Universitat Politècnica de Catalunya

Arquitectura de computadors d'altes prestacions

Práctica 6

Procesador: segmentación lineal con cortocircuitos

Carlota Catot
Miguel Antunez

Grupo 6

Quatrimestre primavera 2020-2021

Índice

1. Pregunta 1 - Módulo LATPROH **3**
1.1. Diseño . . . 3
1.2. Justificación . . . 3

2. Pregunta 2 - Módulo LDC **4**
2.1. Diseño . . . 4
2.2. Justificación . . . 4

3. Pregunta 3 - Módulo LDRD **5**
3.1. Diseño . . . 5
3.2. Justificación . . . 5

4. Pregunta 4 - Módulos LDC, Latproh y LDRD en Quartus **6**
4.1. Módulo LDC . . . 6
4.1.1. Esquema RTL . . . 6
4.1.2. Código . . . 7
4.2. Módulo LATPROH . . . 7
4.2.1. Esquema RTL . . . 7
4.2.2. Código . . . 7
4.3. Módulo LDRD . . . 8
4.3.1. Esquema RTL . . . 8
4.3.2. Código . . . 8

5. Pregunta 5 **9**

6. Pregunta 6 - Cronograma y tiempos **10**
6.1. Cronograma . . . 10
6.2. Tiempos . . . 11
6.3. Justificación . . . 11

7. Pregunta 7 - programa char_sort **12**
7.1. Procesador segmentado CON cortocircuitos . . . 12
7.2. Procesador segmentado SIN cortocircuitos . . . 12
7.3. Ganancia . . . 12
7.4. Justificación . . . 12

8. Pregunta 8 - Módulo LDRS **13**
8.1. Diseño . . . 13
8.2. Justificación . . . 13
8.3. Esquema RTL . . . 13
8.4. Código . . . 14
8.5. Tabla tiempos programa fact_recurs . . . 14

9. Pregunta 9 - Módulo ERSEC **15**
9.1. Diseño . . . 15
9.2. Justificación . . . 15

10. Pregunta 10 - Módulo ERRELL **16**
10.1. Diseño 16
10.2. Justificación 16

11. Pregunta 11 - Módulo LDRS **17**
11.1. Diseño 17
11.2. Justificación 17

12. Pregunta 12 - Módulo LGR **18**
12.1. Diseño 18
12.2. Justificación 18

13. Pregunta 13 - programas euclides y sort **19**

14. Pregunta 14 - mecanismos predicción **20**
14.1. Seguir en secuencia 20
14.2. Modificar secuenciamiento 20

1. Pregunta 1 - Módulo LATPROH

Diseñe el módulo LATPROH utilizando el menor número posible de registros y puertas lógicas, limitando el número de entradas de las puertas a 2. Justifique el diseño de forma sucinta y sistemática.

1.1. Diseño

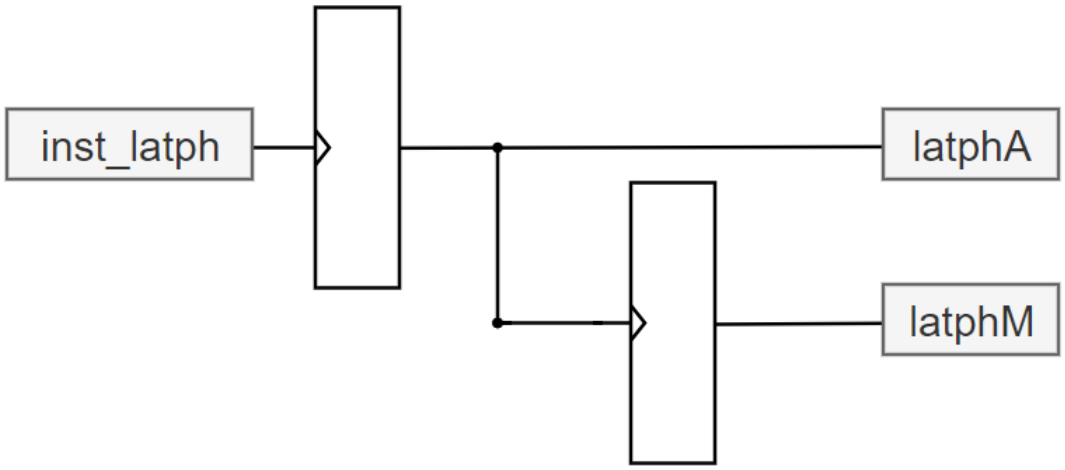


Figura 1: Esquema logico modulo LATPROH

1.2. Justificación

Como se especifica en el enunciado, el módulo LATPROH es el encargado de la propagación de la información identificado en el submódulo *inst_latptob* a las diferentes etapas. Este genera señales para indicar (durante la interpretación del código) si hay alguna instrucción que puede producir un riesgo de datos debido a registros.

Para la etapa ALU encontramos la salida latphA y para la etapa M encontramos la salida latphM.

Se necesitan dos registros porque la señal que indica posibles riesgos de datos se produce en la etapa DL y por tanto para latphA se introduce 1 registro ya que se espera 1 ciclo (DL) y para latphM se introducen 2 registros ya que se ha de esperar 2 ciclos (DL y ALU).

2. Pregunta 2 - Módulo LDC

Diseñe el módulo LDC utilizando el menor número posible de registros y puertas lógicas, limitando el número de entradas de las puertas a 2 (Lógica de cortocircuitos e interbloqueos en la página 415). Justifique el diseño de forma sucinta y sistemática.

