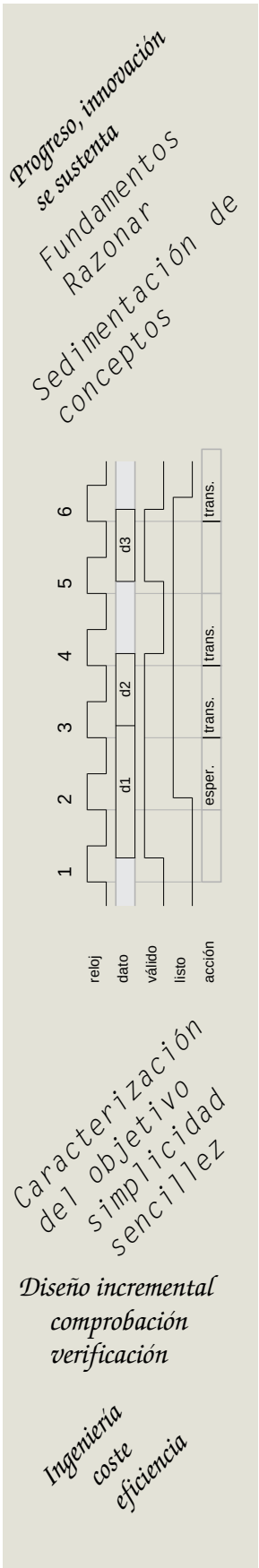
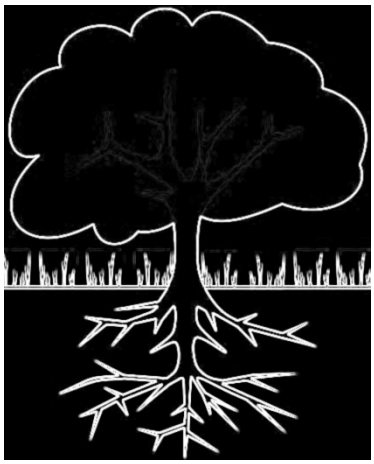


Documentación de prácticas Multiprocesadores (Grau - MP)



J.M. Llabería y
A. Olivé

Contenido

PRACTICA 2

PRACTICA 2

1 CACHE CON ESCRITURA INMEDIATA

En esta práctica se diseñará una cache bloqueante, con escritura inmediata y sin asignación de contenedor en el caso de fallo en escritura.

En la Figura 1 se muestran las interfaces de la cache con el procesador y la memoria. La funcionalidad de las señales se describe en la tabla de la Figura 2. En el procesador se distingue la parte productora (P) y la parte consumidora (C)[1]. En el esquema, el bus se muestra dos veces. El objetivo es que las señales, con alguna excepción, fluyan de izquierda a derecha. Las señales de las interfaces, que se conectan con el bus se nombran en éste con el prefijo “bus_”.

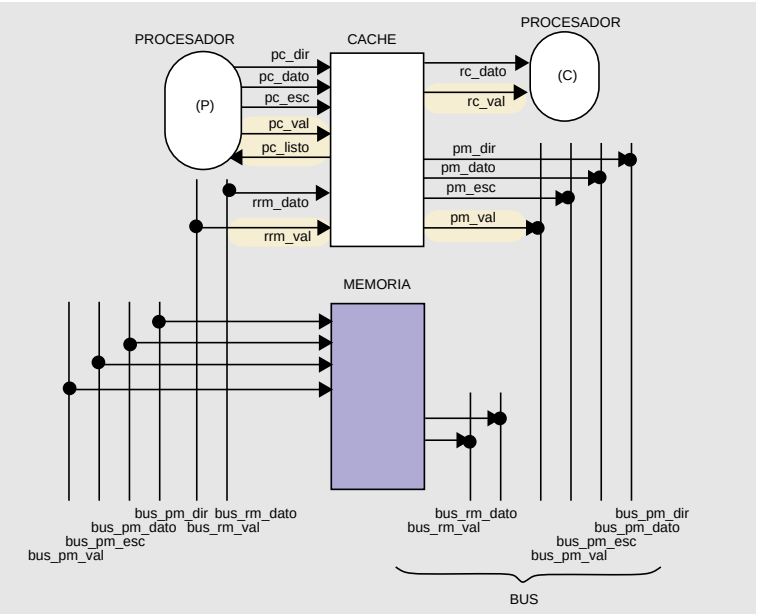


Figura 1 Interfaces entre procesador, cache y memoria.

1. En el diseño de la cache se supone que el productor es una etapa previa en el diseño de un procesador. Por ejemplo, la etapa ALU. El consumidor es la etapa posterior al acceso a cache.

Control de flujo. El control de flujo entre el procesador (P) y la cache, utilizada un protocolo listo/válido (pc_valido/pc_val). El control de flujo, entre la cache y el procesador (C), utiliza un protocolo degenerado (rc_val), más simple, ya que suponemos que el procesador siempre está dispuesto a aceptar el dato suministrado por la cache.

En cuanto al control de flujo entre la cache y memoria (pm_val) y viceversa (rm_val), suponemos que tanto la cache como la memoria siempre están dispuestas a aceptar la petición o el dato.

		Interfaces procesador / cache		**Interfaces cache / (bus) memoria**	
			Descripción de las señales		**Descripción de las señales**
petición	información	pc_dir	dirección: dirección generada por el procesador	pm_dir	dirección: dirección de acceso a memoria
		pc_dato	dato: dato en una instrucción store	pm_dato	dato: dato en un acceso de escritura
		pc_esc	escritura: indicación de escritura	pm_esc	escritura: indicación de escritura
	pr.	pc_val	válido: indicación de acceso	pm_val	válido: indicación de acceso
		pc_listo	listo: cache desocupada (o libre)		
respu.	inf.	rc_dato	dato: respuesta de la cache en una instrucción load	rm_dato	dato: respuesta de la memoria en una petición de bloque
	pr.	rc_val	válido: indicación de respuesta	rm_val	válido: indicación de respuesta

Figura 2 Descripción de las señales de las interfaces entre procesador, cache y memoria. Las abreviaturas pr., inf., respu. indican respectivamente protocolo, información y respuesta.

1.1 Organización de la cache

Como es usual, una dirección de memoria referencia una posición que almacena un byte[2]. Para simplificar el diseño, todos los accesos a memoria generados por el procesador son a byte.

La cache que se diseña es de mapeo directo y tiene 8 contenedores. En un contenedor se distinguen los campos: etiqueta, estado y bloque de datos (Figura 3). Sólo los bloques almacenados en cache tienen un hardware que representa su estado.

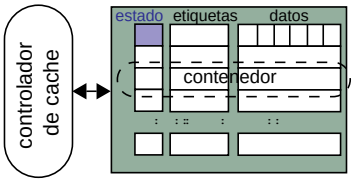


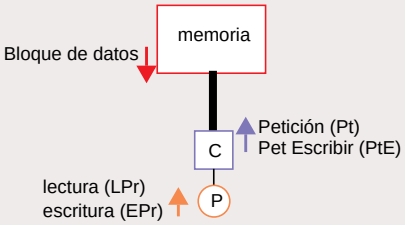
Figura 3 Memoria cache y campos en un contenedor de cache.

Para centrarnos en las acciones, que debe realizar el controlador de cache cuando recibe peticiones del procesador, supondremos que el tamaño de bloque es igual a la granularidad de acceso del procesador, en este caso 1 bytes [3]. El controlador de cache actualiza el estado de los bloques en respuesta a eventos del procesador y genera accesos a memoria.

2. En un procesador, la granularidad de acceso está determinada por el tipo de acceso (codificación), especificado en la instrucción de acceso a memoria (palabra, media palabra, byte).
3. En el diseño no se explota en ningún caso esta simplificación.

1.2 Eventos del procesador y peticiones de acceso

En la tabla de la Figura 4 se describen los eventos del procesador y las acciones del controlador de cache.



Procesador	Controlador de cache
Peticiones	**Acciones**
LPr : lectura del procesador	Pt : petición de bloque
EPr : escritura del procesador	PtE : actualización de memoria
	CcRe: reemplazo de un bloque

Figura 4 Eventos del procesador y acciones del controlador de cache.

1.3 Diagrama de estados y transiciones de un bloque en cache

En la Figura 5 se muestran los dos estados de un bloque en cache y las transiciones entre ellos. El estado de un bloque sólo es explícito cuando está almacenado en cache.

El estado inválido (I) indica que el contenedor no almacena un bloque válido. Un bloque que no está almacenado en un contenedor de cache se dice que no está presente (fallo de cache) y por tanto está en estado inválido.

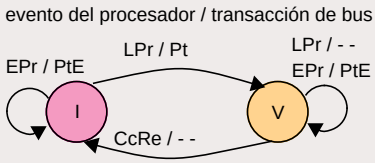


Figura 5 Diagrama de estados y transiciones de un bloque en un contenedor de cache.

El estado válido (V) indica que el contenedor almacena un bloque válido. Una operación de reemplazo, del controlador de cache, selecciona un bloque y habilita el contenedor que lo almacena para otro uso (el bloque se invalida y se indica en el contenedor). En el caso que nos ocupa, escritura inmediata, esta operación no tiene asociada ninguna acción, ya que la memoria siempre está actualizada.

Si en una operación de lectura (LPr) se produce acierto en cache se suministra el dato. En caso contrario, se genera una petición de lectura del bloque, en el cual está ubicado el dato accedido, a memoria (Pt).

En una operación de escritura se genera una petición para actualizar memoria (PtE) y si es acierto, se actualiza el campo de datos en cache. La granularidad de una actualización de memoria, en este diseño, es un byte[4].

4. Notemos la diferencia en la granularidad de los accesos a memoria entre un fallo de lectura (bloque) y una escritura (granularidad determinada por la instrucción). Para simplificar, en esta práctica se utiliza la misma granularidad.

2 CAMINO DE DATOS Y CONTROLADOR DE CACHE

En la Figura 6 se muestran las interfaces del camino de datos de la cache, el controlador de cache, el procesador y la memoria[5]. Las señales entre el controlador de cache y el módulo campos se describen posteriormente

Asociado al camino de datos se distingue un multiplexor. Este multiplexor (muxE) se utiliza para actualizar el módulo campos en fallos de lectura y aciertos de escritura.

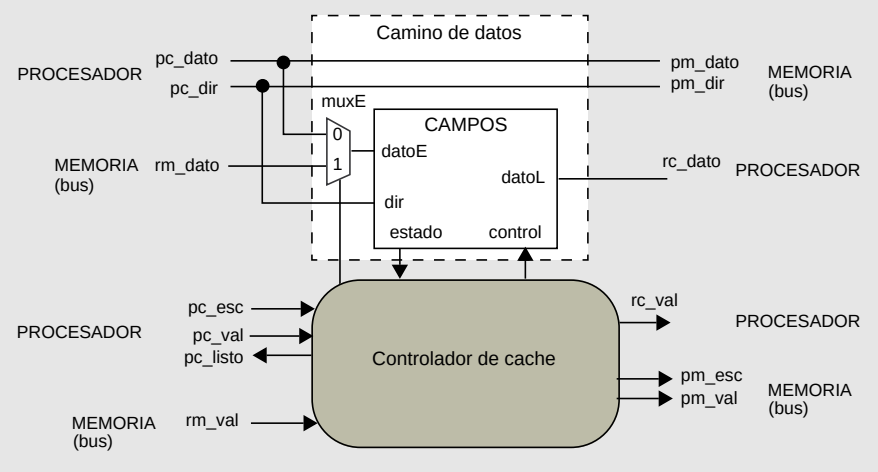


Figura 6 Memoria cache: módulo campos y controlador de cache.

2.1 Elementos en el módulo campos

En la Figura 7 se muestran los elementos del módulo campos. Se distinguen tres elementos de almacenamiento, que se utilizan para implementar los tres campos de los contenedores de cache.

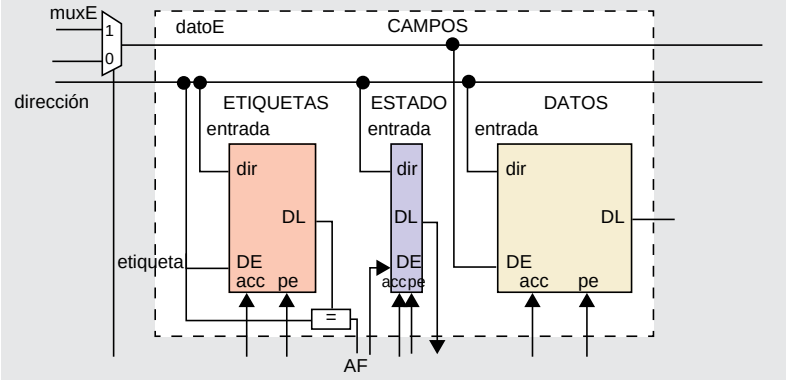


Figura 7 Elementos de almacenamiento y circuitos adicionales en el módulo campos.

Cada campo se implementa con un elemento de almacenamiento que tiene una señal entrada (dir) para indicar la entrada[6] accedida (campo índice, conjunto, de una dirección de memoria). Otra señal de

5. En la Figura 2 han sido descritas las señales.

entrada es el camino de acceso para escrituras (DE), cuyo número de bits depende del campo correspondiente de la cache. Las otras entradas (acc y pe) se utilizan para acceder al campo y la escritura al mismo, si es el caso.

Los parámetros de cada uno de los campos se indican en la tabla de la Figura 8. El número de conjuntos de la cache es 8.

Los elementos de memorización utilizados son síncronos. Esto es, tienen como entrada la señal de reloj. Tanto la lectura como la escritura están sincronizadas con el flanco ascendente de la señal de reloj[7].

Señales	Número de bits
dirección	16 bits
dir (entrada)	3 bits
DE y DL en Etiquetas	13 bits
DE y DL en Estado	1 bit
DE y DL en Datos	8 bits
acc y pe	1 bit

Figura 8 Tamaño de las entradas y salidas de los campos de la cache

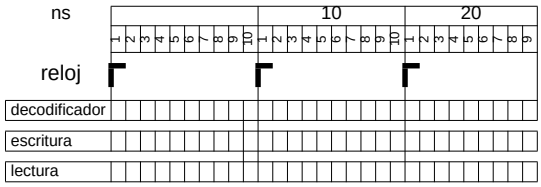
En estas condiciones, sólo hay un puerto de acceso a cada campo de la cache. Un campo con la señal acc activada se está leyendo y puede escribirse, si está activado el permiso de escritura correspondiente[8].

Trabajo 1: En el Apéndice 2.1 se describe un módulo básico de memoria y su funcionamiento. En el directorio “proyecto_1/ cache_con_interface_bus/cache_con_interface_proc/cache/ camino_datos/componentes/campo_datos/componentes/m_datos/ CODIGO” está ubicado un fichero con la especificación de un módulo de memoria, que se utiliza como campo de datos de la cache. El directorio CODIGO tiene asociados los directorios PRUEBAS, QUARTUS y RESULTADOS. Analice el fichero que contiene la especificación del módulo de memoria. Elabore el componente utilizando Quartus. Para ello, dispone de los ficheros *.qsf y *qpf en el directorio QUARTUS. Posteriormente active la simulación con Modelsim. Al activar Modelsim se utiliza como programa de prueba el código ubicado en el directorio PRUEBAS. Este programa tiene especificada una secuencia de accesos, utilizando procedimientos declarados en el mismo fichero. Modifique la secuencia de accesos para mejorar la comprensión del funcionamiento del módulo de memoria.

Trabajo 2: Muestre en sendos diagramas temporales de retardos un acceso de lectura y un acceso de escritura. Los retardos deben indicarse en el instante de tiempo más tardío en que pueden producirse para un funcionamiento correcto. Indique los retardos que pueden afectar a un valor, tanto en una posición de almacenamiento como en un puerto de salida. Utilice una señal de reloj cuadrada con periodo de

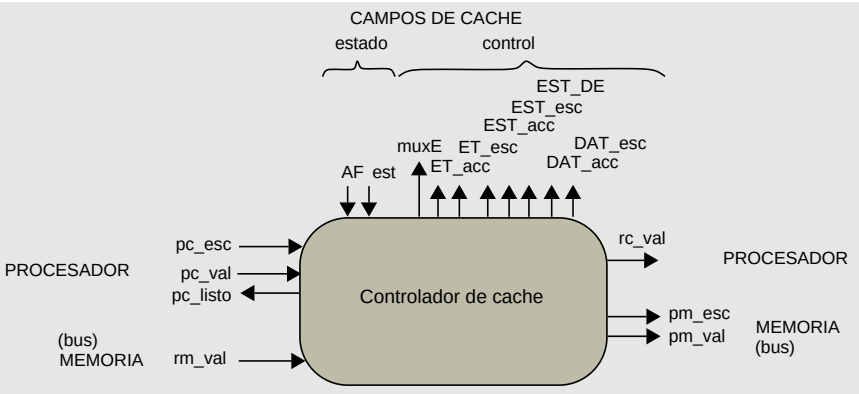
6. Al utilizar correspondencia o mapeo directo, conjunto es sinónimo de contenedor.
7. En el Apéndice 2.1 se detalla el funcionamiento lógico de un elemento básico de almacenamiento.
8. En el ciclo que se escribe, el valor que se lee no tiene porqué corresponderse con el valor escrito. El valor leído no está especificado.

10 ns como referente. Utilice los valores de retardo especificados en el Apéndice 2.16 (ret_deco_dat, ret_dat_leer, ret_dat_esc).



2.2 Controlador de cache

En la Figura 9 se muestran las interfaces entre el controlador de cache y los campos de la cache, además del procesador y la memoria. En el caso de los campos de cache se identifican las señales como de estado o de control.



Señales	Descripción
AF	acierto/fallo
est	estado del bloque en cache
muxE	multiplexor muxE
X_acc	acceso al elemento de almacenamiento
X_esc	actualización del elemento de almacenamiento
X	ET, EST, DAT
EST_DE	valor utilizado en la actualización

Figura 9 Interfaces del controlador de cache con la cache, el procesador y memoria.

Descripción textual del controlador de cache

Cuando la señal pc_val (petición, Figura 1) se activa, el controlador de cache utiliza la señal AF (Figura 9) para determinar si se ha producido acierto o fallo. Los derechos de acceso se determinan utilizando la señal AF y el valor leído del campo estado (est). La señal pc_esc indica si el acceso es de tipo lectura o escritura.

Si la instrucción es un load y se detecta acierto, el dato leído del campo de datos se suministra al procesador. El procesador (C, Figura 1) espera a la activación de la señal rc_val para capturar el dato.

El procesador (P, Figura 1) determina cuándo puede efectuar una nueva petición observando la activación de la señal pc_listo.

En el caso de una escritura o un fallo de lectura es necesario interactuar con memoria, lo cual se realiza mediante la señal de petición pm_val. El tipo de petición a memoria (lectura, escritura) se indica mediante la señal pm_esc.

La activación de la señal rm_val es una indicación de que ha sido servido el acceso a memoria. En una operación de lectura de memoria se actualiza la cache con el bloque, se suministra el dato al procesador y se activan las señales pc_listo y rc_val.

En una operación de escritura a memoria, siendo acierto en cache, se actualiza el campo de datos de cache al recibir la confirmación de memoria (rm_val). Tanto si ha sido acierto o fallo en escritura, el servicio ha finalizado y se activa la señal pc_listo. En la Figura 10 se presenta la descripción en seudocódigo.

```
while (true) {
   while (pc_val = NO) { };
   if (AF = acierto and pc_esc = lectura and estado = válido) then      acierto de lectura
      pc_listo = SI; rc_val = SI;
   else
      pm_val = peticion a memoria;
      pm_esc = pc_esc;
      while (rm_val = NO) { };
      if (pc_esc = lectura) then                                        fallo de lectura
         muxE = memoria;
         DAT_acc, DAT_esc = escribir campo datos;
         EST_acc, EST_esc = escribir campo estado;
         EST_DAT = válido;
         ET_acc, ET_esc = escribir campo etiquetas; rc_val = si;
      else if (AF = acierto and pc_esc = escritura) then                acierto de escritura
            muxE = procesador;
            DAT_acc, DAT_esc = escribir campo datos;
      end if
      pc_listo = si;
   end if
}
```

Figura 10 Descripción textual en seudocódigo de las acciones del controlador.

Fases en el acceso a cache

En el acceso a cache distinguimos varias fases. Las dos primeras fases son comunes a todos los tipos de acceso considerados: a) acceso a los campos etiquetas y estado (ET, EST), b) comparación de etiquetas y determinación de los derechos de acceso, utilizando la señal de estado del bloque (CMPET). En la Figura 11 se muestra la secuencia de fases para cada caso.

	Fases				
acceso / DA	a)	b)	c)	d)	e)
Load / si	ET, EST	CMPET	LEC		
Load / no	ET, EST	CMPET	MEM	ESB	LEC
Store / si	ET, EST	CMPET	MEM	ESP	
Store / no	ET, EST	CMPET	MEM		

Figura 11 Fases en una transacción de memoria. Derechos de acceso: DA (AF and est).

Las siguientes fases dependen del si es acierto o fallo y del tipo de acceso. En el caso de una lectura con derechos de acceso: c) se accede al campo datos (LEC). Si no se dispone de los derechos de acceso: c) se efectúa una petición de bloque a memoria (MEM). Posteriormente: d) se actualiza la cache con el bloque que se recibe (ESB). Finalmente: e) se accede al campo de datos para leer el dato (LEC).

En el caso de una escritura: c) se accede siempre a memoria (MEM). En particular, cuando se dispone de los derechos de acceso: d) se actualiza la cache (ESP).

Autómata del controlador de cache

En el controlador de cache se identifican un conjunto de estados que permiten secuenciar las fases. En la Figura 12 se relacionan los estados del controlador de cache.

