ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

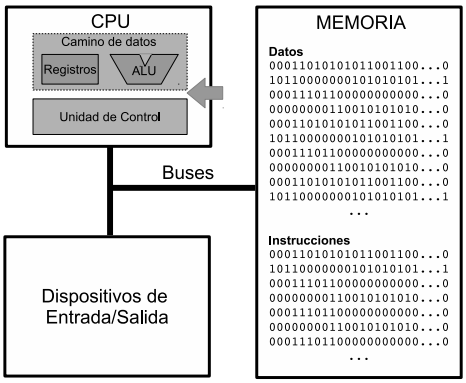
Departamento de Ingeniería y Tecnología de Computadores

Febrero 2021

ÍNDICE GENERAL

CAPÍTULO 2

SISTEMAS DIGITALES SECUENCIALES



2.1 INTRODUCCIÓN

Como se ha estudiado en asignaturas previas, los ordenadores son dispositivos digitales en los cuales la información se representa de forma discreta, a diferencia de los dispositivos analógicos que manejan formas continuas de información. El manejo de esta información digital se consigue mediante diferentes circuitos electrónicos basados, normalmente, en puertas lógicas básicas (AND, OR, NOT, NAND, NOR, XOR y XNOR) construidas a partir de transistores.

Los circuitos utilizados para construir sistemas digitales se pueden clasificar en *combinacionales* o *secuenciales*. En las asignturas previas a ésta, todos los circuitos que se han estudiado han sido circuitos combinacionales.

Los circuitos combinacionales se caracterizan porque la salida en cada instante depende única y exclusivamente de los valores de las entradas en ese mismo instante. En este sentido, se dice que los circuitos combinacionales *carecen de memoria*. Estos circuitos se pueden modelizar directamente mediante funciones booleanas.

Por su parte, la salida de un circuito secuencial no depende únicamente de los valores de las entradas en ese instante, sino de la historia anterior de dichos valores (es decir, de la *secuencia* de valores recibida). Para que estos circuitos se puedan construir es necesario que, de alguna manera, el circuito posea la capacidad de «recordar» su historia. Es decir, el circuito debe ser capaz de almacenar datos. Como veremos, estos circuitos no almacenan literalmente la historia de valores de sus entradas, sino sólo la información necesaria (que llamaremos *estado*) para producir el comportamiento deseado. Por tanto, podemos decir que el estado de un circuito secuencial es un resumen de la historia del circuito.

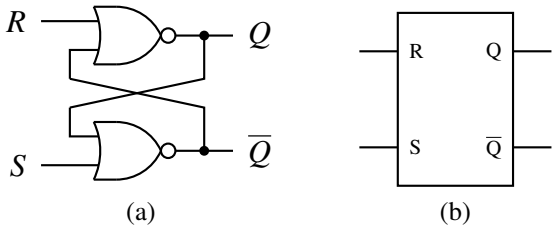


Figura 2.1: Implementación (a) y diagrama lógico (b) de un latch S-R.

Un ejemplo típico de circuito secuencial es la propia memoria de un computador. Si realizamos una operación de lectura de una memoria, para lo cual le proporcionamos los valores adecuados en sus entradas, ésta nos devolverá en su salida el valor almacenado en la posición de memoria solicitada. Pero ese valor depende de la histora previa del circuito, puesto que según qué valores hayan tomado las entradas de la memoria puede encontrarse almacenado cualquier valor en la posición solicitada.

En este tema estudiaremos cómo funcionan y cómo se diseñan los circuitos secuenciales, comenzando por los elementos de memoria más sencillos, los biestables. En el tema siguiente estudiaremos circuitos más complejos, como registros, bancos de registros y memorias. La última parte de este tema se centrará en explicar cómo realizar el diseño y síntesis de circuitos secuenciales y calcular su frecuencia máxima de funcionamiento.

2.2 BIESTABLES ASÍNCRONOS (LATCHES)

Los biestables asíncronos (también llamados cerrojos, o *latches* en inglés) son circuitos asíncronos básicos capaces de almacenar un bit. El valor de un *latch* en la señal de salida es función del valor de la señal de entrada y del valor de la señal de salida actual, que llamaremos «estado actual» (retroalimentación). Estos biestables asíncronos serán utilizados como bloques básicos en la construcción de biestables síncronos, como veremos en la siguiente sección.

2.2.1 Latch tipo S-R

Como hemos comentado, un latch es capaz de almacenar un bit, de manera que pueda ser recordado posteriormente una vez que dejen de estar activas las entradas que provocaron su escritura. La figura 2.1 muestra un latch construido a partir de dos puertas NOR cuyas salidas realimentan a las entradas. Este circuito se conoce como latch S-R (*Set-Reset*).

Las salidas Q y $\overline{Q}$ 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 $\overline{Q}$. Por ejemplo, si la salida Q es 1, entonces el inversor inferior produce una salida 0, que es $\overline{Q}$, y viceversa.

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

La *tabla de excitación* 2.1 resume el funcionamiento lógico del latch S-R, donde Q^* indica el valor del estado siguiente, frente al estado actual Q.

La tabla de excitación muestra cómo cambia el estado para cada combinación de estado actual y entradas de excitación. Las columnas S y R son las entradas aplicadas al

Entradas de excitación		**Estado actual**	**Estado siguiente**	**Acción**
S	R	Q	Q^*	
0	0	0	0	Retener estado
0	0	1	1	
0	1	0	0	Poner a cero
0	1	1	0	
1	0	0	1	Poner a uno
1	0	1	1	
1	1	0	X	No permitido
1	1	1	X	

Tabla 2.1: Tabla de excitación del latch S-R.

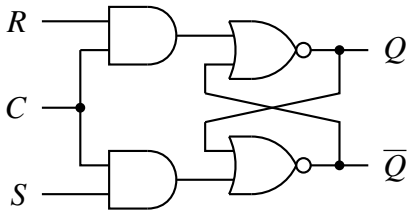


Figura 2.2: Latch S-R con señal de control.

latch. La columna rotulada Q es el estado del latch S-R antes de aplicar una combinación de entradas a S y R. La columna rotulada como Q^* es el estado del latch después de aplicar las entradas al latch. Por tanto, la columna Q es el estado actual y la columna Q^* es el estado siguiente. La información de esta tabla puede expresarse también mediante una ecuación lógica, denominada *ecuación característica* o *función de transición* del latch S-R:

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

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 latch hará caso de sus entradas. Para ello basta emplear dos puertas AND, como se muestra en la figura 2.2. Sólo cuando la señal de control C esté a 1, será posible cambiar el estado del latch mediante S y R.

2.2.2 Latch tipo D

Pasamos ahora a modificar el latch S-R para hacer más sencillo su uso. La figura 2.3 muestra el diagrama de un latch D. Un latch 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 del dato que se va a almacenar, denominado D, y una señal de control, llamada C, que determina cuándo el latch debe almacenar el valor de la entrada D. Cuando C vale 1 el latch está abierto y el valor de entrada D se almacena en el latch. Cuando C vale 0, el latch está cerrado y el valor de la entrada D se ignora. El valor de salida Q es siempre el valor que fue almacenado la última vez que el latch estuvo abierto.

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.

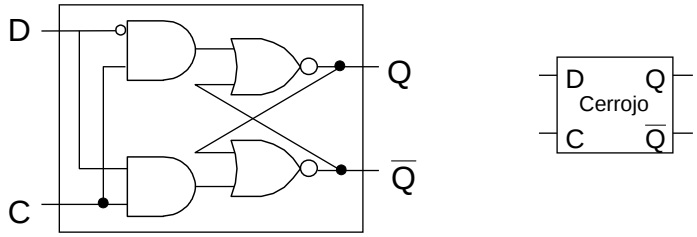


Figura 2.3: Latch D, controlado por la señal C.

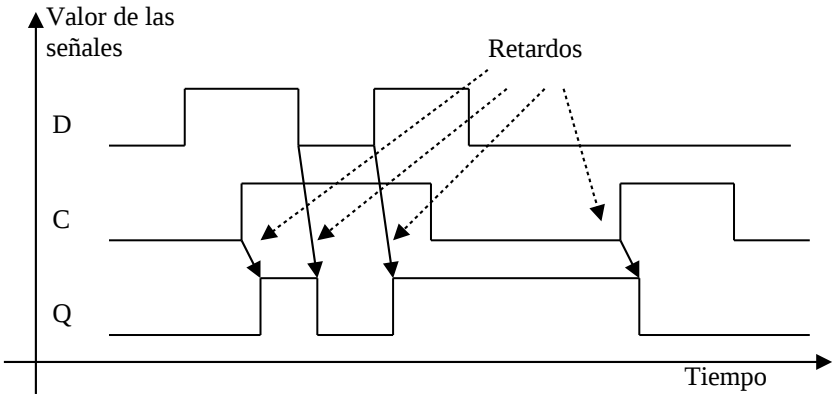


Figura 2.4: Operación de un latch D suponiendo que la salida está inicialmente a 0.

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 cronograma de la figura 2.4, se muestra un ejemplo del funcionamiento de un latch 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 latch 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.

Entrada de control	**Entrada de excitación**	**Estado siguiente**	**Acción**
C	D	Q^*	
0	X	Q	Retener estado
1	0	0	Almacenar 0
1	1	1	Almacenar 1

Tabla 2.2: Tabla de excitación del latch D.

En la tabla 2.2 se observa la tabla de excitación del latch D en forma resumida que da lugar a la ecuación característica:

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

2.3 BIESTABLES SÍNCRONOS (FLIP-FLOPS)

Los biestables síncronos (*flip-flops*, en inglés) son circuitos secuenciales síncronos capaces de almacenar un bit. Estos circuitos se pueden construir a partir de latches, tal y como explicamos en esta sección.

Aunque tanto latches como flip-flops permiten almacenar un bit de información, la diferencia fundamental radica en que el flip-flop es síncrono, mientras que el latch es asíncrono.

En efecto, a diferencia del caso de los latches, 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 de tiempo como ocurría con la señal C de los latches. Es, por tanto, un sistema secuencial síncrono, que depende de una señal de *reloj*.

Una señal periódica, denominada *reloj*, será la encargada de determinar en qué momento concreto el circuito será sensible a su entrada y se podrá escribir 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.

Comenzaremos estudiando el flip-flop tipo D ya que es el más frecuentemente utilizado.

2.3.1 Flip-flop tipo D

El flip-flop tipo D es un circuito secuencial que se construye mediante la conexión en serie de dos latches tipo D. La figura 2.5 (superior, izquierda) muestra el esquema de un flip-flop D maestro-esclavo disparado por flanco descendente. La figura 2.5 (superior, derecha) muestra su representación como bloque lógico, donde el hecho de que la activación se produzca por flanco se expresa colocando un triángulo en la entrada C del bloque lógico. Si, además, incluimos un pequeño círculo, entonces estamos indicando que el flanco activo es el descendente. En caso de no llevar el círculo la activación será por flanco ascendente.

El flip-flop tipo D funciona de la siguiente manera. El primer latch, 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 latch se cierra, pero el segundo latch, llamado *esclavo*, se abre y obtiene su entrada de la salida del primer latch 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 latch maestro, en lugar de al esclavo.

