Universitat Politècnica de Catalunya

Arquitectura de computadors d'altes prestacions

Práctica 4

Procesador: arquitectura, camino de datos y control

Carlota Catot
Miguel Antunez

Grupo 6

Quatrimestre primavera 2020-2021

Índice

1. Pregunta 1

Traduzca las siguientes instrucciones RISC-V a lenguaje ensamblador:

lenguaje máquina	lenguaje ensamblador
FF85AF03	lw R30, 0xFF8(R11)
00C735B3	sltu R11, R14, R12
10000297	auipc R5, 0x10000

lenguaje máquina	lenguaje ensamblador
00E780A3	sb, R15, 0x400(R14)
F7DFF0EF	jar R1, 0xFFFBE
01D09463	bge, R1, R29, 0x4

Para hacer las traducciones de las instrucciones hemos usado la información del apéndice 4.1 (*Formato y Codificación de las instrucciones*). Para ello hemos usado una tabla excel para poder analizar de manera más gráfica y ordenada los bits.

1.1. Traduccion de FF85AF03

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I	**literal12**												**RS1**					**funct3**			**RD**					**Codigo OP**							
FF85AF03	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0	0	0	1	1
lw R30, 0xFF8(R11)	4088 --> FF8												R11					lw			R30					LOAD							

Figura 1: Traducción de la instrucción FF85AF03 desglosada

1.2. Traduccion de 01D09463

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	**funct7**							**RS2**					**RS1**					**funct3**			**RD**					**Codigo OP**							
00C735B3	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0	1	0	1	1	0	1	1	0	0	1	1
sltu R11, R14, R12	-							R12					R14					sltu			R11					OP[a]							

Figura 2: Traducción de la instrucción 01D09463 desglosada

1.3. Traduccion de 10000297

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U	**literal20**																				**RD**					**Codigo OP**							
10000297	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	1	1
auipc R5, 0x10000	65536 --> 10000																				R5					auipc							

Figura 3: Traducción de la instrucción 10000297 desglosada

1.4. Traduccion de 00E780A3

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	**imm[12] imm[10:5]**							**RS2**					**RS1**					**funct3**			**imm[4:1] imm[11]**					**Codigo OP**							
00E780A3	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1
sb, R15, R14(0x400)	1024							R14					R15					sb			2					STORE							
	12	11	10	9	8	7	6	5	4	3	2	1																				
	0	1	0	0	0	0	0	0	0	0	0	0																				

Figura 4: Traducción de la instrucción 00E780A3 desglosada

1.5. Traduccion de F7DFF0EF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
J	**[20]**					**imm[10:1]**						**[11]**			**imm[19: 12]**						**RD**					**Codigo OP**							
F7DFF0EF	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	0	1	1	1	1
jal R1, 0xFFFBE	1048510 --> FFFBE																				R1					jal							
	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0												

Figura 5: Traducción de la instrucción F7DFF0EF desglosada

1.6. Traduccion de 01D09463

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	**imm[12] imm[10:5]**							**RS2**					**RS1**					**funct3**			**imm[4:1] imm[11]**					**Codigo OP**							
01D09463	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0	1	1	0	0	0	1	1
bge, R1, R29, 0x4	4							R29					R1					bge			4					BRANCH							
	12	11	10	9	8	7	6	5	4	3	2	1																				
	0	0	0	0	0	0	0	0	0	1	0	0																				

Figura 6: Traducción de la instrucción 01D09463 desglosada

2. Pregunta 2

Un tipo numérico de datos en un lenguaje de alto nivel tiene un rango de valores limitado. En consecuencia, al efectuar operaciones algebraicas es posible que el resultado no se pueda representar (desbordamiento).

Suponga la operación de suma de los números naturales x e y, con un rango de valores 0 <= x, y <M, donde M-1 es el máximo valor que se puede representar. Por tanto, la operación x + y produce debordamiento cuando x >(M-1) - y.

Escriba una secuencia de instrucciones en lenguaje C que detecte desbordamiento sin efectuar la suma. Suponga que el valor máximo de un número natural está especificado por la constante MAX. En las operaciones que se especifiquen no debe producirse desbordamiento en ningún caso.

