Descripción

En este laboratorio se realizará el diseño, simulación, e implementación de 4 de los módulos de un procesador de arquitectura RISC-V. Estos módulos serán implementados en la tarjeta de desarrollo Terasic DE1-SoC que contiene una FPGA Intel Cyclone V.

1. Unidad Aritmética Lógica

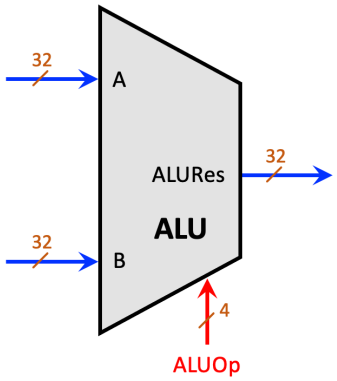

Figura 1. Unidad Aritmética Lógica

La Unidad Aritmética Lógica (Artihmetic Logic Unit ALU) recibe dos operandos, **A** y **B**, de 32 bits cada uno y una señal de entrada llamada **ALUOp** de 4 bits que define la operación a ejecutar por este módulo de acuerdo con la Tabla 1 y entrega el resultado de esta operación en una salida llamada **ALURes** de 32 bits.

	ALUOp		
$A + B$	0000	$A \oplus B$	0100
$A - B$	1000	$A \gg B$	0101
$A \ll B$	0001	$A \ggg B$	1101
$A < B$	0010	$A \vee B$	0110
$A < B\ (U)$	0011	$A \wedge B$	0111

Tabla 1. Valores de ALUOp para las operaciones.

2. Unidad de registros

La Unidad de Registros es modulo que contiene los datos de variables (enteros y apuntadores de memoria) de mas rápido acceso en el procesador. Esta unidad contiene 32 registros de 32 bits cada uno, identificados desde el x0 (00000) hasta el x31 (11111). Esta unidad cuenta con dos puertos de lectura que permiten obtener el contenido de dos registros de forma simultanea y un puerto de escritura. El primer puerto de lectura tiene como entrada la señal **rsl** de 5 bits que debe tener el identificador del registro que se desea leer y retorna los 32 bits del contenido de ese registro por la salida **RU[rs1]**. Esto

mismo sucede con el segundo puerto el cual tiene como entrada **rs2** y salida **RU[rs2]**. El puerto de escritura recibe como entrada **rd** que debe contener los 5 bits del identificador del registro que se desea escribir, **DataWr** que debe contener los 32 bits que se desean escribir en el registro destino y una señal de con **RUWr** que define si el proceso de escritura se va a realizar de forma efectiva en el próximo flanco de subida de la señal de reloj. **RUWr** es una señal de control activa en alto, esto significa que cuando esta en estado 1 se realiza el proceso de escritura y en estado 0 no se realiza.

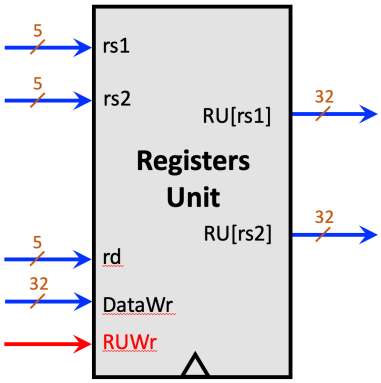

Figura 2. Unidad de registros

3. Memoria de instrucciones

La memoria de instrucciones permite almacena el software que va a ser ejecutado por el procesador. Esta memoria tiene como entrada un bus de direcciones de 32 bits, pero para efectos prácticos en la implementación que se debe hacer en la FPGA son se deben usar de 10 a 15 bits lo cual permite almacenar en ella varios miles de instrucciones. La salida es un bus de 32 bits en donde se encuentra la instrucción que esta almacenada en la dirección que previamente se ingreso en la entrada.

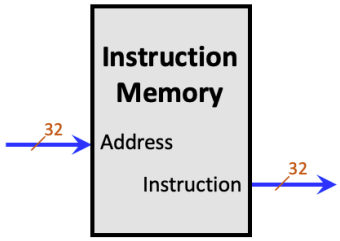

Figura 2. Memoria de instrucciones

4. Memoria de datos

La memoria de datos nos permite almacenar las variables, vectores, matrices y otras estructuras de

datos que va a ser manipuladas por el procesador. Adicionalmente, allí se encuentra almacenada la pila que permita la persistencia de registros a través del llamado a funciones. Esta memoria cuenta con dos puertos, uno de escritura y otro de lectura, pero ambos comparten el bus de direcciones de entrada. Para la escritura se debe ubicar en la entrada los 32 bits de la dirección en donde se quiere escribir y los 32 bits del dato que se quiere escribir y luego se debe activar la señal **DMWr** para que la escritura sea efectiva. Para el proceso de lectura se utiliza el mismo bus de entrada de direcciones y esta memoria retorna el contenido de esa dirección en un bus de salida de 32 bits. La lectura se puede realizar de 8, 16 o 32 bits de acuerdo con una señal de control llamada **DMCtrl**. Las lecturas de 8 y 16 bits se pueden realizar con extensión de ceros o extensión de signo.

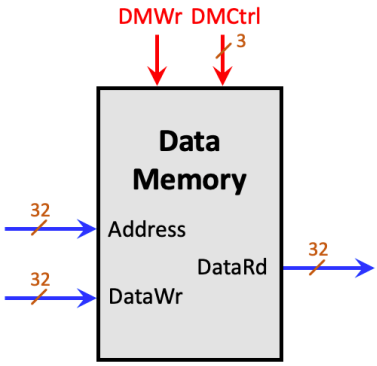

Figura 2. Memoria de datos.

DMCtrl			
B	000	B(*U*)	100
H	001	H(*U*)	101
W	010		

Tabla 2. Valores de **DMCtrl** para las opciones de lectura de memoria de datos.

Todos los módulos deberán ser diseñados en el lenguaje de descripción de hardware SystemVerilog y simulados con un banco de pruebas en la plataforma EDAPlayground. Deberán tomar capturas de pantalla de los archivos de diseño y banco de pruebas y de la simulación realizada en EPWave. Se deberá explicar como los resultados de la simulación coinciden con la respuesta esperada. Luego se deberá diseñar una estrategia para verificar el diseño en la FPGA usando como entradas los interruptores y los pulsadores y como salida los LEDs y los visualizadores de 7 segmentos. Para realizar este proceso se requiere de elementos adicionales que también debe ser diseñados, tales como, multiplexores y decodificadores. Luego se debe proceder con la síntesis y configuración de la FPGA usando el entorno de Intel Quartus y la posterior verificación física del diseño.