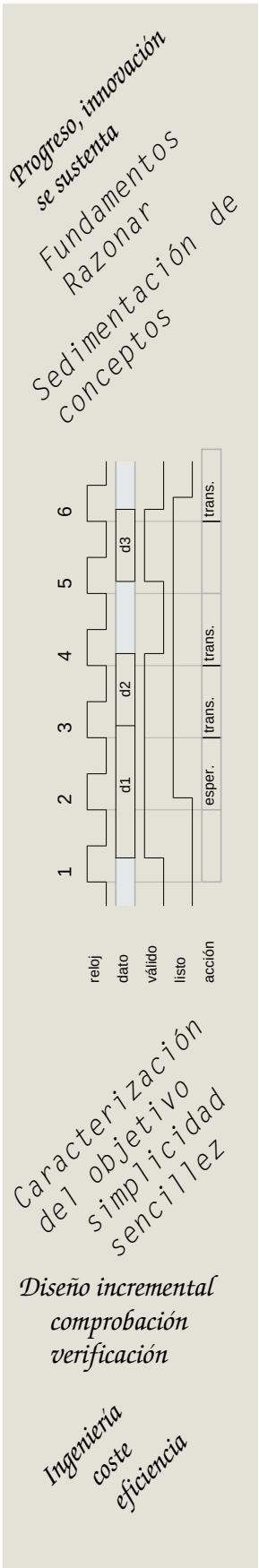
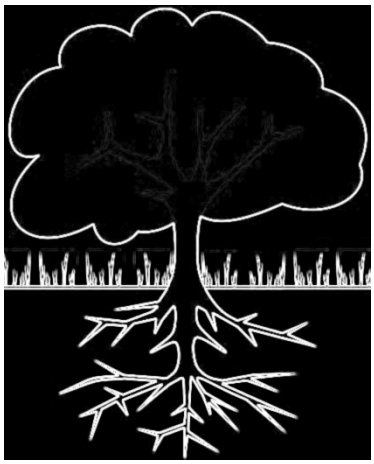


Documentación de prácticas Multiprocesadores (Grau - MP)



J.M. Llabería y
A. Olivé

Contenido

PRACTICA 1

PRACTICA 1

En esta práctica hay dos conjuntos de proyectos. En los dos conjuntos la operación que se efectúa es la misma: cálculo del máximo común divisor.

El primer conjunto de proyectos está completamente implementado en VHDL. El diseño final se plantea mediante diseños incrementales. El objetivo de este primer conjunto de proyectos es sedimentar conceptos. Para ello hay que analizar los diseños y comprenderlos.

En el segundo conjunto de proyectos el camino de datos es multi-ciclo, con el objetivo reducir el tiempo de ciclo e incrementar la productividad. Mediante este conjunto de proyectos se pretende la consolidación de los conceptos desarrollados en el primer conjunto de proyectos. El trabajo se centra en la especificación del camino de datos, el control asociado en VHDL y su comprobación posterior.

1 MÁXIMO COMÚN DIVISOR

El máximo común divisor (mcd, greatest common divisor, gcd) de dos o más números es el mayor número que divide a estos números de forma exacta (resto igual a cero).

El algoritmo de Ecuclides calcula el mcd utilizando divisiones y la propiedad de que el mcd de dos números, también es divisor de su diferencia.

En una división se distingue, el divisor (D), el dividendo (d), el cociente (c) y el resto (r).

$$D = c \times d + r$$

Por otro lado, tenemos que

$$r = D \mod d = D - d \times \left\lfloor \frac{D}{d} \right\rfloor$$

Entonces

$$\mathrm{mcd}(a,0) = a$$
$$\mathrm{mcd}(a,b) = \mathrm{mcd}(a, a \bmod b)$$

Cuando a y b son números positivos (naturales mayores que cero) el algoritmo puede escribirse de la siguiente forma

$$\mathrm{mcd}(a,0) = a$$
$$\mathrm{mcd}(a,b) = \mathrm{mcd}(a-b,b) \quad \text{si } (a > b)$$
$$\mathrm{mcd}(a,b) = \mathrm{mcd}(a,b-a) \quad \text{si } (a < b)$$

seudocódigo	seudocódigo

while (a /= b)
  if (a > b) then
  a := a - b;
  else
  b := b - a;
  end if;
end loop;
mcd := b;
	while (a /= b)
	  if (b > a) then
	  tmp := b;
	  b := a;
	  a := tmp;
	  end if;
	  a := a - b;
	end loop;
	mcd := a;

Figura 1 Seudocódigo de la operación máximo común divisor (mcd).

En la parte izquierda de la Figura 1 se muestra un seudocódigo donde se efectúan operaciones de resta, en función de una condición, con los operandos y resultado intercambiados. En la parte derecha se muestra un seudocódigo donde la operación de resta siempre utiliza los mismos operandos y resultado. En este segundo código se distingue una operación de intercambio, si es necesaria, antes de la operación de resta.

1.1 Camino de datos

Cuando se diseña un camino de datos para el código de la derecha se utiliza un sumador algebraico. Una implementación directa del algoritmo determina dos operaciones en secuencia en cada iteración: a) intercambio, si es necesario y b) resta. En la Figura 2 se muestra un camino de datos, que se utiliza durante varios ciclos para efectuar una operación mcd. En la parte izquierda de la Figura 2 se muestra el camino de datos para la operación y en la parte derecha se añade la lógica necesaria para inyectar datos. Por otro lado, se muestra el control de los registros mediante el permiso de escritura.

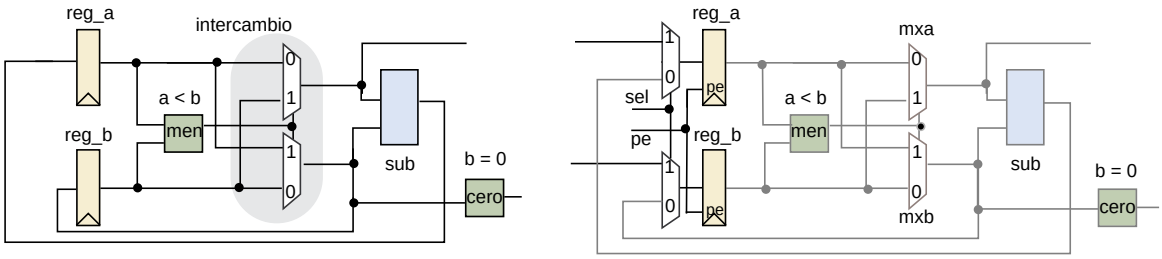


Figura 2 Camino de datos para una operación mcd. Se utiliza de forma repetida.

El tiempo de ciclo de este circuito es aproximadamente el retardo de dos sumadores. Notemos que en una comparación de menor también hay que propagar un acarreo.

1.2 Módulo mcd

En la Figura 3 se muestra un esquema simplificado del camino de datos del módulo mcd. En este esquema se distingue un camino de datos, fondo resaltado, y una unidad de control (autómata). El camino de datos se utiliza para encaminar la información a los elementos que evalúan y transferir información entre ellos. La unidad de control se encarga de controlar el flujo de información entre los elementos del camino de datos.

Entre el camino de datos y la unidad de control hay flujo de información en los dos sentidos. Las señales que fluyen desde el autómata al camino de datos se denominan de control y las señales que fluyen en sentido inverso se denominan de estado.

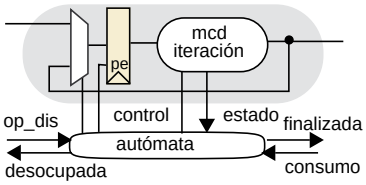


Figura 3 Esquema simplificado del camino de datos.

El módulo mcd tiene una latencia variable, la cual depende de los valores numéricos utilizados en el cálculo. En consecuencia, además del control de la operación, hay que utilizar un protocolo para conocer cuándo se puede iniciar un cálculo y conocer cuándo ha finalizado.

1.3 Autómata de control

El grafo de transiciones entre estados del controlador se muestra en la Figura 4. La mcd está libre (desocupada) en el estado ESP. Mientras no existan datos válidos (op_dis) el estado es ESP. Una vez los datos son válidos se pasa al estado CALC. El autómata permanece en el estado CALC mientras no se activa la señal cero. Cuando se activa la señal cero, el autómata pasa al estado HECHO. Este estado indica que el resultado del cálculo está disponible en la salida. Del estado HECHO se pasa al estado ESP, cuando se recibe la indicación de que ha sido consumido el resultado. Esta última fase se utiliza para preparar la mcd para una nueva operación. Esto es, la mcd queda desocupada.

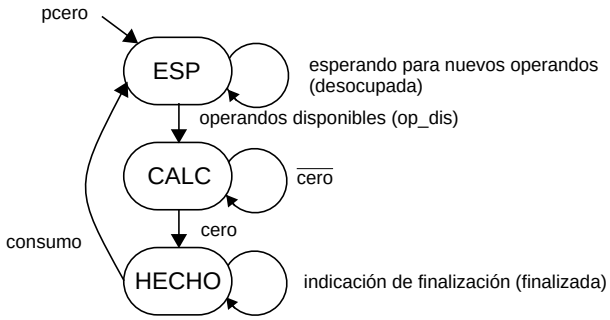


Figura 4 Autómata de control del camino de datos de la mcd.

En el Apéndice 1.4 se describe la utilización del camino de datos al iniciar una operación y mientras se efectúa el cálculo.

En el Apéndice 1.6 se describe en detalle, mediante una tabla, el autómata de control del camino de datos. También se muestra un diseño con puertas lógicas y registros.

Trabajo 1: Analice el flujo de información en el camino de datos y los estados en el autómata de control en los cálculos mcd(21, 12) y mcd(0, 8). Una forma de representar los cálculos es mediante una tabla (Figura 5). Los acrónimos mxa y mxb indican las salidas de los mutiplexores ubicados en las entradas del módulo sub (Figura 2). El multiplexor mxa es el ubicado en la parte superior en la figura.

ciclo	estado	reg_a	reg_b	menor	mxa	mxb	sub	cero

Figura 5 Tabla para representar ciclo a ciclo el funcionamiento.

1.4 Comunicación con un productor y un consumidor

En la Figura 6 se muestra un esquema simplificado del camino de datos del módulo mcd y la interconexión con un productor y un consumidor. Podemos considerar que el diseño tiene tres etapas. Dadas dos etapas, entre ellas hay registros de desacoplo interpuestos. Estos registros los consideraremos asociados a la etapa que alimentan y los denominaremos registros de entrada de la etapa.

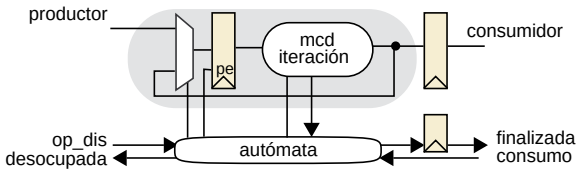


Figura 6 Esquema simplificado del camino de datos y el control de la mcd.

Respecto a la etapa mcd, el registro anterior al módulo mcd es el registro de desacoplo de entrada de datos. El registro de desacoplo asociado a la señal finalizada es interno al autómata.

El módulo mcd se conecta a un productor y a un consumidor mediante interfaces que utilizan un protocolo listo/válido, lo cual determina que las interfaces sean independientes de los actores que interconectan[1].

Canal de comunicación . Para describir un canal de comunicación entre dos etapas utilizaremos dos tipos de datos, que han sido definidos utilizando el constructor record de VHDL. Este constructor permite declarar objetos compuestos cuyos elementos pueden ser de distintos tipos. Un tipo de los declarados se utiliza para describir la información que fluye hacia etapas posteriores en la segmentación (posterior). El otro tipo declarado se utiliza para describir información que fluye hacia etapas previas en la segmentación (anteriores).

1. En el Apéndice 1.3 se efectúa una descripción del protocolo listo/válido.

tipo posterior	tipo anterior	interface productor-mcd	interface mcd-consumidor
type posterior_X[a] is record dat: st_dat; val: std_logic; end record posterior_X;	type anterior is record listo: std_logic; end record anterior;	pet_dv: posterior_X pet_l: anterior	resp_dv: posterior_X resp_l: anterior

a. X indica el número de elementos de tipo st_dat

Figura 7 Especificación en VHDL de las interfaces de la mcd con el productor y el consumidor.

En la Figura 8 se muestra el autómata de la Figura 4 utilizando los tipos de datos mostrados en la Figura 7.

Comunicación con un productor. Un análisis de la mcd determina que, cuando la mcd ha finalizado un cálculo, la señal resp_l.listo (activada por el consumidor), tarda un ciclo en propagarse al productor (pet_l.listo). Esta característica puede observarse en el grafo de transiciones entre estados de la Figura 8, en la transición del estado HECHO al estado ESP y notando que la señal pet_l.listo es de tipo Moore. Esto es, esta última señal sólo depende del estado. Por tanto, no existe un camino combinacional entre ambas señales.

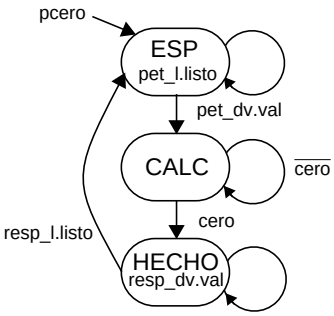


Figura 8 Autómata de la mcd con las señales utilizadas en las interfaces.

La productividad máxima de la mcd es 1/3 (una operación cada tres ciclos, mcd(0,7)).

Protocolos del productor y consumidor con la mcd. En la Figura 9 se muestra la utilización de las señales listo/válido por parte del productor, la unidad funcional (mcd, a la que también etiquetamos como UF) y el consumidor.

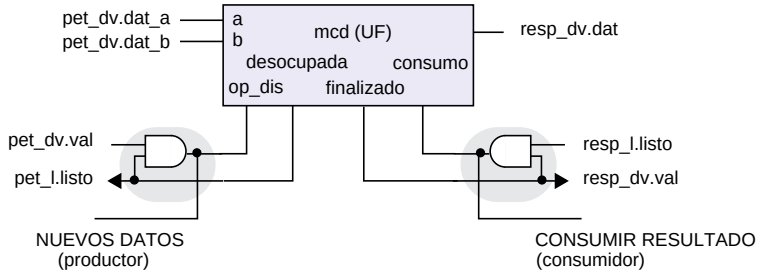


Figura 9 Utilización de las señales de las interfaces por los distintos actores.

En el productor y en el consumidor se está efectuando el bucle mostrado en la Figura 10; descrito en seudocódigo.

productor	consumidor	consumidor

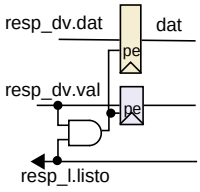
while (producción)
pet_dv.dato <= producir (tiempo procesado);
pet_dv.val <= '1';
wait until rising_edge(reloj) and pet_l.listo = '1';
pet_dv.val <= '0';
end loop;
	while (consumición)
	resp_l.listo <= '1';
	wait until rising_edge(reloj) and resp_dv.val = '1';
	dat <= resp_dv.dat;
	resp_l.listo <= '0';
	consumir (reg, tiempo procesado);
	end loop;
		

Figura 10 Seudocódigo de los actores productor y consumidor.

Evolución de las señales del protocolo de comunicación. En la Figura 11 se muestra un detalle de las señales listo/válido en las dos interfaces: a) productor y mcd y b) mcd y consumidor. En el ciclo 2 hay disponible un resultado en la salida de la mcd (resp_dv.val, UF, estado HECHO). En el mismo ciclo el consumidor está libre (resp_l.listo). En el inicio del ciclo 3, ambos interlocutores determinan que se produce una transferencia de información. Entonces, en el ciclo 3, el consumidor (C) empieza a consumir el dato. El productor (P) tiene datos válidos desde el ciclo 1. La mcd está libre en el ciclo 3 (pet_l.listo). En el inicio del ciclo 4 ambos interlocutores determinan que se produce una transferencia[2].

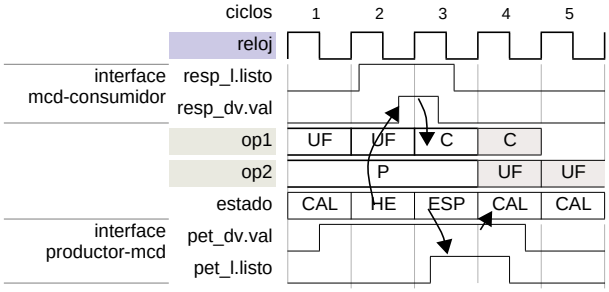


Figura 11 Detalle del funcionamiento de las interfaces.UF indica utilización de la mcd; una iteración. P indica productor y C indica consumidor.

Ejemplo. En al Figura 12 se muestra una secuencia de operaciones en la mcd. Un ciclo marcado como ". . ." representa varios ciclos, que depende de los datos de entrada. En la primera fila se muestra el estado de la mcd. En las siguientes filas se muestran cuatro operaciones. En la segunda de ellas, el consumidor tarda varios ciclos en procesar la información (C2). En paralelo se ha iniciado una operación (OP3), que debe esperar a que finalice el procesado del consumidor en la operación previa (estado HE durante varios ciclos). En las tres primeras operaciones suponemos que el productor produce una operación en un ciclo. Por tanto, está a la espera de que la mcd inicie el procesado de la misma. En la última operación (OP4) la mcd está a la espera de disponer de datos (estado ESP en varios ciclos).

	ciclos																												
operación	1	2	(3)	6	7	8	9	(4)	14	15	16	17	(2)	20	21	22	23	24	25	26	27	(4)	32	33	34
estado mcd	ESP	CAL	. . .	CAL	HE	ESP	CAL	. . .	CAL	HE	ESP	CAL	. . .	CAL	HE	HE	HE	ESP	ESP	ESP	CAL	. . .	CAL	HE	ESP
OP1	P1	UF				C1																			
OP2		P2					UF				C2														
OP3							P3					UF						C3							
OP4												P4									UF				C4

Figura 12 Secuencia de operaciones en la mcd. Entre paréntesis se indican los ciclos transcurridos cuando se indica ". . .". Un fondo oscurecido indica espera.

2. Notemos la diferencia de un ciclo en la activación de las señales listo, lo cual ha sido comentado previamente.

2 PROYECTOS. DISEÑO SECUENCIAL

Esta parte de la práctica incluye varios proyectos, los cuales están completamente diseñados y comprobados. La función de los trabajos que se solicitan es el análisis y compresión de los diseños.

Los proyectos se centran en el controlador de la mcd. También se introducen modificaciones del camino de datos y se analizan las interfaces con el productor y el consumidor.

El objetivo es diseñar una mcd con el control asociado, de forma que la latencia de cálculo sea igual a uno, en los casos factibles, y analizar las interfaces con el productor y el consumidor eliminando posibles caminos combinacionales en las mismas.

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

Los siguientes proyectos tienen como objetivo mejorar el rendimiento. Para ello, se eliminan estados en el controlador o se modifica su funcionalidad con el objetivo de reducir la latencia de cálculo[3].

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.

2.1 Proyecto 1: Diseño del controlador de la mcd

En el Apéndice 1.8 y Apéndice 1.9 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 1.10 se indica la ubicación de la documentación generada con Doxygen.

Trabajo 2: Analice la codificación del camino de datos y de los elementos que lo constituyen.

Trabajo 3: Analice la descripción funcional del control del camino de datos.

Trabajo 4: Analice el ensamblado de los componentes camino de datos y controlador.

3. En este proyecto también se modifica ligeramente el camino de datos.

Trabajo 5: Elabore el componente etapa_mcd con Quartus (directorio mcd_interface, Apéndice 1.8).

Programa de prueba. En el programa de prueba se utiliza un proceso ("process") productor y un proceso consumidor para comprobar el funcionamiento. En las interfaces se utiliza el protocolo listo/válido. En el programa que se suministra hay especificadas una secuencia de operaciones en el productor y una secuencia de extracciones en el consumidor (Apéndice 1.9).

Trabajo 6: Analice el programa de prueba. Describa los procedimientos "producir_datos" y "consumir_datos" mediante diagramas temporales. Para ello, utilice la señal reloj y sus flancos como referente. Céntrese en las señales relativas a los protocolos de las interfaces (productor-mcd y mcd-consumidor) y el valor de las entradas. Tenga en cuenta también el parámetro "timepoproducir".

Trabajo 7: Efectúe una simulación con Modelsim y analice los resultados. Analice en detalle las señales en las interfaces productor_mcd y consumidor_mcd (conjunto de señales 6 y 10 en la Figura 81 del Apéndice 1.9). Identifique en el diagrama temporal los instantes de comunicación en cada interface. Construya una tabla con los ciclos de cálculo (estado ESP hasta HECHO) de las operaciones iniciadas por el productor.

Trabajo 8: Modifique el programa de prueba utilizando operandos de entrada distintos. Efectúe una simulación con Modelsim y analice los resultados.

En los siguientes trabajos utilice el simulador Modelsim para corroborar el razonamiento o análisis.

Trabajo 9: Un ingeniero propone que en el estado ESP, independientemente de la señal op_dis, las señales pe y mxa (mxa se denomina sel en la Figura 2) siempre tomen el valor "1" (Apéndice 1.6). Analice la influencia de esta modificación. En primer lugar determine si la mcd calcula siempre un resultado válido. En segundo lugar indique si influye en el consumo energético.

Trabajo 10: Otro ingeniero propone que en el estado CALC, independientemente de las señales de control internas en la mcd, la señal pe siempre tome el valor "1" (Apéndice 1.6). Analice la influencia de esta modificación. En primer lugar determine si la mcd calcula siempre un resultado válido. En segundo lugar indique si influye en el consumo energético.

Trabajo 11: Un tercer ingeniero propone que en el estado HECHO, independientemente de la señal consumo, la señal pe siempre tome el valor “1” (Apéndice 1.6). Analice la influencia de esta modificación. En primer lugar determine si la mcd calcula siempre un resultado válido. En segundo lugar indique si influye en el consumo energético.

Trabajo 12: En el Apéndice 1.7 se muestra un diagrama temporal donde se utiliza el retardo de los componentes para determinar el tiempo de ciclo. Analice el diagrama (Figura 71 y parte izquierda de la Figura 72).

2.2 Proyecto 2: Eliminación del estado HECHO

En la Figura 12 podemos observar que entre dos operaciones consecutivas existen dos ciclos en las que no se utiliza el camino de datos. En los estados ESP y HECHO no se está efectuando cálculo. En el estado ESP se almacena la información del productor en los registros. En el estado HECHO se está esperando a que el consumidor extraiga el resultado. Una optimización es eliminar estos ciclos.

Autómata de control. En la Figura 13 se muestra el diagramas de transiciones entre estados, cuando se elimina el estado HECHO. Ahora, la indicación de finalización de la operación es una señal de tipo Mealy[4].

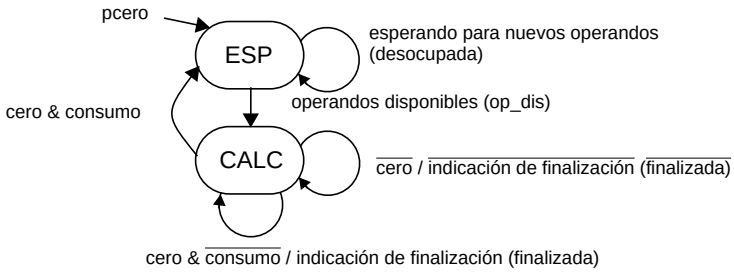


Figura 13 Autómata del camino de datos con dos estados.

En el Apéndice 1.4 se describe la utilización del camino de datos al iniciar una operación y mientras se efectúa el cálculo.

En el Apéndice 1.6 se describe en detalle, mediante una tabla, el autómata de control del camino de datos. También se muestra un diseño con puertas lógicas y registros.

Existe un camino combinacional entre la señal cero y la señal finalizada. Ahora bien, una de ellas es interna. No es de comunicación con las etapas predecesora o sucesora.

La productividad máxima de este diseño de la mcd, con el autómata de la Figura 13, es 1/2 (una operación cada dos ciclos, mcd(0,7)).

4. Estando en el estado CALC, la activación, en el mismo ciclo, de la señal finalización depende de la señal cero.

Ejemplo. En al Figura 14 se muestra la misma secuencia de operaciones en la mcd que en la Figura 12. La reducción en ciclos se observa en las primeras operaciones. Cuando el consumidor o el productor tardan más de un ciclo, el número de ciclos está determinado por estas acciones; cuando la mcd espera para que extraigan el dato o cuando la mcd está esperando por datos válidos.

	ciclos																								
operación	1	2	(2)	5	6	7	8	(4)	13	14	15	(2)	18	19	29	21	22	23	24	25	(4)	30	31
estado mcd	ESP	CAL	. . .	CAL	CAL	ESP	CAL	. . .	CAL	ESP	CAL	. . .	CAL	CAL	CAL	CAL	ESP	ESP	ESP	CAL	. . .	CAL	ESP
OP1	P1	UF				C1																	
OP2		P2				UF				C2													
OP3						P3				UF							C3						
OP4										P4									UF				C4

Figura 14 Secuencia de operaciones en la mcd utilizando el autómata de la Figura 13. Entre paréntesis se indican los ciclos transcurridos cuando se indica ". . .". Un fondo oscurecido indica espera.

En el Apéndice 1.11 y Apéndice 1.12 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 1.13 se indica la ubicación de la documentación genera con Doxygen.

Trabajo 13: Analice la descripción funcional del control del camino de datos.

Trabajo 14: Analice el ensamblado de los componentes camino de datos y controlador.

Trabajo 15: Elabore el componente etapa_mcd con Quartus (directorio mcd_interface, Apéndice 1.11).

Programa de prueba. En el programa de prueba se utiliza un proceso ("process") productor y un proceso consumidor para comprobar el funcionamiento. En las interfaces se utiliza el protocolo listo/válido. En el programa que se suministra hay especificada una secuencia de operaciones en el productor y una secuencia de extracciones en el consumidor (Apéndice 1.12).

Trabajo 16: Efectúe una simulación con Modelsim y analice los resultados. Añada en la ventana temporal de Modelsim las señales correspondientes a la salida de los registros que almacenan los resultados intermedios (reg_a y reg_b). Construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor.

Trabajo 17: Construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor en los Trabajo 7: y Trabajo 16:.

Trabajo 18: Modifique el programa de prueba utilizando operandos de entrada distintos. Efectúe una simulación con Modelsim y analice los resultados.

Trabajo 19: En el Apéndice 1.7 se muestra un diagrama temporal donde se utiliza el retardo de los componentes para determinar el tiempo de ciclo. Analice el diagrama (Figura 71 y parte derecha de la Figura 72).

2.3 Proyecto 3. Ciclo de cálculo en el estado ESP

El objetivo de este proyecto es reducir la latencia del cálculo en un ciclo. En estas condiciones, en una operación como mcd(0,9) el retardo será de un ciclo.

Camino de datos. En este proyecto y en el próximo utilizaremos una modificación del camino de datos[5]. En la Figura 15 se muestra el nuevo camino de datos. Cada registro de datos de la Figura 2 se mueve a cada una de las entradas de los multiplexores. Las señales de permiso de escritura de estos registros son independientes entre sí (pe, pec).

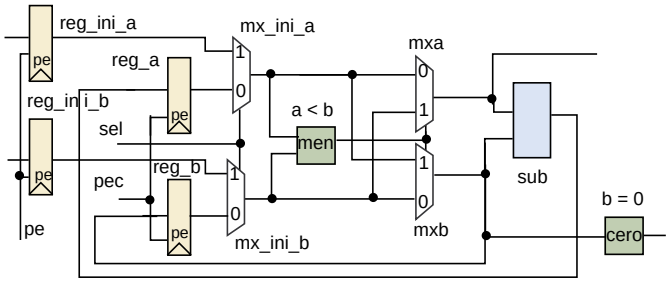


Figura 15 Camino de datos para que la mcd procese información en el primer ciclo.

Los registros asociados a la entrada “1” de los multiplexores (mx_ini_x) alimentan directamente a la lógica combinacional durante el primer ciclo de cálculo. Estos registros se denominan registros de desacoplo de la etapa y almacenan los datos que deben procesarse.

Los registros asociados a la entrada “0” de los multiplexores (mx_ini_x) se utilizan durante el cálculo.

En este contexto, la señal pe está asociada a los registros de desacoplo de entrada de la etapa y la señal pec a los registros utilizados en el camino de datos.

Autómata de control. En la Figura 16 se muestra el diagrama de transiciones entre estados para este diseño, donde en el estado ESP la mcd ya evalúa[6].

5. Esta modificación se efectúa para identificar de forma clara los registros de desacoplo de entrada de la etapa y los registros que se utilizan para almacenar cálculos intermedios.

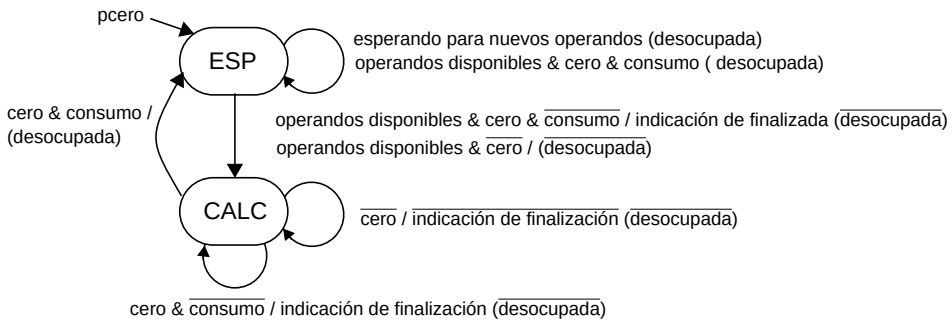


Figura 16 Autómata de control donde se evalúa desde el primer ciclo de ocupación de la mcd.

En el Apéndice 1.5 se describe la utilización del camino de datos al iniciar una operación y mientras se efectúa el cálculo.

En el Apéndice 1.6 se describe en detalle, mediante una tabla, el autómata de control del camino de datos. También se muestra un diseño con puertas lógicas y registros.

En este diseño la señal desocupada es una señal de tipo Mealy[7]. Existe un camino combinacional entre la señal consumo y la señal desocupada.

Trabajo 20: Analice el flujo de información en el camino de datos y los estados en el autómata de control en los cálculos mcd(21, 12), mcd (8,0) y mcd(0, 7). Una forma de representar los cálculos es mediante una tabla (Figura 17). Los acrónimos mxa y mxb indican las salidas de los multiplexores ubicados en las entradas del módulo sub (Figura 2).

ciclo	estado	reg_a	reg_b	menor	mux_ini_a	mux_ini_b	mxa	mxb	sub	cero

Figura 17 Proyecto 3. Tabla para representar ciclo a ciclo el funcionamiento.

Comunicación con el productor. En la Figura 18 se muestra un esquema simplificado del camino de datos y el control del módulo mcd. Notemos que los registros que almacenan los resultados intermedios han sido ubicados en la parte derecha de la figura, para ayudar a identificar los registros de desacoplo de la etapa. Observemos que la señal op_dis es entrada de un registro cuya salida alimenta al autómata de control de la mcd.

6. Aunque ya se evalúa seguimos utilizando el mismo acrónimo.
7. Notemos que la señal desocupada se activa en la transición del estado CALC al estado ESP y esta transición es función de la activación de las señales cero (interna) y consumo (externa).

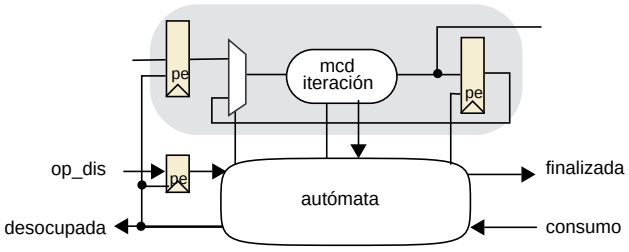


Figura 18 Esquema simplificado del camino de datos para procesar información en el primer ciclo.

La productividad de este diseño de la mcd (Figura 15), con el autómata de la Figura 16 llega a ser uno en función de los datos de entrada.

Ejemplos. En la Figura 19 se muestra una secuencia de operaciones en la mcd. La reducción en ciclos se observa en las primeras operaciones.

	ciclos																						
operación	1	2	3	(3)	7	8	(3)	12	13	(1)	15	16	17	18	19	20	21	22	(3)	26			
estado mcd		ESP	CAL	. . .	ESP	CAL	. . .	ESP	CAL	. . .	CAL	CAL	CAL	CAL	ESP	ESP	ESP	CAL	. . .	ESP			
OP1	P1	UF			C1																		
OP2		P2			UF			C2															
OP3					P3			UF							C3								
OP4								P4									UF			C4			

Figura 19 Secuencia de operaciones en la mcd utilizando el camino de datos de la Figura 15 y el autómata de la Figura 16. Entre paréntesis se indican los ciclos transcurridos cuando se indica ". . .". Un fondo oscurecido indica espera.

En la Figura 20 se muestra una secuencia de operaciones, tales como mcd(0, 7), cuya latencia es un ciclo.

	ciclos													
operación	1	2	. . .											
estado mcd		ESP	ESP	CAL	CAL	CAL	CAL	CAL	CAL	CAL	ESP	ESP	ESP	
OP1	P1	UF	C1											
OP2		P2	UF	C2										
OP3			P3								UF	C3		
OP4											P4	UF	C4	

Figura 20 Secuencia de operaciones en la mcd que tardan un ciclo en efectuarse.

En el Apéndice 1.14 y Apéndice 1.15 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 1.16 se indica la ubicación de la documentación genera con Doxygen.

Trabajo 21: Analice la codificación del camino de datos.

Trabajo 22: Analice la descripción funcional del control del camino de datos.

Trabajo 23: Analice el ensamblado de los componentes camino de datos y controlador.

Trabajo 24: Elabore el componente etapa_mcd con Quartus (directorio mcd_interface, Apéndice 1.14).

Programa de prueba. En el programa de prueba se utiliza un proceso ("process") productor y un proceso consumidor para comprobar el funcionamiento. En las interfaces se utiliza el protocolo listo/válido. En el programa que se suministra hay especificadas una secuencia de operaciones en el productor y una secuencia de extracciones en el consumidor (Apéndice 1.15).

Trabajo 25: Efectúe una simulación con Modelsim y analice los resultados. Analice en detalle las señales en las interfaces productor_mcd y consumidor_mcd (conjunto de señales 6 y 10 en la Figura 81 del Apéndice 1.9). Identifique en el diagrama temporal los instantes de comunicación en cada interface. Construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor.

Trabajo 26: Modifique el programa de prueba utilizando operandos de entrada distintos. Efectúe una simulación con Modelsim y analice los resultados.

Trabajo 27: Una forma de comprobar un autómata es recorrer todas las transiciones en el grafo de estados. En la Figura 21 se etiquetan las transiciones de la Figura 16 con una letra. En la secuencia de operaciones suministrada en el programa de prueba indique las transiciones que se recorren en cada una de las operaciones y entre ellas.

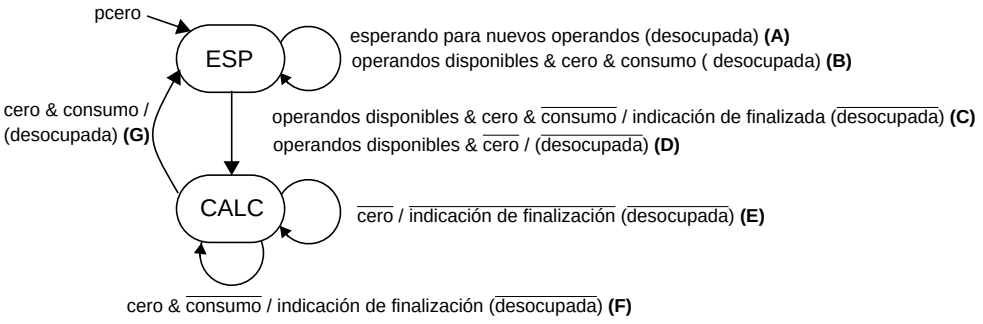


Figura 21 Etiquetado de las transiciones entre estados.

Trabajo 28: En el Apéndice 1.7 se muestra un diagrama temporal donde se utiliza el retardo de los componentes para determinar el tiempo de ciclo. Analice el diagrama (Figura 73 y la Figura 74).

2.4 Proyecto 4. Supresión de un camino combinacional en el protocolo

Para eliminar el camino combinacional entre las señales consumo y desocupada (resp_l.listo y pet_l.listo)[8] se interpone un registro entre ellas (Figura 22).

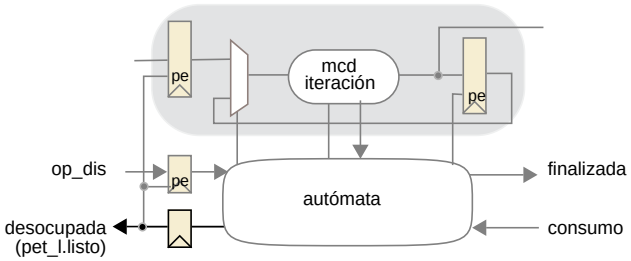


Figura 22 Propagación segmentada de la señal desocupada (listo).

En consecuencia, el productor conoce el valor de la señal (desocupada) un ciclo más tarde. Durante este ciclo, si la señal pet_l.listo está activada, y el productor emite una nueva operación, esta operación se perderá al no ser aceptada por la mcd. Esto se produce cuando la mcd tarda más de un ciclo en efectuar el cálculo. En la Figura 23 la segunda operación no se procesa.

		ciclos							
	operación	1	2	3	4	5	6	7	8
	estado mcd		ESP	CAL	CAL	ESP	ESP	ESP	ESP
	OP1	P	UF	UF	UF	C			
	OP2		P						
	OP3			P	P	P	UF	C	
	OP3						P	UF	C
desactiva	$\overline{\text{pet\_l.listo}}$			P					
activa	pet_l.listo					P			

Figura 23 Propagación segmentada de la señal listo. En las dos filas inferiores se indica la propagación de la señal pet_l.listo. En la casilla correspondiente se indica la etapa. Sólo se indica el primer ciclo de desactivación/activación de la señal.

Una alternativa, para que no se pierda la operación, es utilizar un registro de guarda en la entrada de la mcd[9].

En la Figura 24 se muestra la mcd con el control y el registro de guarda, junto con la propagación no combinacional de la señal, de la mcd, desocupada desde la mcd al productor. La salida del autómata ahora se denomina desocupada_reg y es entrada del registro RL que elimina el camino combinacional entre la señal consumo y desocupada de la Figura 18.

8. Apéndice 1.6.
9. Es suficiente con un registro de guarda, ya que la latencia de comunicación es un ciclo (Apéndice 1.3).