En la figura 2.5 (inferior) se ha añadido, además, una señal adicional de permiso (o habilitación) 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

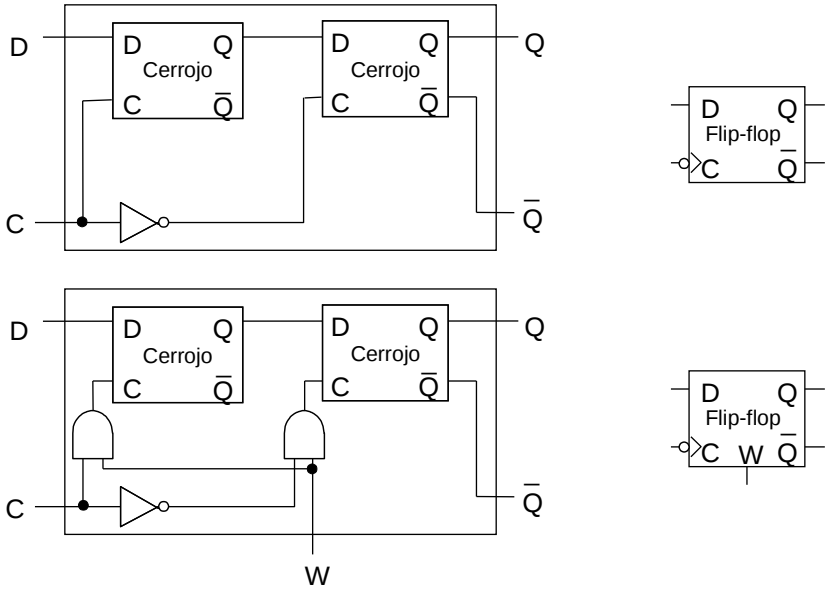


Figura 2.5: Flip-flop tipo D maestro-esclavo, y símbolo de bloque lógico correspondiente: (Superior) Sin señal adicional de escritura. (Inferior) Después de añadirle dicha señal.

(cuando la señal W valga 1). Esta señal, como veremos, será de gran importancia cuando construyamos el camino de datos de la CPU en un tema posterior.

La tabla de excitación de un flip-flop D (sin señal de permiso de escritura) es similar a la de un latch D, la diferencia radica en la temporización. Mientras que en un latch 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 2.3 para el caso de un flip-flop tipo 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**	**Acción**
C	D	Q^*	
$1 \rightarrow 0$	0	0	Almacenar 0
$1 \rightarrow 0$	1	1	Almacenar 1

Tabla 2.3: Tabla de excitación del flip-flop tipo D disparado por flanco descendente.

De modo análogo se puede razonar para el caso de tener la entrada de habilitación W. En ese caso, la tabla quedaría como se muestra en 2.4 y la ecuación característica sería: $Q^* = D \cdot W + Q \cdot \overline{W}$.

Finalmente, mostramos en la figura 2.6 un cronograma que ejemplifica el comportamiento de un flip-flop tipo D activo en el flanco ascendente. En este cronograma suponemos que los retardos son despreciables.

Los flip-flops tipo D son los más importantes, puesto que constituyen el método más cómodo para el almacenamiento de datos en los computadores. En el tema siguiente veremos cómo utilizarlos para construir, por ejemplo, registros y bancos de registros.

Entrada de reloj C	**Permiso de escritura** W	**Entrada de excitación** D	**Estado siguiente** Q^*	**Acción**
$1 \rightarrow 0$	0	X	Q	Retener estado
$1 \rightarrow 0$	1	0	0	Almacenar 0
$1 \rightarrow 0$	1	1	1	Almacenar 1

Tabla 2.4: Tabla de excitación del flip-flop tipo D disparado por flanco descendente con señal de permiso de escritura.

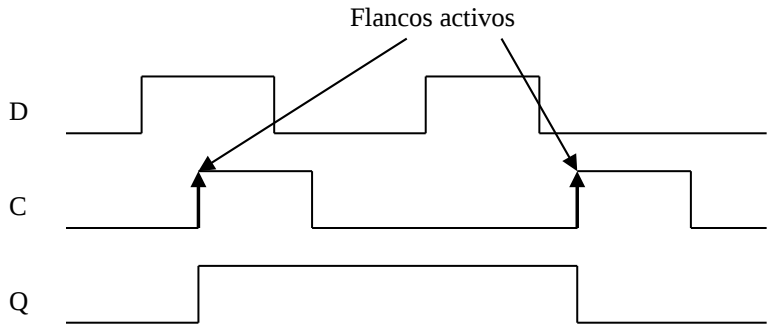


Figura 2.6: Cronograma de operación de un flip-flop tipo D disparado por flanco de subida.

Sin embargo, no son la única alternativa. Pueden construirse también otros tipos de flip-flops, con comportamientos ligeramente distintos, que pueden resultar útiles en algunos casos. Vamos a ver algunos en las siguientes secciones.

2.3.2 Flip-flop tipo S-R

El flip-flop tipo S-R puede construirse a partir de dos latches S-R con una señal de control, como se muestra en la figura 2.7. Las señales de activación de los dos latches son controladas por versiones complementarias de una señal de reloj. Cuando la señal de reloj C es cero, el latch maestro está abierto y el esclavo cerrado, lo que hace que el latch maestro procese sus entradas S y R. Cuando la señal de reloj es uno, ambos latches intercambian sus papeles de forma que el maestro está cerrado y el esclavo abierto. Esto hace que el latch esclavo transmita la salida del latch maestro a la salida Q del flip-flop, mientras que el latch maestro permanece cerrado ignorando cualquier cambio posterior en sus entradas. El flip-flop tipo S-R de la figura 2.7 se activa por flanco ascendente y tiene como ecuación característica: $Q^* = S + \overline{R} \cdot Q$, obtenida a partir de la tabla de excitación 2.5.

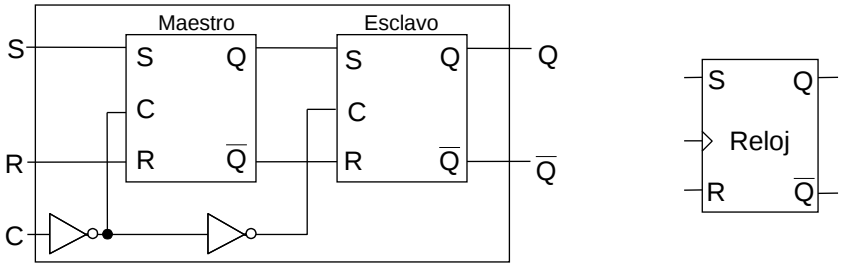


Figura 2.7: Flip-flop S-R (izquierda) y símbolo lógico del flip-flop S-R (derecha).

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

Tabla 2.5: Tabla de excitación del flip-flop tipo S-R disparado por flanco ascendente.

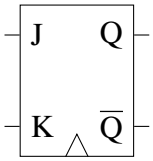


Figura 2.8: Símbolo lógico del flip-flop tipo J-K.

2.3.3 Flip-flop tipo J-K

Otro tipo de flip-flop es el J-K. Puede considerarse como una extensión de un flip-flop tipo 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 $Q^* = \overline{K} \cdot Q + J \cdot \overline{Q}$, obtenida a partir de la tabla de excitación 2.6. En la figura 2.8 se puede ver el símbolo lógico que representa a este flip-flop y su implementación.

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

Tabla 2.6: Tabla de excitación del flip-flop J-K disparado por flanco descendente.

2.3.4 Flip-flop tipo T

Por último, el flip-flop tipo 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 invertir su estado cada vez que la señal de entrada T valga 1 durante el flanco de activación. La tabla 2.7 resume el comportamiento de un flip-flop T activado por flanco descendente cuya ecuación característica es $Q^* = \overline{Q} \cdot T + Q \cdot \overline{T}$. El símbolo lógico que representa a este flip-flop se puede ver en la figura 2.9.

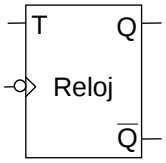


Figura 2.9: Símbolo lógico del flip-flop T.

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

Tabla 2.7: Tabla de excitación del flip-flop T disparado por flanco descendente.

2.4 DISEÑO/SÍNTESIS DE UN CIRCUITO SECUENCIAL

En este apartado estudiaremos cómo diseñar 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 biestables. Estos serán los encargados de codificar el estado en el que se encuentra el circuito en cada momento.

2.4.1 Estructura general de un circuito secuencial

La figura 2.10 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, biestables, 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 por 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.

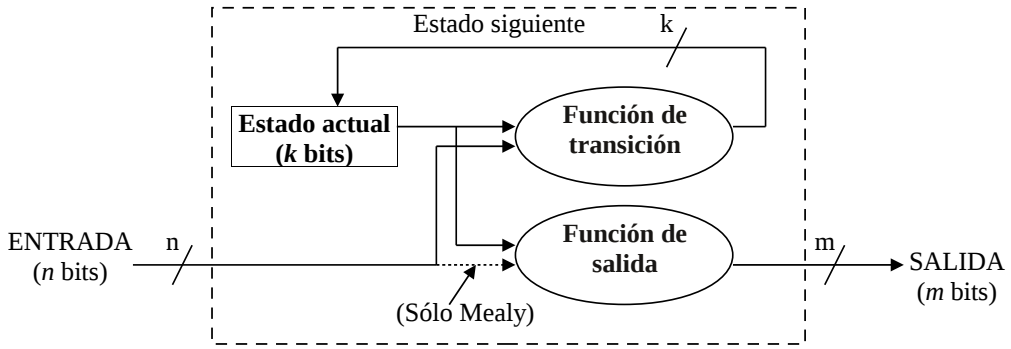


Figura 2.10: Esquema lógico general de un circuito secuencial. También se muestra la diferencia esencial entre los circuitos de Moore y los de Mealy.

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 2.10 se resume la diferencia entre ambos tipos de circuitos: como puede observarse, sólo en los circuitos de tipo Mealy la función de salida depende de las entradas. Ambos tipos de circuito son igual de potentes, ya que para cualquier autómata de Mealy existe un autómata de Moore equivalente, y viceversa.

Otra clasificación de los circuitos secuenciales dependerá del tipo de biestables con los que se implementen. Si se implementan con latches (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** (estos serán los circuitos secuenciales en los que nos centraremos).

2.4.2 Fases en el diseño/síntesis de un circuito secuencial

Nos centraremos en el diseño de circuitos secuenciales síncronos utilizando únicamente autómatas de Moore. 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 biestables necesarios para codificarlos. En concreto, si el autómata tiene M estados, necesitaremos $n = \lceil log_2 M \rceil$ biestables. 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 en esta asignatura por falta de tiempo.

5. **Determinación de las funciones de entrada de los biestables**: El estado del autómata se almacena en varios biestables según la codificación decidida en el paso previo. Dependiendo del tipo de los biestables utilizados, cada uno de ellos tendrá una o dos entradas para permitir controlarlo. En este paso se determinan las funciones lógicas combinacionales que controlarán dichas entradas. Estas funciones, en conjunto, implementan la función de transición.

6. **Minimización de los circuitos lógicos combinacionales**: una vez determinadas las funciones de entrada de los biestables y la función de salida (que se determinó al especificar el autómata), podemos minimizarlas (ya que son circuitos combinacionales normales y corrientes). Con ello obtenemos las expresiones con la que implementaremos dichas partes del circuito.

7. **Implementación del circuito**: finalmente, sólo nos queda utilizar los n biestables (que estarán todos conectados a una misma señal de reloj) y las expresiones obtenidas en el último paso para dibujar el circuito final, siguiendo la estructura de la figura 2.10.

2.4.3 Ejemplo de diseño/síntesis de un circuito secuencial con flip-flops tipo D

En este apartado diseñaremos un circuito de ejemplo siguiendo el procedimiento indicado en el apartado anterior. El circuito que diseñaremos será un detector de una secuencia particular de bits, que se conectaría en una línea de comunicaciones serie de forma que activaría una señal cada vez que detecte la secuencia (podría usarse algo parecido, por ejemplo, para detectar un determinado carácter con un significado especial en la conexión).

Paso 1: Especificación verbal

«*Diseñar un circuito secuencial con una única entrada X y una única salida S. El circuito debe de detectar la aparición de tres unos consecutivos en su línea de entrada X (es decir, debe detectar cuando X vale 1 durante tres ciclos seguidos de reloj). 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 el diseño utilizando flip-flops tipo D disparados por flanco ascendente.»

