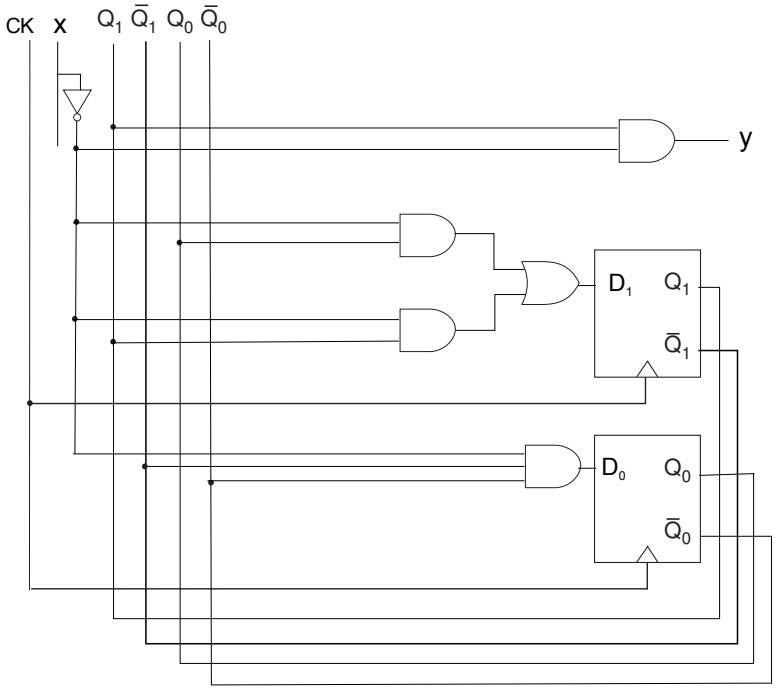
FUNDAMENTOS DOS COMPUTADORES

EJERCICIOS RESUELTOS DE SISTEMAS SECUENCIALES

1. **Obtén la tabla de transición de estados y el diagrama de estados del circuito secuencial de la siguiente figura:**



El primer paso consiste en identificar en el circuito cuales son las entradas, salidas y variables de estado. Como puede verse en la figura, además del reloj (CLK), en este circuito hay una única entrada (x), una única salida (y) y dos variables de estado (Q_1, Q_0), cada una almacenada en un biestable D. Fíjate que como la salida depende tanto de la entrada x como de la variable de estado Q_1, se trata de un sistema tipo Mealy.

El siguiente paso consiste en obtener las expresiones de la salida y las de las entradas de los biestables. Siguiendo las conexiones del esquema del circuito se obtienen las siguientes expresiones:

$$\begin{aligned} \mathrm{D}_1 &= \overline{x}Q_0 + \overline{x}Q_1 \\ D_0 &= \overline{x}\ \overline{Q_1}\ \overline{Q_0} \\ y &= \overline{x}Q_1 \end{aligned}$$

El siguiente paso consiste en calcular la tabla de transición de estados del circuito a partir de las ecuaciones anteriores substituyendo en ellas los diferentes valores de x, Q_1^n y Q_0^n y teniendo en cuenta que $Q^{n+1} = D$. El proceso de cálculo de la primera fila de la tabla (el estado actual en esta fila es $Q_1^n Q_0^n = 00$) es el siguiente:

- Para $x = 0$: $Q_1^{n+1} = D_1 = 1 \cdot 0 + 1 \cdot 0 = 0$; $Q_0^{n+1} = D_0 = 1 \cdot 1 \cdot 1 = 1$
- Para $x = 1$: $Q_1^{n+1} = D_1 = 0 \cdot 0 + 0 \cdot 0 = 0$; $Q_0^{n+1} = D_0 = 0 \cdot 1 \cdot 1 = 0$

Este proceso se repetiría para todos los estados. De la misma manera también se calcularía el valor de la salida y para cada combinación de estado y de entrada. La tabla resultante es la siguiente:

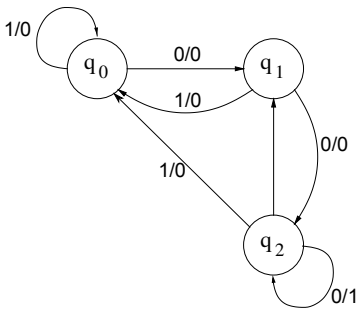
	$Q_1^{n+1}Q_0^{n+1}$		y	
$Q_1^nQ_0^n \backslash x$	0	1	0	1
00	01	00	0	0
01	10	00	0	0
10	10	00	1	0
11	10	00	1	0

Fíjate que los estados 10 y 11 son equivalentes, por lo que podemos eliminar uno de ellos y quedarnos solamente con 3 estados. Si se utiliza la siguiente codificación de estados (podría utilizarse cualquier otra):

Q_1Q_0	Estado (S^n)
00	q_0
01	q_1
10	q_2

se obtiene la siguiente tabla de estados resumida y su diagrama de estados equivalente, que definen completamente el comportamiento del circuito:

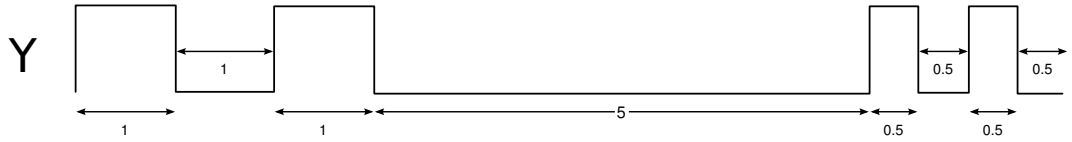
	S^{n+1}		y	
$S^n \backslash x$	0	1	0	1
q_0	q_1	q_0	0	0
q_1	q_2	q_0	0	0
q_2	q_2	q_0	1	0



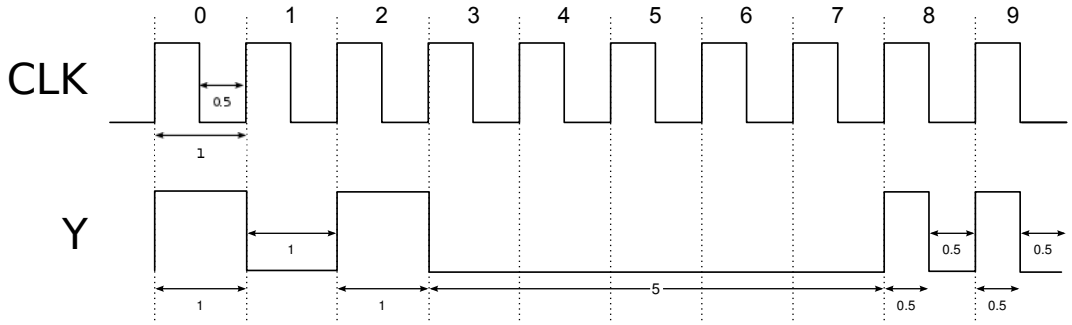
A partir de esta tabla de estados o de su correspondiente diagrama se puede predecir el comportamiento del circuito ante una secuencia determinada de entrada. Por ejemplo, suponiendo que el circuito esté inicialmente en el estado q_0 y que se recibe por la entrada X la secuencia de valores 1100011011, las transiciones de estado y los valores de salida del circuito serían los siguientes:

$\mathbf{X}$:		1	1	0	0	0	1	1	0	1	1	...
$\mathbf{S^n}$:	q_0	q_0	q_0	q_0	q_1	q_2	q_2	q_0	q_0	q_1	q_0	q_0	...
$\mathbf{Y}$:		0	0	0	0	1	0	0	0	0	0	...

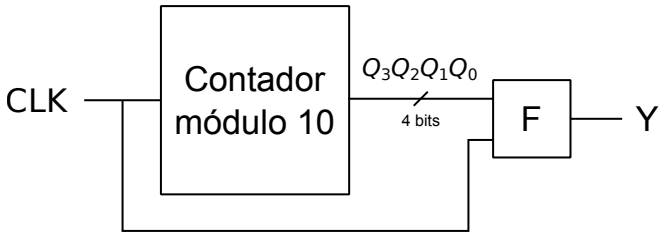
2. **Implementa un circuito que reciba como entrada una señal CLK de 1 Hz. y ciclo de trabajo del 50 % y genere periódicamente una señal Y como la mostrada en la siguiente figura (los tiempos están en segundos):**



Los ejercicios de generación de señales periódicas a partir de una dada deben basarse en el uso de un contador. A la hora de decidir el módulo del contador a utilizar es útil dibujar un cronograma que relacione la señal de CLK disponible con la señal a generar. En este caso la señal de CLK dada tiene una frecuencia de 1Hz. (periodo 1 seg.) y un ciclo de trabajo del 50 % (es decir, está en alta la mitad del tiempo), por lo que el cronograma sería:



De este cronograma se deduce que es necesario tener en cuenta 10 ciclos de la señal CLK. Para llevar la cuenta de estos ciclos se usará un contador binario módulo 10. Deberemos definir además una función F que obtenga la forma de onda deseada a partir del estado (la cuenta) del contador. Como puede observarse en el cronograma de arriba, en los dos últimos estados de la cuenta la salida Y no mantiene un único valor (0 o 1) durante todo el ciclo, sino que toma el mismo valor que tenga CLK. Por lo tanto no es posible calcular Y si se tiene en cuenta únicamente el estado del contador, y será necesario utilizar también la señal CLK. El diseño de alto nivel de este problema se puede observar en la siguiente figura:

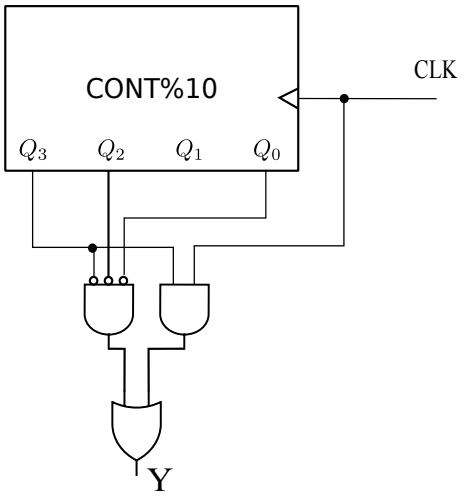


Queda ahora por diseñar la función F que genera la forma de onda deseada a partir del estado del contador y la señal de reloj. Dibujamos directamente el mapa de Karnaugh de la función Y a minimizar. El mapa será de 5 variables: Q_3, Q_2, Q_1, Q_0 y CLK. El mapa y el resultado de la minimización son los siguientes:

$Q_3Q_2 \backslash Q_1Q_0CLK$	000	001	011	010	110	111	101	100
00	1	1	0	0	0	0	1	1
01	0	0	0	0	0	0	0	0
11	-	-	-	-	-	-	-	-
10	0	1	1	0	-	-	-	-

$$Y = \overline{Q_3}\ \overline{Q_2}\ \overline{Q_0} + Q_3 CLK$$

El circuito completo quedaría:



3. **Diseña un contador de 8 bits capaz de realizar diferentes cuentas según una línea de control X de acuerdo a la siguiente tabla:**

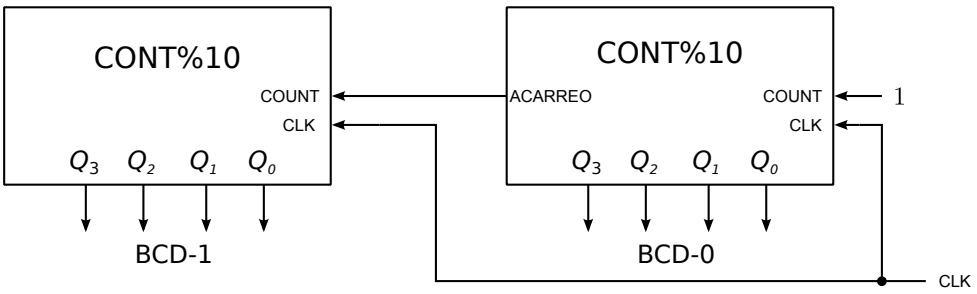
X	Módulo	Cuentas
0	100	00-99 (BCD)
1	60	00-59 (BCD)

La solución al ejercicio consiste en diseñar un contador con la capacidad de seguir dos secuencias de estados diferentes en función de una variable de control X. Por lo tanto, se deben implementar dos contadores en uno.

En las clases de teoría se ha visto una situación parecida a la hora de construir un contador ascendente/descendente. Tal y como se explicó entonces, en primer lugar se debe diseñar cada uno de los dos contadores por separado.

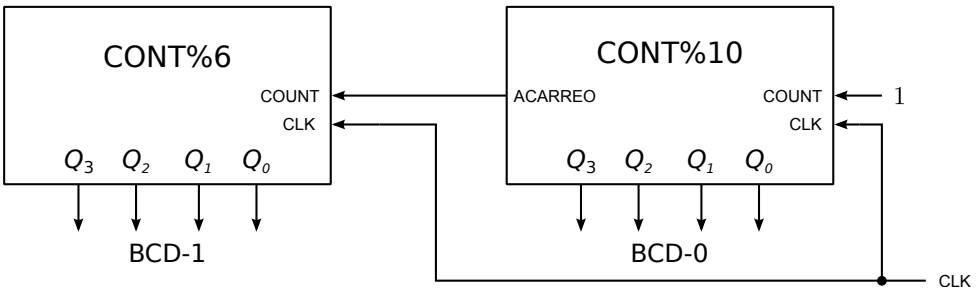
Empezando con el contador módulo 100, como se pide que las cuentas estén en formato BCD, usaremos dos contadores módulo 10 conectados, de forma que cada contador módulo 10 almacenará una cifra BCD (que puede tomar valores de 0 a 9).

El diseño sería entonces:



Nótese que la señal de COUNT del contador de la derecha está a 1, pero la señal de COUNT del contador de la izquierda está conectada a la señal de ACARREO del contador de la derecha. De esta forma el contador de la derecha avanza en cada ciclo de reloj, pero el de la izquierda solo avanza cada vez que el contador de la derecha llega a su última cuenta, cuando su señal de ACARREO se pone a 1. Por tanto, el contador de la derecha cuenta unidades y el de la izquierda cuenta decenas.

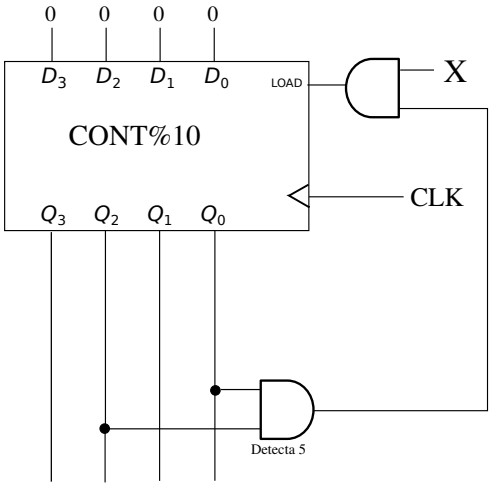
De la misma manera, el contador módulo 60 se puede diseñar conectando un contador módulo 10 con otro módulo 6 (en este caso la cifra BCD más significativa solo toma valores entre 0 y 5):



Comparando los dos diseños anteriores vemos que:

