Contenido

Práctica 5
Procesador: segmentación lineal, camino de datos y control

En esta sesión se trata de consolidar los conocimientos adquiridos sobre segmentación del camino de datos de un procesador y el control del mismo.

Para ello partiremos del procesador serie de la práctica anterior. La segmentación utilizada es lineal y el camino de datos unificado.

El objetivo de la práctica es identificar las etapas y los componentes utilizados en cada una de ellas. Así mismo, se analiza cuál de las etapas determina el tiempo de ciclo y la duración de la señal de reloj en cada uno de los dos niveles lógicos. Para ello es necesario determinar el tiempo requerido por cada etapa, el cual es función del retardo de los componentes utilizados por cada tipo de instrucción y el conexionado de estos componentes.

Descripción de la segmentación del procesador

El procesador segmentado tarda 7 ciclos en interpretar una instrucción y las etapas utilizadas se muestran en la Figura 5.1.

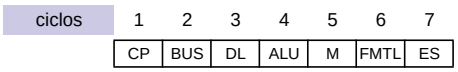


Figura 5.1 *Etapas del procesador segmentado lineal.*

En la tabla de la Figura 5.2 se describe la funcionalidad de las etapas. En un mismo ciclo no puede leerse el valor con el cual se está actualizando un registro del banco de registros[1].

1. En función del periodo de la señal de reloj podría ser factible. Ahora bien, el valor mínimo no lo permite.

ETAPA	FUNCIONALIDAD
CP	Determinación de la dirección de la instrucción.
BUS	Búsqueda de la instrucción.
DL	Decodificación y lectura de operandos en registros del banco de registros.
ALU	Operaciones aritméticas y lógicas. Cálculo de la dirección efectiva en instrucciones de acceso a memoria. Adicionalmente, cálculo de la dirección destino de salto, evaluación de la condición en instrucciones de secuenciamiento condicional y actualización de la entrada del registro CP. Además formateo de la escritura en memoria y activación de las señales de escritura en memoria.
M	Acceso a la memoria de datos.
FMTL	Formateo de la lectura de memoria.
ES	Escritura en el banco de registros.

Figura 5.2 *Descripción funcional de las etapas.*

En el diseño que se presenta, se pueden estar realizando acciones, que no actualizan el estado de procesador, aunque posteriormente no se utilice la salida del módulo que efectúa la acción. Por ejemplo, en cada ciclo se lee del banco de registros utilizando los campos rs1 y rs2 de una instrucción, independientemente de que la información leída se utilice o no posteriormente.

Segmentación de la unidad de cálculo

En la Figura 4.13 se ha mostrado el esquema de la unidad de cálculo (UC) y en la Figura 4.21 la comunicación de la UC para acceder a la memoria de datos. El detalle de la memoria de datos y la lógica asociada para su acceso se muestran en la Figura 4.42 y en la Figura 4.43.

Seguidamente nos centramos en la lógica que se utiliza en cada una de las etapas del procesador segmentado a partir de la etapa DL.

En los esquemas del camino de datos, los registros de desacoplo se identifican con una línea gruesa individual para cada conjunto lógico de información. El conjunto de registros de desacoplo correspondientes a la entrada o salida de una etapa se identifica por una línea vertical a trazos que los atraviesa. En la parte inferior de las figuras se muestra la etiqueta genérica utilizada para cada uno de los conjuntos de registros de desacoplo (por ejemplo BDL).

En los registros de desacoplo no se muestra la entrada correspondiente a la señal de reloj[2]. Tampoco se muestra la entrada de la señal de reloj en el banco de registros y en la memoria[3].

2. Tampoco se muestran las señales de bloqueo o inyección que se utilizan en algunos de ellos.
3. Estas posiciones de almacenamiento se actualizan en el flanco ascendente de la señal de reloj.

Para identificar una señal encaminada por las etapas utilizaremos el acrónimo de la señal, utilizado en el autómata de control, y como sufijo el nombre de la etapa.

Etapa DL. Registro de desacoplo de entrada BDL

En la etapa DL se decodifica la instrucción y se leen los registros cuyo identificador se extrae de los campos rs1 y rs2 de la instrucción, independientemente de que el valor leído sea utilizado posteriormente (Figura 5.3).

En un mismo ciclo no es posible escribir y leer un registro del banco de registros. Ello es debido a que un registro del banco de registros se actualiza en el flanco ascendente de la señal de reloj y el tiempo remanente, del periodo de la señal de reloj, después de la actualización no es suficiente para efectuar la lectura. Por tanto, el valor leído de un registro como muy tarde ha sido actualizado en el ciclo anterior. La lectura de un registro es asíncrona.

Por otro lado, el identificador de registro destino (rd), de la instrucción que ocupa la etapa DL, se utiliza en la etapa ES. Por tanto, hay que encaminarlo desde la etapa DL hacia la etapa ES[4].

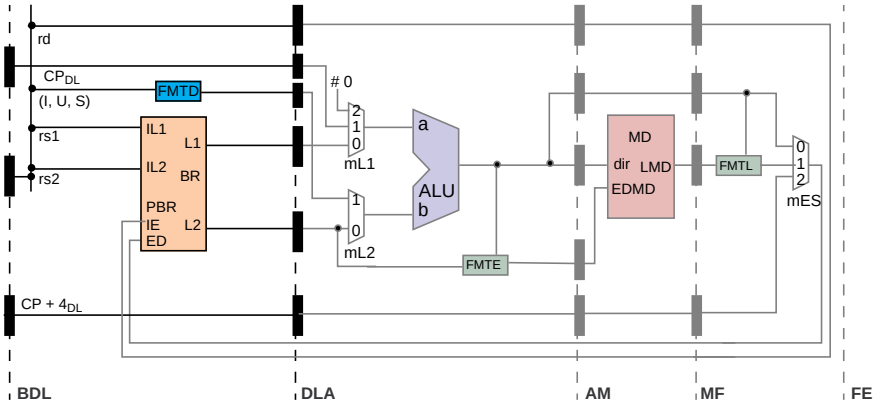


Figura 5.3 *Etapa DL.*

En el camino de datos de la Figura 5.3 no se muestran registros de desacoplo en la etapa ES ya que el banco de registros se actualiza en el flanco ascendente de la señal de reloj.

4. En una escritura, la decodificación del identificador de registro se efectúa antes del flanco ascendente de la señal de reloj (etapa FMTL).

Etapa ALU. Registro de desacoplo de entrada DLA

En la etapa ALU se seleccionan los operandos para efectuar el cálculo (multiplexores mL1 y mL2). Posteriormente se efectúa el cálculo en la ALU (Figura 5.4).

En esta etapa también se formatea el dato con el que se actualiza memoria, si es el caso (módulo FMTE). Notemos que esta acción se efectúa en paralelo con la operación en la ALU[5]. Además, este módulo gestiona las señales de permiso de escritura en los bancos de la memoria de datos.

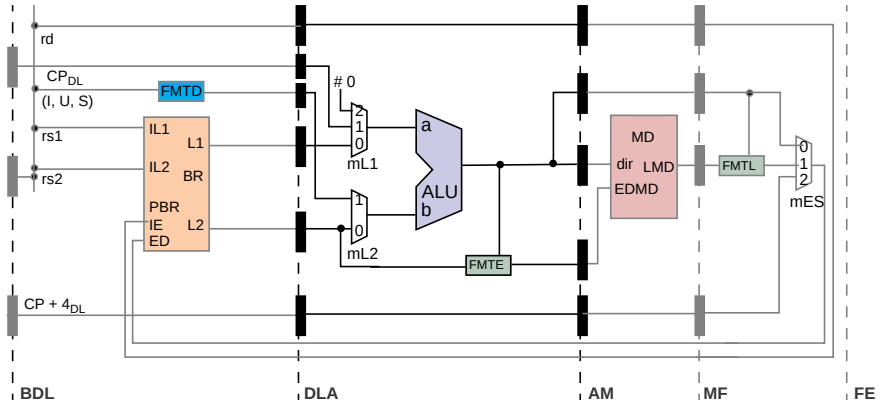


Figura 5.4 *Etapa ALU.*

Adicionalmente se utiliza un módulo, denominado acceso_MD, el cual no se muestra en la Figura 5.4, que comprueba el rango de direcciones de acceso permitido y si la dirección está alineada al tamaño del dato accedido. Este módulo se muestra en la Figura 5.12.

Etapa M: memoria de datos. Registro de desacoplo de entrada AM

La memoria se actualiza en el flanco ascendente de la señal de reloj[6] (Figura 5.5). En consecuencia, las señales de control de la misma deben haberse generado, como muy tarde, en la etapa previa (módulo FMTE),

Por otro lado, los registros que se muestran de forma tramada, en el conjunto AM, no existen de forma explícita en el diseño, ya que la MD se actualiza en el flanco ascendente de la señal de reloj. En la figura se muestran para facilitar la identificación de las etapas.

5. En el procesador serie no se mostraba el módulo FMTE de forma explícita en los esquemas del camino de datos (Figura 4.47). Ahora bien, al detallar la memoria de datos se describía su funcionalidad (Figura 4.43).
6. La decodificación de la dirección se efectúa antes del flanco ascendente de la señal de reloj (etapa ALU).

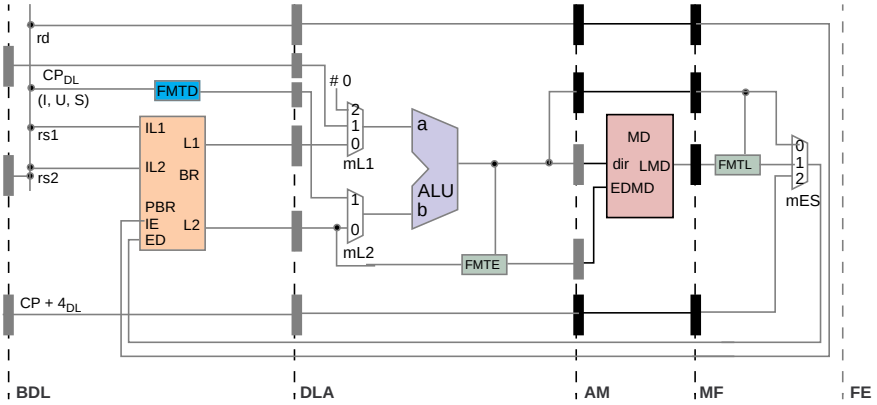


Figura 5.5 *Etapa M.*

Etapa FMTL. Registro de desacoplo de entrada MF

En una instrucción de lectura de memoria, antes de actualizar el banco de registro, se formatea el dato en la etapa FMTL (módulo FMTL)[7]. En el procesador segmentado el módulo FMTL se ubica en la etapa FMTL con el objetivo de reducir el retardo en la etapa M (Figura 5.6).

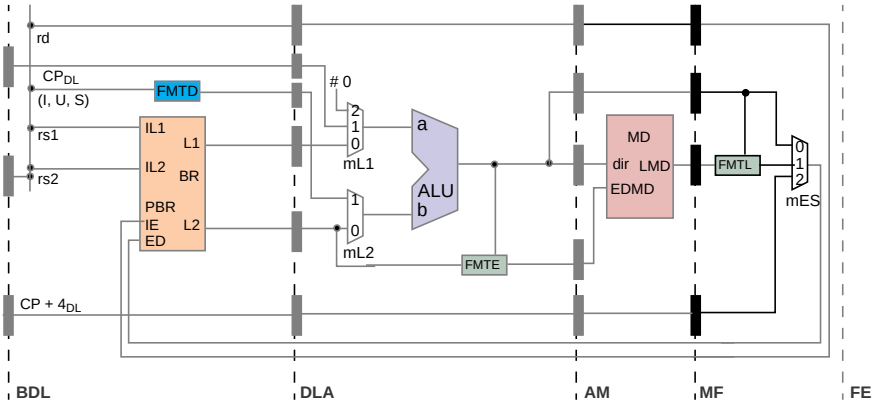


Figura 5.6 *Etapa FMTL.*

7. En el procesador serie el módulo FMTL no se mostraba de forma explícita en los esquemas del camino de datos (Figura 4.47). Ahora bien, al detallar la memoria de datos se describía su funcionalidad (Figura 4.43).

El multiplexor mES, ubicado en esta etapa, se utiliza para seleccionar entre: a) el valor formateado, leído de memoria, b) el valor calculado, dos ciclos antes, en la ALU, ubicada en la etapa ALU, y c) la dirección de la instrucción que se está interpretando en el ciclo 6 más cuatro, contando desde el inicio de interpretación[8]. El valor seleccionado se propaga a la etapa ES.

Por tor lado, en esta etapa se decodifica el identificador del registro de escritura (rd) en el banco de registros (no se muestra el detalle).

Etapa ES. Registro de desacoplo de entrada FE

En esta etapa no se identifican de forma explícita registros de desacoplo de entrada, ya que el banco de registros se actualiza en el flanco ascendente de la señal de reloj[9]. Por otro lado, la señal rd, propagada hasta la etapa FMTL, es necesaria antes del flanco de reloj, ya que hay que tener en cuenta el retardo del decodificador del identificador del registro.

El banco de registros se actualiza en el flanco ascendente de la señal de reloj y el retardo de actualización no permite efectuar una lectura del registro en el mismo ciclo. El valor que se utiliza para la actualización, si es el caso, es la salida del multiplexor mES (Figura 5.7). El identificador de registro destino (rd, IE) ha sido propagado desde la etapa DL. En la figura no se muestra la señal de permiso de escritura, la cual también ha sido propagada desde la etapa DL.

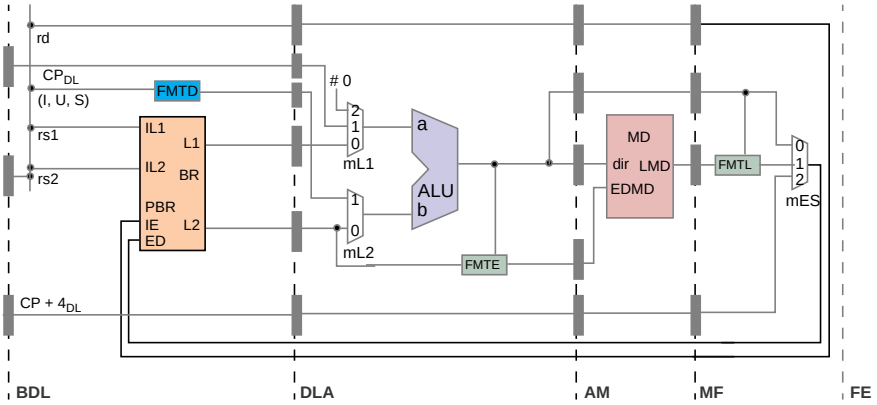


Figura 5.7 *Etapa ES.*

8. La entrada del multiplexor mES etiquetada como dos se detalla al describir el secuenciamiento.
9. En el caso de la MD, que también se actualiza en el flanco ascendente de la señal de reloj, se han utilizados registros tramados en el esquema del circuito (Figura 5.5). En este caso, como es la última etapa, no creemos necesario el añadir al esquema registros tramados para identificar la etapa.

Memoria de instrucciones

El acceso a la memoria de instrucciones requiere un ciclo de reloj y la etapa está ubicada antes de la etapa DL. El acceso está sincronizado con el flanco ascendente de la señal de reloj[10].

La funcionalidad de la memoria de instrucciones, entradas y salidas son las mismas que en el procesador serie.

Segmentación de la unidad de secuenciamiento

El registro de salida de la unidad de secuenciamiento es el registro CP y este registro es el registro de desacoplo de entrada de la etapa BUS.

Para el secuenciamiento implícito se utiliza un sumador cuyas entradas son la salida del registro CP y el valor cuatro, que es el tamaño de una instrucción en bytes (Figura 5.8).

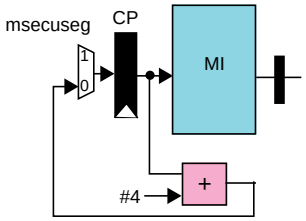


Figura 5.8 *Secuenciamiento implícito en el procesador segmentado.*

En un procesador segmentado la identificación del tipo de instrucción se puede efectuar, como muy pronto, en la etapa DL.

El valor del contenido de los registros está disponible en la etapa DL, ya que se accede al banco de registros. Sin embargo, el retardo necesario para evaluar la condición determina que el secuenciamiento se establezca en la etapa ALU. Por otro lado, en algún tipo de instrucción de secuenciamiento es necesario el contenido de un registro del banco de registros para calcular la dirección destino.

El valor de la dirección de la instrucción más cuatro se ha calculado mientras la instrucción estaba en la etapa BUS (secuenciamiento implícito, salida del sumador). Por tanto, es suficiente propagar el valor a la etapa DL y posteriormente a la etapa ALU. Este valor se utiliza en el secuenciamiento, cuando no se cumple la condición en una instrucción de secuenciamiento condicional y en una instrucción que almacena la dirección de retorno.

10. En este caso no se trama el registro CP como en el caso de la memoria de datos MD (Figura 5.5), ya que la salida del registro también es entrada en otros elementos y no se replica, en el esquema de la figura, como en el caso de la MD. En el procesador serie del capítulo previo la memoria MI era asíncrona.

En resumen, en la etapa ALU se dispone de toda la información utilizada en la lógica USE (Figura 5.9). En la parte derecha de la Figura 5.9 se muestra el multiplexor utilizado antes del registro CP para seleccionar la dirección de la siguiente instrucción que se interpreta. Las distintas entradas del multiplexor se identifican en función de los modos de direccionamiento en una instrucción de secuenciamiento. Una de las entradas se corresponde el secuenciamiento implícito. La implementación de este multiplexor, que podemos considerar conceptual, es función del diseño concreto. En la misma figura se muestra una implementación donde se observa el multiplexor msecuseg utilizado en la parte izquierda de la misma figura.

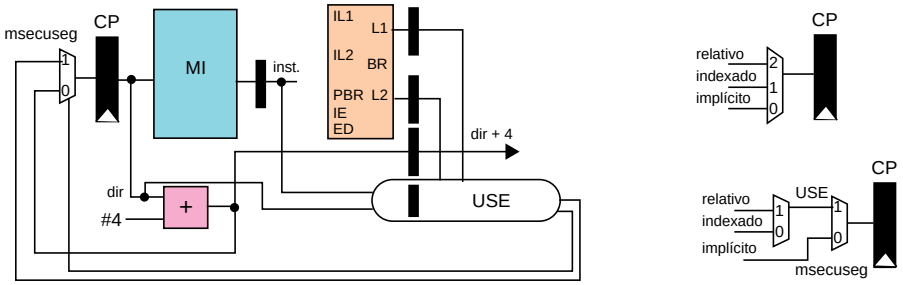


Figura 5.9 *Esquema lógico del secuenciamiento explícito. El acrónimo dir indica la dirección de la instrucción que ocupa la etapa BUS. En la parte derecha se muestra el multiplexor conceptual de la etapa CP y un diseño donde se explicita el multiplexor msecuseg.*

Camino de datos del secuenciamiento

En la Figura 5.10 se muestra la lógica de secuenciamiento. Respecto al diseño serie del procesador, se utiliza un multiplexor adicional para establecer el secuenciamiento implícito (msecuseg). Este multiplexor, de la etapa CP (o etapa ALU), selecciona entre secuenciamiento implícito y secuenciamiento explícito. El control de este multiplexor está determinado por la señal del decodificador que indica que se está interpretando una instrucción de secuenciamiento (es el bit más significativo de $opSEC_A$).

El módulo FMTS se ubica en la etapa DL y el resto de elementos de la USE se ubican en la etapa ALU. La dirección de la instrucción mas cuatro se propaga por las etapas y es entrada del multiplexor mES.

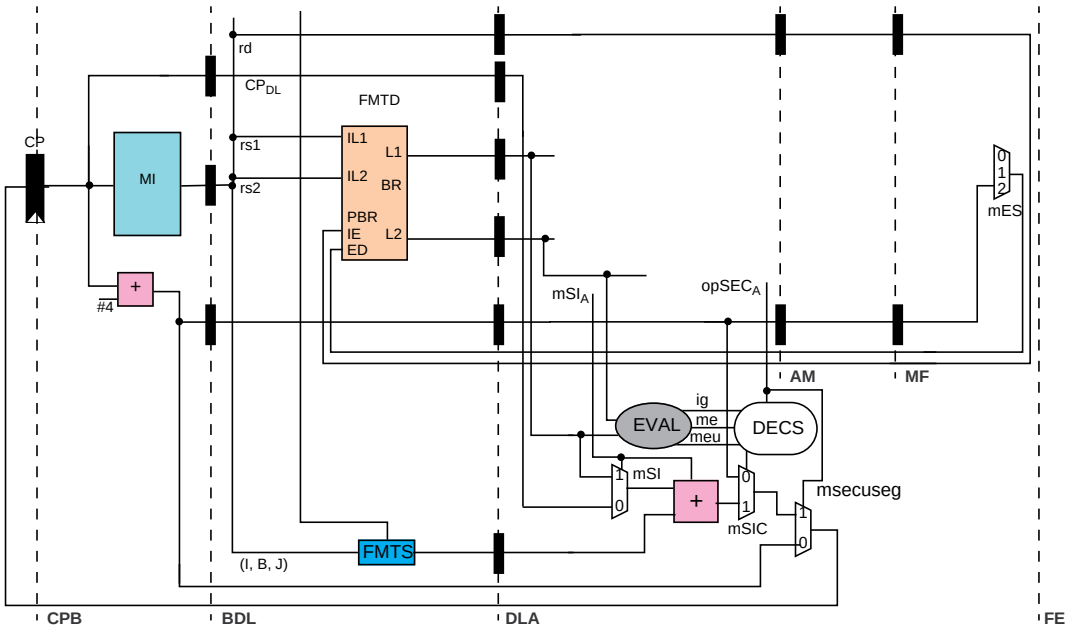


Figura 5.10 *Unidad de secuenciamiento. Camino de datos.*

Autómata de control

El autómata de control está particionado en dos módulos como en el procesador serie (Figura 5.11), denominandose uno de ellos autómata de control. En el módulo excepciones se ubica la gestión de las condiciones de excepción. Para ello se recolectan las condiciones de excepción que se pueden generar en distintas partes del procesador (etapas CP y ALU). Además, cuando la instrucción está en la etapa DL, este módulo genera la señal de código de operación incorrecto (CoErr).

La decodificación de una instrucción y la generación de las señales de encaminamiento y control de las unidades funcionales, cuando es necesario, se efectúa en el automáta de control (AC). Estas señales se propagan por las etapas mientras sea necesaria su utilización en una etapa posterior.

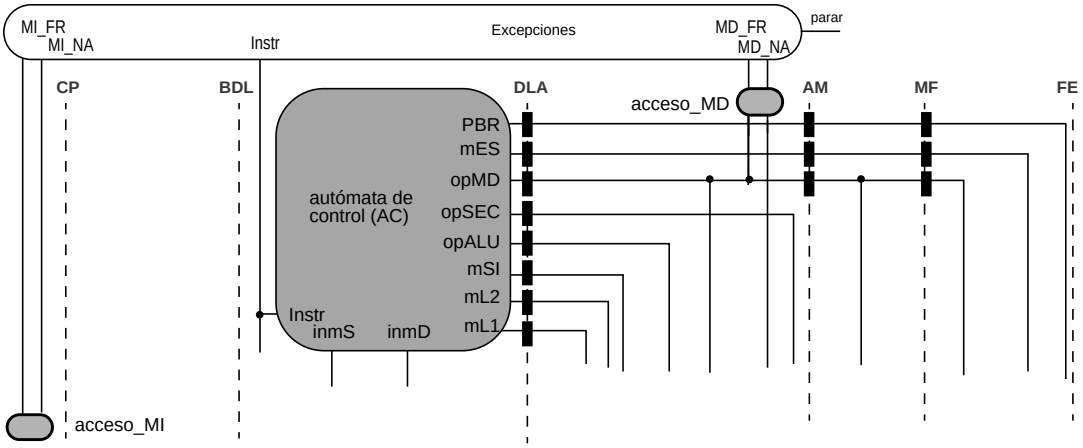


Figura 5.11 *Autómata de control y gestión de excepciones.*

Etapa CP

El multiplexor msecuseg, ubicado en la etapa CP, se controla mediante una señal que genera la lógica de decodificación (instrucción de secuenciamiento, opSEC).

En esta etapa se generan, utilizando el módulo acceso_MI, las señales de excepción correspondientes a dirección no alineada (MI_NA) y dirección fuera de rango (MI_FR) al acceder a la memoria de instrucciones (Figura 5.11).

Etapa BUS

En la etapa BUS se accede a la memoria de instrucciones.

Etapa DL

En esta etapa se generan todas las señales para controlar el encaminamiento y las operaciones en los módulos que pueden realizar varias acciones. La codificación de las señales es idéntica al diseño serie del procesador.

De todas las señales generadas por el autómata de control, en esta etapa solo se utilizan las señales de control de los formateadores (inmD e inmS).

El resto de señales generadas en el decodificador se propagan a la siguiente etapa.

Etapa ALU

En esta etapa se utilizan algunas de las señales de control propagadas desde la etapa DL. En la UC se utilizan las señales mL1, mL2 y las correspondientes a la ALU y al módulo que formatea el dato para actualizar memoria (FMTE y las señales de permiso de escritura en los bancos de la MD).

El módulo FMTE se controla mediante la señal opMD y los bits menos significativos de la dirección calculada en la ALU (Figura 4.43). En concreto, se utilizan los bits uno y cero de opMD que codifican el tamaño del dato que debe almacenarse en memoria.

Las señales de control propagadas desde la etapa DL, que no son utilizadas en la etapa ALU, se propagan a la etapa M, junto con la señal opMD utilizada en esta etapa.

El multiplexor mSI, de la USE, se controla con la misma señal que en el diseño serie del procesador, una vez propagada desde DL. El multiplexor msecuseg se controla con la señal opSEC, la cual ha sido propagada desde DL. El módulo DECS se controla también con la señal opSEC. La salida de este elemento controla el multiplexor mSIC.

En la etapa ALU, módulo acceso_MD, se generan las señales de excepción correspondientes a dirección no alineada (MD_NA) y dirección fuera de rango (MD_FR) al acceder a la memoria de datos (Figura 5.11).

Etapa M

En esta etapa se utiliza la señal que controla el acceso a memoria (opMD). El resto de las señales, propagadas desde DL, junto con la señal opMD, se propagan a la etapa FMTL.

Etapa FMTL

En la etapa FMTL se utilizan las señales opMD y la dirección calculada en la ALU, dos ciclos antes, para formatear la salida de memoria (FMTL). En concreto, se utilizan los bits dos, uno y cero de la señal opMD y los bits uno y cero de la dirección. Los bits uno y cero de la dirección codifican la posición del dato accedido en los cuatro bytes leídos de memoria. Los bits uno y cero de la señal opMD codifican el tamaño del dato leído de memoria. El bit dos codifica la interpretación de los bits leídos de memoria como un número entero o un número natural (Figura 4.101).

Para controlar el multiplexor mES se utiliza la señal mES del procesador serie, propagada desde la etapa DL.

Etapa ES

En esta etapa se actualiza el banco de registro en el flanco ascendente de la señal de reloj. Para ello, se utiliza la señal de control propagada exprofeso, desde la etapa DL, para actualizar el banco de registros (PBR_E).

Camino de datos completo

En la Figura 5.12 se muestra el esquema completo del camino de datos del procesador, en el cual se han especificado todas las señales menos la señal de reloj y la señal de inicialización. Tampoco se especifican las señales de control de algunos registros de desacoplo.

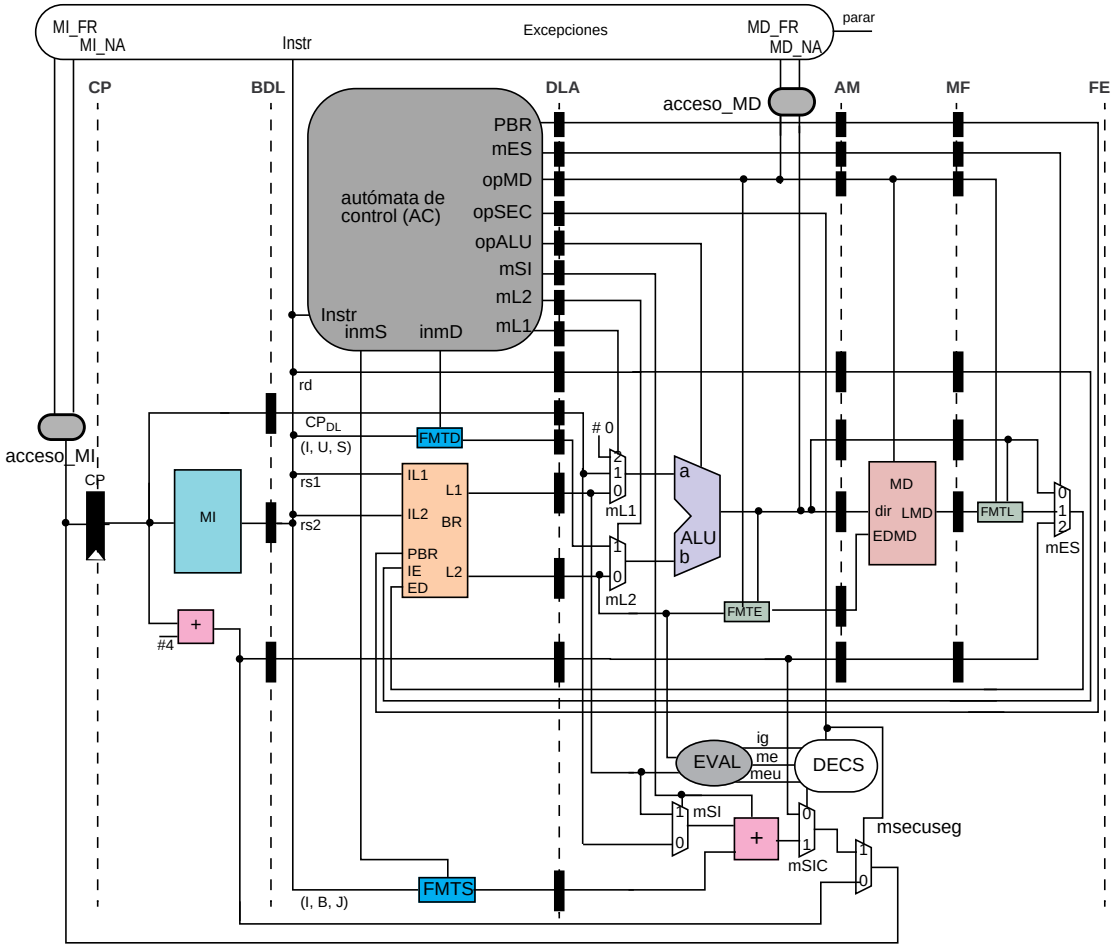


Figura 5.12 *Camino de datos del procesador segmentado.*

Adecuación de la semántica del procesador segmentado al lenguaje máquina

En esta sección se analiza si la semántica del procesador segmentado se adecua a la semántica del lenguaje máquina. Esto es, si la segmentación del procesador mantiene el orden de lecturas y escrituras, especificado por el programador, a posiciones de almacenamiento. En primer lugar se analiza la unidad de cálculo y posteriormente la unidad de secuenciamiento. Por último se diseña la lógica de interbloqueos para respetar el orden de lecturas y escrituras, especificado por el programador, a posiciones de almacenamiento.

Unidad de cálculo

En la Figura 5.13 se muestra un esquema del camino de datos donde se observan las etapas DL hasta ES. La disposición de los elementos del camino de datos en la figura es en vertical en lugar de horizontal. Las etapas se han dibujado en una columna. Los registros de desacoplo en la salida de una etapa (parte derecha de cada etapa) son los registros de desacoplo en la entrada de la etapa representada encima de ella (parte izquierda de cada etapa). En la Figura 5.13, cada conjunto de registros de desacoplo se ha etiquetado con el mismo acrónimo que en la Figura 5.12.

Los registros de entrada en la memoria de datos (MD) y en el banco de registros (BR) se muestran con un tramado, Esta característica es para indicar que los dos conjuntos de posiciones de almacenamiento se actualizan en el flanco ascendente de la señal de reloj[11].

Encima de las etapas se representa un ciclo de la señal de reloj, en el cual no se ha representado a escala el intervalo en cada nivel lógico (Figura 5.13).

Los registros de desacoplo almacenan en el flanco ascendente de la señal de reloj los valores que hay en sus entradas. Estos valores, después de un retardo, son entradas estables de los elementos del camino de datos utilizados en las etapas. Después del retardo de estos elementos, se dispone de nuevos valores estables en la entrada de los registros de desacoplo. En el siguiente flanco ascendente de la señal de reloj los nuevos valores se almacenan en los registros de desacoplo.

El bus ED transporta el valor con el cual se actualiza un registro del banco de registros en el flanco ascendente de la señal de reloj.

11. Tenga en cuenta que la decodificación de la dirección del identificador de registro debe efectuarse antes del flanco ascendente de la señal de reloj.

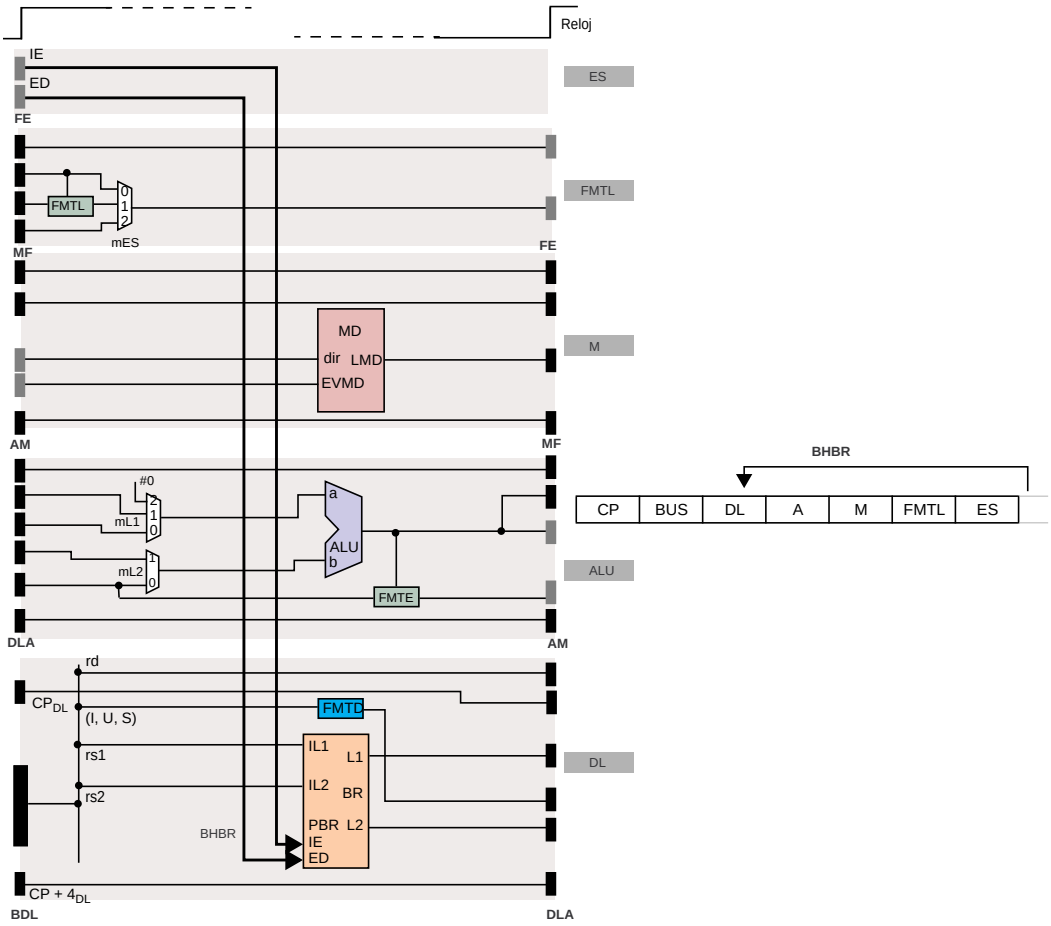


Figura 5.13 *Bucle hardware para actualizar el banco de registros.*

La actualización de un registro del banco de registros utiliza un bucle hardware (BHBR) desde la etapa ES hasta la etapa DL y la latencia del bucle son 5 ciclos (Figura 5.13). Tengamos en cuenta que el valor que se escribe no se puede leerse hasta después de finalizar el ciclo de reloj.

En estas condiciones, una instrucción consumidora que inicie su ejecución antes de 5 ciclos, después de la productora, no leerá el valor correcto. En la Figura 5.14 se observa que las instrucciones en las direcciones 4, 8, 12 y 16 no leen el valor calculado por la primera instrucción. Por tanto, es necesario adecuar la semántica del procesador segmentado a la semántica del lenguaje máquina, la cual permite que la instrucción consumidora se interprete inmediatamente después de la productora (distancia uno).

dir.	instrucción	ciclos 1	2	3	4	5	6	7	8	9	10	11	12
0	add **x6**, x5, x2	CP	BUS	D/L	ALU	M	FMTL	ES					
4	addi **x7**, x6, 0x0400		CP	BUS	D/L	ALU	M	FMTL	ES				
8	sub **x8**, x17, x6			CP	BUS	D/L	ALU	M	FMTL	ES			
12	ori **x9**, x6, 0x5555				CP	BUS	D/L	ALU	M	FMTL	ES		
16	and **x10**, x6, x3					CP	BUS	D/L	ALU	M	FMTL	ES	
20	xor **x12**, x6, x14						CP	BUS	D/L	ALU	M	FMTL	ES
	lecturas prematuras				X	X	X	X					
	lectura correcta								X				

Figura 5.14 *El procesador segmentado no mantiene el orden de lecturas y escrituras especificado por el programador.*

Para adecuar la semántica hay que detectar dependencias de datos entre instrucciones que se están interpretando concurrentemente. Una dependencia de datos no respetada por el procesador segmentado se denomina riesgo de datos. Una vez detectado un riesgo de datos el autómata de control debe emular un funcionamiento serie.

Para detectar un riesgo de datos debido a un registro del banco de registros es necesario disponer de información de las instrucciones. La información correspondiente a una instrucción como muy pronto está disponible en la etapa DL. Por otro lado, como los identificadores de registro se especifican en la propia instrucción es posible detectar el riesgo de datos en la etapa DL.

En la detección de un riesgo de datos se utilizan los identificadores de los registros fuente de la instrucción que ocupa la etapa DL y los identificadores destino de las instrucciones que están en etapas más avanzadas en el proceso de interpretación. Además, hay que utilizar señales de validación de los identificadores de registro.

La forma de emular un funcionamiento serie es suspender la interpretación de la instrucción que detecta el riesgo de datos y de las instrucciones más jóvenes que han empezado a interpretarse. Las instrucciones más viejas prosiguen su ejecución.

Para suspender la interpretación de una instrucción no se modifica el contenido de los registros de desacoplo de entrada de la etapa que ocupa la instrucción y etapas previas. En el procesador diseñado se suspende la interpretación de instrucciones en las etapas DL, BUS y CP (Figura 5.15).

Por otro lado, es necesario indicar a la etapa que sigue en secuencia, a la etapa donde se suspende la interpretación de la instrucción más vieja, que en el siguiente ciclo no procesa información válida y no debe actualizarse el estado del procesador. Está indicación de información inválida se propaga por las etapas hasta llegar a la última etapa.

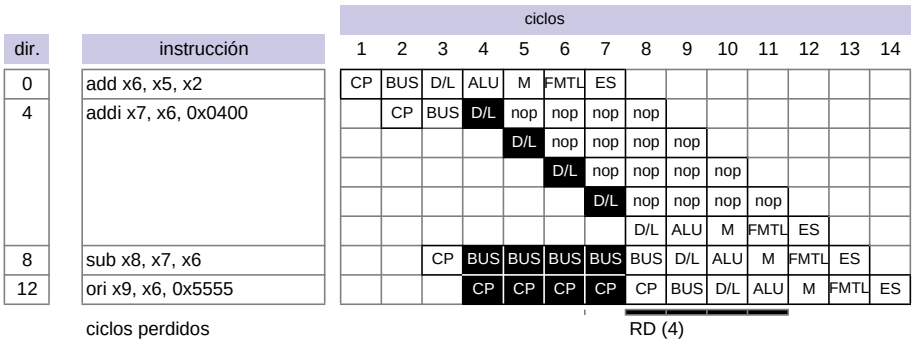


Figura 5.15 *Emulación de un funcionamiento serie para respetar las dependencias de datos.*

La acción de indicar que, en el siguiente ciclo, la etapa ALU no procesará información válida es similar a la inyección de una instrucción nop decodificada desde la etapa DL a la etapa ALU.

Para inhibir la actualización de un registro de desacoplo en el flanco ascendente de la señal de reloj, los registros de desacoplo disponen de una entrada denominada bloqueo.

La inyección de una instrucción nop decodificada hacia la etapa ALU solo afecta a las señales de control generadas en el decodificador[12].

En resumen, cuando existe una dependencia de datos verdadera debida a registros, entre instrucciones separadas una distancia menor de cinco instrucciones, se pierden ciclos.

Memoria de datos

Una instrucción de acceso a memoria puede leer o escribir en la memoria de datos. En la Figura 5.16 se muestra una secuencia de accesos a memoria. La lógica asociada en operaciones de escritura (FMTE) se utiliza en la etapa ALU. La lógica asociada en operaciones de lectura (FMTL) está ubicada en la etapa FMTL, la cual es posterior a la etapa M.

12. En el procesador diseñado todas las señales de control que se propagan desde DL a etapas posteriores toman el valor cero.

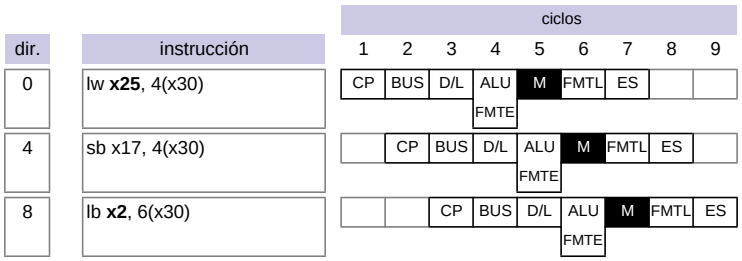


Figura 5.16 *Secuencia de accesos a memoria. En la etapa ALU se detalla la lógica relativa a un acceso a memoria.*

Tanto la lectura como la escritura[13] se efectúan en la etapa M del procesador segmentado (la granularidad mínima de escritura es un byte). Como la latencia de iniciación de instrucciones es un ciclo, los accesos a memoria se efectúan en el orden especificado por el programador y no se produce ningún riesgo de datos debido a memoria.

Unidad de secuenciamiento

En la Figura 5.17 se muestra un esquema donde se observan las etapas CP, BUS, DL y ALU.

El bucle hardware BHS1, de latencia uno, entre la etapa BUS y la etapa CP se utiliza para efectuar el secuenciamiento implícito. Como la latencia del bucle es igual a la latencia de inicio de interpretación de instrucciones, se respeta la semántica del lenguaje máquina.

El bucle hardware BHS2[14], de latencia tres, entre la etapa ALU y la etapa CP se utiliza para el secuenciamiento explícito. La semántica de todas las instrucciones de secuenciamiento indica que, después de la instrucción de secuenciamiento debe interpretarse la instrucción determinada por ella. Entonces, como la latencia de ejecución de una instrucción de secuenciamiento son tres ciclos (efectividad en la modificación o no del secuenciamiento) no se respeta la semántica del lenguaje máquina.

Por tanto, en el diseño descrito, la semántica del procesador segmentado no se adecua a la semántica del lenguaje máquina. En consecuencia hay que diseñar un mecanismo que adecue las semánticas.

13. La operación de escritura se efectúa en el flanco ascendente de la señal de reloj.
14. En este bucle solo se muestra un trazo más grueso en la señal de salida del multiplexor mSIC. Ahora bien, cualquier otra señal que determina este valor en el mismo ciclo pertenece al bucle hardware.

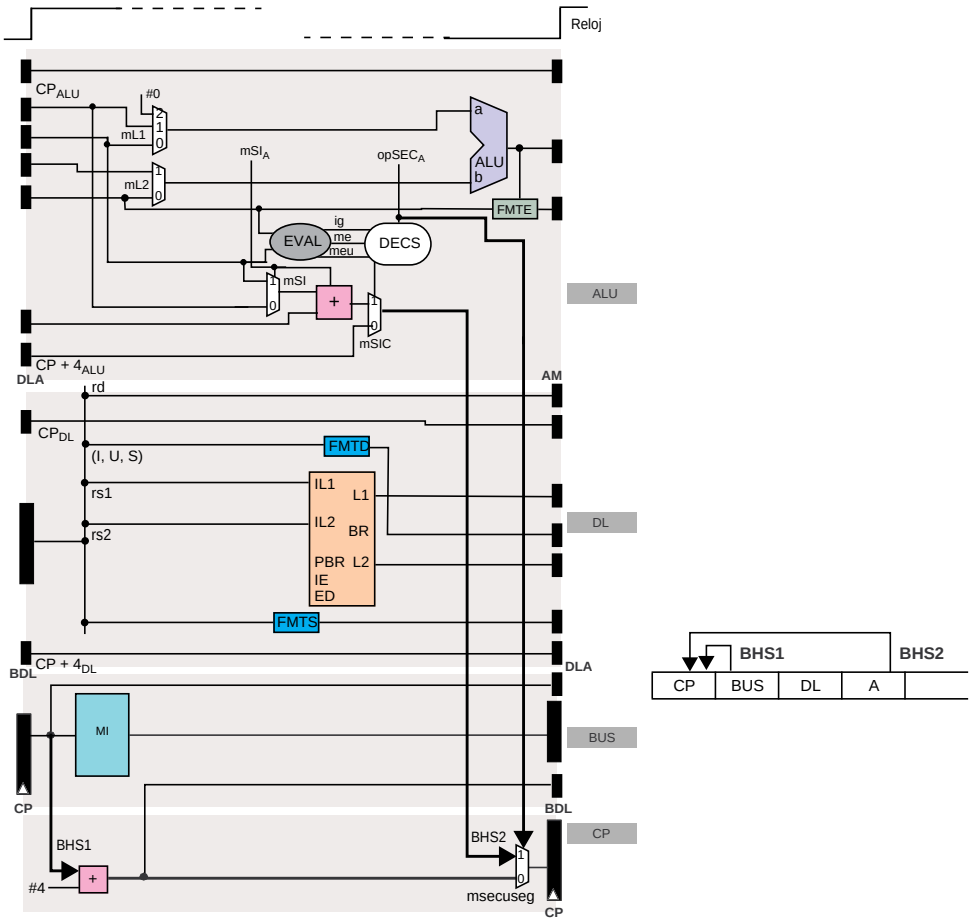


Figura 5.17 *Esquema de la unidad de secuenciamiento.*

En la Figura 5.18 se muestra un ejemplo de una secuencia de instrucciones que incluye una instrucción de secuenciamiento y se cumple la condición. Las dos instrucciones que siguen en secuencia, a la instrucción de secuenciamiento, no deben interpretarse. Esto es, no deben modificar el estado del procesador. Se está utilizando el secuenciamiento implícito (sumador ubicado espacialmente en la etapa CP, Figura 5.17). Estas dos instrucciones deben descartarse. La alternativa seleccionada es inyectar instrucciones nop desde la etapa BUS, una vez ha sido detectada la instrucción de secuenciamiento en la etapa DL. La inyección de instrucciones nop finaliza una vez se establece el secuenciamiento indicado por la instrucción de secuenciamiento.

El registro CP actúa como registro de desacoplo entre la etapa CP y la etapa BUS. Cuando la instrucción de secuenciamiento ocupa la etapa ALU se almacena en el registro CP, al final de ciclo, la dirección que determina la instrucción de secuenciamiento explícito. Esto es, la dirección cálculada en la lógica USE.

	dir.	instrucción	ciclos 1	2	3	4	5	6	7	8	9	10	11
	8	bne x3, x2, 1$	CP	BUS	D/L	ALU	M	FMTL	ES				
	12	add **x26**, x0, x4		CP	BUS	nop	nop	nop	nop	nop			
	15	or **x29**, x2, x6			CP	BUS	nop	nop	nop	nop	nop		
1$:	100	slt **x20**, x5, x6				CP	BUS	D/L	ALU	M	FMTL	ES	
	104	and **x9**, x9, x7					CP	BUS	D/L	ALU	M	FMTL	ES
		recursos		+	+	USE	+						

Figura 5.18 *Ejemplo de secuenciamiento en el procesador segmentado cuando se cumple la condición.*

Interacción entre riesgos

El análisis de la gestión de riesgos de datos y de secuenciamiento ha sido efectuado de forma independiente. Esto es, suponiendo que en un ciclo determinado solo se puede detectar un riesgo o que mientras se gestiona un tipo de riesgo no se detecta otro tipo de riesgo. En esta apartado se analizan las interacciones entre los tipos de riesgos.

Detección de un riesgo de datos en una instrucción de secuenciamiento. En la Figura 5.19 se muestra una secuencia de instrucciones donde la instrucción de secuenciamiento detecta un riesgo de datos. En primer lugar se gestiona el riesgo de datos y posteriormente el riesgo de secuenciamiento.

	dir.	instrucción	ciclos 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	8	lw **x3**, 0 (x17)	CP	BUS	D/L	ALU	M	FMTL	ES								
	12	bne x3, x2, 1$		CP	BUS	D/L	nop	nop	nop	nop							
							D/L	nop	nop	nop	nop						
								D/L	nop	nop	nop	nop					
									D/L	nop	nop	nop	nop				
										D/L	ALU	M	FMTL	ES			
	16	add **x26**, x0, x4			CP	BUS	BUS	BUS	BUS	BUS	nop	nop	nop	nop	nop		
	20	and **x9**, x9, x7				CP	CP	CP	CP	CP	BUS	nop	nop	nop	nop	nop	
1$:	100	slt **x20**, x5, x6									CP	BUS	D/L	ALU	M	FMTL	ES
		riesgos				RD	RD	RD	RD	RS	RS						

Figura 5.19 *Instrucción de secuenciamiento que detecta un riesgo de datos. Suponemos que se modifica el secuenciamiento.*

Detección de un riesgo de datos mientras se está gestionando un riesgo de secuenciamiento. En la Figura 5.20 se muestra una secuencia de instrucciones donde la instrucción (dir. 100) que se interpreta después de la instrucción de secuenciamiento depende de una instrucción load. Observamos que no se produce ningun riesgo mientras se gestiona el riesgo de secuenciamiento.

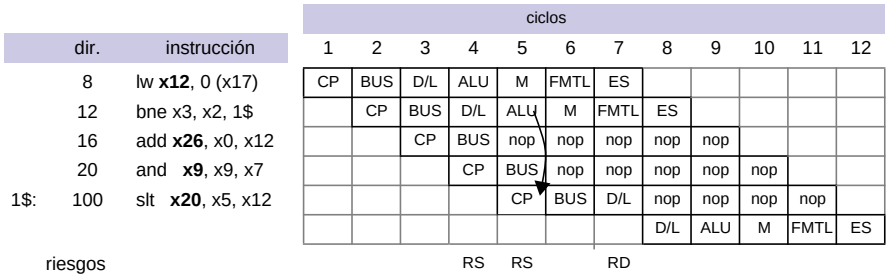


Figura 5.20 *La instrucción que sigue en secuencia a una instrucción de secuenciamiento es dependiente de la instrucción previa a la de secuenciamiento. Suponemos que se modifica el secuenciamiento.*

Lógica de interbloqueos de la segmentación

El control de riesgos se denomina lógica de interbloqueos de la segmentación y gestiona los riesgos emulando un funcionamiento serie.

La detección de riesgos se efectúa en la etapa DL. Para detectarlos se utiliza información de la instrucción en la etapa DL e información de las instrucciones más viejas en curso de interpretación.

Para efectuar las acciones de bloqueo e inyección descritas se incrementa la funcionalidad de algunos registros de desacoplo. En la Figura 5.21 se muestran las distintas funcionalidades.

Acciones	Funcionalidad
Normal	Transferir la entrada a la salida
Bloqueo	Retener la instrucción en la etapa
Inyección (Burbuja)	Inyectar una instrucción nop

Figura 5.21 *Funcionalidades de los registros de desacoplo.*

La circuitería que arropa a los registros de desacoplo, para efectuar las acciones descritas, se muestra en la parte izquierda de la Figura 5.22. Observe que en el diseño a) la señal inyectar es prioritaria respecto de la señal bloquear. Esto es, si bloquear = 1 e inyectar = 1 se inyecta una instrucción NOP.

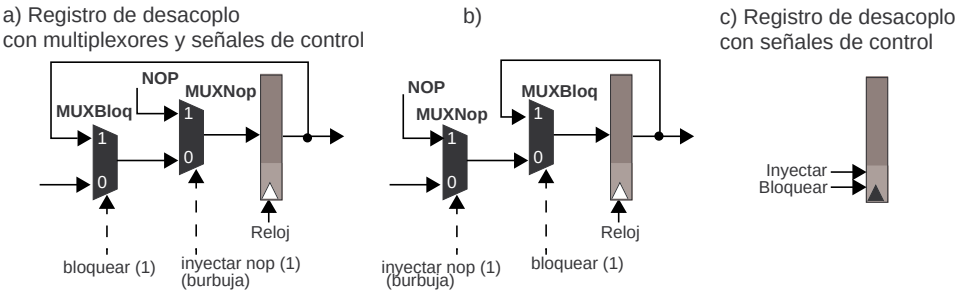


Figura 5.22 *a) y b) Lógica asociada a un registro de desacoplo que permite implementar los tres modos de funcionamiento y c) gráfico para representar un registro de desacoplo con las funcionalidades adicionales.*

En la parte derecha de la Figura 5.22 se muestra la simbología que utilizaremos para representar un registro de desacoplo con las funcionalidades descritas. En este símbolo solo se muestran las señales de control de los multiplexores y la marca de la señal de reloj se tinta de negro.

Por defecto todos los registros tienen una entrada de puesta a cero asíncrona que no se muestra (Pcero, Figura 5.23).

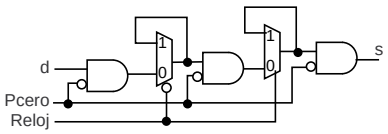


Figura 5.23 *Registro con señal de puesta a cero (Pcero) asíncrona.*

Riesgo de datos. Para resolver un riesgo de datos debido a registros es suficiente: a) parar el flujo de información en las etapas CP, B y DL bloqueando los registros de desacoplo de entrada de estas etapas, y b) inyectar una instrucción NOP desde la etapa DL a la etapa ALU.

En la Figura 5.24 se muestran los conjuntos de registros de desacoplo del camino de datos y las funcionalidades que deben disponer para efectuar el control de riesgos de datos. Cuando no se representa una entrada, por ejemplo inyectarA, el valor de la señal es cero.

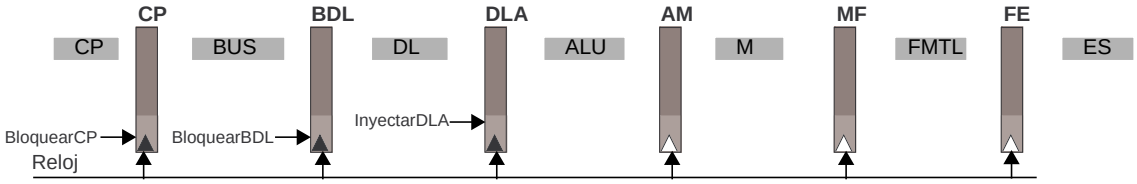


Figura 5.24 *Funcionalidades de los conjuntos de registros de desacoplo en el camino de datos para gestionar un riesgo de datos.*

Riesgo de secuenciamiento. Para resolver un riesgo de secuenciamiento es suficiente inyectar una instrucción NOP desde la etapa BUS a la etapa DL, mientras la instrucción, no establezca el secuenciamiento que determina.

En la Figura 5.25 se muestran los conjuntos de registros de desacoplo del camino de datos y las funcionalidades que deben disponer para efectuar el control de riesgos de secuenciamiento[15]. Cuando no se representa una entrada, por ejemplo inyectarA, el valor de la señal es cero.

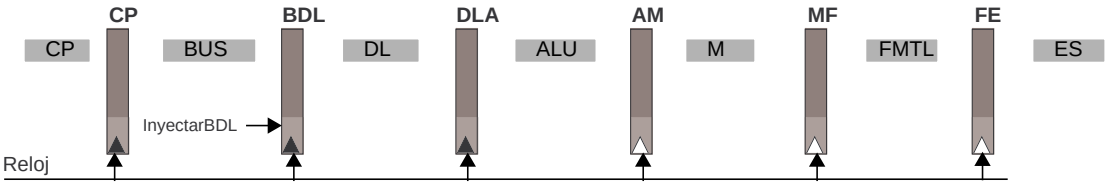


Figura 5.25 *Funcionalidades de los conjuntos de registros de desacoplo en el camino de datos para gestionar un riesgo de secuenciamiento.*

Diseño de la lógica de interbloqueos

La Lógica de InterBloqueos (LIB, Figura 5.26) se divide en varios módulos y submódulos.

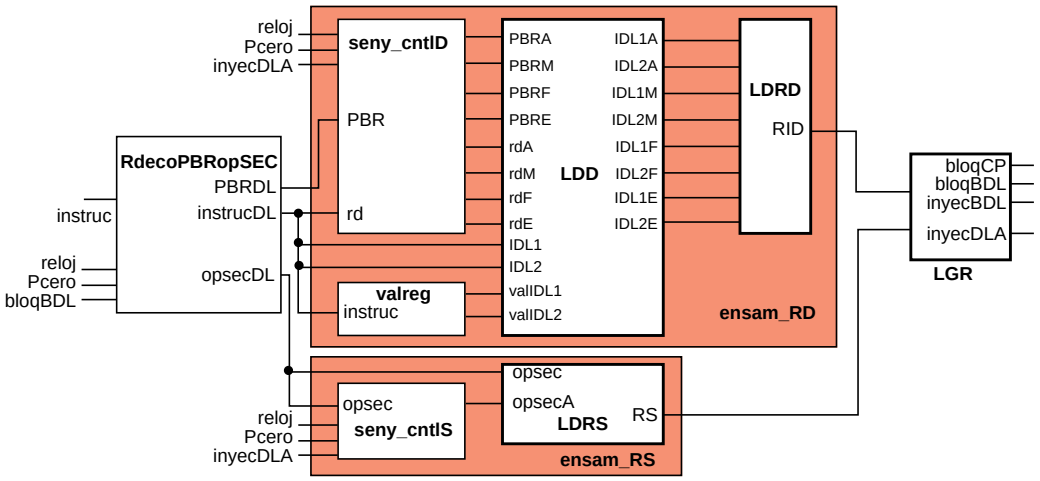


Figura 5.26 *Módulos utilizados en la lógica de interbloqueos (LIB). Los sufijos en las señales de entrada se refieren a etapa (A para ALU).*

15. En la implementación VHDL efectuada del registro BDL, el multiplexor MuxNop de la Figura 5.22 b) está ubicado en el fichero que describe la memoria MI (Figura 5.50). Esta decisión es debida a que una instrucción nop canónica (addi x0, x0, #0) tiene un codificación distinta a un vector de ceros. Por otro lado, un vector de ceros es una instrucción ilegal.

Un conjunto de los módulos y submódulos se utilizan para recrear localmente, en la lógica LIB, señales del camino de datos[16]. En concreto, estas señales son las utilizadas en la detección de riesgos de datos y de secuenciamiento.

El resto de módulos y submódulos se utiliza para detectar los riesgos de datos y de secuenciamiento y decidir la actuación que se lleva a término[17].

En la tabla de la Figura 5.27 se relacionan los módulos y submódulos que se utilizan para recrear señales del camino de datos.

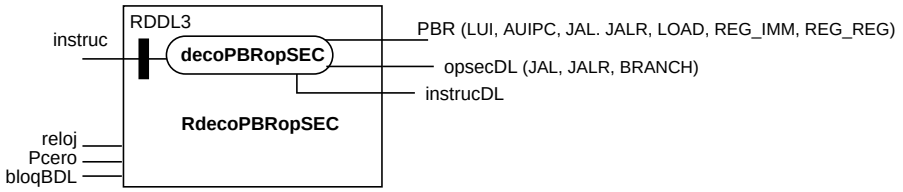
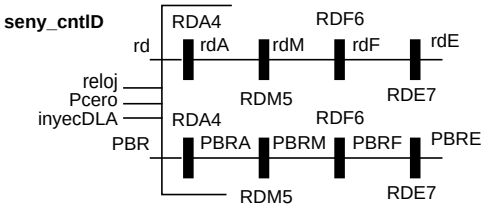
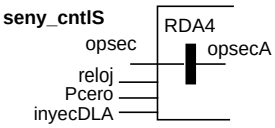
Módulos/submódulos	Descripción
RdecoPBRopSEC	En este módulo se recrean las señales PBR y opsec generadas por el decodificador en la etapa DL. 
seny_cntID	En este submódulo se recrea la propagación de las señales PBR y rd (IDE) por las etapas posteriores a DL. La señal PBRX, siendo X un acrónimo de etapa, indica si la instrucción que ocupa la etapa actualizará el banco de registros. La señal rdX es el identificador de registro destino en la etapa X. 
seny_cntIS	En este submódulo se recrea la propagación de la señal opsec hacia la etapa ALU (opsecA), la cual indica si la instrucción que ocupa la etapa ALU es de secuenciamiento. 

Figura 5.27 *Módulos y submódulos que recrean señales del camino de datos.*

Los submódulos utilizados en la deteción de riesgos están incluidos en los módulos:

- ensam_RD: Lógica de detección de riesgo de datos.
- ensam_RS: Lógica de detección de riesgo de secuenciamiento.

16. Por tanto, hay que efectuar las mismas accciones que en el camino de datos cuando se gestionan riesgos.
17. Recordemos que se emula un funcionamiento serie.

El módulo de actuación o de gestión de riesgos se denomina “Lógica de Gestión de Riesgos” (LGR).

El módulo ensam_RD, excluyendo los submódulos que recrean señales, incluye los siguiente submódulos:

- valreg: Lógica de validación. Módulo que genera las señales de validación de los identificadores de registro rs1 y rs2 de la instrucción que ocupa la etapa DL (Figura 5.28).

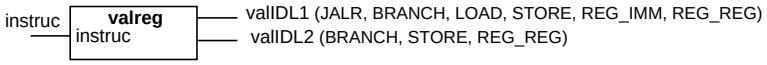


Figura 5.28 *Módulo valreg. Señal de validación de los registros fuente de una instrucción.*

- LDD: Lógica de detección de Dependencias de Datos debido a registros. En la tabla de la Figura 5.29 se describen las entradas y salidas del módulo LDD.
- LDRD: Lógica de Detección de Riesgos de Datos. Está lógica, en función de las dependencias de datos detectadas, genera la señal de riesgo de datos.

Entradas	Descripción
valIDL1, valIDL2	Validez de los identificadores de registros IDL1 (rs1) e IDL2 (rs1),
IDL1, IDL2	Identificadores de registro fuente rs1 (IDL1), rs2 (IDL2) del puerto IL1 y del puerto IL2
rd_A, rd_M, rd_F, rd_E	Identificadores de registro destino de las instrucciones que ocupan las etapas ALU, M, FMTL y ES
PBR_A, PBR_M, PBR_F, PBR_E	Permiso de escritura en el banco de registros de las instrucciones que ocupan las etapas ALU, M, FMTL y ES. Indica la validez de las señales rd_A, rd_M, rd_F y rd_E respectivamente

Salidas	Descripción
IDL1A, IDL2A	Indica que el contenido del registro fuente correspondiente (rs1, rs2) (IDL1, IDL2), de la instrucción que ocupa la etapa DL, se está calculando en la etapa ALU
IDL1M, IDL2M	Indica que el contenido del registro fuente correspondiente (rs1, rs2) (IDL1, IDL2), de la instrucción que ocupa la etapa DL, está en la etapa M
IDL1F, IDL2F	Indica que el contenido del registro fuente correspondiente (rs1, rs2) (IDL1, IDL2), de la instrucción que ocupa la etapa DL, está en la etapa FMTL
IDL1E, IDL2E	Indica que el contenido del registro fuente correspondiente (rs1, rs2) (IDL1, IDL2), de la instrucción que ocupa la etapa DL, está en la etapa ES

Figura 5.29 *Descripción de las señales de entrada y salida del módulo LDD.*

El módulo ensam_RS, excluyendo el módulo que recrea una señal, incluye el siguiente submódulo:

- LDRS: Lógica de Detección de Riesgos de Secuenciamiento, que genera la señal de riesgo de secuenciamiento.

El módulo LGR, a partir de las señales de riesgo de datos y riesgo de secuenciamiento, genera las señales de control para los registros de desacoplo (CP, BDL y DLA, Figura 5.24 y Figura 5.25).

Módulo LIB. El módulo LIB comprueba, en cada ciclo de reloj, si alguno de los operandos fuente de la instrucción que ocupa la etapa DL, aún no ha sido almacenado en el banco de registros. Esto es, aún no está disponible. También determina si la instrucción que ocupa la etapa DL es de secuenciamiento.

En la Figura 5.26 se muestra un esquema de los módulos utilizados en la lógica de interbloqueos y las señales de entrada y salida. Los puertos de lectura del banco de registros están asociados respectivamente a los campos rs1 (IDL1) y rs2 (IDL2) de la instrucción. La señal RID tiene que tomar el valor uno cuando se quiere gestionar un riesgo de datos. Respecto a la detección de un riesgo de secuenciamiento, la señal RS debe tomar el valor uno cuando se quiere efectuar una gestión del mismo. Esto es, mientras perdura el riesgo.

Las señales bloqCP y bloqDL se utilizan para bloquear el registro CP y el registro entre la etapa BUS y la etapa DL respectivamente. Las señales inyecBDL e inyecDLA se utilizan para inyectar una instrucción nop desde las etapas BUS y DL[18], respectivamente, a la siguiente etapa.

En el Apéndice 5.5 se describe la implementación de la gestión de riesgos.

Camino de datos y lógica de interbloqueos

En la Figura 5.30 se muestra el camino de datos con la lógica de interbloqueos. En la figura no se muestra de forma explícita la señal de reloj que controla los registros de desacoplo, el banco de registros (BR) y la memoria de datos (MD).

Las conexiones de las salidas del módulo LIB con los registros de desacoplo no se muestran. El conjunto de registros con un tramado de fondo que los engloba representa el registro DLA de la Figura 5.24. La señal inyecDLA es entrada en estos registros.

La señal inyecBDL es entrada del módulo VHDL que contiene la memoria de instrucciones (MI). Las señales bloqCP y bloqDL son entrada del registro CP y de los tres registros que están identificados como BDL (señales, CP, inst y CP + 4).

18. En la implementación VHDL efectuada el elemento que permite inyecta una instrucción nop desde la etapa BUS está ubicado en la descripción de la memoria MI.

Las señales bloqDL e inyecDLA también son entradas en el módulo LIB. La señal inyecDLA es necesaria ya que en el módulo LIB se recrean señales generadas en la etapa DL y propagadas a etapas posteriores. Por otro lado, en el módulo LIB se dispone de un registro de desacoplo que almacena la información de entrada en la etapa DL[19]. Por ello, es necesaria la señal bloqDL[20].

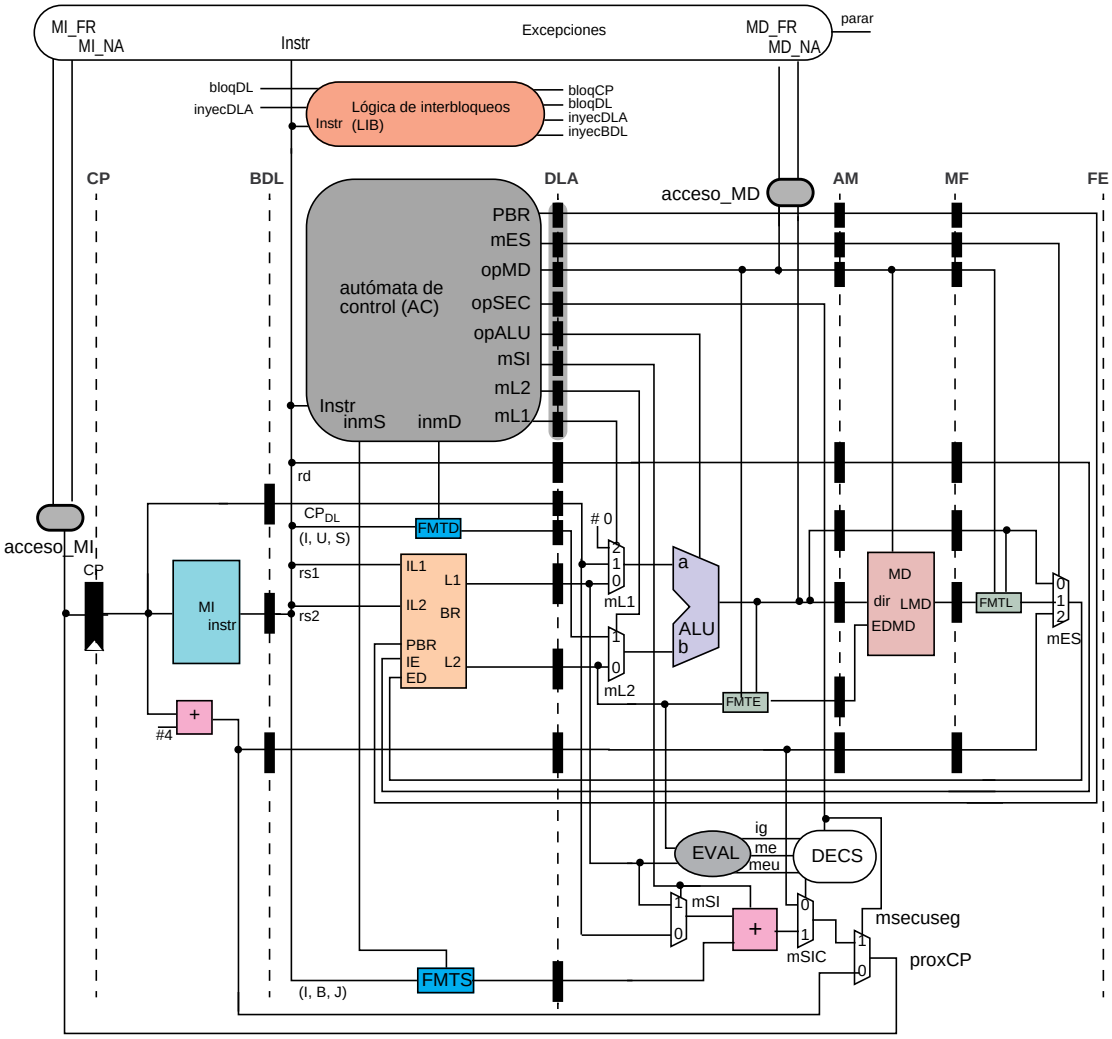


Figura 5.30 *Camino de datos del procesador segmentado con la lógica de interbloqueos.*

19. En el Apéndice 5.2 se muestra cómo se efectúa la descripción VHDL del procesador.
20. Recordemos que en un riesgo de datos hay que bloquear la información en la etapa DL e inyectar una nop decodificada desde DL hacia la etapa ALU.

Tiempo de ciclo

El tiempo de ciclo está predeterminado por la etapa con mayor retardo. Para determinar el tiempo de una etapa hay que tener en cuenta los elementos utilizados del camino de datos, su relación de precedencia y el retardo de cada uno de ellos. Para ello debemos tener en cuenta que la ubicación espacial de un componente en un camino de datos no determina su utilización en una etapa concreta.

Por otro lado, hay que tener en cuenta todos los tipos de instrucción que utilizan la etapa. Como el camino de datos es unificado, las salidas de algunos elementos de una etapa no son utilizados por algún tipo de instrucción. Por ello, hay que calcular el peor caso para cada etapa. Esto es, el retardo máximo que necesitan algunas instrucciones cuando ocupan la etapa.

Para ayudar en el cálculo del tiempo de ciclo, en la parte derecha de la Figura 5.31 se muestra una disposición de los elementos del camino de datos que ayuda a relacionar los retardos con el periodo de la señal de reloj. Las etapas se han dibujado en columna. Los registros de desacoplo en la salida de una etapa (parte derecha de cada etapa) son los registros de desacoplo en la entrada de la etapa representada encima de ella (parte izquierda de cada etapa). En la Figura 5.31 cada conjunto de registros de desacoplo se ha etiquetado con el mismo acrónimo que en la Figura 5.30.

Encima de las etapas se representa un ciclo de la señal de reloj, en el cual no se ha representado a escala el intervalo en cada nivel lógico (Figura 5.31). Los registros de desacoplo almacenan en el flanco ascendente de la señal de reloj los valores que hay en sus entradas. Estos valores, después de un retardo, son entradas estables de los elementos del camino de datos utilizados en las etapas. Después del retardo de estos elementos, se dispone de nuevos valores estables en la entrada de los registros de desacoplo. En el siguiente flanco ascendente de la señal de reloj los nuevos valores se almacenan en los registros de desacoplo.

En la Figura 5.31 los caminos etiquetados como BHBR, BHS1 y BHS2 son bucles hardware del camino de datos.

La salida del decodificador de instrucciones (DEC, Figura 4.50) es un conjunto de señales. Estas señales se propagan por las etapas para controlar los elementos del camino de datos y el encaminamiento. Para simplificar el dibujo, estas señales no se representan pero hay que tener en cuenta el retardo del decodificador en la etapa DL y el retardo de los registros de desacoplo al propagar las señales (Figura 5.30).

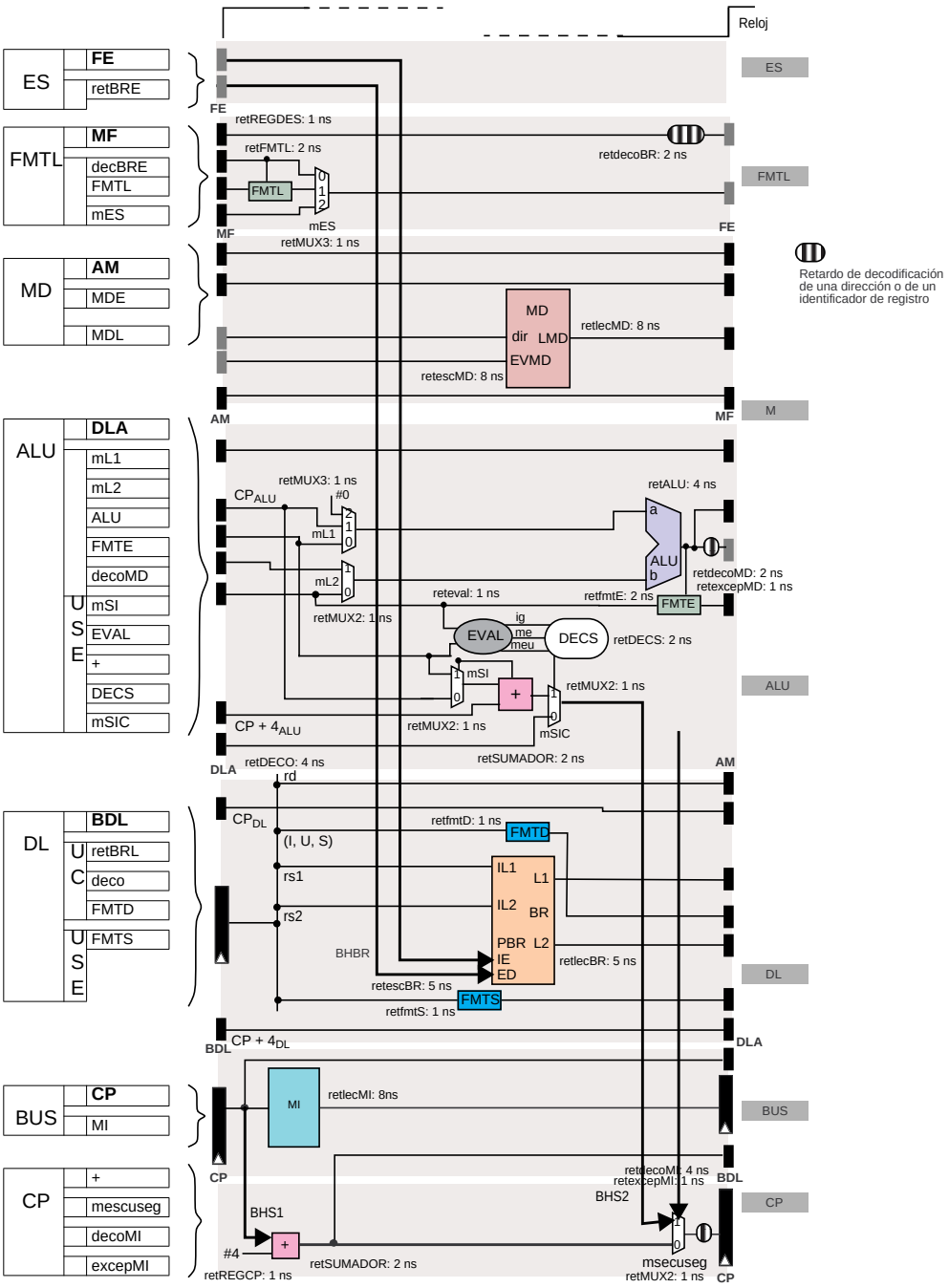


Figura 5.31 *Etapas del camino de datos y señales en el cronograma de retardos. No se muestra el decodificador ni la lógica de interbloqueos.*

Para facilitar el cálculo del tiempo de ciclo utilizaremos un cronograma donde se representa el retardo de los elementos del camino de datos. En el cronograma se han tenido en cuenta las etapas para agrupar los componentes del camino de datos. En la Figura 5.32 se muestran los acrónimos utilizados para denominar componentes del camino de datos.

Todos deben interpretarse como el retardo del componente. Alguno de los 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.

ES		**FE**	
		retBRE	Retardo en la actualización de un registro del banco de registros
FMTL		**MF**	Registro de desacoplo de entrada de la etapa FMTL
		decBRE	Decodificador del puerto de escritura del banco de registros. No está representado en las figuras
		FMTL	Módulo para formatear el dato con el cual se actualiza el banco de registros
		mES	Multiplexor en la etapa ES
MD		**AM**	Registro de desacoplo de entrada de la etapa M
		MDE	Retardo de escritura en memoria
		MDL	Retardo de lectura en memoria
ALU		**DLA**	Registro de desacoplo de entrada de la etapa ALU
		mL1	Multiplexor para seleccionar el operando a que utiliza la etapa ALU
		mL2	Multiplexor para seleccionar el operando b que utiliza la etapa ALU
		ALU	Módulo ALU
		FMTE	Módulo para formatear el dato con el cual se actualiza memoria
		excepMD	Lógica de excepción
		decoMD	Decodificador de la dirección de memoria
	USE	mSI	Multiplexor
	USE	EVAL	Unidad de evaluación de la condición
	USE	+	Sumador
		DECS	Módulo de secuenciamiento condicional
		mSIC	Multiplexor

DL		**BDL**	Registro de desacoplo de entrada de la etapa DL
	UC	retBRL	Retardo en la lectura de un registro del banco de registros
	UC	deco	Retardo del decodificador
	UC	FMTD	Formateador del campo literal en la UC
	USE	FMTS	Formateador del campo literal en la USE
	LIB	LDD	Dependencias de Datos debido a registros
	LIB	RD	Riesgo de Datos
	LIB	RS	Riesgo de Secuenciamiento
	LIB	LGR	Lógica de Gestión de Riesgos
BUS		FMTD	Registro CP
		MI	Retardo de lectura de la memoria de instrucciones
CP		+	Sumador
		mescuseg	Multiplexor cuya salida es entrada en el registro CP
		decoMI	Decodificador de la dirección de memoria
		excepMI	Lógica de excepción

Figura 5.32 *Acrónimos utilizados en los diagramas de retardos.*

Los retardos de los componentes utilizados se detallan en el Apéndice 5.3. Estos retardos no son representativos de un diseño. Solo son de utilidad para efectuar los cálculos de retardo que se solicitan.

Consideraciones

Para determinar el tiempo de una etapa hay que suponer el caso peor. Por ejemplo, en la etapa FMTL, una instrucción load requiere que se utilice el módulo FMTL[21]. En cambio, otras instrucciones solo requieren atravesar el multiplexor (mES). Para esta etapa el peor caso es una instrucción load.

En la etapa CP también hay que considerar los circuitos combinacionales correspondientes a la USE ubicada espacialmente en la etapa ALU.

Para calcular el retardo en las etapas DL y M hay que tener en cuenta que:

- Los registros de desacoplo mostrado de forma tramada en las etapas ES y M no hay que tenerlos en cuenta al calcular el retardo. Estos registros representan que el acceso es síncrono con el flanco ascendente de la señal de reloj. Es por ello que los dos registros tramados de la Figura 5.31 en la etapa ES no han sido representados en las otras figuras que muestran el camino de datos. Respecto a los registros de entrada de MD se ha optado por representarlos, en otras figuras que muestran el camino de datos, con el objetivo de facilitar la identificación de la etapa M.
- En la etapa de escritura (ES) un registro del banco de registros se actualiza de forma síncrona en el flanco ascendente de la señal de reloj.
- En la etapa M el acceso a la memoria de datos (MD) está sincronizado con el flanco ascendente de la señal de reloj.
- En la etapa DL la lectura de los operandos se efectúa de forma asíncrona.
- En una acción de escritura de un registro del banco de registros hay que tener en cuenta el tiempo del decodificador (decBRE, etapa FMTL).
- En un mismo ciclo, el valor con el que se actualiza un registro no es factible leerlo.
- En una acción de lectura o una escritura en la MD hay que tener en cuenta la decodificación de la dirección (etapa ALU).
- Para calcular el tiempo de ciclo supondremos que el retardo de todos los componentes, en cualquier camino, ha transcurrido antes de finalizar el ciclo de reloj.

Cronograma de retardos

En la Figura 5.33 se muestra el cronograma donde se representa el retardo de los componentes del camino de datos y su encadenamiento en un procesador segmentado sin control de riesgos. En la fila superior se representa un ciclo de la señal de reloj. Los

21. Siendo precisos, en el caso de instrucciones que leen media palabra o un byte.

componentes del camino de datos se han agrupado de la misma forma que en la Figura 5.31. En el cronograma, mediante líneas contínuas se muestran algunas de las precedencias entre elementos del camino de datos. Observe las precedencias entre elementos cuyo retardo se representa en etapas distintas. Estas interacciones son debidas a los bucles hardware (BHBR, BHS1 y BHS2) existentes en el camino de datos.

La duración de cada uno de los niveles lógicos de la señal de reloj, si es necesario determinarlo para un funcionamiento correcto, está determinado por las etapas que utilizan componentes secuenciales o síncronos (ES y M). Una vez se han analizado estas etapas se puede determinar el tiempo en cada nivel.

El cálculo de los retardos se efectúa empezando por la última etapa de la segmentación y siguiendo de etapa en etapa hasta la etapa CP. Notemos que los bucles hardware tienen como fuente del bucle etapas posteriores a la etapa destino.

Después del flanco ascendente de la señal de reloj las señales son estables en la salida de los registros de desacoplo después de 1 ns (Figura 5.33). Notemos que en la memoria de datos (MD) y en el banco de registros no se tiene en cuenta el retardo. Los componentes se actualizan en el flanco ascendente de la señal de reloj.

En la etapa ES se actualiza el banco de registros en el flanco ascendente de la señal de reloj (bucle BHBR).

- En una instrucción store o de secuenciamiento no se utiliza esta etapa, salvo que en la instrucción de secuenciamiento se almacene la dirección de retorno.

En la etapa FMTL el decodificador del puerto de escritura del banco de registros (decBRE, ubicado espacialmente en la etapa FMTL) tiene la entrada estable después del retardo del registro de desacoplo MF.

- En una instrucción store o de secuenciamiento no se utiliza esta etapa, salvo que en la instrucción de secuenciamiento se almacene la dirección de retorno.
- En una instrucción load el valor que debe almacenarse en el registro no está disponible hasta que ha transcurrido el retardo en los elementos FMTL y mES. El retardo del flujo de información por estos elementos es mayor que el retardo de decBRE.
- En otras instrucciones solo hay que considerar el retardo del multiplexor mES.

En la etapa M, el acceso a los bancos de memoria es síncrono con la señal de reloj (flanco ascendente). .

- El retardo es el mismo para lectura y escritura (8 ns).

En la etapa ALU el retardo está determinado por la secuencialidad de los módulos mL1, ALU y decoMD. El retardo en el módulo FMTE es menor que la suma de los retardos en los módulos anteriores. El retardo en la lógica USE también es menor. Por tanto, el retardo de la etapa ALU es 8 ns.

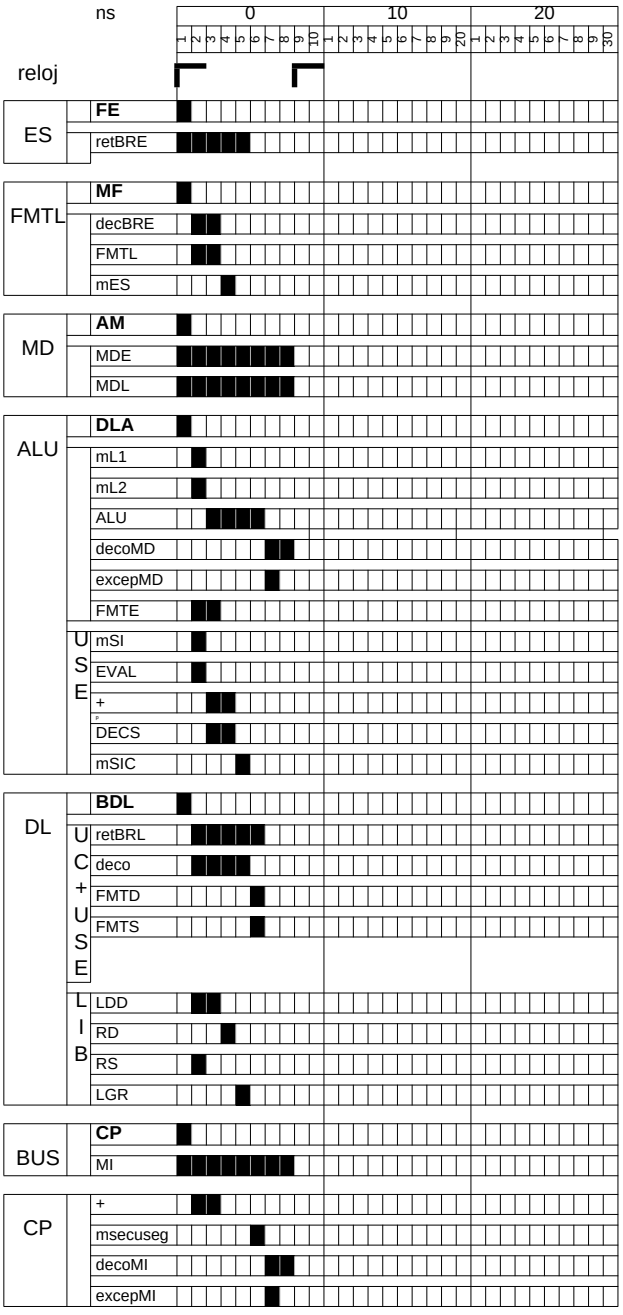


Figura 5.33 *Cronograma de retardos del camino de datos segmentado.*

En la etapa DL, una vez es estable la señal en la salida del registro de desacoplo de entrada (BDL), se considera el retardo del decodificador y de los elementos FMTD de la UC y FMTS de la USE. Por otro lado, hay que considerar el retardo de la lógica de interbloqueos (LDD, LDRD, LDRS, LBGR) y el retardo de lectura del banco de registros. Por ahora, el periodo está determinado por las etapas ALU y M. En consecuencia, no es factible escribir y leer, el valor escrito, en un mismo registro durante el periodo de la señal de reloj.

En la etapa BUS el retardo está determinado por la lectura de la memoria de instrucciones (8 ns).

En la etapa CP, la entrada uno del multiplexor msecuseg es estable después de 5 ns (salida del multiplexor mSIC), mientras que la entrada de selección es estable después de 1 ns (retardo del registro de desacoplo). La entrada cero es estable después del retardo del sumador (2 ns). Si se selecciona la entrada cero del multiplexor msecuseg, la salida de éste es estable después de 4 ns (secuenciamiento implícito). En caso contrario, la salida es estable después de 6 ns (secuenciamiento explícito). Posteriormente hay que considerar el retardo del decodificador de direcciones de la MI (2 ns). Por tanto, el retardo en la etapa CP es 8 ns.

En la Figura 5.34 se muestra el retardo máximo de cada etapa y el tipo de instrucciones que determina este retardo máximo. El tiempo de ciclo es 8 ns y la señal de reloj no tiene restricción en la duración de los niveles lógicos. Todos los elementos de almacenamiento utilizan el flanco ascendente de la señal de reloj.

	etapa						
instrucción	CP	B	DL	A	M	F	E
cálculo		X	X	X			X
secuenciamiento	X	X	X				X[a]
load		X	X	X	X	X	X
store		X	X	X			
Tiempo de etapa (ns)	8	8	6	8	8	4	5

Tiempo de ciclo mínimo	8
Tiempo en el nivel 1	S.R.
Tiempo en el nivel 0	S.R.

a. Solo las de llamada a procedimiento

Figura 5.34 *Resumen del cálculo del tiempo de ciclo. S.R. indica sin restricción.*

Señal de reloj, señal de inicio y elementos de almacenamiento

El registro CP y los registros de desacoplo se actualizan por defecto en el flanco ascendente de la señal de reloj (Figura 5.35). Al activar la señal inicializar (señal Pcero en el programa de prueba) el registro CP y los registros de desacoplo toman el valor cero. En consecuencia se lee la instrucción que está en la dirección cero de la memoria de instrucciones (MI).

El banco de registros se actualiza en el flanco ascendente de la señal de reloj si el permiso de escritura (PBR) está activado (Figura 5.35). La lectura es asíncrona.

Un banco de la memoria de datos se actualiza cuando el permiso de escritura está activado (PMD), el banco ha sido seleccionado (SCx) y la señal de reloj está en un flanco ascendente (Figura 5.35).

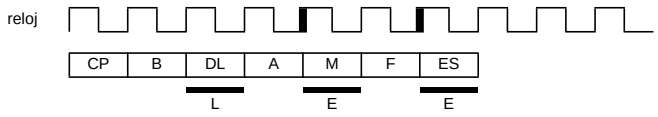


Figura 5.35 *Señal de reloj y actualización de los elementos de memorización. El retardo de actualización no está a escala.*

Un registro de desacoplo no se actualiza si está activada la señal bloquear (Figura 5.22). De forma síncrona, la salida de un registro de desacoplo toma el valor cero o la codificación de una nop canónica cuando la señal inyectar está activada en el flanco ascendente de la señal de reloj.

Simulación

Los ficheros que utiliza Quartus (Rproc_MD_MI.qsf y Rproc_MD_MI.qpf) se encuentran en el directorio LAB5/PROC_SEGMENTADO/ENSAMBLADO/QUARTUS.

En los subdirectorios incluidos en LAB5 solo se incluyen los ficheros modificados respecto al diseño serie (LAB4). En particular, han sido modificadas las especificaciones de MI y MD. En esta práctica, tanto la lectura como la escritura, si es el caso, es síncrona con la señal de reloj (flanco ascendente).

En el fichero Rproc_MD_MI.qsf se puede observar la ubicación de los ficheros que se utilizan. El árbol de directorios de la práctica 4 debe ser accesible. Por tanto, los directorios LAB4 y LAB5 deben estar en el mismo nivel.

Una vez ha sido abierto el proyecto con Quartus, hay que dar la orden “Processing _> Start -> Start Analysis & Elaboration” . Esta orden elabora el diseño. Una vez elaborado el diseño sin errores, mediante la orden “Tools _> Netlist Viewers -> RTL Viewer” se puede comprobar la elaboración efectuada[22].

El fichero Rproc_MD_MI.vhd contiene la descripción en VHDL del procesador descrito. Este fichero está ubicado en el directorio LAB5/PROC_SEGMENTADO/ENSAMBLADO/CODIGO. El esquema RTL, elaborado por Quartus, se muestra en la Figura 5.36.

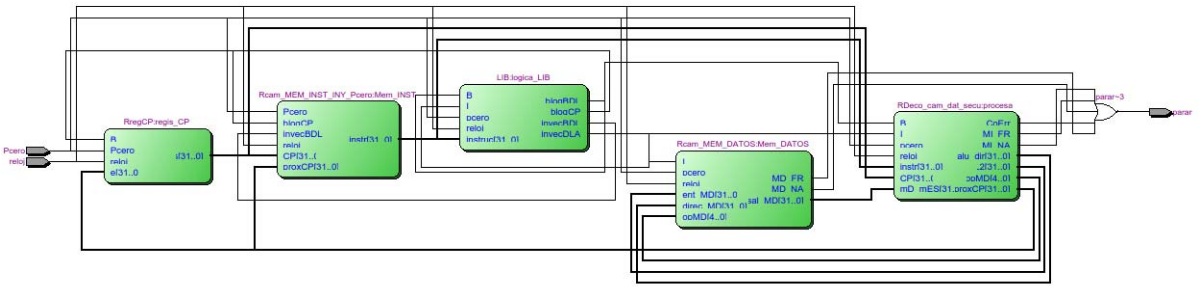


Figura 5.36 *Esquema RTL del procesador segmentado.*

Preparación de la simulación

El fichero de comprobación, denominado prueba_Rproc_MD_MI.vhd se encuentra en el directorio LAB5/PROC_SEGMENTADO/ENSAMBLADO/PRUEBAS. En este directorio también se encuentra el fichero para formatear las señales que se visualizan en la ventana temporal.

El fichero prueba_Rproc_MD_MI.vhd utiliza los mismos genéricos que el fichero parejo de la práctica 4. Este fichero está ubicado en el directorio LAB5/PROC_SEGMENTADO/ENSAMBLADO/PRUEBAS.

Simulación con ModelSim

Para efectuar una simulación efectúe los mismos pasos descritos en la práctica 4 (página 305).

22. En los ficheros que se suministra de la lógica LIB hay que efectuar una descripción de la lógica que se diseña en esta práctica (Apéndice 5.2). Los ficheros que se suministran contienen la declaración de la interface y la especificación de una arquitectura que no es correcta.

Simulación de un programa concreto

Los programas se encuentran en la misma ubicación que en la práctica 4 (página 306).

Evolución de las señales del camino de datos

La ventana de tiempo que dispone ModelSim es útil para visualizar la propagación de una instrucción por las etapas y las interacciones entre ellas. En la ventana de tiempo se muestran algunas de las señales del camino de datos, las cuales están ordenadas para facilitar la observación. Notemos que en la ventana de tiempo la propagación de una instrucción no se puede observar en una fila, ya que una fila muestra una señal. Por ello, hay que observar distintas filas.

Descripción funcional de la ventana temporal. En la ventana temporal de ModelSim, las señales se han dispuesto de forma que señales contíguas corresponden a la misma etapa o a etapas adyacentes. Las señales correspondientes a la etapa CP están en la parte inferior y las señales correspondientes a la etapa ES están en la parte superior de la ventana temporal (parte derecha de la Figura 5.37). Por tanto, en un ciclo determinado, las señales en la parte inferior de la ventana se corresponden con la instrucción más joven y las ubicadas en la parte superior se corresponden con la instrucción más vieja. Entonces, para observar la propagación de una instrucción hay que moverse hacia la derecha (incrementa el tiempo) y hacia arriba (siguiente etapa). Esto es, en diagonal. Respecto a los bucles hardware, estos se observan de arriba a abajo en la ventana de tiempo.

En la Figura 5.37 se muestra un ejemplo. En la parte izquierda de la figura se muestra una secuencia de instrucciones. En la parte derecha se muestra la evolución temporal. Para identificar una instrucción se utiliza la dirección de la instrucción como subíndice en cada una de las etapas. En este ejemplo se produce un riesgo de datos y se establece el valor del registro CP en una instrucción de secuenciamiento.

	dir.	instrucción
	8	lw **x3**, 0 (x17)
	12	bne x3, 1$
	16	add **x26**, x0, x4
	20	and **x9**, x9, x7
	24	sub **x7**, x21, x31
		. . .
1$:	100	slt **x20**, x5, x6

Señal de riesgo de datos

Señal de riesgo de secuenciamiento

ETAPAS	ciclos 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ES							ES_8	nop	nop	nop	nop	ES_{12}	nop	nop	ES_{100}
FMTL						$FMTL_8$	nop	nop	nop	nop	$FMTL_{12}$	nop	nop	$FMTL_{100}$	
M					M_8	nop	nop	nop	nop	M_{12}	nop	nop	M_{100}		
ALU				ALU_8	nop	nop	nop	nop	ALU_{12}	nop	nop	ALU_{100}			
DL			DL_8	DL_{12}	DL_{12}	DL_{12}	DL_{12}	DL_{12}	nop	nop	DL_{100}				
BUS		BUS_8	BUS_{12}	BUS_{16}	BUS_{16}	BUS_{16}	BUS_{16}	BUS_{16}	BUS_{20}	BUS_{100}					
CP	CP_8	CP_{12}	CP_{16}	CP_{20}	CP_{20}	CP_{20}	CP_{20}	CP_{20}	CP_{100}						
				RD	RD	RD	RD								
								RS	RS						

Figura 5.37 *Procesador segmentado. Observación de la propagación de instrucciones en la ventana temporal.*

Ventana temporal de ModelSim. En la ventana temporal de ModelSim se muestran etiquetas que están asociadas a señales del camino de datos. Estas etiquetas están ordenadas para asociarlas a etapas de forma simple. La asociación entre señales y etapas se muestra en la Figura 5.38.

BR	BR	Banco de registros
FMTL	PE_FMTL	Permiso de escritura en el BR de la instrucción que ocupa la etapa FMTL
	IDE_FMTL	Identificador de registro destino de la instrucción que ocupa la etapa FMTL
M	PE_M	Permiso de escritura en el BR de la instrucción que ocupa la etapa M
	IDE_M	Identificador de registro destino de la instrucción que ocupa la etapa M
	PE_MD_M	Permiso de escritura en MD de la instrucción que ocupa la etapa M
ALU	PE_ALU	Permiso de escritura en el BR de la instrucción que ocupa la etapa ALU
	IDE_ALU	Identificador de registro destino de la instrucción que ocupa la etapa ALU
	opALU	Señal para controlar las operaciones en la ALU
	opSEC	Control del secuenciamiento
	opMD	Señal para controlar las operaciones en la memoria de datos (MD)
	me_meu_ig[a]	Salida del circuito que evalúa la condición
	sel_muxSIC	Selección del multiplexor mSIC (secuenciamiento)
	muxSIC	Salida del multiplexor mSIC (secuenciamiento)
	selmxsecseg	Señal de selección del multiplexor mxsecseg (secuenciamiento)
	mxsegsec	Próximo valor del registro CP
DL	PE_DL	Permiso de escritura en el BR de la instrucción que ocupa la etapa DL
	IDE_DL	Identificador de registro destino de la instrucción que ocupa la etapa DL
	val_IDL1	Validez del campo rs1 (IDL1) de la instrucción que ocupa la etapa DL
	IDL1	Campo rs1 (IDL1) de la instrucción que ocupa la etapa DL
	val_IDL2	Validez del campo rs2 (IDL2) de la instrucción que ocupa la etapa DL
	IDL2	Campo rs2 (IDL2) de la instrucción que ocupa la etapa DL
	instruc_DL	Instrucción en la etapa DL. Salida del registro BDL
BUS	instruc_BUS	Salida de la MI
	CP	Salida del registro CP
CP	prox_CP	Entrada del registro CP
	ciclo	Contador de ciclos
	reloj	Señal de reloj

a. Las señales están ordenadas, de izquierda a derecha, en el sentido en que se muestran.

Figura 5.38 *Etiquetas utilizadas en las señales que se muestran en la ventana de tiempo.*

En al Figura 5.39 se relacionan otras señales que se muestran el la ventana temporal de Modelsim. Estas señales indican la detección y necesidad de gestión de riesgos de datos y de secuenciamiento. Adicionalmente se relacionan las señales que se utilizan en los registros de desacoplo, en los cuales son necesarias, para gestionar cada uno de los riesgos.

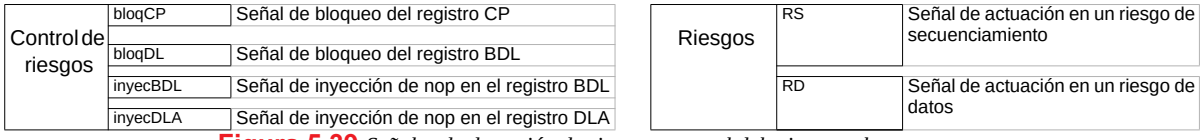


Figura 5.39 *Señales de detección de riesgo y control del mismo en la ventana temporal de ModelSim.*

En la Figura 5.40 se muestra, en la ventana temporal, la evolución de señales en el camino de datos cuando el procesador interpreta una secuencia de instrucciones. En la parte izquierda de la Figura 5.40 se observan las etiquetas de las señales y su asociación con la etapa del procesador. Para facilitar la compresión, la salida del multiplexor mxsecseg ha sido añadida en distintas posiciones de la ventana temporal. Esta señal está en la etapa A_4 y en la etapa CP_1. En esta última etapa se denomina prox_CP.

Debajo del diagrama de la ventana de tiempo se muestra el diagrama temporal que se utiliza usualmente al mostrar la interpretación de instrucciones en un procesador segmentado. Notemos que en una columna se identifican las etapas en el mismo orden que en la ventana de tiempos. Sin embargo, en el diagrama temporal usual, una instrucción se observa en una fila y en la ventana de tiempo se observa en diagonal (tramas).

En el ciclo 29, se observa la dirección “x00000118” 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 29, se observa que se lee la instrucción “0x00D70733” (add x14, x14, x13). Esta instrucción se decodifica en el siguiente ciclo (30).

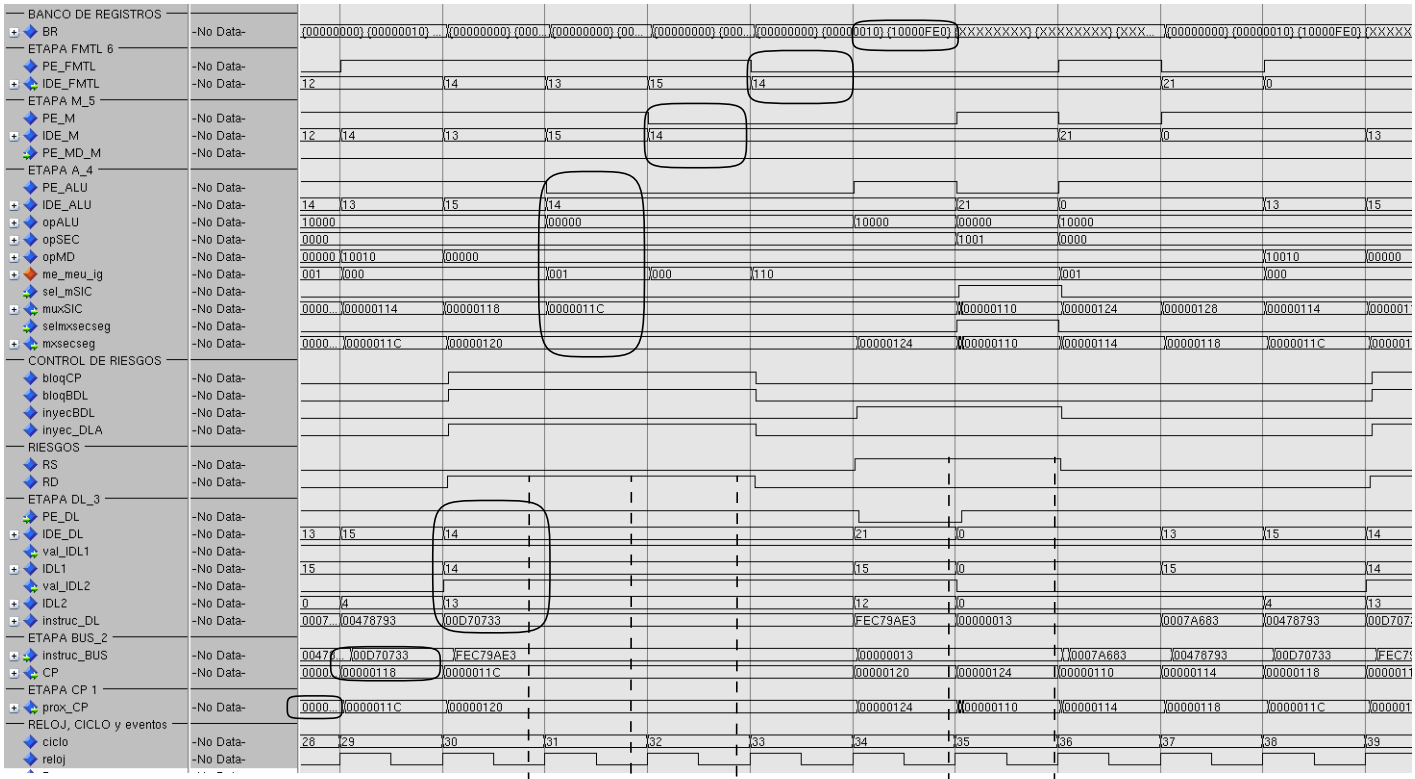
En el ciclo 30, en la etapa DL, las señales muestran que el campo rs1 (IDL1) de la instrucción tiene el valor 14 e identifica un registro (val_IDL1 = 1). El campo rs2 (IDL2) de la instrucción tiene el valor 13 e identifica un registro (val_IDL2 = 1). El campo rd (IDE_DL) de la instrucción tiene el valor 14 e identifica un registro (PE_DL = 1).

En el ciclo 30 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 31, 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 30 perdura durante los ciclos 31 hasta 32. En el ciclo 33 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 34, se observa, en parte, la propagación de la instrucción que ocupaba la etapa DL en el ciclo previo (PE_ALU = 1, IDE_ALU = 14, opALU = 10000. En los siguientes ciclos (35, 36 y 37) esta instrucción se propaga por las etapas M (PE_M = 1, IDE_M = 14), F (PE_F = 1, IDE_F = 14) y ES. La actualización del registro x14 en el ciclo 37 puede observarse expandiendo la etiqueta del banco de registro (BR).



		29	30	31	32	33	34	35	36	37	38
00000110	lw x13, x000(x15)	ALU	M	FMTL	ES						
00000114	addi x15, x15, x004	D/L	ALU	M	FMTL	ES					
00000118	**add x14, x14, x13**	BUS	DLB	nop	nop	nop	nop				
				DLB	nop	nop	nop	nop			
					DLB	nop	nop	nop	nop		
						DL	ALU	M	FMTL	ES	
0000011C	bne x15, x12, xFFA		CP	BUS	BUS	BUS	DL	ALU	M	FMTL	ES
00000120	addi x0 , x0 , x000			CP	CP	CP	BUS	nop	nop	nop	nop
00000124	addi x0 , x0 , x000						CP	BUS	nop	nop	nop
00000110	lw x13, x000(x15)							CP	BUS	DL	DL
Riesgos			RD	RD	RD		RS	RS			

Figura 5.40 *Evolución de las señales en la ventana de tiempo al ejecutar un trozo de código. En la parte izquierda se han agrupado las señales por etapas. Los óvalos en la ventana de tiempo identifica una instrucción. En la parte inferior izquierda se muestra la secuencia de instrucciones que se interpreta. En la parte inferior de la figura se muestra el diagrama temporal utilizado usualmente para facilitar la asociación entre señales etapas e instrucciones.*

Durante los ciclos en que las instrucciones nop inyectadas están en la etapa FMTL (33, 34 y 35) 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 31, 32, 33) no se modifica el secuenciamiento (opSEC = 0).

En el ciclo 34 la etapa DL está ocupada por una instrucción de secuenciamiento (0xFEC79AE3, “bne x15, x12, xFFA”). En este ciclo y en el siguiente, desde la etapa BUS (inyecBDL = 1), se inyecta un instrucción nop (0x 00000013, observada en instruc_BUS). El secuenciamiento se establece en el ciclo 35 y se observa en el ciclo 36 (salida de CP = 0x00000110”). En el ciclo 35 la instrucción de secuenciamiento ocupa la etapa ALU y la salida del multiplexor muxsecseg es igual a “x00000110”. Este valor se almacena en el registro CP al96inicio del siguiente ciclo (36). En el ciclo 36 la salida de la etapa BUS (instruc_BUS) es “0x0007A683” (lw x13, x000(x15)).

Notemos que las instrucciones nop inyectadas desde la etapa BUS actualizan el registro x0 (PE_* = 1). Ahora bien, como este registro no se actualiza en ningún caso, por especificación del L.M., no se modifica el estado del procesador.

Información textual de la simulación

En la ventana “Transcript” de Modelsim se muestra información para cada intrucción interpretada y, al finalizar la simulación, se muestra el contenido del banco de registros y de las memorias. El formato utilizado para mostrar las posiciones de almacenamiento está descrito en la práctica del procesador serie (Figura 4.66, Figura 4.67). El formato de la información relativo a la interpretación de instrucciones se modifica significativamente para mostrar el progreso de las instrucciones por las etapas. Toda la información que se muestra también se almacena en el fichero de resultados ubicado en el directorio RESULTADOS.

En una fila se representa la información mostrada en la Figura 5.41. Notemos que la información se corresponde a la instrucción que ocupa la etapa. Respecto a la etapa CP, etiquetado como prox_CP, se muestra el valor que se almacenará en el registro CP al finalizar el ciclo. Respecto a la etapa DL se muestra en lenguaje ensamblador (LE) la instrucción que ocupa la etapa y en la siguiente columna el acrónimo de la etapa. Para representar la información de una instrucción en una etapa concreta se utilizan acrónimos que se detallan posteriormente. Las dos últimas columnas de cada fila indican la detección (1) o no (0) de riesgos de datos (RD) y de secuenciamiento (RS) y su gestión, si es el caso.

ciclo	prox_CP	BUS	DL	DL	ALU	M	FMTL	ES	RD	RS
número	valor en hexadecimal	salida de MI en hexadecimal	instrucción en LE	acrónimos					detección de riesgo de datos	detección de riesgo de secuenciamiento

Figura 5.41 *Información textual en una fila de la ventana "Transcript" o en el fichero de salida.*

Para observar el progreso de una instrucción por las etapas es necesario analizar 7 ciclos consecutivos, si no se producen bloqueos (Figura 5.42). Dado un ciclo, la información en la columna prox_CP es la dirección en la entrada del registro CP. La información en la columna BUS es la instrucción cuya dirección es la salida del registro CP.

Cuando la instrucción ocupa la etapa DL se observa la actuación en el caso de un riesgo de datos. Cuando la instrucción es de secuenciamiento la actuación se observa en la etapa DL y en la siguiente.

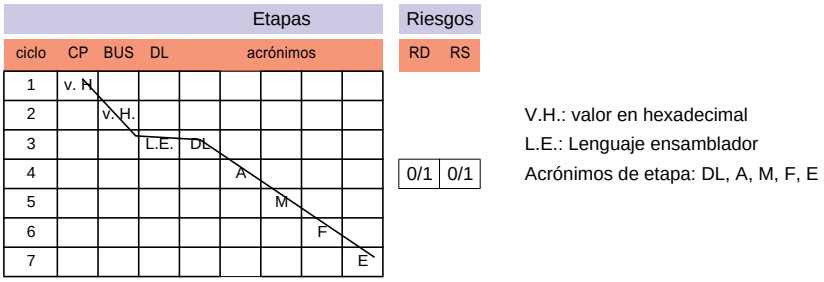


Figura 5.42 *Analísis del progreso de una instrucción por las etapas.*

Representación de las instrucciones. Los acrónimos utilizados para representar una instrucción en las etapas DL, ALU, M, FMTL y ES se muestran en la Figura 5.43. En las instrucciones de secuenciamiento se distingue si se actualiza el banco de registro (sí BR) o no se actualiza (no BR).

Instrucción	Etapas					Comentario
Cálculo	DL	A	R	R	E	R indica retardo - indica instrucción finalizada
Load	DL	A	M	F	E	
Store	DL	A	M	-	-	
secuenciamiento, no BR	DL	A	-	-	-	
secuenciamiento, sí BR	DL	A	R	R	E	

Figura 5.43 *Acrónimos utilizadas en las etapas al progresar una instrucción.*

Representación de un bloqueo en la etapa DL. Se utilizan el acrónimo “DLB” en la etapa DL mientras perdura el bloqueo (Figura 5.44). En las etapas posteriores se observa la propagación de una nop. El acrónimo utilizado es “nop”.

Instrucción	Ciclos				
Cualquiera	DLB	nop	nop	nop	nop
		DLB	nop	nop	nop
			DL	A	M

Figura 5.44 *Acrónimos utilizados en una instrucción nop inyectada desde DL.*

Representación de la inyección de una instrucción nop desde la etapa búsqueda. Al acrónimo de la etapa se le añade el sufijo sufijo N (Figura 5.45).

Instrucción	Etapas				
nop inyectada	DLN	AN	RN	RN	EN

Figura 5.45 *Acrónimos utilizados en una instrucción nop inyectada desde BUS.*

Para observar el progreso de una instrucción, tanto en la representación textual como en la ventana temporal de Modelsim, hay que efectuar un análisis en diagonal. La información que se muestra en una fila en la representación textual (de izquierda a derecha), excepto la información de riesgos, se muestra en una columna en la ventana temporal de Modelsim (de abajo hacia arriba). En la Figura 5.46 se muestra una secuencia de instrucciones. La instrucción “add x14, ..” detecta un riesgo de datos durante 3 ciclos. Desde la etapa DL se inyectan instrucciones nop (DLB).

Ciclo	prox_CP	BUS	DL (L.E:)	DL	A	M	FMTL	E	RD	RS
29	0000011C	00D70733	addi x15, x15, x004	| DL	|A	| R	| R	| nop	RD: 0	RS: 0
30	00000120	FEC79AE3	add x14, x14, x13	| DLB	| A	| M	| R	| E	RD: 1	RS: 0
31	00000120	FEC79AE3	add x14, x14, x13	| DLB	| nop	| R	| F	| E	RD: 1	RS: 0
32	00000120	FEC79AE3	add x14, x14, x13	| DLB	| nop	| nop	| R	| E	RD: 1	RS: 0
33	00000120	FEC79AE3	add x14, x14, x13	| DL	| nop	| nop	| nop	| E	RD: 0	RS: 0
34	00000124	00000013	bne x15, x12, xFFA	| DL	| A	| nop	| nop	| nop	RD: 0	RS: 1
35	00000118	00000013	addi x0 , x0 , x000	| DLN	|A	| R	| nop	| nop	RD: 0	RS: 1
36	00000114	0007A683	addi x0 , x0 , x000	| DLN	| AN	| -	| R	| nop	RD: 0	RS: 0
37	00000118	00478793	lw x13, x000(x15)	| DL	| AN	| RN	| -	| E	RD: 0	RS: 0
38	0000011C	FEC79AE3	addi xx15, x15, x004	| DL	| A	| RN	| RN	|-	RD: 0	RS: 0

Figura 5.46 *Información textual.*

La instrucción “bne” determina un riesgo de secuenciamiento que debe gestionarse durante dos ciclos. Desde la etapa BUS se inyectan instrucciones nop (DLN). En el ciclo 35 la instrucción ocupa la etapa ALU. En el siguiente ciclo la salida del registro CP es el

valor determinado por la instrucción de secuenciamiento. La línea horizontal finalizada en flecha indica que se establece el secuenciamiento determinado por la instrucción de secuenciamiento (bucle hardware BHS2).

Durante los ciclos 30 al 32 se inyectan instrucciones nop desde la etapa DL hacia la etapa ALU, ya que se detecta un riesgo de datos debido al registro x13 (la instrucción fuente de la dependencia no se observa). Durante los ciclos 35 y 36 se inyectan instrucciones nop desde la etapa BUS, ya que existe un riesgo de secuenciamiento. El valor del registro CP, que establece la instrucción de secuenciamiento, se observa en el ciclo 35 en la columna etiquetada como prox_CP.

Apéndice 5.1: Organización de la descripción en VHDL del procesador segmentado

Dada la entrada de un elemento de lógica, la cual está conectada a la salida de un registro de desacoplo, se construye un módulo que incluye todos los registros de desacoplo sin lógica combinacional interpuesta que transportan la señal.

Como ejemplo utilizaremos el elemento mES. Este multiplexor se construye utilizando el componente mux3. El fichero que describe el nuevo elemento, denominado RmES, se ubica, en esta práctica, en el directorio análogo al que está ubicado el fichero que describe el elemento mux3 en la práctica previa. En la Figura 5.47 se muestran los registros asociados a las entradas del multiplexor (registros con perímetro negro y con fondo con rayas en diagonal),

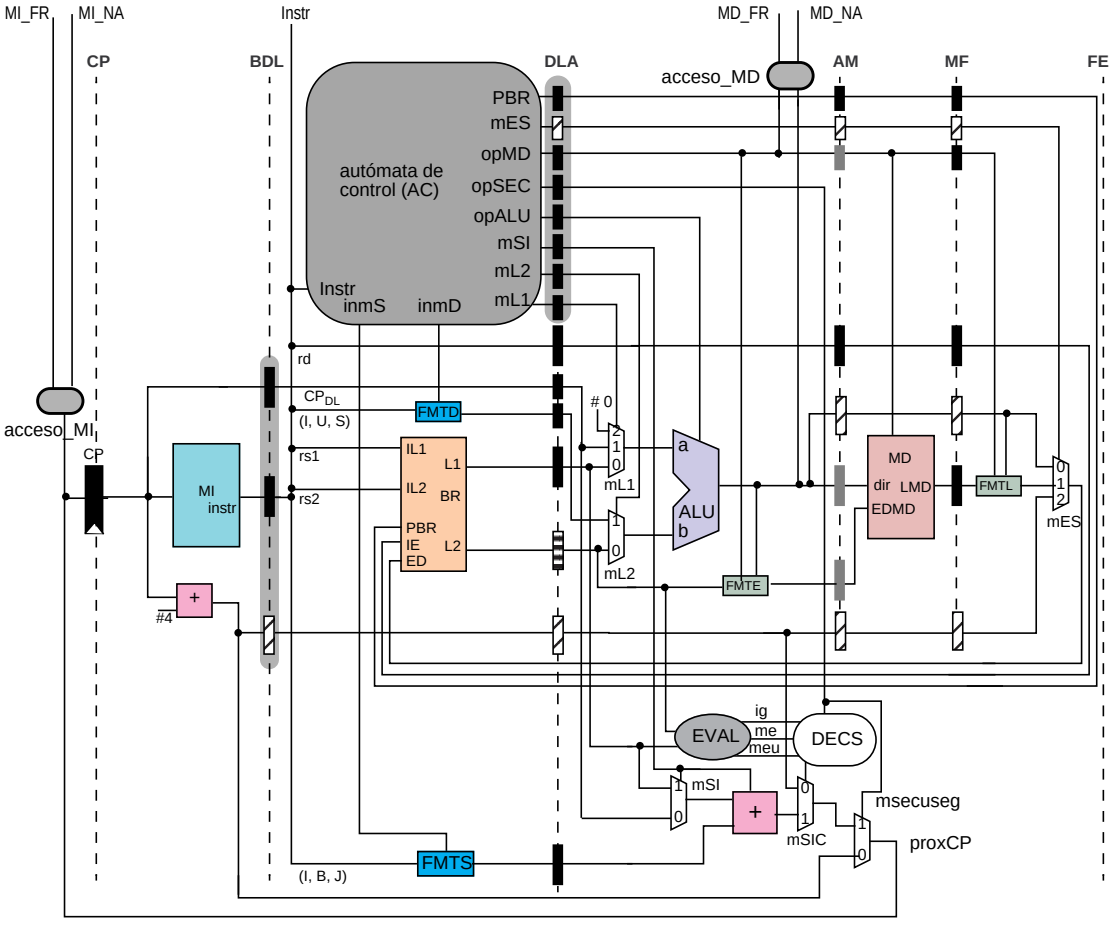


Figura 5.47 *Camino de datos del procesador segmentado. Identificación de registros asociados a compoenentes y réplica de los mismos.*

Seguidamente se muestra el código que describe al elemento RmES (Figura 5.48). En este código se puede distinguir: a) la secuencia de registros de desacoplo que transporta la señal mES, generada en la etapa DL, hasta la etapa ES, b) la secuencia de registros de desacoplo que transporta la salida de la ALU (d0) hasta la etapa ES y c) la secuencia de registros de descoplo que transporta la dirección de retorno (CP + 4, d2) desde la etapa DL hasta la etapa ES. La etiqueta utilizada al instanciar el registro utiliza como prefijo R seguido del nombre de la etapa, si es el caso, y el ciclo contando desde el inicio.

```
RA4_mES: RDI_N generic map(tam => num_mES)
        port map (reloj => reloj, pcero => pcero, I => I, e => SEL, s => SEL_A4);
RM5_mES: RD_N generic map(tam => num_mES)
        port map (reloj => reloj, pcero => pcero, e => SEL_A4, s => SEL_M5);
R6_mES: RD_N generic map(tam => num_mES)
        port map (reloj => reloj, pcero => pcero, e => SEL_M5, s => SEL_6);

RM5_alu: RD_D port map (reloj => reloj, e => d0, s => d0_M5);
R6_alu: RD_D port map (reloj => reloj, e => d0_M5, s => d0_6);

RDL3_CPret: RDB_N port map (reloj => reloj, pcero => pcero, B => B, e => d2, s => d2_DL3);
RA4_CPret: RD_D port map (reloj => reloj, e => d2_DL3, s => d2_A4);
RM5_CPret: RD_D port map (reloj => reloj, e => d2_A4, s => d2_M5);
R6_CPret: RD_D port map (reloj => reloj, e => d2_M5, s => d2_6);

muxRmES: mux3 generic map (sel1 => mES_ALU, sel2 => mES_MEM, sel3 => mES_RET)
            port map(d0 => d0_6, d1 => d1, d2 => d2_6, SEL => SEL_6, s => s);
```

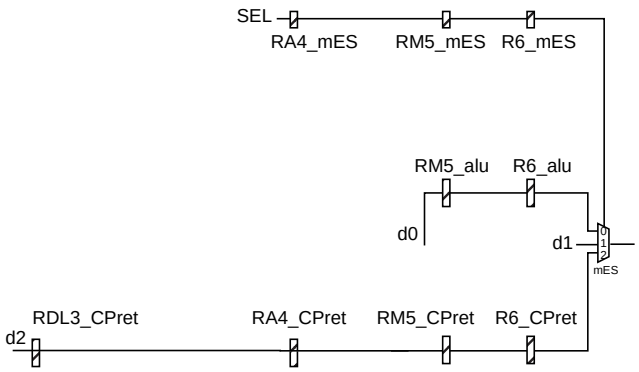


Figura 5.48 *Descripción del elemento RmES.*

Esta decisión da lugar a replicar un mismo registro de desacoplo en todos los elementos de lógica que utilizan la misma señal transmitida. Por ejemplo, el registro de desacoplo de una señal de una entrada de la lógica FMTE, una entrada de la lógica EVAL y una entrada del multiplexor mL2 es el mismo (registros con perímetro negro y con fondo con rayas horizontales en la Figura 5.47). En consecuencia, el registro de desacoplo se replica en la descripción de los tres componentes de lógica.

Los ficheros que incluyen registros de desacoplo tienen el mismo nombre que en el procesador serie con el prefijo R. En la estructura de directorios del procesador segmentado, en el directorio mimético al utilizado en la descipción del procesador serie se ubica el fichero con prefijo R.

Memoria de datos

Respecto a la memoria de datos (MD), los registros, con punteado en gris de la Figura 5.47, están incluidos en la descripción de la memoria (Figura 5.49). Recordemos que un acceso a memoria es síncrono con el flanco ascendente de la señal de reloj[23]. Este hecho da lugar a que la señal de entrada opMD debe ser RA4_opMD en lugar de RM5_ opMD.

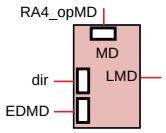


Figura 5.49 *Registros en la descripción de la memoria de datos.*

Memoria de instrucciones

Respecto a la memoria de instrucciones (MI), ésta tiene incluida en su descripción un registro en la entrada (de forma similar a la memoria MD). Este hecho determina que deba añadirse lógica para establecer el valor cero en este registro, cuando se inicializa el procesador (Figura 5.50).

Por otro lado, como se ha comentado previamente, se añade lógica adicional para actuar en una acción de bloqueo de la etapa BUS, cuando se detecta un riesgo de datos. En la Figura 5.50 también se muestra el multiplexor utilizado para inyectar una instrucción nop canónica desde la etapa BUS.

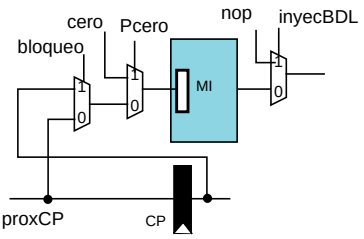


Figura 5.50 *Registro en la descripción de la memoria de instrucciones y lógica asociada.*

23. De forma similar a la descripción del banco de registros, el cual también se actualiza en el flanco ascendente de la señal de reloj.

Registros de desacoplo

En la Figura 5.47 los registros de entrada de la etapa DL, agrupados con un tramado de fondo, tienen la funcionalidad de bloqueo. Además, el registro en la salida de MI tiene la funcionalidad de modo burbuja o inyección de una operación nop (Figura 5.47). Ahora bien, esta funcionalidad, junto con la de bloqueo, está implementada junto con la memoria MI (Figura 5.50).

Por otro lado, cuando se activa la señal Pcero, el registro de entrada del decodificador establece una instrucción nop canónica en su salida[24]. Finalmente, el registro CP utiliza la funcionalidad de bloqueo.

En la tabla de la Figura 5.51 se indican las características de los registros de desacoplo utilizados en la descripción VHDL del procesador segmentado.

Registro	Tamaño datos	Funcionalidades	Registro	Tamaño datos	Funcionalidades
RD_1	1	puesta a cero	RD_D	N	transmisión en el flanco ascendente
RDI_1	1	puesta a cero e inyección modo burbuja	RDB_N	N	puesta a cero, modo bloqueo
RDI_N	N	puesta a cero e inyección modo burbuja	RDB_DL_N	N	puesta a cero (nop canónica), modo bloqueo
RD_N	N	puesta a cero			

Figura 5.51 *Tamaño de los datos y funcionalidades de los registros de desacoplo.*

24. En el apartado “Módulo LIB en la página 371” también se efectúa una descipción.

Apéndice 5.2: Organización de los ficheros: árbol de directorios

El árbol de directorios, desde el directorio raíz, de la práctica es mimético al árbol de la práctica del procesador serie (LAB4). Este árbol incluye un subdirectorio que contiene el control del procesador segmentado (CONTROL_SEGMENTACION) y un subdirectorio que contiene la descripción de los registros (REGISTRO_DESACOPLO). En los subdirectorios miméticos a los de LAB4, solo se incluyen los directorios que contienen ficheros modificados respecto al diseño serie[25].

En el directorio PROC_SEGMENTADO /CONTROL_SEGMENTACION están incluidos todos los ficheros que se utilizan en la lógica de interbloqueos (LIB, Figura 5.52). En el directorio REGISTRO_DESACOPLO están ubicados los ficheros que contienen las descripciones de los registros de descoplo.

Directorio
LAB5/PROC_SEGMENTADO
CONTROL_SEGMENTACION
COMPONENTES
COMPARADOR
DECO_PBR_opSEC
ENSAM_RD
ENSAM_RS
LDD
LDRD
LDRS
PUERTAS
SENYALES_CNTL
VALIDACION
componentes_cntl_seg_pkg
LIB
REGISTRO_DESACOPLO
CODIGO
RegDes_pkg

Figura 5.52 *Arbol de directorios del control de la segmentación y registros de desacoplo.*

En la Figura 5.53 se muestra el contenido del directorio “tipos_constantes_pkg”, el cual contiene un fichero con declaraciones de tipos y constantes adicionales utilizadas en el diseño y en la simulación del mismo.

Directorio	Fichero	descripción
tipos_constantes_pkg		
	cte_tipos_deco_camino_pkg.vhd	ESTE FICHERO SUSTITUYE AL FICHERO CON EL MISMO NOMBRE DE LA PRACTICA 4. Subtipos y constantes utilizadas en el control del camino de datos y de las unidades funcionales (U.F.)
	retardos_*.vhd	Especificación del retardos de componentes del camino de datos

Figura 5.53 *Ficheros ubicados en el directorio tipos_constantes_pkg.*

25. Existe una excepción en el directorio tipos_constantes_pkg Figura 5.53

En la Figura 5.54 se muestra la ubicación de “packages” que se utilizan en el programa de prueba para mostrar la evolución de la simulación. El directorio que se muestra sustituye al mismo directorio de la práctica 4.

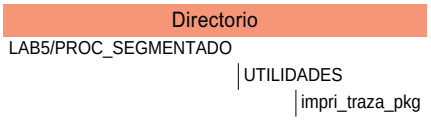


Figura 5.54 *Directorio que contiene “packages” utilizados en el programa de prueba.*

Edición de ficheros. En los directorios LDD, LDRD, LDRS y LGR se incluyen los ficheros que deben utilizarse para diseñar parte de la lógica LIB (Figura 5.26). Los ficheros contienen la declaración de la interface[26], la cual se corresponde con la especificada en el fichero componentes_control_seg_pkg.vhd, ubicado en el directorio componentes_cntl_seg_pkg. La instanciación de estos elementos se efectúa en los ficheros incluidos en los directorios ENSAM_RD y ENSAM_RS.

26. Ninguna de las interfaces debe modificarse.

Apéndice 5.3: Retardos

En el “package retardos_cntl_seg” se declaran los retardos de los elementos de la lógica de interbloqueos del procesador segmentado (Figura 5.55).

```
LDRD    constant retLDRD: time := 1 ns;
LDD     constant retLDD: time := 2 ns;
LDRS    constant retLDRS: time := 1 ns;
LGR     constant retLGR: time := 1ns;
```

Figura 5.55 *Retardos de los elementos de la lógica de interbloqueos.*

Los módulos auxiliares (decoPBRopSEC, valreg, ...) que se utilizan para recrear localmente señales del camino de datos tienen un retardo nulo.

Apéndice 5.4: Documentación

La documentación ha sido generada utilizando la herramienta Doxygen. El fichero que hay que abrir con un navegador es “index.html” ubicado en el directorio LAB5/PROC_SEGMENTADO/documentacio/hmtl.

Las pestañas que se muestran son autoexplicativas.

Dado un módulo se muestra un grafo de dependencias jerárquicas con otros módulos. Pulsando en un nodo del grafo se observan el módulo o módulos que agrupa.

También se puede acceder al código VHDL que describe a un módulo, excepto los módulos que se deben diseñar.

Apéndice 5.5: Implementación de la gestión de riesgos

En el procesador es necesario en primer lugar detectar una dependencia entre instrucciones que se ejecutan concurrentemente o detectar una instrucción de secuenciamiento. Posteriormente hay que actuar. La actuación puede ser: bloquear o suspender la interpretación de instrucciones (riesgo de datos) o descartar instrucciones (riesgo de secuenciamiento).

En el diseño que se utiliza, todas las detecciones de dependencias que afectan a una instrucción se efectúan cuando la instrucción ocupa la etapa DL. Así mismo, la detección de una instrucción de secuenciamiento se efectúa en la etapa DL.

Notemos que la actuación puede prolongarse durante varios ciclos, como en el caso de un riesgo de secuenciamiento. En el caso de riesgos de datos la detección y actuación se efectúan en el mismo ciclo.

En estas condiciones, los módulos LDD, LDRD y RdecoPBRopSEC detectan riesgos y por tanto, sus salidas son de interés en la etapa DL.

El módulo LDRS junto con el módulo seny_cntlS, una vez detectado un riesgo de secuenciamiento, actúa cuando la instrucción está en las etapas DL y ALU (2 ciclos).

El módulo LGR actúa en función del riesgo detectado.

Por otro lado, cuando se detectan de forma concurrente un riesgo de datos y un riesgo de secuenciamiento, es necesario gestionar en primer lugar el riesgo de datos.

Caracterización y enumeración de los riesgos de datos

En primer lugar clasificaremos las instrucciones en grupos, teniendo en cuenta el formato (FI, Apéndice 4.3). Por otro lado, tendremos en cuenta los registros que se leen y actualizan.

- R: Operandos y resultado en registros.
- I: Un operando se especifica en la propia instrucción. El otro operando y el resultado, si es el caso, en registros.
- S: Dos operandos en registros. El resultado se almacena en memoria.
- B: Los dos operandos en registros. La dirección de la siguiente instrucción se calcula de forma relativa a la dirección de la instrucción.
- J: La dirección de la siguiente instrucción se calcula de forma relativa a la dirección de la instrucción. Almacena la dirección de la instrucción mas cuatro en un registro

especificado en la instrucción.

- U: El operando está especificado en la propia instrucción (literal). Es posible que utilice también como operando la dirección de la instrucción. El resultado se almacena en un registro.

Para la caracterización utilizaremos 3 dimensiones ortogonales: a) registros que se actualizan, registros que se leen y distancia entre instrucciones interpretadas. Para especificar el tipo de instrucción utilizaremos el acrónimo del formato de instrucción, seguido de un punto y el sufijo del registro fuente o destino, según el caso.

- Registro que se actualiza.

Registro destino			
U.rd	J.rd	I.rd	R.rd

- Registros que se leen.

Registros fuente						
B.rs1	B.rs2	I.rs1	R.rs1	R.rs2	S.rs1	S.rs2

- Distancia entre instrucciones: 1, 2, 3 y 4.

Posibles riesgos de datos debidos a registros.

La latencia de actualización de un registro del banco de registros es 5 ciclos.

- Distancia 1. En las casillas de la tabla se muestran los ciclos perdidos.

distancia = 1	Registros fuente (consumidora)							
Reg. destino	I_jalr.rs1	B.rs1	B.rs2	I.rs1	R.rs1	R.rs2	S.rs1	S.rs2
U.rd	4	4	4	4	4	4	4	4
J.rd	4	4	4	4	4	4	4	4
I.rd	4	4	4	4	4	4	4	4
R.rd	4	4	4	4	4	4	4	4

- Distancias 2, 3 y 4. En las casillas de la tabla se muestran los ciclos perdidos (d2 / d3 / d4).

distancia = 2, 3, 4	Registros fuente (consumidora)						
Reg. destino	B.rs1	B.rs2	I.rs1	R.rs1	R.rs2	S.rs1	S.rs2
U.rd	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1
J.rd	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1
I.rd	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1
R.rd	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1	3 / 2 / 1

Características específicas del lenguaje máquina RISC V que deben tenerse en cuenta.

- El registro x0 está cableado a cero. Siempre se lee el valor cero. En consecuencia no determina dependencia de datos. Por tanto, siempre que un registro fuente sea el registro x0, el valor que se utiliza es el que se lee del banco de registros (valor cero).