INEL 4215: Proyecto
Fase 1: Simulación de Componentes del Datapath

Tarea:

En esta fase cada integrante del grupo, de manera independiente, diseñará, codificará y simulará en Verilog uno de los tres componentes que se indican adelante. Cada estudiante es solo responsable del componente que le toque. En casos de grupos de menos de tres integrantes, no serán responsables del o los componentes que se queden sin asignar. Eventualmente los tendrán que implementar pero no serán responsables de ellos en esta etapa.

En las descripciones de cada uno de los componentes que siguen es muy probable que se necesiten señales de entrada adicionales para que el componente pueda operar correctamente. Ustedes deben identificar esas señales e incorporarlas en la implementación del componente.

ALU and Shifter/Sign Extender

Se diseñarán dos componentes: un ALU y un shifter/sign extender.

El ALU debe tomar dos números de 32 bits y un bit de carry y realizar cualquiera de las operaciones necesarias para implementar instrucciones "data processing" y cualquier otra instrucción que requiera operaciones aritméticas o lógicas. Las salidas del ALU serán el resultado de la operación y cuatro bits correspondientes a los "condition codes" (N, Z, C, V).

El Shifter/Sign Extender realizará las operaciones necesarias para generar el segundo operando fuente de las instrucciones data processing y load/store requeridas para el proyecto. Tendrá como entradas un número de 32 bits y otro de 12 bits y producirá el operando correspondiente. El número de 32 bits es normalmente el contenido del registro Rm de la instrucción. El número de 12 bits corresponderá a la codificación especificada por los bits I_{11}-I_0 de una instrucción data processing o load/store.

Demostración:

ALU: Realizar todas las operaciones con dos números. Se debe mostrar el valor del resultado y de los condition codes que correspondan a cada caso. En los casos de suma y resta deben utilizar dos números que produzcan overflow y dos que no produzcan overflow.

Para cada caso deben imprimir en una línea el código de la operación, los dos números, el resultado y los condition codes. Los dos números y el resultado se deben imprimir tanto en binario como en decimal. El código de la operación y los condition codes se deben imprimir en binario.

Shifter/Sign Extender: Utilizando el número 11101011000000000000000000000111 en la entrada de 32 bits, provea un código en la entrada de 12 bits para generar el segundo operando fuente para los siguientes addressing modes:

Data processing:

- Immediate (Second source operand = #<immediate>)
- Shift by immediate (Second source operand = <Rm>, <shift> #<shift_imm>). Deben mostrar el resultado para los 4 tipos de shifts.

Load/Store word or unsigned byte

- Immediate offset: (Second source operand = #+/-<offset_12>])
- Register offset: (Second source operand = [<Rn>, +/-<Rm>])

Para cada caso deben imprimir en una línea el número de 32 bits, el código de 12 bits y la salida, todas en binario.

Register File

Este componente debe ser implementado siguiendo la configuración ilustrada en la sección Register File de la lección Pipeline Processing Unit Organization. Sin embargo, El Register File tendrá un puerto de entrada y tres puertos de salida.

El registro R15, como es el PC, deberá tener un hardware particular que le permita ser incrementado como se requiere en la etapa IF del Pipeline Processing Unit (PPU). El circuito para incrementar el PC debe estar fuera del módulo del Register File pero es requisito implementarlo como parte de esta fase. El Register File debe tener una entrada, distinta al puerto de entrada, para proveerle un valor externo al PC, que normalmente sería PC+4. El PC tendrá una señal de "load enable" que cuando esté activa permitirá que el valor externo se cargue en el PC cuando ocurra el "rising edge" del reloj del sistema, excepto cuando el puerto de entrada trate de escribir el mismo, lo cual tiene prioridad. Como cualquier registro, el PC podrá ser leído por cualquiera de los puertos de salida.

Para cada componente de ese circuito debe haber un módulo en Verilog.

Demostración:

Deben escribir un word único en cada uno de los registros utilizando el puerto de entrada. Luego deben mostrar a través de los puertos de salida que los valores fueron guardados correctamente. Entonces, deben poner un word distinto en el registro R10 y leerlo por el puerto A.