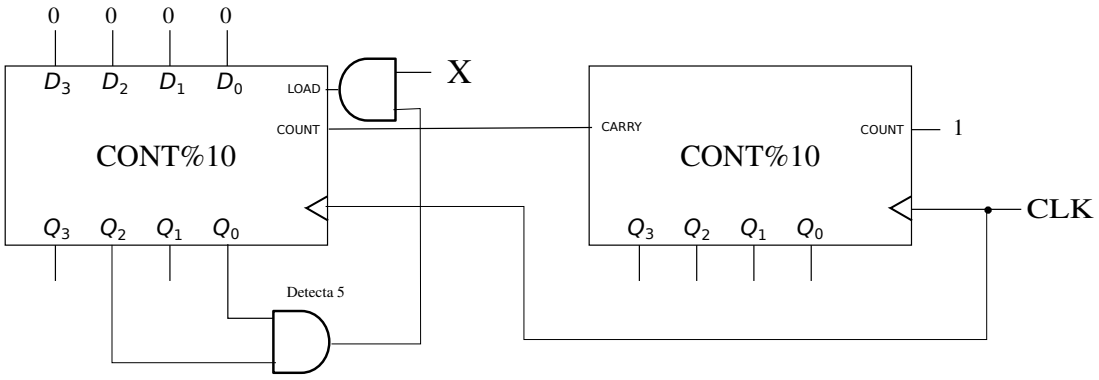
- El contador que mantiene la cifra BCD menos significativa (BCD-0) es el mismo en ambos casos, un contador módulo 10.
- El contador que mantiene la cifra BCD más significativa (BCD-1) es diferente: en el primer caso es un contador módulo 10 y en el segundo un contador módulo 6. Será necesario entonces diseñar un contador que funcione como módulo 10 o como módulo 6 dependiendo de la señal de control X. Es más, como ambas cuentas comienzan en el estado 0, la diferencia entre ellas se reduce únicamente al valor hasta el que se cuenta: 9 cuando $X = 0$ y 5 cuando $X = 1$.

Podemos utilizar para el diseño un contador módulo 10 y hacer que el contador se reinicie después de la cuenta 5 si $X = 1$. Para ello podemos utilizar la señal LOAD del contador (activándola en la cuenta 5 si $X = 1$) como se muestra en esta figura:



BCD-1

La solución completa del ejercicio quedaría:



BCD-1 BCD-0

4. **Construye un sistema secuencial con dos entradas A y B de 4 bits y una señal de reloj CLK que realice las siguientes operaciones:**

   ***a*) Durante 3 ciclos del reloj, la salida S debe ser igual a la entrada A.**

   ***b*) Durante los 2 ciclos siguientes, S debe ser igual a la entrada B.**

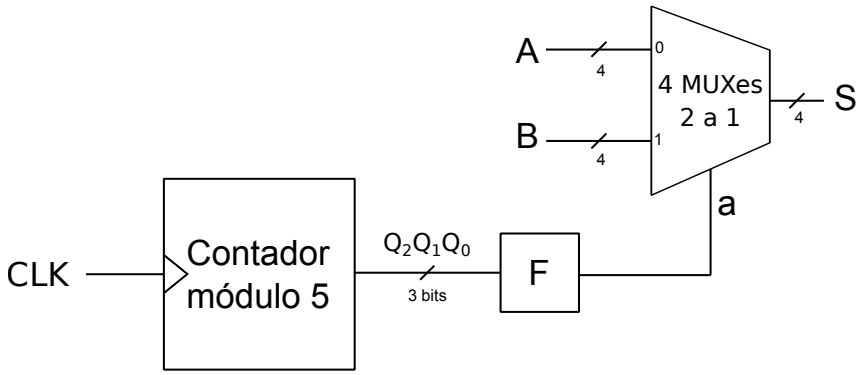
   Estas operaciones se realizan continuamente, es decir, una vez realizada la segunda operación vuelve a comenzarse por la primera.

   En primer lugar, el que las operaciones se realicen en diferentes ciclos y sean repetitivas nos sugiere utilizar un contador para llevar la cuenta de la operación a realizar en cada momento. Como se distinguen solo 5 ciclos de reloj (3 ciclos en la primera fase y 2 ciclos en la segunda), el contador debería ser módulo 5.

   En segundo lugar, se tienen que implementar todas las operaciones que realiza el circuito. En este caso, ambas operaciones consisten únicamente en seleccionar una de las dos entradas del circuito A o B. El circuito que nos permite seleccionar una de entre varias entradas es el MUX. Como hay que escoger entre dos entradas de 4 bits, serán necesarios 4 MUXes (uno por bit) 2 a 1. Los MUXes tendrán por tanto una única variable de control.

   Por último, para escoger la operación adecuada en cada ciclo es necesario diseñar la función que calcule los valores de la variable de control de los MUXes en función del estado del contador. Para ello se definirá una tabla de verdad que indique el valor que toma esta variable para cada cuenta del contador y se realizará la correspondiente minimización por Mapas de Karnaugh.

   Teniendo en cuenta lo anterior, el diseño de alto nivel del circuito sería el siguiente:

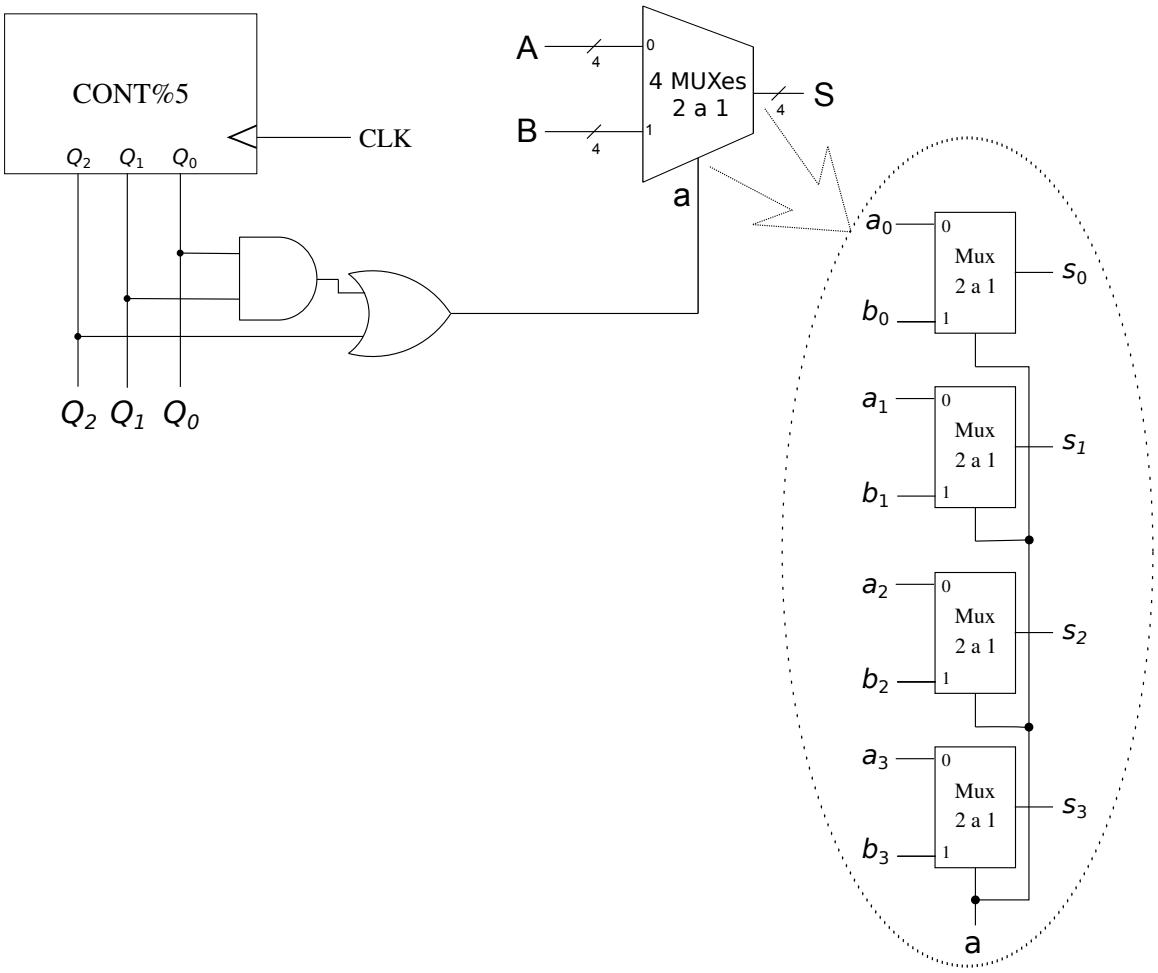


Para el diseño de la función de control de los MUXes, teniendo en cuenta que se ha puesto A en la entrada 0 de los MUXes (ver el diseño de alto nivel) y que el contador módulo 5 realiza la cuenta $\{0, 1, 2, 3, 4\}$, se utilizaría la siguiente tabla de verdad:

Q_2	Q_1	Q_0	a
0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	0
1	1	0	–
1	1	1	–
1	0	1	–
1	0	0	1

Minimizando por Mapas de Karnaugh se obtiene: $a = Q_2 + Q_1Q_0$.

El diseño completo del circuito quedaría de la siguiente manera (se incluye también la implementación detallada de los 4 MUXes):

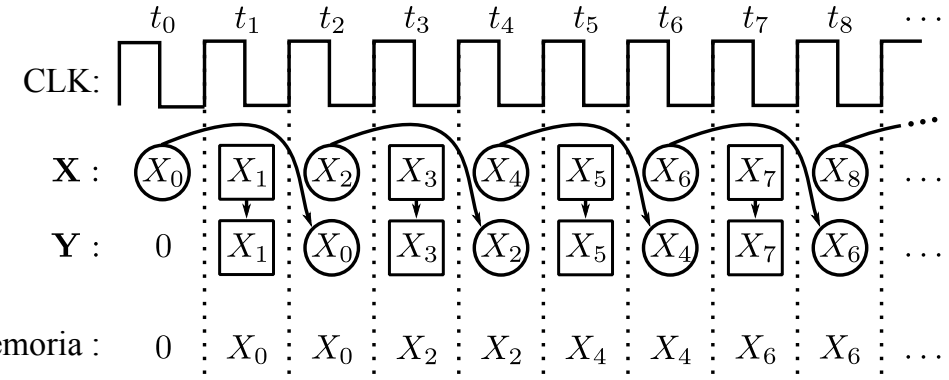


5. **Diseña un circuito que tenga una entrada X (de 4 bits), una señal de reloj CLK y una salida Y (también de 4 bits). El circuito invertirá el orden de los datos de entrada de acuerdo al siguiente patrón de funcionamiento:**

X :	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	...
Y :	0	X_1	X_0	X_3	X_2	X_5	X_4	X_7	X_6	...