2.1. Secuencia de instrucciones C

Debido a que solo disponemos de la variable MAX, solo miramos el desbordamiento de la suma por arriba, es decir, de numeros positivos, por ese motivo se ha supuesto que X >= 0 y Y <MAX.

```
1 desb = (x > (MAX - 1) - y);
```

Suponga que los operandos se representan con vectores de 32 bits (M = MAX+1 = 232). Escriba una secuencia de instrucciones RISC-V en lenguaje ensamblador que detecte desbordamiento sin efectuar la suma. Tenga en cuenta que las operaciones de la secuencia no deben producir desbordamiento en ningún caso. El número de instrucciones debe ser el mínimo. Suponga que los operandos x e y están almacenados en los registros x10 y x11 respectivamente. El resultado se almacena en el registro x9 (el valor 1 indica desbordamiento).

2.2. Secuencia de instrucciones ensamblador

```
1 sub x1, 0xFFFFFFFF, \$1
2 sub x2, x1, x11
3 slt x9, x2, x10
```

En la primera instrucción se carga en el registro x1 el valor de la resta MAX - 1. Sabiendo que 0xFFFFFFFF es el valor de MAX.

En la segunda instrucción se carga en el registro x2 el valor de la resta entre el valor de x1 - x11. Sabiendo que *x1 = (MAX-1)* y *x11 = y*.

En la tercera instrucción se carga en el registro x9 1 en el caso de que se cumpla la condicion de x2 <x10 y 0 en el caso contrario. Sabiendo que *x2 = (MAX-1)-y* y *x10 = x*

3. Pregunta 3

Suponga que el procesador interpreta una instrucción de secuenciamiento condicional que cumple la condición. Indique las instrucciones de secuenciamiento que pueden generar los valores de salida del módulo EVAL. Marque con X cualquier combinación que no se pueda producir.

ig	me	meu	instruccion secuenciamiento condicional
0	0	0	bne, bge, bgeu
0	0	1	bne, bge, bltu
0	1	0	bne, blt, bgeu
0	1	1	bne, blt, bltu
1	0	0	beq, bge, bgeu
1	0	1	X
1	1	0	X
1	1	1	X

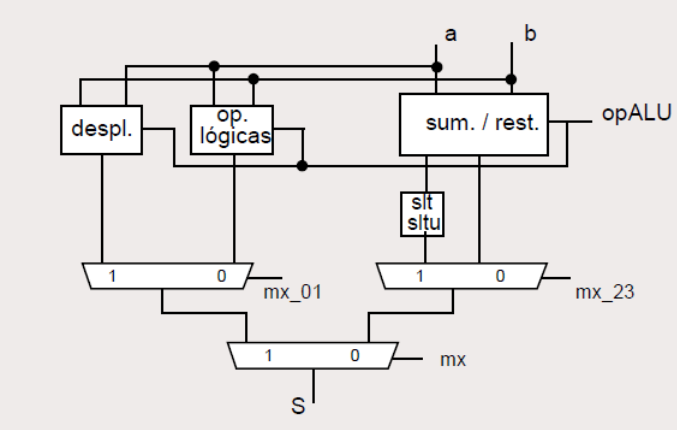
Para poder sacar la información de la tabla se han analizado las diferentes posibilidades y sabiendo que cada caso podía tener 3 instrucciones de secuenciamiento ya que hay 3 bits, de cada tipo de instrucción (podemos ver como se han separado en la figura 7) se ha buscado la combinación de 3 instrucciones que cumpliesen los 3 casos. Para los 3 últimos casos no existe combinación posible.

IG	bne	no igual
	beq	igual
ME	blt	<
	bge	>=
MEU	bgeu	>= u
	bltu	< u

Figura 7: Tabla con el tipo de instrucción separada por bit a analizar

4. Pregunta 4

En la figura se muestran las 3 unidades funcionales de la ALU y el árbol de multiplexores de selección: a) desplazamiento lógico o aritmético (a la derecha o a la izquierda), b) operación lógica, y c) sumador algebraico y comparador de menor (enteros o naturales). El módulo slt/sltu formatea (añade ceros a la izquierda) la salida de condición del sumador.



La ubicación de los ficheros relacionados con este proyecto (ALU.qpf) se indica en la página 295. Las dos primeras unidades funcionales y el formateador ya están diseñados (ficheros despla.vhd, logica.vhd y slt.vhd respectivamente). El fichero ALU.vhd contiene la descripción estructural de la ALU, excepto la selección de las salidas de las 3 unidades y el formateador. Esta seleccción se efectua mediante 2 niveles de multiplexores.

Deduzca las expresiones lógicas de las 3 señales de selección de los multiplexores en función de la señal de control opALU. Inclúyalas en cuerpo de la arquitectura.

4.1. Expressiones logicas multiplexores en funcion de opALU

nivel 1	mx_01	!opALU[1] and opALU[0]
	mx_23	opALU[1]
nivel 2	mx	opALU[2] or (!opALU[1] and opALU[0])

Para poder sacar estos valores nos hemos basado en la tabla de la pagina 330 que podemos ver en la figura 8 y a partir de ella hemos usado Excel para poder sacar de una manera lógica analizando los bits de cada opción las expressiones logicas, en los siguientes subapartados podemos ver la explicación de como se ha hecho.

	opALU					
	4	3	2	1	0	
Nem.		funct7(5) / imm(5)	funct3			acrónimo
add	1	0	000			suma
sub	1	1	000			resta
sll	1	0	001			despl_izquier
slt	1	0	010			cmp_menor_ent
sltu	1	0	011			cmp_menor_nat
slli	1	1	001			despl_izquier
addi	1	0	000			suma
andi	1	0	111			and_bit_a_bit
srai	1	1	101			despl_derec_aritm
auipc	1	0	000			suma
lui	1	0	000			suma

Figura 8: Codificación de la señal opALU en algunas instrucciones

4.1.1. Expressión lógica para *mx_01*

	4	3	2	1	0		
sll	1	0	0	0	1	DESP	mx01(1) mx(1)
slli	1	1	0	0	1	DESP	
srai	1	1	1	0	1	DESP	
andi	1	0	1	1	1	AND	mx01(0) mx(1)

Figura 9: Tabla para las instrucciones que pasan por el multiplexor mx01

En este caso, hemos de mirar que opALU[0] siempre sea 1 y tambien el contrario de opALU[1] (!opALU[1]), es decir el bit 1 de opALU, es 0 corresponderá con una instrucción de desplazamiento, por tanto el valor de mx01 será 1, por tanto el valor contrario de opALU[1]. En el caso de que el bit 1 de opALU sea 1, correspondera a una operación logica y por tanto mx01 será 0.

4.1.2. Expressión lógica para *mx_23*

	4	3	2	1	0		
slt	1	0	0	1	0	CMP	mx23(1) mx(0)
sltu	1	0	0	1	1	CMP	
sub	1	1	0	0	0	RESTA	mx23 (0) mx (0)
add	1	0	0	0	0	SUMA	
addi	1	0	0	0	0	SUMA	
auipc	1	0	0	0	0	SUMA	
lui	1	0	0	0	0	SUMA	

Figura 10: Tabla para las instrucciones que pasan por el multiplexor mx01

En este caso, si opALU[1], es decir el bit 1 de opALU, es 0 corresponderá con una instrucción de suma o resta, por tanto el valor de mx23 será 0, por tanto el valor de opALU[1]. En el caso de que el bit 1 de opALU sea 1, correspondera a una operación slt o stlu y por tanto mx23 será 0.

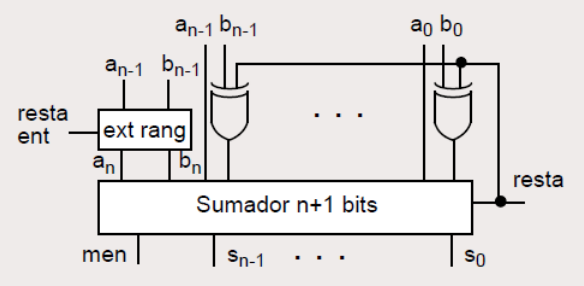
4.1.3. Expressión lógica para *mx*

	-	a	b	c	d		
	4	3	2	1	0		
slt	1	0	0	1	0	CMP	mx23(1) mx(0)
sltu	1	0	0	1	1	CMP	
sub	1	1	0	0	0	RESTA	mx23 (0) mx (0)
add	1	0	0	0	0	SUMA	
addi	1	0	0	0	0	SUMA	
auipc	1	0	0	0	0	SUMA	
lui	1	0	0	0	0	SUMA	
sll	1	0	0	0	1	DESP	mx01(1) mx(1)
slli	1	1	0	0	1	DESP	
srai	1	1	1	0	1	DESP	
andi	1	0	1	1	1	AND	mx01(0) mx(1)

Figura 11: Tabla para las instrucciones que pasan por el multiplexor mx01

Para este caso hemos usado una herramienta online en el siguiente link, de ahí nos ha salido la operación $y = B + C'D$, sabiendo que la suma es una or y la multiplicación una and, hemos sacado la expressión, tambien teniendo en cuenta que el bit a era el 3 y el d el 0 como se puede ver en la figura anterior.

El fichero sumalg.vhd contiene la interface del sumador algebraico y comparación de menor. Se utiliza un sumador de vectores de n+1 bits, puertas xor y lógica para extender el rango de los vectores a sumar. La salida men se activa cuando se cumple la condición a¡b. Recuerde que los operandos se pueden interpretar como enteros o naturales. Esta salida se formatea (añadiendo ceros a la izquierda) en la unidad slt/sltu, que también está diseñada (fichero slt.vhd).



Analice el fichero sumalg.vhd. Deduzca las sentencias de asignación concurrente de las señales resta y ent (en funcion de opALU) y del bit más significativo de los vectores de entrada del sumador de n+1 bits (an, bn). Escriba las sentencias de asignación de las salidas de la unidad (en función del vector resultado de la suma).

4.2. Expressiones logicas para el sumador de n+1 bits

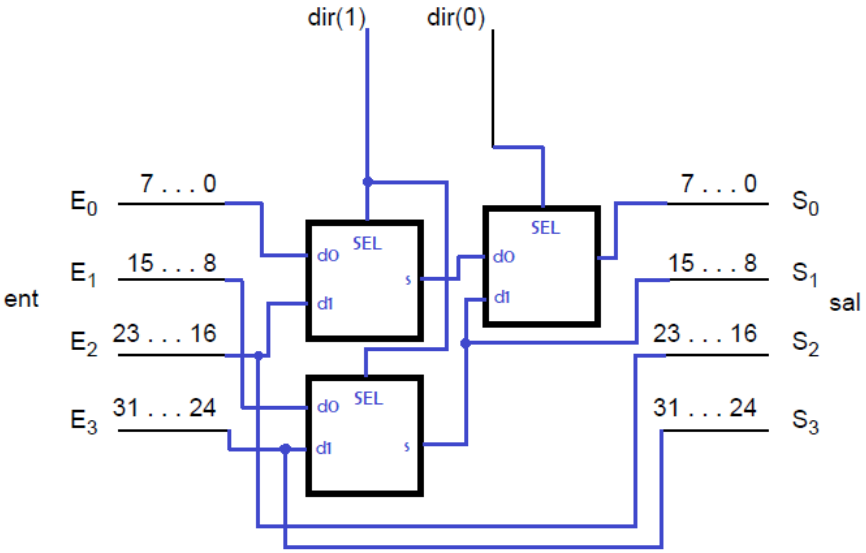
	resta	opALU[3] or opALU[1]	an	ent and a(31)
	ent	opALU[0]	bn	ent and b(31)
salidas	men	s(32)		
	s	s[0..31]		

Para deducir los datos de esta tabla simplemente se han seguido la tabla y con los valores de la tabla de opALU igual que en el ejercicio de antes se han deducido los valores.

5. Pregunta 5

El circuito FMTL (fichero FMTL.vhd) formatea el dato leído de la memoria que se utiliza para actualizar el banco de registros. Analice el componente "alinear" y muestre un esquema del circuito. Como ayuda, utilice el visor RTL de quartus. Rellene la tabla que relaciona los datos de entrada y salida del módulo en función de los dos bits menos significativos de la dirección. Utilice la nomenclatura Ei y Si para indicar el byte i del dato de entrada y de salida.

Basándonos en el modelo RTL que nos presenta quartus (figura 12) podemos sacar el circuito que se pide. Una vez tenemos nuestro propio circuito únicamente tenemos que seguir las diferentes salidas (es decir seguir el esquema del revés) para poder completar la tabla.



dir	sal			
1..0	S3	S2	S1	S0
0 0	E3	E2	E1	E0
0 1	E3	E2	E1	E0
1 0	E3	E2	E3	E0
1 1	E3	E2	E3	E0

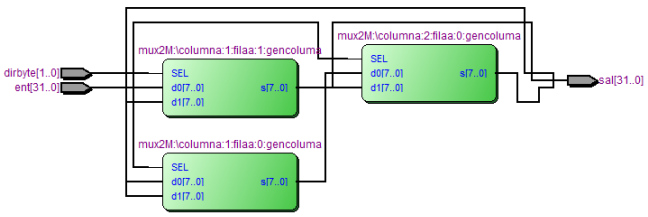
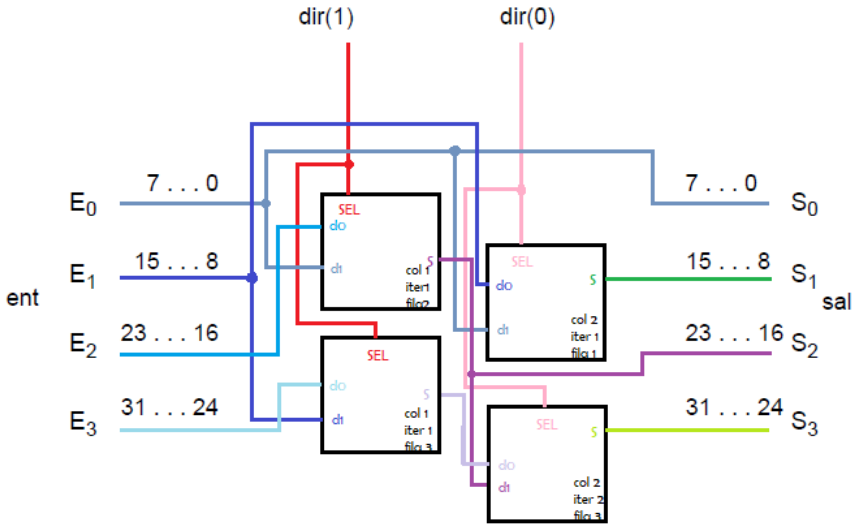


Figura 12: Esquema RTL del componente alinea de FMTL

6. Pregunta 6

El circuito FMTE formatea el dato leído del banco de registros para actualizar los bancos de la memoria de datos. En el diseño base (fichero FMTE.vhd), el circuito consta de 2 módulos: a) alineamiento de datos (fichero alinearE.vhd), y b) selección de los bancos a actualizar (fichero sel_byte.vhd). El módulo alinearE utiliza los 2 bits menos significativos de la dirección. Analice la descripción de la arquitectura del módulo alinearE y dibuje un esquema del circuito. Utilice el visor RTL de quartus. Rellene la tabla que relaciona los datos de entrada y salida del formateador en función de los dos bits menos significativos de la dirección.

Basándonos en el modelo RTL que nos presenta quartus (figura 13) podemos sacar el circuito que se pide. Una vez tenemos nuestro propio circuito únicamente tenemos que seguir las diferentes salidas (es decir seguir el esquema del revés) para poder completar la tabla.



dir	sal			
1..0	S3	S2	S1	S0
0 0	E3	E2	E1	E0
0 1	E2	E2	E0	E0
1 0	E1	E0	E1	E0
1 1	E0	E0	E0	E0

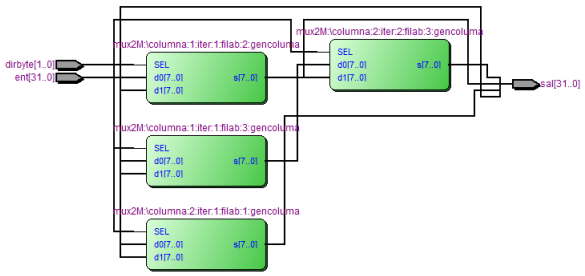
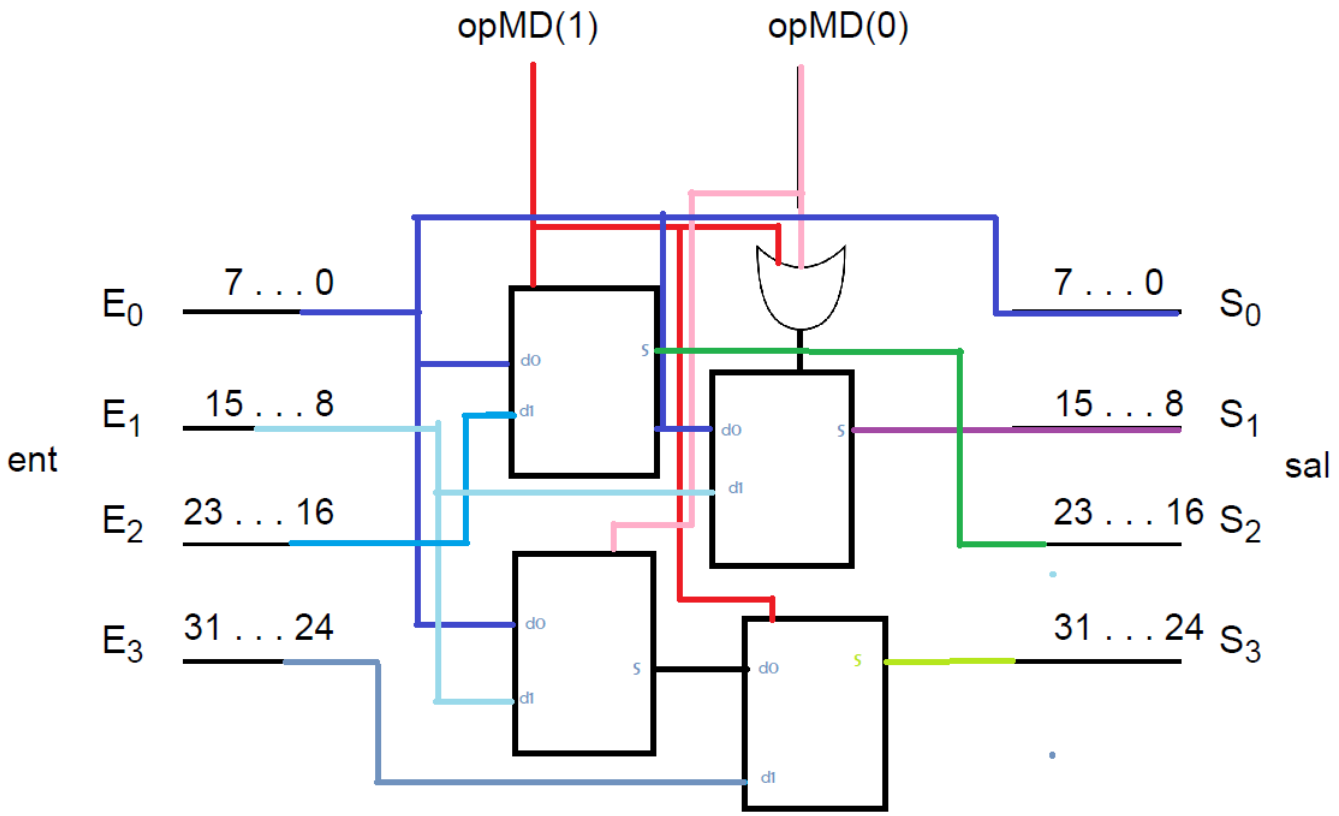


Figura 13: Esquema RTL del componente alinea de FMTE

7. Pregunta 7

Proponga un diseño alternativo del módulo alinearE que utilice únicamente los 2 bits menos significativos de la señal de control opMD para formatear el dato de escritura a memoria. La ubicación de los ficheros relacionados con este proyecto (ALINEAE.qpf) se indica en la página 298. En primer lugar rellene la tabla que relaciona los datos de entrada y salida del formateador en función de los dos bits menos significativos de la dirección. Minimice el número de niveles de selección y el número de multiplexores de 2 entradas.