Paso 2: Especificación del autómata

En este caso el autómata de Moore que recoge el funcionamiento deseado para el circuito es el mostrado en la figura 2.11. Se utiliza un círculo para cada estado Est_i, se especifica la salida asociada a cada estado por tratarse de un autómata de Moore (función de salida), y se especifican mediante flechas las transiciones entre estados asociadas a cada entrada (función de transición). En las tablas 2.8 se recoge la misma información que en el diagrama pero mostrada en forma tabular.

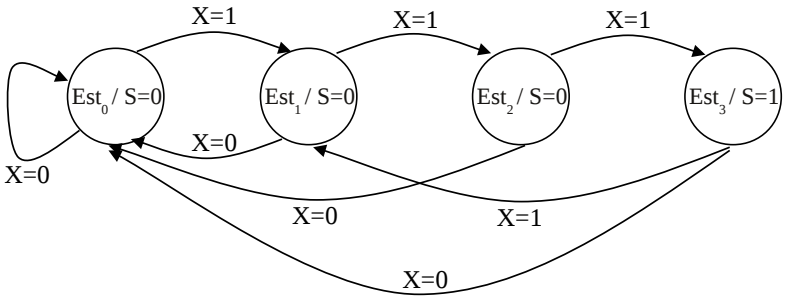


Figura 2.11: Autómata finito determinista del circuito detector de secuencias de tres unos.

Estado actual	Entrada (X)	Estado siguiente
Est_0	0	Est_0
Est_0	1	Est_1
Est_1	0	Est_0
Est_1	1	Est_2
Est_2	0	Est_0
Est_2	1	Est_3
Est_3	0	Est_0
Est_3	1	Est_1

Estado actual	Salida (S)
Est_0	0
Est_1	0
Est_2	0
Est_3	1

Tabla 2.8: Tabla de transición entre estados y función de salida para el ejemplo explicado.

Paso 3: Minimización del autómata obtenido

El estudio de las técnicas de minimización de autómatas es objeto de otras asignaturas. Además, en este caso el autómata ya es mínimo, por lo que no sería necesario simplificarlo.

Paso 4: Codificación de estados

Calculamos el número de biestables necesarios y asignamos una combinación de bits a cada estado. En este caso, al haber 4 estados, es suficiente con 2 biestables cuyas

salidas serán Q_0 y Q_1 (ya que $4 \leq 2^2$). Utilizaremos la codificación en binario natural ($\text{Est}_0 = 00$, $\text{Est}_1 = 01$, $\text{Est}_2 = 10$, $\text{Est}_3 = 11$).

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 2.9. Obsérvese que diferenciamos el estado futuro (Q_i^*) del actual (Q_i) utilizando un asterisco.

Q_1	Q_0	X	Q_1^*	Q_0^*
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	1

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

Tabla 2.9: Tabla de transición entre estados y función de salida con los estados ya codificados para el ejemplo explicado.

Paso 5: Determinación de las funciones de entrada de los biestables

Una vez que se ha decidido cómo codificar los estados, debemos de determinar cuáles deben ser las entradas de los biestables en cada momento para conseguir que se produzcan los cambios de estado que deseamos. Para ello, es necesario recordar cómo funciona el tipo de biestable que estemos utilizando.

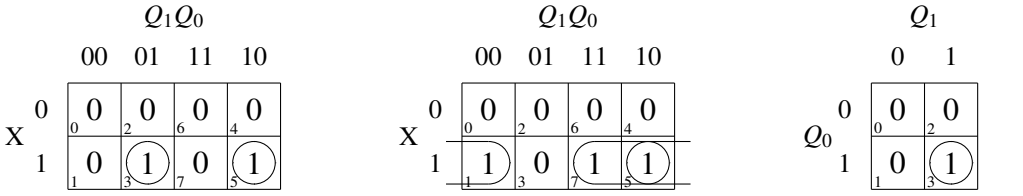
En el caso de utilizar flip-flops tipo D, este paso es trivial, ya que estos biestables almacenan directamente el valor que reciben a su entrada (cuando se produce el flanco de reloj). El resultado para nuestro ejemplo, si llamamos D_1 y D_0 a las entradas de los flip-flops Q_1 y Q_0 respectivamente, se muestra en la tabla 2.10. Obsérvese que no es necesario realizar una nueva tabla, sino que se pueden añadir nuevas columnas a la tabla 2.9.

Q_1	Q_0	X	Q_1^*	Q_0^*	D_1	D_0
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	0	0	0
0	1	1	1	0	1	0
1	0	0	0	0	0	0
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	1	0	1	0	1

Tabla 2.10: Cálculo de los valores de las entradas de excitación de los dos flip-flops tipo D.

Paso 6: Minimización de los circuitos lógicos combinacionales

La figura 2.12 muestra la simplificación de los circuitos que implementarán la función de transición (D_1 y D_0) y la de salida (S).



(a) Simplificación de D_1 (b) Simplificación de D_0 (c) Simplificación de S

Figura 2.12: Simplificación de las entradas de los flip-flops tipo D y de la función de salida.

Con flip-flops tipo D, el resultado de la simplificación sería:

$$D_1 = \overline{Q_1} \cdot Q_0 \cdot X + Q_1 \cdot \overline{Q_0} \cdot X$$
$$D_0 = Q_1 \cdot X + \overline{Q_0} \cdot X$$
$$S = Q_1 \cdot Q_0$$

Paso 7: Implementación del circuito

Ya sólo nos queda dibujar el circuito con biestables y puertas lógicas. La figura 2.13 muestra cómo queda dicha implementación utilizando flip-flops tipo D activos en flanco ascendente.

Finalmente, en la figura 2.14 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 cambien (si cambian) en el siguiente flanco activo. Por simplicidad, en el cronograma se supone que ningún componente del circuito introduce retardo.

2.4.4 Ejemplo de diseño/síntesis de un circuito secuencial con flip-flops tipo J-K

En el caso de utilizar un tipo de flip-flop diferente al tipo D, la determinación de las funciones de entrada de los biestables (paso 5) será ligeramente más complicada, y el resultado de los pasos siguientes también cambiará.

Paso 5: Determinación de las funciones de entrada de los biestables

El funcionamiento de cada biestable viene dado por su tabla de excitación. Esas tablas nos dicen qué entrada hay que proporcionarle al biestable según el estado en que se encuentre para que se almacene en él el estado que deseemos. Para los flip-flops tipo J-K podemos consultar la tabla 2.11. Hay que tener en cuenta que, a diferencia de los flip-flops tipo T o tipo D, cada flip-flop tipo J-K tendrá dos entradas. Es fácil

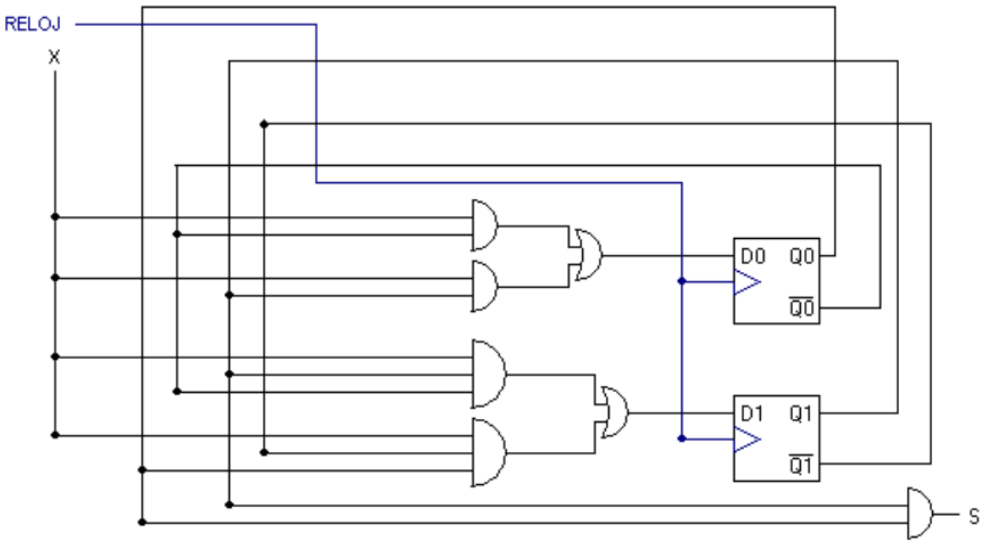


Figura 2.13: Implementación del circuito utilizando flip-flops tipo D.

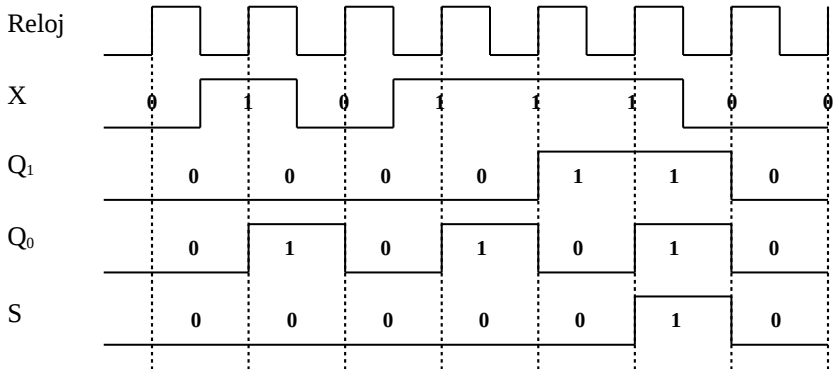


Figura 2.14: Cronograma de ejemplo mostrando el funcionamiento dinámico del circuito.

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

Tabla 2.11: Determinación del valor de la entrada de un flip-flop tipo J-K en función del estado actual y del estado deseado.

construir mentalmente estas tablas si se entiende el funcionamiento del tipo de biestable que usemos.

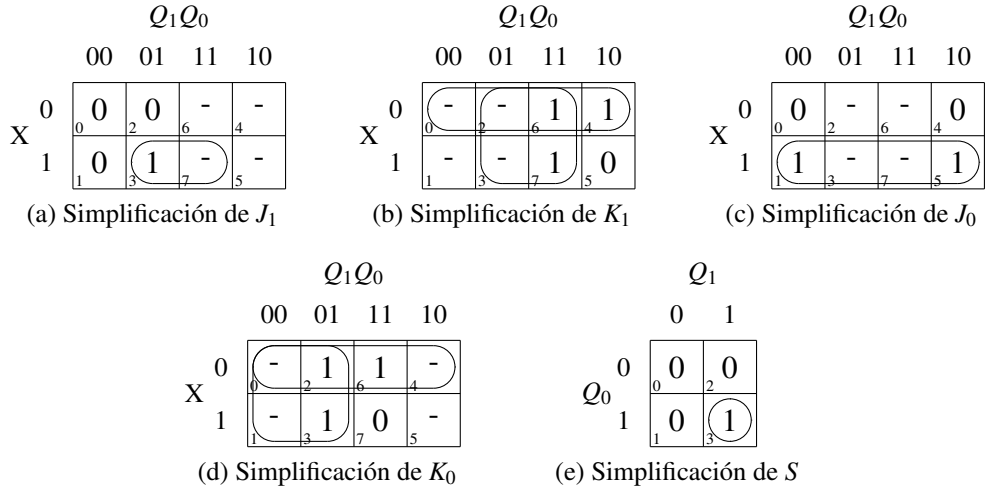
Con esta información, podemos construir una nueva tabla a partir de la tabla 2.9, añadiéndole una columna por cada entrada de los flip-flops (J_1, K_1, J_0 y K_0) en las que mostraremos los valores de las entradas de excitación de los flip-flops a usar en la síntesis del circuito.

