Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

CONTADORES Y SECUENCIADORES

Índice

Presentación

A continuación se presentan los contenidos necesarios para saber cómo usar los **registros** junto con la **lógica combinacional** para generar **circuitos secuenciales**.

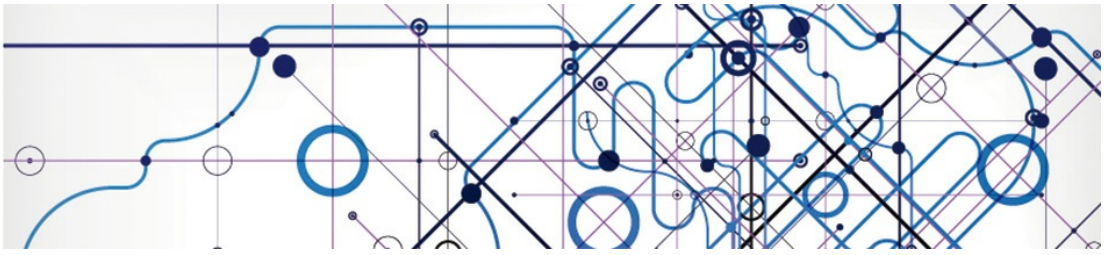
Centraremos el estudio en los contadores y secuenciadores definidos por **Moore**. Hay que dejar claro que hay muchos más sistemas de control que utilizan esta tecnología, como compiladores o analizadores y otras técnicas de implementación (además de Moore), como por ejemplo Mealy.

Tanto Moore como Mealy son dos **maneras diferentes** de construir un sistema secuencial. Por simplicidad, nos centraremos en uno de ellos, principalmente por ser un sistema secuencial de fácil comprensión (que sea un sistema de fácil comprensión no significa que sea el sistema más óptimo).

La principal **diferencia** de un **sistema secuencial** frente a uno **combinacional** es que en los sistemas combinacionales las secuencias que se produzcan en las entradas dependerán de las secuencias que se produzcan en la salida, mientras que en los sistemas secuenciales también dependerá del estado actual, que se verá afectado por la secuencias de entradas anteriores.

Los **objetivos** que se pretenden alcanzar en este tema son los siguientes:

- Aprender qué representan y cómo funcionan los **sistemas secuenciales**.
- Identificar el funcionamiento de la **máquina de Moore** y saber interpretar su diagrama y tablas.
- Analizar casos de circuitos secuenciales para **ejemplificar** los sistemas digitales existentes.



Sistemas secuenciales

En los sistemas secuenciales las **salidas dependen** de las **entradas** y de los **estados** en los que se encuentre. Además, para cada conjunto de entradas la máquina evoluciona cambiando de estado. Así, el comportamiento para las salidas y los estados se puede definir de la siguiente manera:

$$Z(t+1) = E(t) \cdot X(t)$$

$$E(t+1) = E(t) \cdot X(t)$$

En detalle ¿Cuál es la nomenclatura de estas fórmulas?

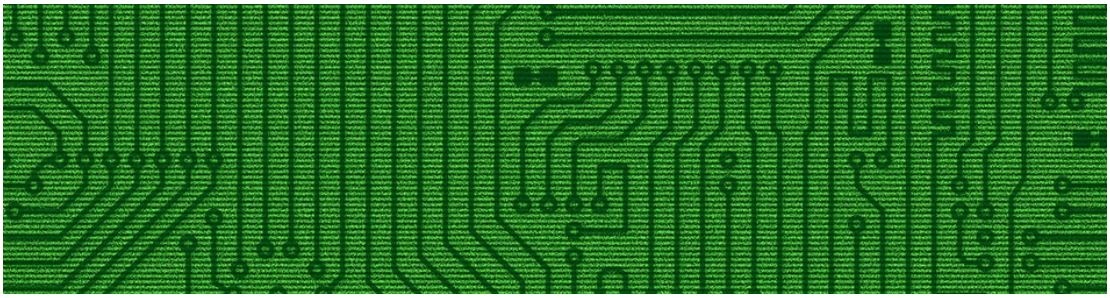
¿Cuál es la nomenclatura de estas fórmulas?

- **Z(t+1)**: salidas futuras del sistema.
- **E(t)**: estado actual.
- **X(t)**: entradas actuales.
- **E(t+1)**: estado futuro.

A continuación, se definen **dos conceptos básicos** para poder avanzar a lo largo del texto:

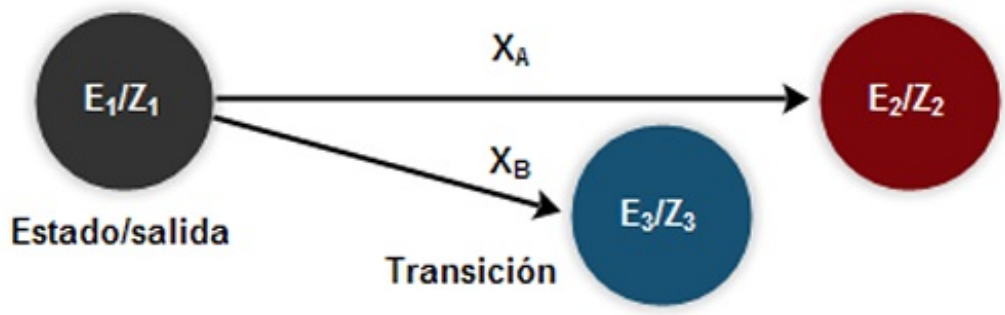
Máquinas finitas	Una máquina de estados finita es una máquina (digital en nuestro caso) en la que su **construcción** se puede realizar en un **conjunto** de **estados finitos** (puede ser de tamaño grande, pero finito). Hay que pensar en un semáforo o en una máquina de vending. Para utilizarlas hay que seguir una serie de pasos. Cada uno de esos pasos es un estado, siendo el comportamiento de la máquina diferente para cada uno de ellos. En el semáforo el avance de estado viene determinado por el reloj y en cada estado la salida es el conjunto de luces a encender, mientras que en la máquina de vending, el avance viene determinado por la intervención del usuario y la salida es el mensaje del display, el avance de motores, etc.

Diagramas y tablas de transición	Antes de profundizar en cada una de las máquinas hay que entender el **significado** de los diagramas y tablas de transición y su correcto **funcionamiento**. Un diagrama de estados es semejante a un **árbol o grafo de estados**, donde reflejaremos los estados mediante un círculo, en el que se indicará el estado en el que nos encontramos y una flecha, que indicará la existencia de una transición entre un estado y otro, o entre el mismo estado. Sobre este diagrama debemos identificar las **entradas** con las **transiciones** (entre un estado y otro, o entre el mismo estado) y las **salidas** con los **círculos**. Se dice que una tabla de transición de estados es la representación de un diagrama de transición de estados, donde la tabla se encuentra compuesta por: • **Filas**, identificando los estados que hay en el diagrama. • **Columnas**, identificando las combinaciones de la entrada.



Máquina de Moore

En una máquina de Moore la salida depende del estado. Debido a que el estado únicamente varía en función de la señal de reloj (de forma síncrona) las salidas se realizarán de la misma manera. Estas son las **funciones** por las que se rige la **máquina de Moore**:

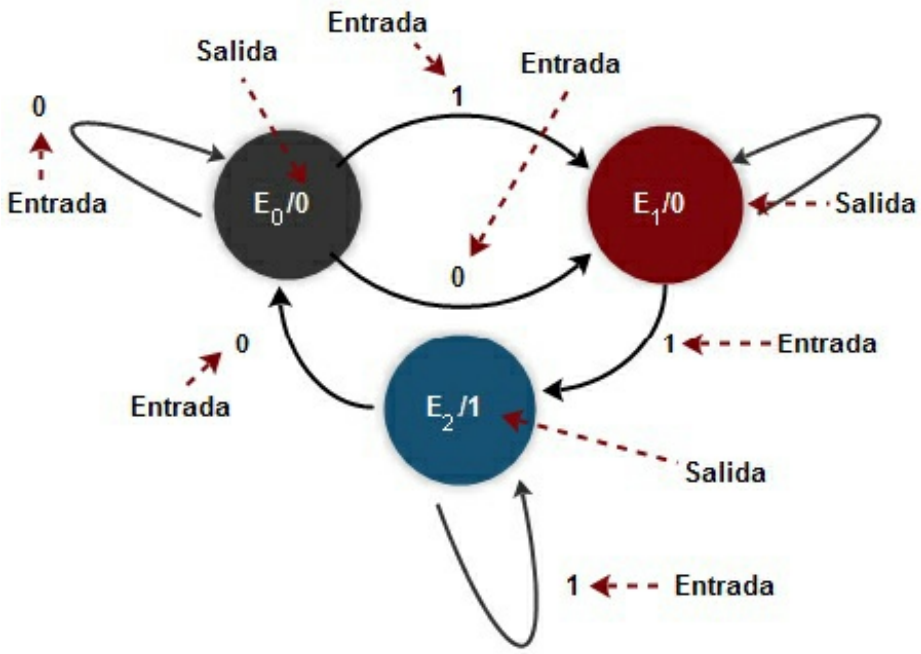


Observemos el diagrama y supongamos que la máquina se encuentra en el estado E_1:

- Si la entrada es X_A la máquina avanza hasta el estado E_2.
- Si la entrada es X_B, avanzaremos al estado E_3.

En cada estado las **salidas** tienen un valor determinado Z_1, Z_2 o Z_3 si la máquina se encuentra en el **estado** E_1, E_2 o E_3 respectivamente.

Representación del diagrama de Moore y sus tablas: el reconocedor de secuencias 11



En este **diagrama**, obtendremos como salida 1 si la secuencia es 11. Desde el punto de vista de los **estados**:

E0 → X(t-1) = 0	Representa el estado de "**no hay nada reconocido**", es decir, la última entrada ha sido un 0.
E1 → X(t-1) = 1, X(t-2) = 0	Representa el estado de "**tengo un 1 reconocido**", es decir, la última entrada ha sido un 1, pero la anterior a esta un 0.
E2 → X(t-1) = 1, X(t-2) = 1	**Tengo dos 1 reconocidos**, es decir, las dos últimas entradas han sido un 1.

Tablas de transición para Moore

Como ya hemos comentado, las **tablas** de transición son una **representación organizada** de la información contenida en un diagrama de transición de estados. Las tablas necesarias para representar toda la información que aparecen en los diagramas son las siguientes:

- Tabla de codificación de estados.
- Tabla de transiciones.
    - Equivalencia con estados codificados.
- Tabla de salidas.

Tabla de codificación de estados

En este caso, al tener más de dos estados, tres en nuestro caso, se necesitan **dos bits** para codificar el estado:

	Codificación	
Estado	Q1	Q0
E0	0	0
E1	0	1
E2	1	0

Tabla de transiciones

Codifica **cómo evoluciona** la máquina de estado según las entradas:

Estado actual	Entradas	Estado futuro
E(t)	X(t)	E(t+1)
E0	0	E0
E0	1	E1
E1	0	E0
E1	1	E2
E2	0	E0
E2	1	E2

Equivalencia con estados codificados

Como se puede ver a continuación, **no importan** los estados:

Estado actual		Entradas	Estado futuro	
Q1(t)	Q0(t)	X(t)	Q1(t+1)	Q0(t+1)
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	0
1	1	0	d	d
1	1	1	d	d

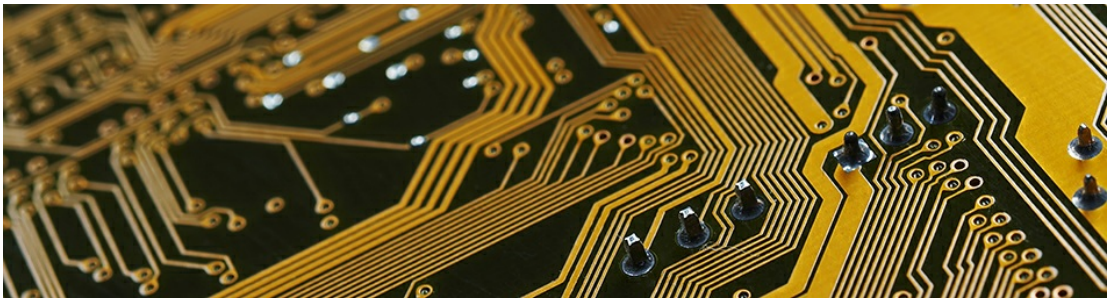
Tabla de salidas

Estado	Codificación de estado		Salida
E	Q1	Q0	Z
E0	0	0	0
E1	0	1	0
E2	1	0	1

En todas estas tablas, que no son más que tablas de verdad, solo queda por **identificar** las **entradas** y las **salidas**.

- Entradas: estado actual [$Q_1(t)$, $Q_0(t)$] y entradas [X(t)].

- Salidas: estado futuro [$Q_1(t+1)$, $Q_0(t+1)$] y salidas [$Z(t)$].



Transformación de las tablas de transición de estados a los flip-flop

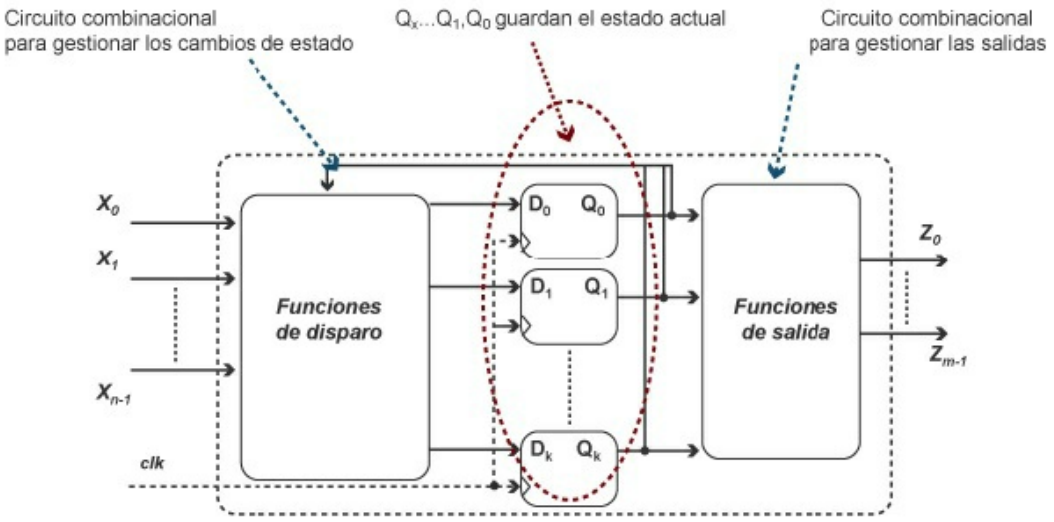
El circuito combinacional resultante se **realimenta**; es decir, en un cambio de estado con la entrada al sistema de una nueva señal de "entrada", el circuito avanza calculando el estado futuro. Posteriormente, este estado futuro pasa a ser **estado actual**.

Esto se realiza mediante una **realimentación** de las **salidas** con las **entradas**. Para que esta realimentación sea síncrona se necesitan intercalar en el circuito biestables que hacen que la transición del estado futuro al estado presente sea coordinada.

Gráfico ¿Cuál sería el esquema general de una máquina secuencial?

¿Cuál sería el esquema general de una máquina secuencial?

El esquema general de una máquina secuencial, haciendo el **sincronismo con biestables D** sería el siguiente:



Ejemplo del reconocedor de secuencia 11 con biestables D

Una vez determinada la tabla de transiciones, y con todos los elementos codificados en binario, solo hay que **codificar** las **funciones** de **salida** con respecto a las de **entrada**:

- Función de salda: $Z(t)=f(Q_1(t),Q_0(t))$.
- Función de transición bit1: $Q_1(t+1)=f(Q_1(t),Q_0(t),X(t))$.

- Función de transición bit0: $Q_0(t+1)=f(Q_1(t),Q_0(t),X(t))$.

Como las salidas del sistema se van a almacenar temporalmente en biestables D, podemos interpretar que la salida de nuestro sistema no es $Q_x(t+1)$, sino D_x. Así $Q_1(t+1)=D1$ y $Q_0(t+1)=D_0$

Como **resultado final** obtenemos las funciones según la tabla:

Entradas			Salidas	
Estado actual: E(t)		Entrada	Estado futuro: E(t+1)	
Q1	Q0	X	D1	D0
0	0	0	0	0
0	0	1	0	**1**
0	1	0	0	0
0	1	1	**1**	0
1	0	0	0	**1**
1	0	1	**1**	0

De la tabla anterior se obtiene la siguiente función:

$$D_1 = x \cdot (Q_0 + Q_1),\ D_0 = x \cdot \bar{Q}_0 \cdot \bar{Q}_1$$

Entradas		Salidas
Estado actual: E(t)		
Q1	Q0	Z
0	0	0
0	1	0
1	0	**1**

De esta segunda tabla se obtiene esta otra función:

$$Z = Q_1$$

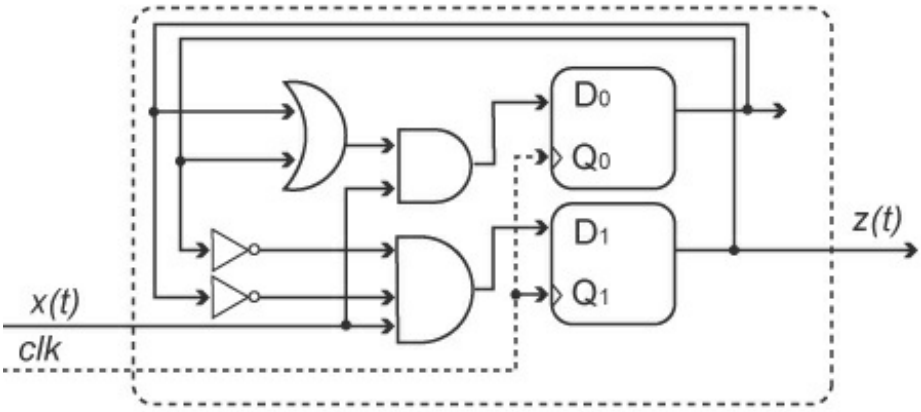
Por último, únicamente nos quedaría implementar el **circuito lógico** para la tabla que anteriormente habíamos diseñado.

Quedaría implementado teniendo como entradas X(t) y la señal de reloj y como salida Z(t).

Gráfico ¿Cómo se realiza la implementación de un circuito lógico?

¿Cómo se realiza la implementación de un circuito lógico?

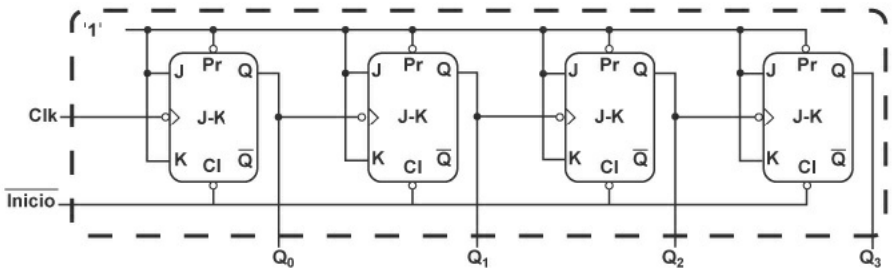
A continuación, se puede ver el **proceso detallado** de implementación de un circuito lógico:



Circuitos básicos: contadores

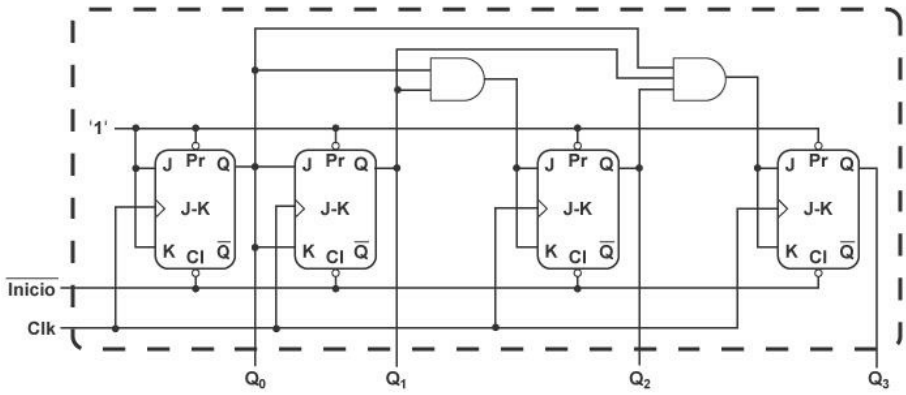
Los circuitos contadores únicamente se van a encargar de realizar la **cuenta del número de pulsos** recibidos en la entrada de reloj.
Este tipo de circuitos se implementa de una manera más sencilla haciendo uso de los **biestables de tipo T**. Dichos biestables los organizaremos en cadena uno detrás de otro.

Se presentan, a continuación, los distintos **tipos** de contadores:

Contador asíncrono	En el contador asíncrono la entrada será la señal de reloj del primer biestable, mientras que la salida de cada uno de los biestables que componen el contador será la entrada de la señal de reloj de cada uno de los siguientes. A continuación, podemos observar el esquema de funcionamiento de un **contador asíncrono de manera ascendente**: 

Contador síncrono

En este caso, la **señal de reloj** será **distribuida** en cada uno de los biestables que componen el contador para que el **sincronismo** sea **efectivo**. Según el diagrama que podemos observar a continuación, la entrada del contador síncrono es la señal de reloj de todos los biestables que lo componen, donde su lógica de conteo se implementa mediante puertas lógicas.



Podemos observar que la entrada de reloj es la misma para todos los biestables J-K. La entrada de 'Inicio' llega a cada uno de los biestables y en la línea '1' el funcionamiento es de los biestables de tipo T.

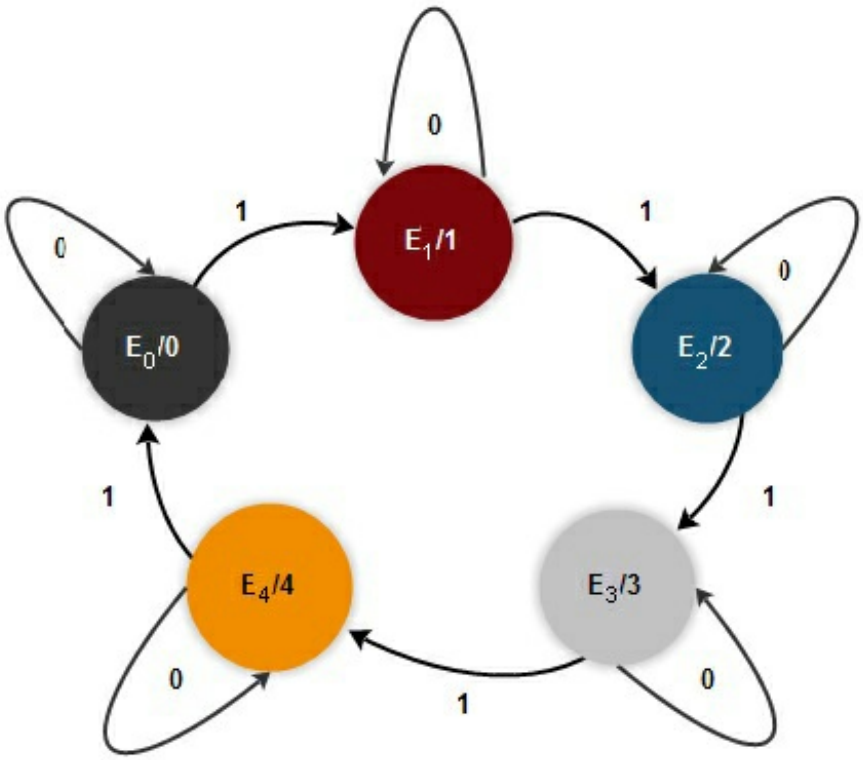
Por otro lado, podemos observar la siguiente lógica de conteo, donde:

$$T_0 = 1;\ T_1 = Q_0;\ T_2 = Q_0 \cdot Q_1;\ T_3 = Q_0 \cdot Q_1 \cdot Q_2$$

Contador en módulo P mediante máquina de Moore

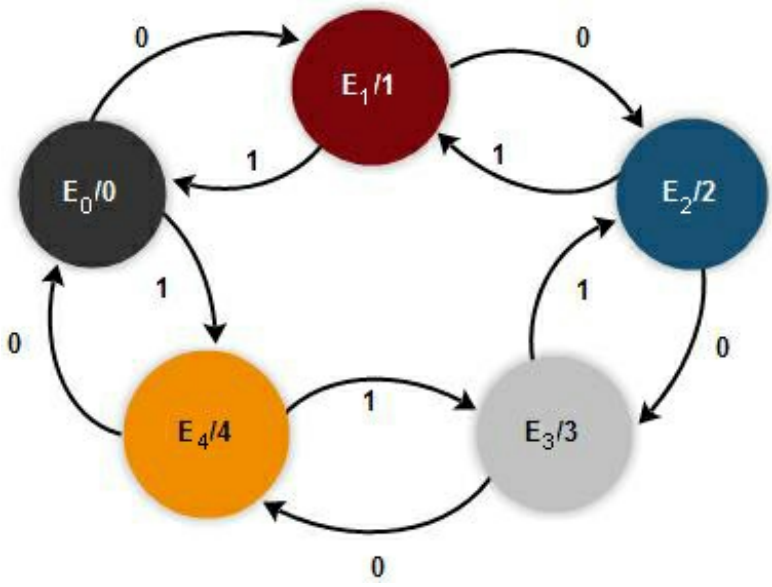
El comportamiento es el mismo que en el caso del contador anterior, con la única diferencia de que lo analizamos mediante una **máquina de Moore**, como se puede observar en el siguiente diagrama de transición de estados.

A continuación, observamos el ejemplo del **contador** de unos en **módulo 5** por máquina de Moore.



En este tipo de circuitos, la salida no son dos posibles estados, sino 5 para este ejemplo. Por lo tanto, al igual que en el caso del reconocedor de secuencias, en el que codificábamos en binario el estado, aquí también hay que **codificar** en binario la **salida**.
Ahora nuestro sistema ya no solo tiene un bit de salida, sino ($2^n >= 5$; n=3) 3 bits.

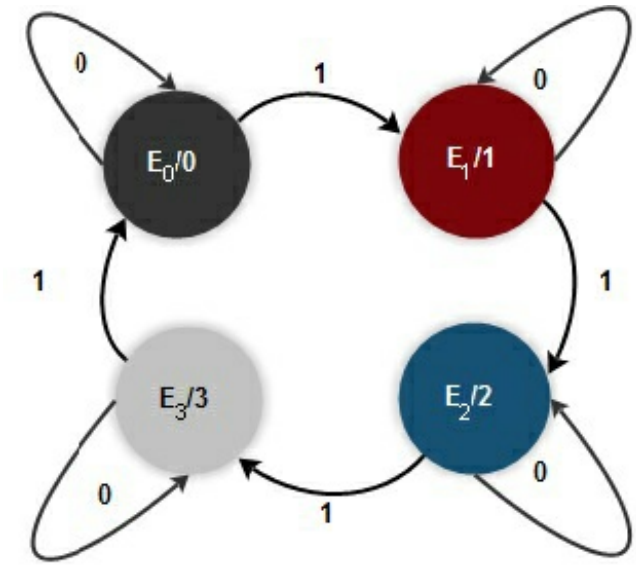
Contador reversible en módulo P mediante máquina de Moore



En este esquema, cada vez que la entrada es un **0** se **incrementa** el contador y si es un **1** se **decrementa** el contador.

Ejemplo completo de contador de unos en módulo 4

A continuación, vamos a realizar el estudio de un caso práctico de contador de unos en módulo 4 mediante la utilización de biestables J-K a través de la máquina de Moore.



En detalle

Tabla de codificación de estados

Tabla de codificación de estados

E	Q_1	Q_0
E_0	0	0
E_1	0	1
E_2	1	0
E_3	1	1

En detalle

Tabla de transiciones

Tabla de transiciones

Estado actual		Entrada	Estado futuro	
Q1	Q0	X	Q1(t+1)	Q0(t+1)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

Dado que lo queremos implementar con biestables J-K, necesitamos calcular la **combinación de entradas** a los biestables para conseguir que el biestable J-K almacene el valor Qx(t+1) que indica la tabla anterior.

Para ello, recordemos la **tabla de funcionamiento del J-K**: {EQ \x \to(Q)}

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Si le damos la vuelta a esta tabla, analizando lo que queremos guardar y teniendo en cuenta el valor guardado actualmente en el biestable, vemos que:

Caso 1	Tenemos $Q_x = 0$ y queremos mantener $Q_x(t+1) = 0$. Por tanto, hay dos opciones: • **Mantener el valor**: por tanto, J=0 y K=0. • **Forzar un 0**: J=0, K=1. En consecuencia, si el valor actual del biestable es 0 y queremos mantener un 0, es suficiente con asegurarnos que J=0, independientemente del valor de K (K=no importa=d).
Caso 2	Tenemos $Q_x = 0$ y queremos cambiar $Q_x(t+1) = 1$. Por tanto, hay dos opciones: • **Intercambiar el valor**: por tanto, J=1 y K=1. • **Forzar un 1**: J=1, K=0. En consecuencia, si el valor actual del biestable es 0 y queremos almacenar un 1, es suficiente con asegurarnos que J=1, independientemente del valor de K (K=no importa=d).

Caso 3	Tenemos $Q_x = 1$ y queremos cambiar $Q_x(t+1) = 0$. Por tanto, hay dos opciones: • **Intercambiar el valor**: por tanto, J=1 y K=1. • **Forzar un 1**: J=0, K=1. En consecuencia, si el valor actual del biestable es 1 y queremos almacenar un 0, es suficiente con asegurarnos que K=1, independientemente del valor de J (J=no importa=d).
Caso 4	Tenemos $Q_x = 1$ y queremos mantener $Q_x(t+1) = 1$. Por tanto, hay dos opciones: • **Mantener el valor**: por tanto, J=0 y K=0. • **Forzar un 1**: J=1, K=0. En consecuencia, si el valor actual del biestable es 1 y queremos mantener un 1, es suficiente con asegurarnos que K=0, independientemente del valor de J (J=no importa=d).

Aplicando estas **definiciones**, podemos transformar la anterior tabla de verdad en una que codifique el valor de J-K.

En detalle

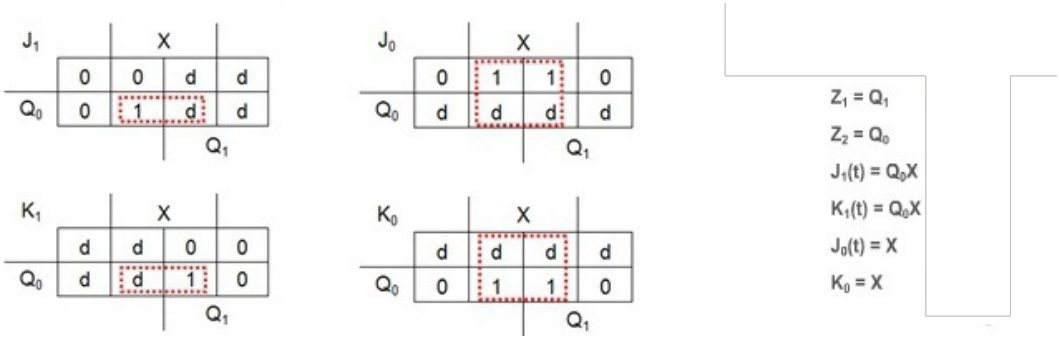
Tabla de verdad con los valores de J - K codificados

Tabla de verdad con los valores de J-K codificados

Estado actual		Entrada	Estado futuro					
Q1	Q0	X	Q1(t+1)	Q0(t+1)	J1	K1	J0	K0
0	0	0	0	0	0	d	0	d
0	0	1	0	1	0	d	1	d
0	1	0	0	1	0	d	d	0
0	1	1	1	0	1	d	d	1
1	0	0	1	0	d	0	0	d
1	0	1	1	1	d	0	1	d
1	1	0	1	1	d	0	d	0
1	1	1	0	0	d	1	d	1

A partir de este punto, solo falta sacar las funciones de correspondencia para J1, K1, J0, K0 .

Funciones de correspondencia para J_1, K_1, J_0, K_0



Resumen

A lo largo de este recurso, se han tratado los siguientes puntos sobre contadores y secuenciadores:

- En los sistemas combinacionales, dependiendo de las secuencias que se produzcan en las entradas serán las secuencias que se produzcan en la salida, mientras que en los secuenciales además depende del estado.
- Decimos que los **autómatas** de estados finitos son sistemas **secuenciales síncronos**.
- Hay que recordar que hay dos maneras de implementar circuitos combinaciones, por Mealy o por Moore. Por simplicidad, solo se ha visto Moore, recordando también que son compatibles.
- No hay ningún circuito que no se puede resolver por Mealy y no por Moore.
- La única diferencia entre ellos es que en Mealy las salidas dependen del estado y las entradas, mientras que en Moore, las salidas dependen solo del estado.
- Un diagrama de estados es semejante a un grafo de estados, donde los **estados** son **círculos** y las **transiciones** entre estados son **flechas**.
- Una **tabla de transición** de estados es la representación de un diagrama de transición de estados.
- Los casos típicos donde se usan **máquinas secuencias**, son aquellos circuitos que necesitan memorizar la secuencia de entrada pasada, como reconocedores de secuencia, contadores módulo, sistemas de seguridad con teclado, etc.

¡Enhorabuena! Has finalizado con éxito.