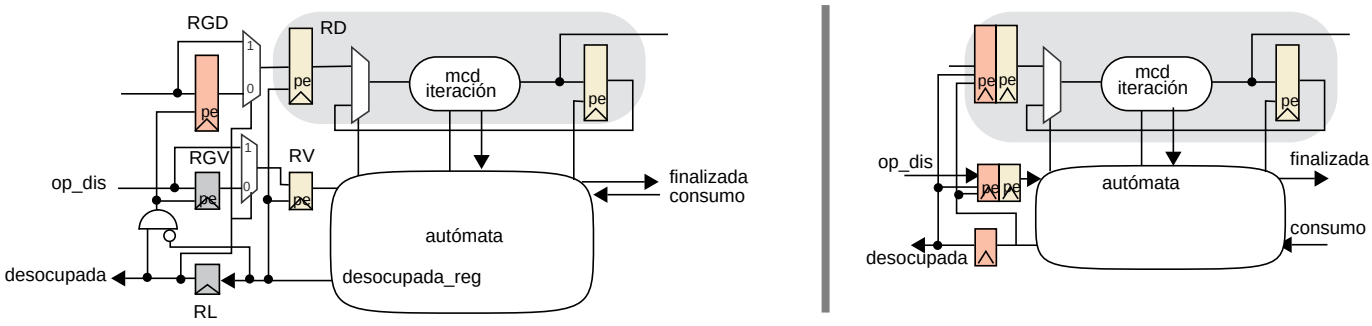


Figura 24 Esquema simplificado del módulo mcd con registro de guarda (RGD, RGV). parte izquierda: detalle del registro de desacoplo con guarda. Parte derecha: exclusión del detalle.

Cuando la mcd está libre el registro de guarda (RGD, RGV) actua como un cortocircuito. La información se almacena en el registro de la etapa (RD, RV). Ahora bien, cuando la mcd está ocupada y en el ciclo previo estaba libre, el registro de guarda almacena la información que llega. Cuando la mcd vuelve a estar libre, se transfiere la información del registro de guarda al registro de entrada de la etapa (RD, RV, Figura 25).

		ciclos								
	operación	1	2	3	4	5	6	7	8	9
	estado mcd		ESP	CAL	CAL	ESP	ESP	ESP	ESP	ESP
	OP1	P	UF	UF	UF	C				
	OP2		P	RG	RG	RG	UF	C		
	OP3			P	P	P	P	UF	C	
	OP3							P	UF	C
desactiva	$\overline{\text{pet\_l.listo}}$			P						
activa	pet_l.listo					P				

Figura 25 Propagación segmentada de la señal de bloqueo. Módulo mcd con registro de guarda (RG).

Al conjunto de elementos registro de guarda, registro de etapa y el control de ambos, lo denominaremos registro de desacoplo con guarda o registro de etapa con guarda.

Comunicación con un productor y un consumidor. En la Figura 26 se muestra un esquema simplificado del módulo mcd y la interconexión con un productor y un consumidor. Por la misma razón que en la interface productor-mcd, en la interface mcd-consumidor se utiliza un registro de desacoplo con guarda. Los registros de desacoplo con guardan se identifican como dos registros adyacentes de distinta tonalidad.

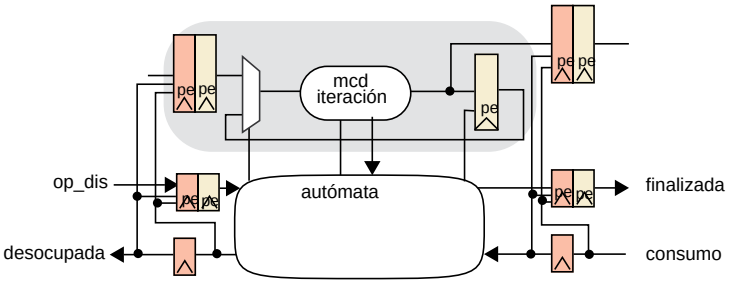


Figura 26 Esquema simplificado del camino de datos para procesar información en el primer ciclo con registros de guarda.

En el Apéndice 1.17 y Apéndice 1.18 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 1.19 se indica la ubicación de la documentación genera con Doxygen.

Trabajo 29: Analice el ensamblado de los componentes camino de datos y controlador y la interface.

Trabajo 30: Elabore el componente etapa_mcd con Quartus (directorio mcd_interface, Apéndice 1.17).

Programa de prueba. En el programa de prueba se utiliza un proceso ("process") productor y un proceso consumidor para comprobar el funcionamiento. En las interfaces se utiliza el protocolo listo/válido. Ahora bien, en esta prueba, en el consumidor no se utiliza un registro de guarda, ya que la señal consumo no se retarda un ciclo. En el programa que se suministra hay especificadas una secuencia de operaciones en el productor y una secuencia de extracciones en el consumidor (Apéndice 1.18).

Trabajo 31: Efectúe una simulación con Modelsim y analice los resultados. Construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor.

Trabajo 32: Modifique el programa de prueba utilizando operandos de entrada distintos. Efectúe una simulación con Modelsim y analice los resultados.

Trabajo 33: Diseñe un programa de prueba en el que se identifiquen situaciones donde se observe la utilización o no del registro de guarda. Razone la utilización o no del mismo en la secuencia de operaciones utilizada.

3 DISEÑO MULTICICLO DE LA MCD

En el diseño de la mcd de la Figura 2 podemos identificar 3 pasos en cada iteración: a) determinación de si el resultado de la resta será positivo, b) intercambio si el resultado no es positivo y c) operación de resta.

En el diseño de la Figura 2, los pasos descritos se efectúan en secuencia dentro del periodo de la señal de reloj.

3.1 Diseño multiciclo

El periodo de la señal de reloj puede reducirse utilizando un diseño multiciclo. En la Figura 27 se muestra un diseño con dos ciclos (etapas). Para ello se interponer registros de desacoplo entre la salida de los multiplexores y las entradas del módulo sub. Una iteración utiliza en secuencia las dos etapas. En el primer ciclo (etapa E1) se efectúa un intercambio, si es el caso. En el segundo ciclo (etapa E2) se efectúa una resta.

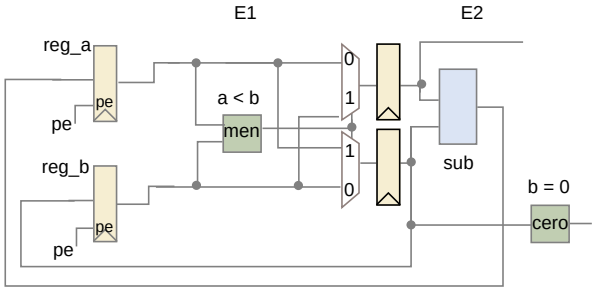


Figura 27 Módulo mcd multiciclo.

En estas condiciones, la latencia de una iteración son dos ciclos. Por otro lado, no se puede empezar la siguiente iteración hasta que ha finalizado la anterior, ya que una iteración necesita el resultado de la iteración previa.

3.2 Especulación

En el análisis de las iteraciones, cuando se efectúa una operación concreta de cálculo del mcd, observamos que en algunas de ellas no se efectúa la operación intercambio. Por tanto, podemos añadir caminos adicionales, en el circuito de la Figura 27, que tengan en consideración este caso. La idea es efectuar el cálculo de la resta mientras se determina si debería haberse efectuado un intercambio. En el caso de ser necesario el intercambio, se descarta el resultado de la resta[10] (Figura 28 y Figura 29).

	ciclos					
Modelo	1	2	3	4	5	6
A	E1	E2	. . .	E2	E1	E2
B	E1	E1	E2	. . .	E1	E2
	E2	E2			E2	

Figura 28 A: mcd multiciclo. B: con especulación. Un fondo tramado indica error de especulación.

10. Podemos decir que la resta se efectúa de forma especulativa. Para ello, estamos prediciendo que no es necesario un intercambio.

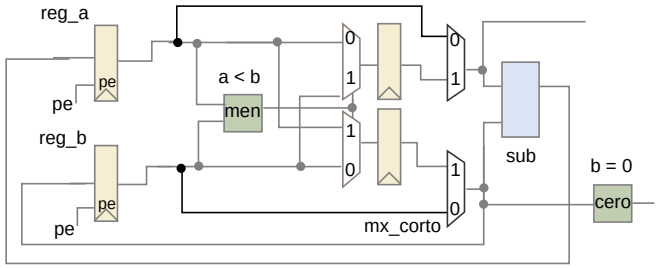


Figura 29 Camino de datos de la mcd multiciclo con cortocircuitos.

En la Figura 30 se muestra el cálculo de mcd(21, 12) utilizando la idea expuesta. Las casillas en blanco indican que se descarta el resultado de sub.

Modificación del camino de datos. El diseño de la Figura 29 puede mejorarse. Teniendo en cuenta el camino adicional (atajo) podemos pensar en utilizar sólo un par de registros.

La idea es almacenar el resultado del intercambio[11] en los registros de entrada, en lugar de almacenarlo en los registros interpuestos entre el intercambio y el módulo resta. Notemos que desde los registros de entrada se está alimentando al módulo "sub".

iter.	1	2		3		4	5	6	
ciclos	1	2	3	4	5	6	7	8	9
a	21	9	12	3	9	6	3	0	3
b	12	12	9	9	3	3	3	3	0
<	0	1	0	1	0	0	0	1	0
sub	9		3		6	3	0		3

Figura 30 Iteraciones en el cálculo de mcd(21, 12). Sub indica el resultado de la operación resta. El acrónimo "<" indica el resultado de la comparación de menor. La primera fila indica la iteración.

Para intercambiar el contenido de dos registros es suficiente el circuito que se muestra en la parte izquierda de la Figura 31. En la parte derecha se muestra la inclusión del módulo resta y detección de finalización (b = 0). Cuando hay que efectuar un intercambio, se selecciona la entrada cero del multiplexor mx_a y se activan los dos permisos de escritura. Cuando se efectúa la operación de resta se selecciona la entrada uno del multiplexor mx_a y se activa el permiso de escritura del registro etiquetado como a.

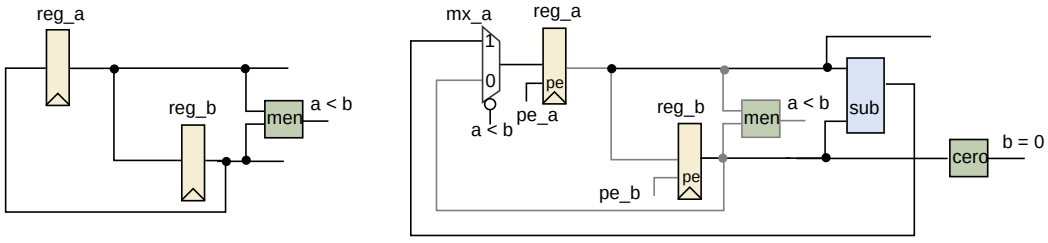


Figura 31 Parte izquierda: circuito para el intercambio entre dos registros. Camino de datos con la operación intercambio y la operación de cálculo.

Para iniciar un cálculo es necesario disponer de un camino que permita inyectar datos en el camino de datos. En la Figura 32 se muestra la inclusión de dos multiplexores para disponer de esta funcionalidad.

11. Cuando se cumple a < b.

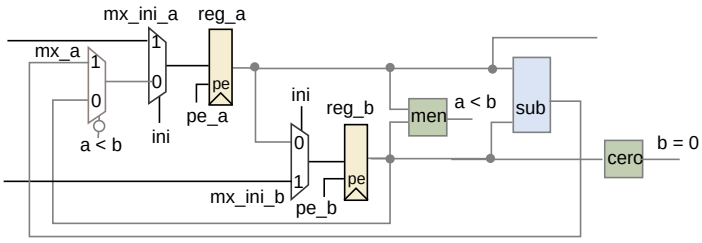


Figura 32 Módulo mcd multiciclo, con el camino para iniciar un cálculo con nuevos valores.

En la parte izquierda de la Figura 33 se muestra un seudocódigo donde la operación de resta siempre utiliza los mismos operandos y resultado. Este seudocódigo ha sido utilizado para el diseño secuencial. En la parte derecha se muestra el seudocódigo del diseño multiciclo.

secuencial	multiciclo

while (a /= b)
  if (b > a) then
  tmp := b;
  b := a;
  a := tmp;
  end if;
  a := a - b;
end loop;
mcd := a;
	while (a /= b)
	  if (b > a) then
	  tmp := b;
	  b := a;
	  a := tmp;
	  else
	  a := a - b;
	  end if;
	end loop;
	mcd := a;

Figura 33 Seudocódigo de la operación máximo común divisor (mcd). Secuencial: parte izquierda. Multicicl: parte derecha.

Trabajo 34: Analice el flujo de información en el camino de datos en los cálculos mcd(21, 12), mcd (0,8) y mcd(7, 0). Una forma de representar los cálculos es mediante una tabla (Figura 34). Los acrónimos mx_ini_a, mx_ini_b y mx_a indican las salidas en los mutiplexores ubicados en las entradas de los registros o en la entrada del multipexor mx_ini_a (Figura 32).

ciclo	estado	mx_a	mx_ini_a	mx_ini_b	reg_a	reg_b	menor	cero	sub

Figura 34 Diseño multiciclo. Tabla para representar ciclo a ciclo el funcionamiento.

4 PROYECTOS. DISEÑO MULTICICLO

En esta parte de la práctica debe codificarse en VHDL el camino de datos y diseñar y codificar el controlador.

Se utiliza la misma organización de directorios que en el conjunto de proyectos previo. En el Apéndice 1.20 se muestra la organización en directorios y ubicación de los ficheros de los 3 proyectos del diseño multiciclo.

4.1 Proyecto 1: Diseño del controlador de la mcd

Trabajo 35: Codifique una descripción estructural en VHDL del camino de datos de la Figura 32 (fichero Multiciclo/proyecto_1/mcd/camino/ensamblado/CODIGO/camino_mcd.vhd). Para ello, utilice los componentes disponibles del proyecto 1 del diseño secuencial (Secuencial/proyecto_1/mcd/camino/componentes/).

Trabajo 36: Utilice el directorio QUARTUS asociado para elaborar el camino de datos con Quartus.

Trabajo 37: Diseñe un autómata de control para el camino de datos de la Figura 32 utilizando 3 estados (ESP, CALC y HECHO). Construya la tabla de transiciones entre estados donde también se especifique la lógica de salida.

Trabajo 38: Codifique una descripción funcional en VHDL del controlador diseñado de 3 estados (fichero mcd/control/CODIGO/control.vhd). Utilice las funciones y procedimientos especificados en el fichero "proc_func_control_pkg.vhd, ubicado en el mismo directorio.

Trabajo 39: Utilice el directorio QUARTUS asociado para elaborar el controlador con Quartus.

Trabajo 40: El ensamblado de los componentes camino de datos y controlador está especificado en mcd/ensamblado/CODIGO/mcd.vhd. Analice el fichero.

Trabajo 41: Utilice el directorio QUARTUS asociado para elaborar el ensamblado del camino de datos y controlador con Quartus. Es necesario efectuar la tarea de creación de un proyecto Quartus.

Trabajo 42: El módulo etapa_mcd está especificado en el fichero mcd_interface/CODIGO/etapa_mcd.vhd. Analice el fichero.

Trabajo 43: Elabore el componente etapa_mcd con Quartus.

Programa de prueba. En el programa de prueba se utiliza un productor y un consumidor para comprobar el funcionamiento. En las interfaces se utiliza el protocolo listo/válido. En el programa que se suministra hay especificadas una secuencia de operaciones en el productor y una secuencia de extracciones en el consumidor.

Trabajo 44: Efectúe una simulación con Modelsim y analice los resultados. Construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor.

Trabajo 45: Compruebe de forma exhaustiva el funcionamiento del diseño.

Trabajo 46: Para una misma secuencia de cálculos, construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor en el proyecto 1 del diseño secuencial y este proyecto.

Trabajo 47: Utilice los retardos especificados en el Apéndice 1.7 para dibujar un diagrama temporal de retardos con el cual determinar el tiempo de ciclo.

4.2 Proyecto 2: Eliminación del estado HECHO

Trabajo 48: Diseñe un autómata de control para el camino de datos de la Figura 32 utilizando 2 estados (ESP y CALC). Construya la tabla de transiciones entre estados donde también se especifique la lógica de salida.

Trabajo 49: Codifique una descripción funcional en VHDL del controlador diseñado de 2 estados (fichero proyecto_2/mcd/control/CODIGO/control.vhd). Utilice las funciones y procedimientos especificados en el fichero “proc_func_control_pkg.vhd, ubicado en el mismo directorio.

Trabajo 50: Utilice el directorio QUARTUS asociado para elaborar el controlador con Quartus.

Trabajo 51: Elabore el componente etapa_mcd con Quartus (fichero proyecto_2/mcd_interface/CODIGO/etapa_mcd.vhd).

Programa de prueba. En el programa de prueba se utiliza un productor y un consumidor para comprobar el funcionamiento. En las interfaces se utiliza el protocolo listo/válido. En el programa que se suministra hay especificadas una secuencia de operaciones en el productor y una secuencia de extracciones en el consumidor.

Trabajo 52: Efectúe una simulación con Modelsim y analice los resultados. Construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor.

Trabajo 53: Compruebe de forma exhaustiva el funcionamiento del diseño.

Trabajo 54: Construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor en los Trabajo 44: y Trabajo 52:.

Trabajo 55: Diseñe con puertas lógicas las lógicas de próximo estado y de salida. Para codificar el estado se utiliza un vector de bits.

Trabajo 56: Utilice los retardos especificados en el Apéndice 1.7 para dibujar un diagrama temporal de retardos con el cual determinar el tiempo de ciclo.

4.3 Proyecto 3. Ciclo de cálculo en el estado ESP

En este proyecto utilizaremos una modificación del camino de datos.

En la Figura 35 se muestra el nuevo camino de datos. Cada registro de datos (Figura 32) se mueve a cada una de las entradas de los multiplexores (mx_ini_X). Las señales de permiso de escritura de estos registros son independientes entre si (pe, pe_a, pe_b). También son independientes entre sí las señales de selección de los multiplexores mx_ini_a y mx_ini_b.

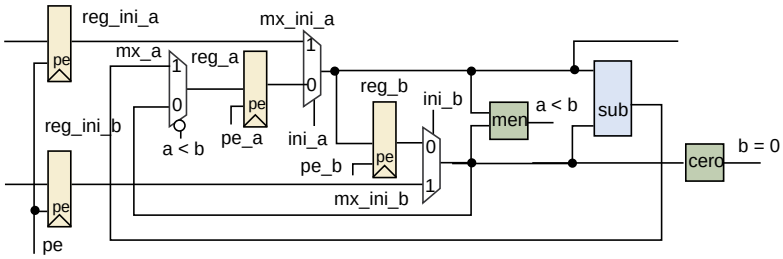


Figura 35 Módulo mcd multiciclo. Camino de datos para que la mcd procese información en el primer ciclo.

Los registros asociados a la entrada “1” de los multiplexores alimentan directamente a la lógica combinacional durante el primer ciclo. Estos registros se denominan registros de desacoplo de la etapa y almacenan los datos que deben procesarse.

Análisis. Observe que hasta que no se produce un intercambio debe de seleccionarse la entrada ‘1’ del multiplexor mx_ini_b. Durante este lapso de tiempo no es relevante la actualización o no del registro reg_b. Una vez ha sido efectuado un intercambio, la entrada que debe seleccionarse del multiplexor mx_ini_b es la entrada ‘0’.

Una vez ha sido efectuado un intercambio, los registros asociados a la entrada “0” de los multiplexores se utilizan durante el cálculo.

En este contexto, la señal pe está asociado a los registros de desacoplo de entrada de la etapa y las señales pe_a y pe_b están asociadas a los registros utilizados en el camino de datos.