Para calcular el valor de cada una de las nuevas celdas, es suficiente con mirar el estado actual y el estado siguiente de la misma fila. Por ejemplo, si el estado actual es $Q_1Q_0 = 10$ y la entrada $X = 0$, el estado siguiente debe ser $Q_1^*Q_0^* = 00$. Eso quiere decir que el flip-flop Q_1 debe de pasar de almacenar un 1 a un 0, lo cual lo podemos conseguir de dos maneras: asignando $J_1 = 0$ y $K_1 = 1$, o bien $J_1 = 1$ y $K_1 = 1$. O lo que es lo mismo, como se puede ver en la tabla 2.11, no nos importa el valor de J_1 y el valor de K_1 debe de ser 1. Con un razonamiento análogo se determina que, para esta fila, el valor de J_0 debe de ser 0 y el de K_0 es irrelevante.

El resultado final se muestra en 2.12.

Q_1	Q_0	X	Q_1^*	Q_0^*	J_1	K_1	J_0	K_0
0	0	0	0	0	0	–	0	–
0	0	1	0	1	0	–	1	–
0	1	0	0	0	0	–	–	1
0	1	1	1	0	1	–	–	1
1	0	0	0	0	–	1	0	–
1	0	1	1	1	–	0	1	–
1	1	0	0	0	–	1	–	1
1	1	1	0	1	–	1	–	0

Tabla 2.12: Cálculo de los valores de las entradas de excitación de los dos flip-flops tipo J-K.



(a) Simplificación de J_1 (b) Simplificación de K_1 (c) Simplificación de J_0

(d) Simplificación de K_0 (e) Simplificación de S

Figura 2.15: Simplificación de las entradas de los flip-flops J-K y de la función de salida.

Paso 6: Minimización de los circuitos lógicos combinacionales

La simplificación para el caso de utilizar flip-flops tipo J-K se muestra en la figura 2.15. El resultado sería:

$$
\begin{aligned}
J_1 &= Q_0 \cdot X \\
K_1 &= \overline{X} + Q_0 \\
J_0 &= X \\
K_0 &= \overline{X} + \overline{Q_1} \\
S &= Q_1 \cdot Q_0
\end{aligned}
$$

Paso 7: Implementación del circuito

El circuito resultante se muestra en la figura 2.16.

2.5 CÁLCULO DE LA FRECUENCIA MÁXIMA DE OPERACIÓN DE UN CIRCUITO SECUENCIAL

Por último, es importante reflexionar sobre la frecuencia máxima de operación que puede tolerar un circuito secuencial síncrono. En principio, esta frecuencia máxima dependerá directamente del retardo introducido por sus componentes. La clave está en que desde un cambio de señal de reloj hasta el siguiente, las señales de entrada de los flip-flops deben ser estables, y por tanto el tiempo de ciclo debe ser lo suficientemente grande para que el valor de los estados se propague de nuevo a la entrada de los flip-flops.

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, y observando el diagrama del circuito anterior usando flip-flops tipo D, mostrado en la figura 2.13 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

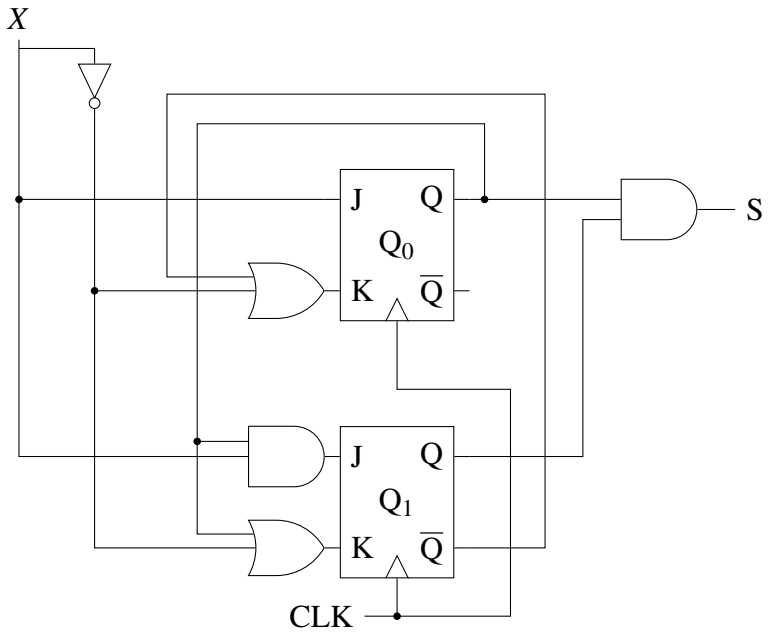


Figura 2.16: Implementación del circuito utilizando flip-flops tipo J-K.

flip-flops, un nivel de puertas AND y otro de puertas OR antes de que las entradas de los flip-flops 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 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.

Por tanto, 40 ns es el tiempo mínimo que debemos dejar transcurrir desde que comienza el ciclo hasta que se produzca 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: $\frac{1}{40\text{ ns}} = \frac{1}{4010^{-9}\text{ s}} = 0.02510^{9}$ Hz $= 25$ MHz.

De manera similar, para la figura 2.16 tendríamos que atravesar una puerta lógica AND o una OR y un flip-flop. Las puertas lógicas conectadas a los flip-flop operan en paralelo. En total, tendremos un retado de 20 ns (flip-flop) + 10 ns (AND u OR) = 30 ns, y por tanto una frecuencia máxima de operación de $\frac{1}{30\text{ ns}} = 33$ MHz.

2.6 COMPONENTES SECUENCIALES DE UN PROCESADOR

2.6.1 Registro

Hemos visto que los flip-flops tipo D sirven para almacenar un bit cuyo valor se actualizará al valor de la entrada en D en el flanco de activación. 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 de forma que compartan las señales de reloj y de permiso de escritura. La figura 2.17 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. Por ejemplo, en un registro de 32 bits, tanto D_{in} como D_{out} serían señales de 32 bits de ancho.

En el procesador que implementaremos usaremos registros construidos a partir de flip-flops tipo D, como los de la figura 2.17. Estos registros tendrán diferentes tamaños

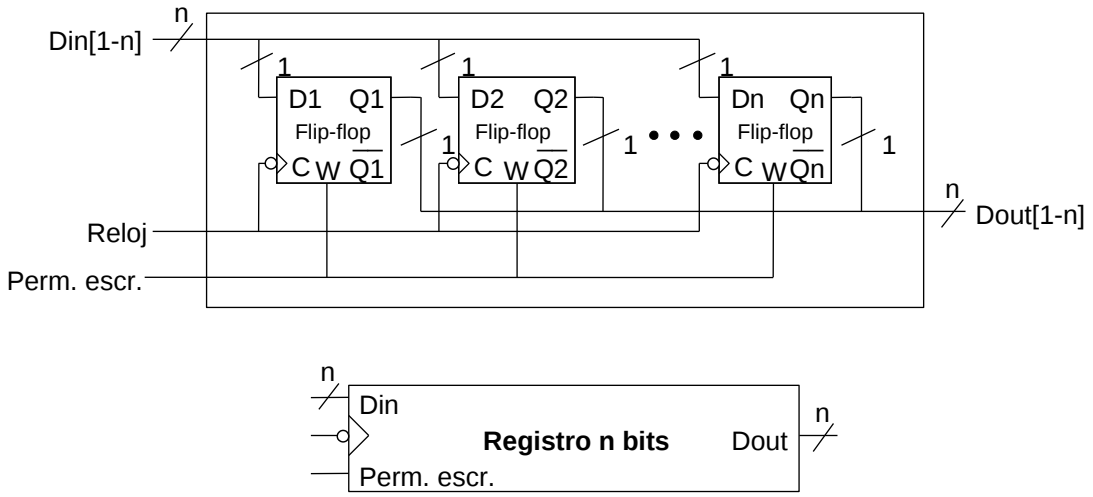


Figura 2.17: Implementación y diagrama lógico de un registro de *n* bits usando flip-flops tipo D.

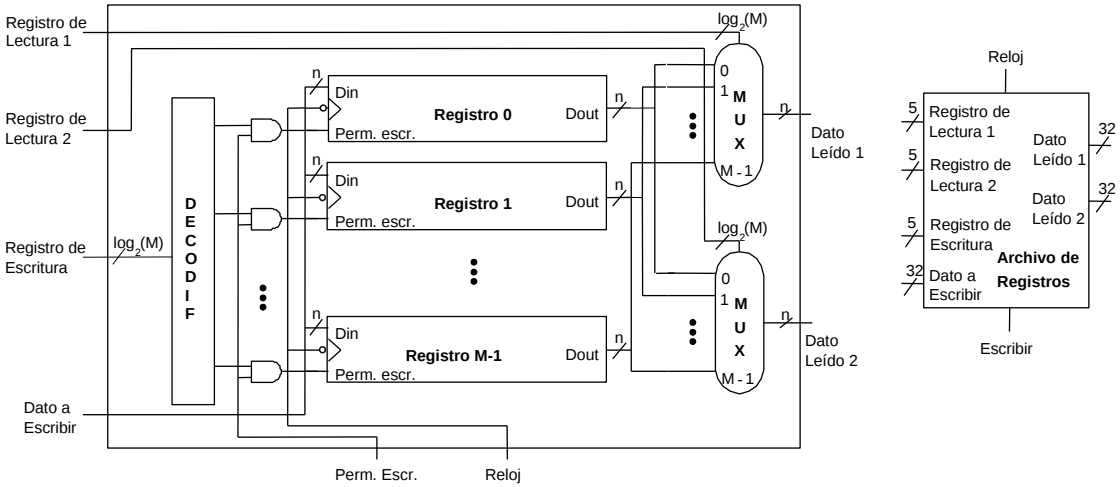


Figura 2.18: Banco de registros con 2 puertos de lectura (izquierda) y 1 puerto de escritura, y banco de registros que usaremos en el camino de datos del MIPS (derecha).

según sea necesario y se utilizarán tanto para implementar los registros del ISA (es decir, los registros visibles al programador) como varios registros auxiliares a los que le programador de bajo nivel no tendrá acceso (registros usados para pasar información entre las distintas etapas de ejecución de las instrucciones).

2.6.2 Banco de registros

Muchos de los registros del procesador se agrupan en una estructura denominada banco de registros. Un 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 *puertos de lectura*. La implementación se realiza mediante una serie de registros construidos a partir de flip-flops tipo D y usando multiplexores y decodificadores para seleccionar los/el 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 el número del registro (podemos tratar este número como una dirección), y seguidamente

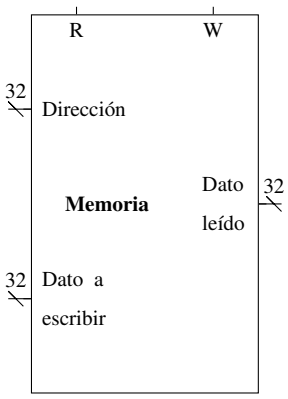


Figura 2.19: Bloque lógico usado para representar la memoria.

en el puerto de salida correspondiente aparecerá el dato contenido en ese registro. Para escribir en un registro especificado por su número (o dirección) necesitaremos tres entradas: el número del registro, el dato a escribir y una señal de control de escritura que indique que se desea realizar la escritura. Esta última señal de control junto con el reloj controlan el momento preciso en el que se escribe el dato.

En la figura 2.18 puede verse el esquema genérico de un banco de M registros de n bits con dos puertos de lectura y uno de escritura. A la derecha aparece el bloque lógico de un banco 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 procesador en un tema posterior.

2.6.3 Memoria

Los registros y bancos de registros que acabamos de ver se pueden emplear 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 ambos tipos de memoria, pero antes de nada describiremos el bloque lógico correspondiente a la memoria que emplearemos en la construcción del procesador en el tema siguiente (sin entrar en detalles de implementación concretos).