Analizando el enunciado del ejercicio se puede deducir que será necesario contar con algún tipo de memoria que almacene los datos de entrada que no son redirigidos directamente a la salida. Por ejemplo, fijándose en X_0 se puede comprobar que no aparece en la salida Y hasta 2 ciclos después de que se haya recibido en la entrada X, por lo que habrá que almacenarlo durante ese tiempo. Fíjese que aquí los subíndices indican el ciclo en el que se recibe el dato por la entrada.

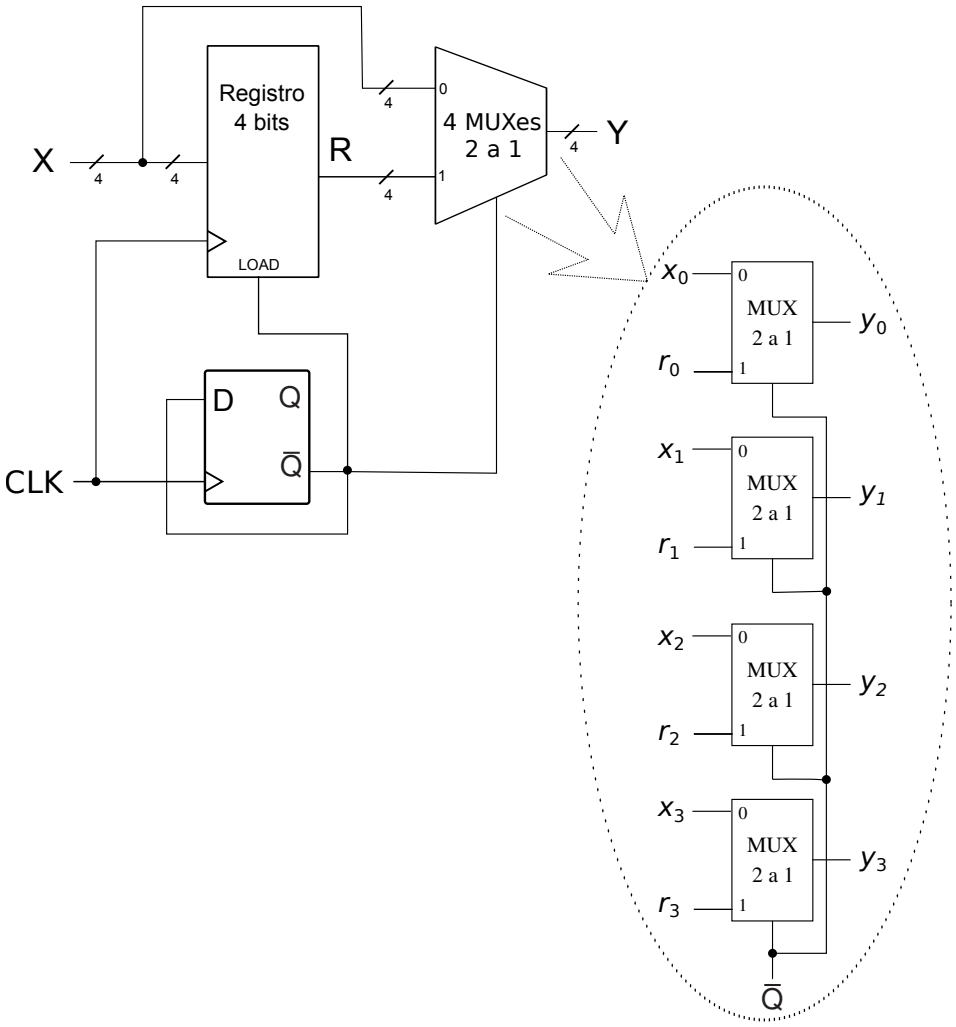
Por lo tanto, para determinar el tamaño de memoria necesario para solucionar el ejercicio, hay que analizar la relación temporal entre los ciclos de reloj en los que los datos aparecen en la entrada y aquellos en los que son enviados a la salida. De esta manera se podrá comprobar qué datos hay que almacenar y durante cuánto tiempo. Este análisis se resume en la siguiente figura:



Como se puede ver, en los ciclos pares el circuito debe almacenar el dato de entrada actual y dar como salida el dato almacenado previamente (2 ciclos antes). En los ciclos impares el dato de entrada pasa directamente a la salida, manteniéndose en memoria el dato de entrada del ciclo anterior. Dado que sólo es necesario almacenar un dato de entrada, en este caso será suficiente con utilizar un registro de 4 bits. En concreto se usará uno con señal de carga síncrona ($LOAD$) activa a nivel alto.

Además de este registro, también va a ser necesario utilizar MUXes para elegir el valor que se envía a la salida del circuito en cada instante: el dato almacenado en la memoria en los ciclos pares o el dato presente en la entrada en los ciclos impares. Se utilizan 4 MUXes (uno por bit) de tamaño 2 a 1 (se elige entre dos valores). Para controlar los MUXes será necesario llevar la cuenta de los ciclos pares e impares. Esto puede hacerse, por ejemplo, con un contador módulo 2: 0 indica un ciclo par y 1 un ciclo impar. Un contador módulo 2 puede realizarse con un único biestable D poniendo como entrada del biestable $\overline{Q}$.

Con los componentes básicos que se han identificado, el diseño del circuito quedaría de la siguiente forma:



6. **Diseña un circuito con dos entradas, *DATO* (de 5 bits en binario puro) y *CLK* (reloj), y una salida *CUENTA* (de 4 bits en binario puro). El circuito funciona en secuencias de 13 ciclos. En los primeros 12 ciclos se reciben por la entrada *DATO* los valores $MIN, MAX, V_0, V_1, ..., V_9$. El circuito contará cuántos de los valores V_i recibidos cumplen que $MIN \leq V_i \leq MAX$. El valor de la cuenta estará en todo momento disponible en la salida *CUENTA*. El último de los ciclos, el ciclo 13, se reserva para restablecer el estado inicial del circuito y prepararlo para volver a comenzar una nueva secuencia. Supón que siempre se cumple que $MAX \geq MIN$.**

Ya que el funcionamiento del circuito se repite cada 13 ciclos, la tabla siguiente resume cuáles son los eventos que ocurren en cada ciclo y las acciones que es necesario realizar:

Ciclo	**Evento**	**Acción**
0	Llega el valor *MIN*	Almacenar *MIN*
1	Llega el valor *MAX*	Almacenar *MAX*
2	Llega V_0	Compararlo con *MIN* y *MAX*
3	Llega V_1	Compararlo con *MIN* y *MAX*
4	Llega V_2	Compararlo con *MIN* y *MAX*
5	Llega V_3	Compararlo con *MIN* y *MAX*
6	Llega V_4	Compararlo con *MIN* y *MAX*
7	Llega V_5	Compararlo con *MIN* y *MAX*
8	Llega V_6	Compararlo con *MIN* y *MAX*
9	Llega V_7	Compararlo con *MIN* y *MAX*
10	Llega V_8	Compararlo con *MIN* y *MAX*
11	Llega V_9	Compararlo con *MIN* y *MAX*
12	Termina	Reiniciar el circuito para la siguiente secuencia

Del análisis de la tabla anterior se puede deducir qué componentes será necesario utilizar en el diseño del circuito. En primer lugar será necesario un contador módulo 13 para llevar la cuenta de los ciclos de reloj. A partir del estado de este contador se calcularán las funciones que permitan controlar el funcionamiento del circuito en cada ciclo. En segundo lugar será necesario utilizar registros de carga paralela para almacenar los valores de *MIN* y *MAX* que se reciben en los dos primeros ciclos. También habrá que utilizar comparadores para realizar la comparación de los valores que se reciben en los ciclos 2 a 11 con *MIN* y *MAX*. Por último se necesitará un segundo contador para llevar la cuenta de cuantos de esos valores cumplen la condición $MIN \leq V_i \leq MAX$.

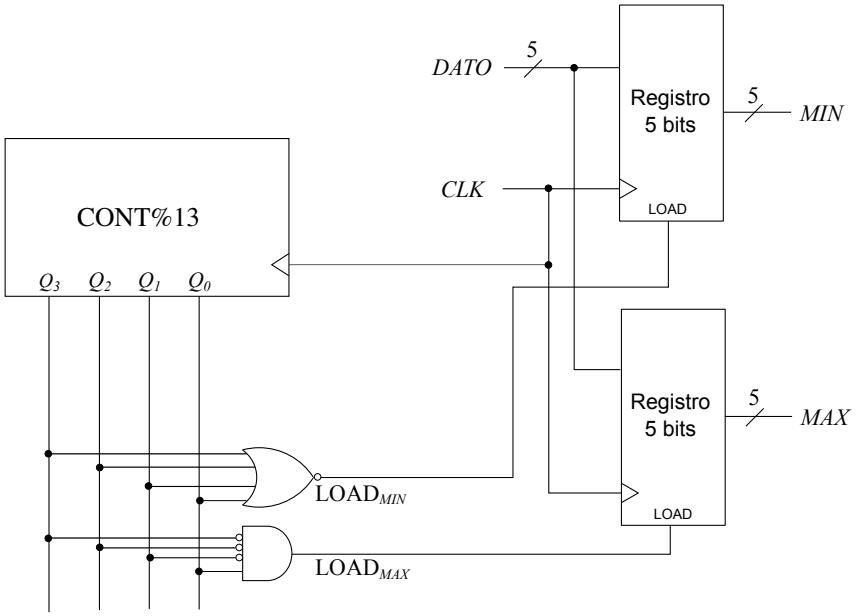
Los registros tendrán que estar controlados por funciones que activen la carga en los ciclos 0 para *MIN* y 1 para *MAX*. Sus tablas de verdad en relación con las cuentas del contador módulo 13 serían:

Q_3	Q_2	Q_1	Q_0	LOAD$_{MIN}$	LOAD$_{MAX}$
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	1	0	0
0	0	1	0	0	0
0	1	1	0	0	0
0	1	1	1	0	0
0	1	0	1	0	0
0	1	0	0	0	0
1	1	0	0	0	0
1	1	0	1	–	–
1	1	1	1	–	–
1	1	1	0	–	–
1	0	1	0	0	0
1	0	1	1	0	0
1	0	0	1	0	0
1	0	0	0	0	0

Las expresiones mínimas de ambas funciones son:

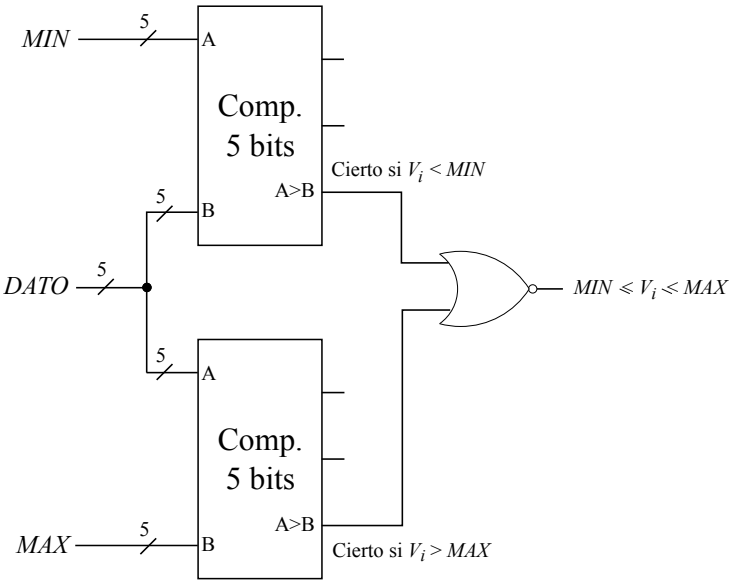
$$\begin{aligned} \text{LOAD}_{MIN} &= \overline{Q_3}\ \overline{Q_2}\ \overline{Q_1}\ \overline{Q_0} = \overline{Q_3 + Q_2 + Q_1 + Q_0} \\ \text{LOAD}_{MAX} &= \overline{Q_3}\ \overline{Q_2}\ \overline{Q_1}\ Q_0 \end{aligned}$$

quedando el circuito de la siguiente manera:



Una vez cargados los valores de *MAX* y *MIN*, en los ciclos siguientes cada valor V_i recibido por la entrada *DATO* debe compararse con *MAX* y con *MIN*, por lo

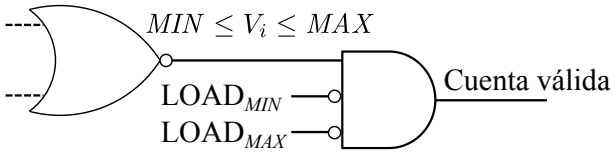
que habrá que usar dos comparadores. Al ser los números a comparar de 5 bits, serán necesarios comparadores de ese tamaño. La parte del circuito que realiza la comparación podría diseñarse de la forma siguiente:



Como puede verse, en todo momento se compara el valor de la entrada *DATO* (V_i) con el contenido de ambos registros. El comparador de la parte superior compara V_i con *MIN*, y la señal $A > B$ indicará si V_i es menor que *MIN* y, por tanto, está fuera de rango. Por su parte, el comparador inferior compara V_i con *MAX*, y la señal $A > B$ indicará si V_i es mayor que *MAX* y, por tanto, está fuera de rango. La puerta NOR final dará un 1 como salida si y solo si V_i no es ni menor que *MIN* ni mayor que *MAX* (o lo que es lo mismo $MIN \leq V_i \leq MAX$). La siguiente tabla muestra algunos ejemplos:

DATO (V_i)	***MIN***	***MAX***	$A > B$ **de *MIN***	$A > B$ **de *MAX***	**NOR**
2	3	5	1	0	0
3	3	5	0	0	1
4	3	5	0	0	1
5	3	5	0	0	1
6	3	5	0	1	0

Tal y como se ha diseñado el circuito de comparación, los comparadores funcionarán en todos los ciclos, incluidos los ciclos 0 y 1 en los que se almacenan los valores *MAX* y *MIN*. Por tanto, es necesario anular el resultado de la comparación durante esos ciclos. Esto puede hacerse añadiendo una puerta AND en la salida del circuito de comparación:



Finalmente, es necesario diseñar un contador que lleve la cuenta del número de ciclos en los que se activa la señal **Cuenta válida**. A lo sumo el contador llegará hasta 10 (el número máximo de valores de V_i que se introducen por la entrada *DATO*), así que puede usarse un contador módulo 16. Para que solo se cuenten los valores V_i que cumplen la condición $MIN \leq V_i \leq MAX$, se conecta la señal **Cuenta válida** a la entrada **COUNT** de este contador. Además como tiene que reiniciarse en el ciclo 13, se puede utilizar la señal **LOAD** para cargar un 0 cuando el contador módulo 13 llegue a su última cuenta (la cuenta 12). Así pues, el circuito completo quedaría así: