

Universidad de Murcia
Facultad de Informática

TÍTULO DE GRADO EN
INGENIERÍA INFORMÁTICA

Estructura y Tecnología de Computadores

Tema 2: Sistemas Digitales - Circuitos Secuenciales

Apuntes

CURSO 2010 / 11

VERSIÓN 2.0

Departamento de Ingeniería y Tecnología de Computadores

Área de Arquitectura y Tecnología de Computadores



Índice general

2.1. Introducción	1
2.2. Biestables (flip-flops)	2
2.2.1. Cerrojo tipo S-R	2
2.2.2. Cerrojo tipo D	3
2.2.3. Flip-flop tipo D	4
2.2.4. Flip-flop tipo S-R	6
2.2.5. Flip-flop tipo J-K	7
2.2.6. Flip-flop tipo T	7
2.3. Registros	8
2.4. Memorias SRAM	8
2.5. Memorias DRAM	12
2.6. Diseño/síntesis de un circuito secuencial	15
2.6.1. Estructura general de un circuito secuencial	15
2.6.2. Fases en el diseño/síntesis de un circuito secuencial	16
2.6.3. Ejemplo de diseño/síntesis de un circuito secuencial	17
A2. Apéndices	22
A2.1. Construcción física de una celda de memoria	22
B2. Boletines de prácticas	25
B2.1. Normas sobre la entrega de prácticas	25
B2.2. Implementación de un Archivo/Banco de Registros con TkGate	25
B2.2.1. Objetivos	25
B2.2.2. Prerequisitos	25
B2.2.3. Plan de trabajo	25
B2.2.4. Archivo/Banco de Registros	26
B2.2.5. Ejercicios	26
E2. Ejercicios	28
E2.1. Sistemas Digitales: Circuitos Secuenciales	28
E2.2. Solución a ejercicios seleccionados	36

2.1. Introducción

Todos los circuitos que vimos en el tema anterior se caracterizan porque la salida en cada instante depende única y exclusivamente de las entradas en ese mismo instante. En este sentido, se dice que los circuitos combinacionales *carecen de memoria*. Pero existe otro tipo de circuitos en los cuales nos interesa que, de alguna manera, el circuito posea la capacidad de recordar su historia anterior, es decir, la *secuencia* de operaciones a la que ha sido sometido¹. El ejemplo más típico es la propia memoria de un computador: puede que queramos almacenar un dato en una determinada posición para poder recuperarlo en otro instante posterior. A este otro tipo de circuitos, que poseen esta capacidad de *recordar*, se les denomina circuitos secuenciales. Estudiaremos los más importantes en este apartado, comenzando por los elementos de memoria más sencillos, capaces de almacenar un bit: los biestables, también conocidos como *flip-flops*.

¹En general, ninguno de los circuitos que veremos almacena literalmente la secuencia de operaciones a la que ha sido sometido. Sin embargo, la información que almacenan es siempre consecuencia de dichas operaciones.

2.2. Biestables (flip-flops)

Antes de estudiar los biestables, estudiaremos previamente unos circuitos más elementales, denominados cerrojos, que serán utilizados como bloques básicos en la construcción de aquellos. Comenzamos con los cerrojos de tipo S-R.

2.2.1. Cerrojo tipo S-R

El primer circuito secuencial que estudiaremos es capaz de almacenar un bit, de manera que pueda ser recordado con posterioridad, cuando dejen de estar activas las entradas que provocaron su escritura. La figura 1 muestra dicho circuito, al que se le llama cerrojo S-R (cerrojo *Set-Reset*), construido a partir de dos puertas NOR cuyas salidas son realimentadas a las entradas.

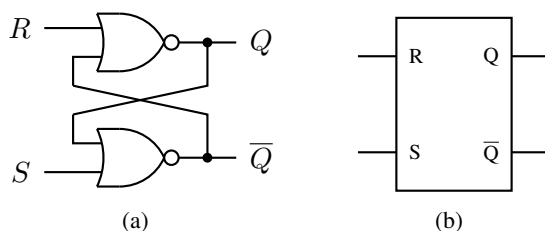


Figura 1: Implementación (a) y diagrama lógico (b) de un cerrojo S-R.

Las salidas Q y \bar{Q} representan el valor del estado almacenado y su complemento. Cuando S y R están a 0, las puertas NOR actúan como inversores y almacenan los valores anteriores de Q y \bar{Q} . Por ejemplo, si la salida Q es 1, entonces el inversor inferior produce una salida 0, que es \bar{Q} , y viceversa.

Veamos ahora cómo escribir en el cerrojo: si S vale 1 la salida \bar{Q} es 0 y Q es 1 (*Set*). Si R es 1 la salida Q es 0 y \bar{Q} es 1 (*Reset*). Cuando S y R toman el valor 1 simultáneamente, la situación del circuito es inestable y las salidas oscilan. Esta última situación, pues, debe evitarse puesto que no está controlada.

La *tabla de excitación 1* resume el funcionamiento lógico del cerrojo S-R (Q^* indica el valor del estado siguiente, frente al estado actual Q).

Entradas de excitación		Estado actual	Estado siguiente	
S	R	Q	Q^*	
0	0	0	0	No cambia
0	0	1	1	
0	1	0	0	Puesta a cero
0	1	1	0	
1	0	0	1	Puesta a uno
1	0	1	1	
1	1	0	X	No permitido
1	1	1	X	

Tabla 1: Tabla de excitación del cerrojo S-R.

La tabla de excitación es simplemente la tabla de estados del cerrojo y muestra las transiciones de estado para cada combinación de entradas de excitación. Las columnas S y R son las entradas aplicadas al cerrojo. La columna rotulada Q es el estado del cerrojo S-R antes de aplicar una combinación de entradas a S y R. La columna rotulada como Q^* es el estado del cerrojo después de aplicar las entradas al cerrojo. Por tanto, la columna Q es el estado actual y la columna Q^* es el estado siguiente. Esta tabla puede expresarse también

mediante una ecuación lógica, denominada *ecuación característica* del cerrojo S-R:

$$Q^* = S + \bar{R} \cdot Q$$

que se obtiene simplificando la salida Q^* en función de las entradas S, R y Q.

Resulta interesante ver cómo es posible incluir una señal de control C , que determinará en qué momentos el cerrojo hará caso de sus entradas. Para ello basta emplear dos puertas AND, como se muestra en la figura 2. Sólo cuando la señal de control C esté a 1, será posible cambiar el estado del cerrojo mediante S y R.

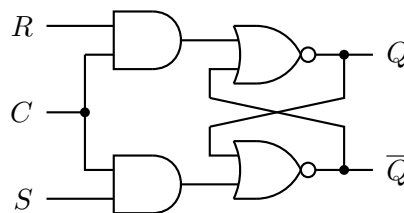


Figura 2: Cerrojo S-R con señal de control.

2.2.2. Cerrojo tipo D

Pasamos ahora a modificar el cerrojo S-R para hacer más sencillo el almacenamiento de los datos. Necesitamos un dispositivo que transfiera un valor lógico de su entrada de excitación, que denominaremos D, a la celda de almacenamiento con acoplamiento cruzado de un cerrojo. La figura 3 muestra el diagrama de un cerrojo D. Un cerrojo D almacena el valor de la señal de la entrada de datos en la memoria interna y posee dos entradas y dos salidas. Las entradas son: el valor de datos que se va a almacenar, denominado D, y una señal de control, llamada C, que determina en qué momento el cerrojo debe almacenar el valor de la entrada D en la estructura acoplada. Cuando C vale 1 el cerrojo está abierto y el valor de salida Q se convierte en el valor de entrada D. Cuando C vale 0, el cerrojo está cerrado y el valor de salida Q es el valor que fue almacenado la última vez que se abrió el cerrojo.

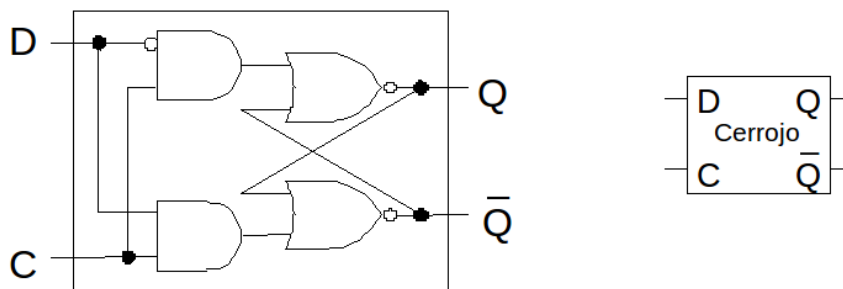


Figura 3: Cerrojo D, controlado por la señal C.

Cuando tratamos con circuitos secuenciales, puesto que su comportamiento depende de su evolución en el tiempo, es útil representar de algún modo esta evolución temporal. Para ello se emplean los cronogramas, que no son más que diagramas donde, en el eje horizontal se representa el tiempo, mientras que en el eje vertical se muestra el valor lógico de determinadas señales (y que, por tanto, pueden tomar los valores cero o uno). En el siguiente cronograma, por ejemplo, se muestra el funcionamiento de un cerrojo D si suponemos que la salida inicialmente tiene el valor cero. Obsérvese cómo el estado almacenado, Q, toma el valor de la entrada D (justo tras un pequeño retardo) en el momento en que la señal C vale 1. Sólo si C vale 1 el cerrojo es sensible a dicha entrada D. A menudo, no obstante, en la representación temporal de las señales se desprecia el valor de dichos retardos, y se muestran los cambios de modo instantáneo en el valor de las mismas.

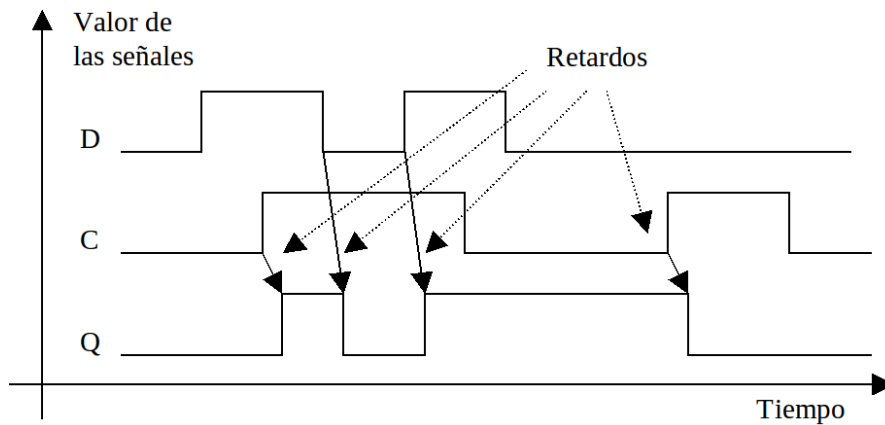


Figura 4: Operación de un cerrojo D suponiendo que la salida está inicialmente a 0.

Entrada de control	Entrada de excitación	Estado siguiente	
C	D	Q^*	
0	X	Q	Retención
1	0	0	Almacenar 0
1	1	1	Almacenar 1

Tabla 2: Tabla de excitación del cerrojo D.

En la tabla 2 se observa la tabla de excitación del cerrojo D en forma resumida que da lugar a la ecuación característica:

$$Q^* = D \cdot C + \bar{C} \cdot Q$$

2.2.3. Flip-flop tipo D

El cerrojo de la figura 3 es la base para otro elemento de memoria ligeramente más complejo que también nos permite almacenar datos y que llamaremos *flip-flop tipo D*.

A diferencia del caso de los cerrojos, las señales de entrada que recibe un flip-flop sólo tienen efecto durante un instante de tiempo, en lugar de a lo largo de un intervalo largo de tiempo como ocurría con la señal C de los cerrojos.

Una señal periódica, denominada *reloj*, será la encargada de determinar en qué momento concreto el circuito será sensible a su entrada y podrá escribirse en él. La señal de reloj oscilará tomando periódicamente valores alternando entre cero y uno. A cada cambio de esta señal se le denomina flanco. Un flanco se denomina ascendente, positivo o de subida si el cambio es de 0 a 1; y descendente, negativo o de bajada en caso contrario.

Aunque, por simplicidad en los diagramas, suponemos que las transiciones de las señales de 0 a 1 o de 1 a 0 ocurren de forma instantánea y, por tanto, dibujamos líneas verticales para representarlas, esto no es así en la realidad. En unos cronogramas más precisos, las transiciones serían casi verticales (pero no del todo) y transcurriría un corto periodo de tiempo desde que la señal comienza a cambiar hasta que se estabiliza en su nuevo valor. Durante este corto tiempo, y sólo entonces, es necesario que la señal cuyo valor se quiera almacenar en el flip-flop se mantenga estable.

El estado almacenado en el flip-flop podrá cambiarse sólo durante uno de los dos flancos y se dirá que el flip-flop es activo en flanco descendente o en flanco ascendente, dependiendo de en qué momento el circuito es sensible a la entrada. El cronograma de la figura 5 ejemplifica el comportamiento deseado para un flip-flop de tipo D activo en el flanco ascendente. En este cronograma se supone que los retardos son despreciables.

Un modo sencillo de implementar la funcionalidad deseada es mediante la conexión en serie de dos cerro-

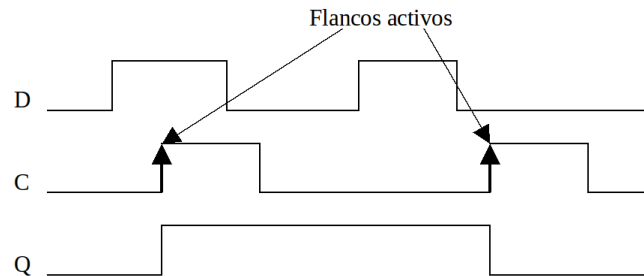


Figura 5: Cronograma de operación de un flip-flop D disparado por flanco de subida.

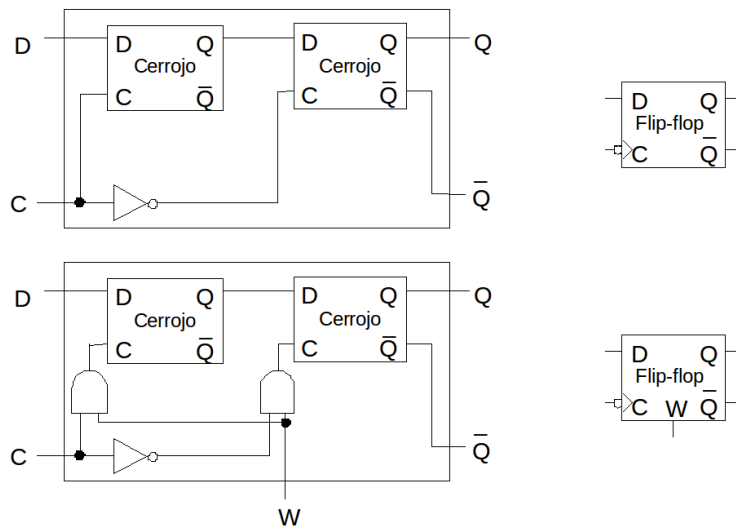


Figura 6: Flip-flop D maestro-esclavo, y símbolo lógico correspondiente: (Superior) Sin señal adicional de escritura. (Inferior) Después de añadirle dicha señal.

jos. En la figura 6 (superior) podemos ver el esquema de un flip-flop D maestro-esclavo disparado por flanco de bajada (que la activación se produce por flanco se expresa colocando un triángulo en la entrada C del bloque lógico, y si además incluimos un pequeño círculo estamos indicando que el flanco activo es el descendente). El primer cerrojo, llamado *maestro*, se abre y sigue a la entrada D cuando la entrada de reloj C vale 1. Cuando la entrada de reloj C cae, el primer cerrojo se cierra, pero el segundo cerrojo, llamado *esclavo*, se abre y obtiene su entrada de la salida del primer cerrojo maestro. Hacer el flip-flop sensible al flanco de subida, en lugar de al de bajada sería trivial, cambiando el inversor para que afectase al cerrojo maestro, en lugar de al esclavo.

En la figura inferior se ha añadido, además, una señal adicional de permiso de escritura (W), para poder controlar que no se realice la escritura en todos los flancos activos de la señal de reloj, sino sólo en aquellos en los que nos interese (cuando la señal W valga 1). Esta señal, como veremos, será de gran importancia cuando construyamos el camino de dato de la CPU multiciclo en el tema 4.

La tabla de excitación de un flip-flop D (sin permiso de escritura) es similar a la de un cerrojo D, la diferencia radica en la temporización. Mientras que en un cerrojo el estado cambia siempre que las entradas adecuadas cambien y la señal de control tenga valor 1, en un flip-flop el estado cambia únicamente en un flanco de reloj (ascendente o descendente), o sea, un flip-flop está gobernado por una señal de reloj (ver tabla 3 para el caso de un flip-flop D disparado por flanco descendente). La ecuación característica queda muy sencilla, puesto que el estado simplemente se actualiza con el valor de la entrada D en el flanco correspondiente: $Q^* = D$.

Entrada de reloj	Entrada de excitación	Estado siguiente	
C	D	Q^*	
1 → 0	0	0	Almacenar 0
1 → 0	1	1	Almacenar 1

Tabla 3: Tabla de excitación del flip-flop D disparado por flanco descendente.

De modo análogo se puede razonar para el caso de tener la entrada de activación W. En ese caso, la tabla quedaría como se muestra en 4 y la ecuación característica sería: $Q^* = D \cdot W + Q \cdot \overline{W}$.

Entrada de reloj	Permiso de escritura	Entrada de excitación	Estado siguiente	
C	W	D	Q^*	
1 → 0	0	0	Q	Retener estado
1 → 0	1	0	0	Almacenar 0
1 → 0	1	1	1	Almacenar 1

Tabla 4: Tabla de excitación del flip-flop D disparado por flanco descendente con señal de permiso de escritura.

2.2.4. Flip-flop tipo S-R

Los flip-flop tipo D son los más importantes, puesto que constituyen el método más cómodo para el almacenamiento de datos en los computadores. En los apartados que siguen veremos cómo utilizarlos para construir registros, cierto tipo de memorias, etc. Sin embargo, no son la única alternativa. Pueden construirse también otro tipo de flip-flops, con comportamientos ligeramente distintos, que pueden resultar útiles en algunos casos. El flip-flop S-R, por ejemplo, puede construirse a partir de dos cerrojos S-R con una señal de control. Las señales de activación de los dos cerrojos son controladas por versiones complementarias de una señal de reloj. Cuando la señal de reloj C es cero, el cerrojo maestro está abierto y el esclavo cerrado. El cerrojo maestro procesa sus entradas S y R. Cuando la señal de reloj es uno ambos flip-flops intercambian sus papeles de

forma que el maestro está cerrado y el esclavo abierto. El esclavo está abierto, enviando la salida del cerrojo maestro a la salida Q del flip-flop, mientras que el cerrojo maestro permanece cerrado ignorando cualquier cambio posterior en sus entradas. El flip-flop S-R de la figura 7 se activa por flanco ascendente y tiene como ecuación característica: $Q^* = S + \bar{R} \cdot Q$, obtenida a partir de la tabla de excitación 5

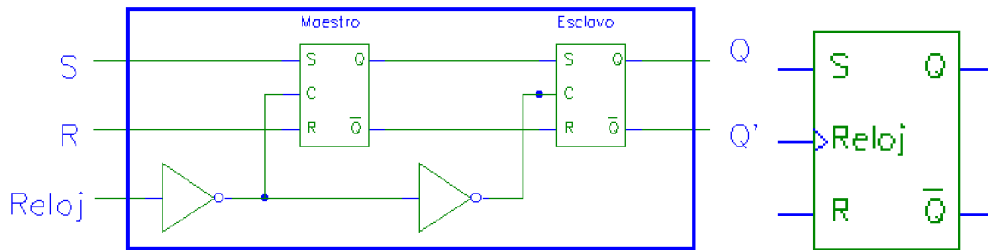


Figura 7: (Izquierda) Flip-flop S-R. (Derecha) Símbolo lógico del flip-flop S-R.

Entrada de reloj	Entradas de excitación		Estado siguiente	
	S	R	Q^*	
$0 \rightarrow 1$	0	0	Q	Retener estado
$0 \rightarrow 1$	0	1	0	Puesta a 0
$0 \rightarrow 1$	1	0	1	Puesta a 1
$0 \rightarrow 1$	1	1	X	No permitido

Tabla 5: Tabla de excitación del flip-flop S-R disparado por flanco ascendente.

2.2.5. Flip-flop tipo J-K

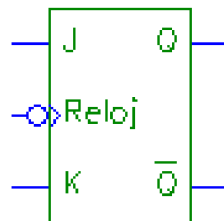


Figura 8: Símbolo lógico del flip-flop J-K.

Otro tipo de flip-flop es el flip-flop J-K. Puede considerarse como una extensión de un flip-flop S-R, a cuyas entradas se asigna $J = S$ y $K = R$. Sin embargo, mientras que en aquel la combinación $S = R = 1$ no está permitida, el flip-flop J-K utiliza este caso particular para agregar un modo de operación muy útil. La característica adicional del dispositivo J-K es que su estado se alterna; es decir, cambia del 0 al 1 ó 1 al 0 cuando $J = K = 1$. La ecuación característica de este flip-flop es $Q^* = \bar{K} \cdot Q + J \cdot \bar{Q}$, obtenida a partir de la tabla de excitación 6.

2.2.6. Flip-flop tipo T

Por último, el flip-flop T se utiliza con frecuencia para la construcción de módulos contadores. Tiene una única entrada de excitación llamada T. La función de este dispositivo consiste en cambiar su estado cada vez

Entrada de reloj	Entradas de excitación		Estado siguiente	
C	J	K	Q^*	
$1 \rightarrow 0$	0	0	Q	Retener estado
$1 \rightarrow 0$	0	1	0	Puesta a 0
$1 \rightarrow 0$	1	0	1	Puesta a 1
$1 \rightarrow 0$	1	1	\overline{Q}	Alternancia

Tabla 6: Tabla de excitación del flip-flop J-K disparado por flanco descendente.

que así lo indique la señal de reloj. La tabla 7 resume el comportamiento de un flip-flop T activado por flanco descendente cuya ecuación característica es: $Q^* = \overline{Q} \cdot T + Q \cdot \overline{T}$

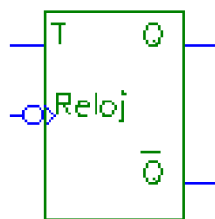


Figura 9: Símbolo lógico del flip-flop T.

Entrada de reloj	Entrada de excitación	Estado siguiente	
C	T	Q^*	
$1 \rightarrow 0$	0	Q	Retener estado
$1 \rightarrow 0$	1	\overline{Q}	Alternancia

Tabla 7: Tabla de excitación del flip-flop T disparado por flanco descendente.

2.3. Registros

Puesto que los flip-flops de tipo D sirven para almacenar un bit, un modo sencillo de implementar un registro que contenga un dato de varios bits, como un byte o una palabra, consiste simplemente en concatenar varios flip-flops que compartan la misma señal de reloj y de permiso de escritura. La figura 10 muestra el esquema y el diagrama lógico de un registro de n bits. Obsérvese que es completamente análogo a un flip-flop sencillo (con una sola entrada de reloj y una sola entrada de permiso de escritura), sólo que en este caso tanto el dato leído (D_{out}) como el dato a escribir (D_{in}) son señales de n bits de ancho:

2.4. Memorias SRAM

Los registros y archivos de registros (que se implementará como práctica de este tema), se utilizan como bloques constructivos para memorias de pequeño tamaño, donde la prioridad es obtener un tiempo de acceso muy pequeño. Las memorias mayores, sin embargo, se construyen utilizando las llamadas SRAM (*Static Random Access Memory*, memoria estática de acceso aleatorio) y las DRAM (*Dynamic Random Access Memory*, memoria dinámica de acceso aleatorio). En este apartado estudiaremos las particularidades de las primeras, dejando las DRAM para el siguiente.

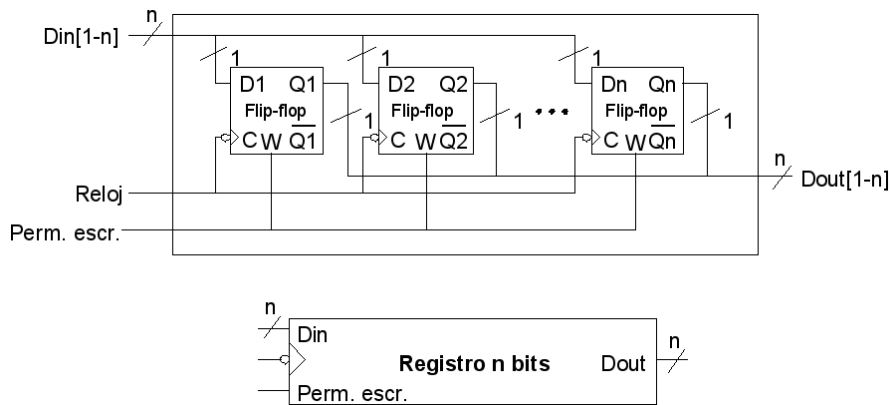


Figura 10: Implementación y diagrama lógico de un registro de n bits usando flip-flops tipo D.

Estructura de una memoria SRAM

Las memorias SRAM se presentan como circuitos integrados que son arrays de memoria (divididos en posiciones, a cada una de las cuales le corresponde una dirección), su principal característica principal es que el tiempo de acceso es muy corto, por ello se emplean principalmente en el diseño de memorias caché.

Las SRAM tienen un tiempo de acceso fijo para cualquier dato, aunque las características de acceso de lectura y escritura pueden a veces variar. Un chip de SRAM tiene una configuración específica en función del número de posiciones direccionables, así como de la anchura del dato almacenable en cada posición. La figura 11 muestra el diagrama lógico de una SRAM de 32K posiciones de 8 bits cada una ($32K \times 8 = 32KB$), y por tanto necesita 15 líneas para ser direccionada ($2^{15} = 32768$). Cada elemento tendrá 8 bits, por lo tanto, tendremos 8 líneas de entrada de datos y 8 líneas de salida de datos. Otras alternativas serían una SRAM de 256K x 1, con la misma capacidad, pero con diferente número de elementos (18 líneas de dirección) de 1 bit (una línea de entrada de datos y otra de salida); o bien, una SRAM de 16K x 16 (14 líneas de dirección, 16 de entrada y 16 de salida). Dejaremos la explicación de las líneas de selección de chip (*chip select*), habilitación de salida (*output enable*) y habilitación de escritura (*write enable*) para más adelante.

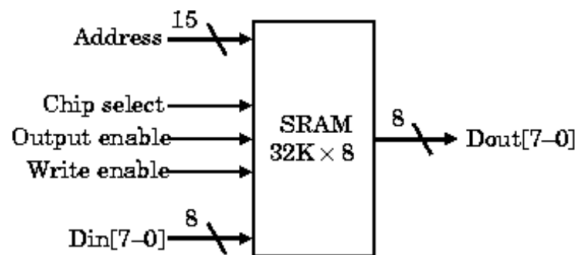


Figura 11: SRAM de 32K x 8 : 15 líneas de dirección, 8 líneas de entrada de datos, tres líneas de control y 8 líneas de salida de datos.

Al igual que en una memoria ROM, el número de posiciones direccionables se denomina *altura* y el número de bits por unidad de memoria se llama *anchura*. Por diversas razones técnicas, las SRAM más modernas y rápidas suelen estar disponibles en configuraciones “delgadas”, habitualmente $\times 1$ y $\times 4$ (1 y 4 bits de anchura, respectivamente).

Podríamos pensar en construir una memoria SRAM de forma similar a un banco de registros, pero este planteamiento presenta algunos inconvenientes. En un archivo de 32 registros, como el anteriormente visto, necesitábamos dos multiplexores de 32 a 1, mientras que en una SRAM 32K x 8, usar un multiplexor de 32K a 1 resulta totalmente impracticable. Para solucionar este primer inconveniente, las SRAM utilizan líneas de salida compartidas, llamadas líneas de bits, que permiten que múltiples fuentes compartan una sola línea de

datos. Para consentir que varias fuentes controlen una misma línea se utiliza un buffer de tres estados (*buffer tri-estado*).

Un buffer de tres estados es como una puerta lógica un tanto especial, con dos entradas (*data* y *enable*) y una salida (*out*). Las entradas son la señal de datos y habilitación de salida, respectivamente, y la salida es igual a la señal de entrada si la línea *habilitación de salida* está activa; en otro caso, permanece en un estado de alta impedancia, con lo que los otros buffers cuya señal de habilitación de salida esté activa, podrá usar la línea compartida. La figura 12 muestra cuatro buffers de tres estados que comparten una única línea de salida. Como vemos, esta estructura puede sustituir a un multiplexor de 4 a 1 (para ello, sólo una de las cuatro entradas de selección puede estar activa en un momento determinado, lo cual se consigue con un decodificador). Esta implementación es más eficiente (es mucho menos costosa en lo que se refiere al número de transistores necesarios) y, por tanto, más escalable (es decir, aumentable a mayor número de entradas).

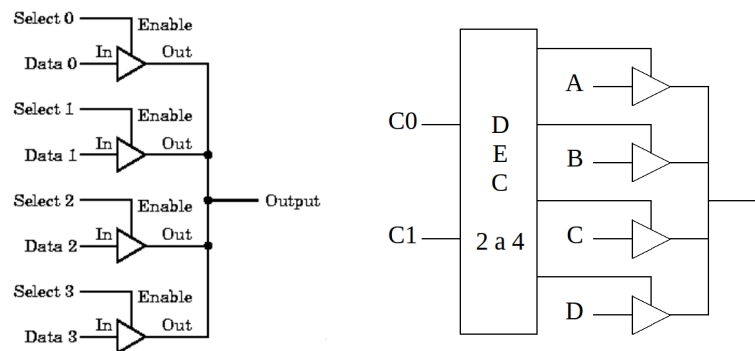


Figura 12: Cuatro buffers triestado para formar un multiplexor 4 a 1.

Estos buffers de tres estados pueden incorporarse a los flip-flops que forman las celdas básicas de la SRAM permitiendo así que aquellas celdas que corresponden a una misma salida compartan una sola línea. La figura 13 muestra una pequeña SRAM de 4x2, donde cada cerrojo (obsérvese que en este caso se usan cerrojos en lugar de flip-flops) tiene una línea de habilitación (*Enable*) que controla el buffer de tres estados y, por tanto, el acceso a las líneas de salida. La señal de habilitación de escritura, debidamente combinada con las salidas del decodificador, decide si la palabra seleccionada es escrita con el dato colocado en las líneas de entrada, D_{in} .

Con este diseño hemos eliminado la necesidad de usar multiplexores; sin embargo, todavía se requiere la utilización de un gran decodificador a la entrada, para memorias con muchas posiciones direccionables. Para eliminar este inconveniente, las grandes memorias se organizan como arrays rectangulares (bidimensionales) y utilizan un proceso de decodificación de dos pasos. La figura 14 muestra como podría conseguirse tal disposición construyendo una memoria SRAM 8x8 en base a 8 bloques 4x2 (descrito en la figura 13). El primer decodificador genera la dirección para los 8 bloques de 4x2 a partir de los 2 bits más significativos de la dirección. Después un conjunto de multiplexores se utiliza para seleccionar un bit de cada conjunto de 2 bits a partir del bit menos significativo de la dirección. Este diseño es más eficiente que una decodificación de un solo paso que necesitaría un decodificador de 3 a 8 o un multiplexor de 8 a 1.

Veamos ahora cómo se realizan las lecturas y escrituras en las SRAM, para lo que tendremos que explicar el uso de las dos líneas de control que aún no hemos utilizado. La señal de selección de chip (*Chip Select*) sirve para seleccionar si el chip se conecta o no a los buses de entrada (D_{in}) y de salida (D_{out}). En caso de que valga 0, el chip queda completamente aislado de dichos buses y significa que no va a utilizarse en ese momento, ni para leer ni para escribir. Este aislamiento se consigue también con buffers triestado controlados por dicha señal. Esta línea de selección tiene utilidad a la hora de construir memorias de mayor capacidad utilizando varios bloques de menor tamaño.

En el caso de las lecturas, debe estar a 1, además de la línea mencionada, la línea de habilitación de salida (*Output Enable*). Esto resulta útil cuando se conectan múltiples memorias a un solo bus de salida,

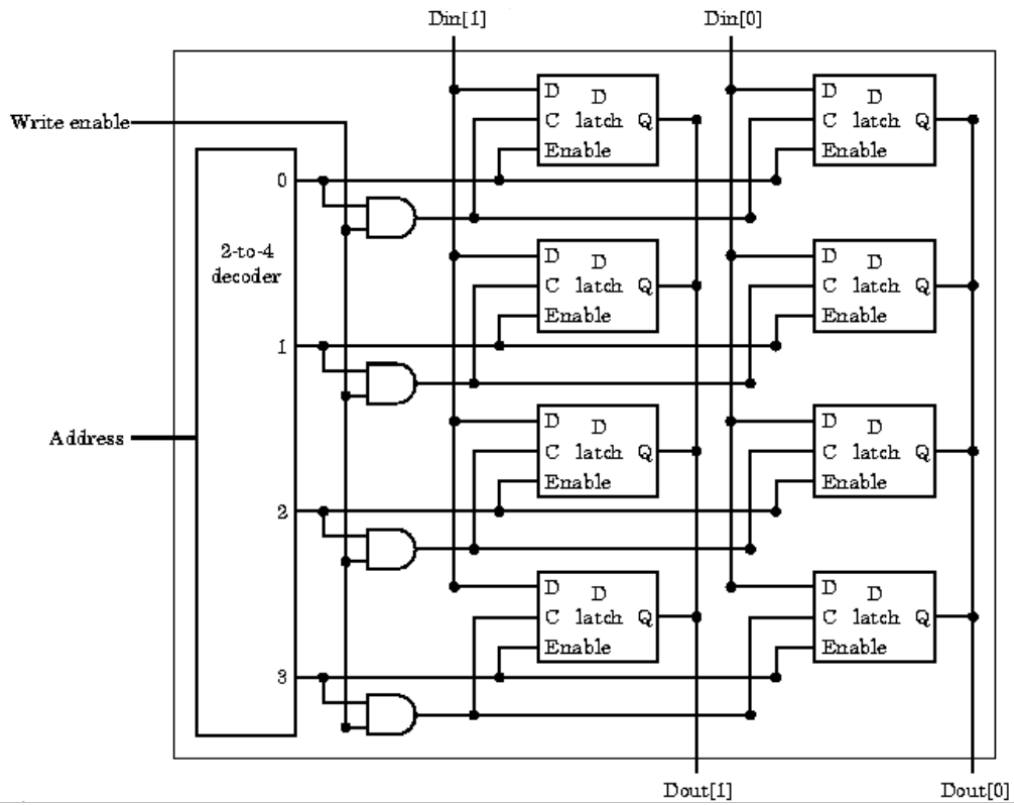


Figura 13: SRAM de 4 x 2. Por simplicidad se omiten las entradas de habilitación de salida y selección de chip.

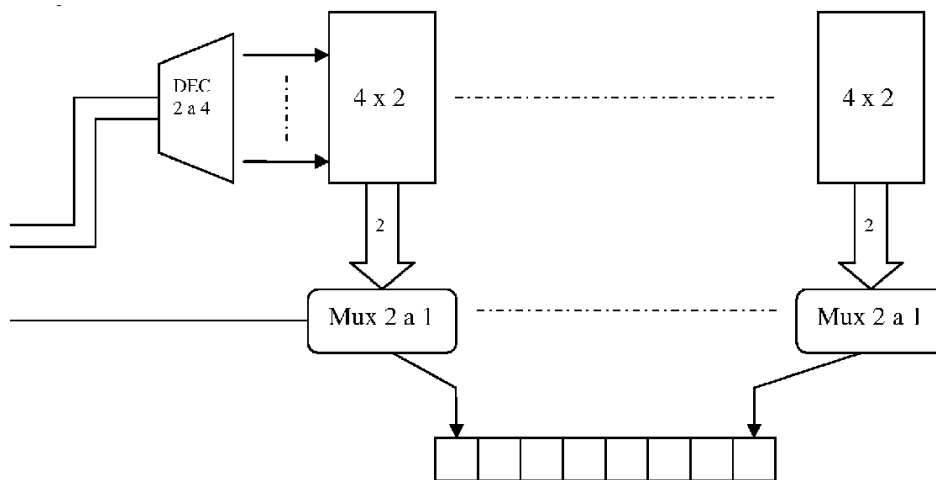


Figura 14: SRAM 8 x 8 como un array de 8 elementos de 4 x 2.

ya que permite determinar la memoria que controla el bus en cada momento. Para cada escritura, debemos suministrar a la memoria el dato a escribir y la dirección, así como las señales que hacen que se efectúe la escritura. Cuando la señal de habilitación de escritura (*Write Enable*) y la señal de selección de chip son verdaderas, el dato en las líneas de entrada de datos se escribe en la celda determinada por la dirección.

2.5. Memorias DRAM

En una memoria SRAM, el valor que se almacena en cada celda se mantiene en el cerrojo correspondiente de forma indefinida mientras que haya alimentación eléctrica. El problema es que una celda para un solo bit necesita bastantes transistores (unos seis, habitualmente). En los casos en los que se desea un alto nivel de integración y bajo consumo, aunque se penalice con velocidades de trabajo menores, se recurre a las celdas dinámicas. Básicamente, el diseño consiste en reducir el número de transistores que componen la celda, consiguiendo las características citadas, obteniendo diferentes configuraciones a cada cual más sencilla, hasta llegar al máximo de simplicidad consistente en construir una celda con un único transistor. Dado que una DRAM usa un único transistor por cada bit almacenado, son mucho más baratas y densas por bit que las SRAM. Sin embargo, este tipo de memorias presenta el inconveniente de que al estar la información almacenada en un condensador, éste se va descargando internamente con el tiempo, de modo que debe ser refrescada periódicamente (por este motivo se llama almacenamiento dinámico). Para refrescar la celda, se lee su contenido y se vuelve a escribir. Este proceso de refresco se lleva a cabo normalmente a través de la propia circuitería del chip DRAM.