ESTADOS	Descripción
DES0 y/ o DES	Estado inicial después de una puesta a cero (Figura 14 y Figura 16). DES estado de espera (Figura 16)
INI	Nada
ESCINI	Actualización de todos los campos, asociados al contenedor referenciado, con la información suministrada en la transacción.
HECHOE	Acceso de escritura finalizado
HECHOL	Acceso de lectura finalizado
CMPET	Lectura del campo etiqueta y campo estado. Comparación de etiquetas
LEC	Lectura del campo datos
PML	Inicio del acceso a memoria para leer un bloque
ESPL	Espera a que la memoria suministre el bloque
ESB	Actualización de todos los campos, del contenedor asociado, con el bloque leído de memoria
PMEA	Inicio del acceso a memoria para escribir un dato cuando la cache contiene el bloque
ESPEA	Espera a que la memoria confirme la escritura del dato cuando la cache contiene el bloque
ESCP	Actualización del campo de datos debido a una escritura que acierta en cache
PMEF	Inicio del acceso a memoria para escribir un dato cuando la cache no contiene el bloque
ESPEF	Espera a que la memoria confirme la escritura del dato cuando la cache no contiene el bloque

Figura 12 Estados del controlador de cache.

Asociación de fases a estados. En la Figura 13 se muestra la asociación de fases (Figura 11) con estados (Figura 12) en el autómata de control. En cada estado se está un ciclo, excepto en los estados ESPL, ESPEA y ESPEF, donde se espera el servicio de memoria. Otro estado donde se puede estar más de un ciclo, cuando no hay peticiones, es en DES. En particular las fases a) y b) se efectúan en el mismo estado o etapa; en un ciclo. Notemos que un acceso de lectura con acierto en cache requiere de 4 ciclos.

En el Apéndice 2.2 se muestra la utilización de los recursos del camino de datos en varios estados del controlador de cache.

acceso / DA	Fases y transiciones entre estados						
Load / si	DES	CMPET	LEC	HECHOL			
fases		ET, EST, CMPET	LEC				
Load / no	DES	CMPET	PML	ESPL	ESB	LEC	HECHOL
fases		ET, EST, CMPET	MEM	MEM	ESB	LEC	
Store / si	DES	CMPET	PMEA	ESPEA	ESCP	HECHOE	
fases		ET, EST, CMPET	MEM	MEM	ESP		
Store / no	DES	CMPET	PMEF	ESPEF	HECHOE		
fases		ET, EST, CMPET	MEM	MEM			

Figura 13 Transiciones entre estados en función del tipo de acceso y de los derechos de acceso. Los estados sombreados pueden requerir más de un ciclo. Derechos de acceso: DA (AF and est).

Grafo de transiciones entre estados. En la Figura 14 se muestra el grafo de transiciones entre estados del controlador de cache. Para cada uno de los casos enumerados en la Figura 13 se puede identificar un camino entre estados en el grafo de estados. Estos caminos entre estados son, en su mayor parte, disjuntos.

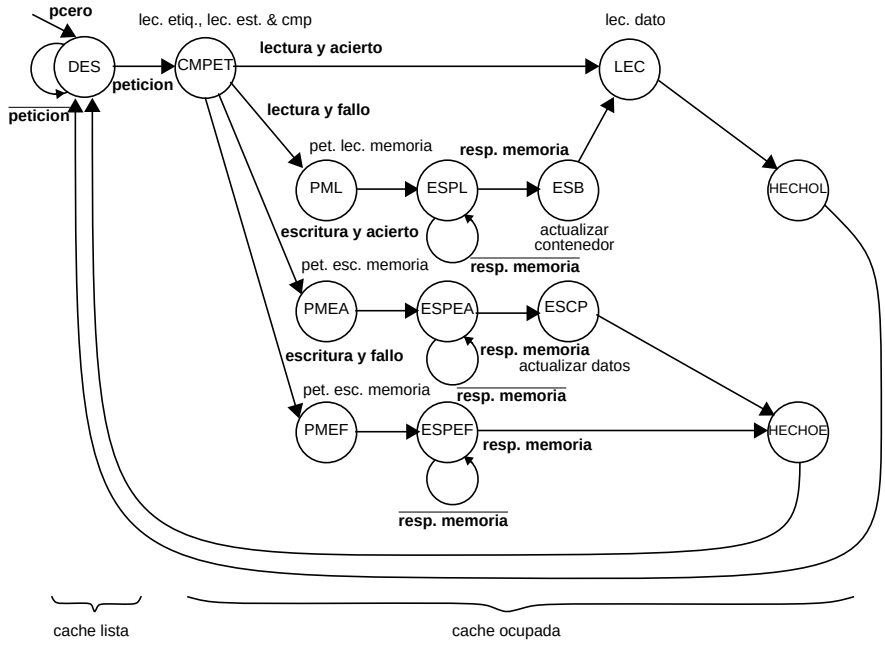


Figura 14 Grafo de transiciones entre estados del controlador de cache. Con letra negrita las acciones que determinan una transición entre estados. Con letra normal las acciones en el estado.

En el grafo de estados se utilizan los estados DES (cache desocupada o lista) y HECHOL y HECHOE (acceso finalizado) para identificar, en un estado concreto, la posibilidad de iniciar una nuevo acceso y la finalización de un acceso iniciado previamente. La cache está desocupada (lista) en el estado DES y ocupada en todos los otros estados.

En el Apéndice 2.3 se utiliza una tabla para describir las transiciones entre estados.

Transiciones entre estados				
acceso				
inicio	DES0	INI	ESCINI	HECHOE

Figura 15 Petición de inicio.

Petición de inicio. En el diseño previo añadimos un nuevo tipo de petición para comprobar el funcionamiento del camino de datos de la cache y de su control. Esta petición permite escribir en los distintos campos de un contenedor de cache, independientemente del estado (Figura 15). Una vez finalizada, el contenedor de cache contiene un bloque válido. Por tanto, posteriormente puede efectuarse una operación de lectura que acierta en cache.

Esta nueva petición, denominada inicio, sólo puede efectuarse después de inicializar (poner a cero) el controlador. En estas condiciones, el nuevo grafo de transiciones entre estados es el que se muestra en la Figura 16. Los estados añadidos se muestran con un trazo más grueso.

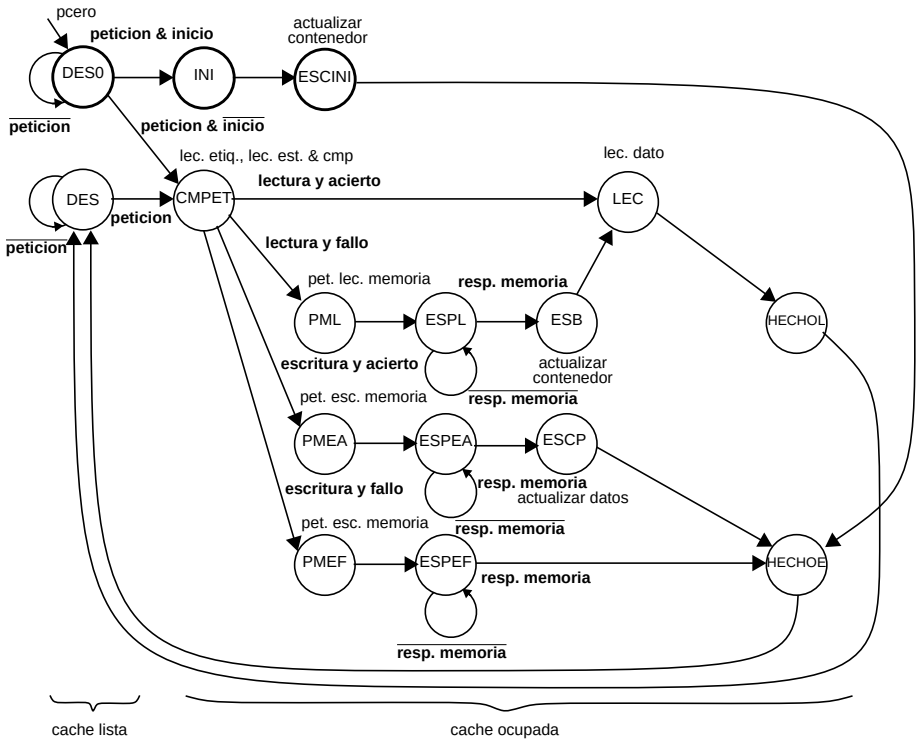


Figura 16 Diagrama de transiciones entre estados del controlador de cache que incluye las transiciones que gestionan la petición inicio. Con letra negrita las acciones que determinan una transición entre estados. Con letra normal las acciones en el estado.

2.3 Procesador-cache-memoria

En la Figura 17 se muestra el camino de datos que constituyen los tres elementos: a) procesador, cache y memoria. En particular se muestran los registros de desacoplo utilizados en el bus y el registro y multiplexor interpuestos entre el procesador y la cache (interface procesador/cache).

La cache es una unidad funcional multiciclo de latencia variable de procesado y con latencia de iniciación igual a la latencia de procesado.

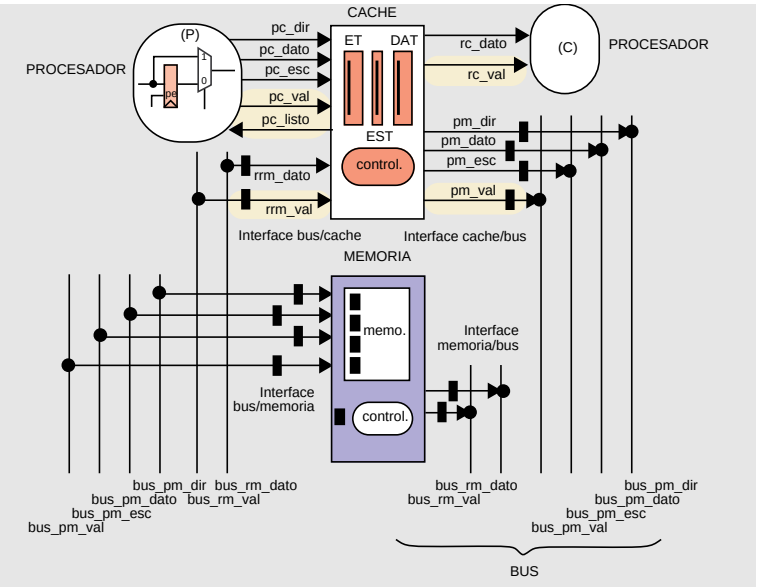


Figura 17 Camino de datos procesador-cache-memoria explicitando los registros de desacoplo (línea negra gruesa), registros internos a los módulos básicos de almacenamiento (Apéndice 2.1) e interface con el productor (procesador). No se muestra la señal "pc_ini".

En la Figura 18 se muestra en detalle la lógica de la interface procesador/cache. La señal pc_val, cuando pc_listo está activado, es observada directamente por el controlador, mediante el multiplexor (entrada 1).

La necesidad de servicio de una petición se determina en el estado DES y/o DES0 (pc_listo = cache_lista).

Por otro lado, recordemos que los módulos de memoria básicos disponen de registros internos (Apéndice 2.1). Esta característica es la que determina que la dirección, a la que se accede, también tiene que estar disponible antes de efectuar la transición del estado DES y/o DES0 al estado CMPET.

El diagrama temporal de la Figura 19 muestra las interfaces procesador/cache y cache/procesador. En la interface cache/procesador sólo se utiliza la señal rc_val. En la interface procesador/cache se utilizan las señales pc_val y pc_listo.

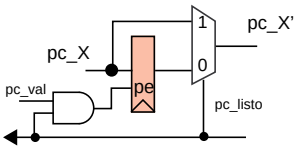
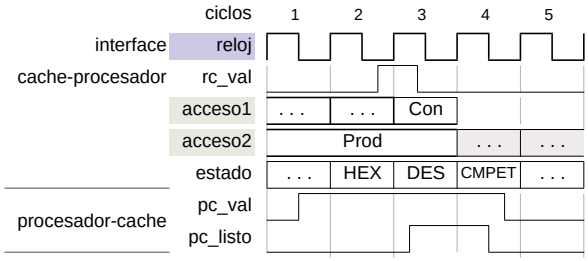


Figura 18 Interface procesador/ cache. pc_X y pc_X', todas las señales, incluyendo pc_val.



Señales	Descripción
Prod	productor
Con	consumidor
HEX	HECHOL, HECHOE

Figura 19 Protocolo en las interfaces de la cache con el procesador.

En el flanco ascendente del ciclo 3 la señal rc_val está activada y el consumidor captura el dato. En el flanco ascendente del ciclo 4 hay una petición pendiente (Prod, pc_val = 1) y la cache está lista (pc_listo = 1). Por tanto, se inicia el procesado de la petición y el productor puede iniciar la generación de una nueva petición.

En el ciclo 4 se accede a los campos ET y EST de la cache. Para ello, la dirección ha sido almacenada en el flanco ascendente de este ciclo en los registros internos de los módulos de almacenamiento correspondientes (Apéndice 2.1). Durante el ciclo 4 y posteriores la lógica de la cache se alimenta de la información almacenada en el registro de la interface (Figura 18), ya que la entrada seleccionada del multiplexor es la cero (pc_listo = 0).

2.4 Acceso a memoria

El acceso a memoria es multiciclo. Los rectángulos negros en la Figura 20 representan registros de desacoplo o registros internos al módulo básico de almacenamiento. En el ciclo posterior, a la detección de una escritura o un fallo de lectura, se transmite por el bus la información oportuna (bus_I). La memoria está ocupada durante dos ciclos[9]. En el ciclo posterior se ocupa el bus (bus_V) y en el siguiente ciclo se detecta, en el controlador de cache (CC)[10], la finalización del acceso a memoria.

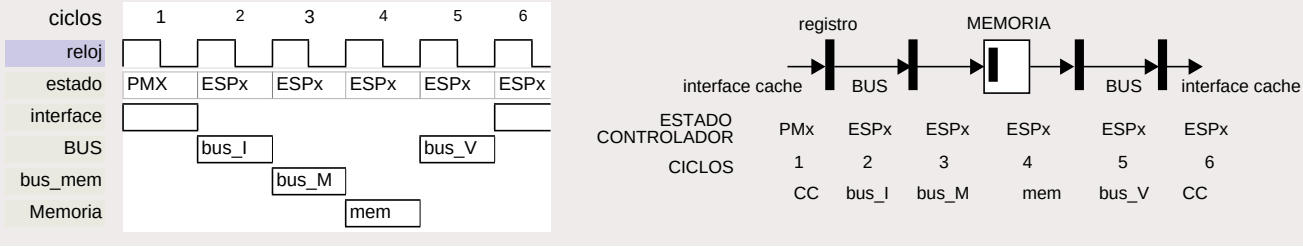


Figura 20 Detalle de una transacción en el bus.

9. En este diseño la memoria no es objeto de análisis. Por tanto, una latencia de 2 ciclos es suficiente (bus_M, mem en la Figura 20).
10. El CC se utiliza en todos los ciclos. En la figura se indica CC sólo cuando no se utiliza otro recurso de bus o memoria.

Ciclos para cada tipo de transacción. En la Figura 21 se muestran los ciclos para cada una de las transacciones. En la última fila se muestra la ocupación del bus (ida y vuelta), de la memoria y del CC al empezar y finalizar una transacción.

	ciclos										
acceso / DA	1	2	3	4	5	6	7	8	9	10	11
Load / si	DES	CMPET	LEC	HECHOL							
fases		ET, EST, CMPET	LEC								
Load / no	DES	CMPET	PML	ESPL	ESPL	ESPL	ESPL	ESPL	ESB	LEC	HECHOL
fases		ET, EST, CMPET	MEM						ESB	LEC	
Store / si	DES	CMPET	PMEA	ESPEA	ESPEA	ESPEA	ESPEA	ESPEA	ESCP	HECHOE	
fases		ET, EST, CMPET	MEM						ESP		
Store / no	DES	CMPET	PMEF	ESPEF	ESPEF	ESPEF	ESPEF	ESPEF	HECHOE		
fases		ET, EST, CMPET	MEM								
acceso a memoria			CC	bus_I	bus_M	mem	bus_V	CC			

Figura 21 Ciclos en función del tipo de transacción. Derechos de acceso: DA (AF and est).

Secuencia de accesos a memoria. En la Figura 22 se muestra una secuencia de dos accesos a memoria que son load (P1, P2). El primer acceso se sirve desde la cache (C1) y el segundo requiere un acceso a memoria (C2). En la primera fila se muestra la etapa previa a la cache (Productor). En la segunda fila se muestra al consumidor (etapa que sigue al acceso a cache). En la tercera fila se muestran, en cada ciclo, los acrónimos del acceso a cache en cada uno de los accesos[11]. En las siguientes filas se muestran los recursos de la jerarquía de memoria utilizados en cada ciclo[12] (Figura 23).

	ciclos															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Productor	P1			P2												
Consumidor					C1											C2
Acrónimos	DE	CM	LE	HL	DS	CM	FL	ML	ML	ML	ML	ML	EB	LE	HL	DS
Recursos																
CC	X			X	X		X					X			X	X
ET		X				X							X			
EST		X				X							X			
CMP		X				X										
DAT			X										X	X		
bus_I								X								
bus_M									X							
mem										X						
bus_V											X					

Figura 22 Diagrama temporal de una secuencia de dos accesos a la jerarquía de memoria.

11. Los acrónimos utilizados se corresponden con los mostrados en la Figura 57 del Apéndice 2.5. Se utilizan en la representación textual de una simulación con Modelsim.

El primer acceso es emitido por el productor en el ciclo 1 y el consumidor captura el dato en el ciclo 5. El segundo acceso es emitido por el productor en el ciclo 4 y la cache lo reconoce en el ciclo 5, una vez está lista. El consumidor captura el dato en el ciclo 16.

Recursos	Descripción
ET	Campo etiquetas
EST	Campo estado
DAT	Campo datos
CMP	Comparador de etiquetas
CC	Controlador de cache

Recursos	Descripción
bus_I	Bus de ida a memoria
bus_M	Conexión del bus a memoria
mem	Memoria
bus_V	Bus de vuelta

Figura 23 Recursos considerados en un acceso a la jerarquía de memoria.

3 PROYECTOS

Esta práctica incluye varios proyectos. Todos los proyectos se refieren al controlador de cache, con alguna modificación necesaria en el camino de datos.

El objetivo es diseñar un CC que permita que la latencia de acceso a cache, cuando el acceso es una lectura y se determina un acierto, sea de 1 ciclo.

El trabajo se plantea de forma incremental. En el primer proyecto el diseño del CC se centra en la funcionalidad. Esto es, el secuenciamiento y temporalidad de las acciones necesarias en función del tipo de acceso, sin expectativas de rendimiento.

Los siguientes proyectos tienen como objetivo mejorar el rendimiento. Para ello, se eliminan estados en el CC con el objetivo de reducir la latencia del acceso a la jerarquía de memoria.

Los ficheros comunes a todos los proyectos tiene como raíz el directorio “proyecto_1”. Los otros proyectos tienen la misma estructura de directorios, pero sólo se incluyen los directorios y ficheros que son específicos del proyecto. Existen excepciones, como algunos “packages” que especifican procedimientos. Aunque los ficheros sean idénticos en varios proyectos, en cada proyecto se replican, con el objetivo de independizar los proyectos en las pruebas.

3.1 Proyecto 1: Diseño del controlador de cache

En el Apéndice 2.4 y Apéndice 2.5 se detalla la organización del proyecto en directorios y la ubicación de los ficheros y los pasos para efectuar la simulación (Quartus y Modelsim). En el Apéndice 2.6 se indica la ubicación de la documentación generada con Doxygen.

12. El CC se utiliza en todos los ciclos. Ahora bien, sólo se explicita su utilización cuando no se utilizan otros recursos.

En los trabajos donde se solicita efectuar un análisis es de utilidad utilizar la documentación generada por Doxygen.

Trabajo 3: Analice la especificación de los distintos elementos que se utilizan como campos de la cache (Apéndice 2.4, Figura 44).

Trabajo 4: Analice la especificación estructural del ensamblado de los componentes de la cache (Apéndice 2.4, Figura 44.

Trabajo 5: Analice la especificación del ensamblado de los componentes: campos de cache, mxE y controlador de cache (Apéndice 2.4, Figura 44).

Trabajo 6: Analice la especificación de la interface procesador/cache y la interface cache/procesador (Apéndice 2.4, Figura 45).

Trabajo 7: Analice la especificación de las interfaces con el bus (Apéndice 2.4, Figura 45).

Trabajo 8: Analice la especificación de la memoria y su controlador (Apéndice 2.4, Figura 45).

Trabajo 9: Para el controlador de cache de la Figura 16, construya una tabla de transiciones entre estados (Apéndice 2.3[13]), donde también se especifique la lógica de salida en la segunda subfila de cada estado (activación, desactivación de las señales). En el caso de la lógica de salida indique sólo la activación de la señal (valor 1). Suponga que por defecto las señales no se activan.

Ubicación del fichero. El fichero con la especificación de la interface del controlador de cache está en el directorio “controlador/CODIGO” (Apéndice 2.4)[14]. En este directorio se incluye, además, un fichero vhdl con la especificación de funciones y procedimiento que son de utilidad. El fichero que contiene la declaración del conjunto de estados se denomina controlador_pkg.vhd y está ubicado en el directorio tipos_constantes_pkg. Analice en detalle las funciones y procedimientos y utilícelos en la codificación del controlador de cache.

Recomendación. Es recomendable codificar y comprobar el funcionamiento de forma incremental. Por ejemplo, empezar con las transiciones entre estados de la transacción inicio y posteriormente seguir con las transiciones entre estados de una instrucción load que acierta

13. En el apéndice se muestra un ejemplo de tabla, donde se explicita la utilización de recursos, pero no la activación de las señales. También se indica el nombre de las señales que deben utilizarse en la especificación. Recuerde que en un acceso a un módulo de almacenamiento las señales correspondientes deben activarse antes del flanco ascendente del reloj (Apéndice 2.1).
14. En el Apéndice 2.3 se describen las señales de la interface.

en cache. Mediante estas dos transacciones se comprueba un funcionamiento básico del camino de datos de la cache y la parte implementada del controlador de cache.

Programa de prueba. Para comprobar el funcionamiento se suministra un fichero con el programa de prueba y otro fichero con procedimientos de utilidad para emitir peticiones (Apéndice 2.5). El programa de prueba, dispone, entre otros, de un proceso (process) productor y un proceso consumidor. En el proceso productor debe especificarse la secuencia de peticiones cuyas transacciones se utilizan para comprobar el funcionamiento del sistema. El proceso consumidor comprueba el valor leído en un acceso de lectura.

En el programa de prueba que se suministra hay especificada la puesta a cero del diseño, una transacción inicio y una transacción load. Las dos transacciones referencian la misma posición de memoria.

Recomendación. Para cada uno de los pasos de implementación incrementales que se indican, prepare una secuencia de accesos a memoria que compruebe el funcionamiento. Analice en detalle la parte del sistema que se comprueba con cada petición y el secuenciamiento de las mismas. Por otro lado, antes de efectuar simulaciones lea detenidamente el Apéndice 2.5.

Trabajo 10: Codifique una descripción funcional del controlador de cache de la Figura 16 que efectúe acciones sólo en las transacciones inicio y lectura con acierto en cache.

Trabajo 11: Elabore la especificación del controlador de cache con Quartus[15]. El directorio QUARTUS asociado almacena los ficheros necesarios (Apéndice 2.5).

Trabajo 12: Elabore la jerarquía de memoria con Quartus (Apéndice 2.5).

Trabajo 13: Analice el programa de prueba. Describa los procesos “productor” y “consumidor” mediante diagramas temporales. Así mismo, describa los procedimientos “Plectura” y “Pescritura”. Para ello, utilice la señal reloj y sus flancos como referente. Céntrese en el protocolo de las interfaces: comunicación entre el procesador y la cache y viceversa.

15. Para que la elaboración tenga lugar incluya todos los estados en la especificación aunque no indique acciones en alguno de ellos. Esta acción también afecta al programa de prueba; parte que imprime información en la ventana textual y ficheros, que se utiliza en la simulación con Modelsim. En caso contrario debe modificar el fichero donde se especifica el conjunto de estados y el procedimiento de impresión de información que se utiliza en el programa de prueba.

Trabajo 14: Compruebe el funcionamiento activando la simulación de Modelsim desde Quartus (Apéndice 2.5). El programa de prueba que se utiliza está almacenado en el directorio PRUEBAS asociado.

Trabajo 15: Añada en la codificación del controlador de cache las acciones necesarias para las transacciones de escritura.

Trabajo 16: Elabore la especificación del controlador de cache con Quartus.

Trabajo 17: Elabore la jerarquía de memoria con Quartus (Apéndice 2.5).

Trabajo 18: Compruebe el funcionamiento activando la simulación de Modelsim desde Quartus (Apéndice 2.5). En el programa de prueba debe especificar transacciones de escritura.

Trabajo 19: Añada en la codificación del controlador de cache las acciones necesarias para la transacción de fallo de lectura.

Trabajo 20: Elabore la especificación del controlador de cache con Quartus.

Trabajo 21: Elabore la jerarquía de memoria con Quartus (Apéndice 2.5).

Trabajo 22: Compruebe el funcionamiento activando la simulación de Modelsim desde Quartus (Apéndice 2.5). En el programa de prueba debe especificar accesos de lectura cuyo bloque no está en cache.

Trabajo 23: Construya una secuencia con el mínimo número de accesos, que muestre una comprobación incremental del diseño. Para este trabajo utilice una única entrada de cache. Muestre en una tabla la información actualizada en cada acceso del programa de prueba.

	Acceso			Memoria		Cache					Camino que se comprueba
ciclo	tipo (load, store)	dirección	trans (Pt, PtE)	variable	valor	contenedor	etiqueta	variable	valor	estado	

Trabajo 24: Una vez esté completamente implementado el controlador de cache, prepare una secuencia de accesos a memoria que compruebe el funcionamiento de forma exhaustiva[16] (comprobar entradas distintas, entrelazar sin accesos consecutivos, accesos a la misma y distinta dirección, conflictos en cache).

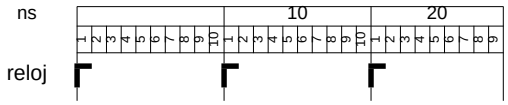
Trabajo 25: En la interface procesador/cache mostrada en la Figura 18, utilizada en el diseño RTL (Apéndice 2.4, Figura 45), todas las señales

16. En el análisis y comprobación tenga en cuenta posibles casos extremos (límite, frontera). Esto es, situaciones que se producen cuando distintos parámetros del sistema están en un extremo del rango permitido.

son entrada del multiplexor y del registro. Además, las señales sólo se almacenan en el registro si se cumple la función lógica “pc_val and pc_listo”. Esto es, si no hay petición y “pc_listo = ‘1’” no se actualiza el registro[17]. Desde el punto de vista de la lógica se utilizan tantos multiplexores y registros como señales (pc_dir, pc_dato, pc_esc y pc_val[18], Figura 17). Rediseñe la interface, dibujando un esquema de circuito, de forma que se utilice el menor número de multiplexores y registros. Considere cada señal pc_dir, pc_dato, pc_esc y pc_val como un todo. Esto, si se utiliza un multiplexor o un registro[19] contabilice una unidad.

Trabajo 26: Represente en un diagrama temporal de retardos los retardos de los componentes en un acierto de lectura. Suponemos que el último ciclo del productor se solapa con el estado DES en el controlador de cache. Los retardos correspondientes al productor deben indicarse en el instante más tardío en el cual es factible, teniendo en cuenta el periodo del reloj. El retardo de la comparación de etiquetas y la puerta “and” de este resultado con la lectura del campo estado tiene un retardo de 0 ns. Recuerde que las señales X_acc y X_esc están incluidas en la lógica de salida del autómata.

1	mux_pet	multiplexor de la interface procesador/cache	10	m_EST	acceso al campo estado
2	reg_pet	registro de la interface procesador/cache	11	m_DAT	acceso al campo datos
3	estado	estado del CC	12	reg_cache_bus	registro de acceso de la cache al bus
4	prx_estado	próximo estado del CC	13	reg bus_mem	registro de acceso del bus a memoria
5	logi_salida	lógica de salida del CC	14	deco_mem	decodificador de memoria
6	deco_ET	decodificador del campo etiquetad	15	memoria	memoria
7	deco_EST	decodificador del campo estado	16	reg mem_bus	registro de acceso de la memoria al bus
8	deco_DAT	decodificador del campo datos	17	reg_bus_cache	registro de acceso del bus a la cache
9	m_ET	acceso al campo etiquetas			



Trabajo 27: Represente en un diagrama temporal de retardos los retardos de los componentes en un fallo de lectura a partir de la primera vez que se está en el estado ESPL.

3.2 Proyecto 2: Reducción del número de estados

El objetivo es reducir la latencia en varias de las transacciones. Para ello se eliminan algunos estados en el controlador de cache[20].

Estos estados son: a) PMX, b) HECHOX y c) el estado LEC en una transacción load que acierta en cache[21].

17. Se reduce el consumo energético.
18. No tenemos en cuenta pc_ini.
19. Para ser precisos hay un multiplexor y un registro por cada bit.
20. La supresión de un estado requiere efectuar las acciones, que se efectuaban en éste estado, en un estado previo en la secuencia de transiciones entre estados.

En la Figura 24, una vez se han eliminado todos los estados, se muestran los ciclos para cada una de las transacciones. En la última fila se muestra la ocupación del bus (ida y vuelta), de la memoria y del CC al finalizar una transacción.

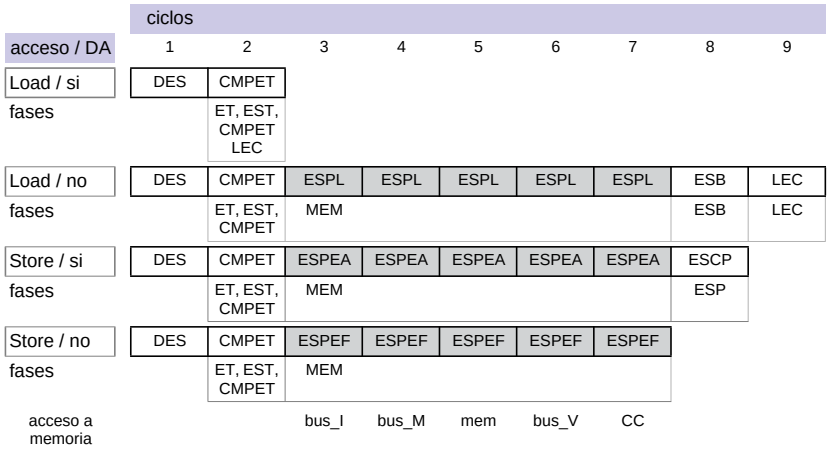


Figura 24 Reducción del número de estados. Ciclos en función del tipo de transacción. Derechos de acceso: DA (AF and est).

Este proyecto se encuentra en el directorio “proyecto_2”. La estructura de directorios es mimética a la del proyecto “proyecto_1”.

En el Apéndice 2.7 y Apéndice 2.8 se detalla la organización del proyecto en directorios, la ubicación de los ficheros y los pasos para efectuar la simulación (Quartus y Modelsim). En el Apéndice 2.9 se indica la ubicación de la documentación generada con Doxygen.

Trabajo 28: El código VHDL de partida para el controlador de cache es el diseñado previamente. Copie el cuerpo de la arquitectura del fichero controlador.vhd del “proyecto_1” en el fichero mimético de este proyecto. En los directorios miméticos incluidos en el directorio “proyecto_2” (Apéndice 2.7), están ubicados los ficheros que contienen la especificación de las funciones y procedimientos utilizados por el controlador (procedimiento_controlador_pkg.vhd) y los posibles estados del controlador (controlador_pkg.vhd).

Recomendación. En los siguientes trabajos se recomienda efectuar una eliminación incremental de los estados. Esto es, estado a estado y comprobar su funcionamiento. Para cada paso incremental, construya la tabla de transiciones entre estados, donde también se especifique la lógica de salida. Modifique el código del controlador de cache que ha copiado. Utilice programas de prueba para comprobar el funcionamiento de forma exhaustiva. Un punto de partida es utilizar

21. Notemos que en este último caso, el acceso de lectura al campo datos es especulativo. Esto es, se efectúa una lectura del campo DAT, la cual no ha modificado la información en la cache, y ésta se utiliza cuando se dispone de los derechos de acceso.

las secuencias de accesos a memoria diseñadas para el proyecto previo. Por otro lado, antes de efectuar elaboraciones o simulaciones lea detenidamente el Apéndice 2.8.

Trabajo 29: Construya una tabla de transiciones entre estados (Apéndice 2.7, Figura 67), donde también se especifique la lógica de salida en la segunda subfila de cada estado (activación, desactivación de las señales), en la cual se eliminen los estados PMX (PML, PMEA, PMEF). En el caso de la lógica de salida indique sólo la activación de la señal (valor 1). Suponga que por defecto las señales no se activan. En un segundo paso modifique la descripción VHDL del controlador de cache para eliminar los estados PMX . Elabore el controlador de cache con Quartus.

Trabajo 30: Elabore la jerarquía de memoria con Quartus (Apéndice 2.8).

Trabajo 31: Compruebe el funcionamiento activando la simulación de Modelsim desde Quartus (Apéndice 2.8). Modifique, si es necesario, el programa de pruebas para comprobar el funcionamiento.

Trabajo 32: Construya una tabla de transiciones entre estados (Apéndice 2.7, Figura 68), donde también se especifique la lógica de salida en la segunda subfila de cada estado (activación, desactivación de las señales), en la cual se eliminen los estados HECHOX (HECHOE, HECHOL). En el caso de la lógica de salida indique sólo la activación de la señal (valor 1). Suponga que por defecto las señales no se activan. En un segundo paso modifique la descripción VHDL del controlador de cache para eliminar, además[22], los estados HECHOX . Elabore el controlador de cache con Quartus.

Trabajo 33: Elabore la jerarquía de memoria con Quartus (Apéndice 2.8).

Trabajo 34: Compruebe el funcionamiento activando la simulación de Modelsim desde Quartus (Apéndice 2.8). Modifique, si es necesario, el programa de pruebas para comprobar el funcionamiento.

Trabajo 35: Finalmente construya una tabla de transiciones entre estados (Apéndice 2.7, Figura 68) donde, además, se elimine el estado LEC en una transacción load que acierta en cache. Modifique la descripción VHDL del controlador de cache de forma oportuna. Elabore el controlador de cache con Quartus.

Trabajo 36: Elabore la jerarquía de memoria con Quartus (Apéndice 2.8).

22. De los estado PMX eliminados en Trabajo 29:.

Trabajo 37: Compruebe el funcionamiento activando la simulación de Modelsim desde Quartus (Apéndice 2.8). Modifique, si es necesario, el programa de pruebas para comprobar el funcionamiento.

Trabajo 38: Construya una secuencia de accesos a memoria donde se comprueben todas las transiciones entre estados.

	Acceso			Memoria		Cache					Secuencia de transiciones
ciclo	tipo (load, store)	dirección	trans (Pt, PtE)	variable	valor	contenedor	etiqueta	variable	valor	estado	

3.3 Proyecto 3. Eliminación completa del estado LEC

El objetivo es reducir la latencia en un fallo de lectura de cache. Para ello hay que eliminar el estado LEC.

En la Figura 25 se muestran los ciclos para cada una de las transacciones. En la última fila se muestra la ocupación del bus (ida y vuelta), de la memoria y del CC al finalizar una transacción.

Para la eliminación del estado LEC, las acciones de actualización del contenedor de cache y suministro del dato (al consumidor) deben de efectuarse en paralelo (Figura 25, load que falla). En consecuencia, hay que modificar el camino de datos, además del controlador de cache.

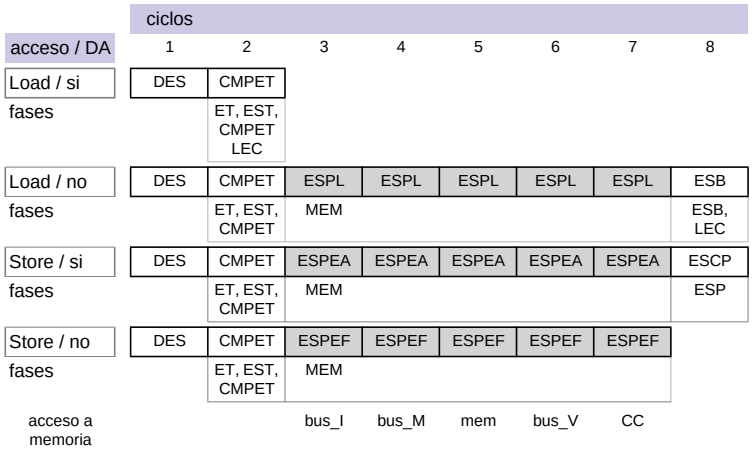


Figura 25 Eliminación completa del estado LECs. Ciclos en función del tipo de transacción. Derechos de acceso: DA (AF and est).

Modificación del camino de datos

En la Figura 26 se muestra la modificación del camino de datos de la cache. Es necesario añadir un multiplexor (resaltado en la figura) y su control para efectuar las acciones en paralelo.

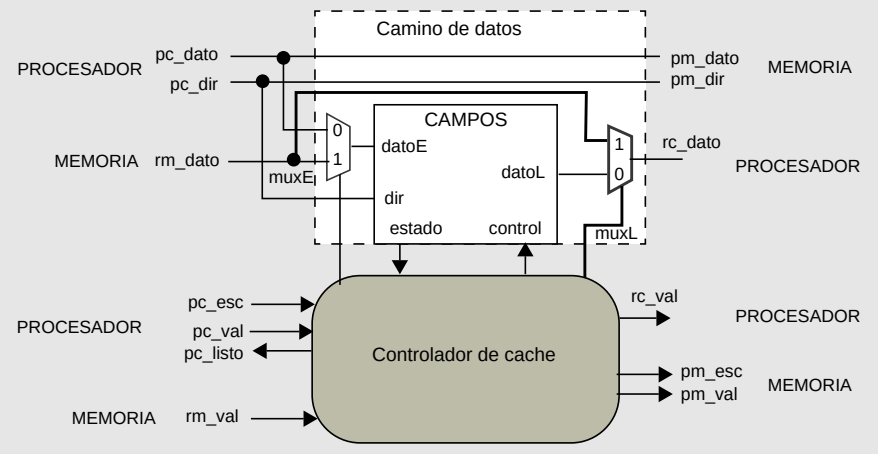


Figura 26 Camino de datos de la cache donde concurrentemente se actualiza la cache y se suministra el dato al procesador.

Este proyecto se encuentra en el directorio “proyecto_3”. La estructura de directorios es mimética a la del proyecto “proyecto_1”.

En el Apéndice 2.10 y Apéndice 2.11 se detalla la organización del proyecto en directorios, la ubicación de los ficheros y los pasos para efectuar la simulación (Quartus y Modelsim). En el Apéndice 2.12 se indica la ubicación de la documentación genera con Doxygen.

Recomendación. Antes de efectuar elaboraciones o simulaciones lea detenidamente el Apéndice 2.11

Trabajo 39: El código VHDL de partida para el controlador de cache es el diseñado en el “proyecto 2”. Copie el cuerpo de la arquitectura del fichero controlador.vhd del “proyecto_2” en el fichero mimético de este proyecto. En los directorios miméticos incluidos en el directorio “proyecto_3” (Apéndice 2.10), están ubicados los ficheros que contienen la especificación de las funciones y procedimientos utilizados por el controlador (procedimiento_controlador_pkg.vhd[23]) y los posibles estados del controlador (controlador_pkg.vhd).

Trabajo 40: Construya la tabla de transiciones entre estados del controlador de cache, donde también se especifique la lógica de salida en la segunda subfila de cada estado (activación, desactivación de las señales) (Apéndice 2.10, Figura 72). Denomine mxL a la señal que controla el multiplexor muxL (Figura 26).

23. En el procedimiento denominado “por_defecto” se incluye el valor por defecto (0) que controla el multiplexor muxL.

Trabajo 41: Modifique la descripción VHDL del controlador de cache para eliminar completamente el estado LEC[24] [25].

Trabajo 42: Elabore el componente controlador.vhd con Quartus. El directorio QUARTUS asociado incluye los ficheros necesarios.

Trabajo 43: El código VHDL de partida para la cache es el suministrado para desarrollar el primer proyecto. Copie el cuerpo de la arquitectura del fichero cache.vhd del "proyecto_1" en el fichero ubicado en el directorio mimético incluido en el directorio "proyecto_3"[26] (Apéndice 2.10).

Trabajo 44: Incluya el multiplexor muxL en el camino de datos de la cache. Modifique de forma estructural la descripción de la cache (fichero cache.vhd).

Trabajo 45: Elabore el componente cache.vhd con Quartus. El directorio QUARTUS asociado incluye los ficheros necesarios. Note que se está utilizando la especificación del controlador de cache. Por tanto, es necesario haber diseñado previamente el controlador de cache.

Trabajo 46: Elabore la jerarquía de memoria con Quartus (Apéndice 2.11).

Trabajo 47: Compruebe el funcionamiento activando la simulación de Modelsim desde Quartus (Apéndice 2.11). Prepare una secuencia de accesos a memoria que compruebe el funcionamiento de forma exhaustiva. Un punto de partida es utilizar las secuencias de acceso a memoria diseñadas para el proyecto previo.

Trabajo 48: Analice la influencia del multiplexor muxL en el tiempo de ciclo.

Trabajo 49: Analice, en este diseño, si es o no factible eliminar los estados ESB y ESCP además del estado LEC.

Trabajo 50: Construya una secuencia de accesos a memoria donde se comprueben todas las transiciones entre estados.

	Acceso			Memoria		Cache					Secuencia de transiciones
ciclo	tipo (load, store)	dirección	trans (Pt, PtE)	variable	valor	contenedor	etiqueta	variable	valor	estado	

24. Recuerde modificar la enumeración de estados del controlador (fichero controlador_pkg.vhd).
25. En el Apéndice 2.11 se indica la señal de control del multiplexor.
26. En este mismo directorio hay un fichero con el mismo nombre que contiene la interface. Elimine este fichero antes de copiar el fichero.

3.4 Proyecto 4: Latencia de un load que acierta en cache igual a uno

El objetivo es reducir la latencia en todas las transacciones y en particular, conseguir que un acierto de lectura en cache tenga una duración de 1 ciclo. Para ello, se elimina el estado CMPET[27] en todas las transiciones entre estados. En estas condiciones es necesario, además de efectuar algunas acciones en un estado previo, efectuar acciones especulativas que no modifiquen la información en la cache.

En la Figura 27 se muestran los ciclos para cada una de las transacciones. En la última fila se muestra la ocupación del bus (ida y vuelta), de la memoria y del CC al finalizar una transacción.

Eliminar el estado CMPET requiere que en el estado DES deben servirse aciertos de lectura e iniciar las acciones oportunas en los otros tipos de accesos. En particular, durante el estado DES, hay que efectuar una lectura de los tres campos de cache correspondientes al contenedor accedido. El acceso a todos los campos de la cache es especulativo. Notemos que es en la etapa DES donde se determina si hay un acceso a cache.

En consecuencia, hay que modificar la interface del procesador con la cache[28], además del camino de datos y el controlador de cache. Respecto a este último, recordemos que la señal de acceso a los campos de la cache debe estar activada antes de efectuar la transición al estado DES.

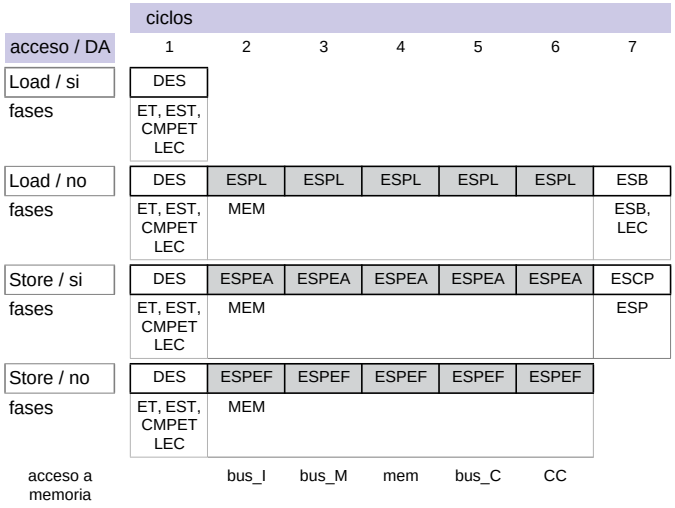


Figura 27 Eliminación completa del estado CMPET. Ciclos en función del tipo de transacción. Derechos de acceso: DA (AF and est).

27. Estrictamente el estado que se elimina es DES. Ahora bien, nosotros seguiremos nombrando DES al estado en la 1ª etapa de un acceso.
28. Recordemos que el acceso a los elementos de almacenamiento está sincronizado con el flanco ascendente de la señal de reloj.

Interface procesador/cache. En la Figura 28 se muestra la interface procesador/cache. Se utiliza un multiplexor (resaltado en la figura) para disponer de la dirección de la petición, en la entrada de los campos de la cache, cuando se efectúa una transición al estado DES. Las otras señales, correspondientes a la petición, se almacenan en un registro de desacoplo, el cual alimenta al camino de datos y al controlador de cache cuando es necesario. Esta interface está gestionada por señales del controlador de cache.

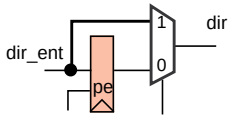


Figura 28 Interface procesador/ cache: señal direccion

Comparación de etiquetas. Por otro lado, es necesario almacenar en un registro la dirección de la petición para poder efectuar la comparación de etiquetas en el estado DES[29]. En la Figura 29 se muestra un esquema de las modificaciones descritas.

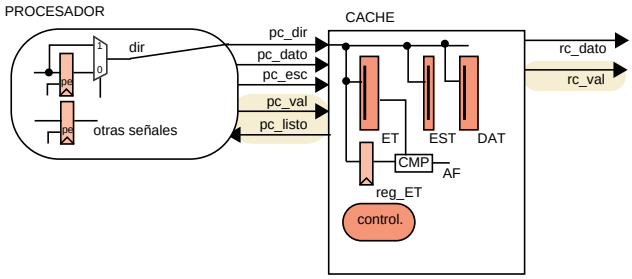


Figura 29 Modificaciones en la interface procesador/cache y en el campo etiquetas. No se muestran los multiplexores muxE y muxL.

Este proyecto se encuentra en el directorio “proyecto_4”. La estructura de directorios es mimética a la del proyecto “proyecto_1” y utiliza la especificación de la cache del “proyecto_3” (cache.vhd).

En el Apéndice 2.13 y Apéndice 2.14 se detalla la organización del proyecto en directorios, la ubicación de los ficheros y los pasos para efectuar la simulación (Quartus y Modelsim). En el Apéndice 2.15 se indica la ubicación de la documentación generada con Doxygen.

Trabajo 51: Analice la especificación de la interface procesador/cache (Apéndice 2.13).

Trabajo 52: Analice la especificación del campo etiquetas (Apéndice 2.13).

Trabajo 53: Para este proyecto, analice la necesidad de disponer de los estados ESB y ESCP.

Trabajo 54: En este diseño la interface procesador/cache ha sido modificada respecto a los diseños previos. Justifique la necesidad de aña-

29. En concreto, los bits correspondientes a la etiqueta. Notemos que el productor (etapa previa a la cache) puede estar generando una nueva petición concurrentemente. Por tanto, la dirección en la entrada dir_ent de la Figura 28 puede cambiar. En un acierto de lectura, el control del multiplexor de la Figura 28 no se modifica durante el ciclo que el CC está en el estado DES. Por tanto, la salida del multiplexor (dir) puede cambiar.

dir, en el camino de datos, un registro que almacene los bits correspondientes a la etiqueta de la dirección. Sustente la explicación utilizando un diagrama temporal similar al de la Figura 19, donde se observe la generación de una dirección por parte del productor, una vez éste reconoce que la cache está procesando la petición previa.

Componente cache. Como especificación de la cache se utiliza el código del proyecto 3. No debe copiarse en la estructura de directorios de este proyecto (Apéndice 2.13).

Trabajo 55: El código VHDL de partida para el controlador de cache es el diseñado en el "proyecto_3". Copie el cuerpo de la arquitectura del fichero controlador.vhd del "proyecto_3" en el fichero mimético de este proyecto[30]. En los directorios miméticos incluidos en el directorio "proyecto_4" (Apéndice 2.13), están ubicados los ficheros que contienen la especificación de las funciones y procedimientos utilizados por el controlador (procedimiento_controlador_pkg.vhd[31]) y los posibles estados del controlador (controlador_pkg.vhd).

Trabajo 56: Construya la tabla de transiciones entre estados del controlador de cache, donde también se especifique la lógica de salida en la segunda subfila de cada estado (activación, desactivación de las señales) (Apéndice 2.13, Figura 76).

Trabajo 57: Modifique la descripción VHDL del controlador de cache para eliminar completamente el estado CMPETIQ.

Trabajo 58: Elabore la especificación del controlador de cache con Quartus.

Trabajo 59: Elabore la jerarquía de memoria con Quartus (Apéndice 2.14).

Trabajo 60: Analice el programa de prueba. Describa los procesos "productor", "consumidor" y "captura". Así mismo, describa los procedimientos "Plectura" y "Pescritura" mediante diagramas temporales. Para ello, utilice la señal reloj y sus flancos como referente. Céntrese en el protocolo de las interfaces.

Trabajo 61: Prepare una secuencia de accesos a memoria que compruebe el funcionamiento de forma exhaustiva[32]. Analice en detalle secuencias de lecturas con acierto en cache a direcciones distintas.

30. En el fichero del "proyecto_4" hay una especificación esquemática de los distintos "process" que se utilizan.
31. Analice en detalle los procedimientos. Hay ligeras modificaciones en la especificación de los mismos y han sido añadidos algunos procedimientos.
32. En el programa de prueba que se suministra hay una secuencia de accesos que puede utilizarse como punto de partida.

Un punto de partida es utilizar las secuencias de accesos a memoria diseñadas para el proyecto previo.

Trabajo 62: Construya una secuencia de accesos a memoria donde se comprueben todas las transiciones entre estados.

	Acceso			Memoria		Cache					Secuencia de transiciones
ciclo	tipo (load, store)	dirección	trans (Pt, PtE)	variable	valor	contenedor	etiqueta	variable	valor	estado	

Trabajo 63: En la interface procesador/cache que se utiliza, todos los registros se actualizan cuando pc_listo = '1'. Rediseñe la interface, dibujando un esquema de circuito, de forma que las señales de entrada, que sea posible, se almacenen en el registro sólo si hay una petición pendiente (esquema de circuito).

Trabajo 64: Represente en un diagrama temporal de retardos los retardos de los componentes en dos acierto de lectura consecutivos a direcciones distintas. El productor tarda 1 ciclo en producir accesos. Los retardos correspondientes al productor deben indicarse en el instante más tardío en el cual es factible, teniendo en cuenta el periodo del reloj. El retardo de la comparación de etiquetas y la puerta "and" de este resultado con la lectura del campo estado tiene un retardo de 0 ns.

1	mux_pet	multiplexor de la interface procesador/cache
2	reg_pet	registro de la interface procesador/cache
3	estado	estado del CC
4	prx_estado	próximo estado del CC
5	logi_salida	lógica de salida del CC
6	deco_ET	decodificador del campo etiquetad
7	deco_EST	decodificador del campo estado
8	deco_DAT	decodificador del campo datos
9	m_ET	acceso al campo etiquetas
10	m_EST	acceso al campo estado
11	m_DAT	acceso al campo datos
12	reg_ET	registro de entrada del comparador (Figura 29)

Apéndice 2.1: Módulo básico de memoria

Una lectura o una escritura a un módulo básico de memoria está sincronizada con el flanco ascendente de la señal de reloj. En la Figura 30 se muestra un esquema donde se identifican registros que almacenan la información existente en la entrada en el flanco ascendente de la señal de reloj. Las señales de entrada y salida se describen en la tabla de la Figura 31.

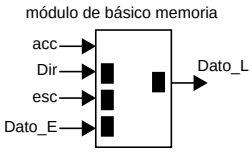


Figura 30 Módulo básico de memoria. Rectángulos negros: registros

Señales	Descripción
acc	acceso al módulo
Dir	dirección a la que se accede
esc	permiso de escritura
Dato_E	dato para actualizar la memoria
Dato_L	dato leído de memoria

Figura 31 Señales de entrada y salida en módulo básico de almacenamiento.

En una operación de lectura hay que activar la señal acc. En una operación de escritura hay que activar las señales acc y esc. La señal Dir debe estar decodificada antes del flanco ascendente de la señal de reloj[33].

El registro en la salida del módulo de memoria de la Figura 30 mantiene el valor leído en el último flanco de reloj en el cual está activa la señal acc[34]. En una operación de escritura el contenido del registro de salida no está definido.

En la Figura 32 se muestra un diagrama temporal de una operación de escritura y posteriormente de una operación de lectura de posiciones de almacenamiento en un banco de memoria.

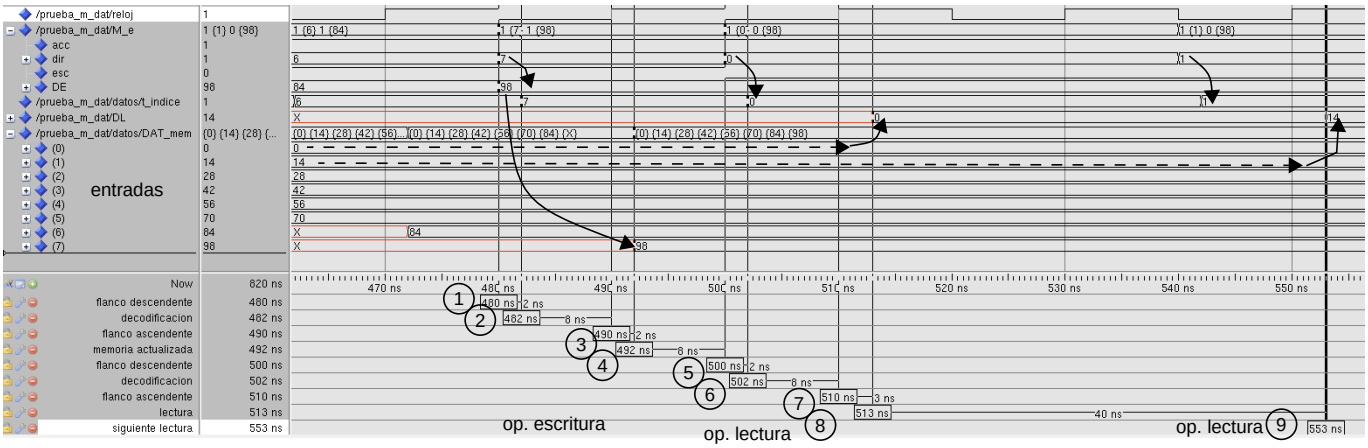


Figura 32 Diagrama temporal de acceso a posiciones de almacenamiento de un banco de memoria.

33. Analice alguno de los ficheros que describen de forma funcional un elemento básico de almacenamiento, como pueden ser m_ET.vhd, m_est.vhd o m_DAT.vhd. La decodificación se emula con una conversión de tipos.
34. En los esquemas de la memoria de la práctica se dibuja una única barra oscurecida para indicar todos los registros.

Los cursores mostrados en la Figura 32 permiten identificar de forma sencilla los retardos.

- Retardo de decodificación: En la marca del primer cursor (flanco descendente) se emiten valores para acceder a memoria. El valor absoluto de la diferencia del tiempo de esta marca con el tiempo de la siguiente marca indica el retardo del decodificador de memoria (2 ns). Se observa el mismo retardo entre las marcas cinco y seis.
- Retardo de actualización de una posición de almacenamiento. La marca del tercer cursor indica el flanco ascendente y la marca del cuarto cursor indica el instante en que se observa el valor con el cual se actualiza la posición de memoria entrada 7 de DAT_mem. La diferencia temporal entre las dos marcas es el retardo de actualización (2 ns).
- Retardo de lectura. Diferencia temporal entre las marcas siete y ocho (señal DL, 3 ns).

Observemos que el valor leído del banco de memoria permanece en la salida hasta que no se vuelve a efectuar un acceso (marca nueve).

A.1.1 Almacenamiento de la información de acceso para un uso posterior

La salida de los registros mostrado en la entrada del módulo de memoria de la Figura 30 no es accesible, Por tanto, para utilizar posteriormente esta información es necesario añadir un registro cuya salida sea accesible.

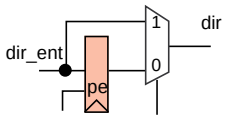


Figura 33 Memorización de la señal dir_ent

En la Figura 33 se muestra un ejemplo de diseño que permite disponer de la dirección que se utiliza al acceder al módulo básico de memoria. En el flanco ascendente, de interés, de la señal de reloj, debe estar seleccionada la entrada ‘1’ del multiplexor. En paralelo, debe almacenarse la información en el registro (pe, permiso de escritura). Posteriormente hay que seleccionar la entrada ‘0’ del multiplexor. En estas condiciones, la señal dir alimenta al módulo básico de memoria y a la lógica.

Un ejemplo de la necesidad descrita está en la fase de detección de acierto o fallo en cache. Hay que comparar el campo etiqueta de la dirección, utilizada en el acceso, con el valor leído del campo etiqueta del contenedor correspondiente Figura 34.

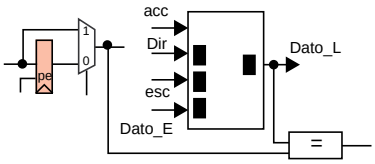


Figura 34 Acceso y comparación del campo etiquetas.

Apéndice 2.2: Utilización del camino de datos

En la Figura 35 se muestra la ubicación explícita de la interface del procesador y la ubicación de los registros en el camino de ida y vuelta de memoria utilizando el bus.

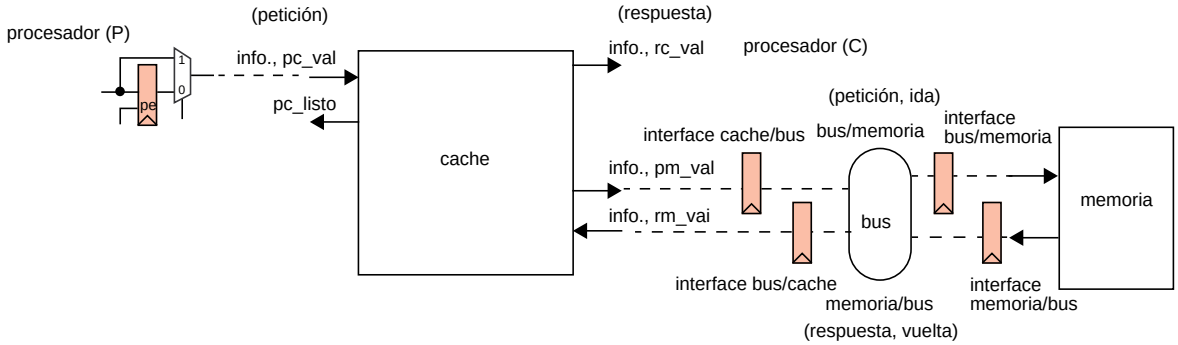


Figura 35 Cache con las interfaces con el procesador y el bus, el cual se utiliza para acceder a memoria. El acrónimo "info." indica cualquier otra información que se transmita y no esté representada.

Acrónimos. Los nombre y acrónimos utilizados en esta memoria de la práctica no se corresponden con los utilizados en el código VHDL que describe los componentes: cache, bus y memoria. En el Apéndice 2.5 se muestra una asociación entre los acrónimos utilizados en el diseño VHDL y las señales utilizadas en la memoria de esta práctica al describir la jerarquía de memoria[35].

En las figuras que siguen en este apéndice no se muestran explícitamente los elementos utilizados en la interface procesador/cache y los registros utilizados en las interfaces con el bus, tanto de la cache como de la memoria.

A.2.1 Elementos utilizados en función de la etapa

Etapa CMPET. En la Figura 36 se muestran, con tono más oscuro, los elementos del camino de datos de la cache utilizados en la etapa CMPET (Figura 14). Estos elementos son los módulos de almacenamiento etiquetas y estado y el comparador. La operación en los campos de cache ET y EST es de lectura (acceso activado y escritura desactivada). El valor leído del campo etiquetas se compara con el campo etiqueta de la dirección. El resultado de la comparación y el valor leído del campo estado son entradas del controlador de cache.

35. En el código VHDL se utiliza de forma rutinaria el constructor "record" para especificar un grupo de señales relacionadas de forma lógica.

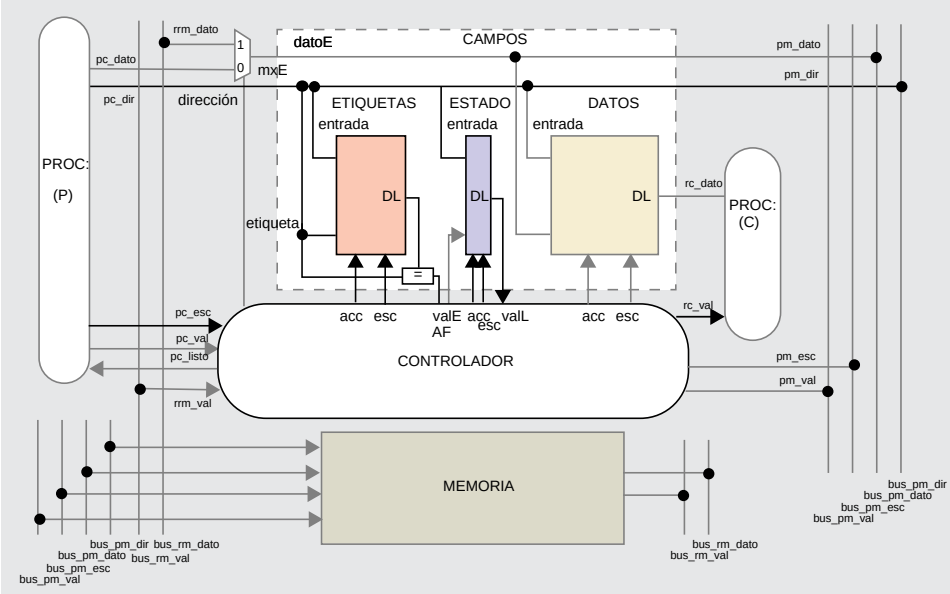


Figura 36 Elementos utilizados de la cache en la etapa CMPET.

Etapa LEC. En la Figura 37 se muestran, con tono más oscuro, los elementos del camino de datos de la cache utilizados en la etapa LEC (Figura 14). Se lee el campo de datos. Para ello, el controlador de cache activa la señal de acceso al campo de datos. La señal de escritura está desactivada.

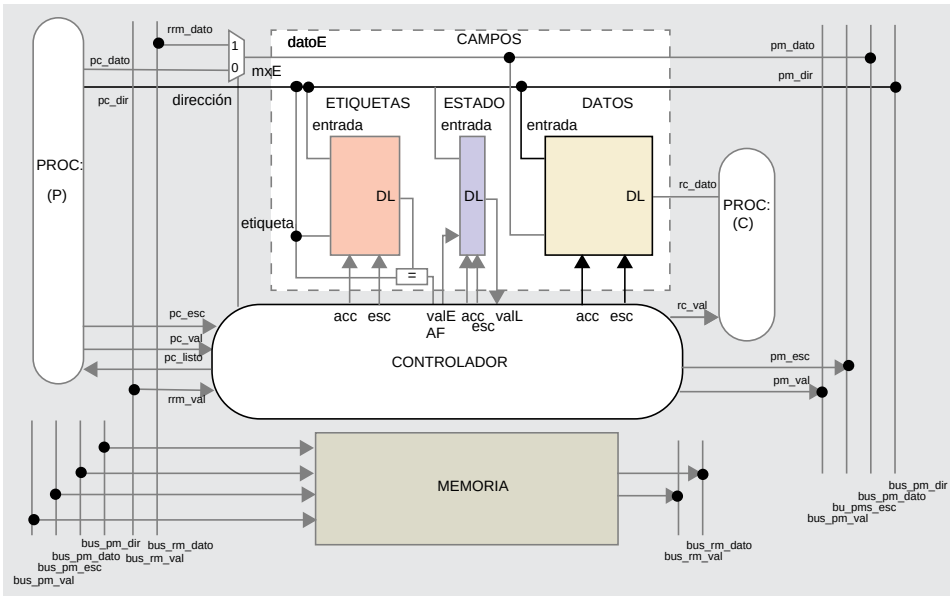


Figura 37 Elementos utilizados de la cache en la etapa LEC.

Etapa HECHOL. En al Figura 38 se muestran, con tono más oscuro, los elementos utilizados en la etapa HECHOL (Figura 14). Se activa la señal rc_val para indicar al procesador (C) que está disponible el dato.

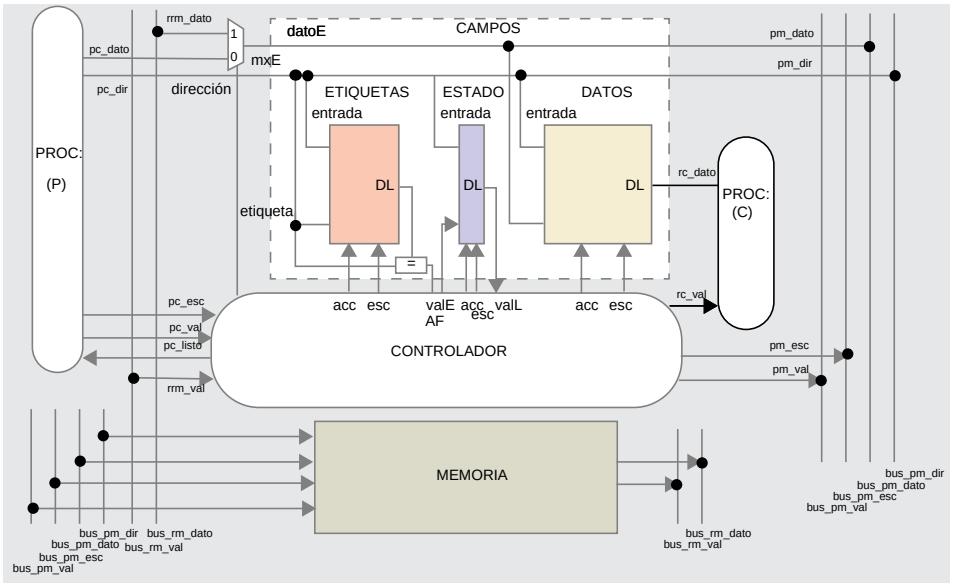


Figura 38 Elementos utilizados de la cache en la etapa HECHOL.

Etapa ESB. En la Figura 39 se muestran, con tono más oscuro, los elementos del camino de datos de la cache utilizados en la etapa ESB (Figura 14). Se actualizan todos los campos de la cache.

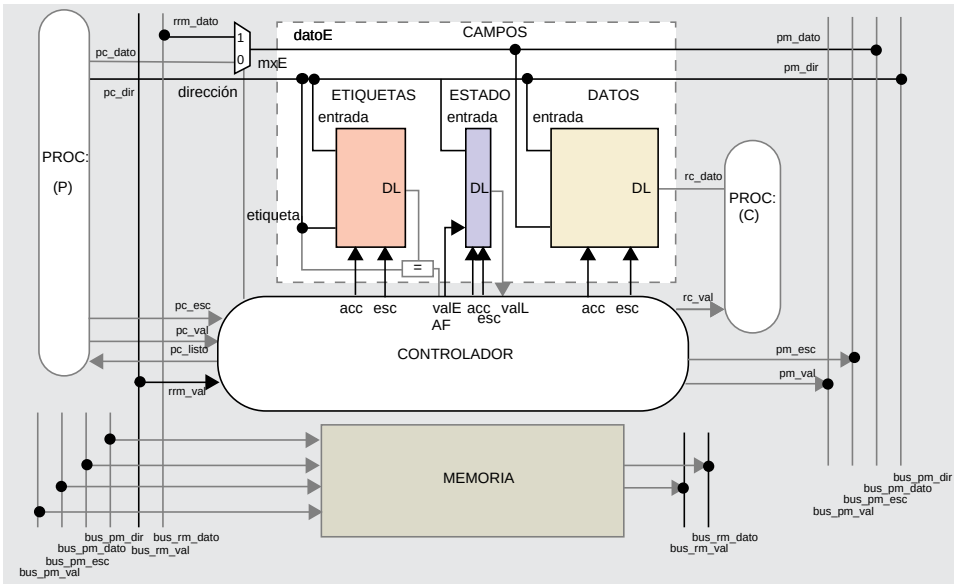


Figura 39 Elementos utilizados de la cache en la etapa ESB.

Apéndice 2.3: Tabla de transiciones entre estados del controlador de cache

Cada fila de la tabla de la Figura 40 se corresponde con un estado del autómata de control. Cada columna se corresponde con una o varias señales de entrada del autómata de control. Por ejemplo, en la primera columna, las señales petición e inicio están activadas. Una señal con una raya encima indica señal negada.

		procesador			señales de estado (AF, est.)				memoria		
		peticion		$\overline{\text{peticion}}$	lectura		escritura		resp. memoria		
		inicio	$\overline{\text{inicio}}$		si	no	si	no	si	no	
Estados	DES0	INI	CMPET	DES0							
		nada	nada	desocupada							
	INI										ESCINI
	ESCINI										HECHOE
											actualiza todos los campos (dato proc.)
	DES		CMPET	DES							
			nada	desocupada							
	CMPET				LEC	PML	PMEA	PMEF			
					leer etq. y comparar, leer est.						
	LEC										HECHOL
											lec. campo dato
	HECHOL										DES
											indicación de dato
	PML										ESPL
											pet. lec. memoria
	ESPL								ESB	ESPL	
									nada	nada	
	ESB										LEC
											actualiza todos los campos (dato mem.)
	PMEA										ESPEA
											pet. esc. memoria
	ESPEA								ESCP	ESPEA	
									nada	nada	
	ESCP										HECHOE
											actualiza campo datos (dato proc.)
	PMEF										ESPEF
											pet. esc. memoria
	ESPEF								HECHOE	ESPEF	
									nada	nada	
	HECHOE										DES
											nada

Figura 40 Tabla de transiciones entre estados y acciones del controlador de cache.

En las columnas identificadas como señales de estado se indica si se dispone de los derechos de acceso; operación “and” de la señal AF y del estado del bloque (est).

En cada casilla de la tabla hay dos tipos de información. En la primera subfila se indica el nuevo estado y en la segunda subfila los recursos que se utilizan, o acción que se efectúa, y el tipo de operación en los recursos. Esta acción no tiene porqué corresponderse con cuándo debe iniciarse la acción. Recordemos que, en elementos como los módulos básicos de almacenamiento, es necesario activar la utilización del mismo antes del flanco ascendente de la señal de reloj. Esto es, en el ciclo previo.

A.3.1 Interface del controlador de cache

En la Figura 41 se muestra un esquema de la interface (nombre de las señales y tipo) del controlador de cache utilizada en el código que se suministra.

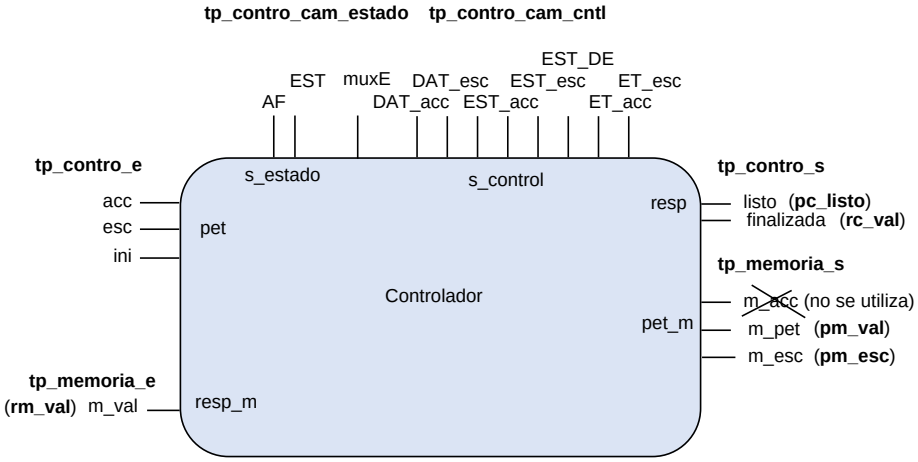


Figura 41 Señales (interior) y tipos (en negrita y subcampo normal) en la interface del controlador de cache. La señal pet_m.m_acc no se utiliza. Los subcampos en negrita son los acrónimos de las señales utilizados en la Figura 1 y Figura 35.

Acrónimos que deben utilizar para la lógica de salida al rellenar la tabla de transiciones entre estados. Los acrónimos que deben utilizarse al rellenar la tabla de la Figura 40, para describir la lógica de salida del controlador de cache, no utilizan el prefijo de la señal. Por ejemplo, en lugar de s_control.DAT_esc se utiliza el acrónimo DAT_esc. Respecto a otras señales, en la Figura 41 se muestra, entre paréntesis y en negrita, el acrónimo que debe utilizarse. Estos últimos acrónimos se corresponden con los utilizados en la Figura 1. Para el multiplexor muxE de la Figura 6 utilice el acrónimo muxE.

En la tabla de la Figura 42 se muestra una relación explícita de los acrónimos que deben utilizarse y el nombre de la señal en el código.

Código		tabla	
resp	resp.listo	pc_listo	cache lista u ocupada
	resp.finalizada	rc_val	en la salida de la cache hay un dato válido
s-control	s_control.ET_acc	ET_acc	acceso al campo etiquetas
	s_control.ET_esc	ET_esc	actualización del campo etiquetas
	s_control.EST_acc	EST_acc	acceso al campo estado
	s_control.EST_esc	EST_esc	actualización del campo estado
	s_control.EST_DE	EST_DE	valor con el cual se actualiza el campo estado
	s_control.DAT_acc	DAT_acc	acceso al campo etiquetas
	s_control.DAT_esc	DAT_esc	actualización del campo etiquetas
	s_control.muxE	muxE	selección de la entrada del multiplexor. Actualización del campo datos con el dato transportado por el bus (muxE = 1)
pet_m	pet_m.m_pet	pm_val	acceso a memoria
	pet_m.m_esc	pm_esc	actualización de memoria
pet	pet.acc		petición de lectura o escritura
	pet.esc		la petición es de escritura
	pet.ini		petición de inicio
resp	resp_m.m_val		respuesta de memoria
s_estado	s_estado.AF		resultado de la comparación de etiquetas
	s_estado.EST		estado leído del campo EST

Figura 42 Acrónimos que deben utilizarse en la descripción del controlador de cache y el nombre asociado en el código.

PRACTICA

Tabla de transiciones entre estados del controlador de cache

Apéndice 2.4: Proyecto 1. Organización de los ficheros

En la Figura 43 se muestra la estructura de directorios del proyecto “proyecto 1”. Los directorios *_pkg contienen ficheros VHDL. Los otros directorios, que son hojas del árbol de directorios, contienen un directorio, denominado CODIGO, que contiene los ficheros VHDL.

```
LAB2
|---- BUS
|   |---- componentes
|   |   |---- registros
|   |---- componentes_interfaces_bus_pkg
|   |---- interfaces
|---- cache_con_interface_bus
|   |---- cache_con_interface_proc
|   |   |---- cache
|   |   |   |---- camino_de_datos
|   |   |   |   |---- componentes
|   |   |   |   |   |---- campo_datos
|   |   |   |   |   |   |---- componentes
|   |   |   |   |   |   |   |---- m_datos
|   |   |   |   |   |   |---- componentes_datos_pkg
|   |   |   |   |   |   |---- ensamblado
|   |   |   |   |   |---- campo_estado
|   |   |   |   |   |   |---- m_estado
|   |   |   |   |   |---- campo_etiquetas
|   |   |   |   |   |   |---- componentes
|   |   |   |   |   |   |   |---- m_etiquetas
|   |   |   |   |   |   |---- componentes_etiq_pkg
|   |   |   |   |   |   |---- ensamblado
|   |   |   |   |   |---- componentes_camino_pkg
|   |   |   |   |   |---- ensamblado
|   |   |   |---- componentes_cache_pkg
|   |   |   |---- componentes_interface_cache_bus
|   |   |   |---- mux_dat
|   |   |---- controlador
|   |   |---- ensamblado
|   |   |---- componentes_cache_con_inter_proc_pkg
|   |   |---- ensamblado_con_interface_proc
|   |   |---- interface_proc_cache
|   |   |   |---- COMPONENTES
|   |   |   |   |---- mux_peticion
|   |   |   |   |---- registro_pet
|   |   |   |---- componentes_interface_proc_cache_pkg
|   |   |   |---- interfaces
|   |---- cache_con_interface_proc_pkg
|   |---- componentes_cache_interface_pkg
|   |---- ensamblado_cache_interfaces
|---- ENSAMBLADO
|---- memoria_con_interface_bus
|   |---- componentes_memoria_interface_pkg
|   |---- ensam_memo_interface
|   |---- memoria_principal
|   |   |---- COMPONENTES
|   |   |   |---- CONTROLADOR
|   |   |   |---- MEMORIA
|   |   |---- componentes_memoria_pkg
|   |   |---- ensam_memoria
|---- tipos_constantes_pkg
|---- UTILIDADES_pkg
|   |---- impri_CACHE_memoria_pkg
|   |---- imprimir_traza_pkg
```

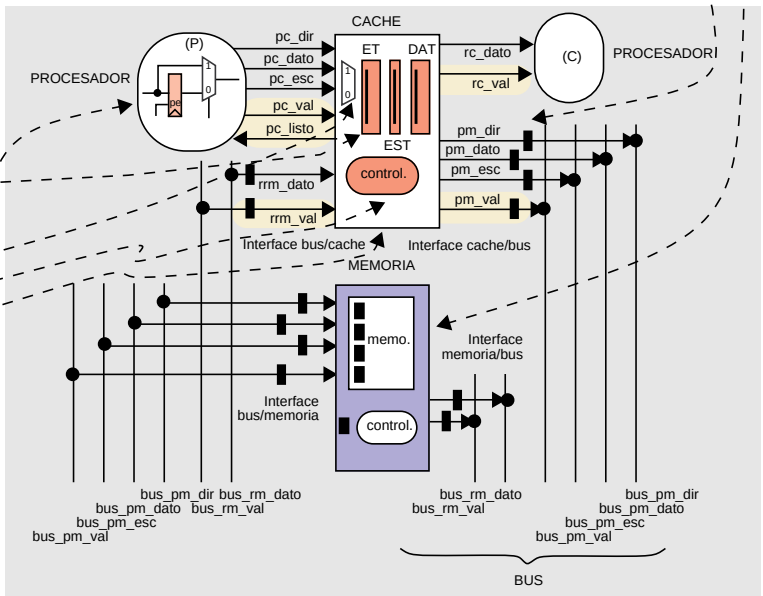


Figura 43 Arbol de directorios. Diseño del controlador de cache.

A.4.1 Descripción de la ubicación de los componentes en los directorios

En la Figura 44 se muestra la ubicación de la descripción de los elementos del camino de datos (ficheros ubicados en el directorio CODIGO)[36].

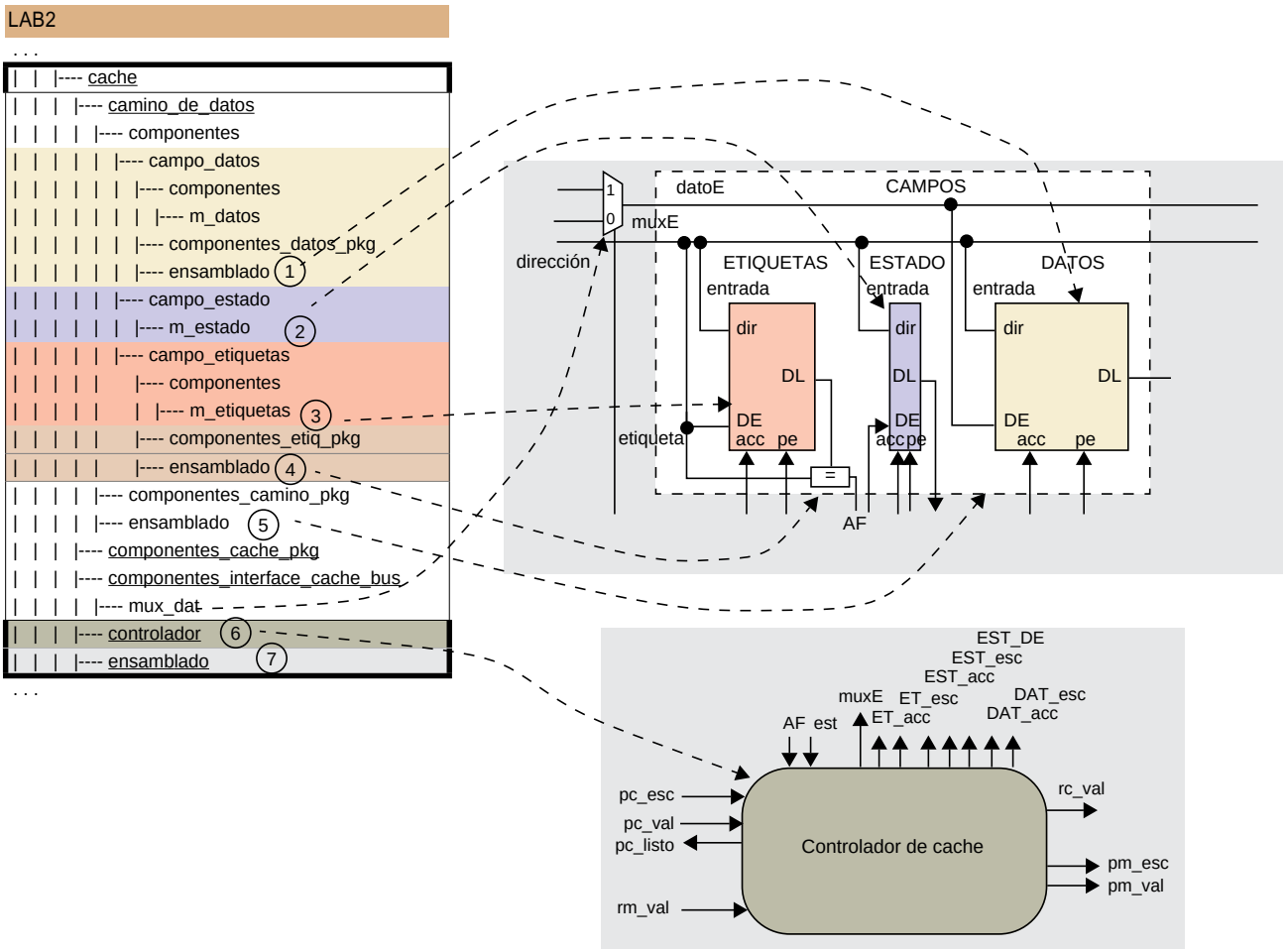


Figura 44 Ubicación en la estructura de directorios del camino de datos y controlador de cache.

1. Campo de datos.
2. Campo de estado.
3. Campo de etiquetas.
4. Campo de etiquetas y comparador.
5. Ensamblado de los campos de la cache
6. Controlador de la cache.
7. Ensamblado, estructural, de los campos de la cache, multiplexor mxE y controlador de cache.

36. Los acrónimos en las figuras no se corresponden con los nombres en la especificación RTL. En el código VHDL se utiliza de forma rutinaria el constructor “record” para declarar un grupo de señales relacionadas de forma lógica.

En la Figura 45 se muestra la ubicación de la descripción de los elementos utilizados en las interfaces, las interfaces mismas y la memoria (ficheros ubicados en el directorio CODIGO)

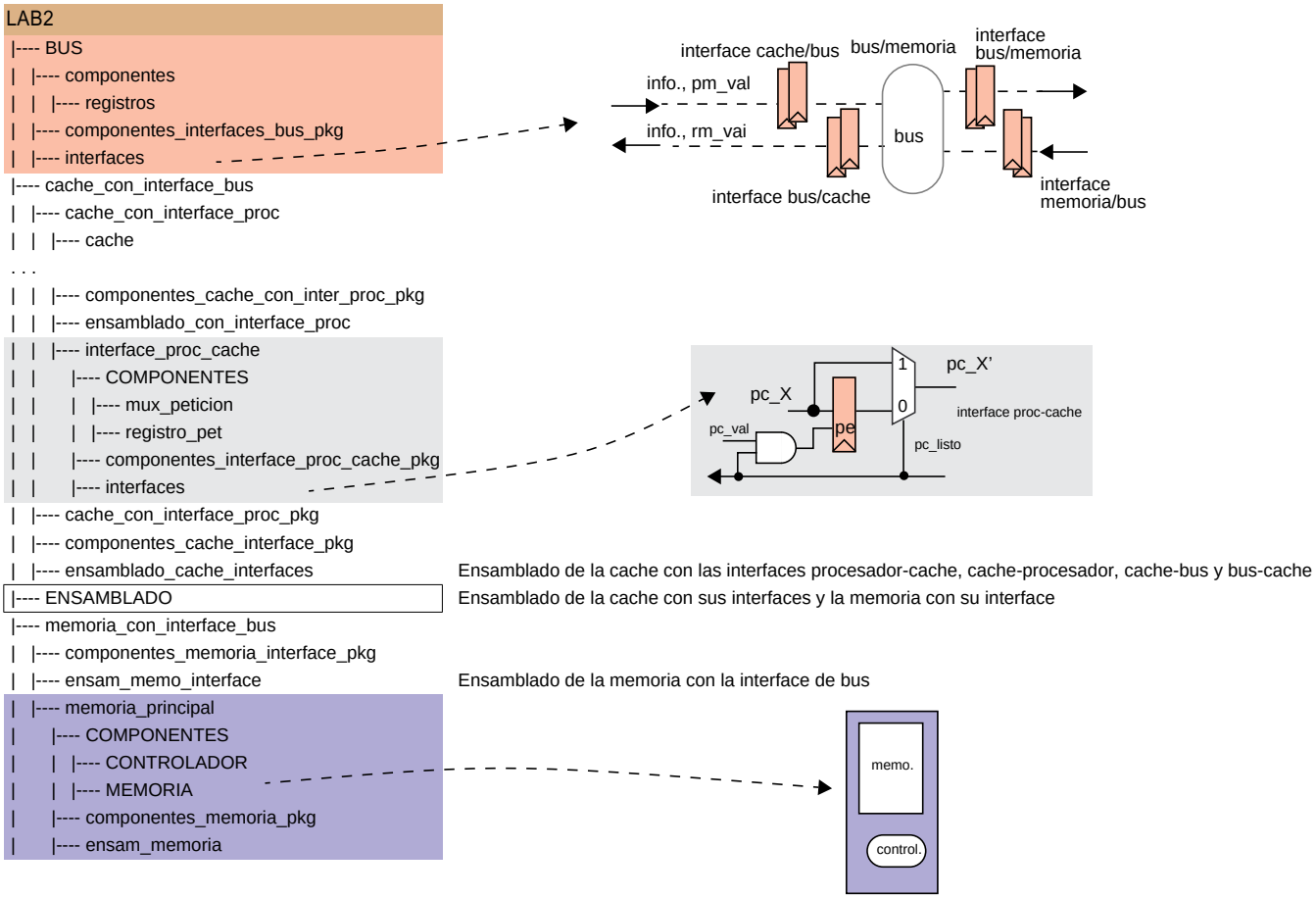


Figura 45 Ubicación en la estructura de directorios de las interfaces y la memoria.

A.4.2 Controlador de cache

El directorio CODIGO, dentro del directorio “controlador” incluye el fichero controlador.vhd y el fichero “procedimientos_controlador_pkg.vhd”. Este último fichero incluye funciones y procedimientos que es obligatorio utilizar en la especificación del controlador de cache.

El fichero controlador.vhd tiene definidas como estado y próximo estado las señales “estado” y “prx_estado” respectivamente. Utilice estas señales para identificar el estado. En caso contrario, si quiere visualizar el estado y el próximo estado, en la ventana temporal de Modelsim, al efectuar la simulación, deberá añadir las nuevas señales utilizadas cuando efectúe la simulación.

La enumeración de los posibles estados del diseño está especificada en el fichero controlador_pkg.vhd, ubicado en el directorio tipos_constantes_pkg. Esta enumeración de estados también se utiliza en el programa de prueba. Más en concreto, en el código que genera la traza (UTILIDADES_pkg/imprimir_traza_pkg).

Edición de ficheros.

Los ficheros que deben editarse contienen la declaración de la interface. Esta no debe modificarse, a menos que se indique lo contrario.

Apéndice 2.5: Proyecto 1. Simulación

A.5.1 Controlador de cache

Un primer paso, para no tener que compilar todo el diseño, es utilizar la herramienta Quartus para comprobar que el controlador se puede elaborar. Para ello, el directorio “controlador”, enmarcado en la Figura 43 (Apéndice 2.4), incluye el directorio QUARTUS. En este directorio, los ficheros *qsf y *.qpf contienen la especificación de un proyecto donde sólo se especifica el controlador.

A.5.2 Jerarquía de memoria

Posteriormente se elabora la cache con el controlador y el sistema de memoria para construir la jerarquía de memoria. El directorio que contiene el ensamblado de estos componentes se denomina “ENSAMBLADO”, enmarcado en la Figura 43, y está ubicado en el primer nivel de directorios (Apéndice 2.4). Los directorios que incluyen los ficheros utilizados por Quartus y Modelsim se indican en la Figura 46. Los ficheros *.qsf y *.qpf del directorio QUARTUS especifican el proyecto Quartus.

Herramienta	Directorio	Directorio
Quartus		
	ENSAMBLADO	QUARTUS
Modelsim		PRUEBAS
		RESULTADOS
	UTILIDADES_pkg	impri_cache_memoria_pkg
		imprimir_traza_pkg

Figura 46 Proyecto 1. Directorios utilizados por las herramientas Quartus y Modelsim.

El directorio “ENSAMBLADO” incluye el directorio PRUEBAS (Figura 47). En este directorio se incluye, entre otros, el programa de prueba.

Directorio	Directorio	Comentario
PRUEBAS		
	prueba_ensamblado.vhd	Programa de pruebas
	procedimientos_peticiones_pkg.vhd	Procedimientos para emitir transacciones
	formato_ventanas.do	Ficheros utilizados por Modelsim para formatear la ventana temporal.
	wave.do	

Figura 47 Proyecto 1. Directorio PRUEBAS.

A.5.3 Programa de prueba

Este programa, entre otros “process” tiene un proceso productor y un proceso consumidor. Estos procesos se sincronizan con la cache utilizando el protocolo listo/válido, completo o degenerado. En

concreto, la interface del productor para emitir una nueva petición y la interface del consumidor para extraer un dato (en instrucciones load) siguen el patrón que se muestra en la Figura 48.

Productor	Consumidor
producción	. . .
wait until rising_edge(reloj) and listo = '1'	wait until rising_edge(reloj) and valido = '1'
. . .	consumo

Figura 48 Interfaces en "process" productor y consumidor del programa de prueba.

Emisión de peticiones. El fichero "procedimientos_peticiones_pkg.vhd" incluye procedimientos para emitir transacciones desde el "process" productor, incluido en el programa de pruebas. Estos procedimientos incluyen la interface productor/cache. Los parámetros de los procedimientos utilizados para emitir peticiones se muestran parcialmente en la Figura 48. En el procedimiento Plectura, el parámetro v_DATO se utiliza para comprobar el valor leído de cache.

Por defecto el programa de pruebas tiene especificadas una transacción de inicio y una transacción de lectura a la misma posición de memoria.

	parámetros	
procedimiento	dirección	valor
Plectura	v_DIR	v_DATO
Pescritura	v_DIR	v_DATO
no_hay_peticion		

Figura 49 Procedimientos para emitir accesos a memoria.

En el directorio pruebas también se incluye un fichero para formatear las señales que se visualizan en la ventana temporal (fichero wave.do).

Transacción de lectura. En un acceso de lectura el procedimiento utilizado requiere como parámetro el valor que se espera leer (v_DATO). Ello permite una comprobación del valor leído. Esta comprobación se efectúa en el proceso consumidor.

Simulación con Modelsim. En el programa de prueba que se suministra, en concreto en el fichero impri_traza_pkg.vhd, se utilizan todos los posibles estados del controlador de cache (controlador_pkg.vhd). Para el diseño incremental del controlador deben de especificar, en la codificación del mismo, todos los posibles estados aunque no se efectúen acciones en algunos de ellos[37]. En caso contrario, debe modificarse el fichero donde se especifican los estados del controlador y deben de modificarse todos los ficheros que utilizan la especificación.

37. En caso contrario, no habrá sido posible elaborar la especificación con Quartus y se producirán errores en la simulación con Modelsim.

A.5.4 Traza

El programa de pruebas utiliza el fichero incluido en el subdirectorio imprimir_traza_pkg del directorio UTILIDADES_pkg. Este fichero incluye procedimientos para visualizar, en la ventana textual de Modelsim, y almacenar en ficheros una traza de la simulación y el contenido de la cache y la memoria.

Ficheros de salida. El programa de pruebas crea dos ficheros en el directorio RESULTADOS. En el fichero resultados_ejecucion.txt se muestra la evolución temporal de cada transacción y el estado del controlador de cache.

En el fichero productor_consumidor.txt se detalla el ciclo de emisión de una petición por el “process” productor, el ciclo en que accede a la cache y en el caso de una transacción load, el ciclo en que finaliza (consumido por el “process” consumidor). También se muestra el contenido de la cache y de la memoria al finalizar la simulación. El contenido de este fichero es, en parte, un resumen del contenido del fichero resultados_ejecucion.txt.

El conjunto (en este caso contenedor) y las posiciones de memoria se representan en decimal. El contenido de los campos de cache etiquetas y datos se representa en hexadecimal. El contenido del campo estado se representa en binario. El contenido de las posiciones de memoria se representa en hexadecimal.

Nota. La información mostrada en estos ficheros no excluye el que deba analizarse en detalle el diagrama temporal que muestra Modelsim. En la generación de estos ficheros no se efectúan comprobaciones detalladas del funcionamiento completo del sistema. Esto es, la información mostrada es sólo una ayuda para comprobar el funcionamiento del sistema.

Ventana textual de Modelsim. El contenido del fichero resultados_ejecucion.txt también se visualiza en la ventana textual de Modelsim a medida que se efectúa la simulación. Al finalizar la simulación se muestra el contenido de los campos de cache y la memoria.

Genéricos en el programa de prueba. En el programa de prueba se utilizan genéricos para controlar la generación de una traza y la posibilidad de una simulación paso a paso, entre otras.

A.5.5 Evolución de las señales del camino de datos

En la Figura 50 se describen algunas de las señales del camino de datos que se muestran en la ventana temporal de Modelsim[38].

1	/prueba_ensamblado/peticion	Peticion del productor (procesador, programa de prueba)
2	/prueba_ensamblado/pet_listo	Señal listo de la cache al productor
3	/prueba_ensamblado/respuesta	Respuesta de la cache al consumidor (procesador, programa de prueba)
4	/prueba_ensamblado/ensa_ca_mmem/cache/cach/Inter_Pr_ca/pe	Permiso de escritura en el registro de la interface procesador/cache
5	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/peticion	Petición que está procesando la cache.
6	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/camino/etiq/ET_Mem/ET_mem	Campo etiquetas de la cache
7	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/camino/estado/EST_mem	Campo estado de la cache
8	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/camino/etiq/DAT_Mem/DAT_mem	Campo datos de la cache
9	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/derechos_acceso	Derechos de acceso de la petición que se está procesando
10	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/estado	Estado del controlador de cache
11	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/prxestado	Próximo estado del controlador de cache
12	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/pet	Petición procesada en el controlador de cache
13	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/s_estado	Señales de estado de entrada en la cache
14	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/s_control	Señales de control de salida de la cache
15	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/resp	Respuesta del controlador de cache al productor y al consumidor
16	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/pet_m	Petición a memoria del controlador de cache
17	/prueba_ensamblado/ensa_ca_mmem/cache/cach/cache_sin/contro/resp_m	Respuesta de memoria procesada por el controlador de cache

Figura 50 Etiquetas utilizadas en las señales que se muestran en la ventana de tiempo.

Un número significativo de las señales ha sido declarado como tipo ”record”. Para visualizar individualmente las señales hay que pulsar con el ratón el símbolo ”+” adjunto a la izquierda de la señal. Las señales con más de 1 bit se representa como “unsigned”.

38. Dada una señal en la ventana temporal se puede utilizar el botón derecho del ratón para que emerja una ventana donde se observa el código donde está declarada la señal. En la ventana emergente, después de pulsar con el ratón, hay que seleccionar “Object Declaration”.

Señales en la ventana temporal y esquema de la jerarquía de memoria.

En la Figura 51 se muestra, utilizando el ordinal de la primera columna de la tabla de la Figura 50, la ubicación de la señal en el diseño de la jerarquía de memoria.

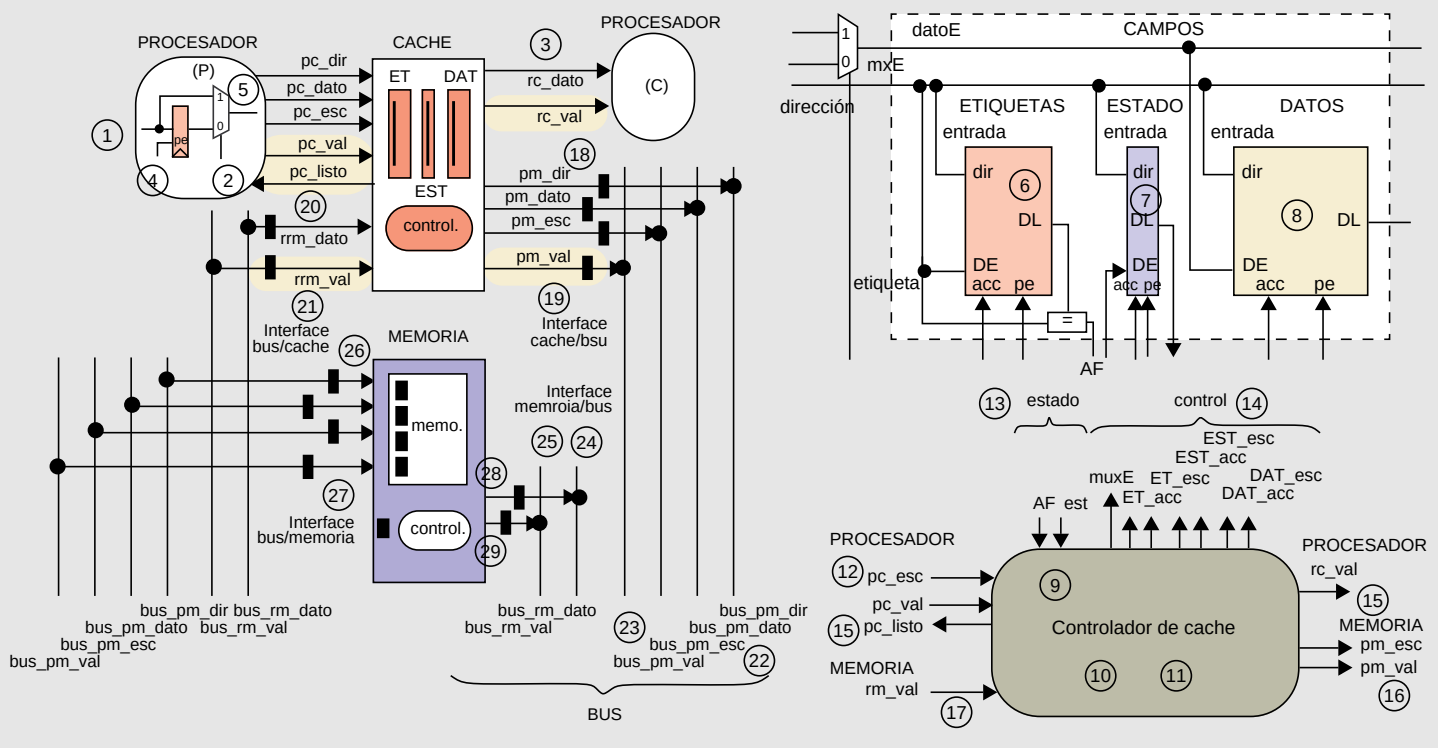


Figura 51 Asociación de señales en el diseño con las señales mostradas en el diagrama temporal de Modelsim.

En la tabla de la Figura 52 se relaciona el resto de señales mostradas en la ventana temporal de Modelsim.

18	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mpet_info_e	Petición: Información en la entrada de la interface cache/bus
19	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mpet_cntl_e	Petición: Control en la entrada de la interface cache/bus
20	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mresp_info_s	Respuesta: Información en la entrada de la interface bus/cache
21	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mresp_cntl_s	Respuesta: Control en la entrada de la interface bus/cache
22	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mpet_info_bus	Petición: Información en el bus
23	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mpet_cntl_bus	Petición: Control en el bus
24	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mresp_info_bus	Respuesta: Información en el bus
25	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mresp_cntl_bus	Respuesta: Control en el bus
26	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mpet_info_s	Petición: Información en la entrada de la interface bus/memoria
27	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mpet_cntl_s	Petición: Control en la entrada de la interface bus/memoria
28	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mresp_info_e	Respuesta: Información en la entrada de la interface memoria/bus
29	/prueba_ensamblado/ensa_ca_mmem/cache/Inter_Pr_BUS_IN/mresp_cntl_e	Respuesta: Control en la entrada de la interface memoria/bus

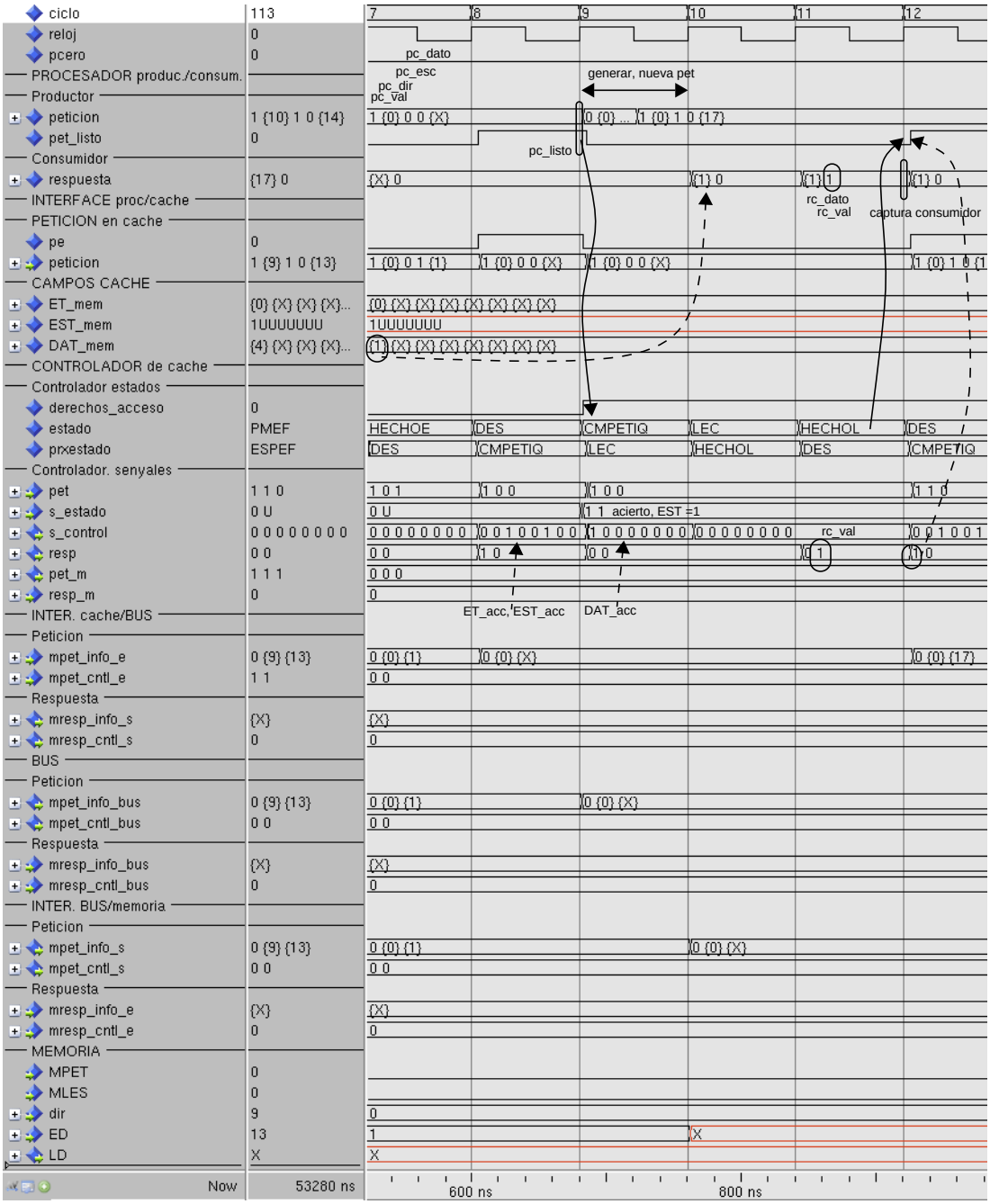
Figura 52 Asociación de señales del bus en el camino de datos con señales mostradas en la ventana temporal de Modelsim.

A.5.6 Ejemplos de las señales en la ventana temporal

En las siguientes figuras, para incrementar el espacio dedicado a la visualización de señales, ha sido eliminado el identificador en la jerarquía de módulos de la señal. Por otro lado, la información en la columna contigua a las señales no es de interés, ya que se corresponde con la posición del curso temporal, el cual no está ubicado en el intervalo observado.

Petición de lectura

En la Figura 53 se muestra el diagrama temporal de una petición de lectura, donde se dispone de los derechos de acceso (acierto y estado del bloque = válido).



	Productor	P1		P2		
	Consumidor					C1
	Etapas	. . .	DES	CM	LEC	HL	DES

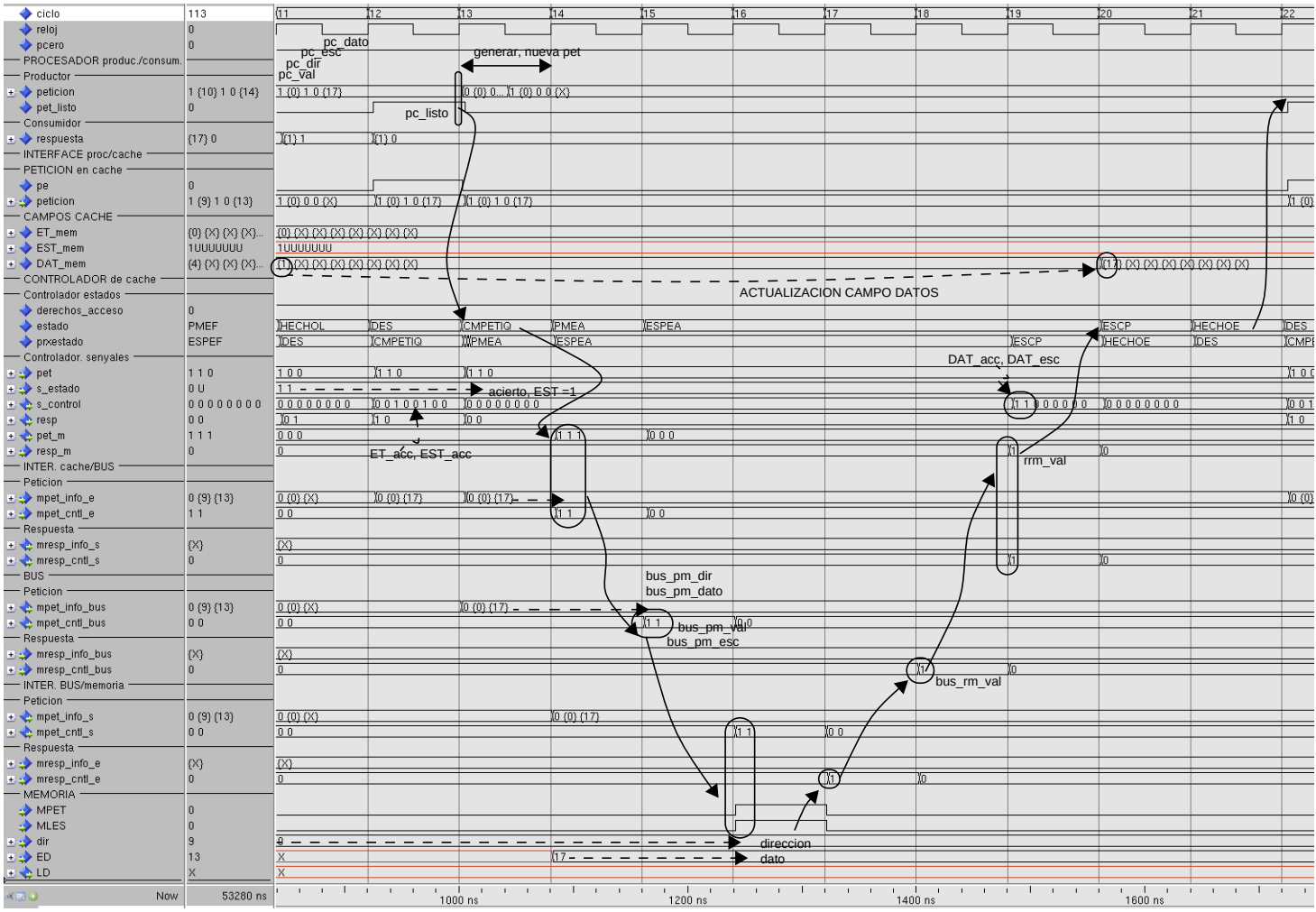
Figura 53 Ventana Temporal de Modelsim en un acceso de lectura con los derechos de acceso.

- Ciclo 7: En la señal petición, debajo del grupo de señales etiquetadas como “Productor”, se observa una petición de lectura (pc_val = 1) en el productor, pendiente de procesar.
- Ciclo 8: En este ciclo el CC activa la señal pc_listo (pet_listo). El estado del CC es DES (señal estado en el grupo de señales “Controlador de cache”). En este estado se activa el acceso a los campos etiquetas y estado de la cache (señal s_control, en el grupo de señales “Controlador, senyales”).
- Ciclo 9: En el flanco ascendente de la señal de reloj, tanto el CC como el productor reconocen que las señales pc_val y pc_listo están activadas. El productor empieza a generar una nueva petición y el CC inicia el procesado de la petición en la entrada (señal peticion en el grupo de señales etiquetadas como “PETICION proc/cache”, “Petición en cache”). El estado del CC es CMPETIQ (señal estado en el grupo de señales “Controlador de cache”). En este ciclo se determina, después de leer los campos etiquetas y estado de cache (contenido de la entrada correspondiente en ET_mem y EST_mem, grupo de señales etiquetadas como “CAMPOS CACHE”), que el bloque está almacenado en cache y el estado del bloque es válido (señal s_estado). Esto es, se dispone de los derechos de acceso (señal derechos_acceso en el grupo de señales “Controlador de cache”). En estas condiciones se activa el acceso al campo datos de la cache (señal s_control).
- Ciclo 10: Se lee el campo datos de la cache (contenido de la entrada correspondiente en DAT_mem). El estado del CC es LEC.
- Ciclo 11: El estado del CC es HECHOL. La señal rc_val (señal respuesta en el grupo de señales “Productor”) se activa y el dato, rc_dato (señal respuesta en el grupo de señales “Consumidor”), está disponible.
- Ciclo 12: En el flanco ascendente de este ciclo el consumidor captura el dato. El estado del CC es DES. El CC activa la señal pc_listo (pet_listo).

Entonces, el número de ciclos de una petición de lectura, que dispone de los derechos de acceso, es 4 ciclos.

Petición de escritura

En la Figura 54 se muestra el diagrama temporal de una petición de escritura, donde se dispone de los derechos de acceso (acierto y estado del bloque = válido).



Productor		P1			P2							
Consumidor												
	Etapas	. . .	DES	CM	AE	ME	ME	ME	ME	ME	EP	HE

Figura 54 Ventana temporal de Modelsim en un acceso de escritura con derechos de acceso.

- Ciclo 11: En la señal petición, debajo del grupo de señales etiquetadas como Productor, se observa una petición de escritura (pc_val = 1, pc_esc = 1) en el productor; pendiente de procesar.
- Ciclo 12: En este ciclo el CC activa la señal pc_listo (pet_listo). El estado del CC es DES (señal estado en el grupo de señales “Controlador de cache”). En este estado se activa el acceso a los campos etiquetas y estado de la cache (señal s_control).
- Ciclo 13: En el flanco ascendente de la señal de reloj, tanto el CC como el productor reconocen que las señales pc_val y pc_listo están activadas. El productor empieza a generar una nueva petición y el CC inicia el procesado de la petición en la entrada (señal peticion en el grupo de señales etiquetadas como “PETI-

CION proc/cache", "Petición en cache"). El estado del CC es CMPETIQ (señal estado en el grupo de señales "Controlador de cache"). En este ciclo se determina, después de leer los campos etiquetas y estado de cache (contenido de la entrada correspondiente en ET_mem y EST_mem), que el bloque está almacenado en cache y el estado del bloque es válido (señal s_estado). Eso es, se dispone de los derechos de acceso (señal derechos_acceso en el grupo de señales "Controlador de cache"). La petición es de escritura y hay que actualizar memoria.

- Ciclo 14: El estado del CC es PMEA. Se efectúa una petición de escritura en memoria, pm_val, pm_esc, (señal pet_m en el grupo de señales "Controlador senyales", también en la entrada del registro de desacoplo con el bus, señal mpet_cntl_e). En la señal mpet_info_e se observan la dirección (0) y el dato (17).
- Ciclo 15: El estado del CC es ESPEA. Las señales mpet_info_e y mpet_cntl_e se propagan al bus (señales mpet_info_bus y mpet_cntl_bus). En el diagrama temporal se identifican las señales, bus_pm_dir, bus_pm_dato, bus_pm_val y bus_pm_esc.
- Ciclo 16: El estado del CC es ESPEA. Las señales correspondientes a la petición se propagan a la memoria (señales mpet_info_s y mpet_cntl_s en el grupo de señales "Peticion" correspondiente al grupo "INTER BUS/memoria"). También se observa en las señales MPET, MLES del grupo de señales "MEMORIA".
- Ciclo 17: El estado del CC es ESPEA. Se actualiza la posición de memoria accedida. Se indica que la operación ha sido realizada (señal mresp_cntl_e en el grupo de señales "Respuesta" correspondiente al grupo "INTER BUS/memoria").
- Ciclo 18: El estado del CC es ESPEA. La respuesta del controlador de memoria se propaga al bus (bus_rm_val, señal mresp_cntl_bus).
- Ciclo 19. El estado del CC es ESPEA. La respuesta del controlador de memoria (rm_val, señal mresp_cntl_s y resp_m) es observada por el CC. El CC activa el acceso y la escritura (DAT_acc, DAT_esc, señal s_control) en el campo de datos de la cache.
- Ciclo 20: El estado del CC es ESCP. El campo de datos de la cache se actualiza (contenido de DAT_mem).
- Ciclo 21: El estado del CC es HECHOE.
- Ciclo 22: El estado del CC es DES. El CC activa la señal pc_listo (pet_listo).

Entonces, el número de ciclos de una petición de escritura, que dispone de los derechos de acceso, es 10 ciclos.

En la Figura 55 se muestra el diagrama temporal de una petición de lectura, donde no se dispone de los derechos de acceso (acierto y estado del bloque = inválido). Partiendo de la descripción detallada de la Figura 53 y la Figura 54 consideramos que es autoexplicativo.

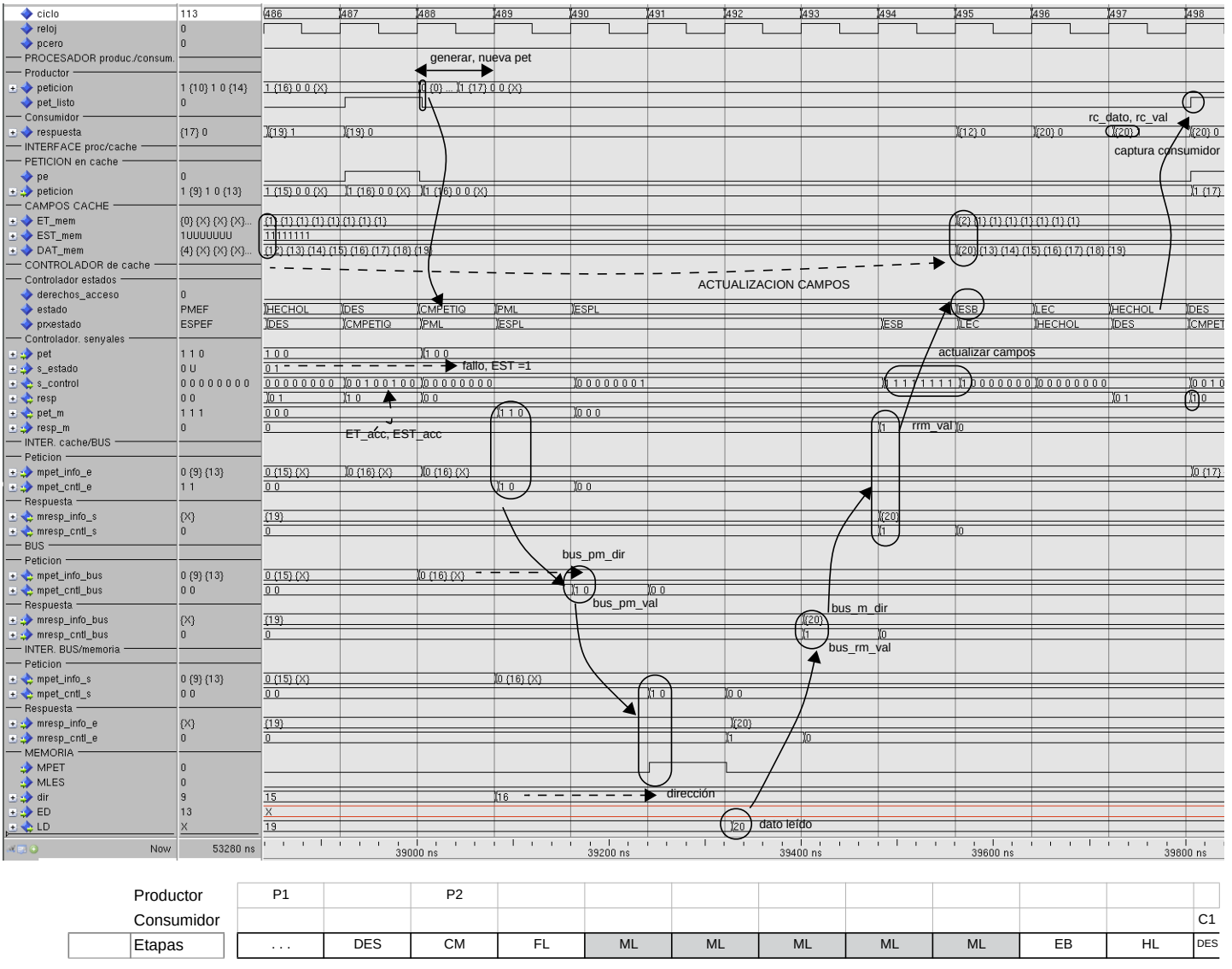


Figura 55 Ventana temporal de Modelsim en un acceso de lectura que no dispone de los derechos de acceso. Adicionalmente se produce un reemplazo.

A.5.7 Información textual de la simulación

En la ventana “Transcript”, textual, de Modelsim se muestra información para cada acceso a memoria y al finalizar la simulación se muestra el contenido del banco de registros y de las memorias[39].

Para observar el progreso de un acceso a memoria hay que analizar como mínimo 4 ciclos consecutivos. En la Figura 56 se muestra una instrucción load que acierta al acceder a cache.

								Campos de cache (actualización)				Campos de cache (contenido)				Memoria
ciclo	acceso pendiente	acceso a cache	AF	etapas				ET	EST	DAT	Estado Controlador	ent	etiq.	est	valor	valor
1	load 0000	load 0000		DS				N	N	N	DES	0	0000	1	04	04
2	no hay		A		CM			N	N	N	CMPETIQ	0	0000	1	04	04
3	store 0002					LE		N	N	N	LEC	0	0000	1	04	04
4	store 0002						HL	N	N	N	HECHOL	0	0000	1	04	04

Figura 56 Representación de un load que acierta al acceder a cache. Los valores se representan en hexadecimal.

Acceso a memoria. Dado un ciclo, la información en la columna "acceso pendiente" (Figura 56) es el acceso a memoria emitido por el productor (procesador). En la columna "acceso a cache" se indica el acceso que está gestionando la cache (sólo se indica el primer ciclo). La información en la columna AF indica si se detecta un acierto o un fallo y el conjunto es válido (derechos de acceso). Esta información se muestra cuando el controlador de cache está en el estado CMPETIQ (segundo ciclo en el acceso).

En las siguientes columnas se representa la segmentación del procesado de un acceso a la jerarquía de memoria. Cada etapa se representa en una nueva fila indicando el ciclo correspondiente. En la Figura 57 se muestran los acrónimos utilizados en el diagrama temporal para representar las etapas y su correspondencia con el estado en el CC.

Estado	DES	INI	ESCINI	HECHOE	CMPETIQ	LEC	HECHOL	PMEA	ESPEA	ESCP	PMEF	ESPEF	PML	ESPL	ESB
Acrónimo	DS	IN	ES	HE	CM	LE	HL	AE	ME	EP	FE	ME	FL	ML	EB

Figura 57 Acrónimos utilizados para representar los estados en el diagrama temporal.

Representación de cada tipo de acceso. La segmentación de cada uno de los posibles tipos de acceso se muestra en la Figura 58. En la última fila se muestra la ocupación del bus, de la memoria y del CC, cuando es necesario su utilización (ML, ME, Figura 20, Figura 21).

Acceso	ciclos										
inic	D0	IN	ES	HE							
Load (acierto)	DS	CM	LE	HL							
Load (fallo)	DS	CM	FL	ML	ML	ML	ML	ML	EB	LE	HL
Store (acierto)	DS	CM	AE	ME	ME	ME	ME	ME	EP	HE	
Store (fallo)	DS	CM	FE	ME	ME	ME	ME	ME	HE		
				bus_i	bus_M	M	bus_V	CC			

Figura 58 Acrónimos utilizados al progresar un acceso a memoria.

39. Las dos informaciones son opcionales. Se controlan mediante genéricos en el programa de prueba.

Actualización de los campos de cache y estado del CC. En cada fila también se representa información sobre la actualización de los campos de cache, el estado del controlador de cache, el contenido de los campos de cache de la entrada accedida y el contenido de la posición de memoria accedida.

La actualización o no de un campo se indica con los acrónimos S y N respectivamente (columnas ET, est y DAT). Para el estado del controlador de cache se utiliza el acrónimo utilizado en el controlador de cache.

Contenido de la entrada accedida y memoria. La entrada accedida se representa en decimal, los campos etiquetas y datos en hexadecimal y el campo estado en binario (columnas ent, etiq, est, val.). El contenido de la posición de memoria se representa en hexadecimal (columna Memoria, valor).

En la Figura 59 se muestra un load que falla al acceder a cache. Además, el contenedor no tiene información válida. En el ciclo 9 (estado ESB) se observa la actualización del contenedor con el bloque[40].

															Campos de cache (actualización)				Campos de cache (contenido)				Mem.
ciclo	acceso pendiente	acceso a cache	AF	etapas											ET	EST	DAT	Estado Controlador	ent	etiq.	est	val.	val.
1	load 00001	load 0001		DS											N	N	N	DES	1	XXXX	U	XX	05
2	no hay		F		CM										N	N	N	CMPETIQ	1	XXXX	U	XX	05
3	store 0002					FL									N	N	N	PML	1	XXXX	U	XX	05
4	store 0002						ML								N	N	N	ESPL	1	XXXX	U	XX	05
5	store 0002							ML							N	N	N	ESPL	1	XXXX	U	XX	05
6	store 0002								ML						N	N	N	ESPL	1	XXXX	U	XX	05
7	store 0002									ML					N	N	N	ESPL	1	XXXX	U	XX	05
8	store 0002										ML				N	N	N	ESPL	1	XXXX	U	XX	05
9	store 0002											EB			S	S	S	ESB	1	0000	1	05	05
10	store 0002												LE		N	N	N	LEC	1	0000	1	05	05
11	store 0002													HL	N	N	N	HECHOL	1	0000	1	05	05

Figura 59 Representación de un load que falla al acceder a cache.

40. Tengamos en cuenta que en la representación de 15 bits en hexadecimal se utilizan 4 símbolos.

En la Figura 60 se muestra un store que acierta en cache. La memoria se actualiza en el ciclo 6 y el campo de datos de la cache en el ciclo 9.

														Campos de cache (actualización)				Campos de cache (contenido)				Mem.
ciclo	acceso pendiente	acceso a cache	AF	etapas										ET	EST	DAT	Estado Controlador	ent	etiq.	est	val.	val.
1	store 0000	store 0000		DS										N	N	N	DES	0	0000	1	11	11
2	store 0002		A		CM									N	N	N	CMPETIQ	0	0000	1	11	11
3	store 0002					AE								N	N	N	PMEA	0	0000	1	11	11
4	store 0002						ME							N	N	N	ESPA	0	0000	1	11	11
5	store 0002							ME						N	N	N	ESPA	0	0000	1	11	11
6	store 0002								ME					N	N	N	ESPA	0	0000	1	11	04
7	store 0002									ME				N	N	N	ESPA	0	0000	1	11	04
8	store 0002										ME			N	N	N	ESPA	0	0000	1	11	04
9	store 0002											EP		N	N	S	ESCP	0	0000	1	04	04
10	store 0002												HE	N	N	N	HECHOE	0	0000	1	04	04

Figura 60 Representación de un store que acierta al acceder a cache.

Almacenamiento en disco de la traza. En el fichero resultados_ejecucion.txt, ubicado en el directorio RESULTADOS, se almacena la traza mostrada en la ventana textual de Modelsim.

Fichero productor_consumidor.txt . En este fichero, ubicado en el directorio RESULTADOS, se almacena el contenido de los contenedores de cache y la memoria al finalizar la simulación.

El formato utilizado para mostrar el contenido de los campos de cache se muestra en la Figura 61.

número de entrada	Campo etiqueta	Campo estado	Campo datos
decimal	valor en hexadecimal	valor en binario	valor en hexadecimal

Figura 61 Formato de impresión de los campos de cache. El valor "X" indica que no ha sido inicializado.

El formato utilizado para mostrar el contenido de las posiciones de almacenamiento de la memoria se indica en la Figura 62. Las posiciones de almacenamiento se muestran en secuencia.

Dirección	Contenido
hexadecimal	valor en hexadecimal

Figura 62 Formato de impresión de memoria.

Posiciones de almacenamiento consecutivas con el mismo valor no se representan, Se representa la primera posición con el valor que es idéntico a las posiciones que siguen en secuencia. Posteriormente se muestra una fila ". . . . .". Seguidamente se muestra otra fila con el valor identificado previamente, que es idéntico. Posteriormente se muestran filas con valores distintos. El valor "X" indica que la posición de memoria no ha sido actualizada.

En el fichero productor_consumidor.txt también se almacena el ciclo de emisión de una petición por parte del productor, el ciclo en que es procesada por la cache y el ciclo de consumo por parte del consumidor, en el caso de un load.

En la Figura 63 se muestra la información en una línea cuando el productor produce un acceso (prod. store, prod. load). También se muestra la identificación del acceso cuando se inicia el procesado en la cache (store cache, load cache).

ciclo	acción	dirección	valor	descripción
	prod. store			producción
decimal	store cache	hexadecimal	hexadecimal	inicio del procesado
	prod. load			producción
	load cache			inicio del procesado

Figura 63 Representación de la emisión de accesos por parte del productor y su procesado en la cache.

En la Figura 64 se muestra la información que se visualiza cuando finaliza el procesado de una instrucción load. Se indica la acción de consumo, el valor leído y la instrucción que ha efectuado la lectura.

ciclo	acción	valor	acceso	dirección
decimal	consumo	hexadecimal	load	hexadecimal

Figura 64 Representación de las acciones del consumidor.

A.5.8 Tiempo de ciclo

El tiempo de ciclo está predeterminado por el camino con mayor retardo. Para determinarlo 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 precedencia.

Por otro lado, hay que tener en cuenta todos los tipos de accesos. Las salidas de algunos elementos no son utilizados por algún tipo de acceso. Por ello, hay que calcular el peor caso. Esto es, el retardo máximo que necesitan algunos accesos.

En la Figura 65 se muestra el camino de datos con los retardos asociados a los componentes. Los retardos de los componentes utilizados se detallan en el Apéndice 2.16. Estos retardos no son representativos de un diseño. Sólo son de utilidad para efectuar los cálculos de retardo que se muestran o solicitan.

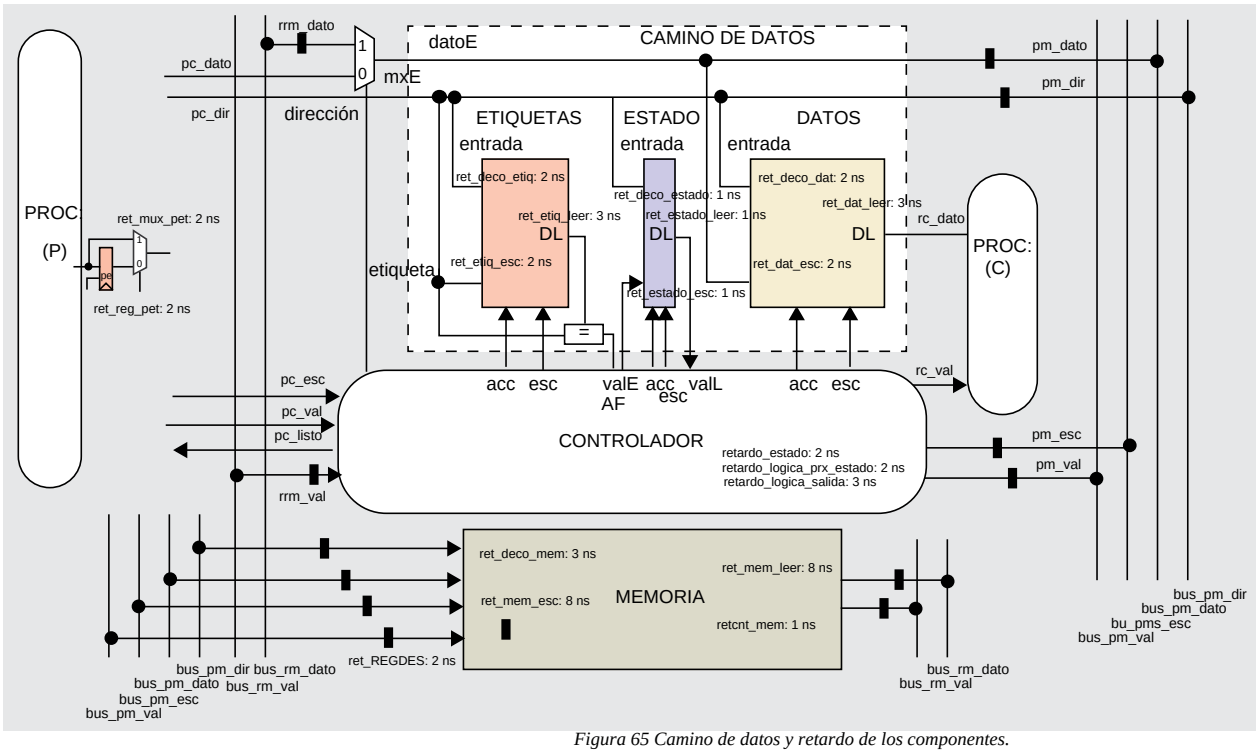


Figura 65 Camino de datos y retardo de los componentes.

Apéndice 2.6: Proyecto 1. 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 DOCUMENTACION/proyecto_1.

Las pestañas que se muestran son autoexplicativas. Por ejemplo, en la pestaña "Archivos" se observa la estructura de directorios y los ficheros que incluyen. En la parte derecha se muestra una secuencia de naturales para seleccionar el nivel jerárquico hasta el cual se muestra la organización.

En la pestaña "Design Unit List" se muestran las unidades de diseño. Después de seleccionar la pestaña previa se observa la pestaña "Design Unit Hierarchy". Seleccionando esta pestaña se observa el diseño jerárquico.

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.

Apéndice 2.7: Proyecto 2. Organización de los ficheros

El árbol de directorios, desde el directorio raíz, de este proyecto es mimético al árbol del proyecto previo. En los subdirectorios miméticos incluidos en este proyecto (“proyecto_2”, Figura 66), sólo se incluyen los directorios que contienen ficheros modificados o que deben modificarse respecto al proyecto previo. También se incluyen ficheros, que aunque no deben modificarse, facilitan la independencia entre proyectos.

```
Proyecto_2
|-- cache_con_interface_bus
|   |-- cache_con_interface_proc
|   |   |---- cache
|   |   |   |---- controlador
|   |   |   |   |---- CODIGO
|   |   |   |   |   |---- controlador.vhd                    Copiar cuerpo del “proyecto 1”
|   |   |   |   |   |---- procedimientos_controlador_pkg.vhd
|-- ENSAMBLADO
|   |-- CODIGO
|   |   |-- ensamblado.vhd
|   |-- PRUEBAS
|   |   |-- procedimientos_peticiones_pkg.vhd
|   |   |-- prueba_ensamblado.vhd
|-- tipos_constantes_pkg
|   |-- controlador_pkg.vhd
|-- UTILIDADES_pkg
    |-- imprimir_traza_pkg
        |-- impri_traza_pkg.vhd
```

Figura 66 Arbol de directorios modificados del proyecto “Proyecto_2”.

A.7.1 Tabla de transiciones entre estados. Eliminación de los estados PMX

En la Figura 67 se muestran las entradas en la tabla de transiciones entre estados.

Estados	procesador			señales de estado (AF, est.)				memoria	
	peticion		$\overline{\text{peticion}}$	lectura		escritura		resp. memoria	
	inicio	$\overline{\text{inicio}}$		si	no	si	no	si	no
DES0									
INI									
ESCINI									
DES									
CMPET									
LEC									
HECHOL									
ESPL									
ESB									
ESPEA									
ESCP									
ESPEF									
HECHOE									

Figura 67 Tabla de transiciones entre estados. Eliminación de los estados PMX.

A.7.2 Tabla de transiciones entre estados. Eliminación de los estados HECHOX

En la Figura 68 se muestran las entradas en la tabla de transiciones entre estados.

Estados	procesador			señales de estado (AF, est.)				memoria	
	peticion		$\overline{\text{peticion}}$	lectura		escritura		resp. memoria	
	inicio	$\overline{\text{inicio}}$		si	no	si	no	si	no
DES0									
INI									
ESCINI									
DES									
CMPET									
LEC									
ESPL									
ESB									
ESPEA									
ESCP									
ESPEF									

Figura 68 Tabla de transiciones entre estados. Eliminación de los estados HECHOX.

Apéndice 2.8: Proyecto 2. Simulación

A.8.1 Controlador de cache

Un primer paso, para no tener que compilar todo el diseño, es utilizar la herramienta Quartus para comprobar que el controlador se puede elaborar. Para ello, el directorio “controlador” incluye el directorio QUARTUS. Los ficheros *qsf y *.qpf contienen la especificación de un proyecto donde sólo se especifica el controlador.

Elaboración con Quartus. La enumeración de los posibles estados del controlador de cache está especificada en el fichero controlador_pkg.vhd ubicado en el directorio “tipos_constantes_pkg” (Apéndice 2.4, Apéndice 2.7), que ha copiado del “proyecto 1”. A medida que vaya eliminando estados modifique la enumeración para que no se produzcan errores de compilación al elaborar el diseño con Quartus.

A.8.2 Jerarquía de memoria

Posteriormente se elabora la cache con el controlador y el sistema de memoria para construir la jerarquía de memoria. El directorio que contiene el ensamblado de estos componentes se denomina “ENSAMBLADO” y está ubicado en el primer nivel de directorios (Apéndice 2.4, Apéndice 2.7). Los directorios que incluyen los ficheros utilizados por Quartus y Modelsim se indican en la Figura 69. Los ficheros *.qsf y *.qpf del directorio QUARTUS especifican el proyecto Quartus.

Herramienta	Directorio	Directorio
Quartus		
	ENSAMBLADO	QUARTUS
Modelsim		PRUEBAS
		RESULTADOS
	UTILIDADES_pkg	impri_cache_memoria_pkg
		imprimir_traza_pkg

Figura 69 Proyecto_2. Directorios utilizados por la herramientas Quartus y Modelsim.

El directorio “ENSAMBLADO” incluye el directorio PRUEBAS (Figura 70). En este directorio se incluye, entre otros, el programa de prueba.

Directorio	Directorio	Comentario
PRUEBAS		
	prueba_ensamblado.vhd	Programa de pruebas
	procedimientos_peticiones_pkg.vhd	Procedimientos para emitir transacciones.
	formato_ventanas.do	Ficheros utilizados por Modelsim para formaterar la ventana temporal.
	wave.do	

Figura 70 Proyecto_2. Directorio PRUEBAS.

Simulación con Modelsim. En el programa de prueba que se suministra, en concreto en el fichero impri_traza_pkg.vhd, han sido comentadas las líneas de código relativas a todos los estados que deben eliminarse. Para la comprobación incremental deben de suprimirse los comentarios correspondientes a los estados que no hayan sido eliminados en el código[41]. Posteriormente al eliminar estos estados en el CC, deben de volverse a comentar. En la ventana temporal de Modelsim se muestran las mismas señales que en el proyecto previo (wave.do).

Programa de prueba. Este programa es una copia del programa suministrado en el “proyecto 1”. Los procedimientos que deben utilizarse para emitir peticiones son los mismos.

41. En caso contrario se producirán errores en la simulación con Modelsim. Es probable que el alineamiento de la información en los dicheros de salida no sea perfecta durante la eliminación incremental.

Apéndice 2.9: Proyecto 2. Documentación

La documentación es la misma que en el proyecto 1.

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 DOCUMENTACION/proyecto_2.

Apéndice 2.10: Proyecto 3. Organización de los ficheros

El árbol de directorios, desde el directorio raíz, de este proyecto es mimético al árbol del proyecto previo. En los subdirectorios miméticos incluidos en este proyecto (“proyecto_3”, Figura 71), sólo se incluyen los directorios que contienen ficheros modificados o que deben modificarse respecto al proyecto previo. También se incluyen ficheros, que aunque no deben modificarse, facilitan la independencia entre proyectos.

Proyecto_3	
|-- cache_con_interface_bus	
| |-- cache_con_interface_proc	
| | |---- cache	
| | | |---- *controlador*	
| | | | |---- CODIGO	
| | | | | |---- controlador.vhd	Copiar cuerpo del “proyecto 2”
| | | | | |---- procedimientos_controlador_pkg.vhd	
| | | |---- *ensamblado*	
| | | |---- CODIGO	
| | | | |---- cache.vhd	Copiar cuerpo del “proyecto 1”
| | | |---- QUARTUS	
|-- ENSAMBLADO	
| |-- CODIGO	
| | |-- ensamblado.vhd	
| |-- PRUEBAS	
| | |-- procedimientos_peticiones_pkg.vhd	
| | |-- prueba_ensamblado.vhd	
|-- tipos_constantes_pkg	
| |-- controlador_pkg.vhd	Se añade la señal muxL al tipo tp_contro_cam_cntl
|-- UTILIDADES_pkg	
|-- imprimir_traza_pkg	
|-- impri_traza_pkg.vhd	

Figura 71 Arbol de directorios modificados del “proyecto_3”.

A.10.1Tabla de transiciones entre estados. Eliminación del estado LEC

En la Figura 72 se muestran las entradas en la tabla de transiciones entre estados.

		procesador			señales de estado (AF, est.)				memoria	
		peticion		$\overline{\text{peticion}}$	lectura		escritura		resp. memoria	
		inicio	$\overline{\text{inicio}}$		si	no	si	no	si	no
Estados	DES0									
	INI									
	ESCINI									
	DES									
	CMPET									
	ESPL									
	ESB									
	ESPEA									
	ESCP									
	ESPEF									

Figura 72 Tabla de transiciones entre estados. Eliminación del estado LEC.

Apéndice 2.11: Proyecto 3. Simulación

En este proyecto se identifican dos subproyectos. Ahora bien, no se distinguen con directorios específicos por subproyecto. En el primer proyecto debe especificarse, de forma funcional, el controlador de cache. En el segundo subproyecto debe especificarse, de forma estructural, la cache. Además, debe efectuarse la elaboración de los mismos. Seguidamente debe elaborarse la jerarquía de memoria y su simulación correspondiente.

A.11.1Subproyecto 1

Control del multiplexor muxL. La señal muxL es una señal de control. Por tanto, es suficiente incluirla en la declaración de tipo "tp_contro_cam_cntl", la cual se especifica en el fichero controlador_pkg.vhd ubicado en el directorio tipos_constantes_pkg. Compruebe la declaración de la misma en el fichero.

Controlador de cache. Hay que modificar el fichero procedimientos_controlador_pkg.vhd, ubicado en el mismo directorio que el fichero controlador.vhd, para añadir un procedimiento mediante el cual se establezca el control del multiplexor (muxL). Adicionalmente hay que modificar el procedimiento que establece los valores por defecto. Posteriormente se utilizan estos procedimientos en el código que especifica el controlador.

El siguiente paso, para no tener que compilar todo el diseño es utilizar la herramienta Quartus para comprobar que el controlador se puede elaborar. Para ello, el directorio "controlador" incluye el directorio QUARTUS. Los ficheros *qsf y *.qpf contienen la especificación de un proyecto donde sólo se especifica el controlador.

Elaboración con Quartus del controlador de cache. La enumeración de los posibles estados del controlador de cache está especificada en el fichero controlador_pkg.vhd ubicado en el directorio "tipos_constantes_pkg" (Apéndice 2.4, Apéndice 2.10). Debe eliminarse el estado LEC de la enumeración de estados para que no se produzcan errores de compilación al elaborar el diseño con Quartus.

A.11.2Subproyecto 2

Modificación de la especificación estructural de la cache. Respecto al camino de datos, en la Figura 71 se muestra la ubicación del fichero "cache.vhd", después de copiarlo, que debe modificarse. Como multiplexor puede utilizar el codificado en el fichero mux_dat.vhd, que se encuentra en el directorio mux_dat del "proyecto_1".

Elaboración con Quartus del camino de datos. El primer paso una vez modificado el fichero cache.vhd, incluyendo el multiplexor muxL, es utilizar la herramienta Quartus para comprobar la elaboración RTL. En el directorio QUARTUS, asociado a este subproyecto (Figura 71), están incluidos los ficheros correspondientes con la especificación del subproyecto.

Jerarquía de memoria. Posteriormente se elabora la cache con el controlador y el sistema de memoria para construir la jerarquía de memoria. El directorio que contiene el ensamblado de estos componentes se denomina “ENSAMBLADO” y está ubicado en el primer nivel de directorios (Apéndice 2.4, Apéndice 2.10). Los directorios que incluyen los ficheros utilizados por Quartus y Modelsim se indican en la Figura 73. Los ficheros *.qsf y *.qpf del directorio QUARTUS especifican el proyecto Quartus.

Herramienta	Directorio	Directorio
Quartus		
	ENSAMBLADO	QUARTUS
Modelsim		PRUEBAS
		RESULTADOS
	UTILIDADES_pkg	impri_cache_memoria_pkg
		imprimir_traza_pkg

Figura 73 Proyecto_3. Directorios utilizados por la herramientas Quartus y Modelsim.

El directorio “ENSAMBLADO” incluye el directorio PRUEBAS (Figura 74). En este directorio se incluye, entre otros, el programa de prueba.

Directorio	Directorio	Comentario
PRUEBAS		
	prueba_ensamblado.vhd	Programa de pruebas
	procedimientos_peticiones_pkg.vhd	Procedimientos para emitir transacciones.
	formato_ventanas.do	Ficheros utilizados por Modelsim para formaterar la ventana temporal.
	wave.do	

Figura 74 Proyecto_3. Directorio PRUEBAS.

Simulación con Modelsim. En el procedimiento imprimir_traza que se suministra, utilizado en el programa de prueba, ya han sido comentadas todas las líneas de código relativas a los estados que no se utilizan. En la ventana temporal de Modelsim se muestran las mismas señales que en el proyecto previo (wave.do).

Programa de prueba. Este programa es una copia del programa suministrado en el “proyecto 1”. Los procedimientos que deben utilizarse para emitir peticiones son los mismos.

Apéndice 2.12: Proyecto 3. Documentación

La documentación es la misma que en el proyecto 1, excepto la modificación que se solicita. En la documentación generada sólo se incluye la especificación de la interface en los ficheros cuyo cuerpo debe diseñarse. Debido a ello, en este proyecto se observan multiples raices.

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 DOCUMENTACION/proyecto_3.

Apéndice 2.13: Proyecto 4. Organización de los ficheros

El árbol de directorios, desde el directorio raíz, de este proyecto es mimético al árbol del proyecto previo, excluyendo el directorio "reg_etiq". En los subdirectorios miméticos incluidos en este proyecto ("proyecto_4", Figura 75), sólo se incluyen los directorios que contienen ficheros modificados o que deben modificarse respecto al proyecto previo. También se incluyen ficheros, que aunque no deben modificarse, facilitan la independencia entre proyectos.

Proyecto_4	
|-- cache_con_interface_bus	
| |-- cache_con_interface_proc	
| | |---- cache	
| | | |---- *camino_de_datos*	
| | | |---- componentes	
| | | | |---- campo_etiquetas	
| | | | | |---- componentes	
| | | | | | |---- reg_etiq	Componente adicional
| | | | | |---- componentes_etiq_pkg	
| | | | | | |---- componentes_etiq_pkg.vhd	
| | | | | |---- ensamblado	
| | | | | |---- CODIGO	
| | | | | | |---- etiquetas.vhd	Campo etiquetas. Incluye un registro para el campo etiquetas
| | | |---- *controlador*	
| | | | |---- CODIGO	
| | | | | |---- controlador.vhd	Copiar cuerpo del "proyecto 3"
| | | | | |---- procedimientos_controlador_pkg.vhd	
| | | |---- *ensamblado*	
| | | |---- CODIGO	
| | | | |---- cache.vhd	Se utiliza la especificación RTL del proyecto 3. No debe copiarse
| | |---- interface_proc_cache	
| | |---- interfaces	
| | |---- CODIGO	
| | |---- interface_cache_proc.vhd	
| | |---- interface_proc_cache.vhd	Interface procesador/cache: específica de este proyecto
|-- ENSAMBLADO	
| |-- CODIGO	
| | |-- ensamblado.vhd	
| |-- PRUEBAS	
| | |-- procedimientos_peticiones_pkg.vhd	
| | |-- prueba_ensamblado.vhd	
|-- tipos_constantes_pkg	
| |-- controlador_pkg.vhd	Se añade la declaración del tipo tp_inter_proc_cache
|-- UTILIDADES_pkg	
|-- imprimir_traza_pkg	
|-- impri_traza_pkg.vhd	

Figura 75 Arbol de directorios modificados del "proyecto_4".

La ubicación de la interface procesador/cache específica de este proyecto se muestra en la Figura 75. También se incluye la interface cache/procesador, la cual es idéntica a la especificación RTL de los proyectos previos.

En la Figura 75 se muestra la ubicación del fichero etiquetas.vhd, el cual incluye la comparación de etiquetas. El registro que se utiliza está ubicado en un subdirectorio del directorio componentes y la especificación de este componente ha sido incluida en el fichero componentes_etiq_pkg.vhd, ubicado en el directorio componentes_etiq_pkg.

A.13.1Tabla de transiciones entre estados

En la Figura 76 se muestran las entradas en la tabla de transiciones entre estados.

		procesador y señales de estado						memoria	
		peticion				$\overline{\text{peticion}}$	resp. memoria		
		inicio	$\overline{\text{inicio}}$				si	no	
			lectura		escritura				
			si	no	si	no			
Estados	DES0								
	INI								
	ESCINI								
	DES								
	ESPL								
	ESB								
	ESPEA								
	ESCP								
	ESPEF								

Figura 76 Tabla de transiciones entre estados.

Apéndice 2.14: Proyecto 4. Simulación

A.14.1Controlador de cache

Un primer paso, para no tener que compilar todo el diseño, es utilizar la herramienta Quartus para comprobar que el controlador se puede elaborar. Para ello, el directorio “controlador” incluye el directorio QUARTUS. Los ficheros *qsf y *.qpf contienen la especificación de un proyecto donde sólo se especifica el controlador.

Elaboración con Quartus. La enumeración de los posibles estados del controlador de cache está especificada en el fichero controlador_pkg.vhd ubicado en el directorio “tipos_constantes_pkg” (Apéndice 2.4, Apéndice 2.13). A medida que vaya eliminando estados modifique la enumeración para que no se produzcan errores de compilación al elaborar el diseño con Quartus.

A.14.2Jerarquía de memoria

Posteriormente se elabora la cache con el controlador y el sistema de memoria para construir la jerarquía de memoria. El directorio que contiene el ensamblado de estos componentes se denomina “ENSAMBLADO” y está ubicado en el primer nivel de directorios (Apéndice 2.4, Apéndice 2.13). Los directorios que incluyen los ficheros utilizados por Quartus y Modelsim se indican en la Figura 77. Los ficheros *.qsf y *.qpf del directorio QUARTUS especifican el proyecto Quartus.

Herramienta	Directorio	Directorio
Quartus		
	ENSAMBLADO	QUARTUS
Modelsim		PRUEBAS
		RESULTADOS
	UTILIDADES_pkg	impri_cache_memoria_pkg
		imprimir_traza_pkg

Figura 77 Proyecto_4. Directorios utilizados por la herramientas Quartus y Modelsim.

El directorio “ENSAMBLADO” incluye el directorio PRUEBAS (Figura 78). En este directorio se incluye, entre otros, el programa de prueba.

Directorio	Directorio	Comentario
PRUEBAS		
	prueba_ensamblado.vhd	Programa de pruebas
	procedimientos_peticiones_pkg.vhd	Procedimientos para emitir transacciones.
	formato_ventanas.do	Ficheros utilizados por Modelsim para formaterar la ventana temporal.
	wave.do	

Figura 78 Proyecto_4. Directorio PRUEBAS.

Simulación con Modelsim. En el programa de prueba que se suministra ya han sido comentadas todas las líneas de código relativas a los estados que no se utilizan. En la ventana temporal de Modelsim se muestran las mismas señales que en el proyecto previo (wave.do).

Programa de prueba. En el programa de prueba hay una secuencia de accesos que puede utilizarse como punto de partida.

Apéndice 2.15: Proyecto 4. Documentación

La documentación es la misma que en el proyecto 1 excepto en la interface procesador/cache, el campo etiquetas y la modificación efectuada en el proyecto 3. En la documentación generada se muestra la interface y el campo etiquetas, pero no la modificación del proyecto_3. Debido a todo ello, en este proyecto se observan multiples raices.

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 DOCUMENTACION/proyecto_4.

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.

Apéndice 2.16: Retardos

En los “package” retardos_controlador_pkg.vhd, retardos_memorias_pkg.vhd y retardos_otros_pkg.vhd” se declaran los retardos de los elementos del diseño (Figura 79) .

Controlador	constant retardo_estado: time := 2 ns;
	constant retardo_logica_prx_estado: time := 2 ns;
	constant retardo_logica_salida: time := 3 ns;	

Elementos de almacenamiento	constant ret_etiq_esc: time := 2 ns;
	constant ret_etiq_leer: time := 3 ns
	constant ret_deco_etiq: time := 2 ns;
	
	constant ret_dat_esc: time := 2 ns;
	constant ret_dat_leer: time := 3 ns;
	constant ret_deco_dat: time := 2 ns;
	
	constant ret_estado_esc: time := 1 ns;
	constant ret_estado_leer: time := 1 ns;
	constant ret_deco_estado: time := 1 ns;
	
	constant ret_mem: time := 8 ns;
	constant ret_deco_mem: time := 3 ns;
	constant retcnt_mem: time := 1 ns;	

Registros de desacoplo	constant retREGDES: time := 2 ns;	
Interface procesador/cache	constant ret_reg_pet: time := 2 ns; constant ret_mux_pet: time := 2 ns;	
Registro en el módulo etiquetas	constant ret_reg_etiq: time := 2 ns;	Proyecto 4

Figura 79 Retardos de los componentes.