Universitat Politècnica de Catalunya

Arquitectura de Computadores de Altas Prestaciones

**Práctica 5**

**Procesador: segmentación lineal, camino**

**de datos y control**

Alumnos:

Manuel Velastegui

Carlos Andres Rodríguez Torres

**Grupo 5**

Cuatrimestre Primavera 2024-2025



**ÍNDICE**

[**1. Pregunta 1 3**](#_qeiifw3jmnqt)

[1.1 Módulo LDD 3](#_ky37df5a3mv4)

[1.2 Justificación 3](#_co22suegaet2)

[**2. Pregunta 2 4**](#_vjdgcayynziq)

[2.1 Módulo LDRD 4](#_d0ajvptpia1p)

[2.2 Justificación 4](#_gtw0bfe2t3cz)

[**3. Pregunta 3 5**](#_ncenpkl1z5b)

[3.1 Módulo LDRS 5](#_sjozf12hzalz)

[3.2 Justificación 5](#_h4d84hna44n9)

[**4. Pregunta 4 5**](#_s4qhexlnev3q)

[4.1 Módulo LGR 5](#_9959i8vtq64m)

[4.2 Justificación 5](#_aknvqjmx78jc)

[**5. Pregunta 5 6**](#_cqz5y91gs9zm)

[5.1 RTL 6](#_t2affzqw0u1q)

[5.2 Archivos 6](#_f7b6z2vlkl3c)

[5.2.1 LDD.vhd 6](#_wrfh0wntbu05)

[5.2.2 LDRD.vhd 7](#_ped2cbhxh869)

[5.2.3 LDRS.vhd 7](#_k4dlpd554h7t)

[5.2.4 LGRS.vhd 7](#_x2dky4wa3zk1)

[**6. Pregunta 6 8**](#_bt8jkipuhntu)

[6.1 Métricas 8](#_on6od15ygzjh)

[6.2 Justificación 8](#_93sj9kknznlc)

[**7. Pregunta 7 9**](#_9psquqwmfkeh)

[7.1 Ventana Temporal 9](#_4kiynwt624iv)

[7.2 Justificación 10](#_54ss2dzgjn7)

[**8. Pregunta 8 11**](#_u6ffqy51q4zh)

[8.1 Métricas 11](#_w3k1rygdfikd)

[8.2 Justificación 12](#_guatt2ch75zt)

# 

# 

# 

# 1. Pregunta 1

Diseñe el módulo LDD utilizando el menor número posible de puertas lógicas y comparadores (“Diseño de la lógica de interbloqueos“ en la página 328). Justifique el diseño de forma sucinta y sistemática.

## 1.1 Módulo LDD

*Figura 1. Lógica de interbloqueos*

## 1.2 Justificación

Este es el módulo propuesto para la detección de dependencia de datos:

- RD: Identificador de registro de destino de las instrucciones.

- RS1, RS2: Identificadores de registro fuente.

- VRS1, VRS2: Para indicar la validez de los identificadores RS

- PBR: Indica si el registro tiene permiso de escritura en el banco de registros.

- IDLXX: Indicamos si existe riesgo.

En función de la etapa en la que se encuentre se irá comparando el valor de RS con RD, se verificará si es válido RS, si es un registro diferente de 0, si existe permiso de escritura PBR. Si todo lo mencionado anteriormente se cumple, se podrá interpretar que existe un riesgo y se podrá identificar en la etapa que se ha detectado a través de la salida.

Ejemplo:

ADD R1, R2, R3 Rd = R1, RS1 = R2, RS2 = R3.

SUB R4, R1, R5 Rd = R4, RS1 = R1, RS2=R5.

Aquí podemos ver que el registro RS1 ES igual al registro Rd de la instrucción anterior. Por lo tanto, podríamos detectar el riesgo de datos en la etapa ALU.

# 2. Pregunta 2

Diseñe el módulo LDRD utilizando el menor número posible de puertas lógicas,

limitando el número de entradas a 2. Justifique el diseño de forma sucinta y sistemática.

## 2.1 Módulo LDRD

*Figura 2. Módulo Detección riesgo de datos*

## 2.2 Justificación

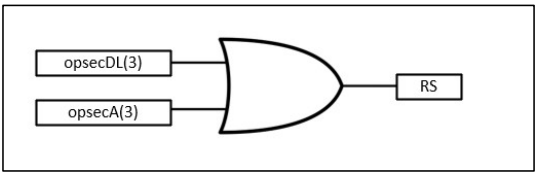
Es un módulo para detección de riesgos de datos. Por tanto, únicamente se deberá mirar si el módulo LDD (de la pregunta anterior) ha encontrado dependencia de datos en alguno de los registros. Para este módulo se trabaja con puertas OR ya que necesitamos tener una dependencia en el caso de que cualquiera de las señales generadas en el módulo LDD sea 1, por eso motivo se comprueba con puertas OR, en el caso de que cualquier valor de IDL será 1 RID ya será 1, ya que no nos importa en qué etapa se detecte, simplemente se ha de saber que se han de inyectar NOPs hasta que deje de haber dependencia de datos. El esquema se ha hecho de esta manera ya que el enunciado ha limitado el número de entradas a 2 pero se podrían usar menos puertas lógicas.

# 

# 3. Pregunta 3

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

## 3.1 Módulo LDRS



*Figura 3. Módulo Detección riesgo de secuenciamiento*

## 3.2 Justificación

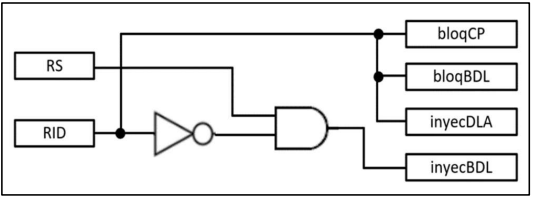
En este módulo nos encargaremos de la lógica de detección de riesgo de secuenciamiento. Disponemos de dos pines de entrada ya que solo nos interesa el valor de DL y ALU, ya que hasta la etapa DL no podemos saber si la instrucción que estábamos ejecutando es de secuenciamiento y en la etapa ALU se escribirá en CP, por lo que en la siguiente etapa ya se podrá empezar a ejecutar la instrucción correcta. Se mide con una OR ya que con que una de las dos entradas sea 1 ya se produce riesgo de secuenciamiento

# 4. Pregunta 4

Diseñe el módulo LGR utilizando el menor número posible de puertas lógicas, limitando

el número de entradas a 2. Justifique el diseño de forma sucinta y sistemática.

## 4.1 Módulo LGR



*Figura 4. Módulo Detección de riesgos*

## 4.2 Justificación

En este módulo, basándonos en el análisis de los módulos anteriores, se generarán las señales de control para los registros de desacoplo, teniendo en cuenta el riesgo de datos (LDRD) y el riesgo de secuenciamiento (LDRS). Las entradas son las siguientes: RS (proveniente del módulo LDRS donde se verifica el riesgo de secuenciamiento) y RID (proveniente del módulo LDRD donde se verifica el riesgo de datos). Si existe riesgo de datos (RID = 1), se establece en 1 el valor de bloqCP y bloqDL para bloquear el registro CP y el registro entre la etapa BUS y la etapa DL. Además, se asigna un valor de 1 a inyecDLA para inyectar "nops" desde la etapa DL hacia la siguiente etapa. Por otro lado, el valor de inyecBDL, que se utiliza para inyectar "nops" desde la etapa BUS hacia las etapas siguientes, solo será 1 en el caso de que exista riesgo de secuenciamiento y no riesgo de datos. Esto se explica mediante las puertas AND y NOT en el esquema lógico.

# 5. Pregunta 5

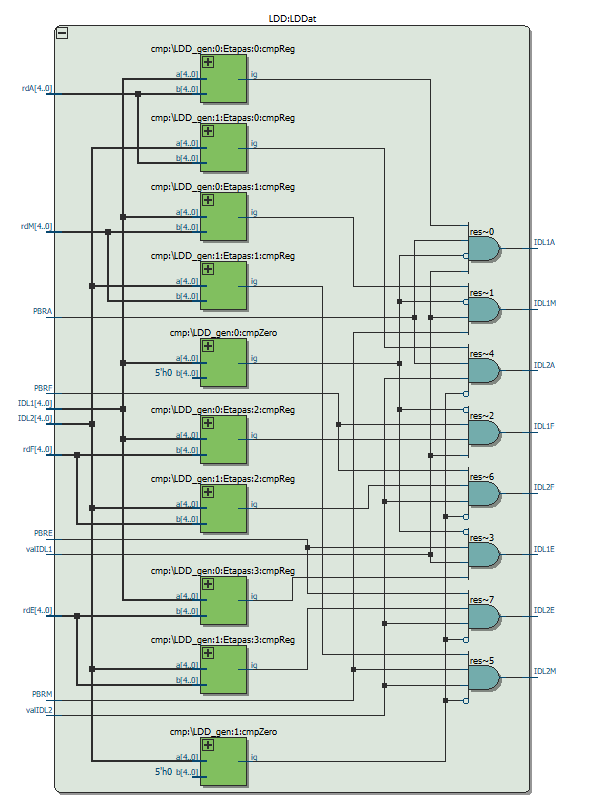
En el subdirectorio LIB (Apéndice 5.2) se encuentran los ficheros asociados al diseño de la Lógica de InterBloqueos (proyecto quartus LIB.qpf). Describa en VHDL los 4 módulos anteriores (ficheros LDD.vhd, LDRD.vhd, LDRS.vhd y LGR.vhd), utilizando un Modelo estructural. Los ficheros contienen la declaración de la interface. El módulo LDD tiene una estructura regular. Utilice sentencias generate en la descripción VHDL. Entregue el esquema RTL del módulo LDD elaborado por quartus.

Compruebe conjuntamente el diseño de los 4 módulos anteriores. El programa de prueba suministrado (prueba\_LIB.vhd) compara en cada ciclo las salidas de los módulos diseñados con los respectivos modelos de referencia correctos

## 5.1 RTL

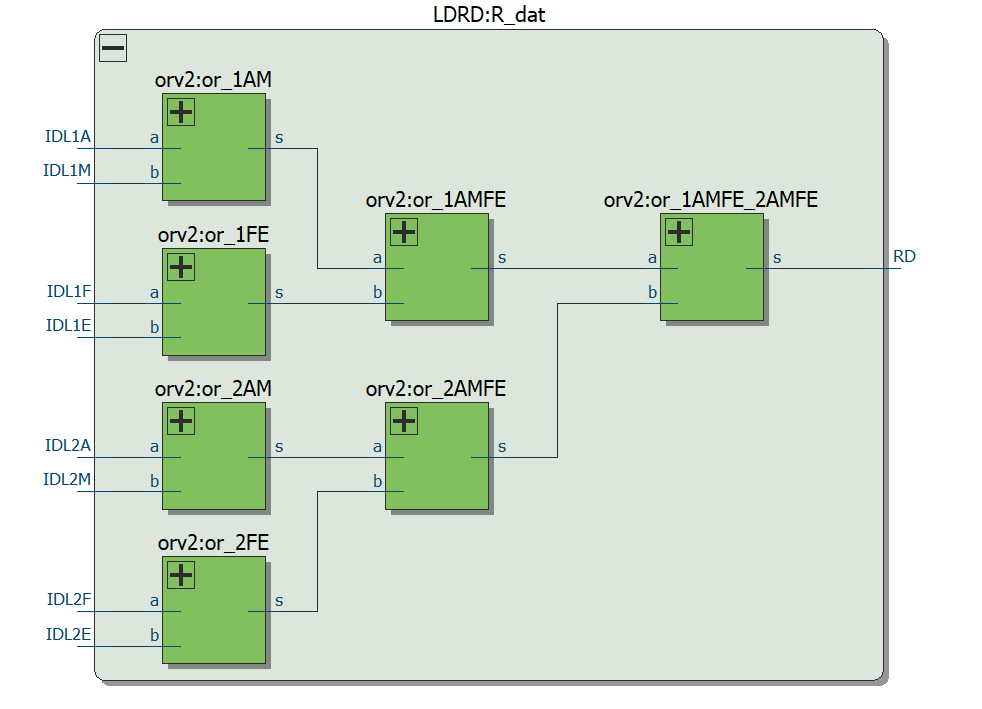
## 5.2 Archivos

### 5.2.1 LDD.vhd



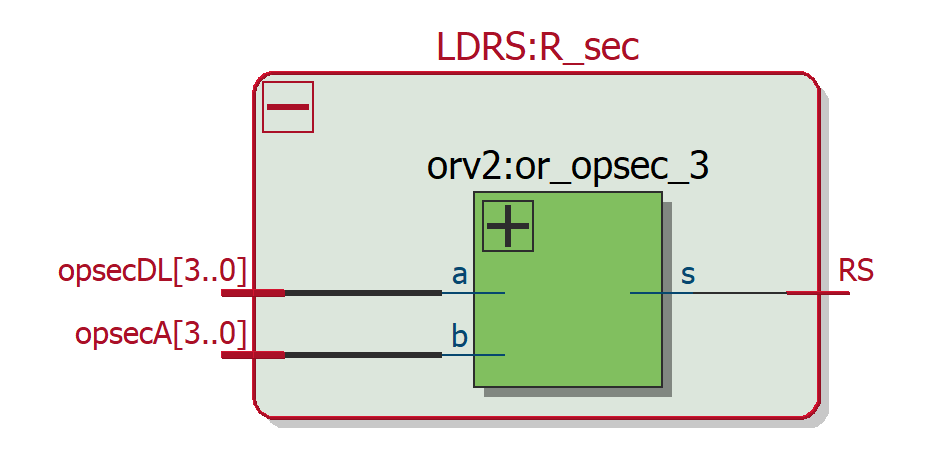
*Figura 5. RTL LDD*

### 5.2.2 LDRD.vhd



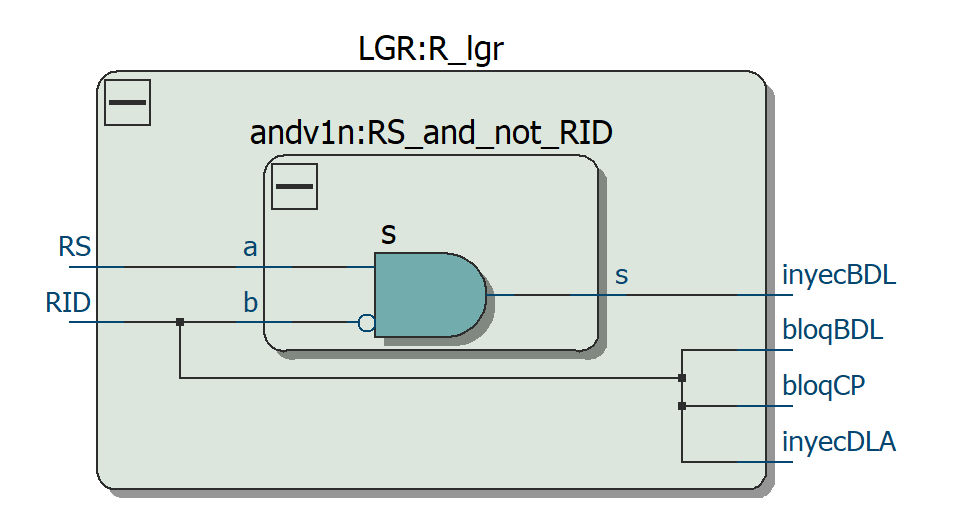
*Figura 6. RTL LDRD*

### 5.2.3 LDRS.vhd



*Figura 7. RTL LDRS*

### 5.2.4 LGRS.vhd



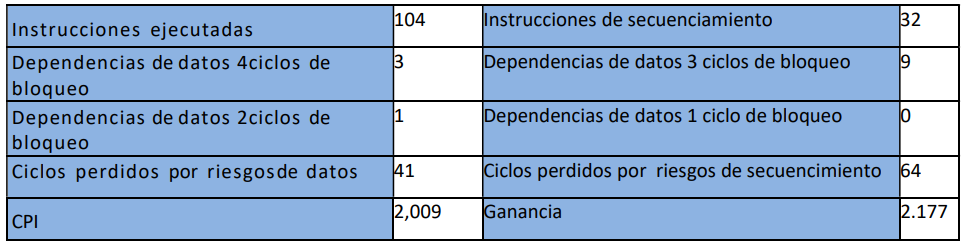
*Figura 8. RTL LGRS*

# 

# 6. Pregunta 6

Utilice el programa euclides2 para comprobar el funcionamiento del procesador segmentado (“Simulación“ en la página 341). Añada un proceso al programa de prueba ENSAMBLADO/PRUEBAS/prueba\_Rproc\_MD\_MI.vhd) para obtener las métricas indicadas en la tabla. Calcule la ganancia respecto del procesador serie.

## 6.1 Métricas



*Tabla 1. Métricas de instrucciones Archivo 1*

## 6.2 Justificación

CPI = Ciclos totales/Instruc. Ejecutadas CPI = 209/104 = 2,009 CPI

Ganancia = Tcserie\*ciclos / Tcseg\*ciclos = 35\*104/8\*(104+41+64) = 2,177

Tcserie = 35 ns

Tcseg = 8ns

# 

# 7. Pregunta 7

Entregue una copia de la ventana de tiempo correspondiente a la ejecución de la segunda y tercera iteración del programa de prueba euclides. Tenga en cuenta la forma de representar las instrucciones

## 7.1 Ventana Temporal

*Figura 9. Ventana Temporal Euclides*

## 7.2 Justificación

En el ciclo 2, se observa la dirección “10001117” en la salida del registro CP. En este ciclo la salida se está utilizando para acceder a MI.La instrucción en esta posición de almacenamiento es el valor que se observa a la salida de la memoria MI. Esta información se indica como “instruc\_BUS” en la ventana temporal. En la salida de la etapa BUS (instruc\_BUS), en el ciclo 2, se observa que se lee la instrucción “00010113” (addi x2 , x2 , x000). Esta instrucción se decodifica en el siguiente ciclo (3). En el ciclo 3, en la etapa DL, las señales muestran que el campo rs1 (IDL1) de la instrucción tiene el valor 2 e identifica un registro (val\_IDL1 = 1). El campo rs2 (IDL2) de la instrucción tiene el valor 0 e identifica un registro (val\_IDL2 = 0). El campo rd (IDE\_DL) de la instrucción tiene el valor 2 e identifica un registro (PE\_DL = 1).

En el ciclo 3, en la etapa DL, las señales muestran que el campo rs1 (IDL1) de la instrucción tiene el valor 2 e identifica un registro (val\_IDL1 = 1). El campo rs2 (IDL2) de la instrucción tiene el valor 0 e identifica un registro (val\_IDL2 = 0). El campo rd (IDE\_DL) de la instrucción tiene el valor 2 e identifica un registro (PE\_DL = 1). En el ciclo 3 se detecta un riesgo de datos (RD = 1). En consecuencia, hay que inyectar una instrucción nop desde DL hacia la etapa ALU (señal inyecBLA = 1) y bloquear las etapas DL, BUS y CP al finalizar el ciclo (bloqBDL = 1, bloqCP = 1). En el ciclo 4, en la etapa ALU se observa una instrucción nop (opALU = opSEC = opMD = 0, PE\_ALU = 0). La información en las etapas DL, BUS y CP no se modifica.

La situación descrita en el ciclo 3 perdura durante los ciclos 4 hasta 6. En el ciclo 7 no se produce riesgo de datos (RD = 0) y la interpretación de la instrucción que ocupa la etapa DL no se bloquea. En el ciclo 8, se observa, en parte, la propagación de la instrucción que ocupaba la etapa DL en el ciclo previo (PE\_ALU = 1, IDE\_ALU = 2, opALU = 10000. En los siguientes ciclos (9, 10 y 11) esta instrucción se propaga por las etapas M (PE\_M = 1, IDE\_M = 2), F (PE\_F = 1, IDE\_F = 2) y ES. La actualización del registro x2 en el ciclo 11 puede observarse expandiendo la etiqueta del banco de registro (BR).

Durante los ciclos en que las instrucciones nop inyectadas están en la etapa FMTL (6,7,8 y 9) puede observarse que no se actualizará el banco de registro en el siguiente ciclo (PE\_FMTL = 0). En el ciclo previo las instrucciones nop están en la etapa M.

También puede observarse que no se actualiza la memoria de datos (PE\_MD\_M = 0). Por otro lado, cuando las instrucciones nop están en la etapa ALU (ciclos 4, 5,6 y7) no se modifica el secuenciamiento (opSEC = 0).

En el ciclo 8 la etapa DL está ocupada por una instrucción (FE010113, “addi x2 , x2 , xFE0”). Donde también hay una dependencia de datos y se gestionará como en el ejemplo anterior.

# 

# 8. Pregunta 8

Suponga que se modifica el periodo de la señal de reloj para permitir leer en un mismo ciclo el valor con el cual se está actualizando un registro del banco de registros. Deduzca cuál debería ser el tiempo el tiempo de ciclo mínimo.

Tiempo min de ciclo: Para calcular el tiempo de ciclo mínimo, partiendo del hecho de que el tiempo de etapa es de 8 ns (según los datos del enunciado), y teniendo en cuenta que en un ciclo se completan dos etapas, podemos determinar que el tiempo de ciclo mínimo es de 10 ns. Utilizando los resultados de la pregunta 6 para el programa Euclides, podemos cuantificar si el rendimiento de esta opción sería mejor.

## 8.1 Métricas

## 

*Tabla 2. Métricas de instrucciones*

En este caso, como se puede leer y escribir en el mismo ciclo, los casos de dependencias de datos se reducen. Por lo tanto, al analizar los datos del ejercicio 6:

- Las dependencias de datos de 4 ciclos de bloqueo pasan a ser 0.

- Las dependencias de datos de 3 ciclos de bloqueo adoptan el valor de las dependencias de datos de 4 ciclos de bloqueo.

- Las dependencias de datos de 2 ciclos de bloqueo adoptan el valor de las dependencias de datos de 3 ciclos de bloqueo.

- Las dependencias de datos de 1 ciclo de bloqueo adoptan el valor de las dependencias de datos de 2 ciclos de bloqueo.

De esta manera, se reducen los ciclos perdidos debido al riesgo de datos a 28.

## 8.2 Justificación

CPI = Ciclos totales/Instrus. Ejecutadas CPI = 196/104 = 1,885

Ganancia = TcsegEj6\*ciclos / TcsegEj8\*ciclos = (8\*(104+41+64)) /(10\*(104+41+64) )= 0,853

Tcseg = 8 ns

Tcsegeje8 = 10 ns