TEMA 6

Reducción de Tablas de Estados. Asignación de estados. Descripción mediante VHDL de Tablas de Estados. Simulación de circuitos síncronos utilizando VHDL.

Diseño de circuitos secuenciales síncronos

Implementación de circuitos secuenciales síncronos: Procedimiento general

Paso 1: A partir de la descripción, usualmente verbal del problema, construir un Diagrama ASM y/ó la Tabla de Estados del circuito (otras alternativas son posibles: VHDL, Diagrama MDS, Diagrama de Estados, …).

Paso 2: Decidir el tipo de implementación y elementos de circuito a utilizar (SSIs, MSIs, PLDs, ROMs, …).

Paso 3: Reducir la Tabla de Estados para eliminar estados redundantes. Condicionado en alguna medida por la elección en el Paso 2.

Paso 4: Seleccionar una Asignación de Estados y decidir el tipo de FFs (condicionado por la elección efectuada en el Paso 2) a utilizar en el diseño.

Paso 5: Obtener las Tablas de Transición y de Salida.

Paso 6: Construir la Tabla de Excitación. Obtener las funciones de Excitación y de Salida a partir de las tablas obtenidas en el Paso 5; depende de alguna manera del tipo de implementación seleccionado en el Paso 2.

Paso 7: Dibujar el esquema del circuito completo empleando bloques funcionales para los diferentes elementos de circuito. Etiquetar las distintas señales y elementos del circuito para su adecuada identificación. Realizar la simulación del circuito.

Minimización de FSMs

Equivalencia y Compatibilidad de Estados: Reducción de Tablas de Estados

N: nº de estados de a máquina

F: nª de Flip-flops necesarios para su implementación

$$F = \lceil \log_2 N \rceil$$

Reducción de Estados:

- **Puede reducir el número de FFs requeridos**
- **Contribuye a simplificar la máquina, reduciendo su circuitería**
- **Facilita y reduce el coste de las tareas de test**
- **Facilita la comprensión del diseño**

◊ **Máquinas completamente especificadas:**

- **Todas las transiciones entre estados y todas las salidas asociadas están especificadas. Todas las entradas o casillas de la Tabla de Estados están completas**

  Equivalencia de estados

◊ **Máquinas incompletamente especificadas:**

- **Algunas transiciones entre estados y/o algunas salidas asociadas no están especificadas. Algunas entradas o casillas de la Tabla de Estados están incompletas**

  Compatibilidad de estados (general, válida para máquinas incompletamente especificadas y para máquinas completamente especificadas -caso particular-)

Compatibilidad de Estados

FSMs incompletamente especificadas

Máquina M

S_p, S_q: Dos estados de M

Z_p, Z_q: Respuestas de M (salidas) en los estados S_p y S_q para una misma condición de entrada E_i

Z_p y Z_q son compatibles si los valores especificados en ambas son iguales

Secuencia de condiciones de entrada: $E_i, E_j, E_k, \ldots$
Secuencia de salida si el estado inicial es S_p: $Z_p, Z_r, Z_t, \ldots$
Secuencia de salida si el estado inicial es S_q : $Z_q, Z_s, Z_u, \ldots$

- Las secuencias $Z_p, Z_r, Z_t, \ldots$ y $Z_q, Z_s, Z_u, \ldots$ son compatibles si los pares correspondiente de salidas Z_p y Z_q, Z_r y Z_s, Z_t y Z_u, …, son compatibles

- Dos estados S_p y S_q de M son compatibles si cualquier secuencia de condiciones de entrada aplicada a M cuando inicialmente se encuentra en S_p y en S_q da lugar a secuencias de salida compatibles

Compatibilidad de Estados (2)

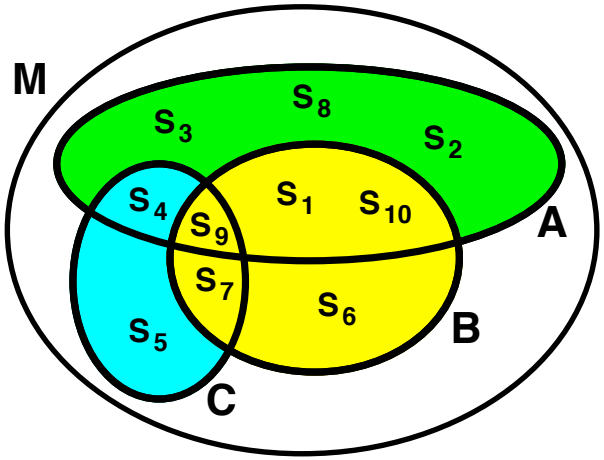
FSMs incompletamente especificadas

- Un conjunto de estados de M forman un compatible, si cada par de estados del conjunto son compatibles
- Un compatible que contiene a todos los estados de otro compatible se dice que lo cubre o que es mayor que el
- Un compatible es máximo si no es cubierto por ningún otro compatible
- Un conjunto de estados que forman un compatible puede ser sustituido por un sólo estado. El estado resultante tiene definidas todas las transiciones y salidas que estén especificadas en cualquiera de los estados del conjunto
- Si la compatibilidad o incompatibilidad de un conjunto de estados está condicionada por la compatibilidad o incompatibilidad de otro conjunto de estados, este último conjunto se denomina compatible implicado

Compatibilidad de Estados (3)

FSMs incompletamente especificadas

- Un conjunto de compatibles que cubre (contiene) todos los estados de M, forma una cobertura de M
- Una cobertura es cerrada si para cada compatible incluido en la misma, todos sus compatibles implicados están también contenidos en (algún compatible de) la misma
- Una máquina puede ser sustituida por una cualquiera de sus coberturas cerradas



$(S_1, S_2, S_3, S_4, S_8, S_9, S_{10}) \Rightarrow (S_4, S_5, S_7), (S_1, S_6)$

$(S_1, S_6, S_7, S_9, S_{10}) \Rightarrow \Phi$

$(S_4, S_5, S_7, S_9) \Rightarrow (S_4, S_7), (S_1, S_6, S_9)$

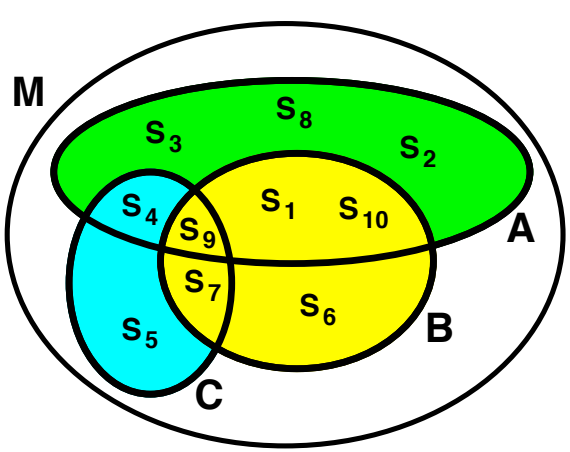
A B C

$\{ (S_1, S_2, S_3, S_4, S_8, S_9, S_{10}) (S_1, S_6, S_7, S_9, S_{10}) (S_4, S_5, S_7, S_9) \}$

Cobertura cerrada

Compatibilidad de Estados (4)

FSMs incompletamente especificadas



$(S_1, S_2, S_3, S_4, S_8, S_9, S_{10}) \Rightarrow (S_4, S_5, S_7), (S_2, S_5)$

$(S_1, S_6, S_7, S_9, S_{10}) \Rightarrow (S_4, S_5, S_7, S_9)$

$(S_4, S_5, S_7, S_9) \Rightarrow (S_4, S_7), (S_1, S_6, S_9)$

$\{ (S_1, S_2, S_3, S_4, S_8, S_9, S_{10}) (S_1, S_6, S_7, S_9, S_{10}) (S_4, S_5, S_7, S_9) \}$

Cobertura no cerrada

Si (S_2, S_5) son compatibles o condicionalmente compatibles (con compatibles implicados)

- Incluir (S_2, S_5) en la cobertura y comprobar si la nueva cobertura es cerrada

(se incrementará el tamaño de la cobertura)

Si (S_2, S_5) son incompatibles

- No existe una cobertura cerrada que incluya los compatibles A, B y C

Probar con otros compatibles o eliminando algunos estados de A, B o C sin que se pierda la cobertura

Compatibilidad de Estados (5)

FSMs incompletamente especificadas

- Encontrar todos los compatibles máximos
- Encontrar una cobertura cerrada con el menor número posible de compatibles

- Una cobertura cerrada mínima no tiene por qué ser única
- Un compatible máximo no siempre forma parte de una cobertura cerrada mínima

- No es posible decidir *a priori* cuál de las coberturas cerrada mínimas es la que da lugar a la implementación de menor coste
- Una cobertura cerrada no mínima puede dar lugar a una implementación de menor coste, dependiendo del criterio considerado, que otra mínima o de menor número de compatibles

- "Trial and Error"

  Acotar las pruebas a realizar en la búsqueda de una cobertura cerrada mínima

  Sistematizar la búsqueda del conjunto de compatibles máximos

  Acotar el número de compatibles de una cobertura cerrada mínima

Tabla de Absorción o Mapa de Implicación

Procedimiento sistemático para obtener los pares de estados compatibles

Paso 1 Construir un Mapa con una celda por cada par de estados. La intersección de la fila, i, y la columna, k, determina la celda $c_{i,k}$ que corresponde a los estados S_i y S_k.

Paso 2 Comparar en la Tabla de Estados las filas correspondientes a dos estados S_i y S_k.
En cada celda $c_{i,k}$ se incluye:

X si S_i y S_k son incompatibles

√ si S_i y S_k son compatibles

(S_p, S_q), …, (S_r, S_s) si la compatibilidad de S_i y S_k está condicionada por la compatibilidad de S_p y S_q, …, S_r y S_s. Los (S_p, S_q), …, (S_r, S_s) son los pares implicados por S_i y S_k.

Paso 3 Una vez construida la Tabla de Absorción, comprobar si se cumple la compatibilidad de los pares incluidos en cada celda $c_{i,k}$. Si en $c_{i,k}$ se incluyen (S_p, S_q), …, (S_r, S_s):

Si en la celda $c_{p,q}$ se incluye X --> sustituir el contenido de $c_{i,k}$ por X.

Si todas las celdas $c_{p,q}$, …, $c_{r,s}$ incluyen √ --> sustituir el contenido $c_{i,k}$ por √.

Repetir este procedimiento con cada celda hasta que no pueda modificarse el contenido de ninguna celda más.

Paso 4 Las celdas que no contengan X corresponden a pares compatibles (puede sustituirse su contenido por √).

Tabla de Absorción (2)

Ejemplo:

Construir la Tabla de Absorción para la máquina cuya Tabla de Estados es la que sigue:

PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
S_1	S_3,00	S_2,--	--,--	S_3,--
S_2	S_1,0-	S_2,00	S_4,-0	--,--
S_3	S_6,--	--,--	S_4,1-	S_1,00
S_4	--,--	S_5,--	S_4,00	S_3,-1
S_5	S_2,--	S_5,01	S_4,10	--,--
S_6	S_6,10	S_7,10	--,--	S_3,--
S_7	S_2,--	S_7,10	S_4,--	--,--
S_8	S_8,01	S_5,--	--,--	S_3,0-

Tabla de Absorción (3)

Ejemplo (cont.)

PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
S_1	S_3,00	S_2,--	--,--	S_3,--
S_2	S_1,0-	S_2,00	S_4,-0	--,--
S_3	S_6,--	--,--	S_4,1-	S_1,00
S_4	--,--	S_5,--	S_4,00	S_3,-1
S_5	S_2,--	S_5,01	S_4,10	--,--
S_6	S_6,10	S_7,10	--,--	S_3,--
S_7	S_2,--	S_7,10	S_4,--	--,--
S_8	S_8,01	S_5,--	--,--	S_3,0-

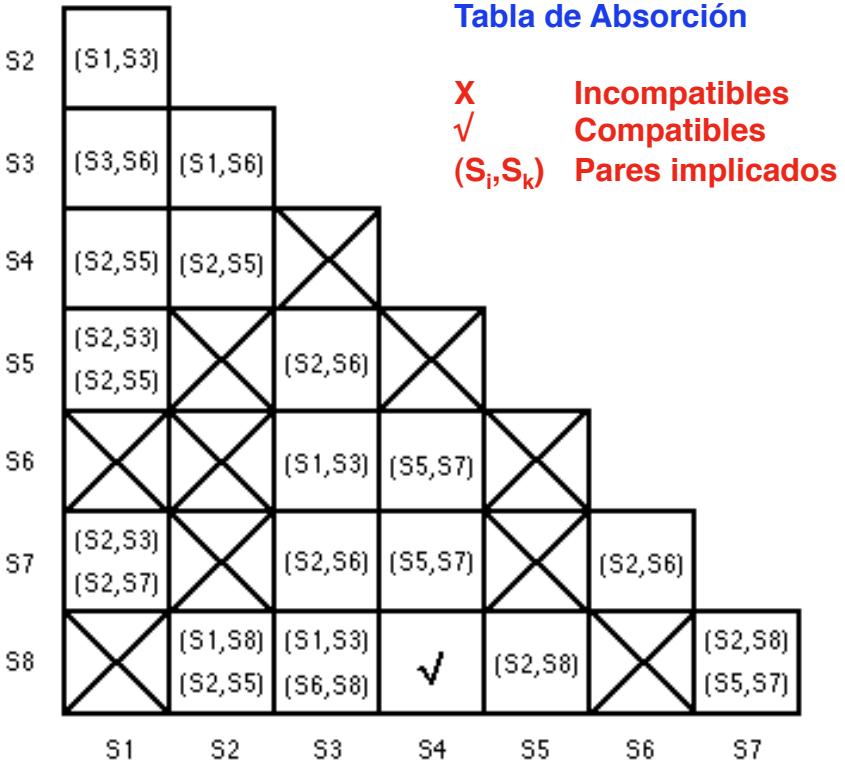


Tabla de Absorción (4)

Ejemplo (cont.)

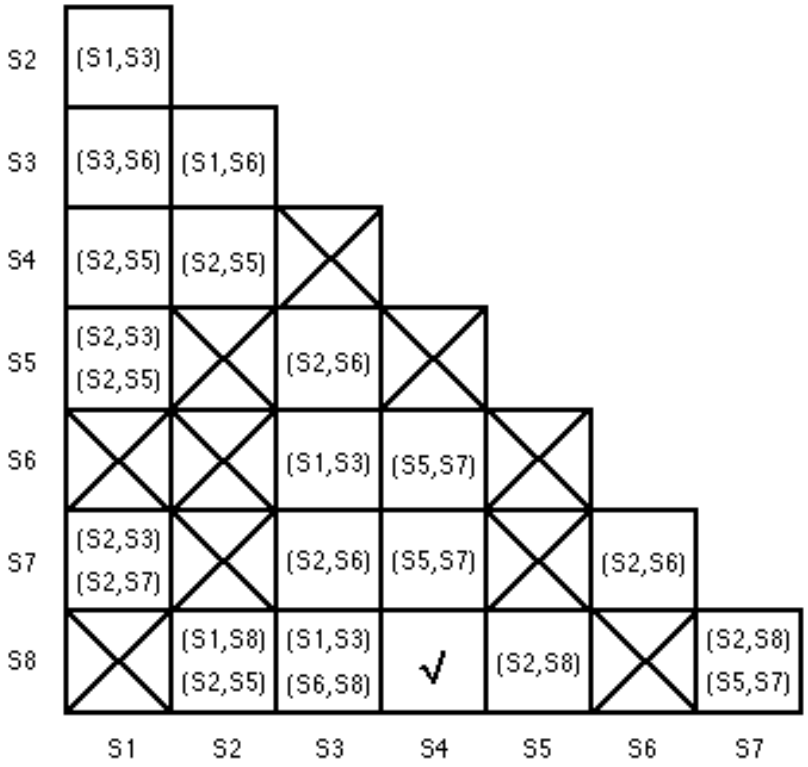


Tabla de Absorción

Comprobación de la compatibilidad de los Pares Implicados

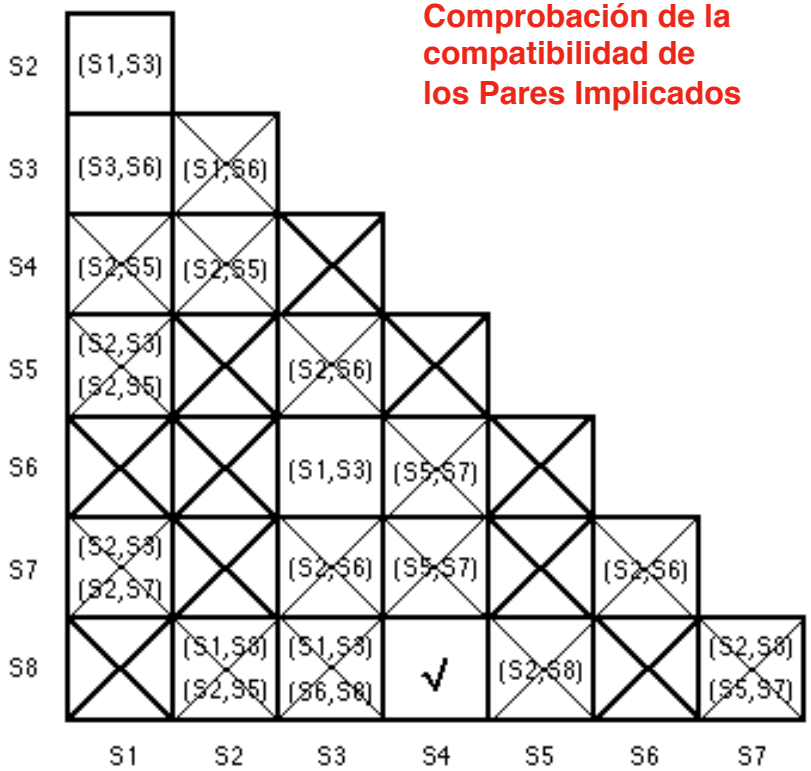
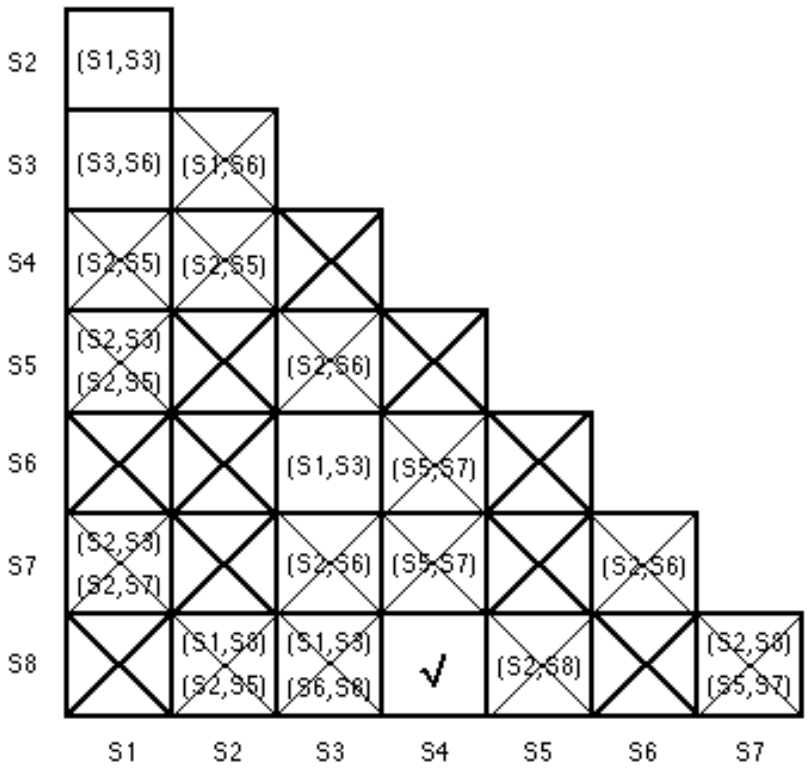


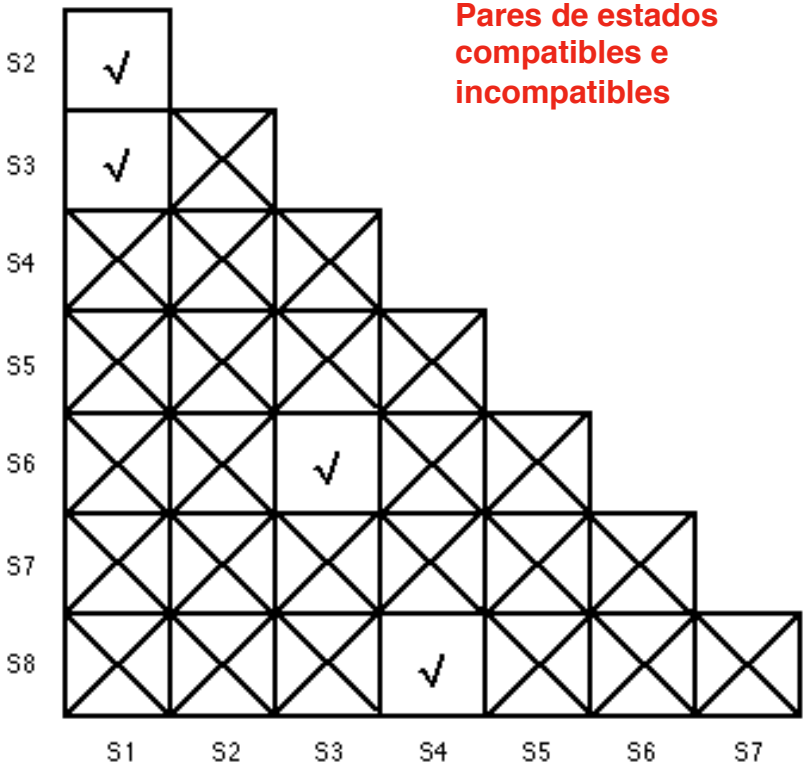
Tabla de Absorción (5)

Ejemplo (cont.)

Tabla de Absorción

Pares de estados compatibles e incompatibles





Compatibles Máximos

Procedimiento Tabular para obtener Compatibles Máximos

Paso 1 Comenzar por la columna más a la derecha de la Tabla de Absorción que contenga al menos un par compatible (celda con √). Listar los pares compatibles de la columna.

Paso 2 Desplazarse hacia la izquierda hasta la siguiente columna que tenga al menos un par compatible.

Comprobar si el estado S_m que encabeza la columna es compatible con todos o alguno de los estados de algún compatible previamente obtenido.

- Compatible con todos los de un compatible previo: añadir S_m a dicho compatible formando uno mayor.
- Compatible con algunos de los estados de un compatible previo: formar un nuevo compatible con esos estados y S_m.
- Listar todos los pares compatibles de la columna no incluidos en ningún compatible obtenido previamente.

Listar junto a los nuevos compatibles, los obtenidos previamente no incluidos en ellos.

Paso 3 Repetir el Paso 2 hasta finalizar con la columna más a la izquierda.

Paso 4 Si en el conjunto de compatibles obtenido falta algún estado de la Tabla de Estados, añadir este estado como un compatible.

Compatibles Máximos (2)

Procedimiento Tabular para obtener Compatibles Máximos

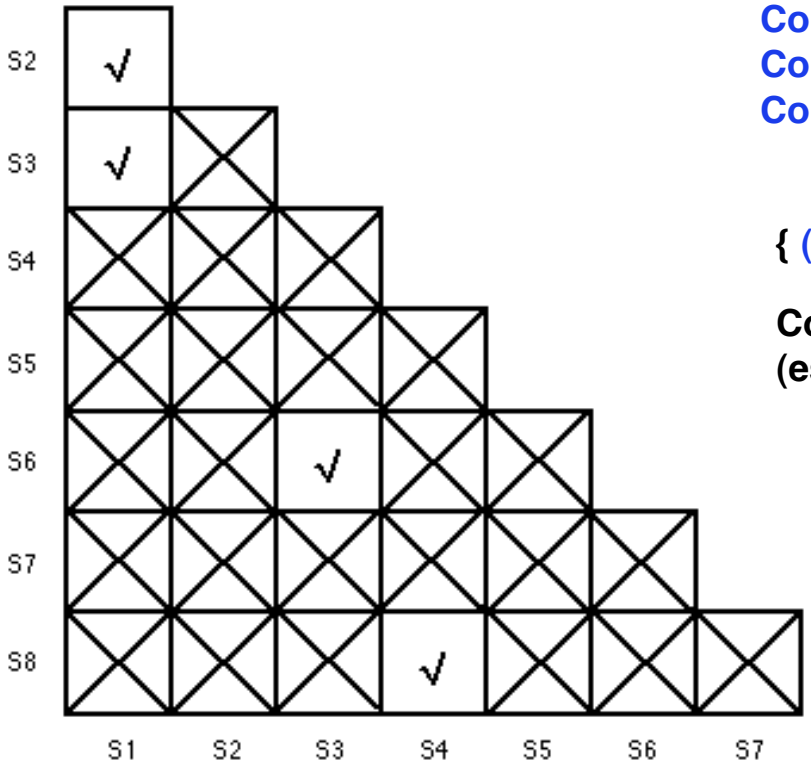
- **El conjunto formado por todos compatibles obtenidos constituye el Conjunto de Compatibles Máximos.**

- **El número total de compatibles obtenidos constituye el límite superior del número de estados de la máquina reducida equivalente a la original.**

- **El menor número de compatibles que cubre a todos los estados de la máquina original constituye el límite inferior del número de estados de la máquina reducida equivalente a la original.**

Compatibles Máximos (3)

Ejemplo



Columna S_4: (S_4,S_8)
Columna S_3: (S_3,S_6) (S_4,S_8)
Columna S_1: (S_1,S_2) (S_1,S_3) (S_3,S_6) (S_4,S_8)

{ (S_1,S_2), (S_1,S_3), (S_3,S_6), (S_4,S_8), (S_5), (S_7) }

Conjunto de Compatibles Máximos
(es una cobertura, no necesariamente cerrada)

{ (S_1,S_2), (S_1,S_3), (S_3,S_6), (S_4,S_8), (S_5), (S_7) }
Lim.Sup. = 6

{ (S_1,S_2), (S_1,S_3), (S_3,S_6), (S_4,S_8), (S_5), (S_7) }
Lim.Inf. = 5

La máquina reducida tiene 5 o 6 estados

Compatibles Máximos (4)

Ejercicio 1

Las Tablas de Estados que se muestran son las correspondientes a una máquina de tipo Mealy y su equivalente de tipo Moore (obtenida en el Tema 5 como ejemplo de conversión de tipo Mealy a tipo Moore). Probar que la maquina de tipo Moore no se puede reducir.

PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
A	B, 11	C, 00	A, 10	A, 01
B	A, 11	C, 00	B, 01	A, 01
C	C, 10	C, 00	A, 01	B, 01

Máquina de tipo Mealy original

PS	NS x_1 x_2 00	01	11	10	z_1 z_2
A_1	B_1	C_1	A_1	A_2	10
A_2	B_1	C_1	A_1	A_2	01
A_3	B_1	C_1	A_1	A_2	11
B_1	A_3	C_1	B_2	A_2	11
B_2	A_3	C_1	B_2	A_2	01
C_1	C_2	C_1	A_2	B_2	00
C_2	C_2	C_1	A_2	B_2	10

Máquina de tipo Moore equivalente

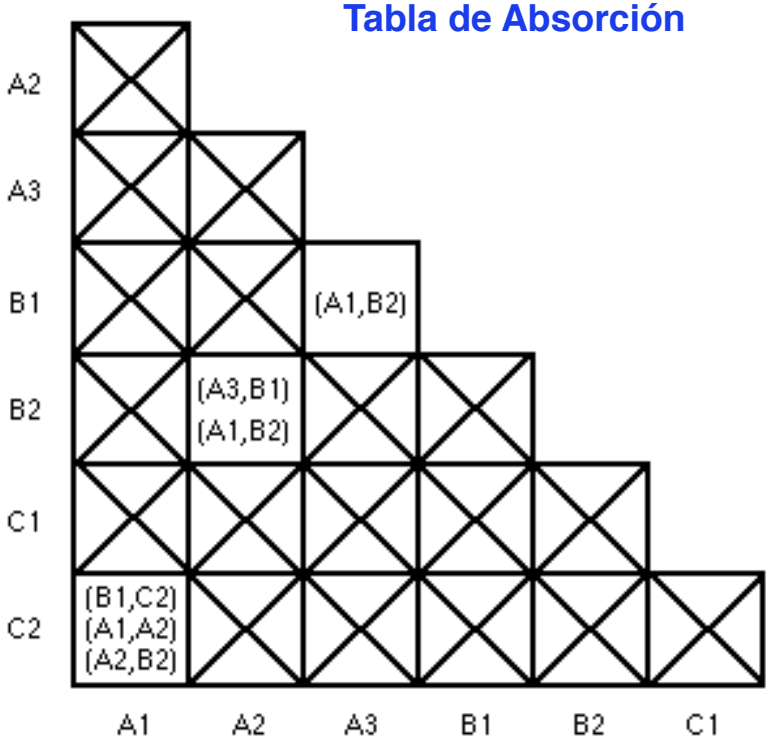
Compatibles Máximos (5)

Ejercicio 1

PS	NS $x_1 x_2$ 00	01	11	10	$z_1 z_2$
A_1	B_1	C_1	A_1	A_2	10
A_2	B_1	C_1	A_1	A_2	01
A_3	B_1	C_1	A_1	A_2	11
B_1	A_3	C_1	B_2	A_2	11
B_2	A_3	C_1	B_2	A_2	01
C_1	C_2	C_1	A_2	B_2	00
C_2	C_2	C_1	A_2	B_2	10

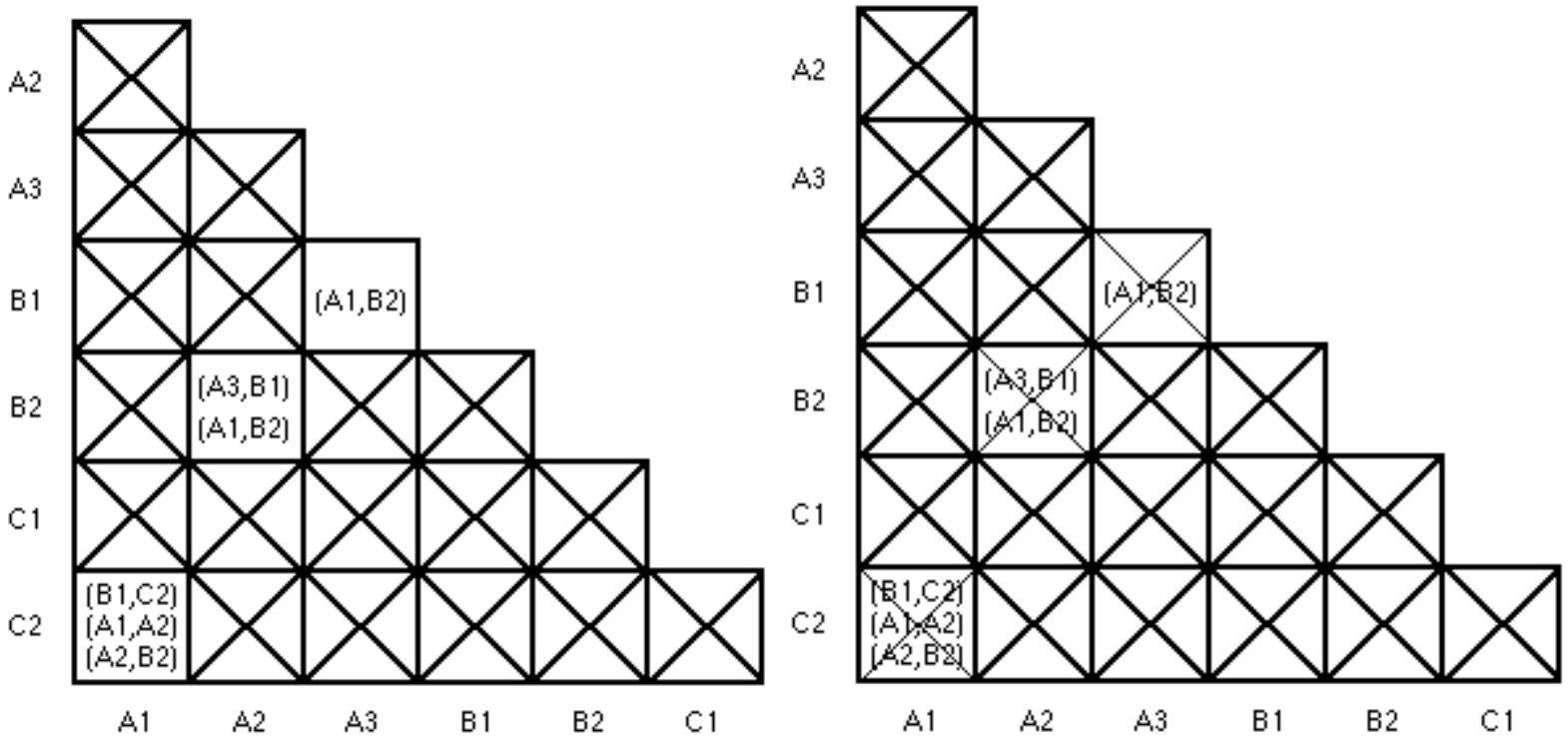
Máquina de tipo Moore equivalente

Tabla de Absorción



Compatibles Máximos (6)

Ejercicio 1



Conjunto de Compatibles Máximos { (A_1), (A_2), (A_3), (B_1), (B_2), (C_1), (C_2) } **=> No reducible**

Compatibles Máximos y Cobertura Cerrada

Ejercicio 2

Con respecto al ejercicio anterior. Convertir la máquina de tipo Moore a tipo Mealy. Minimizar la nueva Tabla de Estados, comprobando que se obtiene la Tabla de Estados original.

PS	NS $x_1\ x_2$ 00	01	11	10	$z_1\ z_2$
A_1	B_1	C_1	A_1	A_2	10
A_2	B_1	C_1	A_1	A_2	01
A_3	B_1	C_1	A_1	A_2	11
B_1	A_3	C_1	B_2	A_2	11
B_2	A_3	C_1	B_2	A_2	01
C_1	C_2	C_1	A_2	B_2	00
C_2	C_2	C_1	A_2	B_2	10

Compatibles Máximos y Cobertura Cerrada (2)

Ejercicio 2

PS	NS $x_1 x_2$ 00	01	11	10	$z_1 z_2$
A_1	B_1	C_1	A_1	A_2	10
A_2	B_1	C_1	A_1	A_2	01
A_3	B_1	C_1	A_1	A_2	11
B_1	A_3	C_1	B_2	A_2	11
B_2	A_3	C_1	B_2	A_2	01
C_1	C_2	C_1	A_2	B_2	00
C_2	C_2	C_1	A_2	B_2	10

PS	NS, $z_1 z_2$ $x_1 x_2$ 00	01	11	10
A_1	B_1,11	C_1,00	A_1,10	A_2,01
A_2	B_1,11	C_1,00	A_1,10	A_2,01
A_3	B_1,11	C_1,00	A_1,10	A_2,01
B_1	A_3,11	C_1,00	B_2,01	A_2,01
B_2	A_3,11	C_1,00	B_2,01	A_2,01
C_1	C_2,10	C_1,00	A_2,01	B_2,01
C_2	C_2,10	C_1,00	A_2,01	B_2,01

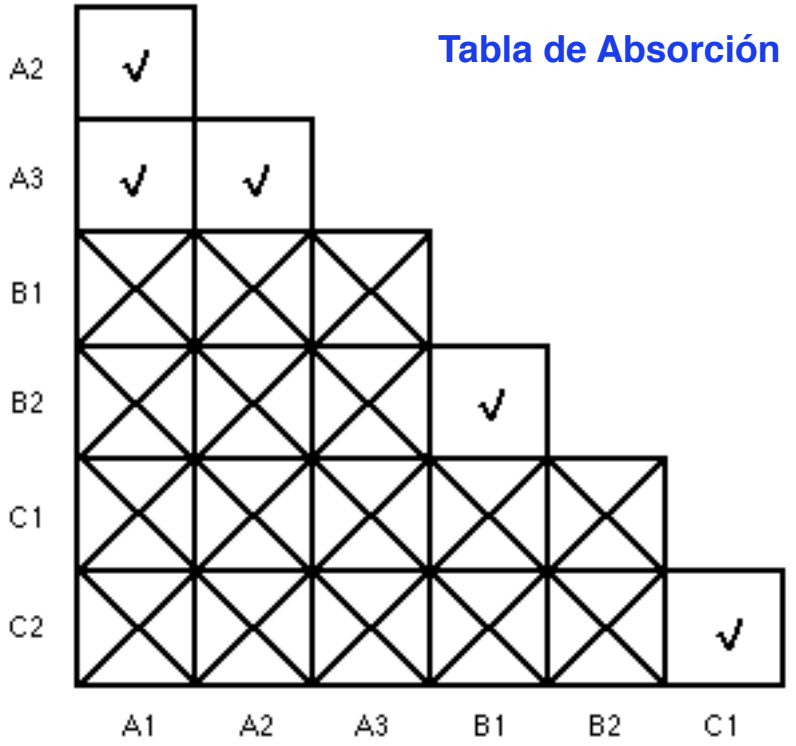
Tabla de de Mealy equivalente

Compatibles Máximos y Cobertura Cerrada (3)

Ejercicio 2

PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
A_1	B_1,11	C_1,00	A_1,10	A_2,01
A_2	B_1,11	C_1,00	A_1,10	A_2,01
A_3	B_1,11	C_1,00	A_1,10	A_2,01
B_1	A_3,11	C_1,00	B_2,01	A_2,01
B_2	A_3,11	C_1,00	B_2,01	A_2,01
C_1	C_2,10	C_1,00	A_2,01	B_2,01
C_2	C_2,10	C_1,00	A_2,01	B_2,01

Tabla de Absorción



Compatibles Máximos y Cobertura Cerrada (4)

Ejercicio 2

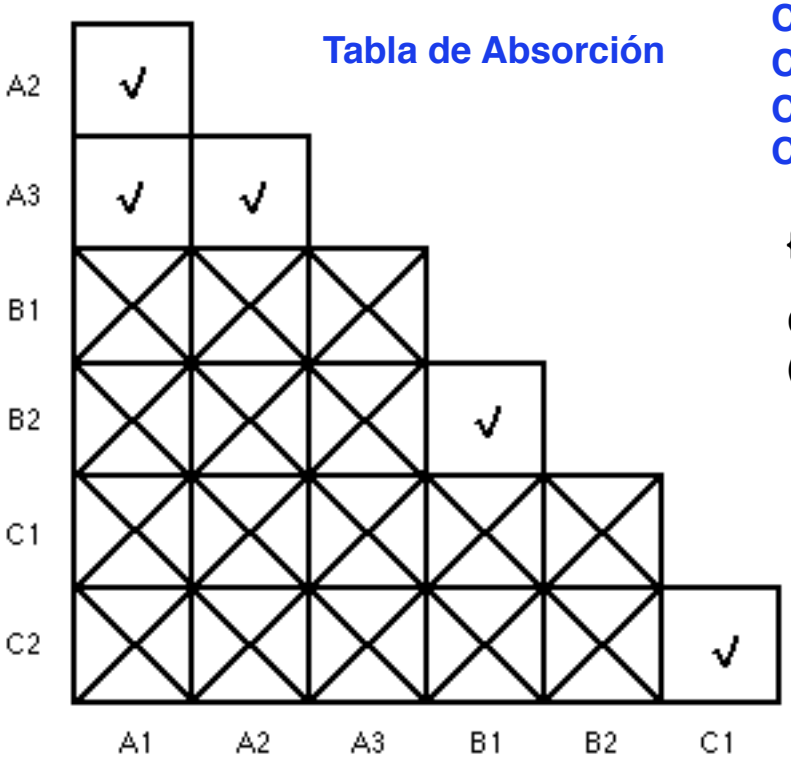


Tabla de Absorción

Columna C_1: (C_1,C_2)
Columna B_1: (B_1,B_2) (C_1,C_2)
Columna A_2: (A_2,A_3) (B_1,B_2) (C_1,C_2)
Columna A_1: (A_1,A_2,A_3) (B_1,B_2) (C_1,C_2)

{ (A_1,A_2,A_3), (B_1,B_2), (C_1,C_2) }

Conjunto de Compatibles Máximos
(es una cobertura, no necesariamente cerrada)

{ (A_1,A_2,A_3), (B_1,B_2), (C_1,C_2) }
Lim.Sup. = 3

{ (A_1,A_2,A_3), (B_1,B_2), (C_1,C_2) }
Lim.Inf. = 3

La máquina reducida tiene 3 estados

Dado que ninguno de los compatibles tiene implicados => Es cobertura cerrada

Compatibles Máximos y Cobertura Cerrada (5)

Ejercicio 2

{ (A_1,A_2,A_3), (B_1,B_2), (C_1,C_2) } **Es cobertura cerrada**

		A	B	C
	PS	NS, z_1 z_2 / x_1 x_2 = 00	01	11	10
A	A_1	B_1,11	C_1,00	A_1,10	A_2,01
A	A_2	B_1,11	C_1,00	A_1,10	A_2,01
A	A_3	B_1,11	C_1,00	A_1,10	A_2,01
B	B_1	A_3,11	C_1,00	B_2,01	A_2,01
B	B_2	A_3,11	C_1,00	B_2,01	A_2,01
C	C_1	C_2,10	C_1,00	A_2,01	B_2,01
C	C_2	C_2,10	C_1,00	A_2,01	B_2,01

PS	NS, z_1 z_2 / x_1 x_2 = 00	01	11	10
A	B, 11	C, 00	A, 10	A, 01
B	A, 11	C, 00	B, 01	A, 01
C	C, 10	C, 00	A, 01	B, 01

Máquina de tipo Mealy original

Cobertura Cerrada

Ejercicio 3

Encontrar una máquina reducida mínima equivalente a la máquina incompletamente especificada cuya Tabla de Estados es la que se muestra.

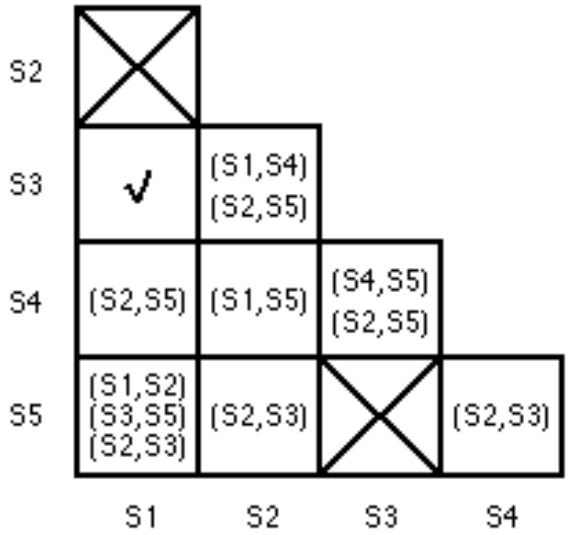
PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
S_1	S_1,-1	--,--	S_5,10	S_3,0-
S_2	S_3,01	S_1,1-	S_2,00	--,--
S_3	S_3,0-	S_4,1-	S_5,-0	S_1,01
S_4	--,--	S_5,1-	S_2,--	--,--
S_5	S_2,-1	--,--	S_3,--	S_2,-0

Cobertura Cerrada (2)

Ejercicio 3

PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
S_1	S_1,-1	--,--	S_5,10	S_3,0-
S_2	S_3,01	S_1,1-	S_2,00	--,--
S_3	S_3,0-	S_4,1-	S_5,-0	S_1,01
S_4	--,--	S_5,1-	S_2,--	--,--
S_5	S_2,-1	--,--	S_3,--	S_2,-0

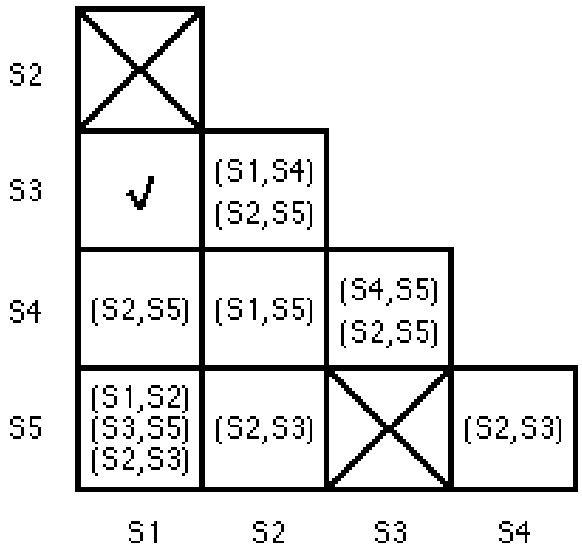
Tabla de Absorción

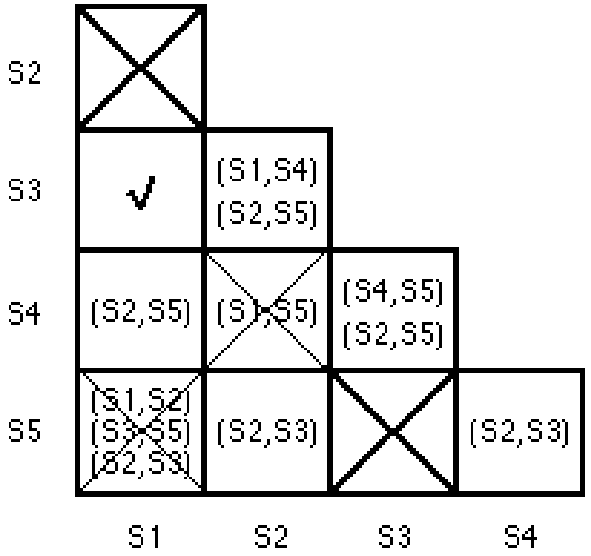


Cobertura Cerrada (3)

Ejercicio 3

Tabla de Absorción

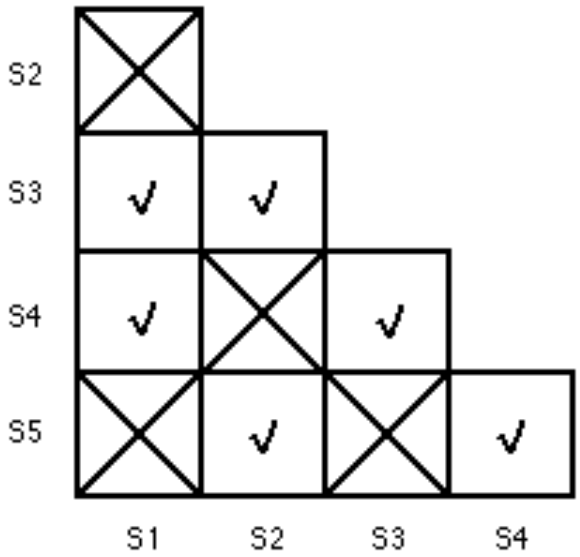




Cobertura Cerrada (4)

Ejercicio 3

Tabla de Absorción



Columna S_4: (S_4,S_5)
Columna S_3: (S_3,S_4) (S_4,S_5)
Columna S_2: (S_2,S_3) (S_2,S_5) (S_3,S_4) (S_4,S_5)
Columna S_1: (S_1,S_3,S_4) (S_2,S_3) (S_2,S_5) (S_4,S_5)

{ (S_1,S_3,S_4), (S_2,S_3), (S_2,S_5), (S_4,S_5) }

Conjunto de Compatibles Máximos
(es una cobertura, no necesariamente cerrada)

{ (S_1,S_3,S_4), (S_2,S_3), (S_2,S_5), (S_4,S_5) }
Lim.Sup. = 4

{ (S_1,S_3,S_4), (S_2,S_5) }
Lim.Inf. = 2

Los compatibles son de tamaño 3 estados (1) y 2 estados (3). Para cubrir los 5 estados de la máquina original con 2 compatibles => Uno de ellos debe ser el (único) de 3 estados

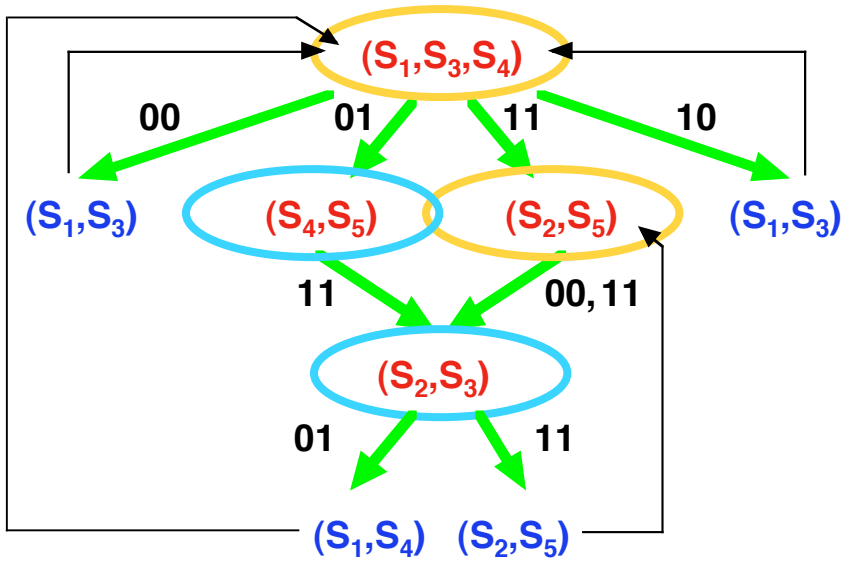
Cobertura Cerrada (5)

Ejercicio 3

La cobertura deberá ser: { (S_1,S_3,S_4), (S_2,S_5) }

Observando la Tabla de Estados:

PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
S_1	S_1,-1	--,--	S_5,10	S_3,0-
S_2	S_3,01	S_1,1-	S_2,00	--,--
S_3	S_3,0-	S_4,1-	S_5,-0	S_1,01
S_4	--,--	S_5,1-	S_2,--	--,--
S_5	S_2,-1	--,--	S_3,--	S_2,-0



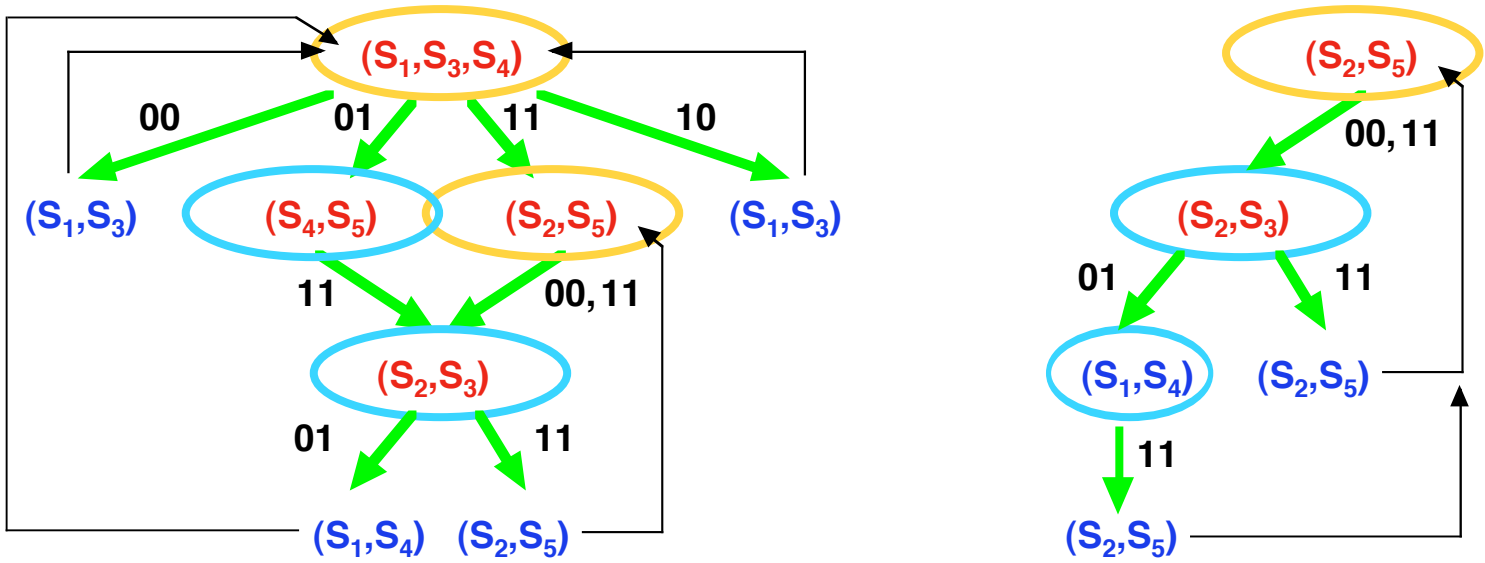
Con los compatibles { (S_1,S_3,S_4), (S_2,S_5) }, la obtención de una cobertura cerrada => tomar 4 compatibles { (S_1,S_3,S_4), (S_2,S_3), (S_2,S_5), (S_4,S_5) } (todos). En consecuencia:

- **No existe una cobertura cerrada con 2 compatibles.**
- **Si existe una cobertura cerrada de 3 compatibles, entonces (S_1,S_3,S_4) no está en ella**

Cobertura Cerrada (6)

Ejercicio 3

Observando el árbol de compatibles anterior:



El conjunto de compatibles { (S_1,S_4), (S_2,S_3), (S_2,S_5) } forma una cobertura cerrada de 3 compatibles (que es mínima)

Cobertura Cerrada (7)

Ejercicio 3

Cobertura cerrada mínima { (S_1,S_4), (S_2,S_3), (S_2,S_5) }

A B C

PS	NS, z_1 z_2 / x_1 x_2 00	01	11	10
S_1	S_1,-1	--,--	S_5,10	S_3,0-
S_2	S_3,01	S_1,1-	S_2,00	--,--
S_3	S_3,0-	S_4,1-	S_5,-0	S_1,01
S_4	--,--	S_5,1-	S_2,--	--,--
S_5	S_2,-1	--,--	S_3,--	S_2,-0

Máquina incompletamente especificada original

PS	NS, z_1 z_2 / x_1 x_2 00	01	11	10
A	A, -1	C, 1-	C, 10	B, 0-
B	B, 01	A, 1-	C, 00	A, 01
C	B, 01	A, 1-	B, 00	B/C, -0

Máquina mínima incompletamente especificada resultante

El conjunto de máquinas resultantes (2^7, según los valores elegidos para cada "-" y para B/C) son las máquinas mínimas compatibles con la descripción de la máquina original

Asignación de Estados

Asignación de Estados (Asignación de Variables de Estado, Asignación Secundaria)

Estado simbólico <=> Conjunto específico de valores de las variables de estado

Coste de la implementación de un circuito secuencial síncrono

- ◊ **Asignación de Variables de Estado**
- ◊ **Tipo de fip-flops**
  - **Imposible saber *a priori* que tipo de FFs darán lugar al menor coste**
  - **En general:**
    - **VLSI: D-FF**
      - **Área**
      - **Conexiones (igual o mayor incidencia en el área ocupada que los propios dispositivos)**
      - **Carácterísticas (capacidad de puesta a 0 y a 1, retención y complementación del estado previo) compensan la necesidad de generar funciones para las 2 entradas con una mayor simplicidad**

Asignación de Estados

NE **#Estados**
n **#Variables de estado (FFs)** $n = \lceil \log_2 NE \rceil$,, $NE \leq 2^n$

Combinaciones de 2^n patrones binarios tomados en grupos de NE: $\frac{2^n!}{NE! \times (2^n - NE)!}$

Es posible escoger 2^n patrones binarios distintos para el primer estado, 2^n-1 para el segundo, 2^n-2 para el tercero, … (2^n - (NE-1)) para el NE-ésimo => El número de patrones distintos que se pueden escoger para los NE estados es: 2^n x (2^n-1) x [(2^n-2) x … x [(2^n-(NE-1)]

Posibles Asignaciones: $PA = \frac{2^n!}{(2^n - NE)!}$

n = 3
NE = 5 **PA = 6720**

n = 3
NE = 8 **PA = 40320**

n = 4
NE = 9 **PA ≈ 4,15. 10^9**

n = 4
NE = 16 **PA ≈ 2,09 . 10^{13}** **(a 10ns por asignamiento: 2,42 días, > 58 h)**

Asignación de Estados (2)

Ejemplo

Encontrar todas la posibles asignaciones para 3 estados: A, B y C.

$$n = \lceil \log_2 NE \rceil = \lceil \log_2 3 \rceil = 2$$

El número de combinaciones distintas de 2^2 patrones binarios tomados de 3 en 3:

$$\frac{2^n!}{NE! \times (2^n - NE)!} = \frac{2^2!}{3! \times (4-3)!} = 4$$

	y_1 y_0
A	0 0
B	0 1
C	1 0

	y_1 y_0
A	0 0
B	0 1
C	1 1

	y_1 y_0
A	0 0
B	1 0
C	1 1

	y_1 y_0
A	0 1
B	1 0
C	1 1

Posibles Asignaciones: En cada una de estos grupos de patrones se pueden permutar los asignamientos de 3!= 6 formas diferentes:

$$PA = \frac{2^n!}{(2^n - NE)!} = \frac{2^2!}{(2^2 - 3)!} = 4! = 24$$

Asignación de Estados (3)

Ejemplo

	y_1 y_0
A	0 0
B	0 1
C	1 0

	y_1 y_0
A	0 0
B	1 0
C	0 1

	y_1 y_0
A	0 1
B	0 0
C	1 0

	y_1 y_0
A	0 1
B	1 0
C	0 0

	y_1 y_0
A	1 0
B	0 0
C	0 1

	y_1 y_0
A	1 0
B	0 1
C	0 0

	y_1 y_0
A	0 0
B	0 1
C	1 1

	y_1 y_0
A	0 0
B	1 1
C	0 1

	y_1 y_0
A	0 1
B	0 0
C	1 1

	y_1 y_0
A	0 1
B	1 1
C	0 0

	y_1 y_0
A	1 1
B	0 0
C	0 1

	y_1 y_0
A	1 1
B	0 1
C	0 0

	y_1 y_0
A	0 0
B	1 0
C	1 1

	y_1 y_0
A	0 0
B	1 1
C	1 0

	y_1 y_0
A	1 0
B	0 0
C	1 1

	y_1 y_0
A	1 0
B	1 1
C	0 0

	y_1 y_0
A	1 1
B	0 0
C	1 0

	y_1 y_0
A	1 1
B	1 0
C	0 0

	y_1 y_0
A	0 1
B	1 0
C	1 1

	y_1 y_0
A	0 1
B	1 1
C	1 0

	y_1 y_0
A	1 0
B	0 1
C	1 1

	y_1 y_0
A	1 0
B	1 1
C	0 1

	y_1 y_0
A	1 1
B	0 1
C	1 0

	y_1 y_0
A	1 1
B	1 0
C	0 1

Todos diferentes

Asignamientos de Distinto Coste

No todos los asignamientos pueden dar lugar a implementaciones de distinto coste:

- ◊ **Complemento de una "columna" (valores asignados a una variable en los distintos estados)**
  - **Conectar a los nudos de los DECs y^*_k en lugar de y_k y viceversa (los FFs disponen de salidas complementadas y no complementadas)**
  - **(En el caso de D-FFs por ejemplo) Generar Y^*_k en lugar de Y_k y viceversa para la conexión a D_k**
- ◊ **Intercambio de "columnas" entre 2 variables y_p e y_q**
  - **Equivale a reetiquetar las variables o cambiar de "posición" los FFs**

Es posible demostrar que el número de asignamientos distintos que pueden dar lugar a distinto coste (asignamientos únicos, UA):

$$UA = \frac{PA}{2^n \times n!} = \frac{2^n!}{(2^n - NE)! \times 2^n \times n!} = \frac{(2^n - 1)!}{(2^n - NE)! \times n!}$$

Métodos CAD exactos: SHR
Métodos (CAD) aproximados

Asignación de Estados: Aproximaciones

Coste de la implementación circuital:

- **Coste de los DECs de estado próximo (atención preferente) y de salida**
- **Coste de los DECs se cuantifica en número de puertas y número de entradas a las mismas**

Diversos métodos orientados a encontrar asignaciones cuasi-óptimas o simplemente a conseguir la simplicidad de los DECs:

- **Basados en obtener adyacencias entre estados, que tratan de conseguir que los valores asignados a las variables de estado próximo puedan ser agrupados, en sus M-Ks, en cubos del mayor tamaño posible (términos producto o factores suma con el menor número posible de literales)**
  Tratan de seguir un número reducido de directivas o reglas, con distintas prioridades

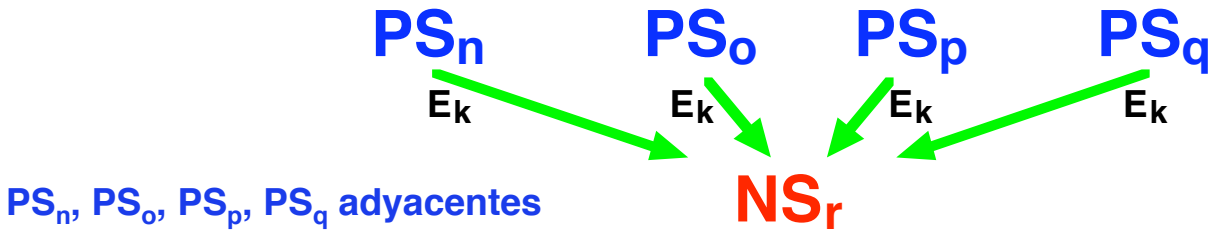
- **Basados en establecer particiones entre los estados de la máquina; tratan de conseguir que las variables de estado próximo tengan una dependencia reducida de las PIs**

- **Asignamientos especiales, muy sencillos de obtener. Emplean mayor número de FFs que el mínimo requerido, tratando de conseguir a cambio una mayor simplicidad de las expresiones para las variables de estado próximo y de salida ("one-hot")**

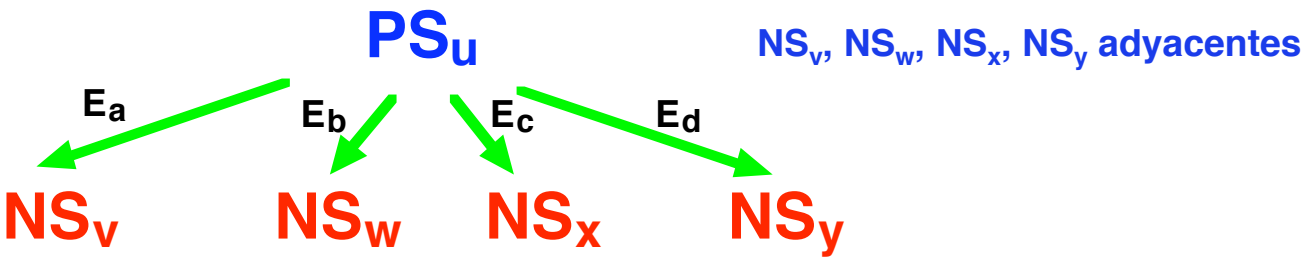
Directivas para la Asignación de Estados

DEC de Estado Próximo:

D1. **Aquellos estados (PSs) que, para una misma condición de entrada, tengan el mismo estado próximo (NS) deberían tener asignaciones adyacentes. (Analizar en cada columna de la Tabla de Estados)**



D2. **Aquellos estados que son los estados próximos (NSs) de un mismo estado (PS), para las distintas condiciones de entrada, deberían tener asignaciones adyacentes. (Analizar en cada fila de la Tabla de Estados)**

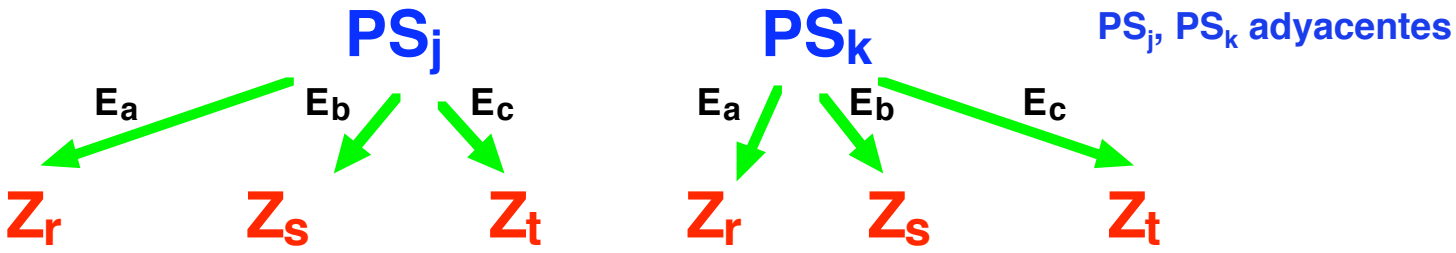


D1 y D2 tratan de minimizar las expresiones (agrupar 1s en los M-Ks, formando cubos tan grandes como sea posible) de las variables de estado próximo.

Directivas para la Asignación de Estados (2)

DEC de Salida:

D3. **Aquellos estados (PSs) que, para cada condición de entrada, tienen las mismas salidas (donde están definidas), deberían tener asignaciones adyacentes.**
(Analizar en la Tabla de Estados cada columna para toda fila)



- **D1, D2 y D3 tratan de minimizar las expresiones (agrupar 1s en los M-Ks, formando cubos tan grandes como sea posible) de las variables de estado próximo**
- **Como norma general D1 tiene preferencia sobre D2, y esta sobre D3. El número de veces que una determinada adyacencia sea sugerida puede cambiar la relación de preferencia anterior**
- **Estas reglas dan lugar generalmente a buenos resultados cuando se utilizan FFs de los tipos D o JK; no son tan adecuadas cuando se emplean FFs de los tipos SR o T**

Asignación Cuasi-Óptima

Ejemplo

Encontrar una asignación de estados cuasi-óptima para el circuito cuya Tabla de Estados se adjunta. Obtener el coste de la implementación con FFs de tipo D en función del número de puertas y número de entradas a las mismas. Suponer que se dispone de puertas AND y OR con cualquier número de entradas y de las PIs en su forma complementada y no complementada y considerar minimización individual de las funciones de excitación y de salida. No tener en cuenta la posibilidad de compartir términos producto parcial o totalmente.

PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
A	A, -1	C, 1-	B, 10	C, 01
B	B, 01	D, 1-	D, 00	B, 01
C	B, 01	A, 1-	B, 0-	C, 10
D	D, 01	B, 1-	B, 01	D, -0

Asignación Cuasi-Óptima (2)

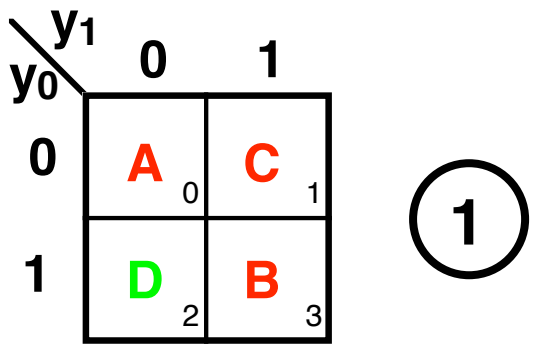
PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
A	A, -1	C, 1-	B, 10	C, 01
B	B, 01	D, 1-	D, 00	B, 01
C	B, 01	A, 1-	B, 0-	C, 10
D	D, 01	B, 1-	B, 01	D, -0

	00	11	10	
D1:	(BC)	(ACD)	(AC)	
	A	B	C	D
D2:	(ABC)	(BD)	(ABC)	(BD)

D3: (CD)

D1: (BC) (AD) (AC)x2 (CD)
D2: (AB)x2 (AC)x2 (BC)x2 (BD)x2
D3: (CD)

D1: (BC) (AD) (AC)x2 (CD)
D2: (AB)x2 (AC)x2 (BC)x2 (BD)x2
D3: (CD)

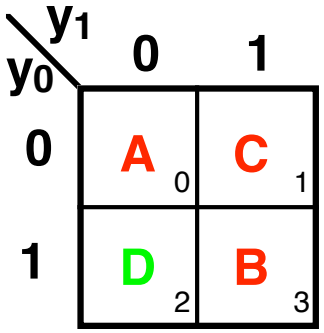


1

Asignación Cuasi-Óptima (3)

D1: (BC) (AD) (AC)x2 (CD)
D2: (AB)x2 (AC)x2 (BC)x2 (BD)x2
D3: (CD)

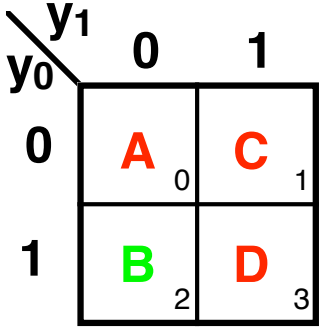
Cumple la mayor parte de las directivas



1

D1: (BC) (AD) (AC)x2 (CD)
D2: (AB)x2 (AC)x2 (BC)x2 (BD)x2
D3: (CD)

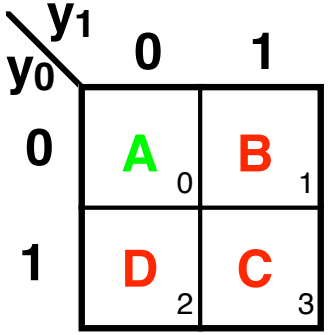
Cumple la mayor parte de las directivas, en menor grado que la anterior, pero se cumple D3



2

D1: (BC) (AD) (AC)x2 (CD)
D2: (AB)x2 (AC)x2 (BC)x2 (BD)x2
D3: (CD)

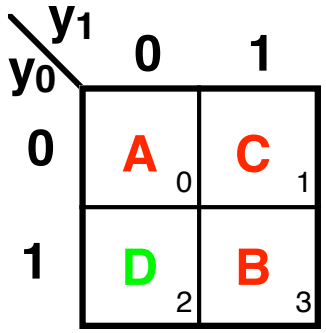
Cumple la mayor parte de las directivas en menor grado que la primera, pero se cumple D3



3

Asignación Cuasi-Óptima (4)

D1: (BC) (AD) (AC)x2 (CD)
D2: (AB)x2 (AC)x2 (BC)x2 (BD)x2
D3: (CD)



1

PS	NS, z_1 z_2 x_1 x_2 00	01	11	10
A	A, -1	C, 1-	B, 10	C, 01
B	B, 01	D, 1-	D, 00	B, 01
C	B, 01	A, 1-	B, 0-	C, 10
D	D, 01	B, 1-	B, 01	D, -0

Tabla de Estados

y_1y_0	Y_1Y_0, z_1 z_2 x_1 x_2 00	01	11	10
00	00, -1	10, 1-	11, 10	10, 01
11	11, 01	01, 1-	01, 00	11, 01
10	11, 01	00, 1-	11, 0-	10, 10
01	01, 01	11, 1-	11, 01	01, -0

Tabla de Transición

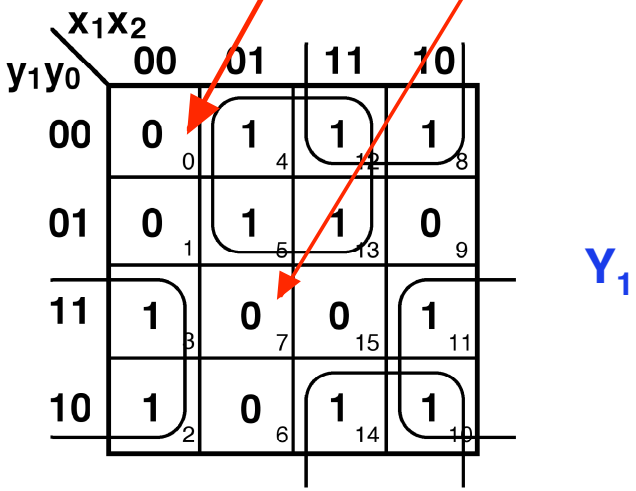
Asignación Cuasi-Óptima (5)

y_1y_0	$Y_1Y_0, z_1\ z_2$ $x_1\ x_2$ 00	01	11	10
00	00, -1	10, 1-	11, 10	10, 01
11	11, 01	01, 1-	01, 00	11, 01
10	11, 01	00, 1-	11, 0-	10, 10
01	01, 01	11, 1-	11, 01	01, -0

Tabla de Transición

y_1y_0	$D_1D_0, z_1\ z_2$ $x_1\ x_2$ 00	01	11	10
00	00, -1	10, 1-	11, 10	10, 01
11	11, 01	01, 1-	01, 00	11, 01
10	11, 01	00, 1-	11, 0-	10, 10
01	01, 01	11, 1-	11, 01	01, -0

Tabla de Excitación (D-FFs)

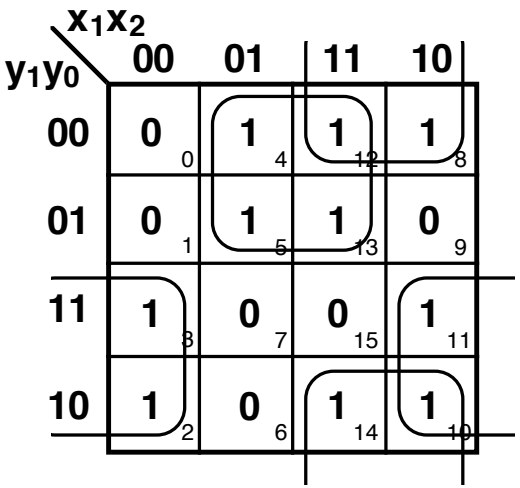


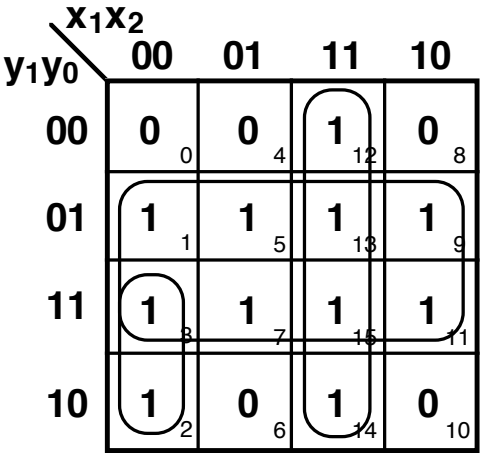
Y_1

Asignación Cuasi-Óptima (6)

$D_1 = x_2 \cdot /y_1 + x_1 \cdot /y_0 + /x_2 \cdot y_1$

4 p., 9 e.



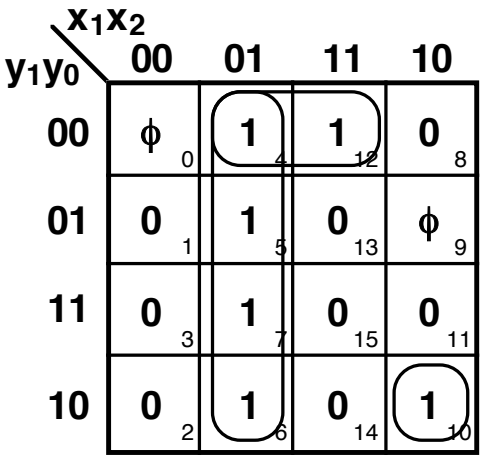


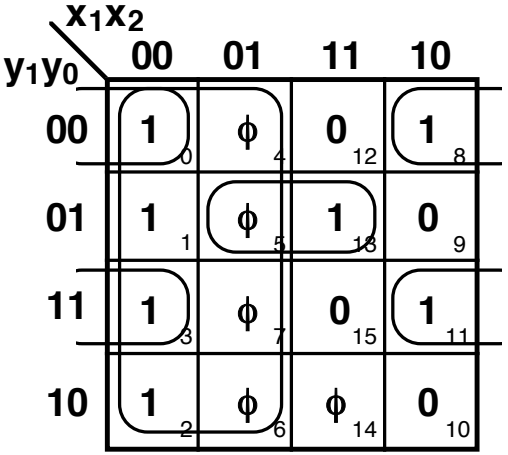
$D_0 = x_1 \cdot x_2 + /x_1 \cdot /x_2 \cdot y_1 + y_0$

3 p., 8 e.

$z_1 = /x_1 \cdot x_2 + x_2 \cdot /y_1 \cdot /y_0 + x_1 \cdot /x_2 \cdot y_1 \cdot /y_0$

4 p., 12 e.





$z_2 = /x_1 + /x_2 \cdot /y_1 \cdot /y_0 + x_2 \cdot /y_1 \cdot y_0 + /x_2 \cdot y_1 \cdot y_0$

4 p., 13 e.

Coste Total: 15 puertas, 42 entradas

Asignación Cuasi-Óptima (7)

Ejercicio 4

Comprobar que las asignaciones 2 y 3 propuestas en el ejemplo anterior dan lugar a los siguientes resultados:

2:

$D_1 = /x_1 \cdot x_2 \cdot /y_1 + x_1 \cdot /x_2 \cdot /y_0 + x_2 \cdot /y_1 \cdot y_0 + /x_2 \cdot y_1 \cdot y_0$ — 5 p., 16 e.

$D_0 = x_1 \cdot x_2 + /x_1 \cdot /x_2 \cdot y_1 + y_0$ — 3 p., 8 e.

$z_1 = /x_1 \cdot x_2 + x_2 \cdot /y_1 \cdot /y_0 + x_1 \cdot /x_2 \cdot y_1$ — 4 p., 11 e.

$z_2 = /x_1 + /x_2 \cdot /y_1 + x_2 \cdot y_1$ — 3 p., 7 e.

Coste Total: 15 puertas, 42 entradas

Mismo coste total que 1, pero considerable reducción del DEC de Salida (7 p., 18 e., frente a 8 p., 25 e. del 1) debido al cumplimiento de D3.

3:

$D_1 = x_2 \cdot /y_1 + /x_2 \cdot /y_1 + x_1 \cdot /y_1 \cdot /y_0 + x_1 \cdot y_1 \cdot y_0$
$x_2 \cdot /y_1 + /x_2 \cdot /y_1 + x_1 \cdot /x_2 \cdot /y_0 + x_1 \cdot x_2 \cdot y_0$ — 5 p., 14 e.

$D_0 = /x_1 \cdot x_2 \cdot /y_0 + x_1 \cdot /x_2 \cdot /y_1 + /x_2 \cdot /y_1 \cdot y_0 + x_1 \cdot /x_2 \cdot y_0 + x_2 \cdot y_1 \cdot /y_0$ — 6 p., 20 e.

$z_1 = /x_2 \cdot /y_1 \cdot /y_0 + /x_1 \cdot x_2 + x_1 \cdot /x_2 \cdot y_0$ — 4 p., 11 e.

$z_2 = /x_1 + /x_2 \cdot /y_0 + x_2 \cdot y_0$ — 3 p., 7 e.

Coste Total: 18 puertas, 52 entradas

Mayor coste total que 1 y 2, pero mantiene la reducción del DEC de Salida (7 p., 18 e., frente a 8 p., 25 e. del 1) debido al cumplimiento de D3.

Descripción VHDL de Tablas de Estados

Unidad de Control

Codificación de estados y síntesis

- ◊ **La herramienta CAD de síntesis elige la codificación**
  - **Cuenta binaria natural**
    Ejemplo (4 estados): y_1y_0: 00, 01, 10, 11
  - **Codificación binaria arbitraria**
    Ejemplo (4 estados): y_1y_0: 01, 00, 11, 10
  - **Codificación "one-hot"**
    Ejemplo (4 estados): $y_3y_2y_1y_0$: 0001, 0010, 0100, 1000

- ◊ **El diseñador especifica a la herramienta CAD la asignación de estados**
  - **Se especifican explícitamente los valores de todas las variables de estado para cada estado**

- ◊ **Algunas herramientas CAD ignoran la asignación especificada por el diseñador si reconocen una Máquina de Estados en la descripción**

Descripción VHDL circuito sec. síncrono

Descripción “behavioral” algorítmica

Ejemplo: Contador Up/Down

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

```
entity contador_up_down is
   port  (   clk, en, up_down : in std_logic;
             async_reset  :   in std_logic;
             out_cont   :   out std_logic_vector (3 downto 0));
end contador_up_down;

architecture behav of contador_up_down is
signal   cuenta  :   std_logic_vector (3 downto 0);
signal   data  :   integer;
begin
process (clk, async_reset)
   begin
      if (async_reset = ‘0’) then
         data <= 0;
      elsif (clk’event and clk = ‘1’) then
         if (en = ‘0’) then
            if (up_down = ‘1’) then
               data <= to_integer(unsigned(cuenta)) +1;
            elsif (up_down = ‘0’) then                          --else
               data <= to_integer(unsigned(cuenta)) -1;
            end if;
         end if;
      end if;
end process;
cuenta <= std_logic_vector(to_unsigned(data,4));
out_cont <= cuenta;
end behav;
```

Compilado con Quartus II

Descripción VHDL circuito sec. Síncrono (2)

Descripción "behavioral" algorítmica

Ejemplo: Contador Up/Down

```
entity contador_up_down is
   port  (   clk, en, up_down : in std_logic;
             async_reset  :   in std_logic;
             out_cont   :   out std_logic_vector (3 downto 0));
end contador_up_down;

architecture behav of contador_up_down is
signal   cuenta  :   std_logic_vector (3 downto 0);
begin
process (clk, async_reset)
   begin
      if (async_reset = '0') then
         cuenta <= "0000";
      elsif (clk'event and clk = '1') then
         if (en = '0') then
            if (up_down = '1') then
               cuenta <= std_logic_vector(to_unsigned(to_integer(unsigned(cuenta))+1),4));
            elsif (up_down = '0') then                                    --else
               cuenta <= std_logic_vector(to_unsigned(to_integer(unsigned(cuenta))-1),4));
            end if;
         end if;
      end if;
end process;
out_count <= cuenta;
end behav;
```

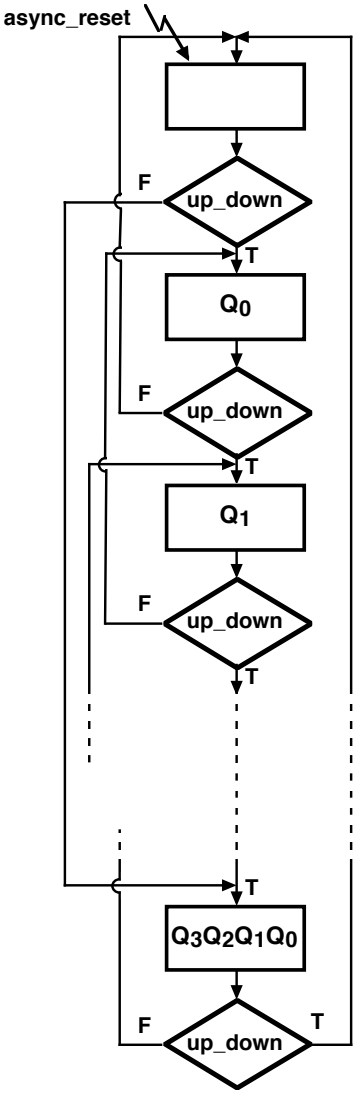
Compilado con Quartus II

Descripción VHDL de Tablas de Estados

Descripción "behavioral" Mediante Tablas de Estado

Ejemplo:
Contador Up/Down

PS	NS up_down 0	NS up_down 1	out_cont $Q_3Q_2Q_1Q_0$
a	p	b	0 0 0 0
b	a	c	0 0 0 1
c	b	d	0 0 1 0
d	c	e	0 0 1 1
e	d	f	0 1 0 0
f	e	g	0 1 0 1
g	f	h	0 1 1 0
h	g	i	0 1 1 1
i	h	j	1 0 0 0
j	i	k	1 0 0 1
k	j	l	1 0 1 0
l	k	m	1 0 1 1
m	l	n	1 1 0 0
n	m	o	1 1 0 1
o	n	p	1 1 1 0
p	o	a	1 1 1 1



Descripción VHDL de Tablas de Estados

Descripción “behavioral”
Mediante Tablas de Estado

Ejemplo: Contador Up/Down

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

entity contador_up_down is
    port  (clk, en, up_down :   in std_logic;
           async_reset  :   in std_logic;
           out_cont   :   out std_logic_vector (3 downto 0);
end contador_up_down;
```

Salida correspondiente
al estado actual

El el arranque se inicializa al estado de reset (si
no existe, al primero de la declaración type)

Compilado con Quartus II

```
architecture behav_tabla_est of contador_up_down is
type   estado_circ is (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p);
signal estado :  estado_circ;
begin
process (clk, async_reset)
    begin
        if (async_reset = ‘0’) then
            estado <= a;
        elsif (clk’event and clk = ‘1’) then
            if (en = ‘0’) then
                if (up_down = ‘1’) then
                    case estado is
                        when a => estado <= b;
                        when b => estado <= c;
                        …
                        when p => estado <= a;
                    end case;
                elsif (up_down = ‘0’) then
                    case estado is
                        when a => estado <= p;
                        when b => estado <= a;
                        …
                        when p => estado <= o;
                    end case;
                end if;
            end if;
        end if;                                          --end process;
        case estado is                                      --process (estado)
            when a => out_cont <= “0000”;                   --begin
            …
            when p => out_cont <= “1111”;
        end case;
end process;
end behav_tabla_est;
```

Descripción VHDL de Tablas de Estados (2)

Descripción "behavioral"
Mediante Tablas de Estado

Ejemplo: Contador Up/Down

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

entity contador_up_down is
    port  (clk, en, up_down :   in std_logic;
           async_reset  :   in std_logic;
           out_cont   :   out std_logic_vector (3 downto 0);
end contador_up_down;
```

Asignación: a->0011, b->1001, …, p->0111. ¿Ignorada?

attribute enum_encoding puede no ser reconocido por algunas herramientas VHDL

Compilado con Quartus II

```
architecture behav_tabla_est of contador_up_down is
type   estado_circ is (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p);
attribute enum_encoding :   string;
attribute enum_encoding of estado_circ : type is "0011 1001 … 0111";
signal estado :  estado_circ;
begin
process (clk, async_reset)              --PROCESS SECUENCIAL
    begin
        if (async_reset = '0') then
            estado <= a;
        elsif (clk'event and clk = '1') then
            if (en = '0') then
                if (up_down = '1') then
                    case estado is
                        when a => estado <= b;
                        …
                        when p => estado <= a;
                    end case;
                elsif (up_down = '0') then
                    case estado is
                        when a => estado <= p;
                        …
                        when p => estado <= o;
                    end case;
                end if;
            end if;
        end if;
end process;                            --PROCESS COMBINACIONAL
case estado is
    when a => out_cont <= "0000";
    …
    when p => out_cont <= "1111";
end case;
end behav_tabla_est;
```

Descripción VHDL de Tablas de Estados (3)

Algunas herramientas VHDL de síntesis, si la descripción es reconocida como FSM (process que realiza la asignación de NS con clk en su lista sensitiva), realizan un procesado específico tratando de encontrar una asignación óptima (ocupación, retardos, …)

Por ejemplo, Max+Plus II si reconoce una FSM, ignora la asignación proporcionada por el usuario con attribute enum_encoding, realizando la asignación automática por omisión: Mínimo número de Bits (FFs), por ej.: "0000 0001 … 1111"

Quartus II permite seleccionar ("settings" de compilación) el procesado de la FSM:

- **Auto:**
  - **FPGAs: 'One-Hot' modificada (asigna al primer estado "todo 0s" y los demás con dos 1s). Ej.: 0000, 0011, 0101, 1001. #bits=#estados**
  - **CPLDs: Codificación con el mínimo número de bits. Ej.: 00, 01, 10, 11**
- **Minimal Bits: Codificación con el mínimo número de bits. Ej.: 00, 01, 10, 11**
- **'One-Hot': 'One-Hot' modificada**
- **User Defined: Propuesta por el usuario:**
  - **Enumerada como 'string'. Ej.: "10 11 00 01"**
  - **"sequential", "gray", "johnson"**

Cuando la descripción es identificada como FSM, usualmente consigue obtener una asignación más adecuada (menor ocupación, menores retrasos, etc.) en la mayor parte de los casos
Si no es identificada se procesa como lógica "regular" (combinación de lógica y FFs)

Descripción VHDL de Tablas de Estados (4)

Descripción "behavioral"
Mediante Tablas de Estado

Ejemplo: Contador Up/Down

Asignación realizada: Propuesta por el diseñador

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

entity contador_up_down is
    port    (clk, en, up_down :     in std_logic;
             async_reset :    in std_logic;
             out_cont :    out std_logic_vector (3 downto 0);
end contador_up_down;
```

Salida correspondiente
al estado presente

process que realiza la asignación de
NS no tiene a clk en su lista sensitiva

En Quartus II, en Analysis & Synthesis Settings seleccionar
Auto, Minimal Bits, One-Hot o User Defined

Compilado con Quartus II

```
architecture behav_tabla_est of contador_up_down is
type   estado is (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p);
attribute enum_encoding :   string;
attribute enum_encoding of estado : type is "0011 1001 … 0111";
signal est_presente, proximo_est : estado;
begin
process (est_presente, up_down)       --process combinacional
    begin
        if (up_down = '1') then
            case est_presente is
                when a => proximo_est <= b; out_cont <= "0000";
                …
                when p => proximo_est <= a; out_cont <= "1111";
            end case;
        else
            case est_presente is
                when a => proximo_est <= p; out_cont <= "0000";
                …
                when p => proximo_est <= o; out_cont <= "1111";
            end case;
        end if;
end process;
process (clk, async_reset)                  --process secuencial
    begin
        if (async_reset = '0') then
            est_presente  <= a;
        elsif (clk'event and clk = '1') then
            if (en = '0') then
                est_presente <= proximo_est;
            end if;
        end if;
end process;
end behav_tabla_est;
```