Estructura de una memoria DRAM

Las celdas de memoria se organizan en una matriz bidimensional con objeto de multiplexar las líneas de selección de cada una de las dimensiones y ahorrar costes y espacio en la fabricación de los diferentes chips que componen un módulo de memoria. Esto obliga al controlador de memoria a descomponer cada dirección de acceso a una palabra de memoria en dos coordenadas, donde primero se selecciona la fila de una celda en la matriz y luego su columna. Dicho controlador de memoria proporciona las señales de control necesarias, RAS (*Row Address Strobe*) y CAS (*Column Address Strobe*), que gobiernan la temporización. Por lo tanto, las DRAM utilizan la misma idea de la decodificación en dos niveles introducida en el apartado anterior.

En la figura 15 se muestra un chip de DRAM de 16 megabits, configurada como 2M x 8. Las celdas están organizadas en forma de matriz de 4K x 4K. Las 4096 celdas de cada fila se dividen en 512 grupos de 8 celdas, de manera que una fila puede memorizar 512 bytes de datos. Se requieren pues 12 bits de direcciones para seleccionar una fila. Otros 9 bits son necesarios para especificar el grupo de 8 bits dentro de la fila seleccionada. Por tanto, para acceder a un byte de esta memoria se necesitan 21 bits de direcciones. Los 12 bits más significativos, y los 9 bits de orden inferior, constituyen respectivamente las direcciones de fila y de columna de un byte. Durante una operación de lectura o de escritura, se aplica en primer lugar la dirección de la fila. Ésta se carga en el cerrojo (*latch*) de direcciones de filas como consecuencia de un pulso de señal en la entrada de validación de dirección de fila RAS. Se inicia entonces una operación de lectura en la que se leen y refrescan todas las celdas de la fila seleccionada. El contenido de las columnas de la fila seleccionada pasa a almacenarse en el conjunto de circuitos de detección/escritura. Inmediatamente después de haber cargado la dirección de fila, se carga la dirección de columna en el cerrojo de direcciones de columnas bajo el control de la señal de validación de dirección de columnas CAS. La información de este cerrojo es decodificada para seleccionar el grupo apropiado de 8 circuitos de detección/escritura. Si la señal de control R/W indica una operación de lectura, los valores de los circuitos seleccionados son transferidos a las líneas de datos, D_{7-0} . Para una operación de escritura, la información de las líneas D_{7-0} se transfiere a los circuitos seleccionados. Esta información se utiliza para sobrescribir el contenido de las celdas seleccionadas. El esquema de direccionamiento de dos niveles, junto con la circuitería interna necesaria, hace que los tiempos de acceso a las DRAM sean mucho mayores que los tiempos de acceso de las SRAM.

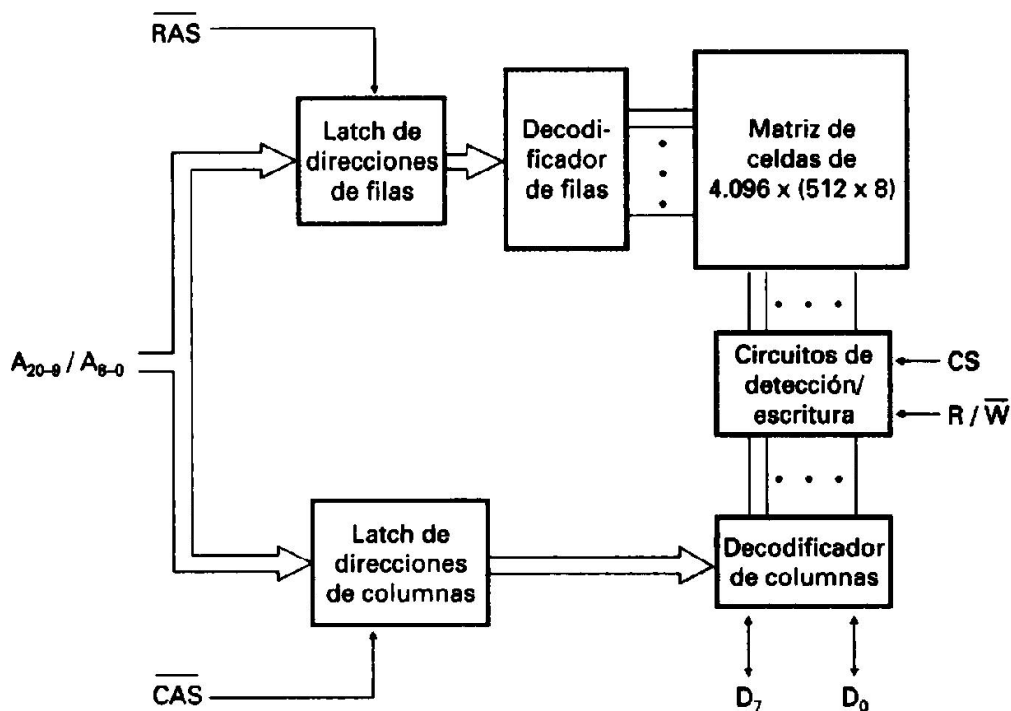


Figura 15: DRAM 2M x 8.

Debido a su alta densidad y coste reducido, las DRAM se usan ampliamente en las memorias de los computadores. Para disponer de la suficiente flexibilidad al diseñar sistemas de memoria, se fabrican chips con organizaciones diversas. Por ejemplo, un chip de 64 Mbits puede estar organizado como 16M x 4, 8M x 8 o 4M x 16.

A menudo es interesante que, para minimizar el número de patillas de un chip de memoria RAM, las líneas de entrada y salida de datos estén compartidas. En ese caso, si el tamaño de palabra utilizado en las memorias es de n bits, el chip tendrá n líneas de entrada/salida, $Din/Dout_{1..n}$, en lugar de las n de entrada $Din_{1..n}$, y n de salida $Dout_{1..n}$. Para poder hacer esto, es necesario que separemos las funciones de entrada y salida de dichas líneas, para evitar su uso simultáneo. Esto se consigue de nuevo a través de buffers triestado, tal y como se indica en la figura 16, controlados por una señal L/\bar{E} , que determinará si se está usando la memoria para leer o para escribir en ella. Si $L/\bar{E}=1$, estaremos leyendo de la memoria, y en ese caso los buffers triestado de la entrada dejarán aisladas de la salida a las señales $Din_{1..n}$. Si $L/\bar{E}=0$, entonces estaremos escribiendo, y serán los buffers de la salida los que aislen a las señales $Dout_{1..n}$.

Como hemos visto, un acceso a memoria supone un acceso de fila seguido por un acceso de columna. Primero se envía al chip la dirección de fila y, a continuación, la dirección de columna. Por último, la memoria pone en el bus de datos el elemento deseado. Por tanto, en un acceso a memoria se dedica más tiempo a localizar el dato que a transmitirlo.

Si leemos varios datos que están consecutivos en memoria, sólo es necesario especificar la dirección para el primero de ellos, los demás nos vendrán dados por añadidura siempre que se encuentren en la misma fila. Supongamos que el ancho de nuestra memoria es de 64 bits, que cada fila consta de 256 bits y que deseamos leer 4 bloques de 64 bits cada uno (que supondremos que están en la misma fila). Basta con enviar a la memoria la dirección del primer bloque para que se seleccione la fila que lo contiene que, además, incluye los otros tres bloques que buscamos. De esta forma el acceso a los otros tres bloques será más rápido porque no necesitamos proporcionar su dirección, y porque tampoco hace falta seleccionar ninguna fila nueva. Esta forma de acceder a la memoria se conoce como *acceso en modo ráfaga*. Para especificar el número de ciclos empleados en el acceso a memoria en un chip en modo ráfaga, se utiliza la siguiente nomenclatura: “x-y-y-y”. El primer

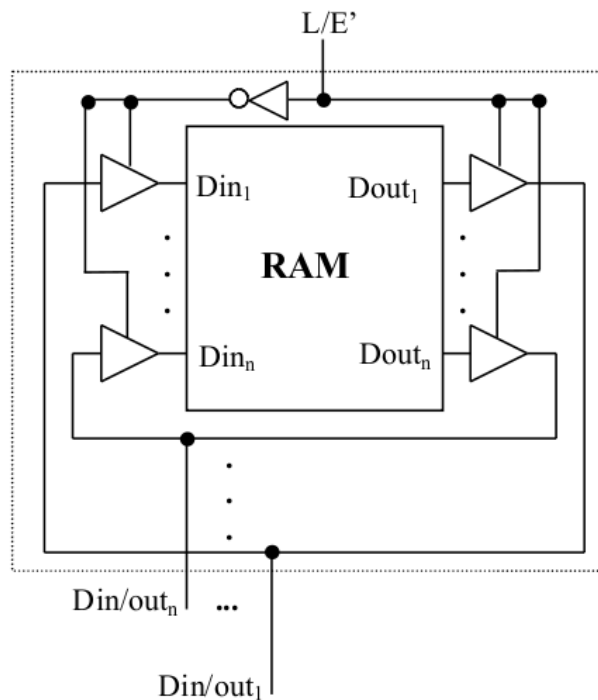


Figura 16: RAM con puerto compartido de lectura/escritura (bidireccional). Por simplicidad, se omiten las señales de dirección, selección de chip, etc.

número, 'x', representa el número de ciclos de reloj necesarios para efectuar la primera lectura de un bloque de 64 bits, mientras que el segundo número representa el número de ciclos necesarios para los tres accesos subsiguientes. Un ejemplo sería "5-2-2-2", que significa 11 ciclos para leer una fila completa (4 bloques de 64 bits), de los cuales 5 se emplean en el primer acceso y 2 en cada uno de los siguientes.

Cuando se accede a la DRAM de 2M x 8 de la figura 15, se detecta el contenido de las 4096 celdas de la fila seleccionada, pero sólo se colocan 8 bits en las líneas de datos D7-0. Este byte es seleccionado por los bits de la dirección de columna. Si añadimos un cerrojo a la salida del circuito detector de cada columna, de manera que la aplicación de una dirección de fila haga que se carguen los cerrojos correspondientes a todos los bits de la fila seleccionada, entonces sólo se requiere aplicar distintas direcciones de columna para transferir los diferentes bytes a las líneas de datos.

Por último, la memoria, como todos los circuitos vistos hasta ahora, es un dispositivo de almacenamiento electrónico, y como tal es susceptible de cometer errores, devolviendo información diferente a la que fue almacenada originalmente. La memoria DRAM, además, almacena los bits en pequeños condensadores que son refrescados continuamente para asegurar que la información no se pierde, y ello la hace más propensa a la posible generación de errores eventuales.

Tradicionalmente, los módulos de memoria han estado disponibles con y sin paridad. Los módulos de memoria sin paridad contienen exactamente un bit por cada bit de datos almacenado. Los módulos de memoria con paridad añaden un bit extra por cada 8 bits de datos almacenados que se utiliza para la detección y corrección de errores. Este bit adicional puede usarse como bit de paridad, o bien, en un esquema de detección y corrección de errores denominado *Error Correction Code* o ECC. La memoria ECC utiliza el algoritmo de Hamming, que permite no sólo detectar, sino también corregir errores en un bit dentro de bloques de 64 bits. La memoria con paridad o ECC es más cara y difícil de encontrar que la convencional, y además ralentiza ligeramente los accesos a memoria. Sin embargo, la utilización de algún tipo de mecanismo de detección de errores aumenta la fiabilidad del sistema y facilita la localización de errores que con frecuencia son atribuidos a la memoria. Por ello, es habitual su uso en ordenadores servidores cuya fiabilidad es crítica dentro de alguna

institución (tales como bancos, grandes empresas, etc). Para saber si un módulo de memoria dispone o no de paridad, basta contar el número de chips que el módulo posee. Si el número no es potencia de dos, entonces se trata de un módulo con paridad.

2.6. Diseño/síntesis de un circuito secuencial

En este apartado estudiaremos el diseño de un circuito secuencial genérico, es decir, aquel en el que la salida en un instante dado no depende sólo de las entradas en ese momento, sino también del estado en el que se encuentra el circuito. Éste, a su vez, depende de la historia de las entradas que hasta ese momento el sistema ha recibido. Puesto que, como vemos, es necesario que estos circuitos posean una capacidad de memorizar su estado, será necesario que en su construcción empleemos cerrojos o flip-flops. Estos serán los encargados de codificar el estado en el que se encuentra el circuito en cada momento.

2.6.1. Estructura general de un circuito secuencial

La figura 17 muestra la estructura general de un circuito secuencial. En él se observan sus componentes más importantes, que se detallan a continuación:

- **Entrada:** se trata de los n bits de entrada del circuito.
- **Salida:** son los m bits de salida producidos por el circuito.
- **Estado actual:** es la parte genuinamente secuencial del circuito, puesto que está encargada de codificar el estado en el que se encuentra éste. Está compuesta por k circuitos elementales de memoria, cerrojos o flip-flops, capaces de almacenar hasta 2^k estados diferentes (puesto que cada uno puede almacenar un 0 o un 1).
- **Función de transición:** se trata de una función combinacional (en realidad es una multifunción formada k funciones elementales, puesto que tiene k bits de salida) que determina, a partir del estado y la entrada actuales cuál es el estado siguiente al que debe pasar el circuito. Es, por tanto, la parte encargada de determinar la evolución en el estado del mismo.
- **Función de salida:** es otra función combinacional (otra multifunción con m bits de salida), que depende del estado del circuito (y quizá también de las entradas; en seguida volveremos sobre este punto), y que determina la salida del circuito en cada instante.

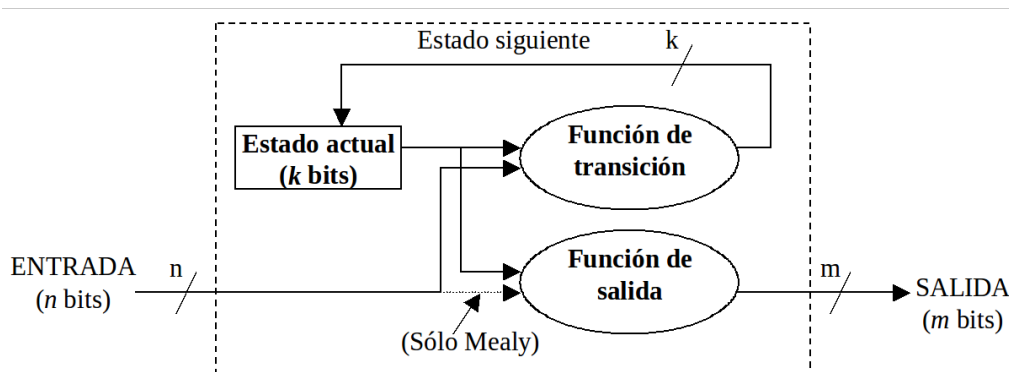


Figura 17: Esquema lógico general de un circuito secuencial. También se muestra la diferencia esencial entre los circuitos de Moore y los de Mealy.

A la hora de hablar de circuitos secuenciales, suele hacerse una clasificación en dos tipos principales, según el modo en que se calculan las salidas en función del estado y de las entradas. A estos dos tipos de circuitos se les denomina, respectivamente, de Moore y de Mealy. En los circuitos de Moore, la salida en cada instante puede computarse única y exclusivamente a partir del estado del circuito, sin necesidad de tener en cuenta las entradas. Así, las entradas provocan los cambios de estado, y sólo a través de este cambio influyen en las salidas. En los de Mealy, por el contrario, en cada instante la salida depende simultáneamente del estado del circuito y del valor de las entradas. En la figura 17 se resume la diferencia entre ambos tipos de circuitos: como puede observarse, sólo en los circuitos de tipo Mealy la función de salida también depende de las entradas.

Otra clasificación de los circuitos secuenciales dependerá del tipo de cerrojos o flip-flops con los que se implementen. Si se implementan con cerrojos (es decir, carecen de señal de reloj), entonces se denominarán asíncronos. Si, por el contrario, se implementan con flip-flops que comparten una misma señal de reloj, entonces se denominarán síncronos.

2.6.2. Fases en el diseño/síntesis de un circuito secuencial

Por cuestiones temporales nos centraremos en el diseño de circuitos utilizando únicamente autómatas de Moore que a la postre son iguales de potentes que los autómatas de Mealy, ya que para cualquier autómata de Mealy existe un autómata de Moore equivalente (y viceversa). En general, cuando abordemos el diseño de un sistema digital, lo haremos en una serie de pasos, que a continuación resumimos y que posteriormente aclararemos con un ejemplo:

1. **Especificación verbal:** se trata de resumir con palabras el funcionamiento deseado para el circuito. Es la especificación previa de los requisitos que queremos que cumpla.
2. **Especificación del autómata:** una vez especificado el comportamiento deseado verbalmente, tenemos que formalizarlo. Para ello se utiliza un diagrama de estados especial, denominado autómata (en concreto, el denominado autómata finito determinista o AFD). Este autómata, básicamente, sirve para representar gráficamente los posibles estados del sistema, la función de transición (es decir, los cambios de estado ante cada combinación de estado y entrada actuales) y la función de salida (es decir, el valor de la salida correspondiente a cada estado, en el caso de tratarse de un autómata de Moore).
3. **Minimización del autómata obtenido:** puede ocurrir que la especificación inicial del autómata sea correcta (es decir, lleve a cabo la tarea a realizar perfectamente), pero pueda conseguirse el mismo comportamiento con un autómata equivalente con menor número de estados. En ese caso, nos interesará hallar el AFD mínimo equivalente, con el fin de reducir la circuitería que posteriormente será necesaria. Dejaremos el estudio de las técnicas de minimización de autómatas para cursos posteriores, puesto que en este curso trabajaremos con autómatas lo suficientemente simples como para no tener que preocuparnos por su minimización.
4. **Codificación de estados:** una vez que conocemos el total de estados distintos en los que se podrá encontrar nuestro circuito, podemos decidir el número de flip-flops necesarios para codificarlos. En concreto, si el autómata tiene M estados, necesitaremos $n = \lceil \log_2 M \rceil$ flip-flops. En esta etapa a cada estado le asociaremos una combinación de n bits. Esta asignación de estados se puede hacer al azar o buscando que el número de puertas necesarias en el circuito resultante sea mínimo a través de un conjunto de reglas que no veremos por falta de tiempo.
5. **Minimización de la función de transición:** una vez que tenemos codificados los estados, minimizamos la función de transición (que ya es una función booleana combinacional normal y corriente). Con ello obtenemos la expresión con la que implementaremos dicha parte del circuito.

6. **Minimización de la función de salida:** seguidamente haremos lo propio con la función de salida, que en nuestro caso (al tratarse un autómata de Moore) depende única y exclusivamente de la codificación de cada estado. Con ello obtenemos otra expresión booleana mínima, que usaremos para implementar lo que nos queda del circuito.
7. **Implementación del circuito:** finalmente, sólo nos queda utilizar los n flip-flops (que estarán todos conectados a una misma señal de reloj) y las expresiones obtenidas en los pasos 5 y 6 para dibujar el circuito final, siguiendo la estructura de la figura 17.

2.6.3. Ejemplo de diseño/síntesis de un circuito secuencial

En este apartado diseñaremos un circuito de ejemplo siguiendo el procedimiento indicado en el apartado anterior. Elegiremos diseñar un circuito detector de una determinada secuencia de bits que le llegan por una línea serie, que podría utilizarse en una aplicación de comunicaciones (por ejemplo, para detectar un determinado carácter con un significado especial en la conexión).