En cada ciclo del reloj deben imprimir en una línea el PC, las señales de selección de cada puerto de salida y su contenido y finalmente las líneas de selección del puerto de entrada y el valor a la entrada del puerto.

Memoria RAM

Se diseñarán dos memorias RAM, una para instrucciones y otra para data.

La memoria de instrucciones tendrá un bus de salida (*Data Out*) y un bus de localización (*Address*), ambos de 32 bits . Estará en modo de lectura perpetuamente. Debe poder almacenar al menos 256 bytes. Debe ser precargada con un archivo de texto. El archivo consistirá de bytes separados con espacios o returns. Cada cuatro bytes forma una instrucción. El primer byte que se lee es el mas significativo de la instrucción y el cuarto el menos significativo (Big Endian).

La memoria de data debe estar organizada Big Endian y debe proveer acceso tanto a bytes, halfwords y words. Los double words se pueden manejar generando dos accesos de word consecutivos. Debe poder almacenar al menos 256 bytes. Tendrá buses independientes de entrada de data (*Data In*) y salida de data (*Data Out*). Tendrá además un bus de localización (*Address*) una señal *R/W* que con un valor de 0 indicará lectura y con un valor de 1 escritura. Debe ser precargada con un archivo de texto. El archivo consistirá de bytes separados con espacios o returns.

Demostración:

Memoria de instrucciones: Primero precargar la memoria utilizando un file con 16 bytes. Los bytes deben estar separados por espacios o returns. Utilizando los buses de la memoria deben leer el contenido de las primeras 16 localizaciones de memoria.

Para cada lectura o escritura deben imprimir en una línea las señales *Address*, y *Data Out*. La señal *Address* se debe imprimir en decimal y Data *Out* en hexadecimal.

Memoria de data: Primero precargar la memoria utilizando un file con 16 bytes. Los bytes deben estar separados por espacios o returns. Utilizando los buses y señales de la memoria deben hacer en orden lo siguiente:

- Leer un Word de las localizaciones 0, 4, 8 y 12.
- Leer un byte de la localización 0, un halfword de la localización 2 y un halfword de la localización 4.
- Escribir un byte en la localización 0, un halfword en la localización 2, un halfword en la localización 4 y un word en la localización 8.
- Leer un Word de las localizaciones , 4 y 8.

Para cada lectura o escritura deben imprimir en una línea las señales *Address*, *R/W*, *Data In* y *Data Out*. La señal *Address* se debe imprimir en decimal y *Data In* y Data *Out* en hexadecimal.

Entrega:

1. Subir a NEO un documento de texto (puro texto) con el código Verilog de los módulos del componente que le ha tocado y un módulo de prueba del mismo. El nombre del archivo debe terminar con uno de los siguientes keywords dependiendo de cual es el módulo: mem.v, alu.v, rf.v. Eso debe ser precedido por su primer apellido, segundo apellido y su nombre. Si yo fuese a someter un módulo del RAM el nombre del file sería el siguiente: PF1_Rodriguez_Rivera_Nestor_ram.v

Para los que les toque el componente de la memoria y tienen un files para precargar la memoria, entonces deberán nombrar ese file de la misma manera que el del código del componente pero terminando con ramintr.text y ramdata.txt.

2. Demonstrar en clase la operación de los tres componentes.

3. Reglas de juego para la demostración:
   Además de mostrar lo que he solicitado para cada caso, puedo pedirles que alteren su código de Verilog para mostrar otras cosas. Es muy probable que les haga preguntas sobre el código que desarrollaron. Si usted no puede contestar algunas preguntas sobre el código, que alguien que lo desarrolló debería contestar, entonces por deducción lógica voy a cuestionar la autoría del código y como consecuencia asignaré una puntuación de cero a esta fase.

Rúbrica:

- Se adjudicarán 15 puntos si el componente opera correctamente.

- Se podrían adjudicar puntos parciales en caso de que el componente no funcione correctamente dependiendo de cuán avanzado esté el diseño y la simulación del mismo.

- Se asignará una puntuación de cero a cada componente para el que no se someta código en Verilog.