2.1. Diseño

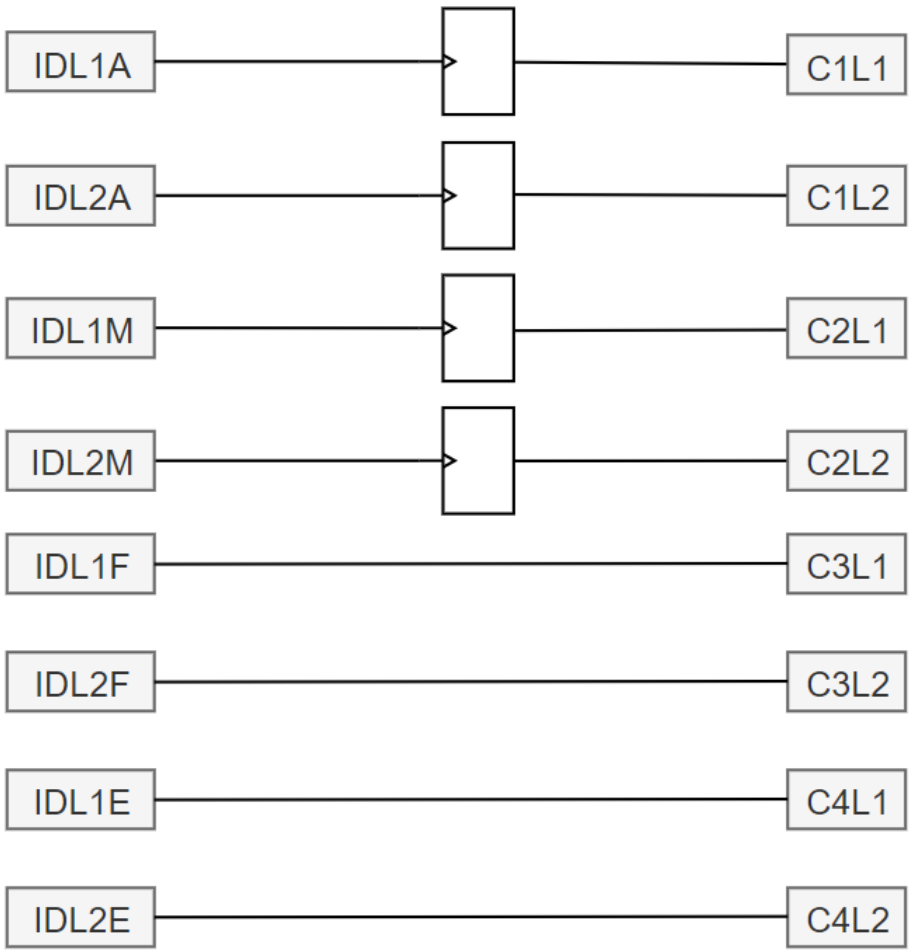


Figura 2: Esquema logico modulo LDC

2.2. Justificación

Como se especifica en el enunciado, el módulo LDC (Lógica de decisión para el control de cortocircuitos) es el encargado de activar los cortocircuitos en caso necesario. La generación de las señales para controlar estos cortocircuitos se efectuará en la etapa DL, pero se pueden usar también en la etapa ALU.

Los registros se colocan para indicar el cortocircuito y de esta manera se retrasa 1 ciclo la salida de manera que estén en la etapa correcta en el siguiente ciclo.

3. Pregunta 3 - Módulo LDRD

Diseñe el módulo LDRD utilizando el menor número posible de puertas lógicas, limitando el número de entradas de las puertas a 2. Justifique el diseño de forma sucinta y sistemática.

3.1. Diseño

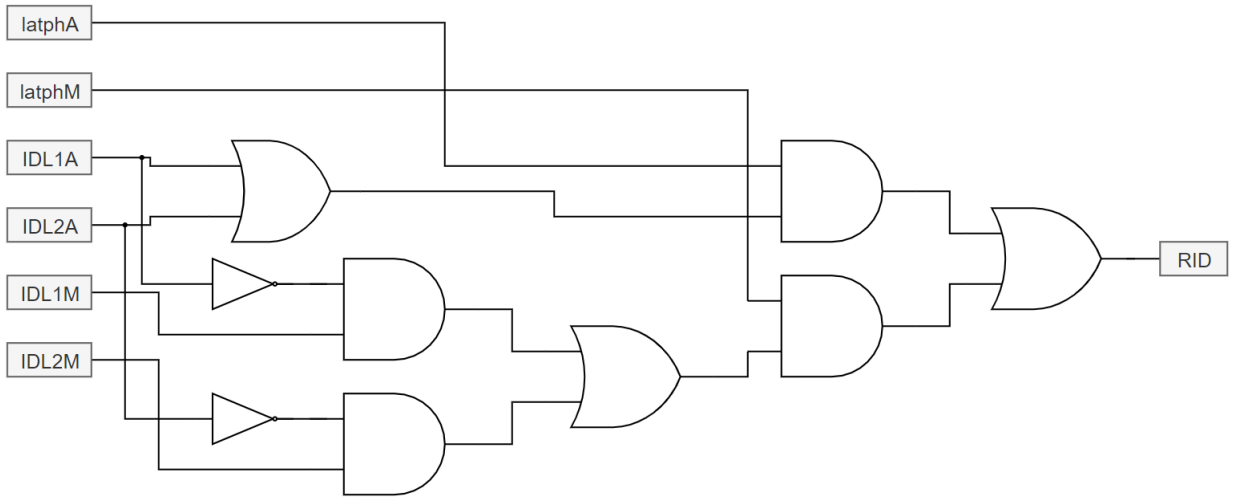


Figura 3: Esquema logico modulo LDRD

3.2. Justificación

Como se especifica en el enunciado, el módulo LDRD es el encargado de la lógica de detección de riesgos de datos debido a registros. Por lo tanto, existirá riesgo en el caso de que el identificador del registro fuente coincida con los de la etapa M y no con los de la etapa ALU.

4. Pregunta 4 - Módulos LDC, Latproh y LDRD en Quartus

En el subdirectorio LIB se encuentran los ficheros asociados al diseño de la lógica de interbloqueos. Describa en VHDL los 3 módulos anteriores (LDC.vhd, latproh.vhd y LDRD.vhd), utilizando un modelo estructural. Entregue los esquemas RTL de los módulos elaborados por Quartus. Compruebe el diseño la Lógica de Cortocircuitos e Interbloqueos. El programa de prueba suministrado compara a cada ciclo las salidas de los 3 módulos diseñados con los respectivos modelos de referencia correctos.

Para cada módulo se presenta el esquema RTL juntamente con el código implementado para poder sacar dicho esquema.

4.1. Módulo LDC

4.1.1. Esquema RTL

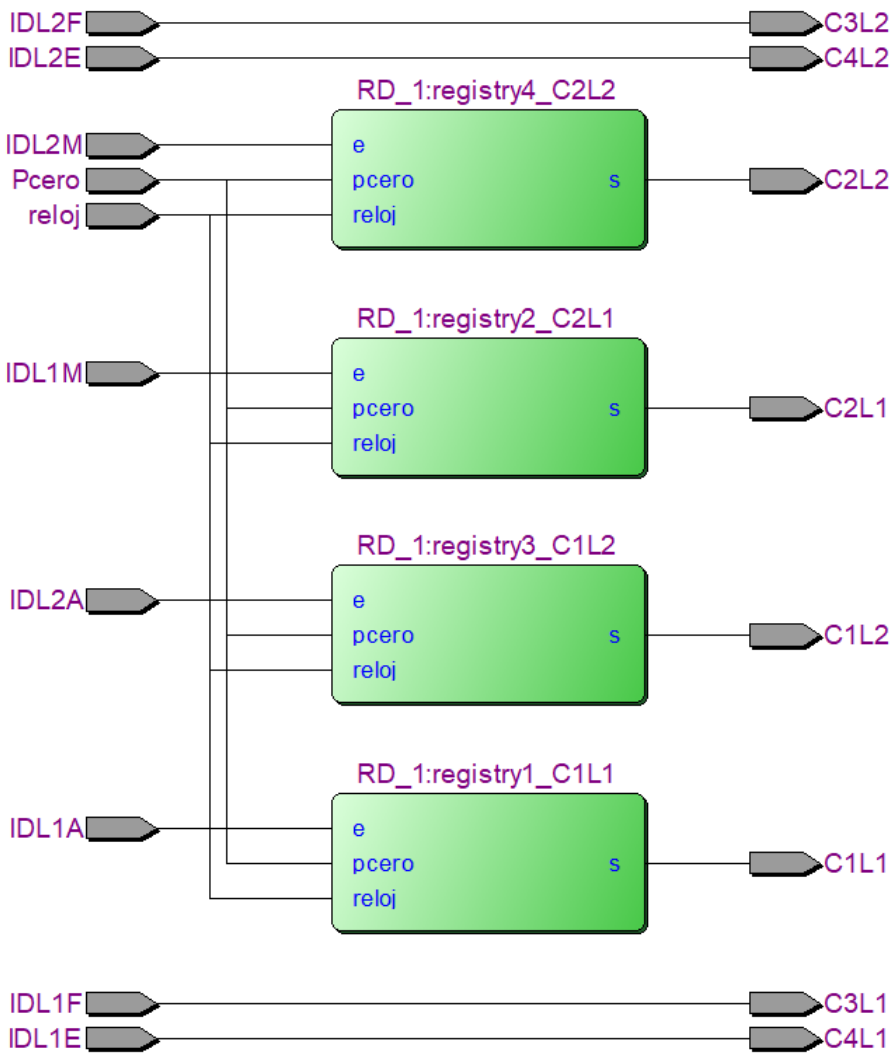


Figura 4: Esquema RTL del módulo LDC

4.1.2. Código

```
architecture estructural of LDC is

begin
    registry1_C1L1: RD_1 port map(reloj=>reloj,Pcero=>Pcero,e=>IDL1A  ,s=>C1L1);
    registry3_C1L2: RD_1 port map(reloj=>reloj,Pcero=>Pcero,e=>IDL2A ,s=>C1L2);
    registry2_C2L1: RD_1 port map(reloj=>reloj,Pcero=>Pcero,e=>IDL1M  ,s=>C2L1);
    registry4_C2L2: RD_1 port map(reloj=>reloj,Pcero=>Pcero,e=>IDL2M  ,s=>C2L2);

    C3L1 <= IDL1F after retLDC;
    C3L2 <= IDL2F after retLDC;
    C4L1 <= IDL1E after retLDC;
    C4L2 <= IDL2E after retLDC;
end;
```

4.2. Módulo LATPROH

4.2.1. Esquema RTL

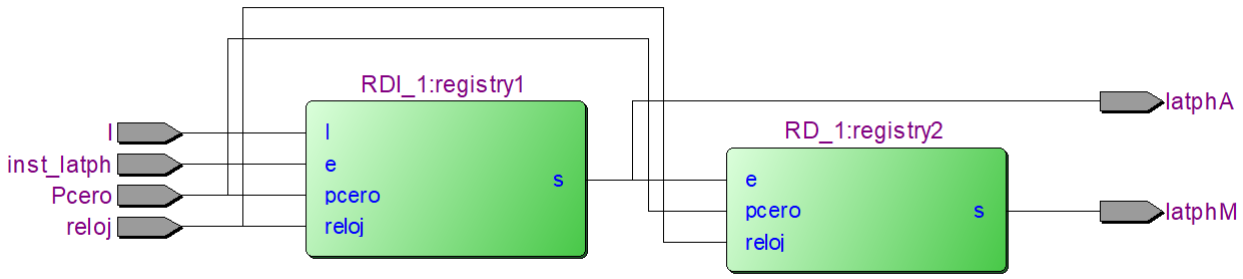


Figura 5: Esquema RTL del módulo LATPROH

4.2.2. Código

```
architecture estruc of latproh is
signal s_regstry1: std_logic;

begin
    registry1: RDI_1 port map(reloj => reloj, pCero=> pCero, I=>I,e=>inst_latph,s=>s_regstry1);
    registry2: RD_1 port map(reloj => reloj, pCero=> pCero,e=>s_regstry1,s=>latphM);
    latphA <= s_regstry1;
end;
```

4.3. Módulo LDRD

4.3.1. Esquema RTL

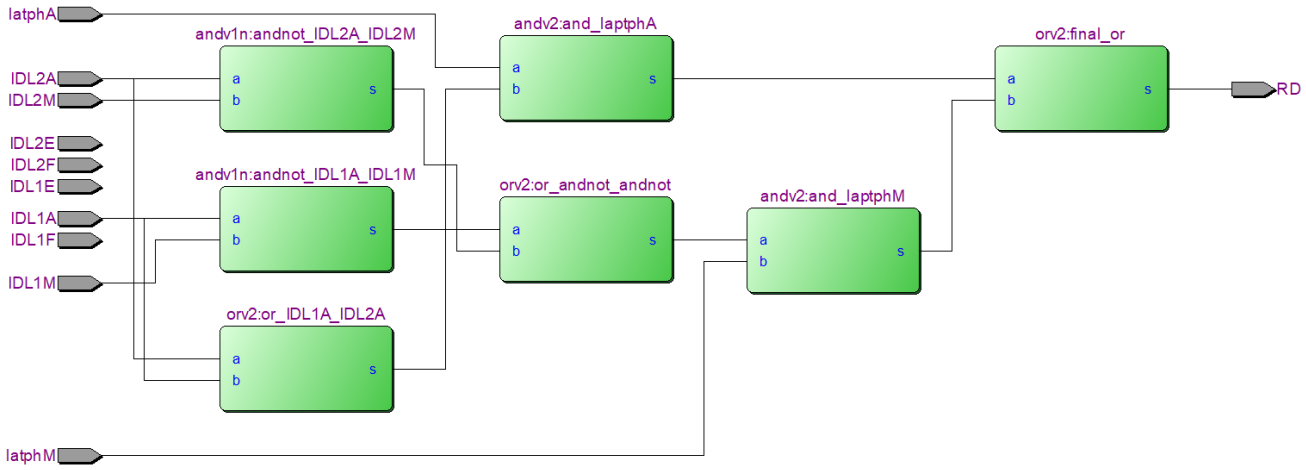


Figura 6: Esquema RTL del módulo LDRD

4.3.2. Código

```
architecture comportamiento of LDRD_C is

signal s_andnot_IDL1A_IDL1M: std_logic;
signal s_andnot_IDL2A_IDL2M: std_logic;
signal s_or_IDL1A_IDL2A: std_logic;
signal s_or_andnot_andnot: std_logic;
signal s_and_laptphA: std_logic;
signal s_and_laptphM: std_logic;
signal s_or: std_logic;

begin
    andnot_IDL1A_IDL1M: andv1n port map(a => IDL1A, b => IDL1M, s => s_andnot_IDL1A_IDL1M);
    andnot_IDL2A_IDL2M: andv1n port map(a => IDL2A, b => IDL2M, s => s_andnot_IDL2A_IDL2M);
    or_IDL1A_IDL2A: orv2 port map(a => IDL2A, b => IDL1A, s => s_or_IDL1A_IDL2A);
    or_andnot_andnot: orv2 port map(a => s_andnot_IDL1A_IDL1M, b => s_andnot_IDL2A_IDL2M,
      s => s_or_andnot_andnot);
    and_laptphA: andv2 port map(a => latphA, b => s_or_IDL1A_IDL2A, s => s_and_laptphA);
    and_laptphM: andv2 port map(a => s_or_andnot_andnot, b => latphM, s => s_and_laptphM);
    final_or: orv2 port map(a => s_and_laptphA, b => s_and_laptphM, s => s_or);

    RD <= s_or after retLDRD_C;
end comportamiento;
```

5. Pregunta 5

La siguiente tabla relaciona las entradas y las salidas de la lógica de interbloqueos LIBC durante 8 ciclos consecutivos. Suponga que en el ciclo 1 la etapa DL está ocupada por la instrucción "lw x3, 0(x1)" y que las etapas posteriores procesan datos inválidos. Deduzca una posible secuencia de las instrucciones que ocupan la etapa DL durante los siguientes 7 ciclos.

		Salidas control Cortocircuitos								Salidas LGR			
										CP	BDL		DLA
ciclo	instrucción DL	C1L1	C2L1	C3L1	C4L1	C1L2	C2L2	C3L2	C4L2	bloq	bloq	inyec	inyec
1	lw x3, 0(x1)	0	0	0	0	0	0	0	0	0	0	0	0
2	Jarl x4, 0(x3)	0	0	0	0	0	0	0	0	1	1	0	1
3	Jarl x4, 0(x3)	1	0	0	0	0	0	0	0	1	1	0	1
4	Jarl x4, 0(x3)	0	1	1	0	0	0	0	0	0	0	1	0
5	NOP	0	0	0	0	0	0	0	0	0	0	1	0
6	NOP	0	0	0	0	0	0	0	0	0	0	0	0
7	add x2, x4, x3	0	0	1	0	0	0	0	0	0	0	0	0
8	add x4, x5, x4	0	0	0	0	0	0	0	1	0	0	0	0

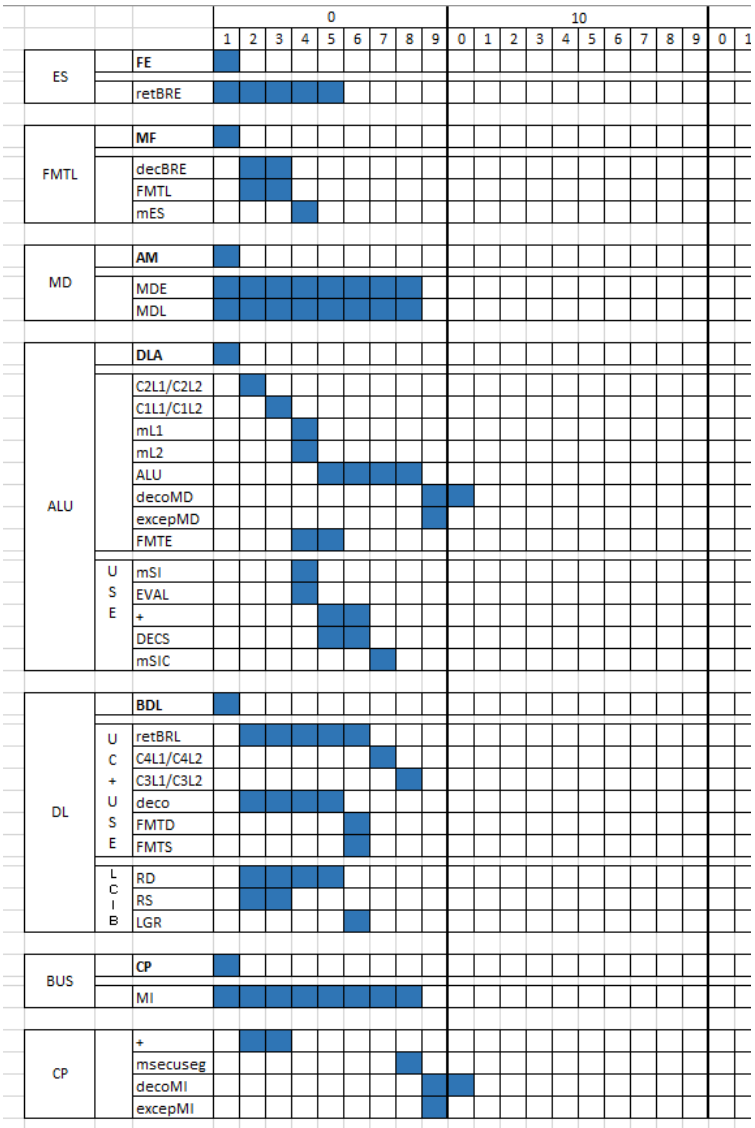
Para poder sacar las instrucciones especificadas en la tabla, se han analizado los valores que se indican en la instrucción anterior para poder obtener la siguiente instrucción.

En primer lugar las 3 primeras instrucciones son de secuenciamiento incondicional, son 3 ya que como podemos ver se bloquea el CP y el BDL en las instrucciones anteriores i por ese motivo se arrastra. En el ciclo 4 podemos ver como en la salida BDL se indica como se inyectan NOPs, por eso motivo los ciclos 5 y 6 corresponden a NOPs, por último, y teniendo en cuenta que registros coger para tener en cuenta los riesgos y los cortocircuitos añadimos dos operaciones aritmético-lógicas.

6. Pregunta 6 - Cronograma y tiempos

Para facilitar el cálculo del tiempo de ciclo utilizaremos un cronograma donde se representa el retardo de cada elemento del camino de datos. En el cronograma se han tenido en cuenta las etapas para agrupar los componentes del camino de datos. En la tabla siguiente se muestran los acrónimos utilizados para denominar los componentes del camino de datos. Todos deben interpretarse como el retardo del componente. Algunos componentes están incluidos dentro de un módulo que se visualiza, por ejemplo, el decodificador del identificador del registro cuando se escribe en un registro del banco de registros. Los retardos de los componentes utilizados se detallan en los apéndices de la documentación denominados "Retardos", excepto los componentes de la lógica de interbloqueos LCIB. Estos retardos no son representativos de un diseño. Solo son de utilidad para efectuar los cálculos de retardo que se soliciten. Para los componentes de la LCIB suponga los retardos indicados en la tabla.

6.1. Cronograma



6.2. Tiempos

Etapa	**CP**	**B**	**DL**	**A**	**M**	**F**	**E**
Tiempo de etapa (ns)	10	8	8	10	8	4	5

Tiempo de ciclo (ns)	10 ns

6.3. Justificación

Para poder sacar el cronograma se ha usado la información del cronograma expuesto en la práctica 5 juntamente con la información expuesta en el enunciado y en la hoja de problemas.

En cuanto a los tiempos, se ha contado en el cronograma para poder sacar cuantos ns tarda cada etapa. Para el tiempo de ciclo se ha cogido el valor más alto ya que será el caso peor. Ya que para los valores más pequeños no afecta al tiempo de ciclo el hecho de que sobren nanosegundos, lo importante es que todos los caso esten contemplados.

7. Pregunta 7 - programa char_sort

Utilice el programa char_sort. Añada un proceso al programa de prueba para obtener las métricas indicadas en la tabla. Compare el rendimiento del procesador con cortocircuitos respecto del procesador segmentado sin cortocircuitos.

7.1. Procesador segmentado CON cortocircuitos

Ciclos perdidos por riesgo de datos	380
Ciclos perdidos por r.secuenciamiento	664
Ciclos de ejecución	2307
Tiempo de ciclo (ns)	10 ns

7.2. Procesador segmentado SIN cortocircuitos

Ciclos perdidos por riesgo de datos	1940
Ciclos perdidos por r.secuenciamiento	664
Ciclos de ejecución	3939
Tiempo de ciclo (ns)	8 ns

7.3. Ganancia

$$Ganancia = \left(1 - \frac{ciclos_{sin} * tc_{sin}}{ciclos_{con} * tc_{con}}\right) * 100 = \left(1 - \frac{3939 * 8}{2307 * 10}\right) * 100 = 36.60\%$$

7.4. Justificación

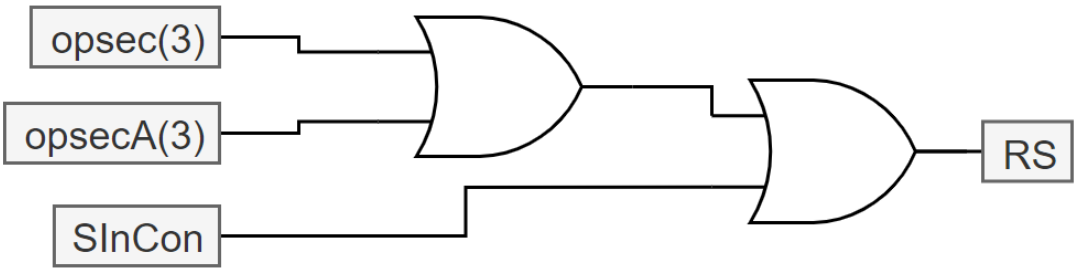
Para calcular todos los datos se han usado los códigos de las practicas 4 y 5, con ello sacaremos los datos de los apartados 7.1 y 7.2.

Para la ganancia usaremos la formula presentada en el apartado 7.3 comparando los ciclos y el tiempo de ciclo de ambos casos.

8. Pregunta 8 - Módulo LDRS

Considere la unidad de secuenciamiento con reducción de la latencia de la instrucción "jal". Diseñe el módulo LDRS utilizando el menor número posible de puertas lógicas, limitando el número de entradas de las puertas a 2. Justifique el diseño de forma sucinta y sistemática.

8.1. Diseño

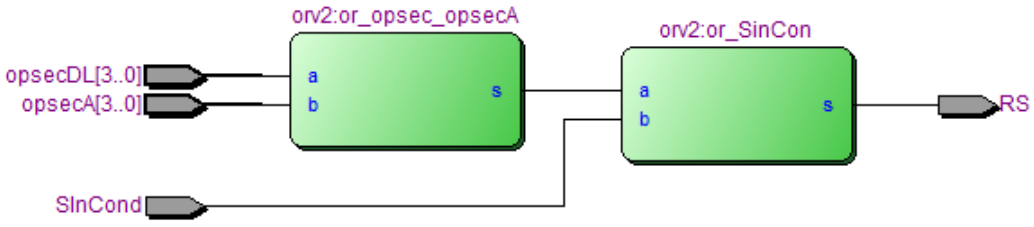


8.2. Justificación

Como especifica el enunciado, el módulo LDRS se encarga de la lógica de detección de riesgos de secuenciamiento.

Para poder detectar este riesgo usaremos las entradas opsec(3), opsecA(3) y SinCon y en el caso de que una de ellas sea 1 ya se detecta riesgo de secuenciamiento. Es por eso que si hay una instrucción de secuenciamiento incondicional en DL o en ALU ya se detecta este riesgo.

8.3. Esquema RTL



8.4. Código

```
architecture comportamiento of LDRS_ModSecu_relI is

signal s_or_opsec_A: std_logic;
signal s_or_SinCon: std_logic;

begin
    or_opsec_opsecA: orv2 port map (a => opsecDL(3), b => opsecA(3), s => s_or_opsec_A);
    or_SinCon: orv2 port map (a => s_or_opsec_A, b => SinCond, s => s_or_SinCon);
    RS <= s_or_SinCon after retLDRS;
end comportamiento;
```

8.5. Tabla tiempos programa fact_recurs

	Ciclos perdidos por riesgo de secuenciamiento		
Programa	Antes de reducir la latencia	Una vez reducida la latencia	Reduccion (%)
fact_recurs	232	223	3.88 %

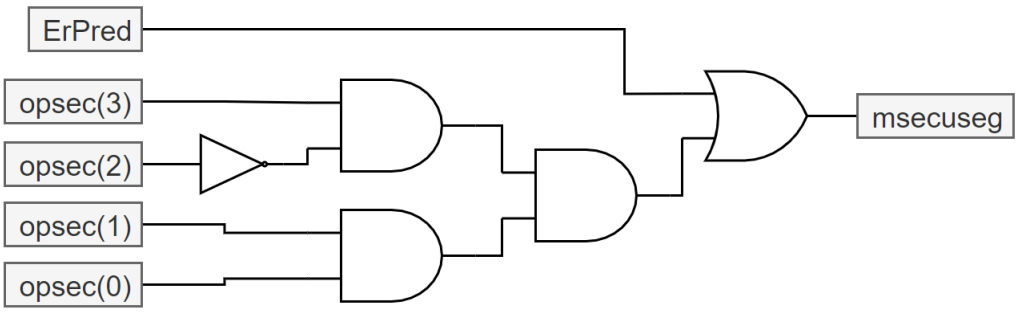
Para calcular los ciclos de riesgo de secuenciamiento se ha programado un proceso que se ha ejecutado en el programa prueba con tal de contar todos los ciclos perdidos. Para calcular los ciclos perdidos antes de reducir la latencia se ha ejecutado el proyecto *PROC_SEC_CORTOS* mientras que para conocer los ciclos perdidos una vez reducida la latencia se ha ejecutado en el proyecto *PROC_SEG_ModSecu_relI*. Para conocer la reducción se ha aplicado la siguiente fórmula:

$$Reducción = \left(1 - \frac{n.ciclos\ con\ reducción\ latencia}{n.ciclos\ antes\ reducción\ latencia}\right) * 100 = \left(1 - \frac{223}{232}\right) * 100 = 3.88\%$$

9. Pregunta 9 - Módulo ERSEC

Diseñe el módulo Ersec utilizando el menor número posible de puertas lógicas, limitando el número de entradas de las puertas a 2. Justifique el diseño de forma sucinta y sistemática.

9.1. Diseño



9.2. Justificación

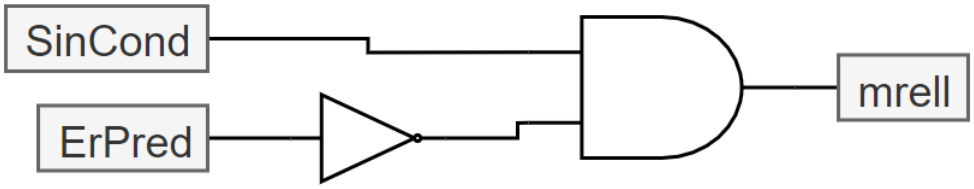
Como se especifica en el enunciado, el módulo ERSEC forma parte de la logica de control de camino de datos. En concreto este módulo se encarga de discernir entre instrucciones de secuenciamiento condicional y la instrucción "jarl", ya que para las instrucciones de secuenciamiento condicional solo se modifica el secuenciamiento si hay un error de predicción.

En el diagrama vemos que tenemos varias entradas, la entrada opsec que corresponde a la operacion de secuenciamiento y ErPred que indica si hay error en la predicción. Con la circuiteria de opSec se controla si es una operación de secuenciamiento incondicional, ya que la salida de este modulo sera '1' en caso de que la operacion de secuenciamiento sea incondicional o haya un error de predicción.

10. Pregunta 10 - Módulo ERRELL

Diseñe el módulo Errel utilizando el menor número posible de puertas lógicas, limitando el número de entradas de las puertas a 2. Justifique el diseño de forma sucinta y sistemática.

10.1. Diseño



10.2. Justificación

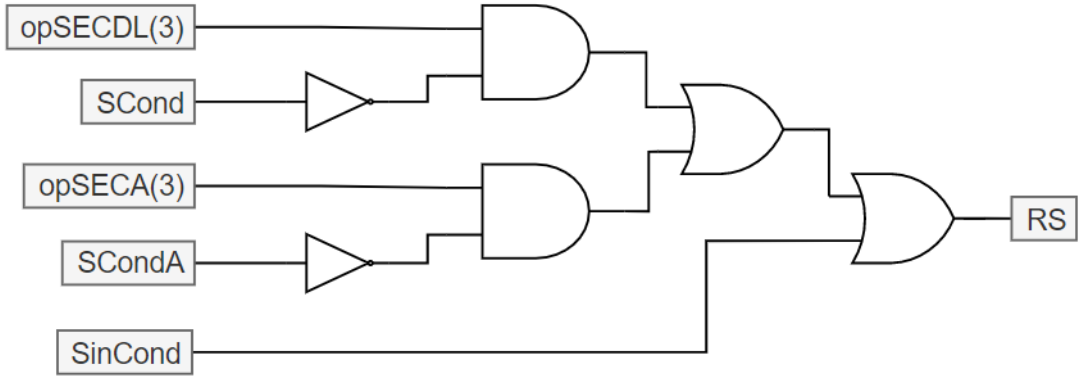
Como se especifica en el enunciado, el módulo ERRELL, al igual que el módulo ERSEC, forma parte de la logica de control de camino de datos. En concreto este módulo se encarga de anular el secuenciamiento que pueda establecer una instrucción "jar"mal predicha.

Es por eso que SinCond y ErPred han de tener valores opuestos para poder anular el secuenciamiento.

11. Pregunta 11 - Módulo LDRS

Diseñe el módulo LDRS utilizando el menor número posible puertas lógicas, limitando el número de entradas de las puertas a 2. Justifique el diseño de forma sucinta y sistemática.

11.1. Diseño



11.2. Justificación

Como se especifica en el enunciado, el módulo LDRS solo debe activar la señal RS en el caso de que exista un riesgo de secuenciamiento.

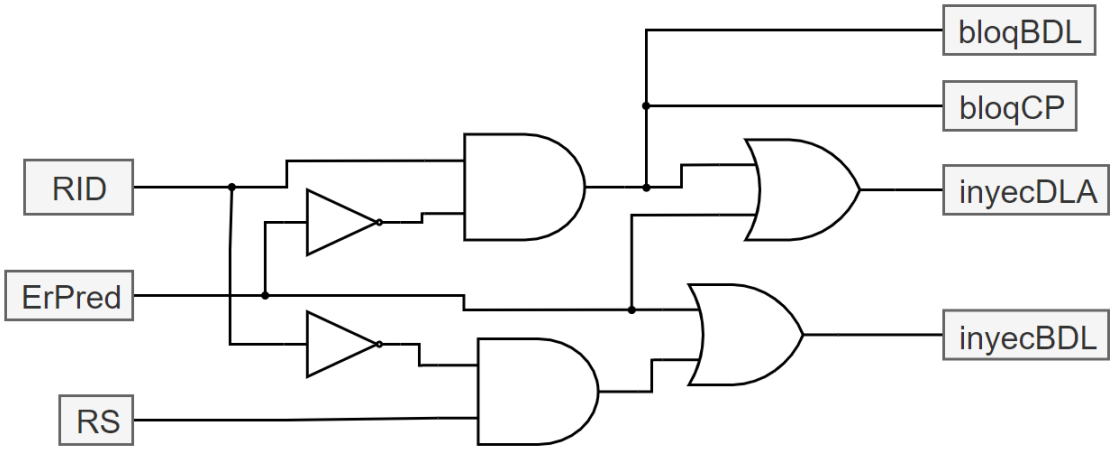
La circuitería de este módulo tiene varias entradas, para las etapas DL y ALU podemos ver la operación de secuenciamiento y el secuenciamiento condicional. Asi como también la entrada SinCond.

Para las etapas DL y ALU el comportamiento será el mismo ya que en el caso de que la operación sea de solo sera '1' en caso de que la operación sea de secuenciamiento incondicional, y en el caso de que una de los dos lo sea, ya se activará la señal RS. Así como también se activará si la entrada SinCond es '1'.

12. Pregunta 12 - Módulo LGR

Diseñe el módulo LGR utilizando el menor número posible puertas lógicas, limitando el número de entradas de las puertas a 2. Justifique el diseño de forma sucinta y sistemática.

12.1. Diseño



12.2. Justificación

Como se especifica en el enunciado, el módulo LGR es un modulo de actuación o riesgo que se encarga de la lógica de gestión de riesgos. A partir de las señales de riesgo de datos y riesgo de secuenciamiento genera las señales de control para los registros de desacoplo. En otras palabras se controla cuando bloquear y cuando inyectar nops.

Las salidas de bloqueo tanto para la etapa BDL como para la etapa CP se calculan de la misma manera, se activará en el caso de que no haya error de predicción y haya riesgo de datos.

Para las salida de inyección de la etapa DLA se usará la entrada RID, se activará en el caso de que el haya bloqueo en CP o BLD o haya error de predicción.

Para las salida de inyección de la etapa BDL usará la entrada RS, se activará en el caso de que haya error de predicción o en el caso de que no haya riesgo de datos y si riesgo de secuenciamiento.

13. Pregunta 13 - programas euclides y sort

Indique para los siguientes programas la reducción en ciclos perdidos por riesgos de secuenciamiento cuando se predice seguir en secuencia. Para ello modifique el programa de pruebas. Indique también el número de predicciones y el número de errores de predicción.

Programa	Una vez reducida la latencia (jar)	Una vez reducida la latencia y predicción de seguir en secuencia	Reduccion	Predicciones	Errores
Euclides	58	32	44.83 %	25	12
Sort	669	506	27.61 %	348	246

Para sacar los valores de ciclos perdidos por secuenciamiento, una vez reducida la latencia, hemos ejecutado el mismo código que en el ejercicio 8 para conocer los ciclos perdidos tanto para el programa euclides como para sort.

Para los valores de ciclos perdidos por secuenciamiento una vez reducida la latencia y predicción de seguir en secuencia hemos usado el valor que nos da y lo hemos sumado (8 y 14) a los errores multiplicado por 2 ya que como dicen los apuntes, en cada error se pierden dos ciclos.

Para la reducción hemos aplicado la formula igual que en el ejercicio 8 para sacar los porcentajes.

Para las predicciones y los errores hemos generado un código para el fichero de pruebas que detecte predicciones y después compruebe si han fallado o no.

14. Pregunta 14 - mecanismos predicción

Observe que los dos mecanismos de predicción fija descritos utilizan la misma unidad de secuenciamiento, solo cambia el control de los multiplexores de encaminamiento de direcciones. Indique, para cada mecanismo de predicción, qué multiplexores de la USE se podrían eliminar.

Predicción	Mx a eliminar
Fija seguir en secuencia	MSIC
Fija modificar secuenciamiento	MSI

14.1. Seguir en secuencia

Si queremos seguir en secuencia y basandonos en el diagrama del enunciado pagina 438, podemos prescindir del multiplexor MSIC ya que siempre se ha de pasar por msi para comprovar el literal, por lo tanto como msic siempre será 1 podemos prescindir de el.

14.2. Modificar secuenciamiento

Si queremos modificar el secuenciamiento y basandonos en el diagrama del enunciado pagina 442, al no tener que comprovar nunca el literal ya que siempre se salta, siempre pasara por la entrada 0 del multiplexor, es decir que nunca pasara por msi y por tanto podemos prescindir de el.