Trabajo 57: Analice con detalle la forma de utilizar el camino de datos para las operaciones intercambio y calcular, tanto al iniciar un operación mcd como en régimen permanente. Este análisis es necesario para diseñar el autómata de control. Por ejemplo, es de interés analizar la utilización del camino de datos de la Figura 35 en los cálculos mcd(21, 12), mcd(8,0), mcd(0,7) y mcd(8, 1). En primer lugar debe analizarse el inicio de la operación y el siguiente ciclo. Posteriormente se analiza el régimen permanente.

Trabajo 58: Codifique una descripción estructural en VHDL del camino de datos de la Figura 35, excluyendo los registros reg_ini_X, (fichero proyecto_3/mcd/camino/ensamblado/CODIGO/camino_mcd.vhd). Para ello utilice los componentes disponibles del proyecto 1 del diseño secuencial (Secuencial/proyecto_1/mcd/camino/componentes/).

Trabajo 59: Utilice el directorio QUARTUS asociado para elaborar el camino de datos con Quartus.

Para el diseño del autómata de control es de utilidad utilizar 3 estados (ESP, CALC, CALCINI). Mediante los dos últimos se discierne sobre la selección en el multiplexor mux_ini_b y la actualización del registro reg_b (Figura 35, operación intercambio).

Trabajo 60: Diseñe un autómata de control para el camino de datos de la Figura 35 utilizando 3 estados (ESP, CALC, CALCINI), donde en el estado ESP ya se inicia el cálculo. Construya la tabla de transiciones entre estados donde también se especifique la lógica de salida.

Trabajo 61: Codifique una descripción funcional en VHDL del controlador diseñado de 3 estados (fichero proyecto_3/mcd/control/CODIGO/control.vhd). Utilice las funciones y procedimientos especificados en el fichero “proc_func_control_pkg.vhd, ubicado en el mismo directorio.

Trabajo 62: Utilice el directorio QUARTUS asociado para elaborar el controlador con Quartus.

Trabajo 63: El ensamblado de los componentes camino de datos y controlador está especificado en mcd/ensamblado/CODIGO/mcd.vhd. Analice el fichero.

Trabajo 64: Utilice el directorio QUARTUS asociado para elaborar el ensamblado del camino de datos y controlador con Quartus. Es necesario efectuar la tarea de creación de un proyecto Quartus.

Trabajo 65: El módulo etapa_mcd está especificado en el fichero mcd_interface/CODIGO/etapa_mcd.vhd. Este fichero incluye los registros reg_ini_X. Analice el fichero.

Trabajo 66: Elabore el componente etapa_mcd con Quartus.

Programa de prueba. En el programa de prueba se utiliza un productor y un consumidor para comprobar el funcionamiento. En las interfaces se utiliza el protocolo listo/válido. En el programa que se suministra hay especificadas una secuencia de operaciones en el productor y una secuencia de extracciones en el consumidor.

Trabajo 67: Efectúe una simulación con Modelsim y analice los resultados. Construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor.

Trabajo 68: Compruebe de forma exhaustiva el funcionamiento del diseño.

Trabajo 69: Para una misma secuencia de cálculos, construya una tabla con los ciclos de cálculo de las operaciones iniciadas por el productor en proyecto 1 del diseño multiciclo y este proyecto.

Trabajo 70: Utilice los retardos especificados en el Apéndice 1.7 para dibujar un diagrama temporal de retardos con el cual determinar el tiempo de ciclo.

Trabajo 71: En el diseño de la mcd de las Figura 32 y Figura 35 se utilizan componentes distintos para efectuar la operación de resta y determinar si “a < b”[12]. Proponga una implementación de la operación resta, que además de determinar el resultado, indique si se produce desbordamiento (a < b). El módulo diseñado tendría, además de la salida s, la salida “a<b”. Nota: razone sobre efectuar la operación de resta extendiendo el rango.

12. En el código VHDL que se suministra también se utilizan dos componentes.

Apéndice 1.1: Circuitos secuenciales

Los circuitos digitales complejos tienen incluido el concepto de estado. En otras palabras, las salidas dependen de valores previos de las entradas y de los valores actuales de las entradas. El valor previo de las entradas se almacena implícitamente, de forma codificada, en lo que se denomina estado. Estos sistemas se denominan secuenciales en contraposición a los sistemas combinacionales.

En la Figura 36 se muestra un modelo genérico de un sistema secuencial. El estado del sistema se almacena en registros. La salida de los registros es el estado actual y la entrada de los registros es el estado futuro.

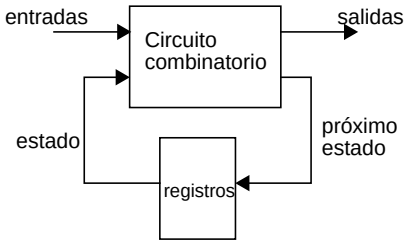


Figura 36 Modelo genérico de un sistema secuencial.

Los circuitos secuenciales se pueden dividir en dos clases básicas: asíncronos y asíncronos. Un circuito síncrono actualiza el estado cuando cambia una señal de reloj. Un circuito asíncrono cambia el estado en el mismo instante en que se modifica el estado futuro. Nosotros nos centraremos en los circuitos asíncronos. Un circuito síncrono facilita la verificación y la comprobación del funcionamiento.

En un circuito combinatorio se puede producir lo que se denomina riesgo o condición de carrera. Esto es, las salidas puede mostrar valores espurios debido al retardos distintos en los caminos desde las entradas a las salidas. Un circuito secuencial debe diseñarse para que si tales riesgos se producen sean ignorados.

Para asegurar que un circuito secuencial es capaz de ignorar los riesgos que pueden producirse en un circuito combinatorio se utiliza un reloj para sincronizar los datos. Cuando la señal de reloj tiene el valor cero se ignora cualquier riesgo. Este comportamiento se implementa mediante puertas AND. Se efectúa la operación AND de cada salida con la señal de reloj. En estas condiciones el sistema aún es susceptible a riesgos cuando la señal de reloj toma el valor uno. Por tanto, es usual utilizar registros que sólo muestrean la señal de entrada en cambios de la señal de reloj. Un cambio en el valor de la señal de reloj (flanco) tiene una duración mucho menor que un semiperiodo de la señal de reloj. Por tanto, el dato sólo debe ser

estable durante la duración del flanco. Por ello, los registros de estado de un circuito secuencial síncrono son registros que actúan en un flanco de la señal de reloj.

Circuito síncrono. Un circuito síncrono utiliza los registros como elementos de memorización y todos los registros están controlados por un único reloj.

El diagrama básico de un circuito síncrono se muestra en la Figura 37. El elemento de memorización (registro) conocido como registro de estado es un conjunto de registros, sincronizados por una única señal de reloj. La salida del registro es la señal de estado que representa el estado interno del sistema. La lógica próximo_estado es un circuito combinacional que determina el próximo estado. La lógica salida es otro circuito combinacional.

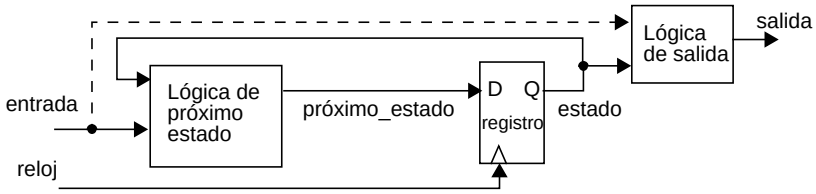


Figura 37 Modelo esquemático de un circuito secuencial síncrono.

En la Figura 37 las señales de salida dependen del estado y de las señales de entrada (línea de trazos). Este tipo de circuito síncrono se denomina autómata de Mealy. Si las señales de salida sólo dependen del estado, el circuito síncrono se denomina autómata de Moore. En este apéndice nos centraremos en estos últimos.

El funcionamiento del circuito es el siguiente.

- En el flanco ascendente de la señal de reloj, el valor de la señal próximo_estado (D) se muestrea y se propaga a la salida (Q), siendo ahora el nuevo estado. El valor se almacena en el elemento de memorización (registro) y durante todo el periodo de la señal de reloj no se modifica. Representa el estado del sistema.
- Las lógicas combinacionales próximo_estado y salida determinan respectivamente el próximo estado y la salida.
- En el siguiente flanco ascendente de la señal de reloj se repite el proceso.

En la Figura 38 se muestra un diagrama temporal de las relaciones de dependencia entre los retardos de los componentes de un circuito secuencial síncrono. La magnitud de los retardos que se representa es un ejemplo.

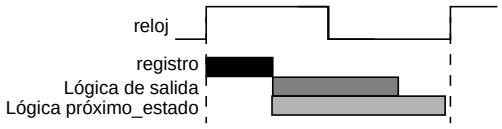


Figura 38 Diagrama temporal, en un periodo de la señal de reloj, de un circuito síncrono. Relaciones de dependencia entre los retardos.

Dado un circuito síncrono podemos distinguir, de forma informal, varios tipos: a) circuito secuencial regular, b) circuito secuencial aleatorio y c) circuito secuencial combinado. La diferencia entre el primer tipo y el segundo es la complejidad de las transiciones entre estados y la complejidad de las lógicas combinacionales. En un circuito secuencial regular la representación binaria de los estados usualmente tiene una interpretación. Ejemplos del primer tipo son contadores y registros de desplazamiento. Los circuitos secuenciales del segundo tipo se denominan máquinas de estados finitos. En el tercer tipo se incluyen los circuitos que constan de elementos del primer tipo y del segundo tipo. En este tercer caso, la máquina de estados finitos se utiliza para controlar el circuito secuencial regular (autómata subordinado).

A.1.1 Diseños simples

Registro con puesta a cero asíncrona. En la tabla de la Figura 39 se representa el funcionamiento del circuito que se muestra en la parte derecha de la misma figura. El símbolo Q* indica el futuro valor de la salida y el símbolo Q indica el valor actual de la salida. La señal PE (permiso de escritura) sólo se tiene en cuenta en el flanco ascendente de la señal de reloj. Esto significa que la señal PE está sincronizada con la señal de reloj. En el flanco ascendente de la señal de reloj se muestrean las señales PE y D. Si PE = 0 se mantiene el valor en el elemento de memorización. En caso contrario, cuando PE = 1, el funcionamiento es el de un registro. Esto es, la entrada se propaga a la salida.

reloj	PE	Q*
0	-	Q
1	-	Q
flanco	0	Q
flanco	1	D

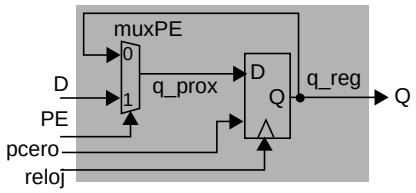


Figura 39 Registro con permiso de escritura síncrono.

En la parte derecha de la Figura 39 se ha mostrado el diagrama de un registro con permiso de escritura. Observemos que podemos asimilar elementos de este diagrama con elementos mostrados en la

Figura 37. La lógica del próximo_estado es el multiplexor y no existe lógica para determinar la señal de salida, ya que es directamente la salida del registro.

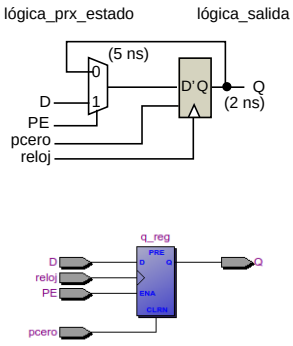
En la Figura 40 se muestra una descripción VHDL del registro con permiso de escritura síncrono y puesta a cero asíncrona. Se distinguen tres partes. El proceso denominado reg modela el elemento de memorización denominado registro. El elemento mux se modela mediante una sentencia de asignación de señal de forma condicional[13]. La sentencia de asignación de señal modela la lógica de salida. La lógica de salida es un cable que conecta la salida del registro con el puerto de salida.

Registro con puesta a cero asíncrona

```
library IEEE;
use IEEE.std_logic_1164.all;

entity regen is
port (reloj, pcero, PE, D: in  std_logic;
      Q: out                   std_logic );
end regen;

architecture trespart of regen is
signal q_reg: std_logic;
signal q_prox: std_logic;
begin
--Estado
reg: process (reloj, pcero)
begin
     if (pcero ='1') then
        q_reg <= '0' after 2 ns;
     elsif (reloj'event and reloj = '1') then
        q_reg <= q_prox after 2 ns;
     end if ;
end process;
-- Lógica de próximo estado
     q_prox <= D after 5 ns when PE = '1' else q_reg after 5 ns;
-- Lógica de salida
     q <= q_reg;
end trespart;
```



Elaboración efectuada por QUARTUS

Figura 40 Descripción VHDL de un registro con puesta a cero asíncrona, distinguiendo los elementos combinacionales y secuenciales.

En la descripción VHDL de la Figura 40 el registro tiene un retardo de actualización de 2 ns y la lógica de próximo estado tiene un retardo de 5 ns.

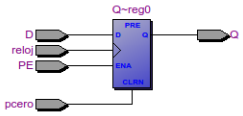
La descripción VHDL de la Figura 40 sigue el modelo de la Figura 37. En este modelo se pueden identificar de forma clara los elementos combinacionales y los elementos secuenciales. Esto permite comprobar y verificar el funcionamiento de los componentes de forma

13.Esta sentencia de asignación de señal de forma condicional puede sustituirse por un proceso que describa un circuito combinacional.

aislada. En la Figura 41 se muestra una descripción VHDL del mismo tipo de registro donde se entremezclan los distintos elementos, lo cual dificulta la comprobación y verificación.

Registro con permiso de escritura síncrono

```
library IEEE;
use IEEE.std_logic_1164.all;
entity regen is
port (  reloj, pcero, PE, D: in  std_logic;
        Q: out                   std_logic );
end regen;
architecture compor of regen is
begin
regen: Process (reloj, pcero)
begin
        if (pcero ='1') then
          Q <= '0' after 2 ns;
        elsif (reloj'event and reloj = '1') then
          if PE = '1' then
             Q <= D after 5 ns;
          end if ;
        end if ;
end process ;
end compor;
```



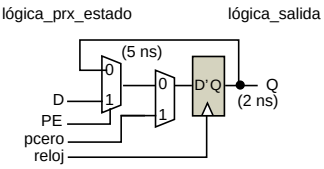
Elaboración efectuada por QUARTUS

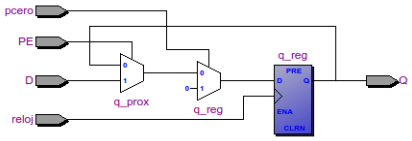
Figura 41 Descripción VHDL de un registro con permiso de escritura utilizando un único proceso.

Puesta a cero síncrona. La puesta a cero puede efectuarse de forma síncrona. En la Figura 42 se muestra la descripción VHDL.

Registro con puesta a cero síncrono

```
library IEEE;
use IEEE.std_logic_1164.all;
entity regen is
port (  reloj, pcero, PE, D: in  std_logic;
        Q: out                   std_logic );
end regen;
architecture trespart of regen is
signal q_reg: std_logic;
signal q_prox: std_logic;
begin
--Estado
reg: process (reloj, pcero) begin
        if (reloj'event and reloj = '1') then
          if (pcero ='1') then
            q_reg <= '0' after 2 ns;
          else
            q_reg <= q_prox after 2 ns;
          end if ;
        end if ;
end process;
-- Lógica de próximo estado
        q_prox <= D after 5 ns when PE = '1' else
        q_reg after 5 ns;
-- Lógica de salida
        q <= q_reg;
end trespart;
```





Elaboración efectuada por QUARTUS

Figura 42 Descripción VHDL de un registro con puesta a cero síncrona, distinguiendo los elementos combinacionales y secuenciales.

Descripción estructural de un registro con permiso de escritura y puesta a cero síncrona. En la Figura 43 se muestra la especificación del circuito de la Figura 42 utilizando un registro especificado en VHDL, sin permiso de escritura ni puesta a cero. En la descripción queda explícito el elemento que almacena el estado.

Descripción estructural. Registro con puesta a cero síncrona

```
library IEEE;
use IEEE.std_logic_1164.all;

entity regen is
port (    reloj, pcero, PE, D: in   std_logic;
          Q: out                    std_logic );
end regen;

architecture estruc of regen is
component registro is
  generic (retardo: time := 2 ns);
  port ( reloj, D: in std_logic;
       Q: out std_logic);
end component;
signal q_reg: std_logic;
signal q_prox: std_logic;
begin
--Estado
reg:      registro generic map (retardo => 2 ns)
          port map (reloj => reloj, D => q_prox, Q => q_reg);
-- Lógica de próximo estado
          q_prox <= '0' when pcero = '1' else D when PE = '1' else q_reg;
-- Lógica de salida
          Q <= q_reg;
end estruc;
```

registro

```
library IEEE;
use IEEE.std_logic_1164.all;

entity registro is
  generic (retardo: time := 2 ns);
  port ( reloj, D: in std_logic;
       Q: out std_logic);
end;
architecture compor of registro is
begin
process (reloj)
begin
  if (reloj'event and reloj = '1') then
     Q <= D after retardo;
  end if ;
end process;
end compor;
```

Figura 43 Descripción estructural en VHDL de un registro con puesta a cero síncrona.

En la Figura 43 se muestra la elaboración efectuada por QUARTUS.

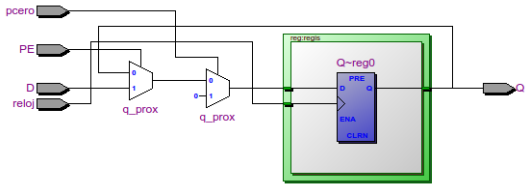


Figura 44 Elaboración efectuada por Quartus a partir de la especificación de la Figura 43.

Contador binario. Un contador binario pasa por una secuencia de estados cuya codificación en binario se puede interpretar como números naturales. Por ejemplo un contador binario de 2 bits repite indefinidamente la secuencia: 00, 01, 10, 11.

Un contador binario tiene un registro que almacena n bits y su salida se interpreta como un número natural codificado en base 2. El contador incrementa el contenido del registro en cada ciclo de la señal de reloj; contando desde 0 hasta 2^n - 1. La cuenta se repite indefinidamente.

En la Figura 45 se muestra un esquema de un contador binario. Comparando esta figura con la Figura 37 podemos identificar que la lógica próximo_estado es un incrementador, el cual calcula el nuevo valor del próximo estado. No existe lógica para determinar la señal de salida, ya que es directamente la salida del registro.

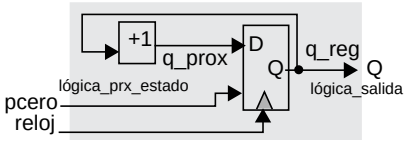


Figura 45 Contador binario.

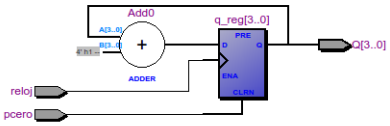
En la Figura 46 se muestra una descripción VHDL del contador binario donde se distinguen tres partes. El proceso denominado reg modela el elemento de memorización denominado registro. Después del proceso, la primera sentencia de asignación de señal modela el incrementador unidad. La última sentencia de asignación de señal modela la lógica de salida. La lógica de salida es un cable que conecta la salida del registro con el puerto de salida.

Contador binario

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;

entity contador is
port ( reloj, pcero : in  std_logic;
       Q: out           std_logic_vector (3 downto 0) );
end contador;

architecture trespart of contador is
signal q_reg: std_logic_vector (3 downto 0);
signal q_prox: std_logic_vector (3 downto 0);
begin
--Estado
reg: Process (reloj, pcero)
begin
       if (pcero ='1') then
          q_reg <= (others => '0') after 2 ns;
       elsif (reloj'event and reloj = '1') then
          q_reg <= q_prox after 2 ns;
       end if ;
end process ;
-- Lógica de próximo estado
       q_prox <= std_logic_vector(unsigned(q_reg) +1) after 8 ns ;
-- Lógica de salida
       q <= q_reg;
end trespart;
```



Elaboración efectuada por QUARTUS

Figura 46 Descripción VHDL de un contador binario.

Contador módulo. En la Figura 47 se muestra el esquema de circuito de un contador módulo 7. La lógica próximo estado es un incrementador, un multiplexor y un comparador.

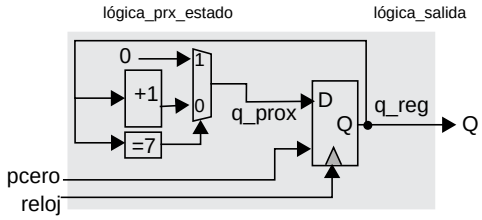


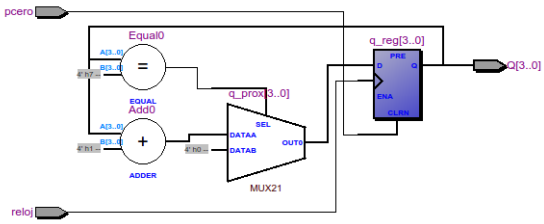
Figura 47 Contador módulo.

En la Figura 48 se muestra la descripción VHDL siguiendo el modelo de la Figura 37 donde se identifican tres partes: a) estado, b) lógica de próximo estado y c) lógica de salida. El límite del contador se describe mediante una constante. Para describir la lógica de próximo estado se utiliza un incrementador, un comparador y un multiplexor. La condición que se evalúa es el valor del contador (comparación).

Contador módulo

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;
entity cntrmod is
port (  reloj, pcero : in    std_logic;
        Q: out               std_logic_vector (3 downto 0) );
end cntrmod;
architecture trespart of cntrmod is
constant modu: integer := 7;
signal q_reg: std_logic_vector (3 downto 0);
signal q_prox: std_logic_vector (3 downto 0);
begin
--Estado
reg:Process (reloj, pcero)
begin
        if (pcero ='1') then
           q_reg <= (others => '0') after 2 ns;
        elsif (reloj'event and reloj = '1') then
           q_reg <= q_prox after 2 ns;
        end if ;

end process ;
-- Lógica de próximo estado
        q_prox <= (others => '0') after 10 ns when to_integer( unsigned(q_reg)) = modu else std_logic_vector(unsigned(q_reg) +1) after 10 ns ;
-- Lógica de salida
        q <= q_reg;
end trespart;
```



Elaboración efectuada por QUARTUS

Figura 48 Descripción VHDL de un contador módulo.

Apéndice 1.2: Esquemas de descripción en VHDL de circuitos secuenciales

En este apéndice se presentan los esqueletos de una especificación VHDL de los autómatas de Moore y Mealy.

A.2.1 Autómata de Moore

- próximo estado = función (entradas, estado)
- salida = función (estado)

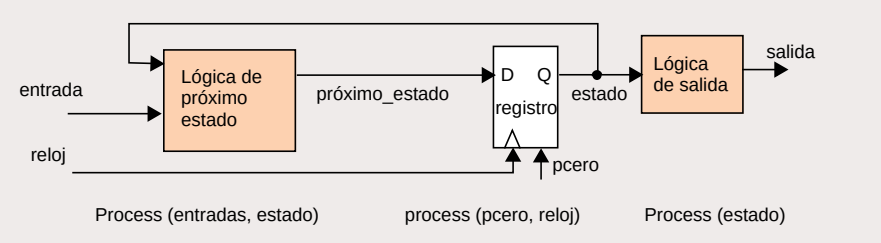


Figura 49 Esquema lógico de un autómata de Moore.

A.2.2 Autómata de Mealy

- próximo estado = función (entradas, estado)
- salida = función (estado, entradas)

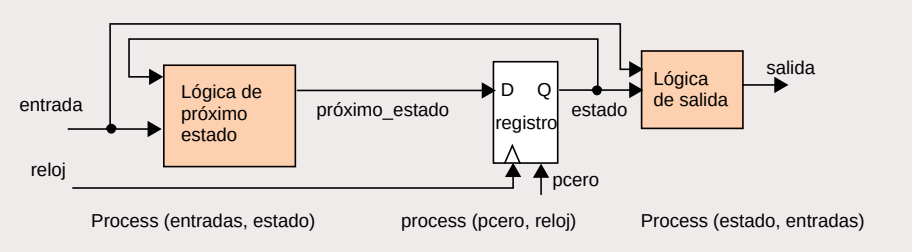


Figura 50 Esquema lógico de un autómata de Mealy.

La diferencia entre un autómata de Moore (Figura 49) y un autómata de Mealy (Figura 50) es que la lógica de salida, en el caso del autómata de Mealy, tiene como entradas las entradas del sistema.

Seguidamente se describen los esqueletos de dos alternativas para especificar en VHDL los autómatas de Moore y de Mealy. En el primer caso se utilizan tres procesos. Un proceso para cada uno de los elementos de las Figura 49 y Figura 50. Posteriormente se describe el esqueleto con dos procesos. En este caso la lógica de próximo estado y la lógica de salida se describen en el mismo proceso.

A.2.3 Especificación en VHDL utilizando 3 procesos.

Las posiciones de las diferencias se marcan con un trazo en la parte izquierda de la especificación (Figura 51).

Automáta de Moore

```
library ieee;
use ieee.std_logic_1164.all;
entity patron_Moore is
    port (reloj, pcero: in std_logic;
       entrada: in std_logic;
       salida: out std_logic);
end;
architecture func of patron_Moore is
    -- definir el tipo para el estado
    type tipo_estado is (S0, S1, . . . );
-- declarar estado y próximo estado
    signal estado, prx_estado: tipo_estado;
begin
-- registro de estado
    process (reloj, pcero)
    begin
       if pcero = '1' then
--determinar estado en la puesta a cero
            estado <= S0;
       elsif reloj'event and reloj = '1' then
                estado <= prx_estado;
       end if;
    end process;
-- lógica de próximo estado
-- dependiente de la entrada y del estado
    process (estado, entrada)
    begin
       case estado is
          when S0  => if (entrada = ...) then
                prx_estado <= S1;
             else
                prx_estado <= S0;
             end if;
          . . .
          when others => prx_estado  ... ;
       end case;
    end process;
-- lógica de salida - también se pueden utilizar sentencias de asignación de señal concurrentes
-- lógica de salida: la diferencia entre Moore and Mealy está en la lógica de salida, Mealy depende de la entrada
    process (estado)
    begin
       case estado is
          when S0 =>
             salida < = . . .
          ...
          when . .
       end case;
    end process ;
end;
```

Autómata de Mealy

```
library ieee;
use ieee.std_logic_1164.all;
entity patron_Mealy is
   port (reloj, pcero: in std_logic;
      entrada: in std_logic;
      salida: out std_logic);
end;
architecture func of patron_Mealy is

   type tipo_estado is (S0, S1, . . .);

   signal estado, prx_estado: tipo_estado;
begin
-- registro de estado
   process (reloj, pcero)
   begin
      if pcero = '1' then

         estado <= S0;
      elsif reloj'event and reloj = '1' then
         estado <= prx_estado;
      end if;
   end process;
-- lógica de próximo estado

   process (estado, entrada)
   begin
      case estado is
         when S0 => if (entrada = ...)  then
               prx_estado <= S1;
            else
               prx_estado <= S0;
            end if;
         . . .
         when others => prx_estado  ... ;
      end case;
   end process;
-- lógica de salida - también se pueden utilizar sentencias de asignación de señal concurrentes
-- lógica de salida: la diferencia entre Moore and Mealy está en la lógica de salida, Mealy depende de la entrada
   process (estado, entrada)
   begin
      case estado is
         when S0 =>
            if (entrada = ...) then
         ...
         when . . .
      end case;
   end process ;
end;
```

Figura 51 Especificación de autómatas utilizando tres procesos.

A.2.4 Especificación en VHDL utilizando 2 procesos

Las posiciones de las diferencias se marcan con un trazo en la parte izquierda de la especificación (Figura 52).

Automáta de Moore

```
library ieee;
use ieee.std_logic_1164.all;
entity patron_Moore is
    port (reloj, pcero: in std_logic;
        entrada: in std_logic;
        salida: out std_logic);
end;
architecture func of patron_Moore is
    -- definir el tipo para el estado
    type tipo_estado is (S0, S1, . . .);
-- declarar estado y próximo estado
    signal estado, prx_estado: tipo_estado;
begin
-- registro de estado
    process (reloj, pcero)
    begin
        if pcero = '1' then
--determinar estado en la puesta a cero
            estado <= S0;
        elsif reloj'event and reloj = '1' then
                estado <= prx_estado;
        end if;
    end process;
-- lógica de próximo estado, lógica de salida
-- dependiente de la entrada y del estado
    process (estado, entrada)
    begin
        case estado is
          when S0  =>
          --salida de tipo Moore antes de una senten-
          cia condicional
              salida <= ... ;
              if (entrada = ...) then
                  prx_estado <= S1;
              else
                  prx_estado <= S0;

              end if;
          . . .
          when others => nextstate  ... ;
              salida <= ... ;
              prx_estado  ... ;
        end case;
    end process;
end;
```

Autómata de Mealy

```
library ieee;
use ieee.std_logic_1164.all;
entity patron_Mealy is
    port (reloj, pcero: in std_logic;
        entrada: in std_logic;
        salida: out std_logic);
end;
architecture func of patron_Mealy is

    type tipo_estado is (S0, S1, . . .);

    signal estado, prx_estado: tipo_estado;
begin
-- registro de estado
    process (reloj, pcero)
    begin
        if pcero = '1' then

            estado <= S0;
        elsif reloj'event and reloj = '1' then
            estado <= prx_estado;
        end if;
    end process;
-- lógica de próximo estado, lógica de salida

    process (estado, entrada)
    begin
        case estado is
            when S0 =>
            --salida de tipo Mealy después de una sentencia
            condicional
                if (entrada = ...)  then
                  prx_estado <= S1;
                  salida <= ... ;
                else
                  prx_estado <= S0;
                  salida <= ... ;
                end if;
            . . .
            when others =>
              salida <= ... ;
              prx_estado  ... ;
        end case;
    end process;
end;
```

Figura 52 Especificación de autómatas utilizando dos procesos.

Resumen. Especificación de un autómata utilizando tres procesos.

- proceso estado: registros del autómata
- (el mismo proceso para Moore y Mealy)
- proceso próximo estado
- (el mismo proceso para Moore y Mealy)
- Lógica de salida: pueden utilizarse sentencias concurrente o procesos.

  Autómata de Moore: dependen sólo del estado.

  Autómata de Mealy: dependen del estado y de las entradas.

Resumen. Especificación de un autómata utilizando dos procesos.

- En un autómata de Moore las salidas se asignan antes de la sentencia condicional que evalúa entradas.
- En un autómata de Mealy las salidas se asignan después de la sentencia condicional que evalúa entradas.

Nota. Cuando se especifican retardos es recomendable utilizar variables en el cuerpo de un “process”. Antes de la sentencia “end” se asignan las variables a las señales especificando el retardo.

Apéndice 1.3: Protocolo en un canal de comunicación listo/válido

En la Figura 53 se muestra un canal de comunicación entre un emisor y un receptor. En el canal se distingue una línea de comunicación de datos entre un emisor y un receptor. Adicionalmente existen dos líneas de control en sentidos opuestos: a) válido y b) listo. El protocolo se utiliza para controlar el flujo de información entre el emisor y el receptor.

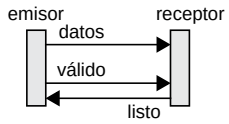


Figura 53 Canal de comunicación.

La línea de control denominada listo, del receptor al emisor, indica al emisor que el receptor acepta información. La línea de control denominada válido, del emisor al receptor, indica al receptor que hay información válida en el canal de comunicación.

En el canal se distinguen tres estados, que se muestran en la Figura 54.

El emisor tiene un comportamiento persistente. Esto es, mantiene el dato mientras no reconoce que ha sido capturado por el receptor.

Estado	válido	listo
Transferencia	1	1
Sin transferencia	0	{0, 1}
Espera	1	0

Figura 54 Estados en un canal de comunicación.

La transferencia o captura de un dato se produce en el flanco ascendente de la señal de reloj. Todos los componentes utilizan el mismo reloj.

El emisor activa la señal válido independientemente de la señal listo. Una vez ha activado la señal válido, no debe modificarla (persistente) hasta que se ha producido la transferencia (listo = 1 y flanco de reloj, Figura 55).

El receptor captura la información cuando la señal válido está activa y también está activada la señal listo. La captura se produce en el flanco de la señal de reloj.

Este protocolo permite que el emisor y el receptor bloqueen su “función” durante un tiempo arbitrario utilizando las señales válido y listo.

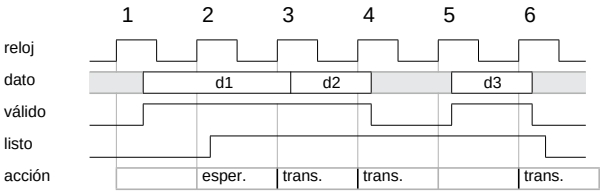


Figura 55 Ejemplo del protocolo de comunicación.

En el flanco del ciclo 2, de la Figura 55, la señal listo está desactivada y no se produce transferencia de información. El comportamiento del emisor es persistente. Por tanto, en el flanco del ciclo 3 aún mantiene activada la señal válido. En este flanco, la señal listo está activada. En consecuencia, se produce la transferencia de información (d1).

En el flanco del ciclo 4 se produce la misma situación que en el ciclo 3 y se efectúa otra transferencia de información (d2). En el flanco del ciclo 5, el emisor no tiene activada la señal válido. En consecuencia, no hay transferencia de información. En el flanco del ciclo 6, las señales válido y listo están activadas y se produce transferencia de información (d3).

Este tipo de interface elimina la necesidad de efectuar hipótesis temporales en el diseño y elimina lógica combinacional entre módulos. Es un ejemplo de los que se denomina interface independiente de la latencia. Los detalles de implementación de un módulo quedan ocultos (caja negra) por la interface. Esto permite la sustitución del módulo actual por otra implementación del mismo con otras temporalidades. Por otro lado, este tipo de interface aporta una conexión, compuesta por 2 cables, para conectar módulos.

Cuando se diseñe con este tipo de interface hay que estar seguro que las señales válido y listo en una interface no están conectadas entre ellas de forma combinacional, ya sea en el emisor o en el receptor. Esto es, dependan, en el mismo ciclo, una de la otra.

A.3.1 Propagación combinacional de la señal listo

En la Figura 56 se muestra un esquema donde la señal listo se propaga de forma combinacional. La señal val (v) se propaga de forma segmentada con los datos. Cuando la información almacenada en el registro de desacoplo de entrada no es válida se absorbe el bloque (no listo). En la parte derecha de la figura se muestra un diseño que no utiliza el permiso de escritura en el registro de validez.

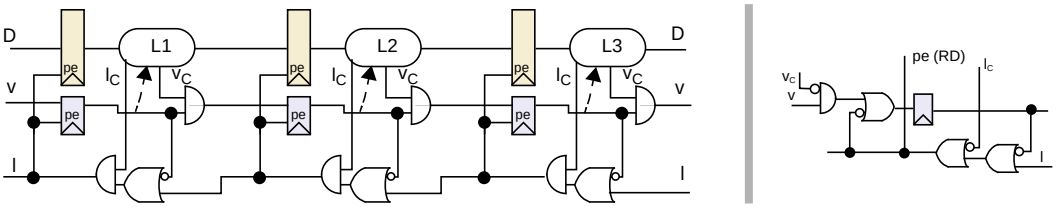


Figura 56 Protocolo listo/válido. Propagación combinacional de la señal listo (l).

En la Figura 57 se muestra un diagrama temporal. Para representar una situación de bloqueo (no listo), cuando es una etapa interna, se utiliza una nueva fila. En la fila previa se indica la propagación de información inválida por las etapas utilizando el acrónimo “Xn”, siendo X el acrónimo de la etapa. Cuando no existe bloqueo se muestra la propagación de la operación por las etapas. La etapa E determina 2 ciclos de bloque (ciclos 5 y 6). La etapa B determina un ciclo de bloqueo (ciclo 4). Un ciclo de bloqueo, producido por la etapa E, es absorbido en la etapa C al propagarse la operación op3 a la etapa C en el ciclo 6.

ciclos	1	2	3	4	5	6	7	8	9	10	11	12	13
op1	A	B	C	D	E	E	E						
op2		A	B	C	D	D	D	E					
op3			A	B	Cn								
					B	C	C	D	E				
op4				A	A	B	B	C	D	E			
op5						A	A	B	C	D	E		
op6								A	B	C	D	E	
op7									A	B	C	D	E

Figura 57 Propagación combinacional de la señal listo. Absorción del bloqueo en una etapa con información inválida.

A.3.2 Propagación segmentada de la señal listo

En la Figura 58 se muestra un esquema donde la señal listo se propaga de forma segmentada.

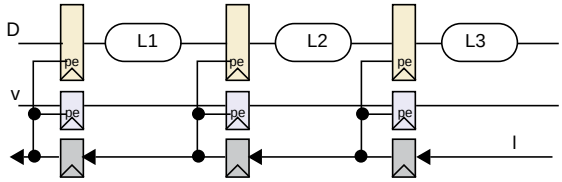


Figura 58 Propagación segmentada de la señal listo (l). Sin registro de guarda.

En la Figura 59 se muestra un diagrama temporal de funcionamiento del circuito de la Figura 58. Cuando en una fila no se muestra la ocupación de las etapas posteriores, la información ha sido modificada por una nueva operación. Esto es, la información se ha perdido. En las dos últimas filas de la figura se muestra la propagación de la señal de bloqueo (no listo) por las etapas (bloqueo y desbloqueo, sólo se muestra la primera activación de bloqueo).

ciclos	1	2	3	4	5	6	7	8	9	10	11	12	13	14
op1	A	B	C	D	E									
op2		A	B	C	Dn	En								
					C	Dn	En							
						C	Dn	En						
							C	Dn	En					
								C	D	E				
op3			A	B										
op4				A	B	B	B	B	B	C	D	E		
op5					A									
op6						A	A	A	A	A	B	C	D	E
bloq.					B	A								
desb.									B	A				

Figura 59 Propagación segmentada de la señal de bloqueo (no listo). Bloqueo se indica como bloq. y desbloqueo como desb.

En el ciclo 4 se activa la señal de bloqueo en la etapa C y perdura durante 4 ciclos. La señal de bloqueo llega a la etapa A en el ciclo 6. Durante los siguientes 4 ciclos sigue llegando una señal de bloqueo. La señal de desbloqueo (bloqueo desactivado), es activada desde la etapa C y llega en el ciclo 9 a la etapa B. En el ciclo 5 se pierde la información ubicada en la etapa B. En el ciclo 6 se pierde la información ubicada en la etapa A.

Notemos que el bloqueo en una etapa no es observado en las etapas que lo alimentan hasta el siguiente ciclo. Entonces, para gestionar la propagación segmentada de la señal de bloqueo es necesario disponer de capacidad de almacenamiento adicional en la salida de un módulo o en las entradas de los módulos conectados a su salida. El registro adicional, o registro de guarda (RG), se utiliza durante la propagación de la señal de bloqueo. En la Figura 60 se muestra un registro de desacoplo con registro de guarda. Notemos que el camino de la señal de validez y datos (D, v) se muestran de forma conjunta.

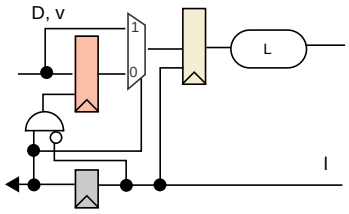


Figura 60 Registro de desacoplo con almacenamiento de guarda.

En la Figura 61 se muestra un diagrama temporal. Para representar la utilización de un registro de guarda se utiliza como sufijo de la etapa el acrónimo G.

ciclos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
op1	A	B	C	D	E	F	G									
		A	B	C	Dn	En	Fn	Gn								
					C	Dn	En	Fn	Gn							
						C	Dn	En	Fn	Gn						
							C	Dn	En	Fn	Gn					
								C	D	E	F	G				
op2			A	B	CG	CG	CG	CG	C	D	E	F	G			
op3				A	B	B	B	B	B	C	D	E	F	G		
op4					A	BG	BG	BG	BG	B	C	D	E	F	G	
op5						A	A	A	A	A	B	C	D	E	F	G
bloq.					B	A										
desb.									B	A						

Figura 61 Propagación segmentada de la señal de bloqueo (no listo). El sufijo G indica registro de guarda en la etapa correspondiente. Bloqueo se indica como bloq. y desbloqueo como desb.

En el ciclo 4 la etapa C activa la señal de bloqueo. En el ciclo 5 la información transmitida desde la etapa B se almacena en el registro de guarda de la etapa C. La etapa B almacena la información trans-

mitida desde la etapa A. En el ciclo 6, la información en el registro de guarda de la etapa C y la información en la etapa B se mantienen. La información transmitida desde las etapa A se almacena en el registro de guarda de la etapa B.

En el ciclo 8 desaparece la situación de bloqueo de la etapa C. En el ciclo 9 la información en CG ha sido transferida al registro de entrada de la etapa C. Un hecho similar se produce en la etapa B en el ciclo 10.

Apéndice 1.4: Diseño secuencial: autómata de control de tres o dos estados

A.4.1 Utilización del camino de datos en cada estado

En la Figura 62 se muestran los componentes del camino de datos utilizados en cada uno de los tres estados del autómata de control.

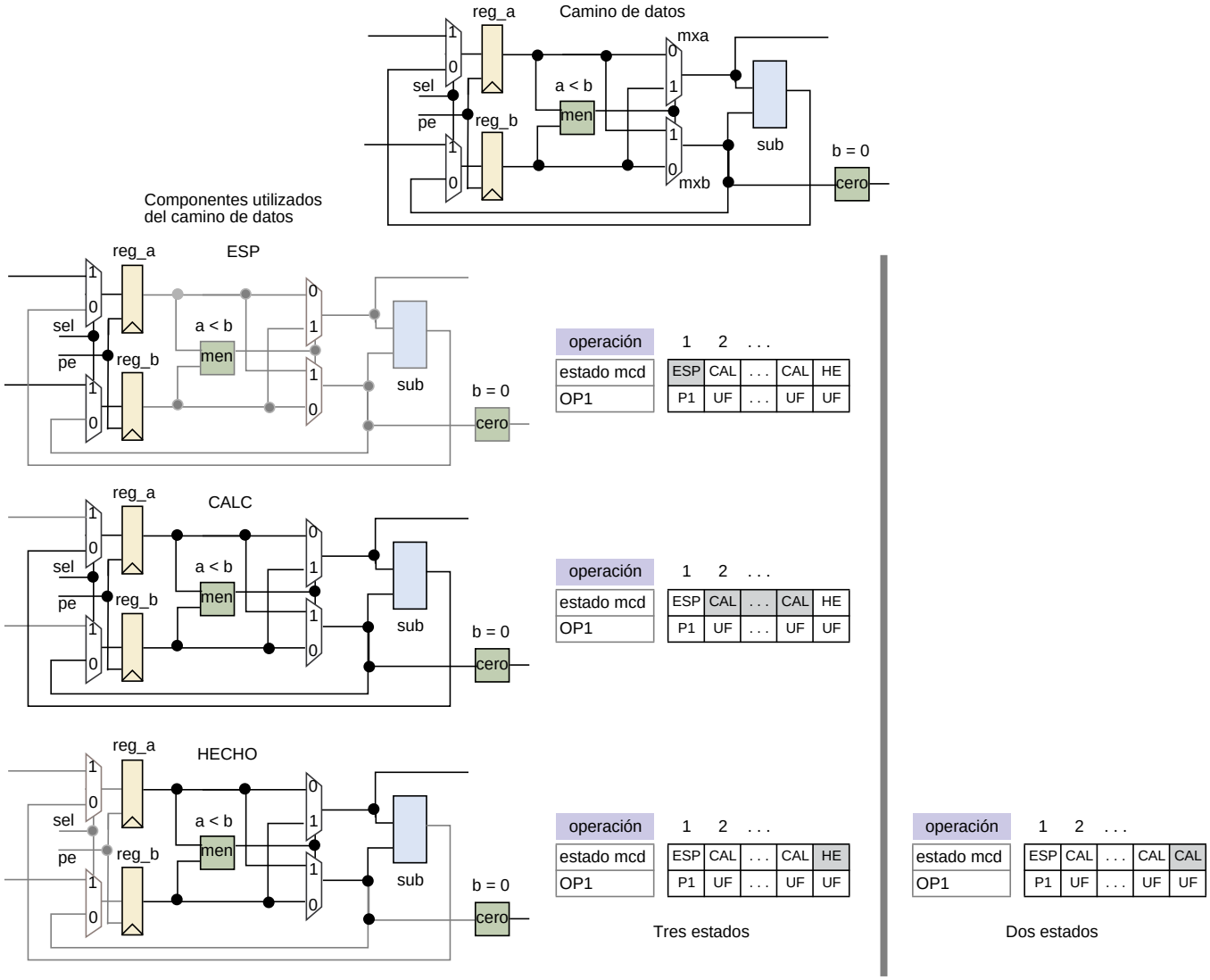


Figura 62 Diseño secuencial. recursos utilizados del camino de datos en cada estado. Diseños del controlador con tres y dos estados.

En el estado ESP se almacenan los datos de entrada en los registros. En el estado CALC se utilizan todos los componentes del camino de datos y se almacena el resultado en los registros. En el estado HECHO no es necesario actualizar los registros. El camino de datos está evaluando siempre lo mismo.

En la parte derecha de la Figura 62 se muestra la utilización del camino de datos cuando el autómata de control utiliza dos estados. La ocupación de los recursos es similar al caso de tres estados. La diferencia radica en que no se cambia de estado cuando la señal cero es activada.

Apéndice 1.5: Diseño secuencial: inicio del cálculo en el estado ESP

A.5.1 Utilización del camino de datos en cada estado

En la Figura 63 se muestran los componentes del camino de datos utilizados en cada uno de los dos estados del autómata de control.

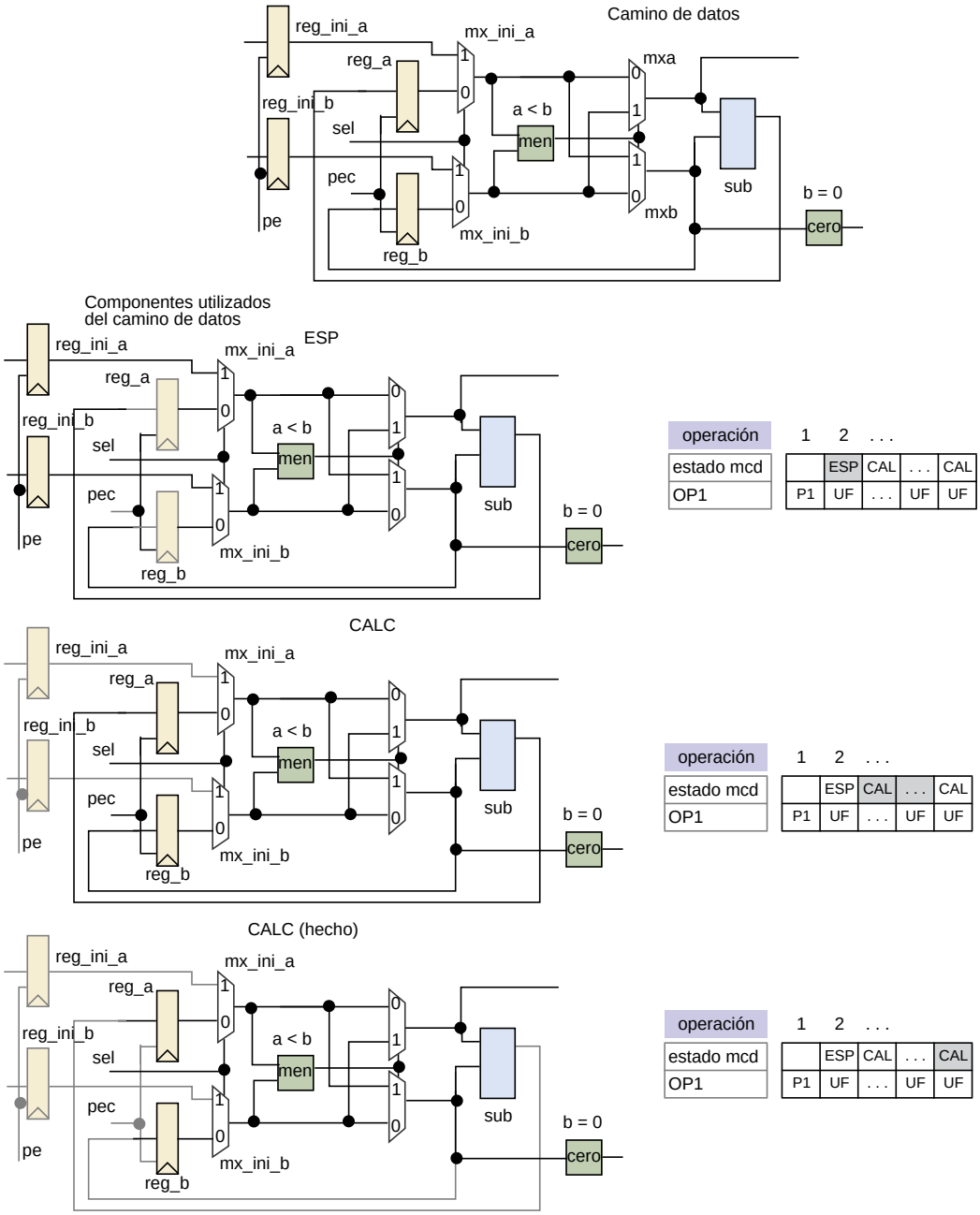


Figura 63 Diseño secuencial. recursos utilizados del camino de datos en cada estado. Diseños del controlador con dos estados e inicio del cálculo en el estado ESP.

En el estado ESP los registros de desacoplo de entrada almacenan los datos de entrada. La salida de estos registros alimenta al camino de datos. Al inicio del siguiente ciclo, el cálculo se almacena en los registros internos del camino de datos.

En el estado CALC los registros internos del camino de datos alimentan al camino de datos. Una vez finalizado el cálculo, estado CALC(hecho), no es necesario actualizar los registros internos del camino de datos, ya que el camino de datos estaría evaluando lo mismo.

Apéndice 1.6: Diseño secuencial. Autómatas de control

A.6.1 Estados ESP, CALC y HECHO

En la tabla de la Figura 64 se muestran las transiciones entre estados y el valor de las señales de salida en función de las señales de entrada del autómata de la Figura 4 o de la Figura 8. En horizontal se muestran las señales de entrada y en vertical los estados. En cada casilla se muestra el estado destino y el valor de las señales.

		interfaces				interna (control)	
		op_dis	$\overline{\text{op\_dis}}$	consumo	$\overline{\text{consumo}}$	cero	$\overline{\text{cero}}$
Estados	ESP	CALC pe <= '1'; mxa <= '1'; finalizada <= '0'; desocupada <= '1';	ESP pe <= '0'; mxa <= '0';				
	CALC					HECHO pe <= '0'; mxa <= '0'; finalizada <= '0'; desocupada <= '0';	CALC pe <= '1';
	HECHO			ESP pe <= '0'; mxa <= '0'; finalizada <= '1'; desocupada <= '0';	HECHO		

Figura 64 Estados ESP, CALC y HECHO: Tabla de transiciones entre estados y valor de las señales de salida. La señal mxa se denomina sel en la Figura 2.

En la Figura 65 se muestra la lógica de cambio de estado y de salida utilizando un diseño con puertas y registros.

Para la codificación de los estados se utiliza un vector de bits. Cada uno de los bit está asociado a un estado[14]. En las siguientes figuras un estado se comprueba o valida utilizando el resto de bits.

Por otro lado, la lógica no ha sido simplificada. Ello permite una correlación directa entre cada entrada en la tabla (Figura 64), que describe las transiciones entre estados, y una cadena de puertas lógicas.

Las señales de salida finalizada y desocupada son de tipo Moore.

No existe un camino combinacional entre la entrada op_dis y las salidas finalizada y desocupada. Tampoco existe un camino combinacional entre la entrada consumo y las salidas finalizada y desocupada.

14. En ingles “one-hot”. Grupos de bits donde las combinaciones aceptables sólo tienen un bit activo.

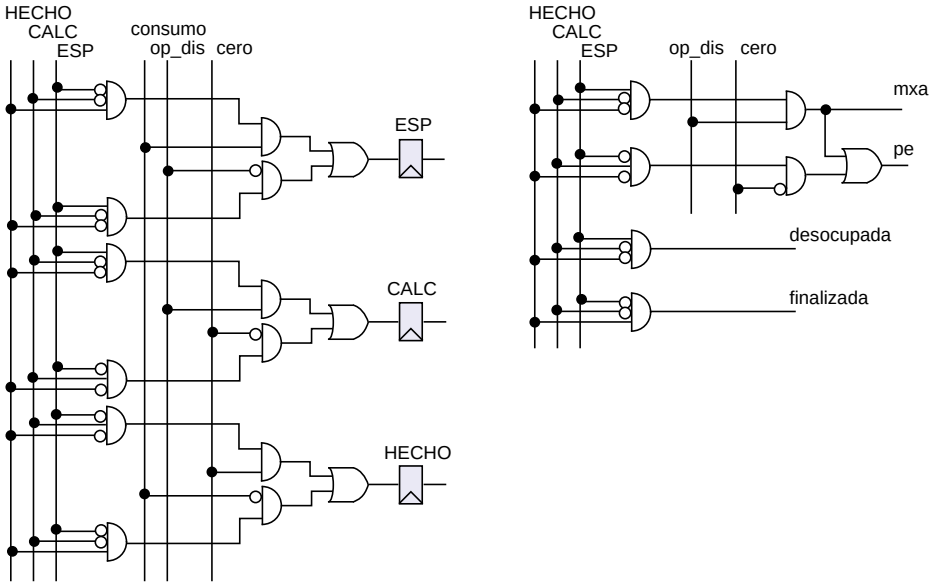


Figura 65 Esquema de circuito con puertas lógicas y registros del autómata de la Figura 64. La señal mxa se denomina sel en la Figura 2.

A.6.2 Estados ESP y CALC

En la tabla de la Figura 66 se muestran las transiciones entre estados y el valor de las señales de salida en función de las señales de entrada del autómata de la Figura 13. En horizontal se muestran las señales de entrada y en vertical los estados. Para las señales de entrada se utilizan dos filas. En cada casilla se muestra el estado destino y el valor de las señales.

		interface		interna (control), interface consumidor		
		op_dis	$\overline{\text{op\_dis}}$	cero		$\overline{\text{cero}}$
				consumo	$\overline{\text{consumo}}$	
Estados	ESP	CALC	ESP			
		pe <= '1'; mxa <= '1';	pe <= '0'; mxa <= '0';			
		finalizada <= '0'; desocupada <= '1';				
	CALC			ESP	CALC	CALC
				pe <= '0'; mxa <= '0';		pe <= '1'; mxa <= '0';
				finalizada <= '1'; desocupada <= '0';		finalizada <= '0'; desocupada <= '0';

Figura 66 Estados ESP y CALC : Tabla de transiciones entre estados y valor de las señales de salida. La señal mxa se denomina sel en la Figura 2.

En la Figura 67 se muestra la lógica de cambio de estado y de salida utilizando un diseño con puertas y registros.

La señal de salida finalizada es de tipo Mealy (depende de la señal cero) y la señal de salida desocupada es de tipo Moore.

No existe un camino combinacional entre la entrada op_dis y las salidas finalizada y desocupada. Tampoco existe un camino combinacional entre la entrada consumo y las salidas finalizada y desocupada.

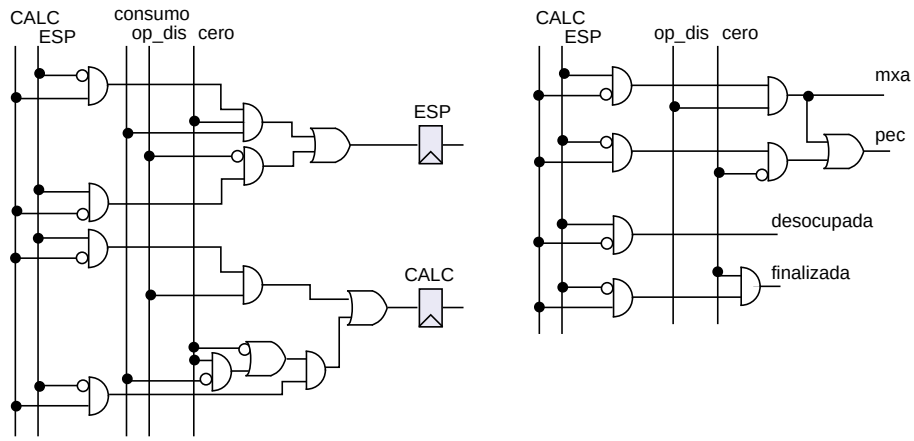


Figura 67 Esquema de circuito con puertas lógicas y registros del autómata de la Figura 66. La señal mxa se denomina sel en la Figura 2.

A.6.3 Estados ESP y CALC y evaluación en ESP

En la tabla de la Figura 68 se muestran las transiciones entre estados y el valor de las señales de salida en función de las señales de entrada del autómata de la Figura 16. En horizontal se muestran las señales de entrada y en vertical los estados. Para las señales de entrada se utilizan tres filas. En cada casilla se muestra el estado destino y el valor de las señales. La señal r_op_dis es la salida del registro cuya entrada es op_dis (Figura 18).

		interfaces, interna (control),				interna (control), interface consumidor		
		r_op_dis			$\overline{\text{r\_op\_dis}}$	cero		$\overline{\text{cero}}$
		cero		$\overline{\text{cero}}$		consumo	$\overline{\text{consumo}}$	
		consumo	$\overline{\text{consumo}}$					
Estados	ESP	ESP	CALC	CALC	ESP			
		pec = ‘0’;	pec = ‘1’;	pec = ‘1’;	pec = ‘0’;			
		mxa <= ’1’;						
		desocupada <= ’1’;	desocupada <= ’0’;	desocupada <= ’0’;	desocupada <= ’1’;			
		finalizada <= ’1’;		finalizada <= ’0’;				
	CALC					ESP	CALC	CALC
						pec = ‘0’;	pec = ‘0’;	pec <= ’1’;
						mxa <= ’0’;		
						desocupada <= ’1’;	desocupada <= ’0’;	desocupada <= ’0’;
						finalizada <= ’1’;		finalizada <= ’0’;

Figura 68 Estados ESP y CALC y evaluación en ESP: Tabla de transiciones entre estados y valor de las señales de salida. La señal mxa se denomina sel en la Figura 15.

En la Figura 69 se muestra la lógica de cambio de estado y de salida utilizando un diseño con puertas y registros.

Las señales de salida finalizada y desocupada son de tipo Mealy. Dependen de las señales cero, op_dis y consumo.

Hay un camino combinacional entre la entrada op_dis y las salidas desocupado y finalizada. Además, existe un camino combinacional entre la entrada op_dis y las salidas finalizada y desocupada. También existe un camino combinacional entre la entrada consumo y la salida desocupada. No existe un camino combinacional entre la entrada consumo y la salida finalizada.

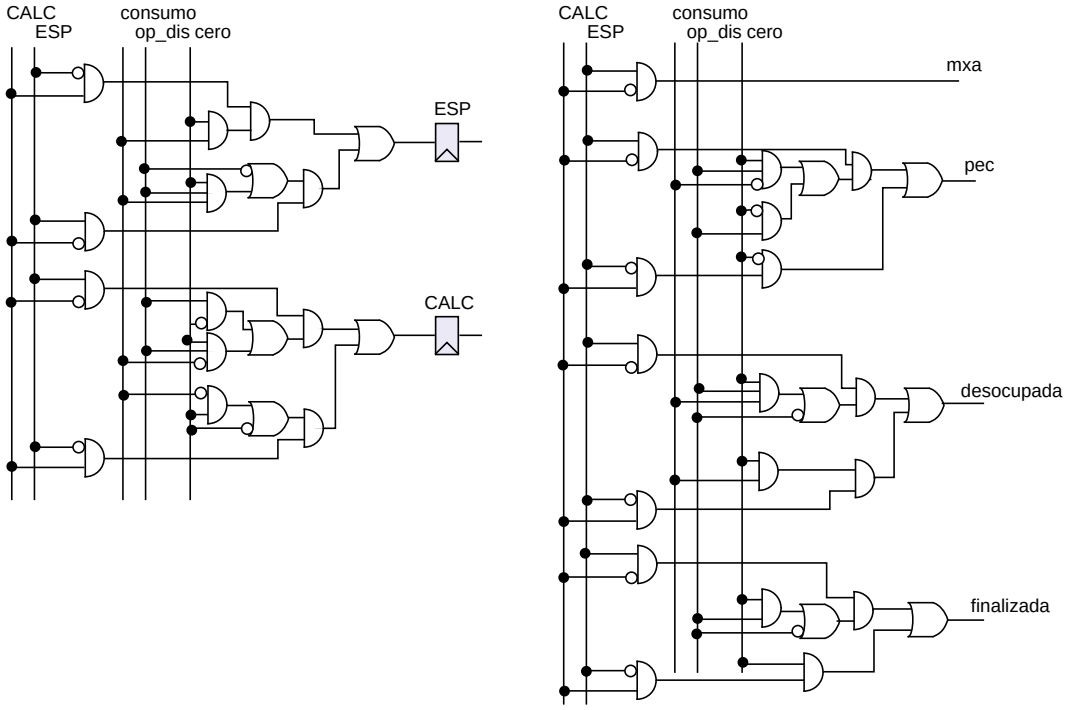


Figura 69 Esquema de circuito con puertas lógicas y registros del autómata de la Figura 68. La señal mxa se denomina sel en la Figura 15.

Apéndice 1.7: Retardos

En la Figura 70 se muestran los retardos que se utilizan en los diagrama temporales que se muestran en este apéndice y que se solicitan en varios trabajos[15].

retardos	
constant retMUX2: time := 1 ns;	multiplexores
constant retREGDES: time := 1 ns;	registros
constant retcero: time := 1 ns;	evaluación de valor cero
constant retmenor: time := 4 ns;	evaluación de menor
constant retsumador: time := 4 ns;	sumador
constant retardo_estado: time := 1 ns;	registro de estado
constant retardo_logica_estado: time := 1 ns;	lógica de próximo estado
constant retardo_logica_salida: time := 1 ns;	lógica de salida

Figura 70 Retardos que deben utilizarse en los diagramas temporales.

En los diagramas temporales que se muestran se supone que las señales op_dis y consumo son estables durante todo el periodo de la señal de reloj. Esta hipótesis también se aplica cuando se solicita un diagrama temporal.

A.7.1 Proyectos 1 y 2

Por completitud, en la Figura 71 se muestra el camino de datos de los proyectos 1 y 2 del diseño serie. En el mismo han sido especificados los retardos.

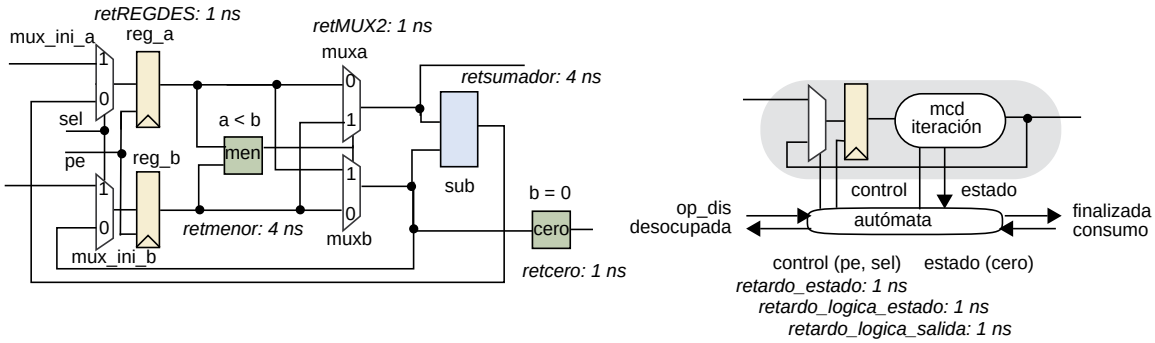


Figura 71 Camino de datos y control de los proyecto 1 y 2 del diseño serie.

En la Figura 72 se muestran el diagrama temporal de los retardos de los proyectos 1 y 2 del diseño serie. También se muestra el periodo de la señal de reloj, especificando dos flancos de subidas contíguos. El flanco de baja puede estar en cualquier punto del periodo ya que no se utiliza.

15. Los retardos no se corresponden con los retardos especificados en el código VHDL.

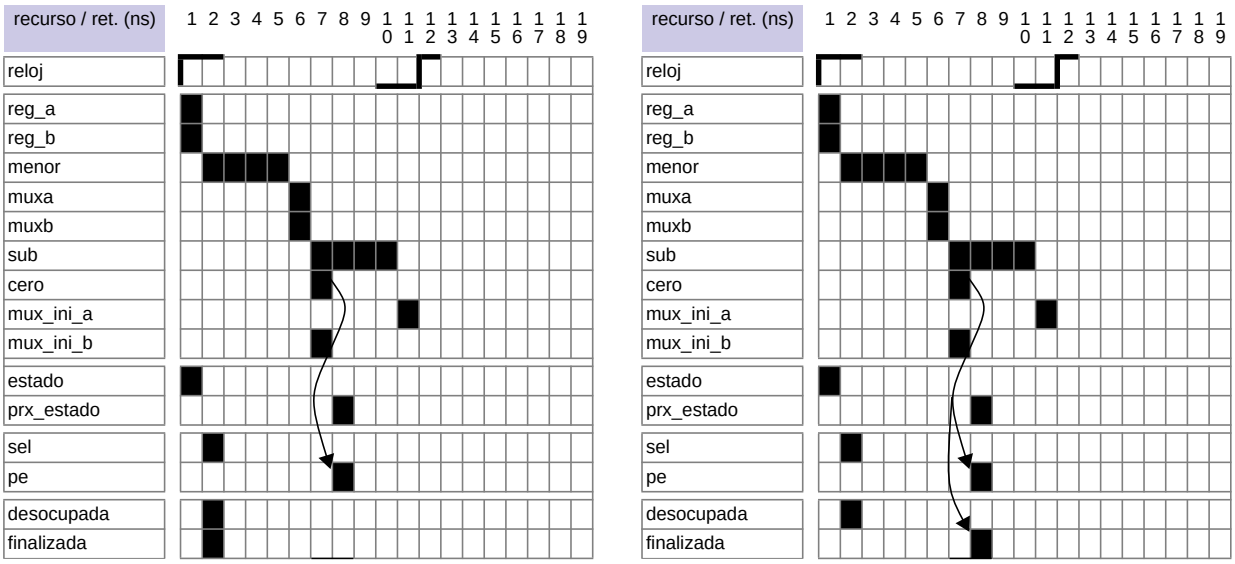


Figura 72 Diagramas de retardos de los proyectos 1 (izquierda) y 2 (derecha) del diseño serie.

A.7.2 Proyecto 3

Por completitud, en la Figura 73 se muestra el camino de datos del proyecto 3 del diseño serie. En el mismo han sido especificados los retardos.

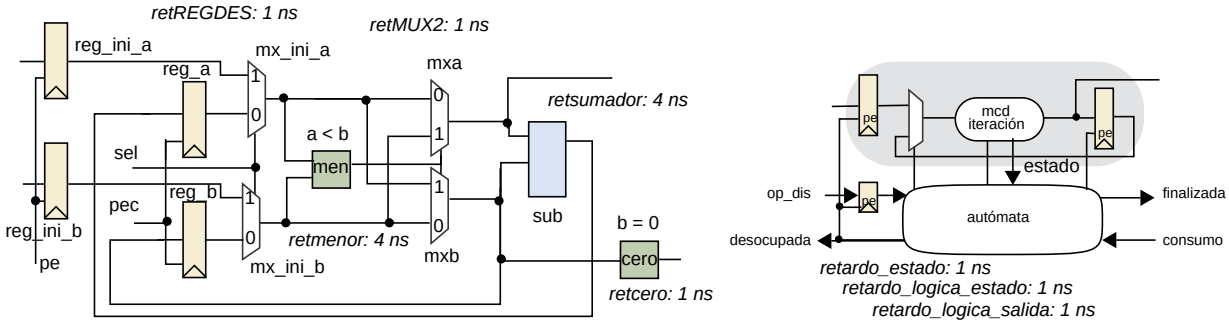


Figura 73 Camino de datos y control del proyecto 3 del diseño serie.

En la Figura 74 se muestran el diagrama temporal de los retardos del proyecto 3 del diseño serie. También se muestra el periodo de la señal de reloj, especificando dos flancos de subidas contíguos. El flanco de baja puede estar en cualquier punto del periodo ya que no se utiliza.

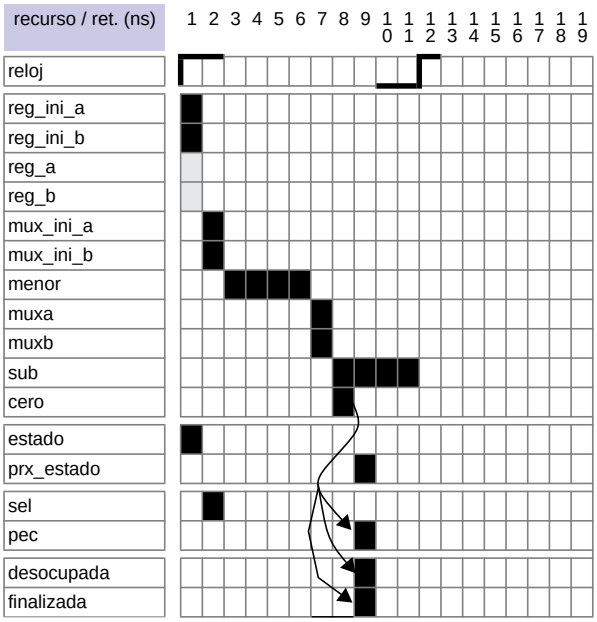


Figura 74 Diagrama de retardos del proyecto 3 del diseño serie

Apéndice 1.8: Proyecto 1. Organización de los ficheros

A.8.1 Diseño secuencial

En la Figura 75 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.

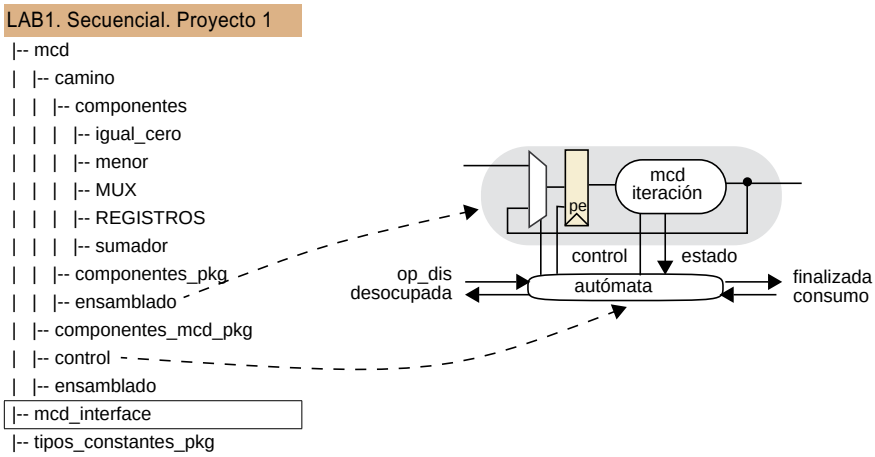


Figura 75 MCD. Proyecto 1. Organización de los directorios.

El directorio componentes incluye los elementos que se utilizan para ensamblar el camino de datos (directorio camino/ensamblado).

El directorio control/CODIGO incluye el fichero donde está especificado el autómata de control.

El directorio ensamblado/CODIGO contiene un fichero con una descripción estructural de la mcd.

El directorio mcd_interface/CODIGO contiene un fichero con el módulo etapa_mcd.

En el directorio tipos_constantes_pkg se ubican ficheros con tipos y constantes utilizados en el diseño.

Apéndice 1.9: Proyecto 1. Simulación

A.9.1 MCD

El directorio que contiene el módulo etapa_mcd se denomina "mcd_interface", enmarcado en la Figura 75, y está ubicado en el primer nivel de directorios (Apéndice 1.8). Los directorios que incluyen los ficheros utilizados por Quartus y Modelsim se indican en la Figura 76. Los ficheros *.qsf y *.qpf del directorio QUARTUS especifican el proyecto Quartus.

Herramienta	Directorio	Directorio
Quartus		
	ENSAMBLADO	QUARTUS
Modelsim		PRUEBAS
		RESULTADOS

Figura 76 Proyecto 1. Directorios utilizados por las herramientas Quartus y Modelsim.

El directorio "mcd_interface" incluye el directorio PRUEBAS (Figura 77). En este directorio se incluye, entre otros, el programa de prueba.

Directorio	Directorio	Comentario
PRUEBAS		
	prueba_ensamblado.vhd	Programa de pruebas
	procedimientos_prueba_pkg.vhd	Procedimientos para iniciar operaciones y capturar resultados
	formato_ventanas.do	Ficheros utilizados por Modelsim para formatear la ventana temporal.
	wave.do	

Figura 77 Proyecto 1. Directorio PRUEBAS.

A.9.2 Programa de prueba

Este programa, entre otros "process" tiene un proceso productor y un proceso consumidor. Estos procesos se sincronizan con la mcd utilizando el protocolo listo/válido. En concreto, la interface del productor para emitir una nueva petición y la interface del consumidor para extraer un dato siguen el patrón que se muestra en la Figura 78.

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

Figura 78 Interfaces en "process" productor y consumidor del programa de prueba.

A.9.3 Emisión de peticiones

El fichero "procedimientos_prueba_pkg.vhd" incluye un procedimiento para emitir operaciones a la mcd desde el "process" productor, incluido en el programa de pruebas. También se dispone de un procedimiento para extraer resultados de la mcd, utilizado por el proceso

consumidor. Estos procedimiento incluyen la interface correspondiente. Los parámetros de los procedimientos utilizados se muestran parcialmente en la Figura 79. En el procedimiento “producir_datos”, el parámetro “tiempoproducir” se utiliza para emular un tiempo de producción de las entradas para un cálculo. En el procedimiento “consumir_datos” el mismo parámetro se utiliza para emular un tiempo de procesado del valor en el consumidor, antes de aceptar un nuevo valor.

	parámetros				
procedimiento	tiempoproducir	a	b	dato_ext	dato_proc
producir_datos	ciclos antes de solicitar un cálculo	valor	valor		
consumir_datos	ciclos para consumir un resultado			valor capturado	valor capturado después de tiempoproducir

Figura 79 Algunos parámetros de los procedimientos utilizados en el program de prueba.

En la Figura 80 se muestra la secuencia de operaciones emitidas por el productor, con el intervalo de tiempo entre ellas (tiempoproducir) en el programa de prueba. También se muestra la secuencia de extracciones del consumidor, con el intervalo de tiempo entre disponibilidades de extracción.

	productor			consumidor
operaciones	a	b	T	T
OP1	21	12	1	1
OP2	84	48	1	8
OP3	15	10	1	1
OP4	0	8	12	4
OP5	7	0	1	1

Figura 80 Proyectos 1 y 2. Secuencia de operaciones en el programa de prueba. T indica tiempoproducir.

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

A.9.4 Evolución de las señales del camino de datos

En la Figura 81 se describen algunas de las señales del camino de datos que se muestran en la ventana temporal de Modelsim[16].

16. 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 emerge después de pulsar con el ratón hay que seleccionar “Object Declaration”.

1	/prueba_etapa_mcd/ciclo	ciclo
2	/prueba_etapa_mcd/reloj	reloj
3	//prueba_etapa_mcd/pcero	puesta a cero. Inicialización
	Flujo de informacion	
	Productor	
4	/prueba_etapa_mcd/a	dato de entrada
5	/prueba_etapa_mcd/b	dato de entrada
	Interface Pro-mcd	
6	/prueba_etapa_mcd/val_lista_Pro_mcd	protocolo productor/mcd
	Estado automata y salida mcd	
7	/prueba_etapa_mcd/pr_mcd/pr_mcd/cnt/estado	estado del controlador
8	/prueba_etapa_mcd/pr_mcd/pr_mcd/cnt/prxestado	próximo estado del controlador
9	/prueba_etapa_mcd/s	salida de la mcd
	Interface mcd_Consu	
10	/prueba_etapa_mcd/val_lista_mcd_Consu	protocolo mcd/consumidor
	Consumidor	
11	/prueba_etapa_mcd/DATextraido	dato extraido
	Registros	
12	/prueba_etapa_mcd/pr_mcd/pr_mcd/cam/reg_a	salida del registro a
13	/prueba_etapa_mcd/pr_mcd/pr_mcd/cam/reg_b	salida del registro b
14	/prueba_etapa_mcd/pr_mcd/pr_mcd/pe	permiso de escritura en los registros
15	/prueba_etapa_mcd/pr_mcd/pr_mcd/ini	selección en los multiplexores

Figura 81 Etiquetas utilizadas en las señales que se muestran en la ventana de tiempo.

Algunas 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".

En la Figura 82 se muestra, utilizando el ordinal de la primera columna de la tabla (Figura 81), la ubicación de la señal en el diseño de la mcd.

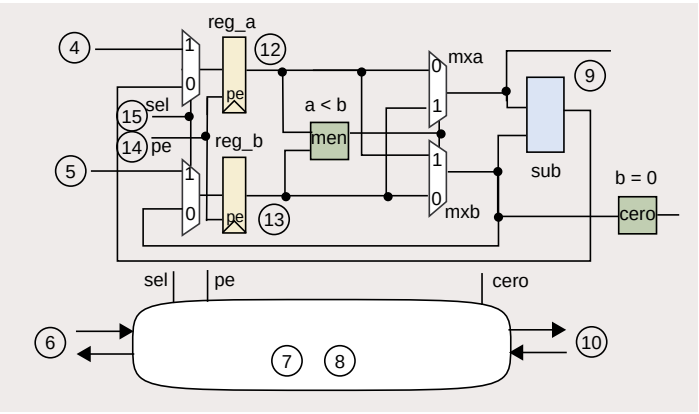


Figura 82 Señales que se observan en la simulación con Modelsim. La señal sel se denomina ini en el código.

A.9.5 Ejemplos de las señales en la ventana temporal

En la siguiente figura, 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 contígua 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.

En la Figura 83 se muestra el diagrama temporal de un cálculo.

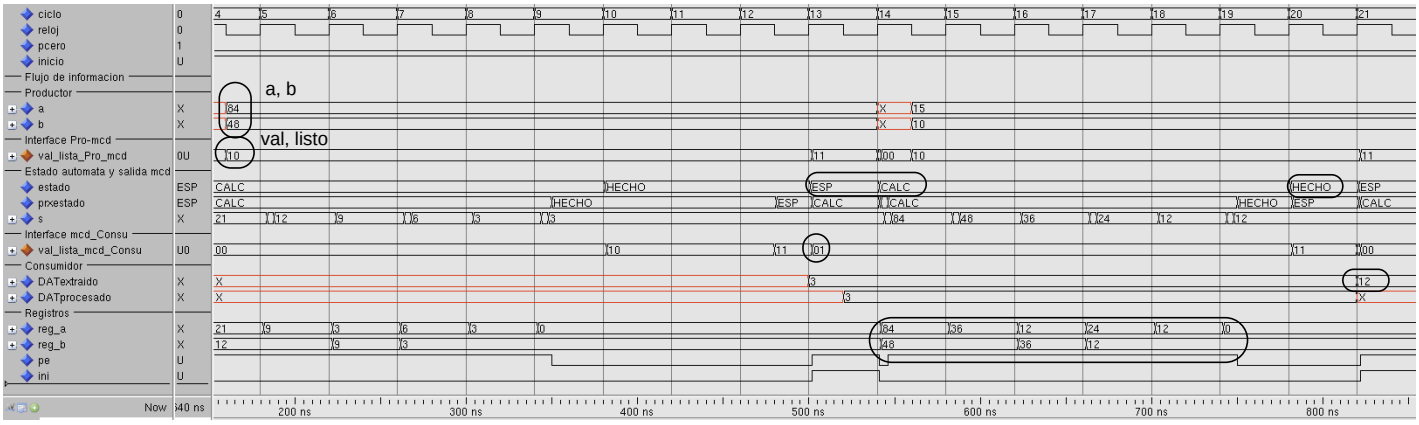


Figura 83 Modelsim. Diagrama temporal de un cálculo en la mcd.

En el ciclo 4 el productor solicita un cálculo. La mcd no está lista hasta el ciclo 13. Durante los siguientes ciclos (5 ciclos) la mcd calcula un resultado. El consumidor está dispuesto a consumir un nuevo dato desde el ciclo 13. En el ciclo 20 el consumidor captura el dato. El procesado del dato en el consumidor no se observa.

Simulación ciclo a ciclo

En el programa de prueba se utilizan genéricos para controlar la posibilidad de una simulación paso a paso. Este tipo de simulación es muy recomendable cuando se está comprobando un diseño.

Apéndice 1.10: Proyecto 1. Documentación

A.10.1Diseño secuencial

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/Secuencial/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 ordinales 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 1.11: Proyecto 2. Organización de los ficheros

A.11.1Diseño secuencial

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 84”), 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.

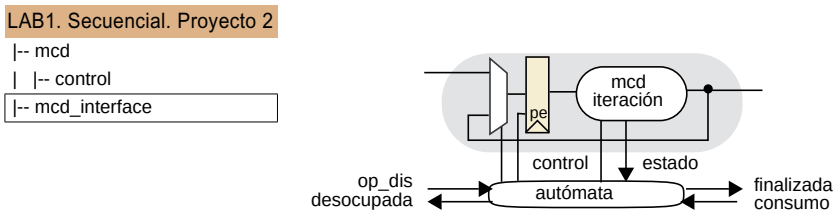


Figura 84 MCD. Proyecto 2. Organización de los directorios.

En este proyecto sólo se modifica el autómata de control respecto del proyecto 1.

Apéndice 1.12: Proyecto 2. Simulación

A.12.1Diseño secuencial

Utilizando los directorios asociados al proyecto, la simulación se efectúa de forma idéntica a la descrita en el proyecto 1.

La secuencia de operaciones, su inicio y extracción, en el programa de prueba, es la misma que en el proyecto 1.

Apéndice 1.13: Proyecto 2. Documentación

A.13.1Diseño secuencial

La documentación es la misma que en el proyecto 1. Hay que diseñar el autómata de control. Ahora bien también ha sido generada en otro directorio: DOCUMENTACION/Secuencial/proyecto_2.

Apéndice 1.14: Proyecto 3. Organización de los ficheros

A.14.1Diseño secuencial

El árbol de directorios, desde el directorio raíz, de este proyecto es mimético al árbol del proyecto 1. En los subdirectorios miméticos incluidos en este proyecto (“proyecto 3”, Figura 85), 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.

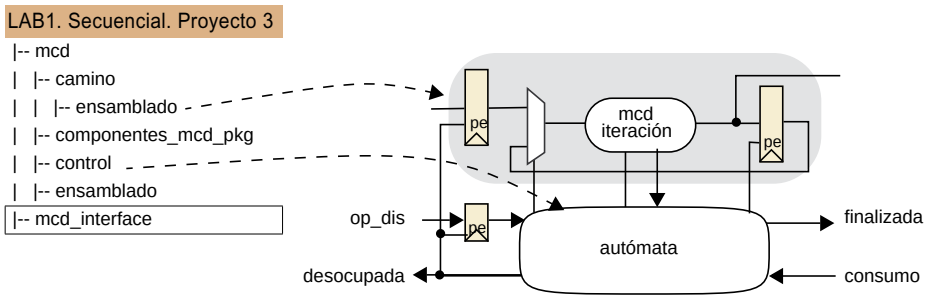


Figura 85 MCD. Proyecto 3. Organización de los directorios.

En este proyecto se modifica el camino de datos y el control respecto del proyecto 1.

Apéndice 1.15: Proyecto 3. Simulación

A.15.1Diseño secuencial

Utilizando los directorios asociados al proyecto, la simulación se efectúa de forma idéntica a la descrita en el proyecto 1.

En la Figura 86 se muestra la secuencia de operaciones emitidas por el productor, con el intervalo de tiempo entre ellas (tiempoproducir), en el programa de prueba. También se muestra la secuencia de extracciones del consumidor, con el intervalo de tiempo entre disponibilidades de extracción.

	productor			consumidor
operaciones	a	b	T	T
OP1	21	12	1	1
OP2	84	48	1	1
OP3	15	0	1	4
OP4	8	1	1	1
OP5	8	0	1	4
OP6	0	7	1	1
OP7	10	5	1	1

Figura 86 Proyectos 3. Secuencia de operaciones en el programa de prueba. T indica tiempoproducir

En la tabla de la Figura 87 se describen señales adicionales que se muestran en la ventana temporal de Modelsim.

	Reg. Entrada	
16	sim:/prueba_etapa_mcd/pr_mcd/reg_ent_a	dato de entrada
17	sim:/prueba_etapa_mcd/pr_mcd/reg_ent_b	dato de entrada
18	sim:/prueba_etapa_mcd/pr_mcd/reg_ent_op_dis	datos disponibles
	Reg. Internos	
19	sim:/prueba_etapa_mcd/pr_mcd/pr_mcd/cam/reg_a	salida del registro
20	sim:/prueba_etapa_mcd/pr_mcd/pr_mcd/cam/reg_b	salida del registro

Figura 87 Proyecto 3. Etiquetas de las señales adicionales que se muestran en la ventana de tiempo.

En la Figura 88 se muestran, en el camino de datos, utilizando el ordinal de la Figura 87, las señales relacionadas en la Figura 87.

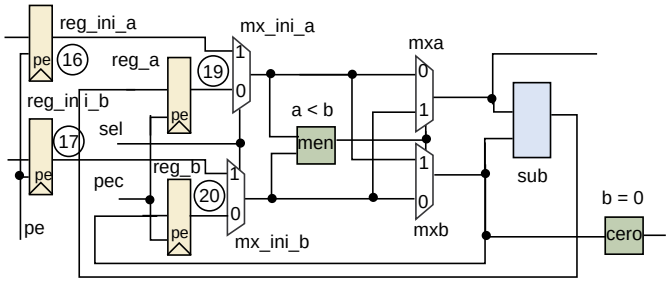


Figura 88 Proyecto 3. Señales adicionales que se muestran en la ventana temporal.

Apéndice 1.16: Proyecto 3. Documentación

A.16.1Diseño secuencial

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/Secuencial/proyecto_3.

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 1.17: Proyecto 4. Organización de los ficheros

A.17.1Diseño secuencial

El árbol de directorios, desde el directorio raíz, de este proyecto es mimético al árbol del proyecto 3. En los subdirectorios miméticos incluidos en este proyecto (“proyecto 4”, Figura 89), 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.

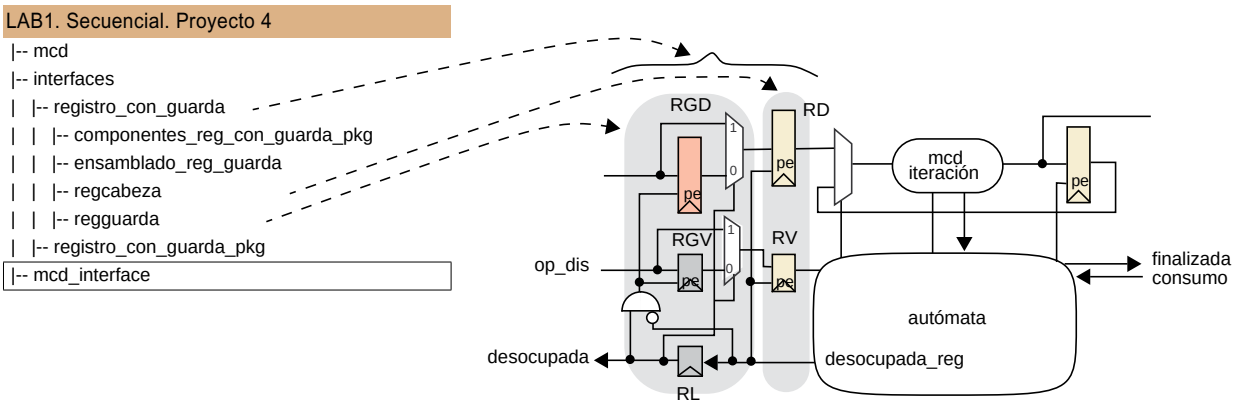


Figura 89 MCD. Proyecto 1. Organización de los directorios.

En este proyecto el productor observa la señal listo un ciclo más tarde que en el proyecto 3. Por ello, se incluyen registros de guarda en el camino de datos.

Apéndice 1.18: Proyecto 4. Simulación

A.18.1Diseño secuencial

Utilizando los directorios asociados al proyecto, la simulación se efectúa de forma idéntica a la descrita en el proyecto 1.

La secuencia de operaciones, su inicio y extracción, en el programa de prueba, es la misma que en el proyecto 3.

En la tabla de la Figura 90 se describen señales relativas a los registros con guarda que se muestran en la ventana temporal de Modelsim.

	Detalle registro con guarda	
	registro de guarda (entrada)	
21	sim:/prueba_etapa_mcd/pr_mcd/r_c_gua_a/reg_gua/rdv_e.dat	entrada a: dato
22	sim:/prueba_etapa_mcd/pr_mcd/r_c_gua_a/reg_gua/VAL_LISTO_RG	entrada a: válido/listo (salida)
23	sim:/prueba_etapa_mcd/pr_mcd/r_c_gua_a/reg_gua/rdv_s.dat	salida a: dato
24	sim:/prueba_etapa_mcd/pr_mcd/r_c_gua_a/reg_gua/val_listo_RG_RD	salida a: válido (entrada)/salida
	registro de cabeza	
25	sim:/prueba_etapa_mcd/pr_mcd/r_c_gua_a/reg_cab/rdv_e.dat	entrada a: dato
26	sim:/prueba_etapa_mcd/pr_mcd/r_c_gua_a/reg_cab/VAL_listo_RD	entrada a: válido/listo (salida)
27	sim:/prueba_etapa_mcd/pr_mcd/r_c_gua_a/reg_cab/rdv_s.dat	salida a: dato
28	sim:/prueba_etapa_mcd/pr_mcd/r_c_gua_a/reg_cab/val_listo_CAB	salida a: válido (entrada)/salida

Figura 90 Proyecto 4. Etiquetas de las señales relativas al registro con guarda a que se muestran en la ventana de tiempo.

En la Figura 91 se muestran en el camino de datos, utilizando el ordinal de la Figura 90, las señales relacionadas en la Figura 90.

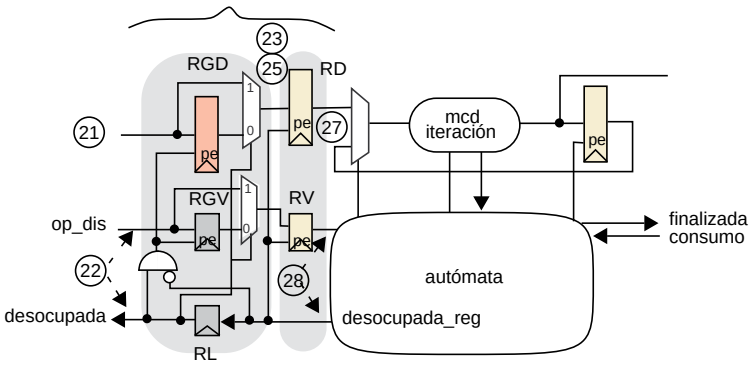


Figura 91 Proyecto 4. Señales relativas al registro con guarda "a" en la ventana temporal

Apéndice 1.19: Proyecto 4. Documentación

A.19.1Diseño secuencial

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/Secuencial/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 1.20: Diseño multiciclo. Organización de los ficheros

A.20.1Proyectos 1, 2 y 3

En la Figura 92 se muestra la estructura de directorios de los proyectos correspondientes al diseño multiciclo.

LAB1. Multiciclo. Proyecto 1	LAB1. Multiciclo. Proyecto 2	LAB1. Multiciclo. Proyecto 3
|-- mcd	|-- mcd	|-- mcd
| |-- camino	| |-- control	| |-- camino
| | |-- ensamblado	| | |-- CODIGO	| | |-- ensamblado
| | |-- CODIGO	| | |-- control.vhd	| | |-- CODIGO
| | |-- camino_mcd.vhd	| | |-- estado_pkg.vhd	| | |-- camino_mcd.vhd
| |-- componentes_mcd_pkg	| | |-- proc_func_control_pkg.vhd	| |-- componentes_mcd_pkg
| | |-- componentes_mcd_pkg.vhd	|-- mcd_interface	| | |-- componentes_mcd_pkg.vhd
| |-- control	| |-- CODIGO	| |-- control
| | |-- CODIGO	| | |-- etapa_mcd.vhd	| | |-- CODIGO
| | |-- control.vhd	| |-- PRUEBAS	| | |-- control.vhd
| | |-- estado_pkg.vhd	| | |-- procedimientos_prueba_pkg.vhd	| | |-- estado_pkg.vhd
| | |-- proc_func_control_pkg.vhd	| | |-- prueba_etapa_mcd.vhd	| | |-- proc_func_control_pkg.vhd
| |-- ensamblado	| |-- RESULTADOS	| |-- ensamblado
| |-- CODIGO		| |-- CODIGO
| |-- mcd.vhd		| |-- mcd.vhd
|-- mcd_interface		|-- mcd_interface
| |-- CODIGO		| |-- CODIGO
| | |-- etapa_mcd.vhd		| | |-- etapa_mcd.vhd
| |-- PRUEBAS		| |-- PRUEBAS
| | |-- procedimientos_prueba_pkg.vhd		| | |-- procedimientos_prueba_pkg.vhd
| | |-- prueba_etapa_mcd.vhd		| | |-- prueba_etapa_mcd.vhd
| |-- RESULTADOS		| |-- RESULTADOS

Figura 92 MCD. Proyecto 1. Organización de los directorios.

El directorio “componentes” del diseño serie o secuencial incluye los elementos que se utilizan para ensamblar el camino de datos (directorio Secuencial/camino/componentes). Del diseño secuencial también se utilizan los ficheros ubicados en el directorio tipos_constantes_pkg.

Proyecto 1 y proyecto 3. El directorio camino/ensamblado/CODIGO incluye el fichero donde debe especificarse el camino de datos.

Los 3 proyectos . El directorio control/CODIGO incluye el fichero donde debe especificarse el autómata de control.

Los 3 proyectos . El directorio ensamblado/CODIGO contiene un fichero con una descripción estructural de la mcd.

Los 3 proyectos . El directorio mcd_interface/CODIGO contiene un fichero con el módulo etapa_mcd.

Apéndice 1.21: Diseño multiciclo. Simulación

A.21.1Proyectos 1, 2 y 3

Utilizando los directorios asociados al proyecto, la simulación se efectúa de forma idéntica a la descrita en el proyecto 1.

La secuencia de operaciones, su inicio y extracción, en el programa de prueba, es la misma que en el proyecto 1.

A.21.2Señales en la ventana temporal

En la tabla de la Figura 93 se describen señales que se muestran en la ventana temporal de Modelsim. Las señales que se indican están definidas en las interfaces de los componentes del diseño o definidas como “signal” en los módulos que se suministran.

1	/prueba_etapa_mcd/ciclo	ciclo
2	/prueba_etapa_mcd/reloj	reloj
3	//prueba_etapa_mcd/pcero	puesta a cero. Inicialización
4	//prueba_etapa_mcd/inicio	indicación de inicio del productor y consumidor. Declarada en el programa de prueba
	Flujo de informacion	
	Productor	
5	/prueba_etapa_mcd/a	dato de entrada
6	/prueba_etapa_mcd/b	dato de entrada
	Interface Pro-mcd	
7	/prueba_etapa_mcd/val_lista_Pro_mcd	protocolo productor/mcd
	Estado automata y salida mcd	
8	/prueba_etapa_mcd/pr_mcd/pr_mcd/cnt/estado	estado del controlador
9	/prueba_etapa_mcd/pr_mcd/pr_mcd/cnt/prxestado	próximo estado del controlador
10	/prueba_etapa_mcd/s	salida de la mcd
	Interface mcd_Consu	
11	/prueba_etapa_mcd/val_lista_mcd_Consu	protocolo mcd/consumidor
	Consumidor	
12	/prueba_etapa_mcd/DATextraido	dato extraido
13	/prueba_etapa_mcd/DATprocesado	dato procesado por el consumidor
	OTRAS_mux_registros	
14	/prueba_etapa_mcd/pr_mcd/pr_mcd/ini	Declarada en el módulo mcd
15	/prueba_etapa_mcd/pr_mcd/pr_mcd/menor	Declarada en el módulo mcd
16	/prueba_etapa_mcd/pr_mcd/pr_mcd/pe_a	Declarada en el módulo mcd
17	/prueba_etapa_mcd/pr_mcd/pr_mcd/pe_b	Declarada en el módulo mcd

Figura 93 Diseño multiciclo. Señales que se muestran en la ventana temporal.

Apéndice 1.22: Diseño multiciclo. Documentación

A.22.1Proyectos 1, 2 y 3

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/Multiciclo/proyecto_X, siendo X el ordinal 1, 2 o 3.

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.