En este caso a diferencia de los dos ejercicios anteriores hemos sacado la tabla a partir de la información de opMD, una vez conseguida la tabla, hemos realizado el circuito.



opMD	sal			
1..0	S3	S2	S1	S0
0 0	E0	E0	E0	E0
0 1	E1	E0	E1	E0
1 0	E3	E2	E1	E0
1 1	x	x	x	x

8. Pregunta 8

Obtenga las siguientes métricas cuando el procesador base ejecuta el programa euclides: instrucciones ejecutadas, aritmético-lógicas, load, store, secuenciamiento condicional e incondicional. Para ello, añada un proceso al programa de prueba (prueba_proc_MD_MI.vhd). Este proceso debe utilizar únicamente las señales de control opALU, opMD y opSEC para determinar el tipo de instrucción interpretada en cada ciclo. Adjunte el código vhdl del proceso.

8.1. Código

Para sacar el numero de instrucciones de cada tipo hemos creado un componente dentro del fichero de pruebas del procesador serie. Para el número de instrucciones totales, simplemente hemos sumado el total de cada tipo de instrucción con tal de no añadir más cosas al proceso. El código añadido es el siguiente:

```
procesoInstrucciones: process is
        variable instrStore : integer := 0;
        variable instrLoad : integer := 0;
        variable instrAritmLog : integer := 0;
        variable instrSCond : integer := 0;
        variable instrSIncond : integer := 0;

begin
        wait until reloj'event and reloj ='1';

        -- Esto corresponderá a las instrucciones load (LB, LH, LW, LBU i LHU).
        if (s_opMD(4) = '1' and s_opMD(3) = '0') then instrLoad := instrLoad + 1;

        -- Esto corresponderá a las instrucciones store (SB, SH, SW).
        elsif (s_opMD(4) = '1' and s_opMD(3) = '1') then instrStore := instrStore + 1;

        -- Esto corresponderá a las instrucciones Aritmetico Logicas
        elsif (s_opALU = ALU_ADD or
               s_opALU = ALU_SUB or
               s_opALU = ALU_SLL or
               s_opALU = ALU_SLT or
               s_opALU = ALU_SLTU or
               s_opALU = ALU_XOR or
               s_opALU = ALU_SRL or
               s_opALU = ALU_SRA or
               s_opALU = ALU_OR or
               s_opALU = ALU_AND) then instrAritmLog := instrAritmLog + 1;

        -- Esto corresponderá a las instrucciones de segmentacion condicional
        elsif (s_opSEC = DECS_BEQ or
               s_opSEC = DECS_BNE or
               s_opSEC = DECS_BLT or
               s_opSEC = DECS_BGE or
               s_opSEC = DECS_BLTU or
```

```
35                 s_opSEC = DECS_BGEU) then instrSCond := instrSCond  + 1;
36
37         -- Esto corresponderá a las instrucciones de segementación incondicionada
38         elsif (s_opSEC(3)) then instrSIncond := instrSIncond + 1;
39
40         end if;
41
42         report " Instrucciones Aritmetico Logicas: " & integer'image(instrAritmLog)
43         & " Instruciones Load : " & integer'image(instrLoad)
44         & " Instruciones Store: " & integer'image(instrStore)
45         & " Instruciones Secuenciamiento Condicional: " & integer'image(instrSCond)
46         & " Instruciones Secuenciamiento Incondicional: " & integer'image(instrSIncond);
47
48 end process procesoInstrucciones;
```

Para que este código funcione ha sido necesario añadir la librería *use work.cte_tipos_UF_pkg.all;* para poder usar todos los valores directamente de la ALU usando las constantes.

8.2. Instrucciones

Instrucciones TOTALES	103	Instrucciones Aritmetico-Logicas	66
Instrucciones Load	2	Instrucciones Store	3
Instrucciones secuenciamiento condicional	25	Instrucciones secuenciamiento incondicional	7