En la figura 2.19 se muestra el bloque lógico que usaremos para representar la memoria del procesador. Supondremos que esta memoria es capaz de almacenar palabras de 4 bytes (longitud de palabra de 32 bits) empleando direcciones de memoria también de 32 bits. En un ciclo dado la memoria puede realizar una operación de lectura o escritura de una palabra. A través de las señales «R» y «W» podemos ordenar a la unidad de memoria la realización de una operación de lectura y escritura respectivamente. Estas señales nunca deben ser activadas simultáneamente, ya que en este caso el comportamiento de la unidad no está definido.

Para realizar una lectura, la dirección que se desea leer se pondrá en el puerto «Dirección» y se activará la señal de control «R». Al final del ciclo, el contenido de la palabra almacenada en la dirección dada aparecerá en el puerto «Dato leído».

Para realizar una escritura, la dirección que se desea leer se pondrá en el puerto «Dirección», el contenido que se desea almacenar se pondrá en el puerto «Dato a escribir» y se activará la señal de control «W» al principio del ciclo. Al final del ciclo, el contenido se habrá almacenado en la memoria.

Memorias 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 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 2.20 muestra el diagrama lógico de una SRAM de 32768 posiciones de 8 bits cada una ($32768 \times 8 = 32\ KiB$), y por tanto necesita 15 líneas para ser direccionada ($2^{15} = 32768$). Cada elemento tendrá 8 bits, por lo tanto, tendemos 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.

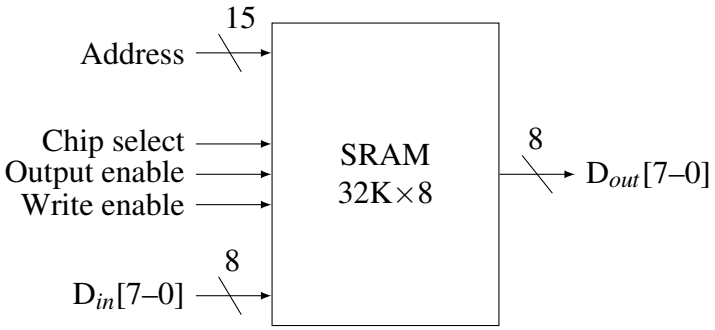


Figura 2.20: SRAM de 32K×8: 15 líneas de dirección, 8 líneas de entrada de datos, tres líneas de control y 8 líneas de salida de datos.

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 banco 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 2.21 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).

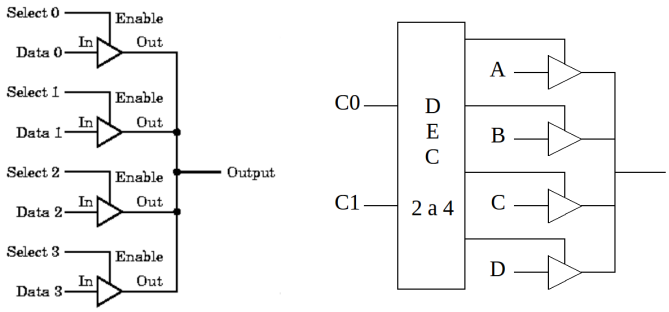


Figura 2.21: Cuatro buffers triestado para formar un multiplexor 4 a 1.

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 2.22 muestra una pequeña SRAM de 4×2, donde cada latch (obsérvese que en este caso se usan latches 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 decodificador a la entrada, que para memorias con muchas posiciones direccionables sería excesivamente grande, y por tanto, costoso de implementar. Para eliminar este inconveniente, las grandes memorias se organizan como arrays rectangulares (bidimensionales) y utilizan un proceso de decodificación de dos pasos. La figura 2.23 muestra como podría conseguirse tal disposición construyendo una memoria SRAM con 4M posiciones de 8 bits cada una (memoria $4M\times8$) en base a 8 bloques $4K\times1024$. Observar que las direcciones para esta memoria tendrían una longitud de 22 bits. El decodificador 12 a 4096 genera la dirección para los 8 bloques de $4K\times1024$ a partir de los 12 bits más significativos de la dirección (la fila del array rectangular a leer). En total se leen 8192 bits (1024 por cada uno de los bloques de $4K\times1024$). Después a través de un conjunto de 8 multiplexores 1024 a 1 se selecciona un bit de cada uno de los 8 bloques (8 columnas del array rectangular, 1 por bloque). Para ello se emplean los 10 bits menos significativo de la dirección. Este diseño es más eficiente que una decodificación de un solo paso que necesitaría un decodificador de ¡22 a 4M!.

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

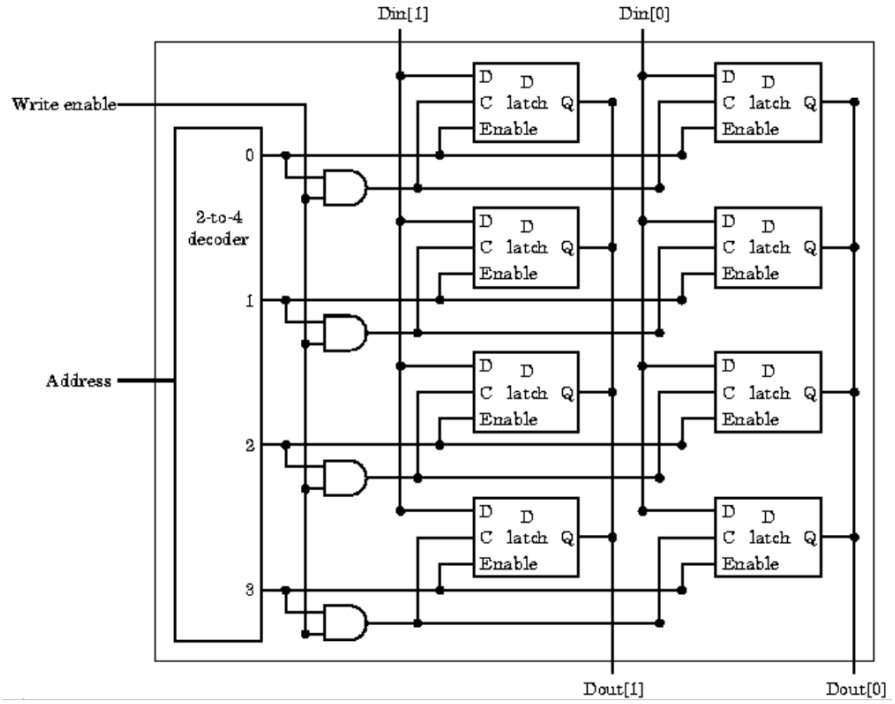


Figura 2.22: SRAM de 4×2. Por simplicidad se omiten las entradas de habilitación de salida y selección de chip.

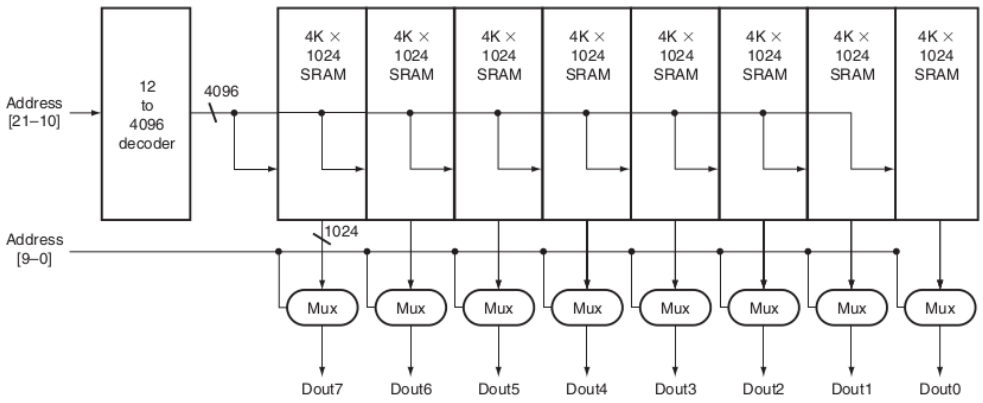


Figura 2.23: SRAM 4M×8 como un array de 8 elementos de 4K×1024.

memorias a un solo bus de salida, 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.

Memorias DRAM

En una memoria SRAM, el valor que se almacena en cada celda se mantiene en el latch 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.

En 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 2.24 se muestra un chip de DRAM de 16 megabits, configurada como 2M$\times$8. Las celdas están organizadas en forma de matriz de 4K$\times$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 los latches 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 los latches de direcciones de columnas bajo el control de la señal de validación de dirección de columnas CAS. La información de este conjunto de latches es decodificada para seleccionar el grupo apropiado de 8 circuitos

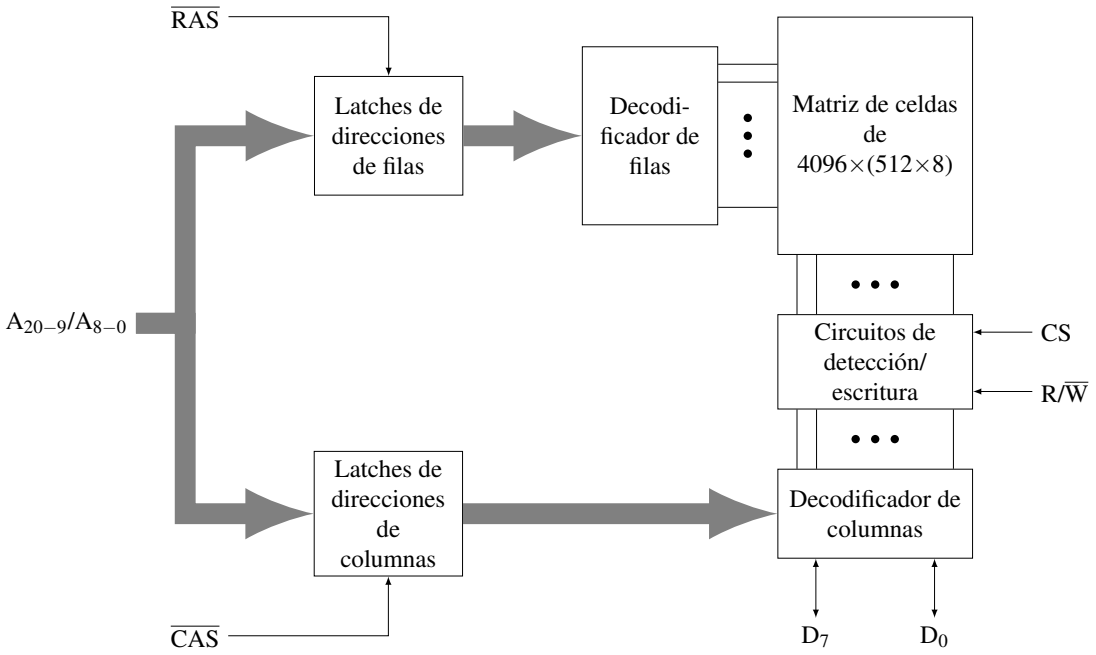


Figura 2.24: DRAM 2M×8.

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.

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 256 Mbits puede estar organizado como 64M×4, 32M×8 o 16M×16.

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*. Cuando se accede a la DRAM de 2M x 8 de la figura 2.24, 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 latch a la salida del circuito detector de

cada columna, de manera que la aplicación de una dirección de fila haga que se carguen los latches 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.

Para conseguir que este proceso sea más rápido y preciso, las memorias DRAM actuales controlan las transferencias de manera síncrona, es decir, con una señal de reloj. Esto permite que los grupos de bits dentro de una fila puedan ser transferidos uno detrás de otro al ritmo que determina la señal de reloj. Este tipo de memorias son conocidas como SDRAM (*Synchronous* DRAM o DRAM síncronas). Además, las memorias DRAM de hoy en día permiten la transmisión de datos en los dos flancos de la señal de reloj (tanto en el de subida como en el de bajada), por lo que reciben el nombre de DDRRAMs (*Double Data Rate* RAMs).

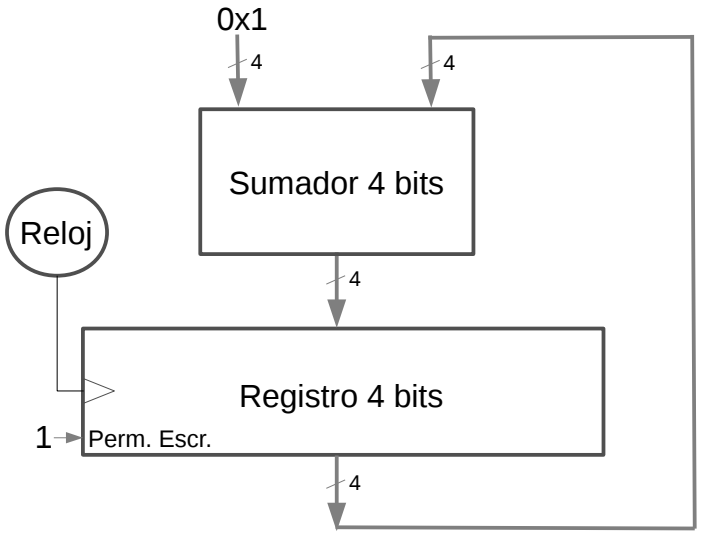
Por último, la memoria, como todos los circuitos vistos hasta ahora, es un dispositivo de almacenamiento electrónico, y como tal es susceptible de cometer errores, devolviendo información diferente a la que fue almacenada originalmente. La memoria DRAM, además, almacena los bits en pequeños condensadores 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.

2.7 EJERCICIOS: CIRCUITOS SECUENCIALES

1. Obtener la tabla de verdad para la determinación del valor de la entrada de un biestable S-R en función del estado actual y del estado siguiente, similar a la tabla 2.11 para el biestable tipo J-K.

2. (*) Considera el siguiente circuito que emplea un sumador y un registro, ambos de 4 bits, como los vistos en este tema.

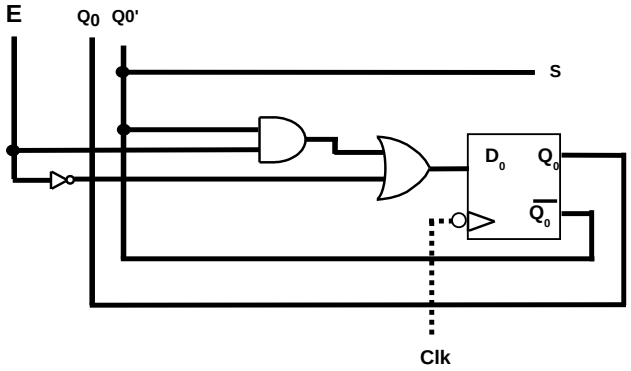


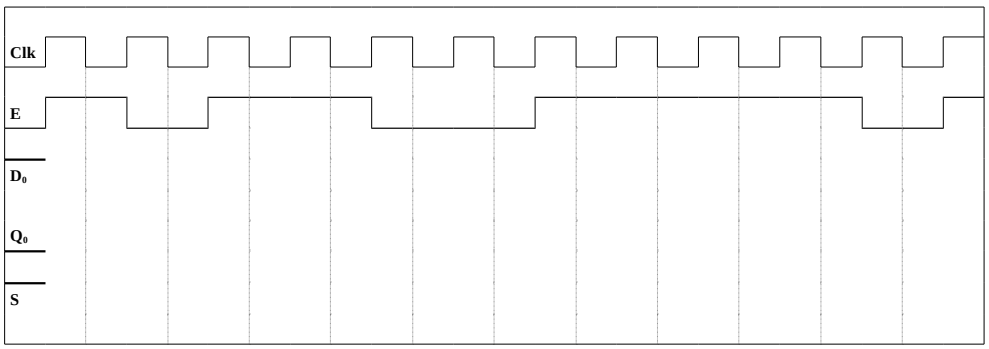
   a) Muestra cómo se implementaría un sumador completo de 1 bit (*full-adder*) a partir de puertas lógicas básicas (AND, OR y NOT).

   b) Muestra cómo a partir de 4 módulos como los diseñados en el apartado anterior podría realizarse el sumador de 4 bits.

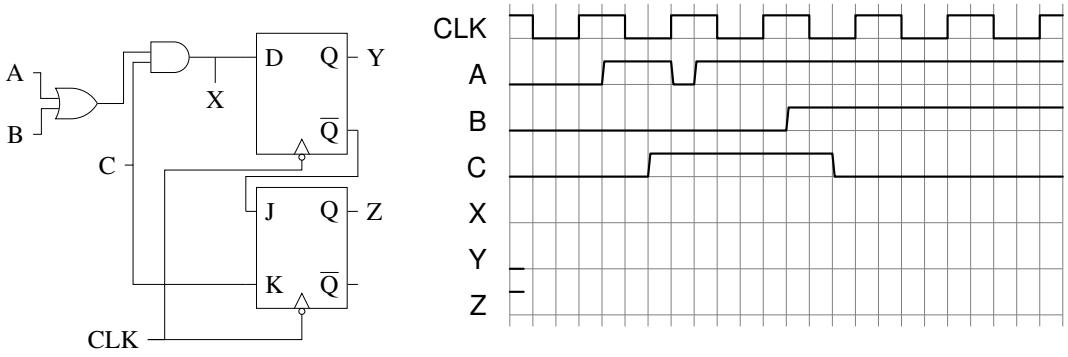
   c) Sabiendo que el registro tiene un retardo de 11 ns y que las puertas AND, OR y NOT tienen retardos de 2 ns, 2 ns y 1 ns respectivamente, calcula la frecuencia máxima de funcionamiento a la que podría operar dicho circuito.

3. (*) Dado el circuito siguiente, correspondiente a un sistema secuencial síncrono tipo Moore, complete el cronograma sabiendo que se parte del estado inicial $Q_0 = 0$ y que los biestables de la memoria de estado se cargan en flanco descendente. *Clk* es la señal de reloj de los biestables, *E* es la entrada y *S* la salida. Supóngase que no se genera retardo alguno en las puertas lógicas ni biestables.

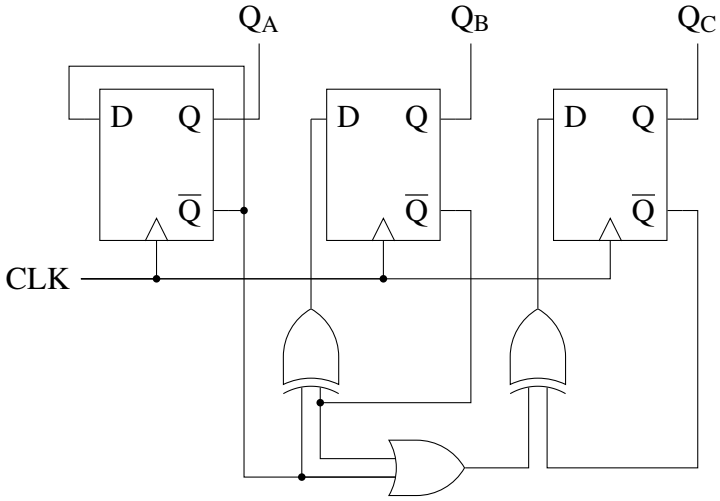




4. Dado el circuito que se muestra a la izquierda, complete el cronograma de la derecha y calcule la frecuencia máxima del circuito, justificando la respuesta. Suponga que el retardo de una puerta AND u OR es de 10 ns y el de un flip-flop es de 30 ns.

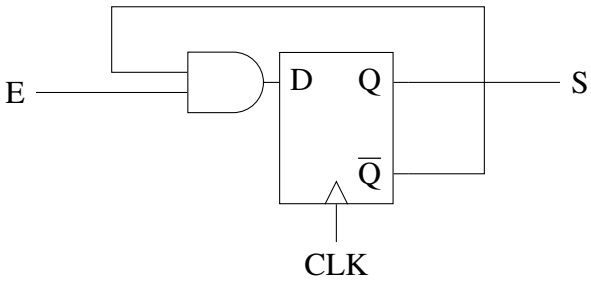


5. (*) Dibujar el cronograma y obtener la secuencia de cuenta del contador de la figura:

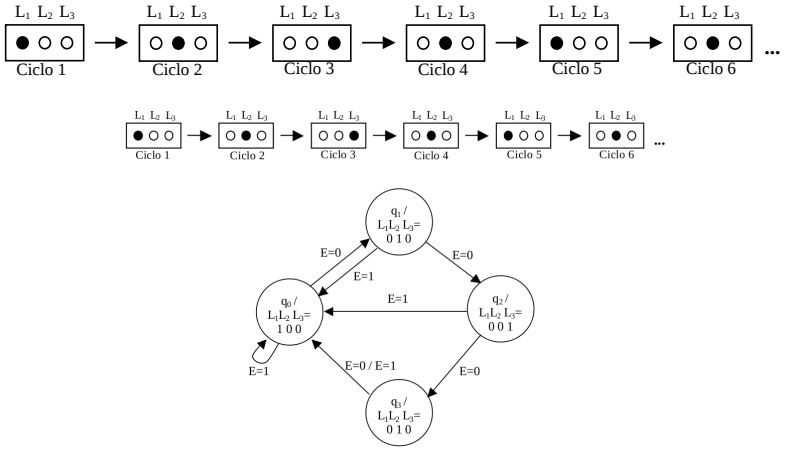


¿Cuál es la frecuencia máxima de funcionamiento si el retraso de un biestable es de 40 ns, el de una puerta OR 20 ns y el de una XOR 25 ns?

6. Obtener la tabla de estados y salidas del circuito de la figura (es decir, obtener el autómata que lo describe).



7. (*) 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.

8. 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.

9. (*) Diseñe un circuito secuencial síncrono con tres entradas (E_2, E_1 y E_0) y una salida (S), y en el que la salida se activará (valdrá 1) 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.

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

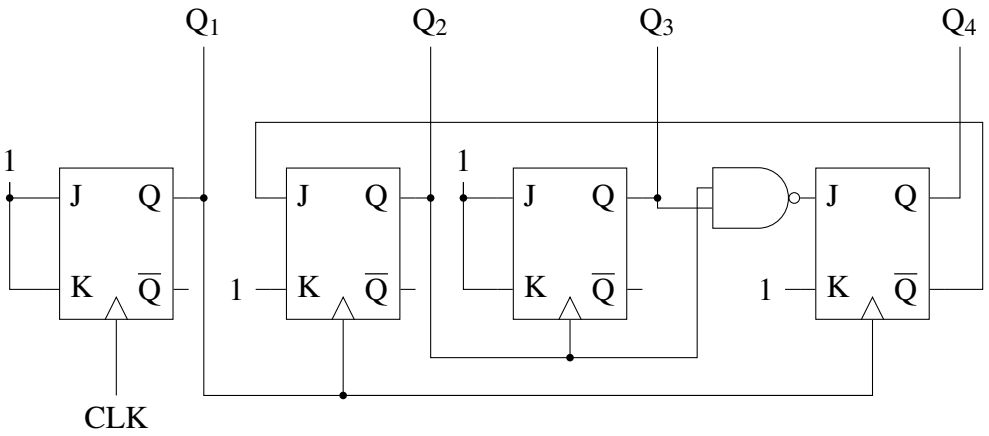
c) Dibuje el circuito que implementa el autómata usando solo puertas NAND y biestables tipo D disparados por flanco descendente.

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.

10. 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).

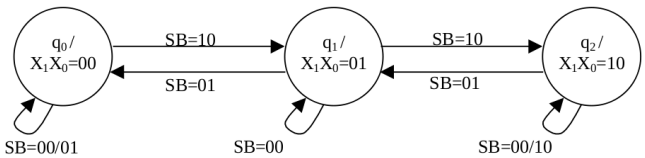
11. (*) Implementar un flip-flop tipo D activo por flanco ascendente a partir de uno tipo T activo por flanco descendente y el mínimo número de puertas NOR (no hay disponibles puertas NOT).

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



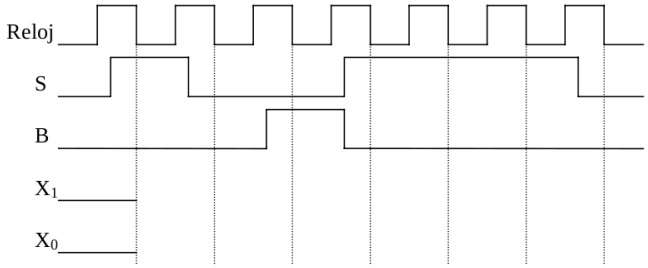
¿Cuál es la frecuencia máxima de funcionamiento si el retraso de un biestable J-K es de 40 ns y el de una puerta NAND 20 ns?

13. 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.

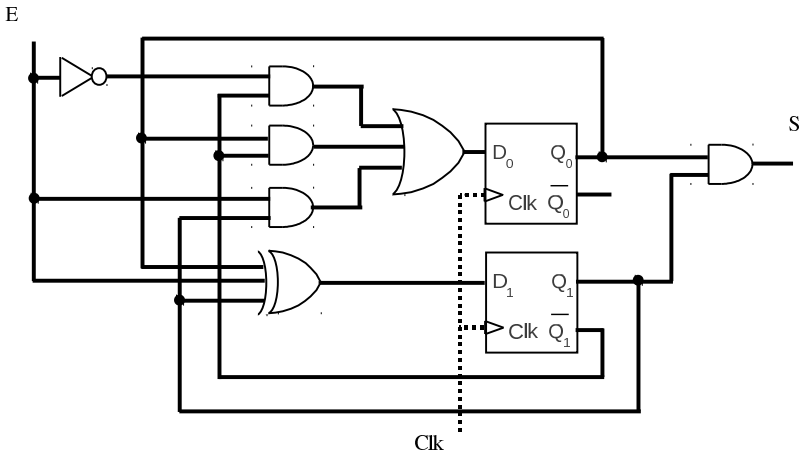


Cuando las entradas S y B valgan ambas cero, el circuito no cambia de estado. Cuando S valga uno y B cero, entonces el número de estado se aumenta en una unidad, salvo que ya estemos en el estado más alto (de 00 se pasa a 01, de 01 a 10, y si estamos en 10 permanecemos en él). Si B vale uno y S cero,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

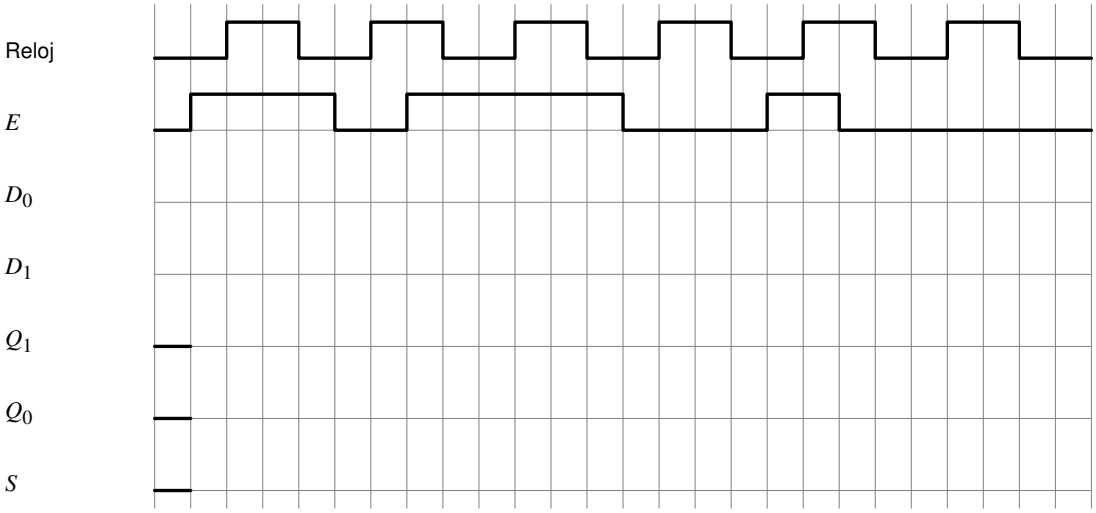
- Construir el circuito secuencial síncrono simplificado que implemente dicho autómata, utilizando flip-flops tipo T, activos en el flanco descendente.
- Rellenar el siguiente cronograma de funcionamiento:



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



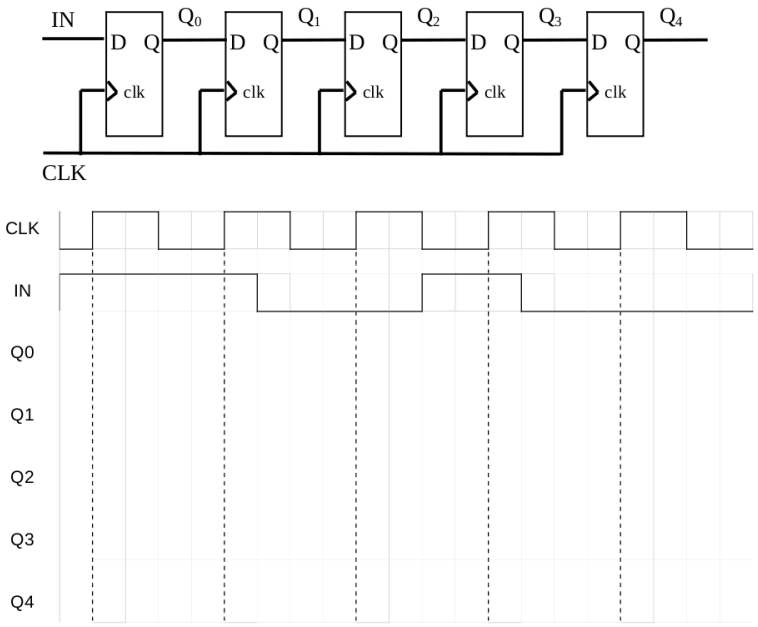
a) Rellenar el siguiente cronograma:



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.

15. Obtener los estados del registro de 5 bits mostrado (inicialmente 00000), para las señales de reloj (CLK) y entrada de datos (IN) indicadas.



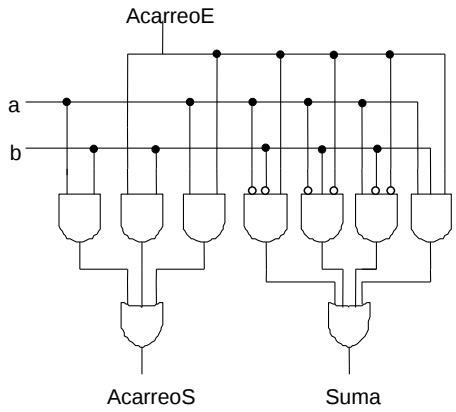
2.8 SOLUCIÓN A EJERCICIOS SELECCIONADOS

2. **Apartado a**

El sumador completo de 1 bit tiene tres entradas, las dos correspondientes a los sumandos (a y b) y una correspondiente al acarreo de entrada (AcE). Las dos salidas corresponden al resultado de la suma (Suma) y al posible acarreo de salida que pudiera tener (AcS). La relación entre las entradas y las salidas de un sumador total está especificado en siguiente tabla de verdad:

Entradas			**Salidas**		**Función**
a	**b**	**AcE**	**AcS**	**Suma**	
0	0	0	0	0	0+0+0=00
0	0	1	0	1	0+0+1=01
0	1	0	0	1	0+1+0=01
0	1	1	1	0	0+1+1=10
1	0	0	0	1	1+0+0=01
1	0	1	1	0	1+0+1=10
1	1	0	1	0	1+1+0=10
1	1	1	1	1	1+1+1=11

Tal y como se ha explicado en el apartado **??**, el circuito lógico que implementa las funciones de salida del sumador completo sería el siguiente:



Apartado b

Para construir un sumador de 4 bits a partir de 4 sumadores completos de 1 bit como el diseñado en el apartado anterior, basta con conectarlos en serie, es decir la salida AcarreoS de cada sumador se conectaría con la entrada AcarreoE del siguiente.

Apartado c

El retardo introducido por el sumador ($T_{suma\ 4\ bits}$) será: $3T_{AcS} + T_{Suma}$, donde T_{AcS} es el tiempo necesario para generar el bit de acarreo en cada sumador completo de 1 bit y T_{Suma} es el tiempo para obtener el bit de suma. A partir del circuito del sumador completo de 1 bit, vemos que $T_{AcS} = 2+2 = 4$ ns y $T_{Suma} = 1+2+2 = 5$ ns. De esta forma, $T_{suma\ 4\ bits} = 17$. Dado que el registro requerirá 11 ns en almancenar el valor de entrada, su salida no estará disponible hasta pasados esos 11 ns. De esta forma, el periodo mínimo de la señal de reloj debería ser 28 ns, o lo que es lo mismo, la frecuencia máxima sería 35.71 MHz.

3. Simulando las entradas (E y Q_0), rellenamos la tabla de la función de transición y la de salida. D_0 se deduce por su ecuación que es:

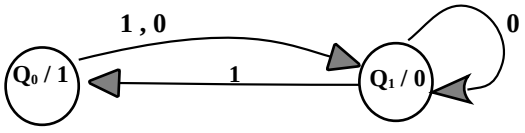
$$D_0 = \bar{E} + E\bar{Q}_0$$

.

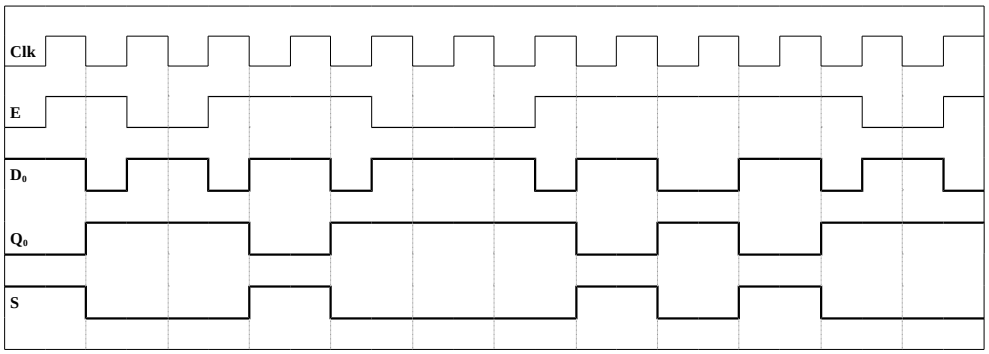
Para rellenar D_0, aplicamos la ecuación que nos dice que cuando E sea igual a 0, D_0 será siempre 1 y cuando E sea igual a 1 habrá que mirar el valor de $\bar{Q}_0$.

Q_0	E	$Q_0^* = D_0$	S
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Si deseamos deducir un autómata finito con este comportamiento (para rellenar más fácilmente el cronograma), sería:



El cronograma queda:



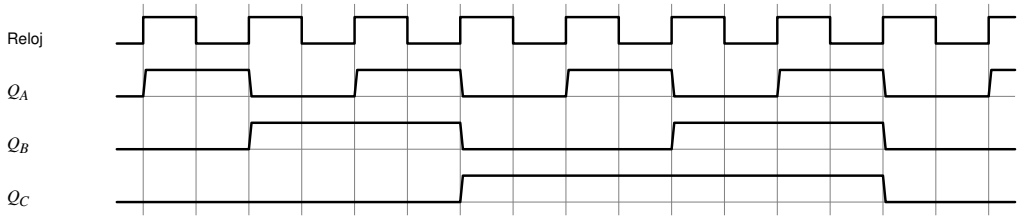
Para ser más correctos, todas las señales deberían reflejar un cierto retardo debido al tiempo de respuesta de los componentes, pero no lo pide el enunciado.

5. 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.

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

$$f_{max} = \frac{1}{T_{min}} = \frac{1000}{85} \times 10^6 Hz = 11.76MHz$$

7. 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
q_0	0	0
q_1	0	1
q_2	1	0
q_3	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

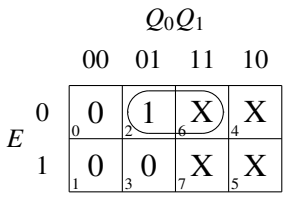
Las expresiones simplificadas de las funciones de salida serán:

$$\begin{aligned} L_1 &= \overline{Q_0}\,\overline{Q_1} \\ L_2 &= \overline{Q_0}\,Q_1 + Q_0\,Q_1 = Q_1 \\ L_3 &= Q_0\,\overline{Q_1} \end{aligned}$$

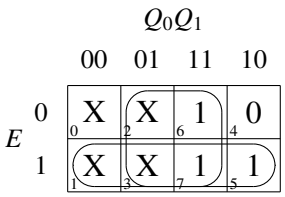
El siguiente paso es la simplificación de las funciones de las entradas de los biestables J-K. Las simplificaciones se muestran en la figura 2.25 y los resultados obtenidos son los siguientes:

$$J_0 = Q_1\,\overline{E} \qquad K_0 = Q_1 + E \qquad J_1 = \overline{E} \qquad K_1 = 1$$

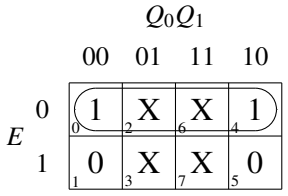
Con lo que el circuito resultante usando puertas AND, OR y NOT será:



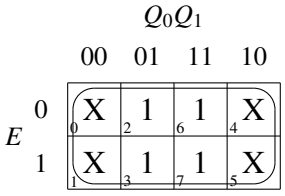
(a) Simplificación de J_0



(b) Simplificación de K_0

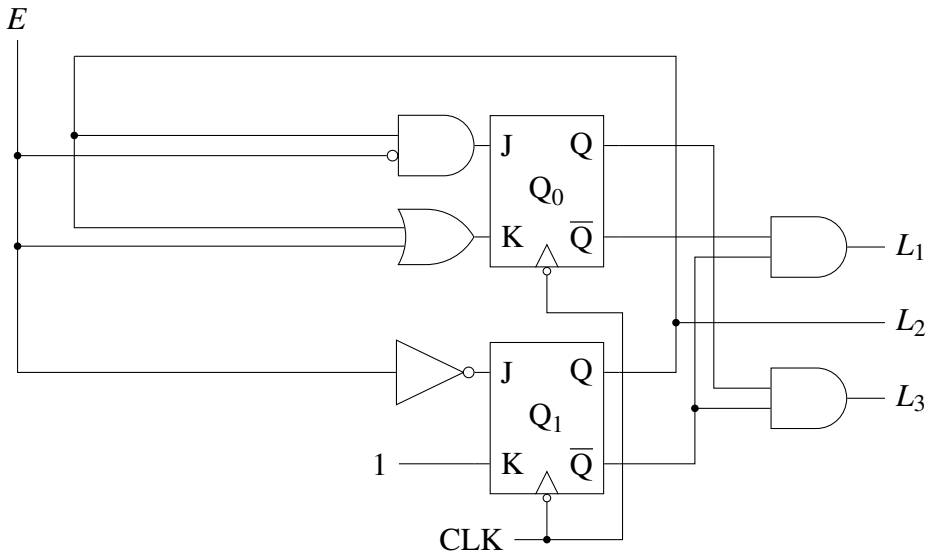


(c) Simplificación de J_1



(d) Simplificación de K_1

Figura 2.25: Simplificación de las entradas de los biestables J-K para el autómata de control del sistema de tres luces.



9. a) Necesitaremos 4 estados para representar la siguiente información:

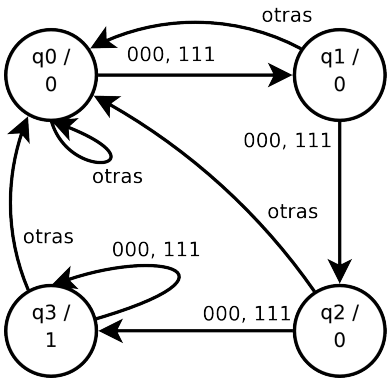
q_0: no ha habido coincidencia en el ciclo anterior.

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).

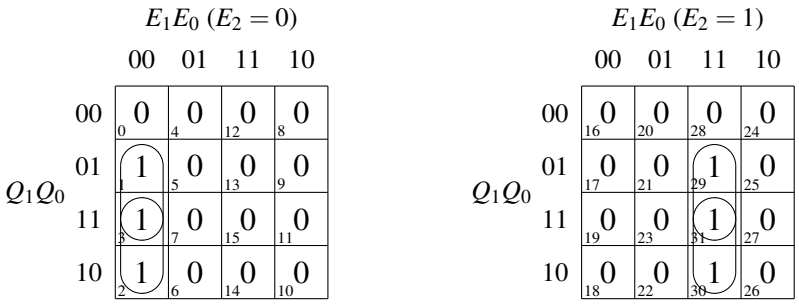
b) Para codificar los 4 estados, necesitaremos 2 flip-flops tipo D cuyas salidas serán Q_0 y Q_1. Como codificación de estados usaremos la usual: $q_0 = 00, ..., q_3 = 11$.

Q_1	Q_0	E_2	E_1	E_0	$Q_1^* = D_1$	$Q_0^* = D_0$
0	0	0	0	0	0	1
0	0	1	1	1	0	1
0	0	X	X	X	0	0
0	1	0	0	0	1	0
0	1	1	1	1	1	0
0	1	X	X	X	0	0
1	0	0	0	0	1	1
1	0	1	1	1	1	1
1	0	X	X	X	0	0
1	1	0	0	0	1	1
1	1	1	1	1	1	1
1	1	X	X	X	0	0

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

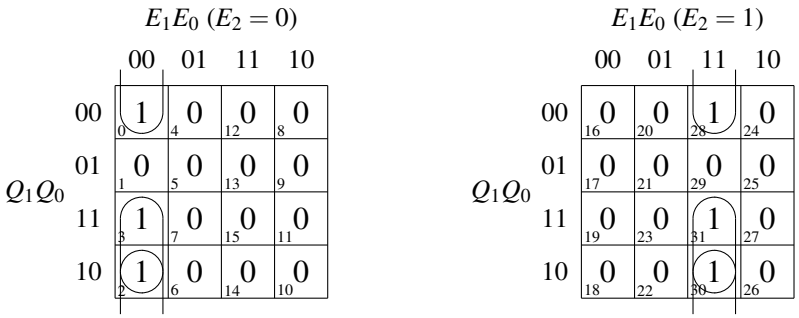
$S = Q_1 Q_0$

El mapa de Karnaugh para D_1 es:



$D_1 = Q_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_0 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_1 \cdot E_2 \cdot E_1 \cdot E_0 + Q_0 \cdot E_2 \cdot E_1 \cdot E_0$

El mapa de Karnaugh para D_0 es:



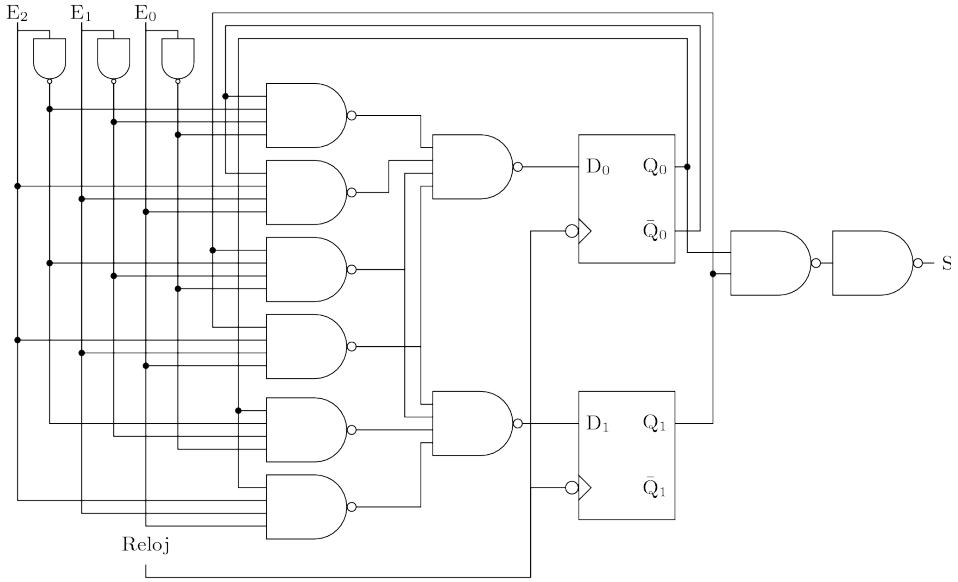
$$D_0 = Q_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + \overline{Q_0} \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_1 \cdot E_2 \cdot E_1 \cdot E_0 + \overline{Q_0} \cdot E_2 \cdot E_1 \cdot E_0$$

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

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

$$\begin{aligned} D_1 &= Q_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_0 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_1 \cdot E_2 \cdot E_1 \cdot E_0 + Q_0 \cdot E_2 \cdot E_1 \cdot E_0 = \\ &= \overline{\overline{Q_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_0 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_1 \cdot E_2 \cdot E_1 \cdot E_0 + Q_0 \cdot E_2 \cdot E_1 \cdot E_0}} = \\ &= \overline{\overline{(Q_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0})} \cdot \overline{(Q_0 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0})} \cdot \overline{(Q_1 \cdot E_2 \cdot E_1 \cdot E_0)} \cdot \overline{(Q_0 \cdot E_2 \cdot E_1 \cdot E_0)}} \end{aligned}$$

$$\begin{aligned} D_0 &= Q_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + \overline{Q_0} \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_1 \cdot E_2 \cdot E_1 \cdot E_0 + \overline{Q_0} \cdot E_2 \cdot E_1 \cdot E_0 = \\ &= \overline{\overline{Q_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + \overline{Q_0} \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0} + Q_1 \cdot E_2 \cdot E_1 \cdot E_0 + \overline{Q_0} \cdot E_2 \cdot E_1 \cdot E_0}} = \\ &= \overline{\overline{(Q_1 \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0})} \cdot \overline{(\overline{Q_0} \cdot \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0})} \cdot \overline{(Q_1 \cdot E_2 \cdot E_1 \cdot E_0)} \cdot \overline{(\overline{Q_0} \cdot E_2 \cdot E_1 \cdot E_0)}} \end{aligned}$$



d) 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:

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

Por tanto, la frecuencia máxima será de 12.5 MHz. Nótese, en cualquier caso, que para alcanzar esta frecuencia máxima de funcionamiento las entradas E_i tendrán que estar disponibles al menos $15+15+15 = 45$ ns antes del flanco activo de la señal de reloj (debido a la tres etapas de puertas NAND situadas a la entrada) y la salida no estará disponible hasta $50+15+15 = 80$ ns después del mismo (debido al biestable y a las dos etapas de puertas NAND situadas justo a la salida).

11. Escribimos en primer lugar la tabla de excitación del flip-flop D y vemos las entradas que habría que dar al flip-flop T para emular ese comportamiento:

D	Q	Q^*	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Obtenemos, simplificando por ceros, que

$$T = (D+Q)(\bar{D}+\bar{Q}) = \overline{\overline{(D+Q)(\bar{D}+\bar{Q})}} = \overline{\overline{(D+Q)} + \overline{(\bar{D}+\bar{Q})}}$$

y el circuito resultante es el siguiente: