**Práctica 6**

**Procesador: segmentación lineal con**

**cortocircuitos**

**••••••**

|  |  |
| --- | --- |
| Nombre y Apellidos | Nils Pachler de la Osa |
| Nombre y Apellidos | Genís Bosch Pou |

1

Número de grupo de laboratorio

**Preguntas 1** Diseñe los módulos incluidos en la Lógica de Cortocircuitos e InterBloqueos: LRD, LRS, LDC y LDR (Figura 6.13). Para ello utilice registros y puertas lógicas de dos o tres entradas. Justifique el diseño de forma razonada. Todos los diseños solicitados deben utilizar el menor número posible de componentes lógicos, ya sean combinacionales o secuenciales. No muestre las señales Pcero y reloj en los módulos secuenciales.

# 

Módulo LRS

Justificación

m

ADL mIDL

DL(3)

RS

opsec

•

•

* **1**

•

•

•

# 

Módulo LRD

Justificación

opMDA (4)

RD

rsA rtA

Módulo LDC

Justificación

PBRM

rsA rtA rsM rtM rsE rtE

C1L1 C1L2 C2L1 C2L2 C3L1 C3L2

# 

Módulo LDR

Justificación

RD RDAT

RS RSEC

•

•

* Práctica 6

**2**

•

•

* 1. En la siguiente tabla se identifican los ficheros que se utilizarán en el diseño de la Lógica de Cortocircuitos e InterBloqueos. Los módulos LRD, LRS, LDC y LDR deben implementarse en los ficheros LRD.dwv, LRS.dwv, LDC.dwv y LDR.dwv respectiva- mente. En el módulo LDD utilice el diseño que ha efectuado para el procesador segmentado sin cortocircuitos. El módulo VAL se suministra en el fichero VAL.dwv. En la distribución, los módulos de los ficheros LRD.dwv, LRS.dwv, LDC.dwv, LDR.dwv y LDD.dwv tienen las salidas cableadas al valor cero. Esto es, no se gestionan riesgos ni se activan los cortocircuitos.

lcib.dwv INST, rdA, rdM, rdE, PBRA, PBRM, PBRE, reloj, Pcero

C1L1, C1L2, C2L1, C2L2, C3L1, C3L2, RDAT, RDSEC

**Descripción**

Especificación estructural del circuito para el control de cortocircuitos y riesgos.

**Nombre Señales de entrada Señales de salida**

VAL.dwv INST vrs, vrt Validación de los registros fuente de una instrucción. LDD.dwv rs, rt, vrs, vrt, rdA, rdM, rdE, PBRA, PBRM, PBRE rsA, rsM,rsE, rtA, rtM, rtE Especificación estructural del módulo LDD. Utilice el

módulo diseñado para el procesador segmentado sin

cortocircuitos.

LRD.dwv rsA, rsM, rsE, rtA, rtM, rtE, opMDA RD Especificación estructural del módulo LRD. LRS.dwv mADL, mIDL, opsecDL RS Especificación estructural del módulo LRS.

LDC.dwv rsA, rsM, rsE, rtA, rtM, rtE, PBRM, reloj, Pcero C1L1, C1L2, C2L1,

C2L2, C3L1, C3L2

Especificación estructural del módulo LDC.

LDR.dwv RD, RS, reloj, Pcero RDAT, RSEC Especificación estructural del módulo LDR.

•

•

* **3**

•

•

•

Como puede observar, el diseño en VHDL es estructural. La librería de componentes que puede utilizar se denomina libRiesCortos.clf.

Describa en VHDL los diseños que ha efectuado de los módulos LRD, LRS, LDC y LDR. Entregue una copia de los mismos.

* 1. Para facilitar el cálculo del tiempo de ciclo utilizaremos un cronograma donde se repre- senta 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 siguiente tabla se muestran los acrónimos utilizados para denominar los componentes del camino de datos. Todos deben interpretarse como el retardo del componente. Alguno de los componentes están incluido dentro de un módulo que se visualiza, por ejemplo, los decodificadores del banco de registros.

|  |  |  |  |
| --- | --- | --- | --- |
| ES |  | **ME** | Registro de desacoplo de entrada de la etapa ES |
|  | FMTL | Módulo que formatea los datos leídos de la memo- ria de datos |
| mES | Multiplexor en la etapa ES |

|  |  |  |  |
| --- | --- | --- | --- |
| DL |  | **BDL** | Registro de desacoplo de entrada de la etapa DL |
| U C | decBRE | Decodificador del puerto de escritura del banco de registros. No está representado en las figuras |
| retBRE | Retardo en la actualización de un registro del banco de registros |
| decBRL | Decodificador de los puertos de lectura del banco de registros. No está representado en las figuras |
| retBRL | Retardo en la lectura de un registro del banco de registros |
| decot | Retardo del decodificador |
| mIE | Mutiplexor para seleccionar el identificador de regis- tro destino |
| FMT | Módulo formateador del literal en la UC |
| C3L1/C3L2 | Multiplexores de cortocircuito |
| U S E | FMTI | Formateador del campo literal en la USE |
| + | Sumador en la USE |
| C | Cálculo de la dirección cuando el direccionamiento es absoluto |
| mA | Multiplexor |
| L C I B | VAL | Módulo de validación |
| LDD | Módulo de detección de dependencias |
| LRS | Módulo de detección de riesgo de secuenciamiento |
| LRD | Módulo de detección de riesgo de datos |
| LDC | Módulo para e oontrol de los multiplexores de cortocircuito |
| LDR | Módulo para la gestión de resgos |
| BUS |  | **CP** | Registro CP |
|  | MI | Memoria de instrucciones |

|  |  |  |  |
| --- | --- | --- | --- |
| M |  | **AM** | Registro de desacoplo de entrada de la etapa M |
|  | CNTL | Módulo interno al módulo MDS. Control de la memoria |
| MDE | Retardo de escritura en memoria |
| MDL | Retardo de lectura en memoria |
| mAD | Multiplexor en la etapa M |

|  |  |  |  |
| --- | --- | --- | --- |
| ALU |  | **DLA** | Registro de desacoplo de entrada de la etapa ALU |
|  | C2L1/C2L2 | Multiplexores de cortocircuito |
| C1L1/C2L2 Multiplexores de cortocircuito  mL1/mL2 Multiplexores para seleccionar los operandos a y b que utiliza la etapa ALU  ALU Módulo ALU  retPBR Retardo en el modulo que modifica la señal PBR en la etapa ALU. No está representado en las figuras | | |
|  | FMTE | Módulo para formatear el dato con el cual se actua-  liza memoria |
| U S E | EVAL | Unidad de evaluación de la condición |
| DECS | Módulo de secuenciamiento condicional |
| SEL | Módulo de selección del secuenciamiento |
| mI | Multiplexor |
| mAIR | Multiplexor |

|  |  |  |  |
| --- | --- | --- | --- |
| CP |  | + | Sumador |
|  | mCP | Multiplexor cuya salida es entrada en el registro CP |

Los retardos de los componentes utilizados se detallan en el apéndice Apéndice 6.1 (Figura 6.21 y Figura 6.22). Estos retardos no son representativos de un diseño. Sólo son de utilidad para efectuar los cálculos de retardo que se solicitan.

Tenga en cuenta las mismas consideraciones que en la práctica del procesador segmentado sin cortocircuitos.

Calcule el tiempo de ciclo y la duración mínima de de los niveles de la señal de reloj. Rellene el cronograma de retardos cuando la duración del nivel alto de la señal de reloj es el mínimo. Marque el camino crítico en el esquema del camino de datos que se suministra.

|  |  |
| --- | --- |
| Tiempo de ciclo mínimo (ut) | 22 ns |
| Tiempo nivel alto mínimo (ut) | 6 ns |
| Tiempo nivel bajo mínimo (ut) | 2 ns |

•

•

* Práctica 6

**4**

•

•

•

•

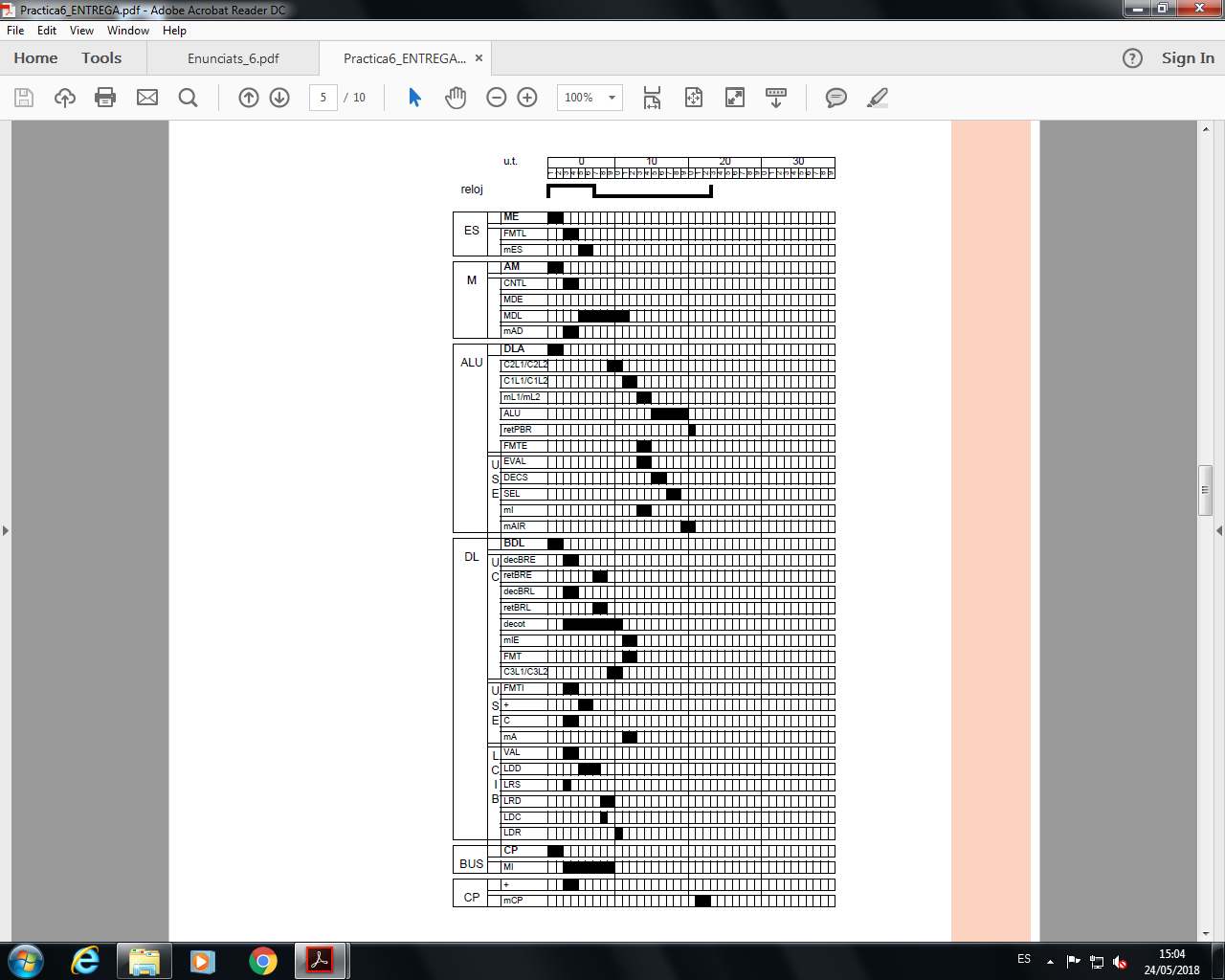
•

Procesador: segmentación lineal con cortocircuitos • **4**

•

•

•



•

•

Procesador: segmentación lineal con cortocircuitos • **5**

•

•

•

Reloj



FMTL

|  |  |  |
| --- | --- | --- |
| ES |  | **ME** |
|  | |
|  | FMTL |
|  |
| mES |

**ME**



|  |  |  |
| --- | --- | --- |
| M |  | **AM** |
|  | |
|  | CNTL |
|  |
| MDE |
|  |
| MDL |
|  |
| mAD |

**AM**

|  |  |  |
| --- | --- | --- |
| ALU |  | **DLA** |
| C2L1/C2L2 C1L1/C2L2  mL1/mL2 ALU  retPBR | |
|  | FMTE |
|  |  |
| U S E | EVAL |
|  |
| DECS |
|  |
| SEL |
|  |
| mI |
|  |
| mAIR |
|  |  |  |
| DL |  | **BDL** |
|  | |
| U C | decBRE |
|  |
| retBRE |
|  |
| decBRL |
|  |
| retBRL |
|  |
| decot |
|  |
| mIE |
|  |
| FMT |
|  |
| C3L1/C3L2 |
|  | |
| U S E | FMTI |
|  |
| + |
|  |
| C |
|  |
| mA |
|  | |
| L C I B | VAL + LDD |
|  |
| LRS |
|  |
| LRD |
|  |
| LDC |
|  |
| LDR |
|  |  |  |
| BUS |  | **CP** |
|  | |
|  | MI |

1

0

mES

MDS

dir EVMD

0

1 mAD

0 0

1 1

mL1 1 a

0

mL2

ES

M

1 ALU

0 0 b

1 1 0

FMTE

ig

ALU

EVAL DECS

me

**DLA**

**decot**

# 31 2

1

0

mIE

31

shamt

funct rd rt rs Co

FMT IL1

L1

mI 1

0

1

0

1 mAIR 0

SEL

IL2 BR

offset

1

offset

**(B)**

0

**BDL** J I R

FMTI

C

PBR L2 0

IE ED

+ 1

0

mA

**(C)**

**(A)**

DL

BUS

**CP BDL**



mCP

+

CP

MI

BHS1 BHS2

#4 +

1

0

mCP

CP

**CP**

•

•

* Práctica 6

**6**

•

•

•

•

•

* **7**

•

•

•

1. Entregue una copia de la ventana de tiempo que se observa al ejecutar una iteración completa de la función mult del programa de prueba “fact\_recurs”. Identifique en el diagrama temporal las activaciones de los cortocircuitos. Identifique también los ciclos perdidos por riesgos de datos y de secuenciamiento. Para ello, marque los ciclos en los cuales la etapa E está procesando nops inyectadas por la lógica de interbloqueos
2. Ejecute el programa de prueba “sort” y calcule la ganancia respecto del procesador segmentado sin cotocircuitos. Para calcular la ganancia tenga en cuenta el tiempo de ciclo que ha calculado en las dos prácticas. Modifique el módulo contadores (“Conta- dores de eventos“ en la página 285).

|  |  |
| --- | --- |
|  | Procesador CON cortocircuitos |
| Ciclos perdidos por riesgos de datos | 162 |
| Ciclos perdidos por riesgos de secuenciamiento | 350 |
| Ciclos de ejecución | 2139 |
| Tiempo de ciclo (ns) | 22 |
| Ganancia | 1.31 |

|  |
| --- |
| Procesador SIN cortocircuitos |
| 1283 |
| 988 |
| 3855 |
| 16 |

1. Un ingeniero analiza la actuación al interpretarse una instrucción de secuenciamiento y observa que si se sigue en secuencia se está estableciendo como valor en el registro CP la dirección de la instrucción que se descarta.

 ciclos

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CP | BUS | D/L | ALU | M | ES |  |  |  |  |
|  | CP | BUS | D/L | ALU | M | ES |  |  |  |
|  |  | CP | BUS | nopS | nopS | nopS | nopS |  |  |
|  |  |  | CP | BUS | D/L | ALU | M | ES |  |
|  |  |  |  | CP | BUS | D/L | ALU | M | ES |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | dir. | instrucción |  | 1 | 2 | 3 | 4 5 | 6 | 7 | 8 | 9 | 10 |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | 8 | blez r3, 1$ |  |  |  |  |  |  |  |  |  |  |
|  | 12 | add **r26**, r0, r4 |  |  |  |  |  |  |  |  |  |  |
|  | 16 | sub **r24**, r2, r7 |  |  |  |  |  |  |  |  |  |  |
|  | 16 | sub **r24**, r2, r7 |  |  |  |  |  |  |  |  |  |  |
|  | 20 | and **r9**, r9, r7 |  |  |  |  |  |  |  |  |  |  |
|  |  | recursos |  |  | + | + | USE |  |  |  |  |  |

En estas condiciones decide que sólo se descarte la instrucción que sigue a la instruc- ción que está en el retardo de la instrucción de secuenciamiento cuando se cumple la condición evaluada. En otras palabras, decide efectuar una predicción del resultado de evaluar la condición.

Para simplificar el diseño, se decide predecir seguir en secuencia de forma fija en todas las instrucciones de secuenciamiento. Por tanto, en instrucciones de secuencia- miento condicional, si no se cumple la condición la instrucción de secuenciamiento no

Procesador: segmentación lineal con cortocircuitos

actualiza el registro CP y no se descarta ninguna instrucción. En cualquier otro caso, la instrucción de secuenciamiento actualiza el registro CP y se descarta la instrucción que ha empezado a interpretarse en el ciclo previo.

En el diseño que se solicita, solo se requiere modificar el módulo SEL de la Unidad de Secuenciamiento Explícito (Figura 6.5) y algunos módulos de la Lógica de Cortocir- cuitos e InterBloqueos (Figura 6.13).

En cuanto al módulo SEL, observe que, cuando se detecta un error de predicción, el multiplexor mCP debe seleccionar la dirección destino de la etapa ALU. Por tanto, la señal de salida mCP debe activarse en caso de error de predicción.

* + Utilizando puertas lógicas, diseñe el módulo SEL.

Módulo SEL

Justificación

mAA mIA opsecA(3)

mR mAA mIA mR

mCP

mAIR

Observe que este mecanismo de predicción requiere también modificar la gestión de riesgos de secuenciamiento: se descarta la instrucción buscada cuando se detecta un error de predicción. En consecuencia, los módulos de LCIB a modificar son LRS y LDR.

* + Suponga que la señal de control mCP (error de predicción) es una señal de entrada de LCIB. Diseñe los módulos LRS y LDR. Tenga en cuenta que la señal de salida RSEC debe activarse en caso de error de predicción.

Módulo LRS

Justificación

mADL RS

mIDL

opsecDL(3)

•

•

* Práctica 6

**8**

•

•

•

# 

Módulo LDR

Justificación

RD RDAT

RS

mCP

RSEC

•

•

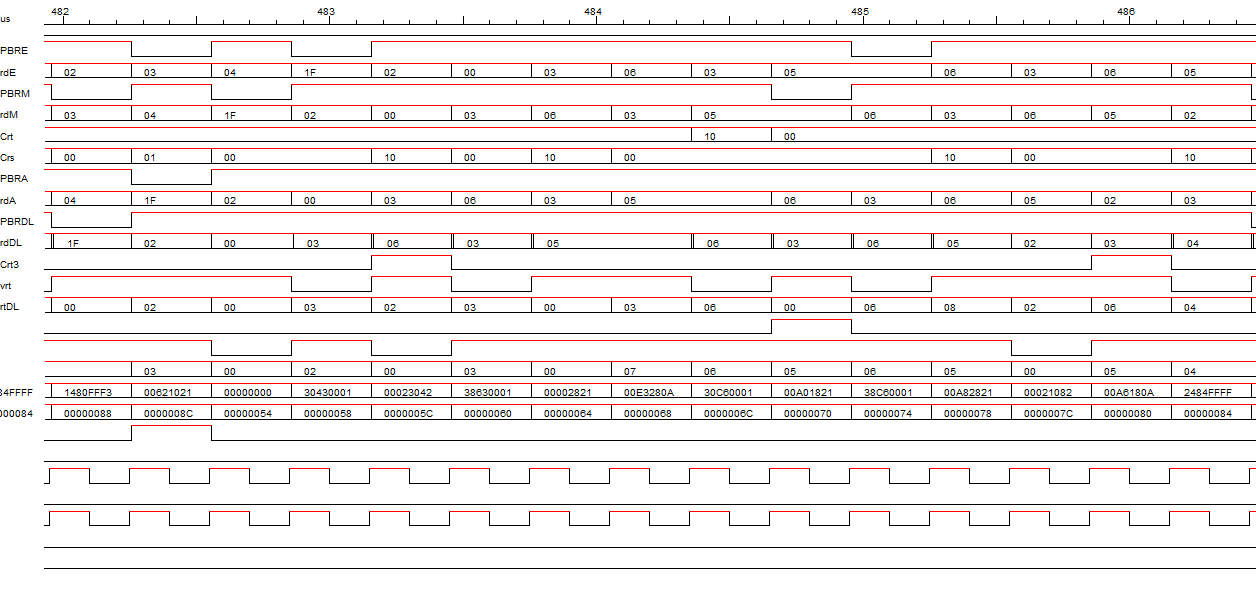
9

•

•

•

Procesador: segmentación lineal con cortocircuitos



NOPS

**LDC**

library IEEE;

use IEEE.std\_logic\_1164.all;

use work.retardos.all;

library libs;

use libs.libRiesCortos.all;

use libs.libReDe.all;

entity LDC is

generic(tam: natural := 5);

port(reloj, Pcero: in std\_logic;

rsA, rsM,rsE: in std\_logic;

rtA, rtM, rtE: in std\_logic;

PBRM: in std\_logic;

C1L1, C1L2, C2L1, C2L2, C3L1, C3L2: out std\_logic);

end LDC;

architecture estructural of LDC is

component rd1

port (reloj, pes: in std\_logic;

e: in std\_logic;

s: out std\_logic);

end component rd1;

component andv

port(a, b: in std\_logic;

s: out std\_logic);

end component andv;

signal PBRE : std\_logic;

signal rsAr, rsMr: std\_logic;

signal rtAr, rtMr: std\_logic;

begin

r2: rd1 port map (reloj => reloj, pes => Pcero, e => PBRM, s => PBRE);

r3: rd1 port map (reloj => reloj, pes => Pcero, e => rsA, s => rsAr);

r4: rd1 port map (reloj => reloj, pes => Pcero, e => rsM, s => rsMr);

r5: rd1 port map (reloj => reloj, pes => Pcero, e => rtA, s => rtAr);

r6: rd1 port map (reloj => reloj, pes => Pcero, e => rtM, s => rtMr);

a1: andv port map (a => rsE, b => PBRE, s => C3L1);

a2: andv port map (a => rtE, b => PBRE, s => C3L2);

a5: andv port map (a => rsAr, b => PBRM, s => C1L1);

a6: andv port map (a => rtAr, b => PBRM, s => C1L2);

a3: andv port map (a => rsMr, b => PBRE, s => C2L1);

a4: andv port map (a => rtMr, b => PBRE, s => C2L2);

end estructural;

**LDR**

library IEEE;

use IEEE.std\_logic\_1164.all;

use work.retardos.all;

library libs;

use libs.libRiesCortos.all;

use libs.libReDe.all;

entity LDR is

port(reloj, Pcero: in std\_logic;

RD, RS: in std\_logic;

RDAT, RSEC: out std\_logic);

end LDR;

architecture estructural of LDR is

component rd1

port (reloj, pes: in std\_logic;

e: in std\_logic;

s: out std\_logic);

end component rd1;

component andv1n

port(a, b: in std\_logic;

s: out std\_logic);

end component andv1n;

signal t : std\_logic;

begin

RDAT <= RD;

a: andv1n port map (a => RS, b => RD, s => t);

r: rd1 port map (reloj => reloj, pes => Pcero, e => t, s => RSEC);

end estructural;

**LRD**

library IEEE;

use IEEE.std\_logic\_1164.all;

use work.retardos.all;

library libs;

use libs.libRiesCortos.all;

entity LRD is

generic(tam: natural := 5);

port(rsA, rtA: in std\_logic;

rsM, rtM: in std\_logic;

rsE, rtE: in std\_logic;

opMDA : in std\_logic\_vector(tam-1 downto 0);

RD: out std\_logic);

end LRD;

architecture estructural of LRD is

component orv

port(a, b: in std\_logic;

s: out std\_logic);

end component orv;

component andv

port(a, b: in std\_logic;

s: out std\_logic);

end component andv;

signal c: std\_logic;

begin

o: orv port map (a => rsA, b => rtA, s => c);

a: andv port map (a => opMDA(4), b => c, s => RD);

end estructural;

**LRS**

library IEEE;

use IEEE.std\_logic\_1164.all;

use work.retardos.all;

library libs;

use libs.libRiesCortos.all;

entity LRS is

generic(tam: natural := 4);

port(mADL, mIDL : in std\_logic;

opsec : in std\_logic\_vector(tam-1 downto 0);

RS: out std\_logic);

end LRS;

architecture estructural of LRS is

component or3v

port(a, b, c: in std\_logic;

s: out std\_logic);

end component or3v;

begin

o: or3v port map (a => mADL, b => mIDL, c => opsec(3), s => RS);

end estructural;