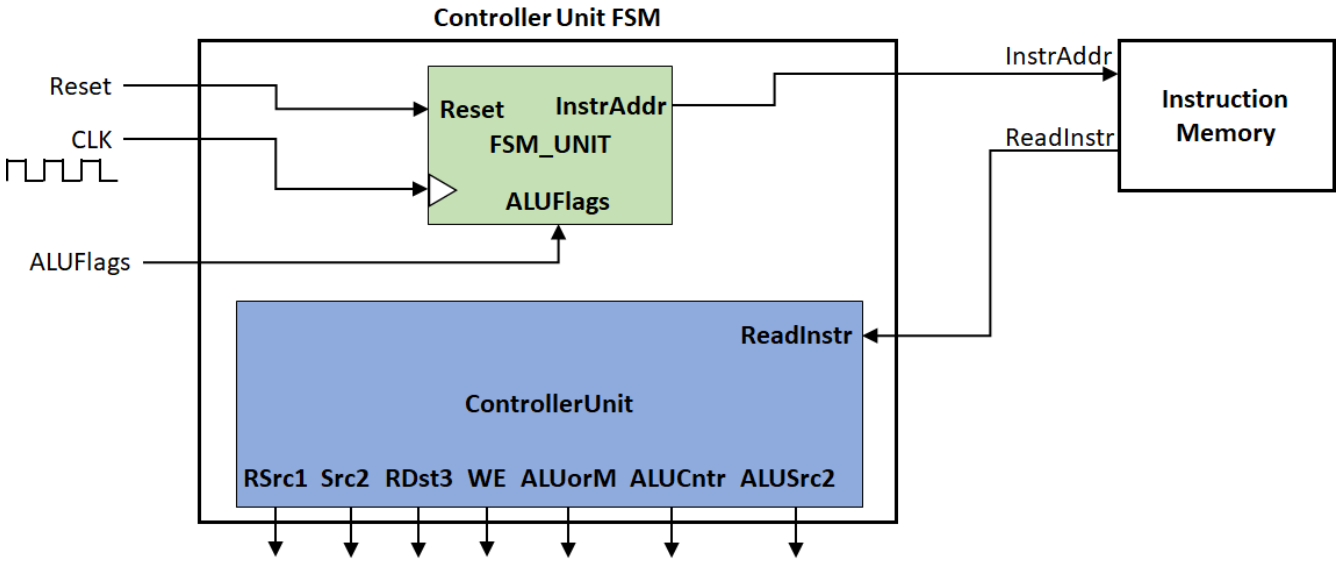

Fig. 2: Unidad de Control y máquina de estados

5 Diseño

El módulo principal de su diseño en SystemVerilog debe tener la siguiente declaración:

```
module SimpleCPU_DM_IM_yy(input logic CLK,
                          input logic RST,
                          output logic WE,
                          output logic [7:0] InstrAddr,
                          output logic [7:0] ReadInstr,
                          output logic [7:0] DataAddr,
                          output logic [7:0] ReadData,
                          output logic [7:0] WriteData,
);
```

donde, yy corresponde a las iniciales de los integrantes del grupo.

CPU Inst	OpCode [15:13]			RDst [12:11]	RSrc1 [10:9]	R/Imm [8]	RSrc2 [7:6]	Immediate Value [5:0]						Instruction Example
AND	0	0	0	Reg	Reg	0	Reg	X	X	X	X	X	X	AND R0, R1, R2: R0 ← R1 & R2
AND	0	0	0	Reg	Reg	1	Immediate Value 8-Bits							AND R0, R1, #0xFF: R0 ← R1 & 0xFF
NOT	0	0	1	Reg	XX	0	Reg	X	X	X	X	X	X	NOT R0, R1: R0 ← ~R1
NOT	0	0	1	Reg	XX	1	Immediate Value 8-Bits							NOT R0, #0xEF: R0 ← ~0xEF
XOR	0	1	0	Reg	Reg	0	Reg	X	X	X	X	X	X	XOR R2, R3, R2: R2 ← R3 ^ R2
XOR	0	1	0	Reg	Reg	1	Immediate Value 8-Bits							XOR R1, R2, #0xA0: R1 ← R2 ^ 0xA0
ADD	0	1	1	Reg	Reg	0	Reg	X	X	X	X	X	X	ADD R0, R1, R2: R0 ← R1 + R2
ADD	0	1	1	Reg	Reg	1	Immediate Value 8-Bits							ADD R0, R1, #0x5A: R0 ← R1 + 0x5A
SUB	1	0	0	Reg	Reg	0	Reg	X	X	X	X	X	X	SUB R0, R1, R2: R0 ← R1 – R2
SUB	1	0	0	Reg	Reg	1	Immediate Value 8-Bits							SUB R0, R1, #0x5A: R0 ← R1 – 0x5A
MOV	1	0	1	Reg	XX	0	Reg	X	X	X	X	X	X	MOV R0, R1: R0 ← R1
MOV	1	0	1	Reg	XX	1	Immediate Value 8-Bits							MOV R0, #0xFF: R0 ← 0xFF
LDR	1	1	0	Reg	Reg	1	Immediate Value 8-Bits							LDR R0, [R1, #0x20]: R0 ← [R1 + 0x20]
STR	1	1	1	Reg	Reg	1	Immediate Value 8-Bits							STR R0, [R1, #0x00]: [R1 + 0x00] ← R0

Fig. 3: Instrucciones disponibles

Guarde su módulo en un archivo de nombre **SimpleCPU_DM_IM_yy.sv**. El reloj o señal **CLK** que entra al controlador será de 1khz. A continuación describa en SystemVerilog su módulo de manera que realice la funcionalidad pedida y que pueda ser sintetizado y simulado. Incluya sub-módulos para tener un diseño modular.

6 Simulación

Cree un archivo denominado testbench_yy.sv que demuestre, de manera convincente, que el *SimpleCPU*, la memoria de datos y la memoria de instrucciones que usted ha descrito en SystemVerilog desarrolla toda la funcionalidad requerida de manera correcta, de acuerdo con la Fig. 1. Emplee ModelSIM o cualquier otra herramienta de simulación para esta actividad.

Es posible que la primera vez usted obtenga errores en el módulo de su diseño o en el testbench (banco de pruebas). Interprete de manera apropiada los mensajes que recibe de la herramienta de simulación y corrija los problemas que se han presentado.

7 Entrega

El grupo de trabajo deberá escribir un reporte que contenga los siguientes elementos:

a. Resumen del diseño del *SimpleCPU*, la memoria de datos y la memoria de instrucciones.

b. Diagrama digital de cada componente de su diseño con una breve explicación. Se aceptan imágenes escaneadas o fotografías siempre que sean claras.

c. Conjunto de instrucciones que resuelvan el problema asignado.

d. Resultados de la simulación mostrando que su diseño realiza correctamente la funcionalidad descrita en esta guía.

e. Conclusiones.

Crear un archivo comprimido que incluya el reporte y los archivos importantes de su proyecto en Xilinx Vivado o Quartus Prime como se describe a continuación:

a. Reporte: archivo con extensión .pdf

b. Archivos fuente:SimpleCPU_DM_IM_yy.sv, testbench_yy.sv, otros archivos .sv que haya creado.

El nombre del archivo comprimido debe tener el siguiente formato:
p2_primerapellidointegrante1_primerapellidointegrante2_horariolaboratorio.zip.
Ejemplo: si el primer apellido de ambos integrantes es **Benavides** y **Buitrago**, respectivamente, y el laboratorio es el Martes 9-12, entonces el archivo debe ser nombrado: *p2_benavides_buitrago_m9-12.zip*.

8 Evaluación

La evaluación de la práctica se divide en tres partes: funcionamiento (40%), sustentación (40%) y reporte (20%). La nota del funcionamiento se asigna por igual a todos los integrantes del grupo de trabajo (máximo dos personas por equipo), mientras que la nota de sustentación es individual. En caso un estudiante obtenga una nota inferior a 3.0 en la sustentación, la nota final de la práctica para el estudiante en mención será la que obtuvo en la sustentación, es decir, no se tendrá en cuenta el funcionamiento en el cálculo.

Cada grupo de trabajo, con un máximo de dos integrantes, deberá sustentar la práctica en un tiempo de 20 minutos, 10 minutos para revisar la simulación y 10 minutos para preguntas. Durante la sustentación, el profesor hará un par de preguntas a cada uno de los integrantes del grupo de trabajo.

9 Referencias

a. Xilinx Vivado WebPack
`https://www.xilinx.com/support/download.html`

b. Quartus Prime Lite Edition
`https://fpgasoftware.intel.com/?edition=lite`

c. Ambiente de Simulación Online
`https://www.edaplayground.com/home`

d. Tutorial de SystemVerilog
`https://verilogguide.readthedocs.io/en/latest/`