Guía Práctica 1 - Verilog Risc V

Importante: De cada circuito es necesario diseñar su correspondiente banco de pruebas para detectar y corregir errores qué de otro modo serán difíciles de identificar en etapas posteriores de su aplicación.
El banco de pruebas debe crearse y ejecutarse antes de pasar al siguiente desarrollo de circuito.

Parte 1:
Empezaremos por diseñar en verilog los circuitos correspondientes al risc-V qué mantienen el estado del sistema. De cada circuito es necesario diseñar su correspondiente banco de pruebas para detectar y corregir errores qué de otro modo serán difíciles de identificar en etapas posteriores de su aplicación.

1. Registro Contador de programa.
El mismo es un registro de 16 bits de tipo biestable D.
Nombre: PC
Entradas: clk, pcNext(16).
Salidas: pc(16).

2. Memoria de Instrucciones.
El mismo es un arreglo de 64 registros de 32 bits.
Nombre: IM
Entradas: addres(16).
Salidas: rd(32).

3. Banco de Registros.
El mismo es un arreglo de 32 registros de 32 bits con:
Nombre: BR
Entradas: clk, a1(5), a2(5), a3(5), wd3(32), we.
Salidas: rd1(32), rd2(32).

4. Memoria de Datos.
El mismo es un arreglo de 32 registros de 32 bits con:
Nombre: DM
Entradas: clk, addres(16), wd(32), we.
Salidas: rd(32).

Parte 2:
Continuaremos por diseñar en verilog los circuitos qué resulten auxiliares para el camino de datos.

5. Extensión de Signo
El mismo es dispositivo qué extiende el bit de signo para operar con la ALU con:
Nombre: SE
Entradas: inm(25) src(2).
Interiores i0(31:20), is1(31:25, 11:7), ib2(31, 7, 30:25, 11:8, 1’b0), i 3( )
Salidas: inmExt(32).

6. Unidad Aritmetico Logica
El mismo es dispositivo qué: suma, resta por C2, AND, OR, SLT. a continuacion su tabla de verdad.

$ALUControl_{2:0}$	Function
000	add
001	subtract
010	and
011	or
101	SLT

Nombre: ALU
Entradas: srcA(32), srcB(32), ALUControl(3).
Salidas: res(32).

7. Unidad Aritmetico Logica
El mismo es dispositivo que suma
Nombre: Adder
Entradas: op1(32), op2(32).
Salidas: sal(32).

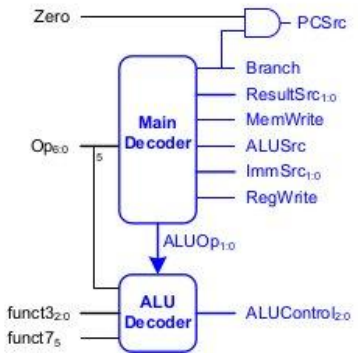
8. Multiplexor 2x1
El mismo es multiplexor 2x1 de 32 bits
Nombre: Mux2x1
Entradas: e1(32), e2(32) sel.
Salidas: sal(32).

Parte 3:
Continuaremos por diseñar en verilog los circuitos de la unidad de control.

9. Unidad de Control
El mismo es la unidad qué controla el camino de datos y contendra dos modulos: el Decodificador principal y el Decodificador de ALU.
Nombre: UC
Entradas: f7, f3(2:0), op(6:0), zero.
Salidas: sal(0:9) {pcSrc, resSrc, memWrite, ALUcontrol(2:0), aluSrc, inmSrc(1:0), regWrite …

10. Decodificador Principal
El mismo es la unidad qué decodifica el tipo de instrucción para generar el camino de datos.
A continuación definimos su tabla de verdad.

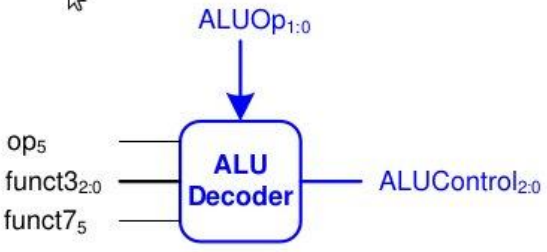
op	Instruct	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
3	`lw`	1	00	1	0	1	0	00
35	`sw`	0	01	1	1	X	0	00
51	**R-type**	1	XX	0	0	0	0	10
99	`beq`	0	10	0	0	X	1	01



Nombre: mainDeco
Entradas: op(6:0).
Salidas: branch, resSrc(1:0), memWrite, aluSrc, inmSrc(1:0), regWrite, aluOp(1:0)

11. Decodificador de ALU
El mismo es la unidad qué decodifica el tipo de operación para generar el control de la ALU.
A continuación definimos su tabla de verdad.

ALUOp	funct3	op_5, $funct7_5$	Instruction	$ALUControl_{2:0}$
00	x	x	`lw, sw`	000 (add)
01	x	x	`beq`	001 (subtract)
10	000	00, 01, 10	`add`	000 (add)
	000	11	`sub`	001 (subtract)
	010	x	`slt`	101 (set less than)
	110	x	`or`	011 (or)
	111	x	`and`	010 (and)



Nombre: aluDeco
Entradas: op(5), f7, f3(2:0), aluOp(1:0)
Salidas: aluControl(2:0)