Especificación verbal

“Diseñar un circuito secuencial con una única línea de entrada X y una única línea de salida S . El circuito debe ser capaz de detectar la aparición de tres unos consecutivos (es decir, durante tres ciclos de reloj) en su línea de entrada X . En ese caso deberá mostrar un uno en la salida S . En caso contrario deberá producir un cero. Si el circuito mostró un uno en la salida en el último paso, entonces debe volver a comenzar a contar unos en la entrada, y no volver a mostrar otro uno en la salida hasta que cuente otros tres unos en la entrada. Realizar dos diseños, uno utilizando biestables J-K y otro utilizando biestables D, en ambos casos disparados por flanco ascendente.”

Especificación del autómata

En este caso el autómata de Moore que recoge el funcionamiento deseado para el circuito es el mostrado en la figura 18. Obsérvese el empleo de círculos para cada estado Est_i , la especificación de la salida asociada a cada estado, por tratarse de un autómata de Moore (función de salida) y la especificación mediante flechas de las transiciones entre estados asociadas a cada entrada (función de transición). En la tabla E2.1 se recoge la misma información que en el diagrama pero mostrada en forma tabular.

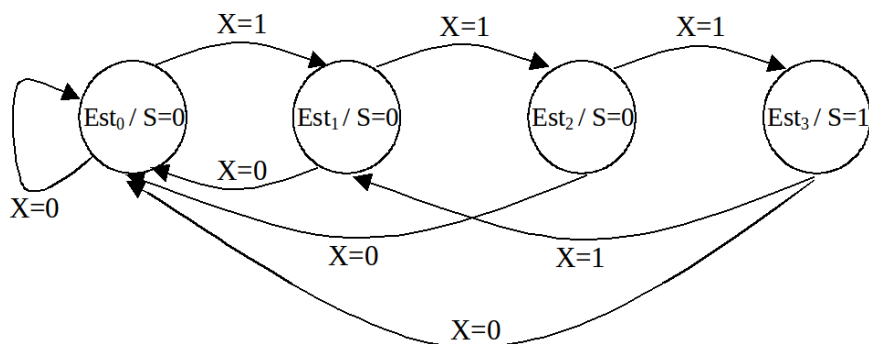


Figura 18: Autómata finito determinista del circuito detector de secuencias de tres unos.

Codificación de estados

Calculamos el número de flip-flops necesarios y asignamos una combinación de bits a cada estado. En este caso, al haber 4 estados, es suficiente con dos flip-flops cuyas salidas serán Q_0 y Q_1 . Utilizaremos la

Estado Actual	Estado Siguiente		Salida
	X=0	X=1	S
Est_0	Est_0	Est_1	0
Est_1	Est_0	Est_2	0
Est_2	Est_0	Est_3	0
Est_3	Est_0	Est_1	1

Tabla 8: Tabla de transición entre estados y función salida para el ejemplo explicado.

codificación en binario natural ($Est_0 = 00$, $Est_1 = 01$, $Est_2 = 10$, $Est_3 = 11$).

Minimización de la función de transición y de la función de salida

Una vez realizada la codificación de estados el siguiente paso es reescribir dicha tabla haciendo uso de la codificación elegida tal y como se muestra en la tabla E2.2. Obsérvese que diferenciamos el estado futuro (Q_i^*) del actual (Q_i) utilizando un asterisco.

Q_1Q_0	$Q_1^*Q_0^*$		Salida
	X=0	X=1	S
00	00	01	0
01	00	10	0
10	00	11	0
11	00	01	1

Tabla 9: Tabla de transición entre estados y función de salida con los estados ya codificados para el ejemplo explicado.

Una vez realizada dicha codificación debemos determinar qué valores deben de tener las entradas de los biestables para que dicha transición se realice en la salida. Para el caso de biestables tipo D el valor de la entrada se corresponde con el que deseamos en la salida, pero en el caso de biestables tipo J-K debemos de calcularlos en función del estado actual y del estado siguiente del biestable como se muestra en la tabla 10.

$Q \rightarrow Q^*$	J	K
0 \rightarrow 0	0	—
0 \rightarrow 1	1	—
1 \rightarrow 0	—	1
1 \rightarrow 1	—	0

Tabla 10: Determinación del valor de la entrada de un biestable J-K en función del estado actual y del estado deseado.

La tabla 11 muestra el cálculo de las entradas de excitación cuando utilizamos biestables J-K mientras que la tabla 12 hace lo mismo con biestables D. Obsérvese que no es necesario realizar una nueva tabla sino que se puede añadir nuevas columnas a la tabla E2.2 en donde se calcule los valores de las entradas de excitación de los biestables a usar en la síntesis del circuito. La simplificación para el caso de utilizar biestables J-K se muestra en la figura 19, mientras que en 20 se muestra la simplificación cuando utilizamos biestables tipo D.

Con biestables tipo J-K, el resultado de la simplificación sería:

$$J_1 = Q_0 \cdot X; K_1 = \bar{X} + Q_0; J_0 = X; K_0 = \bar{X} + \bar{Q}_1; S = Q_1 \cdot Q_0$$

Con biestables tipo D, el resultado de la simplificación sería:

$$D_1 = \bar{Q}_1 \cdot Q_0 \cdot X + Q_1 \cdot \bar{Q}_0 \cdot X; D_0 = Q_1 \cdot X + \bar{Q}_0 \cdot X; S = Q_1 \cdot Q_0$$

	$Q_1^*Q_0^*$		J_1K_1		J_0K_0	
Q_1Q_0	X=0	X=1	X=0	X=1	X=0	X=1
00	00	01	0-	0-	0-	1-
01	00	10	0-	1-	-1	-1
10	00	11	-1	-0	0-	1-
11	00	01	-1	-1	-1	-0

Tabla 11: Cálculo de los valores de las entradas de excitación de los dos biestables tipo J-K.

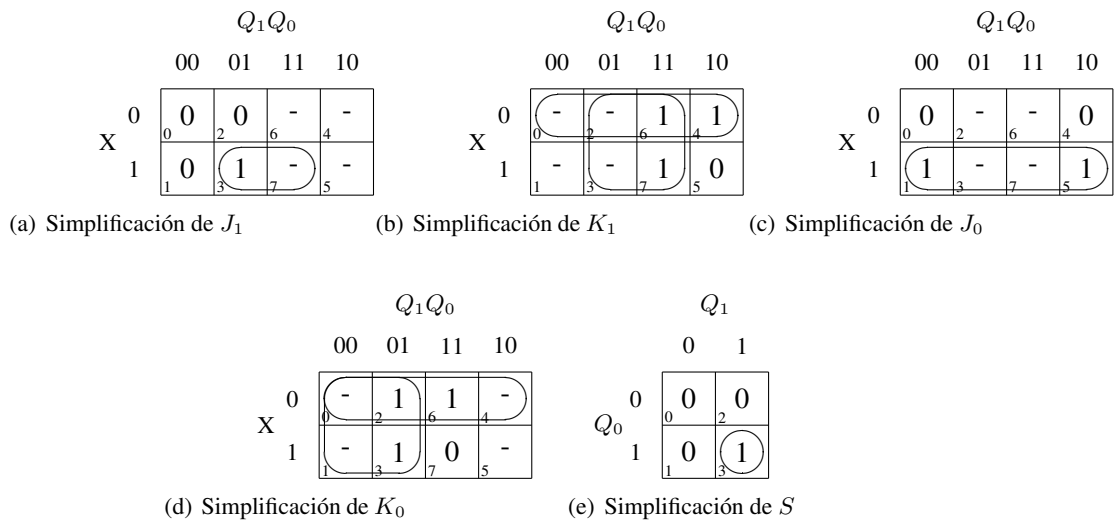


Figura 19: Simplificación de las entradas de los biestables JK y de la función de salida.

	$Q_1^*Q_0^*$		D_1D_0	
Q_1Q_0	X=0	X=1	X=0	X=1
00	00	01	00	01
01	00	10	00	10
10	00	11	00	11
11	00	01	00	01

Tabla 12: Cálculo de los valores de las entradas de excitación de los dos biestables tipo D.

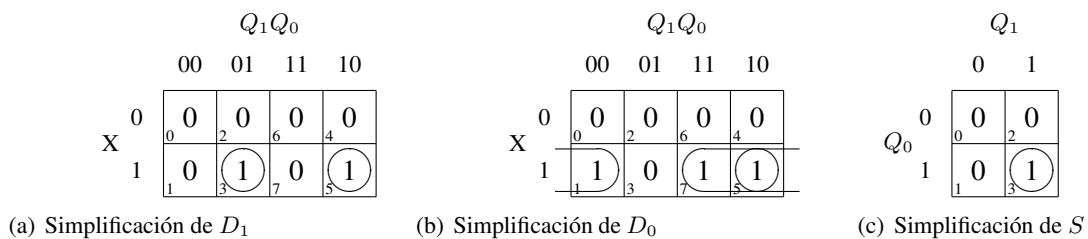


Figura 20: Simplificación de las entradas de los biestables D y de la función de salida.

Implementación del circuito

Ya sólo nos queda implementar el circuito con flip-flops y puertas lógicas. La figura 21 muestra cómo queda dicha implementación, en la que se han utilizado flip-flop tipo D activos en flanco ascendente (se deja para el alumno la implementación con biestables J-K por ser la misma directa a partir de las ecuaciones obtenidas en el apartado anterior).

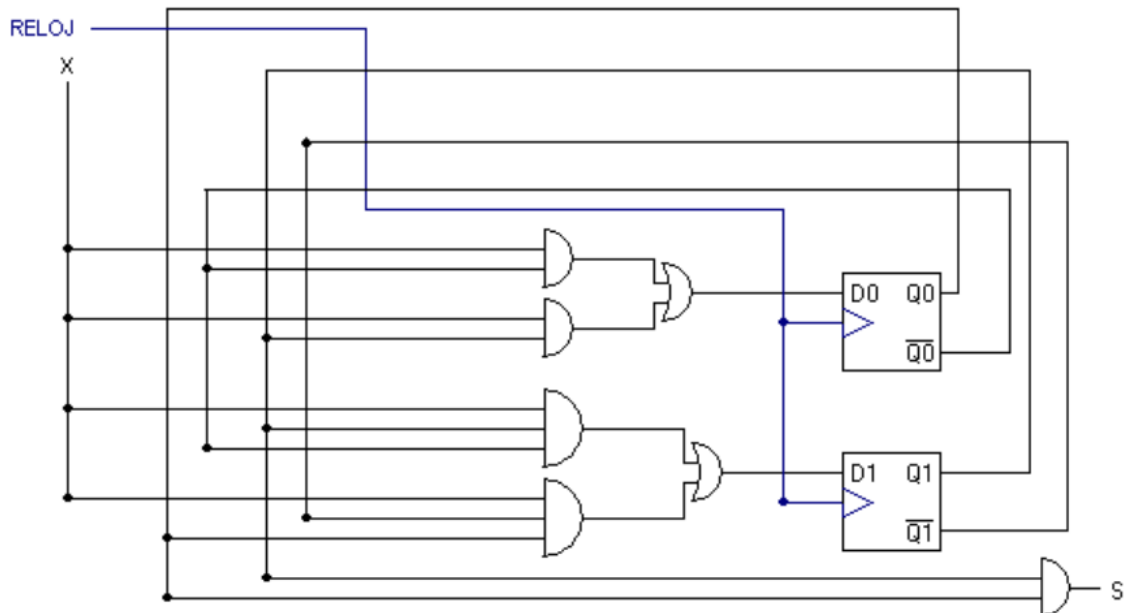


Figura 21: Implementación del circuito utilizando biestables tipo D (la implementación con biestables JK sería similar).

Finalmente, en la figura 22 mostramos un cronograma de ejemplo de funcionamiento del circuito, para una determinada secuencia de entrada. Si miramos los valores binarios de las señales, superpuestos sobre el cronograma, podemos observar cómo el valor de la entrada se lee en los instantes marcados por los flancos activos de la señal de reloj (en este caso el ascendente), mientras que los valores del estado y la salida pueden leerse a lo largo de todo el ciclo que comienza en ese momento, hasta que posiblemente cambien en el siguiente flanco activo. Por simplicidad, en el cronograma se supone un retardo nulo para todos los componentes del circuito.

Es también instructivo reflexionar sobre la frecuencia máxima de operación de un circuito de estas características. En principio, esta frecuencia máxima dependerá directamente del retardo introducido por sus componentes. La mejor manera de verlo es con un ejemplo. Si suponemos retardos despreciables para las conexiones, de 10 ns para las puertas lógicas AND y OR, y 20 para los flip-flops tipo D, observando el diagrama del circuito de la figura 21 nos daremos cuenta de que, como mínimo, desde que comienza un nuevo ciclo las señales de la entrada actual y el estado anterior atravesarán un nivel de flip-flops, un nivel de puertas AND y otro de puertas OR para que las entradas de los biestables queden de nuevo estables (nuevo estado siguiente). Las distintas puertas y flip-flops de cada tipo operan en paralelo entre sí. La salida, al calcularse con un solo nivel AND que opera en paralelo con los anteriores, no introduce retardo adicional. Así, el retardo total para actualizarse el estado será: 20 ns (flip-flop) + 10 ns (AND) + 10 ns (OR) = 40 ns.

Este es el tiempo mínimo que debemos dejar transcurrir antes de que llegue el próximo flanco activo de la señal de reloj, puesto que si no la actualización del estado no se realizará de forma correcta. Por tanto, la frecuencia máxima de operación sería de: $1/(40 \text{ ns}) = 1/(40 \cdot 10^{-9} \text{ s}) = 0,025 \cdot 10^9 \text{ Hz} = 25 \cdot 10^6 \text{ Hz} = 25 \text{ MHz}$.

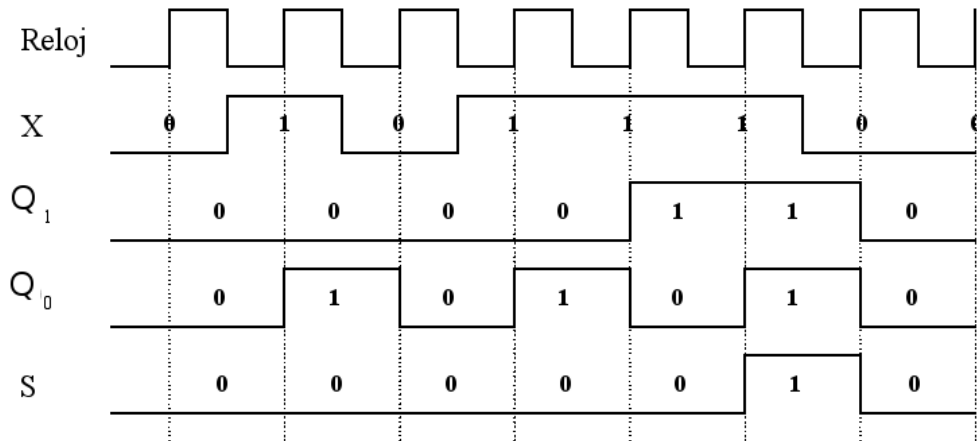


Figura 22: Cronograma de ejemplo mostrando el funcionamiento dinámico del circuito.

Apéndices

A2.1. Construcción física de una celda de memoria

En este apéndice analizaremos las distintas alternativas de diseño de una celda de memoria, tanto en su versión estática (SRAM) como dinámica (DRAM).

Celda SRAM

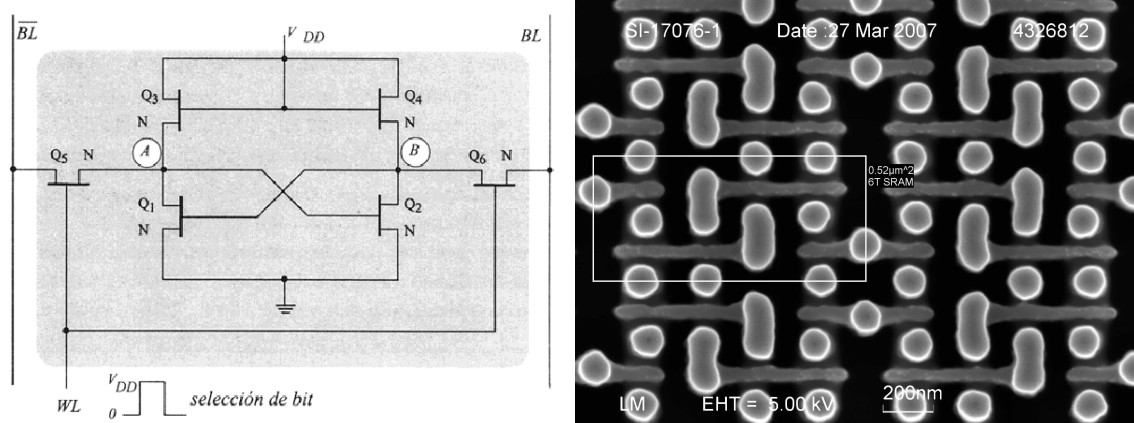


Figura A2.1: Implementación con transistores MOS de una celda de memoria SRAM: Esquemático (izq) y microfotografía (der).

La característica principal de las memorias SRAM es que el tiempo de acceso es muy corto, por ello se emplean principalmente en el diseño de memorias caché.

En la figura A2.1 se muestra el diseño de una celda básica de memoria SRAM utilizando transistores NMOS. Como vemos, se dispone de una patilla de selección de celda WL y de las líneas de datos BL y \overline{BL} . Partimos de dos inversores acoplados ($Q_1 - Q_3$ y $Q_2 - Q_4$) y se añaden otros dos ($Q_5 - Q_6$) para habilitar la lectura/escritura. Con este diseño, supondremos que se almacena un “1” cuando Q_1 está conduciendo y Q_2 está cortado. El funcionamiento sería el siguiente:

- **Selección de celda:** la selección de celda se producirá introduciendo un nivel alto (“1”) en WL , provocando que los transistores Q_5 y Q_6 conduzcan y que, por lo tanto, se pueda leer o escribir en la celda.
- **Lectura:** tras seleccionar la celda con WL , Q_5 y Q_6 conducirán y, por lo tanto, los valores de los puntos A y B aparecerán en las líneas BL y \overline{BL} , respectivamente.
- **Escritura:** se empieza seleccionando la celda mediante WL e introduciendo el dato que queremos almacenar (nivel lógico 1) por BL y \overline{BL} . Por ejemplo, supongamos que se quiere introducir un “0”; para ello, activaremos WL e introduciremos un “0” por BL y un “1” por \overline{BL} . Un “0” en BL provoca que la puerta de Q_2 se polarice a 0V y, por lo tanto, corte y la $V_{DS} = V_{DD}$. Dicha tensión se aplica a la puerta de Q_1 y por lo tanto $V_{GS1} = V_{DD}$. Esto provoca que Q_1 conduzca y, por lo tanto, su $V_{DS} = 0V$, reforzando y memorizando el nivel inicialmente introducido de 0V.

Finalmente, es importante darse cuenta que el circuito mostrado en la figura A2.1 no es sino la implementación física de un cerrojo tipo D, como el mostrado en la figura 3, en donde WL desempeña el papel de la entrada C , y BL y \overline{BL} equivalen a las líneas Q y \overline{Q} , respectivamente.

Celda DRAM

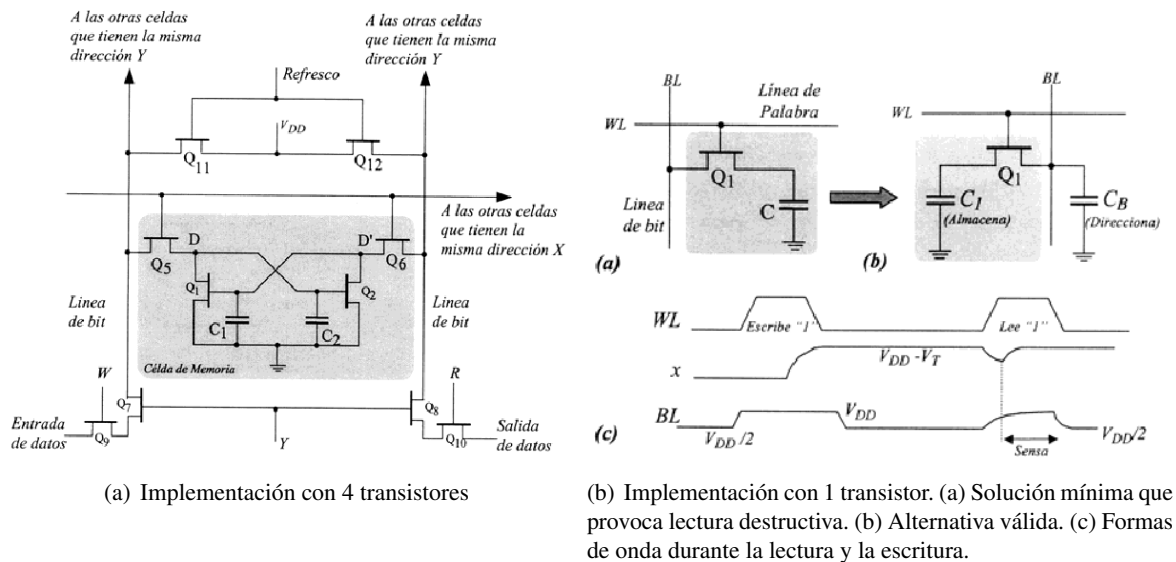


Figura A2.2: Celda de Memoria DRAM.

En la figura A2.2(a) se puede apreciar un primer diseño de celda DRAM. Si se compara con la celda estática se aprecia que se han eliminado dos transistores y que el elemento almacenador de información son los condensadores C_1 y C_2 . Estos condensadores no se implementan explícitamente en la celda, sino que son las capacidades parásitas de los propios transistores las que sirven como elementos almacenadores. Para evitar la pérdida de carga de estos condensadores se introduce un sistema de refresco a través de los transistores Q_{11} y Q_{12} que hace que la información de salida se vuelva a meter en la entrada.

El paso siguiente es pasar de cuatro a tres transistores. Se consigue en base a eliminar la redundancia de almacenamiento, ya que hasta este momento se almacena la información deseada y su complementaria. Como ahora la célula de almacenamiento es la capacidad parásita del transistor MOS, no es necesaria dicha redundancia y por lo tanto se puede eliminar la mitad de la celda. La última reducción del tamaño de celda en la RAM dinámica consiste en utilizar un único transistor y su capacidad parásita como elemento almacenador tal y como se aprecia en la figura A2.2(b).

Su funcionamiento es sencillo, basta un transistor que permita la entrada y salida de carga al condensador. Durante el ciclo de escritura se habilita la celda mediante WL y se introduce el nivel en la entrada BL ; al conducir Q_1 , dicha carga se introduce en C_1 y queda almacenada. Para su lectura, simplemente se habilita la celda mediante WL , provocando que Q_1 conduzca y que, por lo tanto, el valor de C_1 aparezca en BL .

Para finalizar, en la figura A2.3(arriba) se observa la evolución de las celdas DRAM conforme ha sido necesario reducir su tamaño para conseguir mayor capacidad de integración. Finalmente, en la figura A2.3(abajo) se muestra una microfotografía de una celda de la primera memoria, con una capacidad de un 1 Mbit, en donde se utilizó condensadores de zanja (*trench capacitor*) que permiten una mayor capacidad de integración.

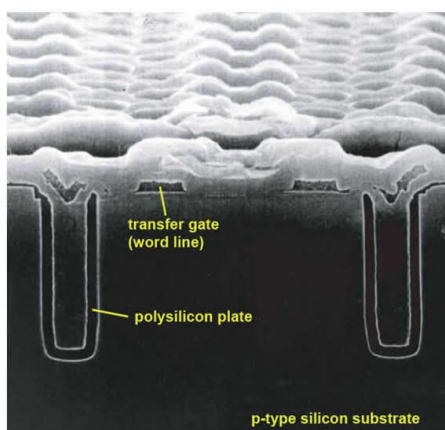
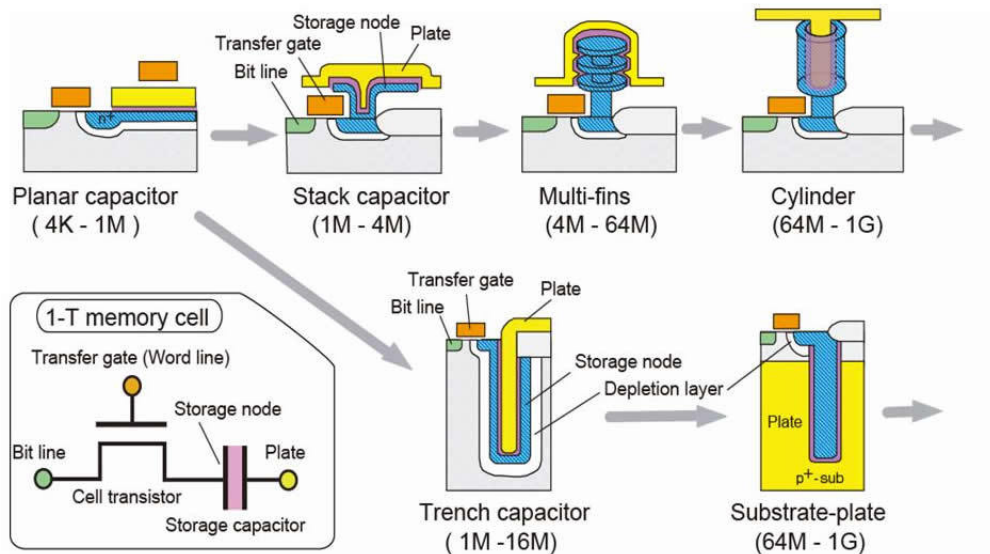


Figura A2.3: Evolución de la tecnología DRAM (arriba) y microfotografía del la primera memoria con condensadores de zanja (abajo).

Boletines de prácticas

B2.1. Normas sobre la entrega de prácticas

Será necesario seguir las siguientes reglas para entregar las prácticas:

- Las prácticas se entregarán mediante la opción de contenidos del alumno en SUMA.
- Se entregará un único archivo comprimido en formato *.tar.gz* o *.zip* que contendrá la memoria en formato PDF, los circuitos y programas que se hayan generado (código fuente) y cualquier otro fichero que se considere oportuno. El nombre del archivo será *prácticas-DNI-BOLETIN.FORMATO* (por ejemplo: *prácticas-12345678-B2.3.tar.gz*).
- La memoria incluirá, al menos, la siguiente información en un solo documento PDF:
 - Nombre y DNI del autor o autores de la práctica.
 - Descripción de los ficheros y directorios contenidos en el archivo entregado.
 - Contestación a las preguntas planteadas en los boletines. La respuesta a cada pregunta debe ser independiente, y debe estar claramente identificada.
 - Explicación de las pruebas realizadas para comprobar la corrección de la práctica entregada e instrucciones para su reproducción. Cuando sea posible, se incluirán los ficheros utilizados en dichas pruebas.
 - Explicación del trabajo realizado y cualquier aclaración que el alumno considere pertinente.
 - Lista de bibliografía y otras fuentes de información consultadas.

No se corregirá ninguna práctica que no se ciña estrictamente a los formatos especificados anteriormente.

B2.2. Implementación de un Archivo/Banco de Registros con TkGate

B2.2.1. Objetivos

El objetivo de esta sesión es que el alumno diseñe un Archivo/Banco de Registros, elemento secuencial fundamental en el diseño de un procesador tal y como veremos en temas posteriores, utilizando la herramienta TkGate.

B2.2.2. Prerequisitos

Haber leído los apuntes de teoría.

B2.2.3. Plan de trabajo

El plan de trabajo de esta sesión será el siguiente:

1. Lectura por parte del alumno de la sección **B2.2.4**.
2. Realización individual del ejercicio propuesto en el boletín (con supervisión del profesor).

B2.2.4. Archivo/Banco de Registros

Un archivo/banco de registros no es más que un conjunto de registros que pueden ser leídos y escritos selectivamente, a través de unas entradas y salidas que denominaremos, respectivamente, *puertos de escritura* y de *puertos de lectura*. La implementación se realiza mediante una serie de registros construidos mediante flip-flops tipo D, y usando multiplexores y decodificadores para seleccionar el/los registros concretos que se leen/escriben en los puertos de lectura y escritura, respectivamente. Como al leer un registro no cambia ningún estado, solamente necesitamos dar como entrada un número de registro (podemos pensar en este número como en una dirección), y en el puerto de salida correspondiente aparecerá el dato contenido en ese registro. Para escribir en un registro indicado por su número (o dirección) necesitaremos tres entradas: el número de registro, el dato a escribir y una señal de control de escritura que, junto con el reloj, controle el momento en el que se desea escribir el dato. En la figura B2.1 puede verse el esquema genérico de un archivo de M registros de n bits con dos puertos de lectura y uno de escritura. A la derecha aparece el bloque lógico de un archivo de 32 registros de 32 bits, con dos puertos de lectura y uno de escritura, como el que se empleará en la construcción del camino de datos del MIPS en el tema 4.

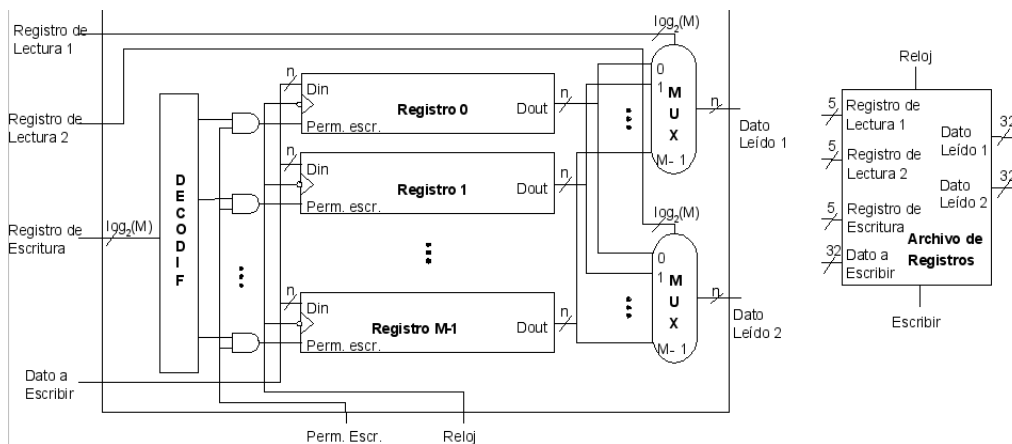


Figura B2.1: (Izquierda) Archivo de registros con 2 puertos de lectura y 1 puerto de escritura. (Derecha) Archivo de registros que usaremos en el camino de datos del MIPS.

B2.2.5. Ejercicios

Para cada ejercicio, el portafolios deberá incluir una explicación de cómo se ha resuelto el mismo, los ficheros con los diferentes módulos/circuitos y, para cada uno de ellos, un cronograma en donde se muestre el funcionamiento del mismo bajo diferentes situaciones con el fin de que se pueda apreciar el correcto funcionamiento del circuito.

1. Construir el módulo denominado "BufferTriestado8". Se trata de un buffer triestado en donde la entrada y salida es de 8 bits y que utilizaremos para implementar un multiplexor.
2. Utilizando como base el buffer triestado del apartado anterior y un decodificador de 3 a 8, cree un módulo denominado "Multiplexor8" que implemente un multiplexor de 8 a 1 con entradas y salidas de 8 bits. Compruebe el funcionamiento del circuito resultante mediante un cronograma en el que se pueda observar la respuesta del circuito ante diversos valores de las señales de entrada.
3. Utilizando 8 registros tipo D, diseñe un archivo de registros con 2 puertos de lectura y 1 de escritura capaz de dar acceso a los 8 registros mencionados. El archivo de registros poseerá además una entrada adicional, CLR, activa en alta que permita poner a cero el contenido de todos los registros. Encapsule

el circuito resultante en el módulo "BancoRegistros8". El encapsulado resultante debería ser similar al que se muestra en la figura B2.2.

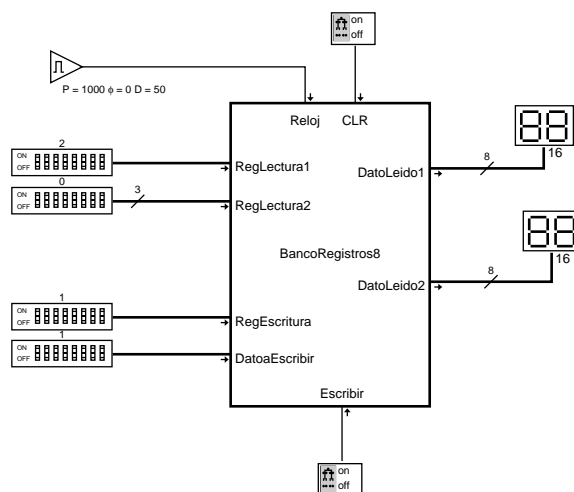


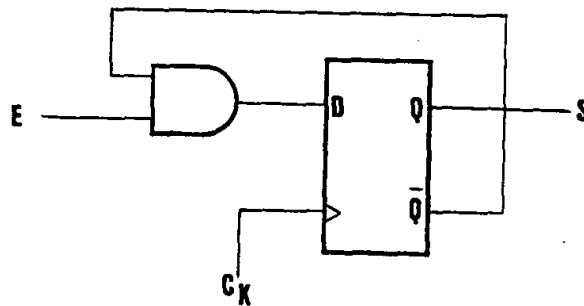
Figura B2.2: Banco de 8 registros con dos puertos de lectura, uno de escritura y señal de *clear*.

4. Compruebe el funcionamiento del circuito mediante un cronograma en el que se pueda observar diversas operaciones de lectura y escritura (al menos 5). Determine la frecuencia máxima a la que puede funcionar este circuito.

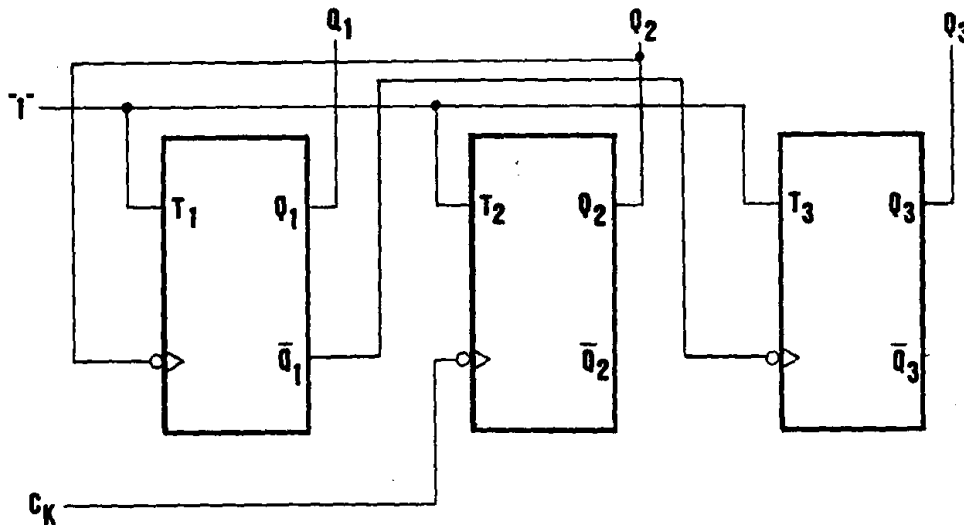
Ejercicios

E2.1. Sistemas Digitales: Circuitos Secuenciales

1. Diseñar un circuito secuencial síncrono, utilizando flip-flops tipo J-K, con 1 variable de entrada X, tal que tenga como salida Z un 1 si y sólo si los dos últimos bits de entrada recibidos valían igual (dos ceros o dos unos).
2. Obtener la tabla de estados del circuito de la figura. ¿Realiza la función de algún circuito conocido?



3. Dibujar el cronograma y obtener la secuencia de cuenta del contador de la figura:

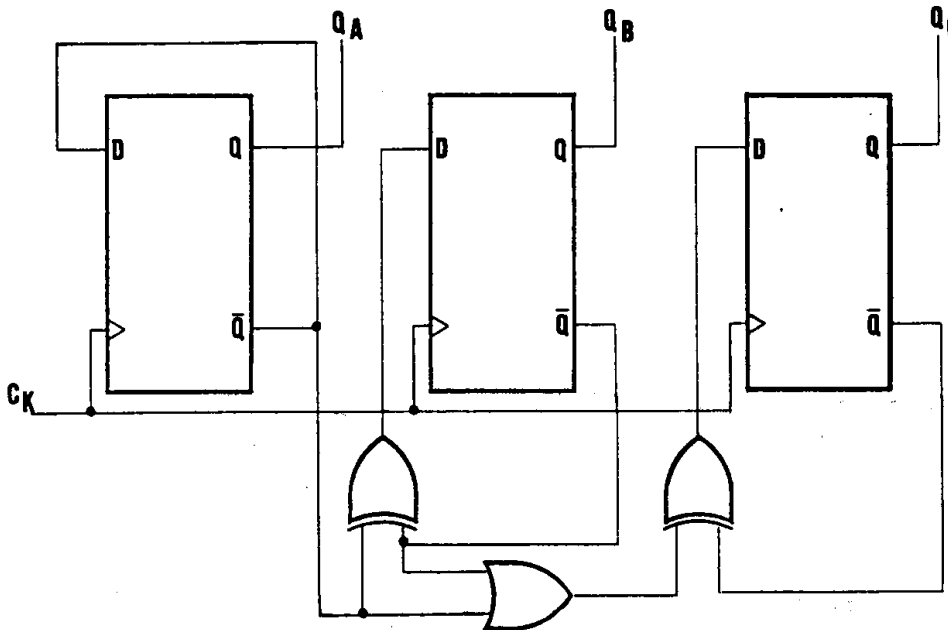


¿Cuál es la frecuencia máxima de funcionamiento si el retraso de un biestable es de 40 ns?

4. Diseñar un circuito secuencial síncrono, utilizando flip-flops tipo D, sin variables de entrada (excepto el reloj), tal que tome como salida, sucesivamente, los valores 0, 1, 2 y 3 en binario natural (es decir, 00, 01, 10 y 11; es un contador síncrono de 2 bits, o lo que es lo mismo, de 4 estados).
5. Diseñar un circuito, con el mínimo número de componentes, que proporcione a sus 3 salidas el equivalente binario de la secuencia 5 - 2 - 4 - 1, en cuanto reciba una orden de activación por su entrada *START*. Cuando esta orden cese, el circuito quedará con el estado en que se encontrase en ese instante, esperando a que se produzca de nuevo la activación de *START* para comenzar a generar la secuencia. Utilizar biestables tipo S-R.
6. Se desea activar un circuito de toma de datos cuando la información presente en 3 líneas de los buses del

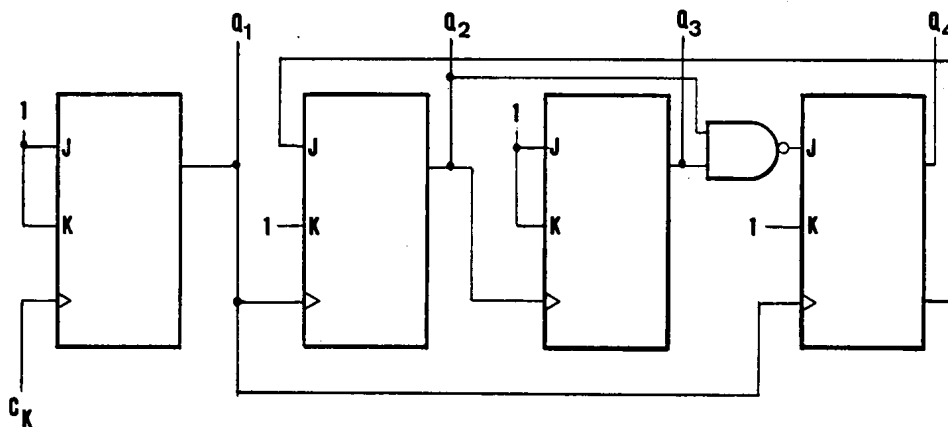
circuito bajo test evolucionan del siguiente modo: 000 - 001 - 010 - 100. Realizar el diseño utilizando flip-flops tipo T.

7. Diseñar un contador síncrono con biestables J-K que cuente según la secuencia: 0 - 1 - 6 - 7 - 2 - 3 - 4 - 5 - 0. Calcular la máxima frecuencia de funcionamiento, si el retraso de un biestable J-K es de 50 ns y el de una puerta AND/OR 10 ns.
8. Dibujar el cronograma y obtener la secuencia de cuenta del contador de la figura:



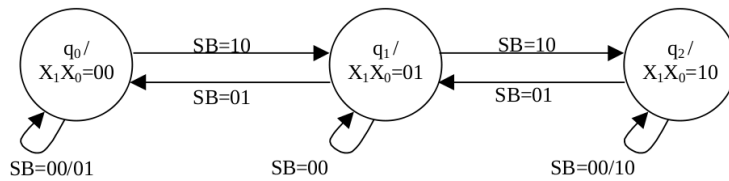
¿Cuál es la frecuencia máxima de funcionamiento si el retraso de un biestable es de 40 ns, el de una puerta AND 20 ns y el de una XOR 25 ns?

9. Ídem para la figura:



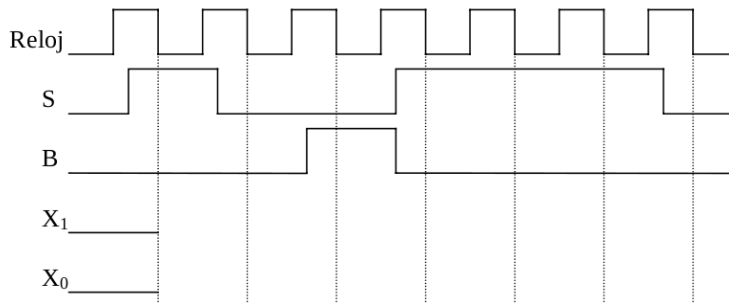
¿Cuál es la frecuencia máxima de funcionamiento si el retraso de un biestable es de 40 ns, el de una puerta AND 20 ns y el de una XOR 25 ns?

10. El autómata de Moore de la siguiente figura expresa el funcionamiento requerido para un circuito secuencial síncrono que implementará un contador ascendente/descendente de tres estados. Se observa que el circuito debe tener dos entradas, S y B (de Sube y Baja, respectivamente), y como salida un valor binario natural de dos bits, X_1 y X_0 .

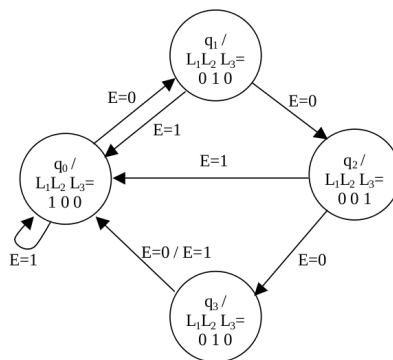
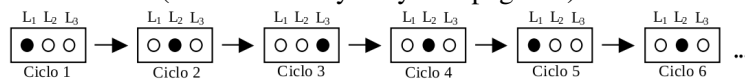


Cuando las entradas S y B valgan ambas cero, el circuito no cambia de estado. Cuando S valga uno y B cero, entonces el número de estado se aumenta en una unidad, salvo que ya estemos en el estado más alto (de 00 se pasa a 01, de 01 a 10, y si estamos en 10 permanecemos en él). Si B vale uno y S cero, entonces el estado se decremanta, salvo que ya estemos en el estado más bajo (de 10 se pasa a 01, de 01 se pasa a 00, y si estamos en 00 permanecemos en él). La entrada S=B=1 no está permitida.

- Construir el circuito secuencial síncrono simplificado que implemente dicho autómata, utilizando flip-flops tipo T, activos en el flanco descendente.
- Rellenar el siguiente cronograma de funcionamiento:

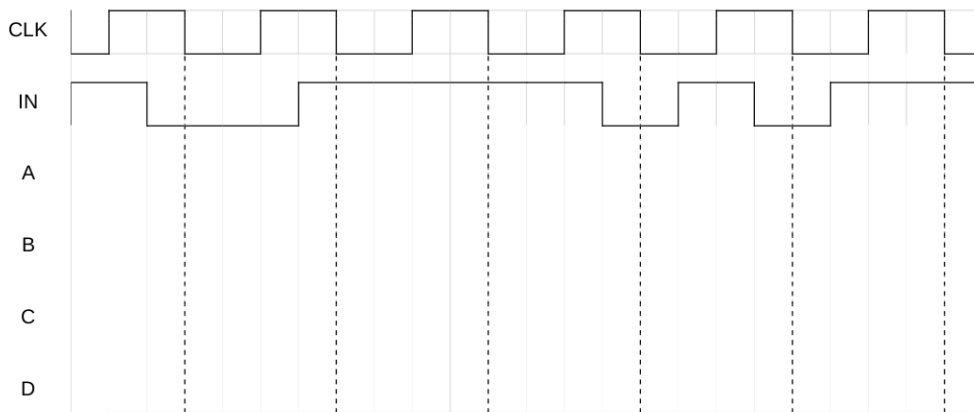


11. El siguiente autómata de Moore expresa el funcionamiento requerido para un circuito secuencial síncrono que implementa un controlador de un sistema de tres luces (L1 L2 L3) que se van encendiendo sucesivamente de izquierda a derecha y de derecha a izquierda, del modo indicado. Se observa que el autómata tiene una sola entrada E, que hace la función de *reset*, de modo que mientras vale 0 el sistema opera en el modo cíclico habitual anteriormente indicado, pero que cuando vale uno fuerza al sistema a colocarse en su estado inicial (L1 encendido y L2 y L3 apagados).



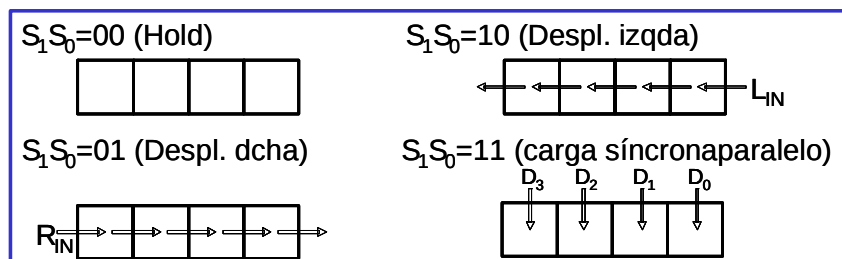
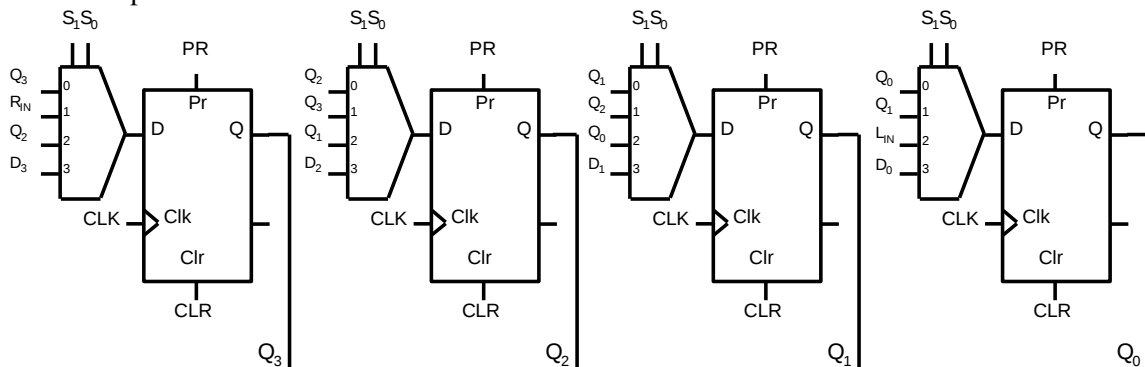
Se pide construir el circuito secuencial síncrono simplificado que implemente dicho autómata, utilizando flip-flops tipo J-K, activos en el flanco descendente.

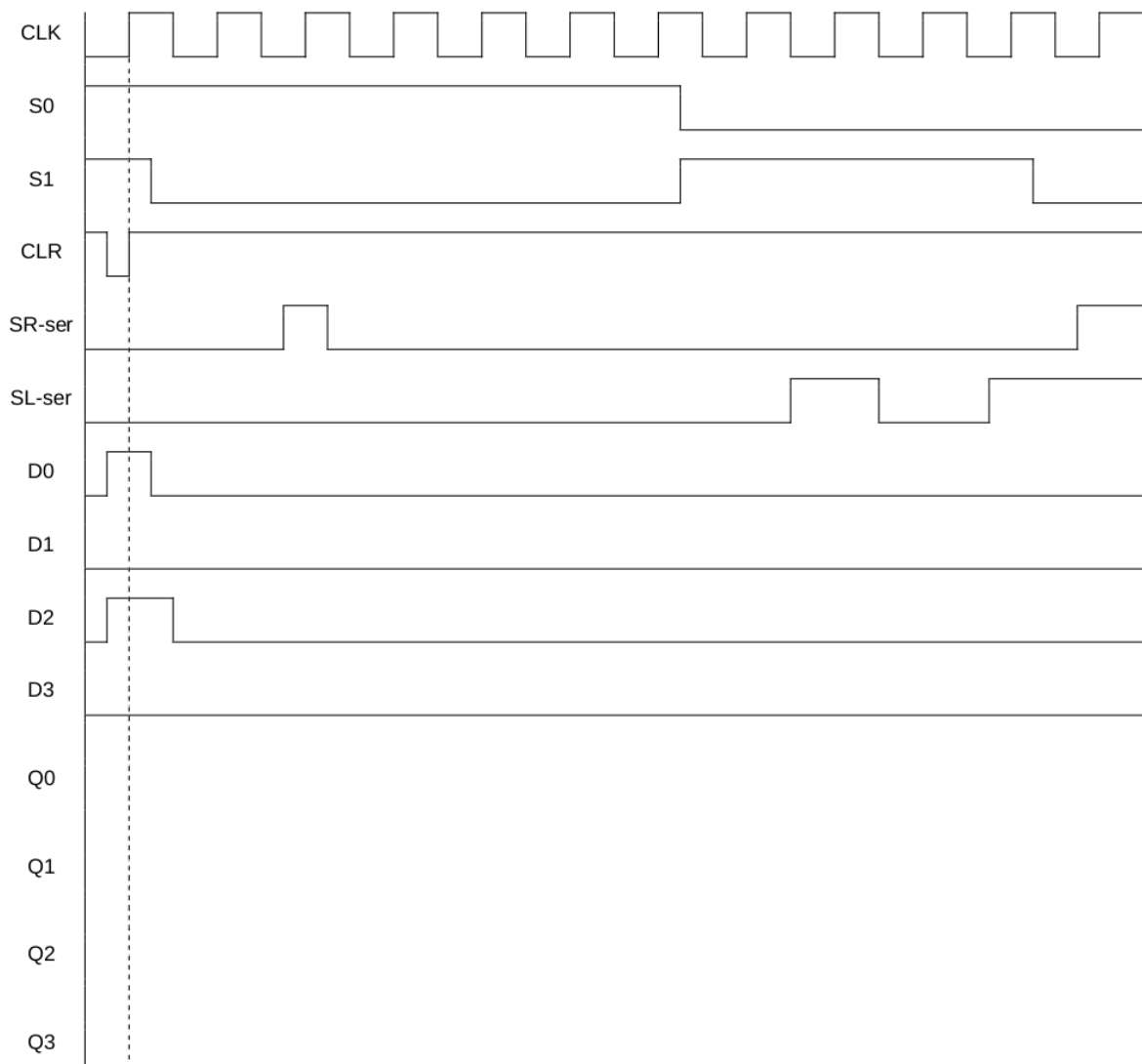
12. Repetir el ejercicio anterior, pero cambiando el controlador para un sistema de 4 luces (en lugar de 3), que siga la misma regla de funcionamiento (de izquierda a derecha y viceversa, también con una señal de *reset*). Utilizar biestables tipo T en lugar de J-K.



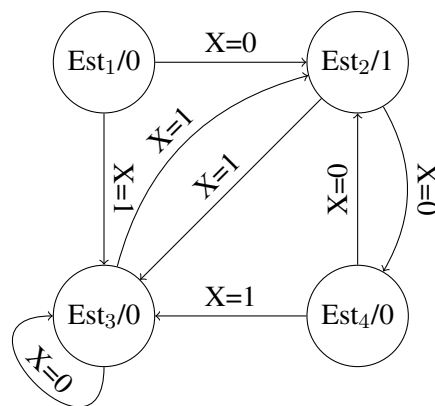
16. Determinar el estado en cada pulso de reloj de un registro de desplazamiento universal bidireccional de 4 bits en el que se aplican las señales de entrada y de control mostradas (sincronización con flanco de subida del reloj). Suponer que solo CLR es asíncrona y que las señales de control S0-S1 codifican el siguiente comportamiento: S0=S1=1, carga paralela (D0-D3); S0=S1=0, no hay cambios; S0=0 y S1=1, desplazamiento a la izquierda; y S0=1 y S1=0, desplazamiento a la derecha. SR-ser y SL-ser son las entradas de datos serie derecha e izquierda respectivamente.

A continuación se muestra un esquema de la implementación del desplazador y el cronograma que se debe completar.





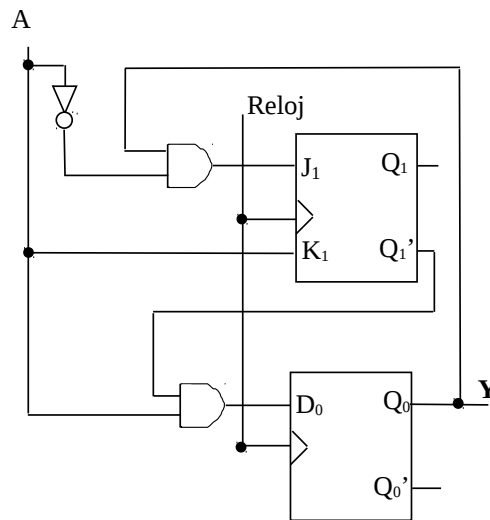
17. Diseñe el circuito secuencial síncrono con una entrada X y un bit de salida (S) que vendría especificado por el siguiente autómata finito determinista, utilizando flip-flops T disparados por flanco descendente:



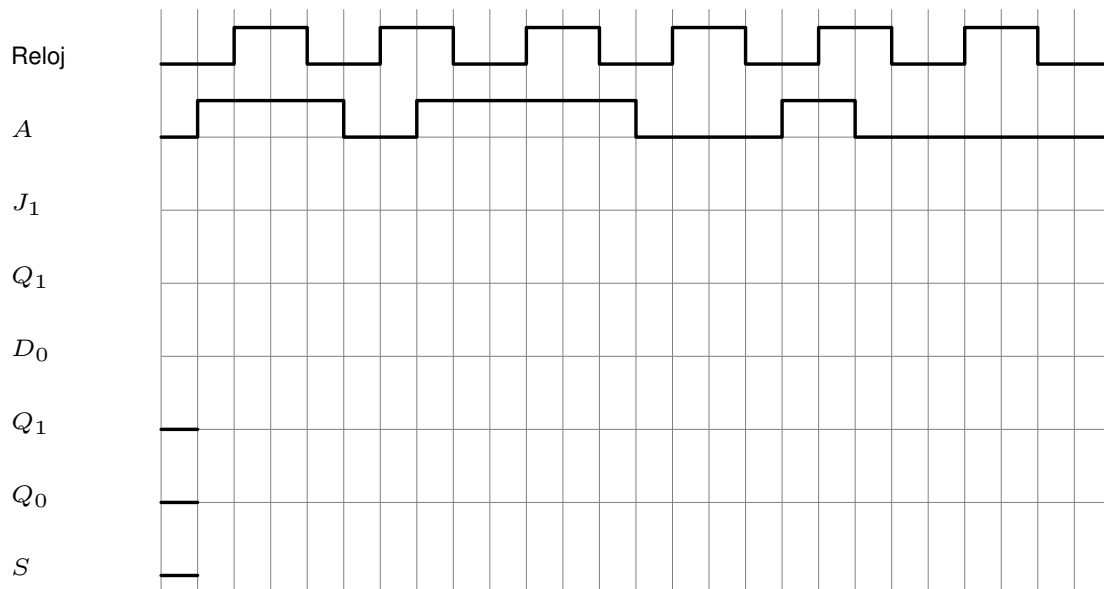
18. Implementar, utilizando biestables tipo D disparados por flanco descendente, un circuito capaz de de-

testar un retraso de un ciclo entre dos señales de un bit. El circuito tendrá dos entradas, A y B, y una salida, S, que se pondrá a 1 siempre que la entrada B se corresponda con la A del ciclo anterior. En el momento en que esta condición no se cumpla la salida quedará fijada a 0 hasta que una señal de Reset, activa en baja, vuelva a iniciar el proceso de comprobación. En el primer ciclo de reloj, donde la salida todavía no tiene sentido, supondremos que se cumple la condición. (Pista: Observad que solamente necesito saber si hasta el momento se está cumpliendo la condición y el valor anterior de A para poder implementar el circuito.)

19. Dado el sistema secuencial síncrono que se muestra a continuación se pide:



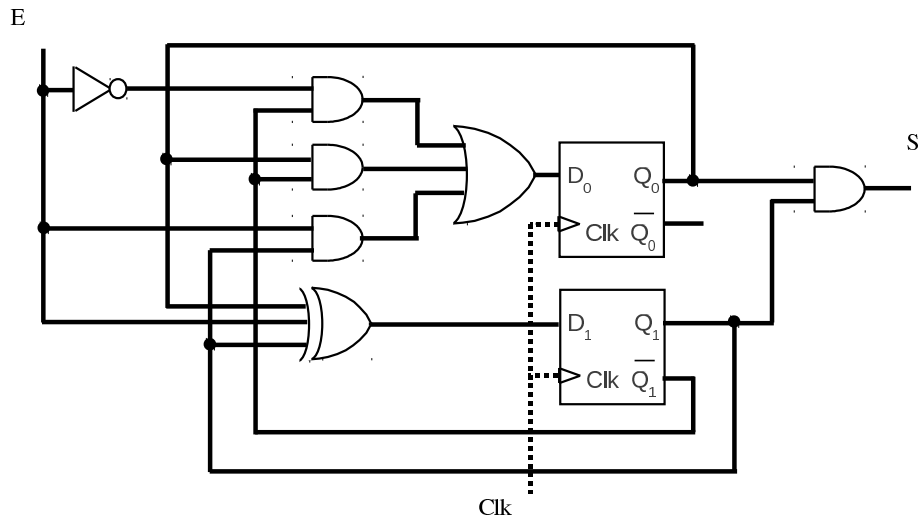
a) Rellenar el siguiente cronograma:



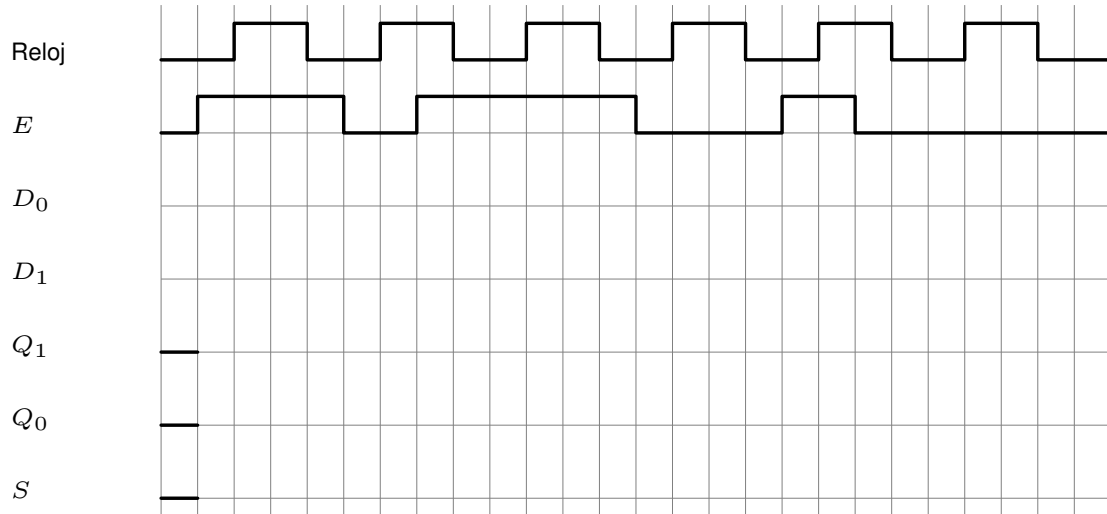
- b) Obtener el autómata de Moore que modela el comportamiento del circuito, sabiendo que inicialmente ambos biestables tienen salida 0. (Nota: el autómata consta de 3 estados).
- c) Calcular la frecuencia máxima a la que puede funcionar el circuito suponiendo un retardo de 20 ns

para los biestables, 5 ns para las puertas NOT y 10 ns para la puerta AND. (Nota: suponer que la entrada del circuito no varía una vez se produce el flanco activo del reloj).

20. Dado el sistema secuencial síncrono que se muestra a continuación se pide:



a) Rellenar el siguiente cronograma:



- b) Obtener el autómata de Moore que modela el comportamiento del circuito, sabiendo que, inicialmente, ambos biestables tienen salida 0.
- c) Calcular la frecuencia máxima a la que puede funcionar el circuito suponiendo un retardo de 20 ns para los biestables, 5 ns para las puertas NOT, 15 para las puertas XOR y 10 ns para las puertas AND y para las puertas OR.

E2.2. Solución a ejercicios seleccionados

5. Diseñar un circuito, con el mínimo número de componentes, que proporcione a sus 3 salidas el equivalente binario de la secuencia 5 - 2 - 4 - 1, en cuanto reciba una orden de activación por su entrada *START*. Cuando esta orden cese, el circuito quedará con el estado en que se encontrase en ese instante, esperando a que se produzca de nuevo la activación de *START* para comenzar a generar la secuencia. Utilizar biestables tipo S-R.

Solución:

El autómata que modeliza el sistema a implementar constará de 4 estados, correspondientes con las 4 posibles salidas a mostrar. La única entrada del sistema es la señal *START* que habilita la transición entre estados. El autómata será por tanto el indicado en E2.1.

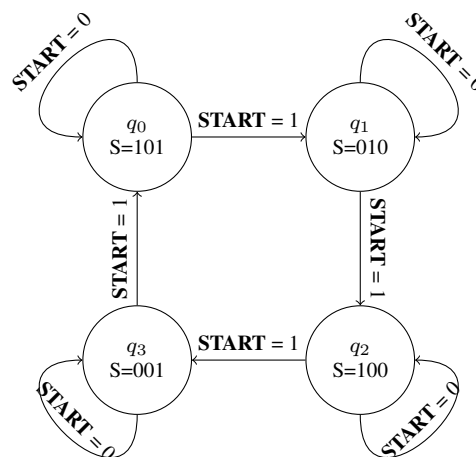


Figura E2.1: Autómata que modeliza el sistema secuencial a implementar.

Representado en forma de tabla tendremos:

Estado Actual	Estado Siguiete		Salida $Sal_2Sal_1Sal_0$
	$START = 0$	$START = 1$	
q_0	q_0	q_1	101
q_1	q_1	q_2	010
q_2	q_2	q_3	100
q_3	q_3	q_0	001

Tabla E2.1: Tabla de transición entre estados y función salida para nuestro autómata.

Al haber 4 estados, es suficiente con dos flip-flops cuyas salidas serán Q_0 y Q_1 . Utilizaremos la codificación en binario natural ($q_0 = 00$, $q_1 = 01$, $q_2 = 10$, $q_3 = 11$). Una vez realizada la codificación de estados el siguiente paso es reescribir dicha tabla haciendo uso de la codificación elegida tal y como se muestra en la tabla E2.2. Obsérvese que diferenciamos el estado futuro (Q_i^*) del actual (Q_i) utilizando un asterisco.

Una vez realizada dicha codificación debemos determinar qué valores deben de tener las entradas de los biestables para que dicha transición se realice en la salida. En el caso de biestables tipo S-R usaremos la tabla E2.3.

Q_1Q_0	$Q_1^*Q_0^*$		Salida
	$START = 0$	$START = 1$	$Sal_2Sal_1Sal_0$
00	00	01	101
01	01	10	010
10	10	11	100
11	11	00	001

Tabla E2.2: Tabla de transición entre estados y función de salida con los estados ya codificados.

$Q \rightarrow Q^*$	S	R
$0 \rightarrow 0$	0	—
$0 \rightarrow 1$	1	0
$1 \rightarrow 0$	0	1
$1 \rightarrow 1$	—	0

Tabla E2.3: Determinación del valor de la entrada de un biestable S-R en función del estado actual y del estado deseado.

La tabla E2.4 muestra el cálculo de las entradas de excitación cuando utilizamos biestables S-R.

Q_1Q_0	$Q_1^*Q_0^*$		S_1R_1		S_0R_0	
	$START = 0$	$START = 1$	$START = 0$	$START = 1$	$START = 0$	$START = 1$
00	00	01	0 —	0 —	0 —	10
01	01	10	0 —	10	— 0	01
10	10	11	— 0	— 0	0 —	10
11	11	00	— 0	01	— 0	01

Tabla E2.4: Cálculo de los valores de las entradas de excitación de los dos biestables tipo S-R.

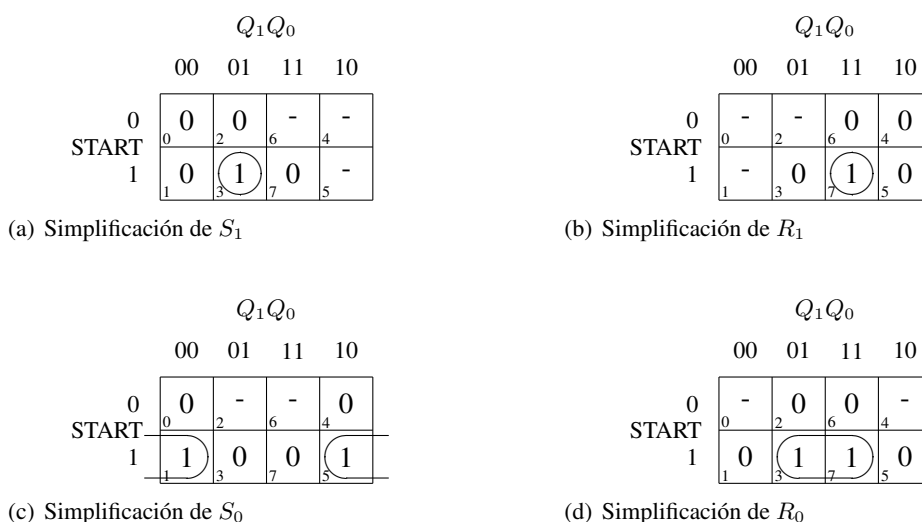


Figura E2.2: Simplificación de las entradas de los biestables S-R.

El siguiente paso consiste en la simplificación de las entradas de excitación de ambos biestables así como de las tres salidas. La figura E2.2 muestra la simplificación de las mismas utilizando mapas de karnaugh. Los valores obtenidos son:

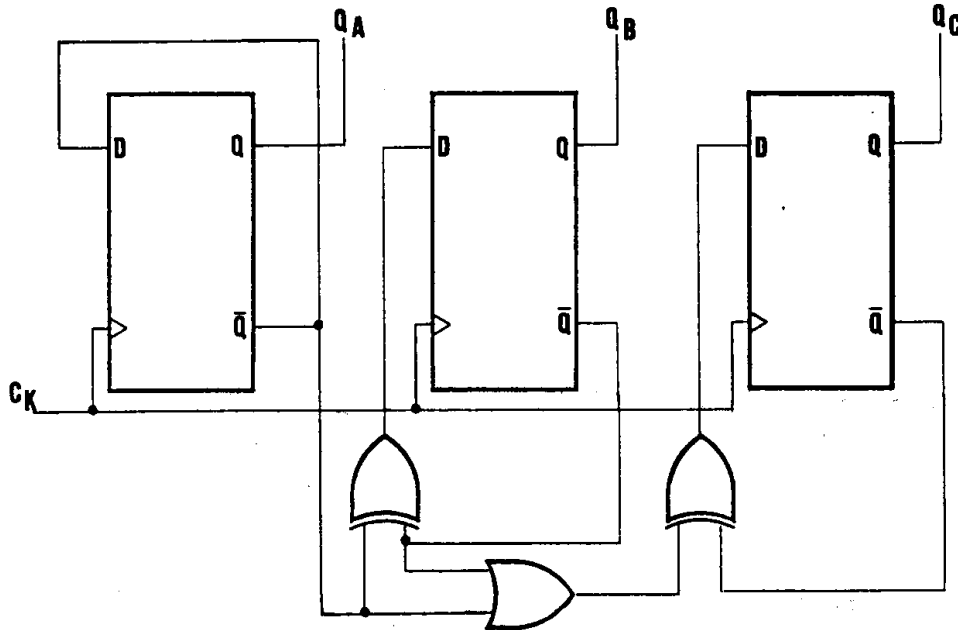
$$S_1 = \overline{Q_1} \cdot Q_0 \cdot START; R_1 = Q_1 \cdot Q_0 \cdot START$$

$$S_0 = \overline{Q_0} \cdot START; R_0 = Q_0 \cdot START$$

$$Sal_2 = \overline{Q_0}; Sal_1 = \overline{Q_1} \cdot Q_0; Sal_0 = \overline{Q_1} \cdot \overline{Q_0} + Q_1 \cdot Q_0 = \overline{Q_1} \oplus \overline{Q_0}$$

El dibujo del circuito resultante se deja como ejercicio para el alumno.

8. Dibujar el cronograma y obtener la secuencia de cuenta del contador de la figura:



¿Cuál es la frecuencia máxima de funcionamiento si el retraso de un biestable es de 40 ns, el de una puerta AND 20 ns y el de una XOR 25 ns?

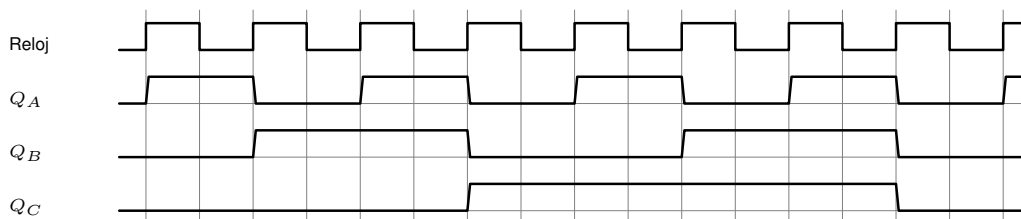
Solución:

Del circuito obtenemos las siguientes ecuaciones del estado siguiente:

$$Q_A^* = \overline{Q_A}$$

$$Q_B^* = \overline{Q_A} \oplus \overline{Q_B} = Q_A \oplus Q_B$$

$$Q_C^* = (\overline{Q_A} + \overline{Q_B}) \oplus \overline{Q_C} = Q_A Q_B \overline{Q_C} + \overline{Q_A} Q_C + \overline{Q_B} Q_C$$



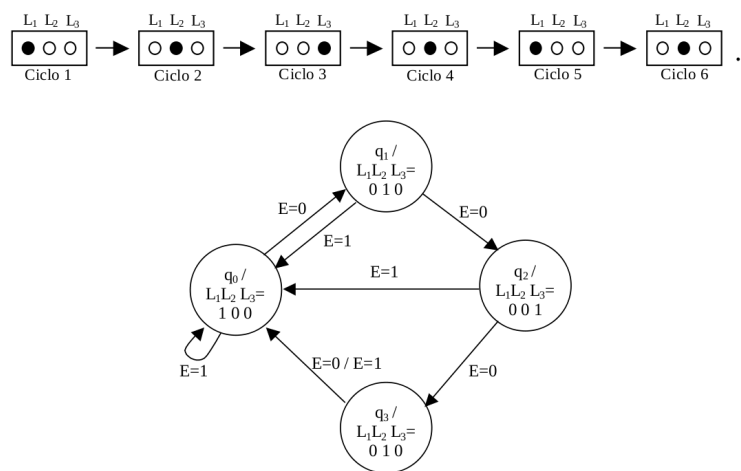
Se trata, por tanto, de un contador binario de 3 bits cuya cuenta es: 0-1-2-3-4-5-6-7.

La frecuencia máxima de funcionamiento viene determinada por el camino más largo desde que se produce el flanco activo del reloj (flanco ascendente) hasta que las salidas de los biestables y las entradas de excitación de los mismos están estables. En nuestro caso, el camino más largo es la entrada D del biestable cuya salida es Q_C .

$$T_{min} = T_{FF} + T_{OR} + T_{XOR} = 40 + 20 + 25 = 85ns$$

$$f_{max} = \frac{1}{T_{min}} = \frac{1000}{85} \times 10^6 Hz = 11,76MHz$$

11. El siguiente autómata de Moore expresa el funcionamiento requerido para un circuito secuencial síncrono que implementa un controlador de un sistema de tres luces (L1 L2 L3) que se van encendiendo sucesivamente de izquierda a derecha y de derecha a izquierda, del modo indicado. Se observa que el autómata tiene una sola entrada E, que hace la función de *reset*, de modo que mientras vale 0 el sistema opera en el modo cíclico habitual anteriormente indicado, pero que cuando vale uno fuerza al sistema a colocarse en su estado inicial (L1 encendido y L2 y L3 apagados).



Se pide construir el circuito secuencial síncrono simplificado que implemente dicho autómata, utilizando flip-flops tipo J-K, activos en el flanco descendente.

Solución:

Dado que tenemos 4 estados, necesitaremos dos flip-flops para codificarlos (Q_0 y Q_1). Utilizaremos la siguiente codificación para los estados:

Estado	Q_0	Q_1
q0	0	0
q1	0	1
q2	1	0
q3	1	1

Tendremos tres funciones de salida (o una función de 3 bits) cuya tabla de verdad será:

Q_0	Q_1	L_1	L_2	L_3
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	1	0

Y la tabla de la función de transición (Q_0^* y Q_1^*) y las entradas de los dos biestables (J_0, K_0, J_1 y K_1) será:

Q_0	Q_1	E	Q_0^*	Q_1^*	J_0	K_0	J_1	K_1
0	0	0	0	1	0	—	1	—
0	0	1	0	0	0	—	0	—
0	1	0	1	0	1	—	—	1
0	1	1	0	0	0	—	—	1
1	0	0	1	1	—	0	1	—
1	0	1	0	0	—	1	0	—
1	1	0	0	0	—	1	—	1
1	1	1	0	0	—	1	—	1

Las expresiones simplificada de las funciones de salida serán:

$$L_1 = \overline{Q_0} \overline{Q_1}$$

$$L_2 = \overline{Q_0} Q_1 + Q_0 Q_1 = Q_1$$

$$L_3 = Q_0 \overline{Q_1}$$

La simplificación de la función J_0 es (otras soluciones también son posibles):

		$Q_0 Q_1$			
		00	01	11	10
E	0	0	1	X	X
	1	0	0	X	X

$$J_0 = Q_1 \overline{E}$$

La de K_0 :

		$Q_0 Q_1$			
		00	01	11	10
E	0	X	X	1	0
	1	X	X	1	1

$$K_0 = Q_1 + E$$

La de J_1 :

		$Q_0 Q_1$			
		00	01	11	10
E	0	1	X	X	1
	1	0	X	X	0

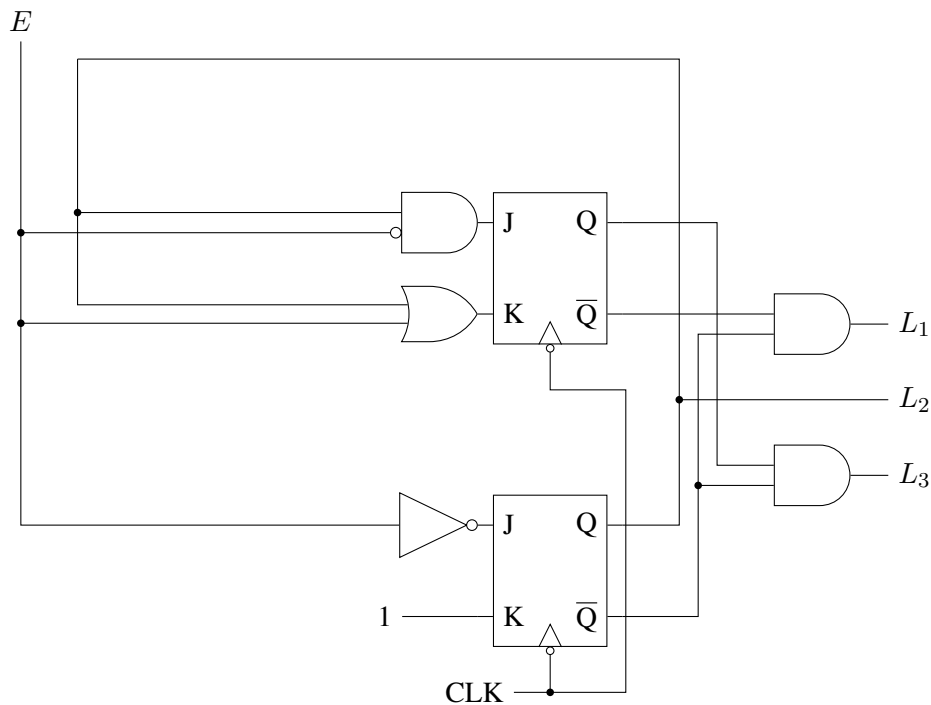
$$J_1 = \overline{E}$$

Y la de K_1 :

		Q_0Q_1			
		00	01	11	10
E	0	X ₀	1 ₂	1 ₆	X ₄
	1	X ₁	1 ₃	1 ₇	X ₅

$$K_1 = 1$$

Con lo que el circuito resultante usando puertas AND, OR y NOT será:



13. Diseñe un circuito secuencial síncrono con biestables tipo D disparados por flanco ascendente y usando un autómata de Moore que tenga tres entradas (E_2 , E_1 y E_0) y una salida (S). La salida se activará si el valor de las tres entradas ha coincidido durante los tres últimos ciclos. Es decir, si $E_i(t)$ es el valor de la entrada E_i al comienzo del ciclo t , S valdrá 1 en el ciclo t si y sólo si $E_0(t) = E_1(t) = E_2(t)$, $E_0(t-1) = E_1(t-1) = E_2(t-1)$ y $E_0(t-2) = E_1(t-2) = E_2(t-2)$.

- a) Dibuje el autómata que modela el comportamiento del circuito, incluyendo para cada estado el valor de la salida.

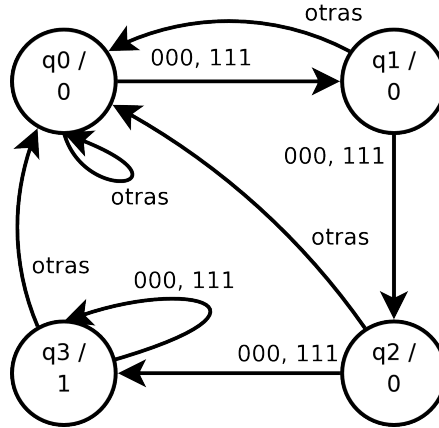
Solución:

Necesitaremos 4 estados para representar la siguiente información:

q_0 : No ha habido coincidencia en el ciclo anterior.

- q_1 : Ha habido coincidencia en el ciclo anterior pero no en el previo.
- q_2 : Ha habido coincidencia en los dos ciclos anteriores pero no en el previo.
- q_3 : Ha habido coincidencia en los tres ciclos anteriores.

Con estos estados, el autómata para el circuito sería el siguiente:



Obsérvese que lo único importante de las entradas es si coinciden o no, por lo que podríamos implementar un autómata con una sola entrada si realizáramos una comparación antes de calcular la función de transición, de forma que la entrada del autómata fuera 0 para entradas no coincidentes y 1 para entradas coincidentes (o viceversa).

- b) Realice la asignación de estados que crea conveniente y obtenga las expresiones minimizadas de las funciones de transición y de salida.

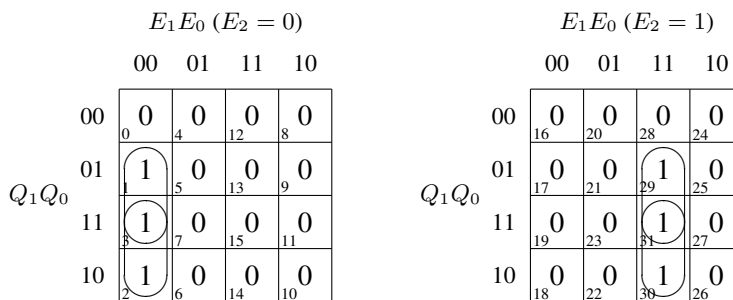
Solución:

Para codificar los 4 estados, necesitaremos 2 flip-flops tipo D cuyas salidas serán Q_0 y Q_1 . Como codificación de estados usaremos la usual: $q_0 = 00, \dots, q_3 = 11$.

Q_1Q_0	$Q_1^*Q_0^* = D_1D_0$			S
	$E_2E_1E_0 = 000$	$E_2E_1E_0 = 111$	$E_2E_1E_0 = otras$	
00 (q_0)	01	01	00	0
01 (q_1)	10	10	00	0
10 (q_2)	11	11	00	0
11 (q_3)	11	11	00	1

$S = Q_1Q_0$

El mapa de Karnaugh para D_1 es:



$D_1 = D_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_0 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_1 \cdot E_2 \cdot E_1 \cdot E_0 + D_0 \cdot E_2 \cdot E_1 \cdot E_0$

El mapa de Karnaugh para D_0 es:

		$E_1 E_0 (E_2 = 0)$			
		00	01	11	10
$Q_1 Q_0$	00	1	0	0	0
	01	0	0	0	0
	11	1	0	0	0
	10	1	0	0	0

		$E_1 E_0 (E_2 = 1)$			
		00	01	11	10
$Q_1 Q_0$	00	0	0	1	0
	01	0	0	0	0
	11	0	0	1	0
	10	0	0	1	0

$$D_0 = D_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + \overline{D_0} \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_1 \cdot E_2 \cdot E_1 \cdot E_0 + \overline{D_0} \cdot E_2 \cdot E_1 \cdot E_0$$

c) Dibuje el circuito que implementa el autómata usando solo puertas NAND y biestables tipo D.

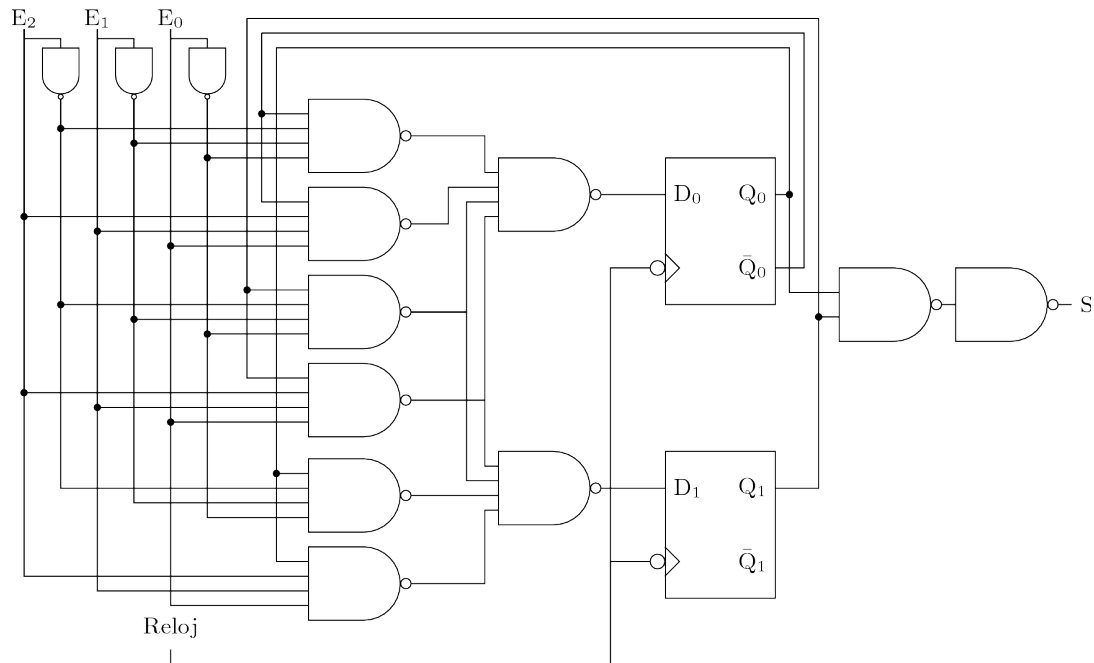
Solución:

Transformamos las expresiones de las funciones de salida y transición usando las leyes de De Morgan:

$$S = Q_1 Q_0 = \overline{\overline{Q_1 Q_0}}$$

$$\begin{aligned} D_1 &= D_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_0 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_1 \cdot E_2 \cdot E_1 \cdot E_0 + D_0 \cdot E_2 \cdot E_1 \cdot E_0 = \\ &= \overline{\overline{D_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_0 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_1 \cdot E_2 \cdot E_1 \cdot E_0 + D_0 \cdot E_2 \cdot E_1 \cdot E_0}} = \\ &= \overline{(D_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0}) \cdot (D_0 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0}) \cdot (D_1 \cdot E_2 \cdot E_1 \cdot E_0) \cdot (D_0 \cdot E_2 \cdot E_1 \cdot E_0)} \end{aligned}$$

$$\begin{aligned} D_0 &= D_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + \overline{D_0} \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_1 \cdot E_2 \cdot E_1 \cdot E_0 + \overline{D_0} \cdot E_2 \cdot E_1 \cdot E_0 = \\ &= \overline{\overline{D_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + \overline{D_0} \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + D_1 \cdot E_2 \cdot E_1 \cdot E_0 + \overline{D_0} \cdot E_2 \cdot E_1 \cdot E_0}} = \\ &= \overline{(D_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0}) \cdot (\overline{D_0} \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0}) \cdot (D_1 \cdot E_2 \cdot E_1 \cdot E_0) \cdot (\overline{D_0} \cdot E_2 \cdot E_1 \cdot E_0)} \end{aligned}$$



- d) Calcule la frecuencia máxima a la que puede operar el circuito resultante. El retardo de las puertas NAND es de 15 ns y el de los biestables es de 50 ns.

Solución:

El retardo máximo viene dado por el tiempo de respuesta del biestable más el tiempo necesario para propagar los nuevos valores a través de los dos niveles de puertas NAND, el tiempo mínimo de ciclo será:

$$t_{min} = 50 + 15 + 15 = 80 \text{ ns}$$

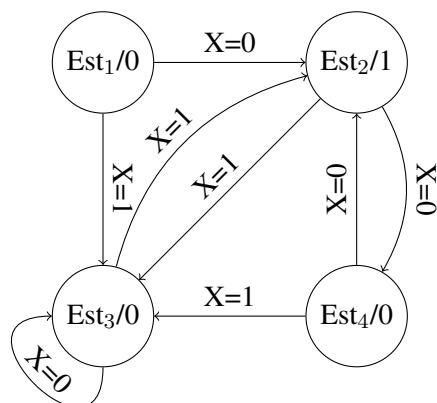
Por tanto, la frecuencia máxima será de 12.5 MHz.

- e) ¿Se podría implementar la función de transición mediante un PLA? Si se puede: ¿cuáles serían las dimensiones de dicho PLA?. Si no se puede: ¿por qué?

Solución:

Sí, se podría implementar con un PLA de $5 \times 6 \times 2$.

17. Diseñe el circuito secuencial síncrono con una entrada X y un bit de salida (S) que vendría especificado por el siguiente autómata finito determinista, utilizando flip-flops T disparados por flanco descendente:



Solución:

Dado que tenemos 4 estados, necesitaremos dos flip-flops para codificarlos (Q_0 y Q_1). Utilizaremos la siguiente codificación para los estados:

Estado	Q_0	Q_1
Est ₁	0	0
Est ₂	0	1
Est ₃	1	0
Est ₄	1	1

Por tanto, la tabla de verdad de la función de salida (S) será:

Q_0	Q_1	S
0	0	0
0	1	1
1	0	0
1	1	0

Y la tabla de la función de transición (Q_0^* y Q_1^*) y las entradas de los dos biestables (T_0 y T_1) será:

Q_0	Q_1	X	Q_0^*	Q_1^*	T_0	T_1
0	0	0	0	1	0	1
0	0	1	1	0	1	0
0	1	0	1	1	1	0
0	1	1	1	0	1	1
1	0	0	1	0	0	0
1	0	1	0	1	1	1
1	1	0	0	1	1	0
1	1	1	1	0	0	1

La expresión simplificada de la función de salida será:

$$S = \overline{Q_0}Q_1$$

La simplificación de la función T_0 es (otras soluciones también son posibles):

		Q_0Q_1			
		00	01	11	10
X	0	0	1	1	0
	1	1	1	0	1

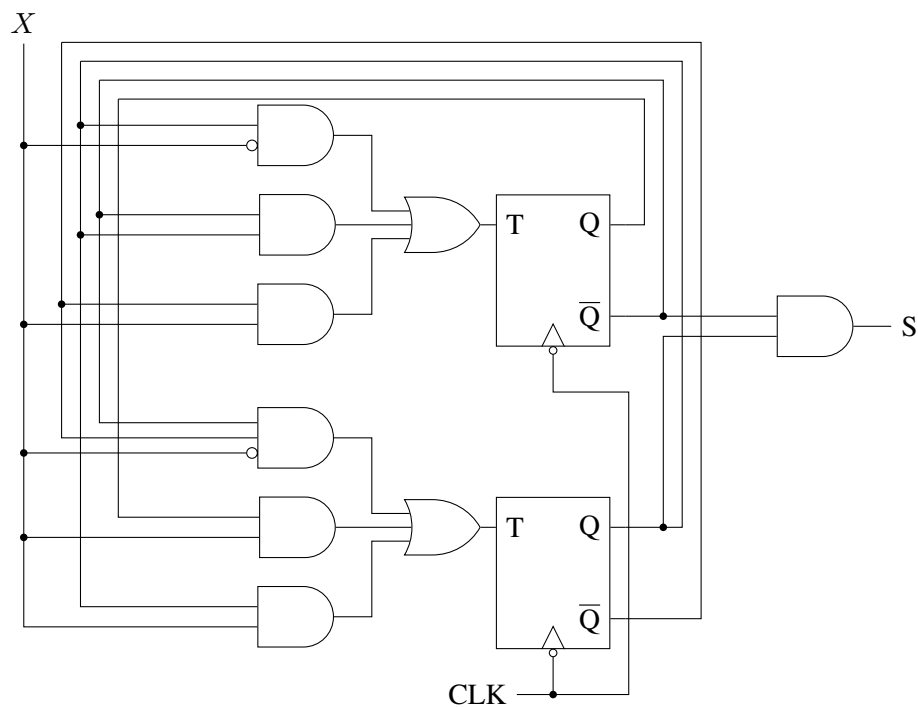
$$T_0 = Q_1\overline{X} + \overline{Q_0}Q_1 + \overline{Q_1}X$$

Y la de T_1 :

		Q_0Q_1			
		00	01	11	10
X	0	1	0	0	0
	1	0	1	1	1

$$T_1 = \overline{Q_0}\overline{Q_1}\overline{X} + Q_0X + Q_1X$$

Con lo que el circuito resultante usando puertas AND, OR y NOT será:



18. Implementar, utilizando biestables tipo D disparados por flanco descendente, un circuito capaz de detectar un retraso de un ciclo entre dos señales de un bit. El circuito tendrá dos entradas, A y B, y una salida, S, que se pondrá a 1 siempre que la entrada B se corresponda con la A del ciclo anterior. En el momento en que esta condición no se cumpla la salida quedará fijada a 0 hasta que una señal de Reset, activa en baja, vuelva a iniciar el proceso de comprobación. En el primer ciclo de reloj, donde la salida todavía no tiene sentido, supondremos que se cumple la condición. (Pista: Observad que solamente necesito saber si hasta el momento se está cumpliendo la condición y el valor anterior de A para poder implementar el circuito.)

Solución:

Tal y como se indica en la pista del ejercicio, el estado del sistema a implementar se reduce a conservar el último valor visto en la entrada A, así como si hasta el momento se cumple o no la condición. Por tanto podemos establecer los siguientes estados:

Estado 0 (q_0): Estado inicial en donde se supone que la entrada B es la entrada A retrasada un ciclo y no importa el valor de A.

Estado 1 (q_1): La entrada B es la entrada A retrasada un ciclo y el valor anterior de A fue 0.

Estado 2 (q_2): La entrada B es la entrada A retrasada un ciclo y el valor anterior de A fue 1.

Estado 3 (q_3): La entrada B no se corresponde con la entrada A retrasada un ciclo.

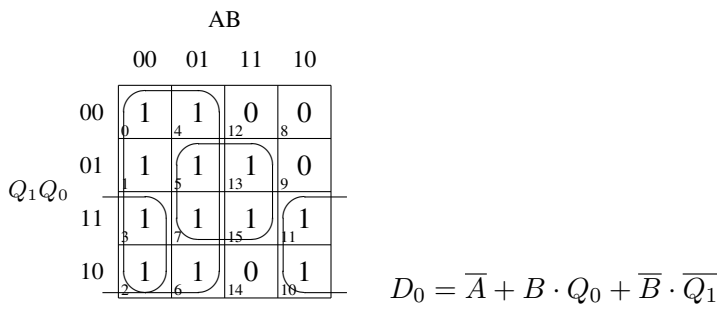
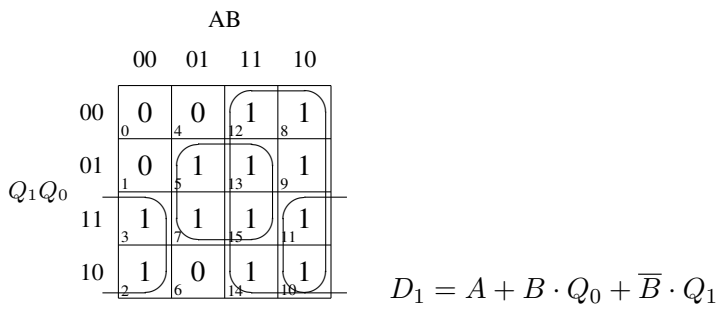
La tabla de transición de estados y de la salida será la que se indica a continuación (nos olvidaremos por ahora de la señal de reset):

Estado	$AB = 00$	$AB = 01$	$AB = 10$	$AB = 11$	SALIDA
q_0	q_1	q_1	q_2	q_2	1
q_1	q_1	q_3	q_2	q_3	1
q_2	q_3	q_1	q_3	q_2	1
q_3	q_3	q_3	q_3	q_3	0

Realizando la asignación de estados por defecto y teniendo en cuenta que se utilizan biestables D, con lo que la entrada de excitación del biestable coincide con la salida, tendremos la siguiente tabla:

Q_1Q_0	D_1D_0				SALIDA
	$AB = 00$	$AB = 01$	$AB = 10$	$AB = 11$	
00 (q_0)	01	01	10	10	1
01 (q_1)	01	11	10	11	1
10 (q_2)	11	01	11	10	1
11 (q_3)	11	11	11	11	0

Minimizando utilizando Mapas de Karnaugh, tenemos:



$$S = \overline{(Q_1 \cdot Q_0)} = \bar{Q}_1 + \bar{Q}_0$$

Para incluir la señal de reset (\bar{R}) bastará con asegurarnos que volvemos al estado 00 cuando dicha señal esté activa (sea 0). Para ello podemos simplemente multiplicar por la señal \bar{R} las entradas de ambos biestables. El circuito resultante será:

