Tema 7. Circuitos secuenciales

Circuitos Electrónicos Digitales
E.T.S.I. Informática
Universidad de Sevilla

Jorge Juan <jjchico@dte.us.es>

17/11/2025 19:26:50

Contenidos

- Introducción
- Biestables
- Máquinas de estados finitos y circuitos secuenciales síncronos (CSS)
- Análisis de CSS
- Diseño de CSS

Introducción

- Muchos aplicaciones prácticas requieren sistemas cuya salida en cada instante depende no sólo del valor de las entradas en ese momento, sino también de lo que valieron en instantes previos.

- Estos sistemas no pueden implementarse usando únicamente circuitos combinacionales: Se necesitan componentes que “recuerden” cierta información sobre la historia de las entradas (estado).

- Para almacenar el estado de un sistema es necesario un nuevo tipo de componente: el biestable.

Biestables

- Los biestables son circuitos electrónicos con dos estados estables, normalmente denominados 0 y 1.
- Son el elemento básico de los circuitos secuenciales.
- Poseen como mínimo una salida (*salida de estado*, *q*) que permite conocer el estado en el que se encuentra el biestable.
- Poseen una o más entradas que hacen que sea posible conmutar entre los dos estados estables denominadas *señales de excitación*.
- Un circuito con *n* biestables puede tener hasta 2^n estados distintos.

Biestable SR asíncrono

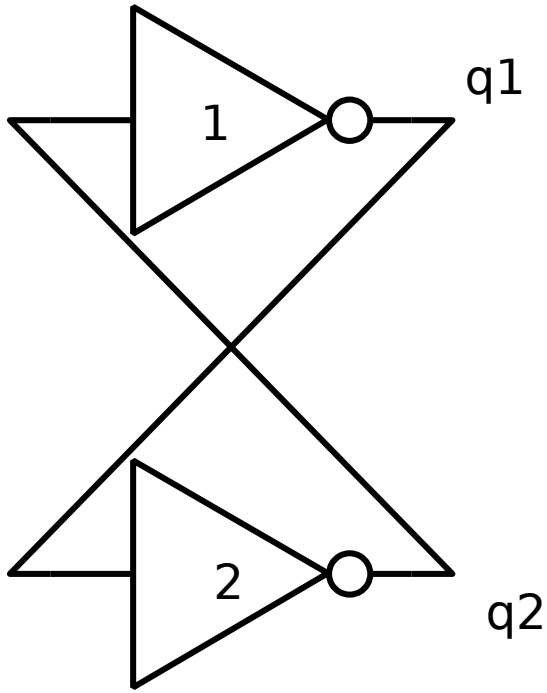
La capacidad de almacenar información se obtiene de la "realimentación" de las salidas hacia las entradas: el valor de la salida refuerza el de las entradas y viceversa.

Estados estables:

q1=0, q2=1
q1=1, q2=0

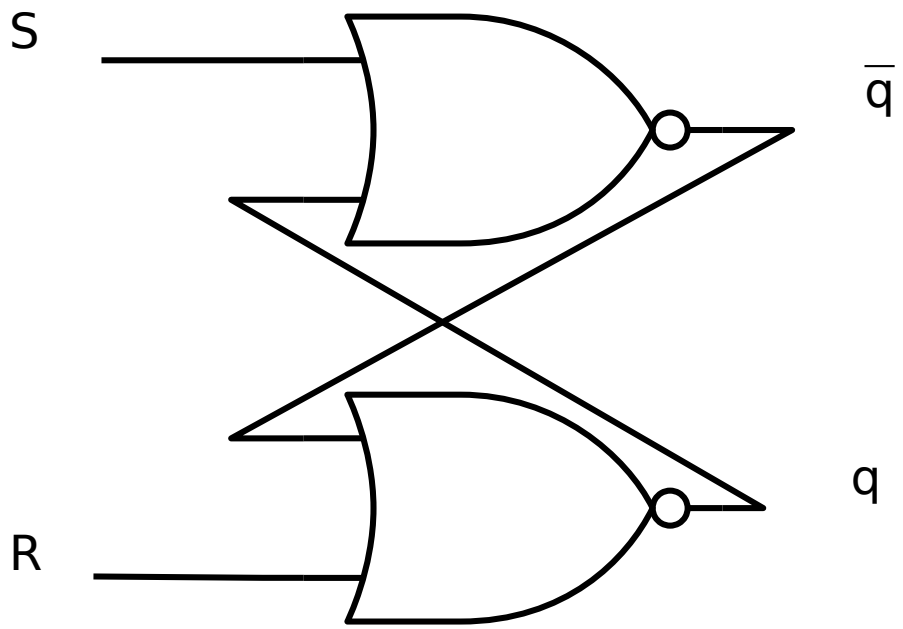
Convenio

$q = q2$
$\overline{q} = q1$



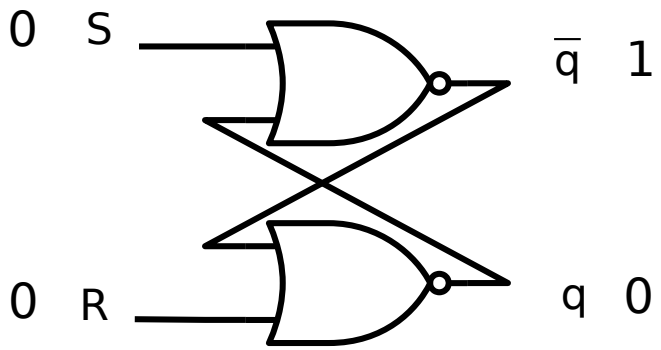
Biestable SR asíncrono

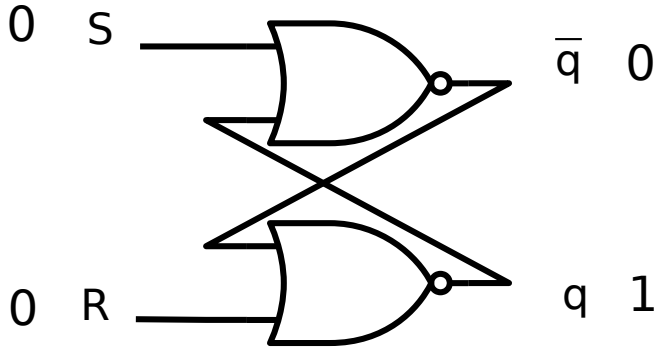
- Necesitamos un biestable que pueda cambiar su estado, para lo que debe tener señales de excitación.
- Las señales de excitación del circuito de la figura se llaman S y R, por lo que este biestable denomina SR (o RS).



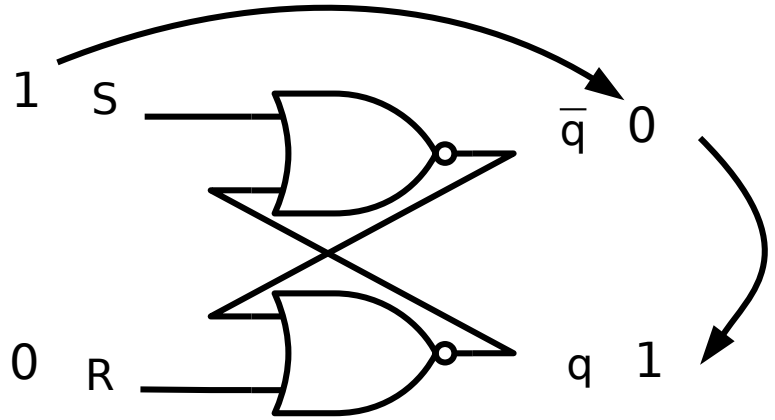
Biestable SR asíncrono

R=S=0 conserva el estado

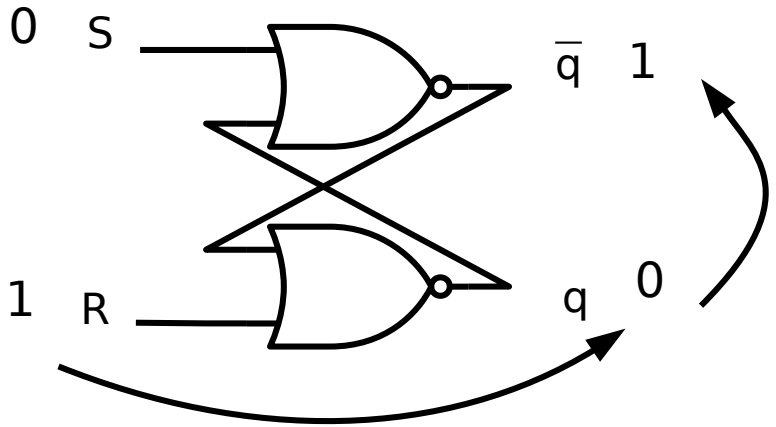




S=1, R=0 cambia a 1 (set)

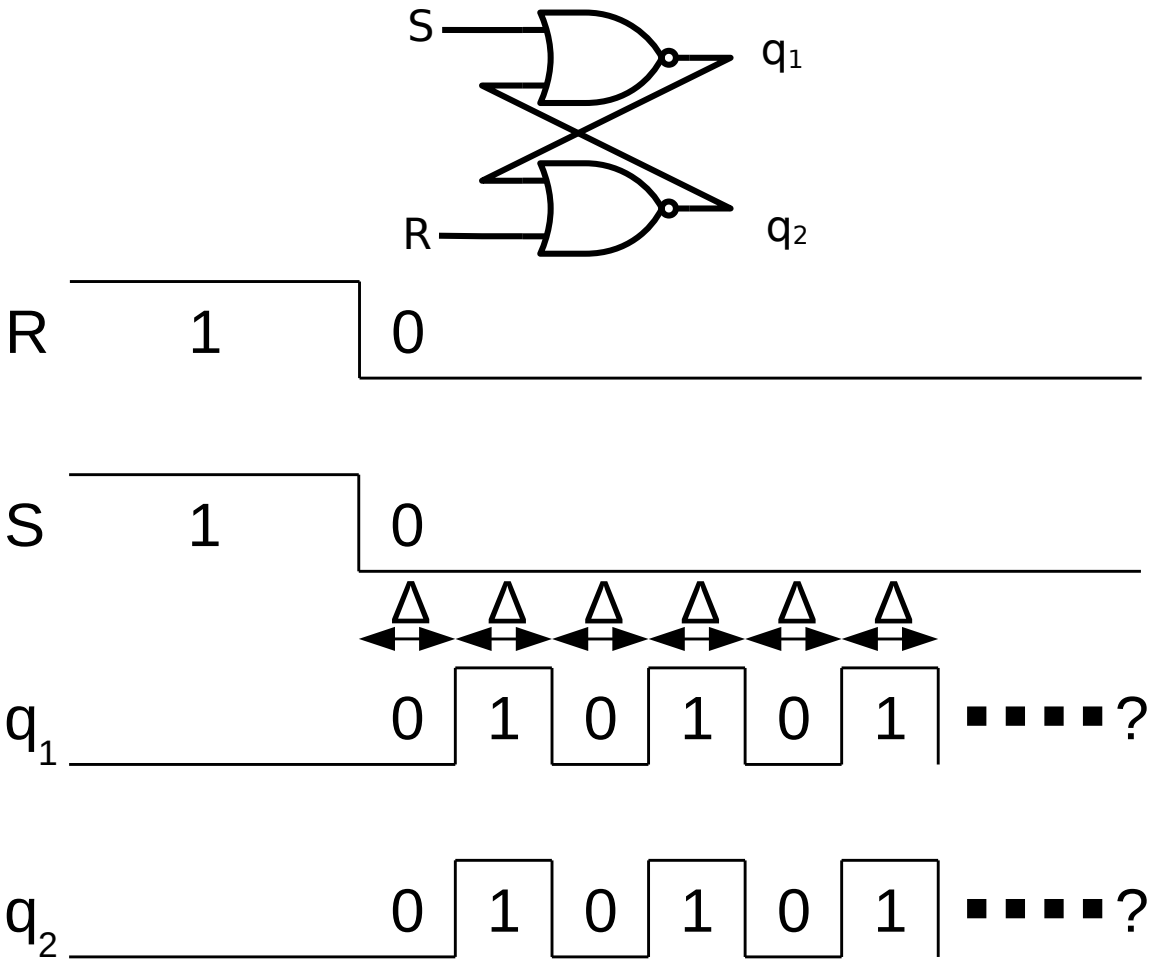


S=0, R=1 cambia a 0 (reset)



Biestable SR asíncrono

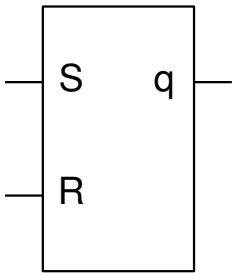
¿Que ocurre si se pasa de R=S=1 a R=S=0?



Esto se denomina *metaestabilidad*.

Biestable SR. Representación formal

Símbolos



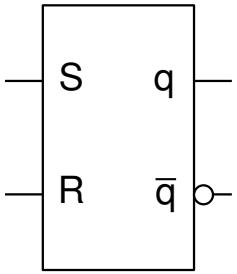


Tabla de estados

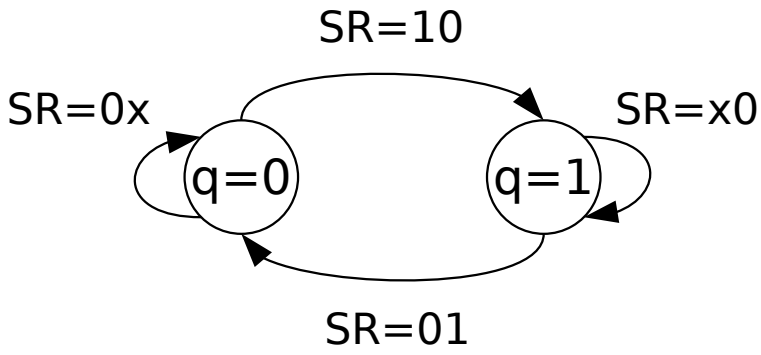
q \ SR	00	01	11	10
0	0	0	-	1
1	1	0	-	1

Q

Tabla de excitación

q → Q	SR
0 → 0	0x
0 → 1	10
1 → 0	01
1 → 1	x0

Diagrama/grafo de estados



Verilog

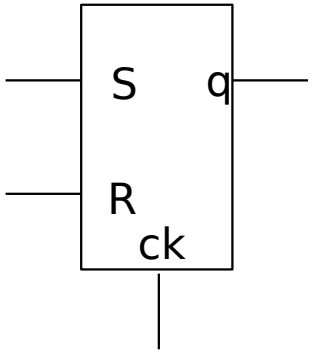
```
module sra(
  input s,
  input r,
  output reg q);

  always @(s, r)
    case ({s, r})
      2'b01: q = 1'b0;
      2'b10: q = 1'b1;
      2'b11: q = 1'bx;
    endcase
endmodule
```

Biestables síncronos

- En circuitos reales con miles (o millones) de biestables es muy útil que todos cambien de estado a la vez, pues simplifica el proceso de diseño.
- En los circuitos secuenciales síncronos los cambios de estado se producen “sincronizados” con al menos una señal de control denominada “señal de reloj” (CLOCK, CLK, CK)
- Tipos de biestables sincronos:
  - Disparados por nivel (latch): los biestables pueden cambiar de estado cuando CK tiene un valor determinado. Se subdividen en disparados por nivel alto (1) o bajo (0).
  - Disparados por flanco (flip-flop): los biestables pueden cambiar de estado cuando CK cambia en un determinado sentido (flanco activo) que puede ser de 0 a 1 (flanco de subida) o de 1 a 0 (flanco de bajada). Que el instante de cambio esté determinado de forma precisa facilita el diseño.

Biestables síncronos

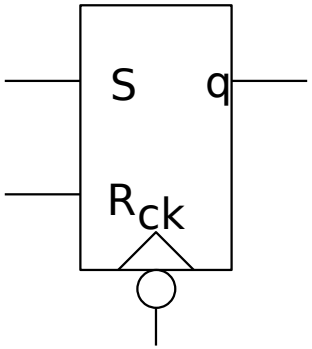
Disparado por nivel




```
module srl(input ck, s, r,output reg
q);
  always @(ck, s, r)
    case ({ck, s, r})
      3'b101: q = 1'b0;
      3'b110: q = 1'b1;
      3'b111: q = 1'bx;
    endcase
endmodule
```

El cambio de estado sólo puede producirse cuando ck=1 (nivel alto) o ck=0 (nivel bajo)

Disparado por flanco

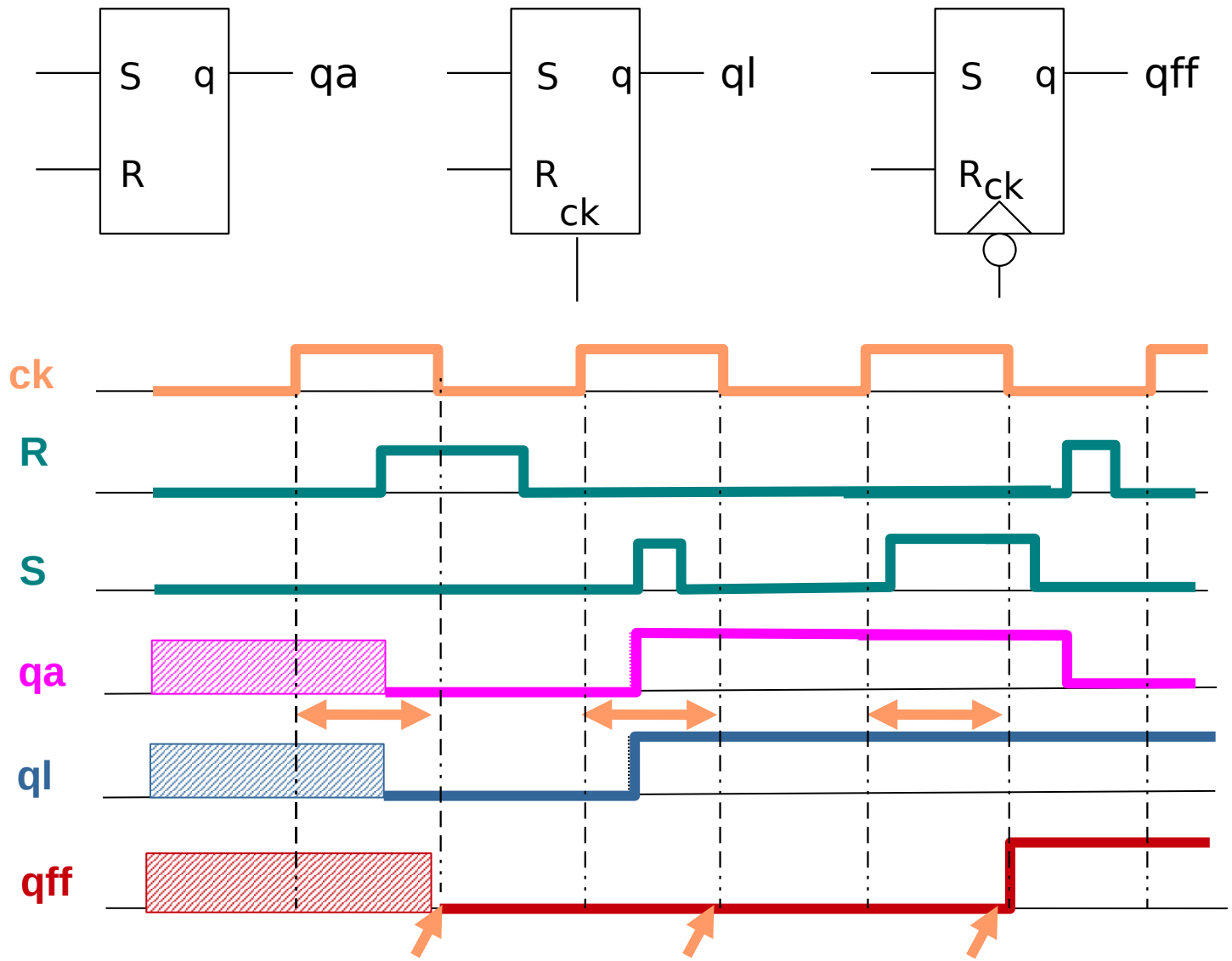



```
module srff(input ck, s, r, output reg q);

  always @(negedge ck)
    case ({s, r})
      2'b01: q = 1'b0;
      2'b10: q = 1'b1;
      2'b11: q = 1'bx;
    endcase
endmodule
```

El cambio de estado sólo puede producirse cuando ck cambia de 1 a 0 (flanco de bajada) o de 0 a 1 (flanco de subida).

Mejor precisión en el cambio de estado.

Biestable asíncrono VS síncronos

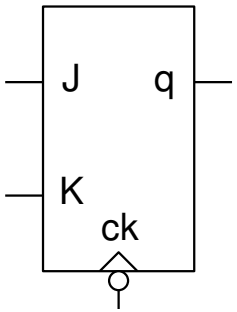


Otros biestables síncronos

- SR
- JK (sólo disparados por flanco)
  - Similar a SR: J~S, K~R
  - Función de cambio de estado (toggle) para J=K=1
- D
  - Una única entrada que indica el próximo estado.
  - Fácil de usar e implementar.
- T (sólo disparados por flanco)
  - Una única entrada que permite complementar el estado.
  - Útil en aplicaciones especiales.

Biestable JK

Símbolos



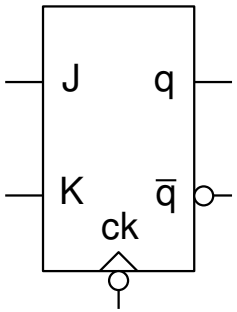


Tabla de estados

q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Q

Diagrama/grafo de estados

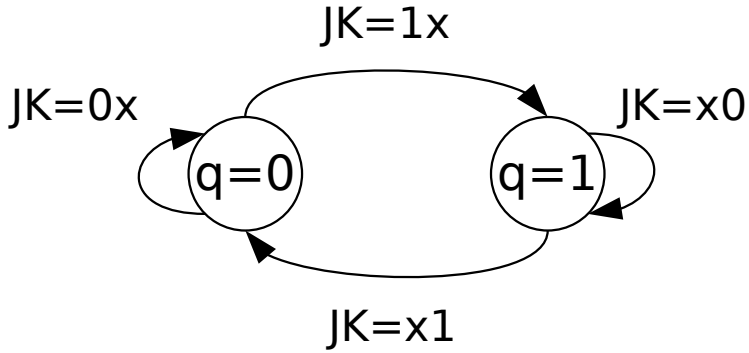


Tabla de excitación

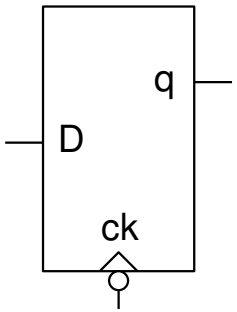
q → Q	JK
0 → 0	0x
0 → 1	1x
1 → 0	x1
1 → 1	x0

Verilog

```
module jkff(
  input ck,
  input j,
  input k,
  output reg q);

  always @(negedge ck)
    case ({j, k})
      2'b01: q = 1'b0;
      2'b10: q = 1'b1;
      2'b11: q = ~q;
    endcase
endmodule
```

Biestable D

Símbolos



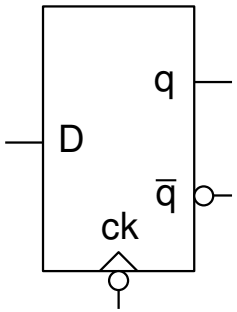


Tabla de estados

q \ D	0	1
0	0	1
1	0	1

Q

Diagrama/grafo de estados

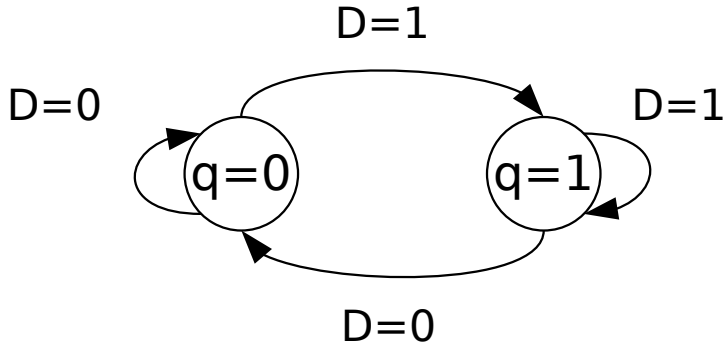


Tabla de excitación

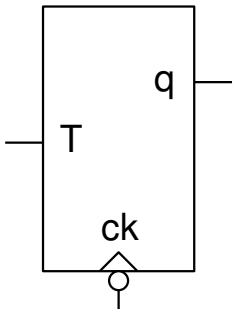
q → Q	D
0 → 0	0
0 → 1	1
1 → 0	0
1 → 1	1

Verilog

```
module dff(
  input ck,
  input d,
  output reg q);

  always @(negedge ck)
    q <= d;

endmodule
```

Biestable T

Símbolos



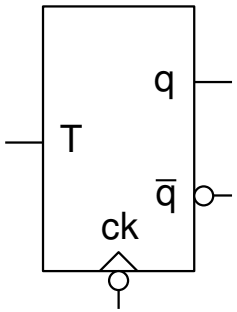


Tabla de estados

q \ T	0	1
0	0	1
1	1	0

Q

Diagrama/grafo de estados

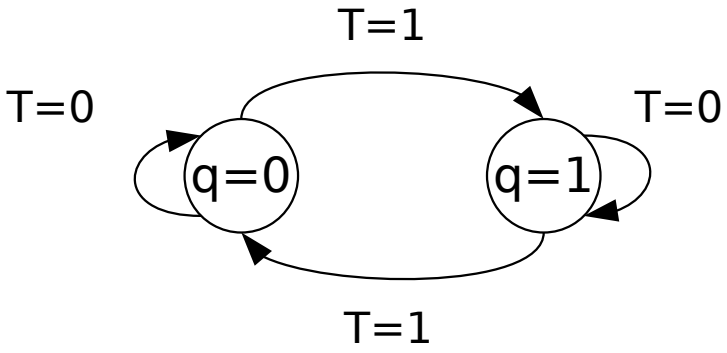


Tabla de excitación

q → Q	T
0 → 0	0
0 → 1	1
1 → 0	1
1 → 1	0

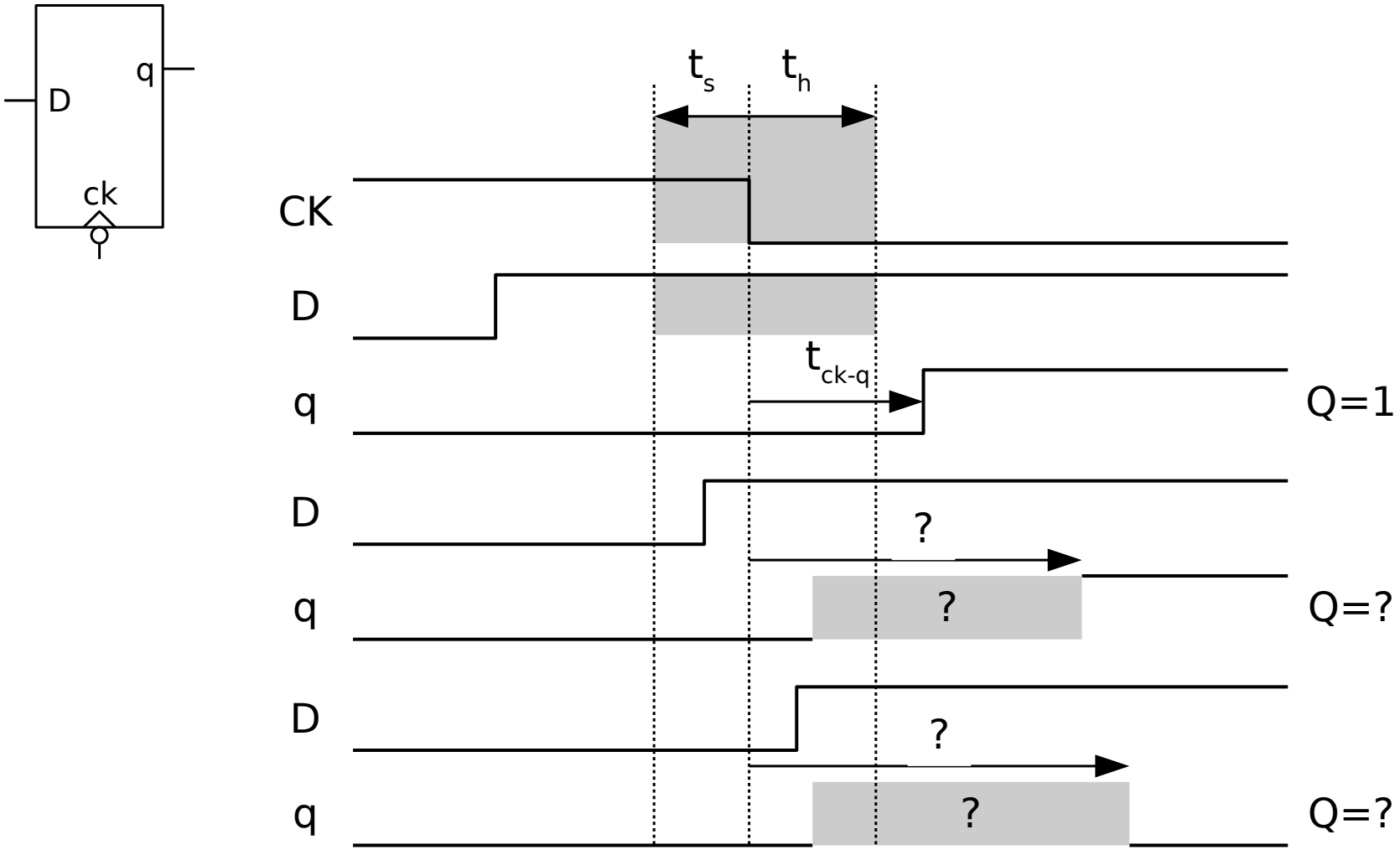
Verilog

```
module tff(
  input ck,
  input t,
  output reg q);

  always @(negedge ck)
    if (t == 1)
      q <= ~q;

endmodule
```

Restricciones temporales

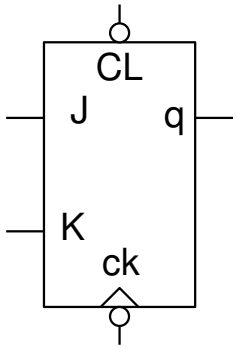
- Igual que las puertas lógicas, los biestables síncronos presentan un retraso de propagación desde el flanco de reloj a la salida: t_{CQ}
- Las entradas de excitación no deben cambiar en las proximidades del flanco activo del reloj en los flip-flops ni en las proximidades del fin del nivel activo en latches. De lo contrario el biestable puede acabar en estado metaestable, lo que es indeseable porque:
  - La salida cambia a un valor no determinado a priori.
  - La salida tarda un tiempo indeterminado en cambiar (t_{CQ} desconocido)
- El periodo de tiempo durante el cual las entradas de excitación no deben cambiar es la ventana de metaestabilidad. Se define en base a dos parámetros:
  - **Tiempo de Set-up** (t_s)
    - Tiempo transcurrido desde el inicio de la ventana de metaestabilidad hasta el flanco activo (flip-flops) o el fin del nivel activo (latches).
  - **Tiempo de Hold time** (t_h)
    - Tiempo transcurrido desde el flanco activo (flip-flops) o el fin del nivel activo (latches) hasta el fin de la ventana de metaestabilidad.

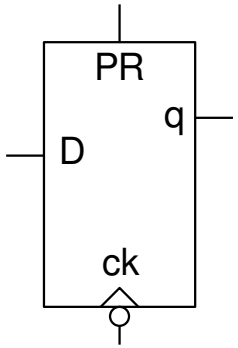
Restricciones temporales. Ejemplo

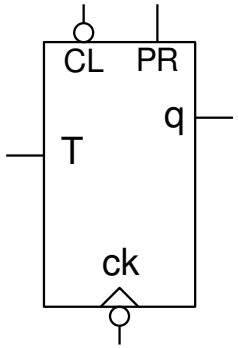


Entradas asíncronas de los biestables

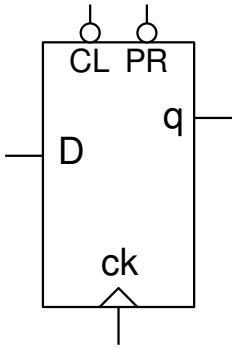
- Permiten forzar un estado determinado de forma sencilla
  - CL (clear): puesta a cero
  - PR (preset): puesta a uno
- Operan inmediatamente cuando se activan:
  - Activas en nivel bajo (0)
  - Activas en nivel alto (1)
- Las entradas asíncronas tienen prioridad sobre las síncronas (J, K, D, T, ...)
- Resuelven el problema de la iniciación en los circuitos digitales complejos
  - millones de biestables
  - necesidad de partir de un estado conocido

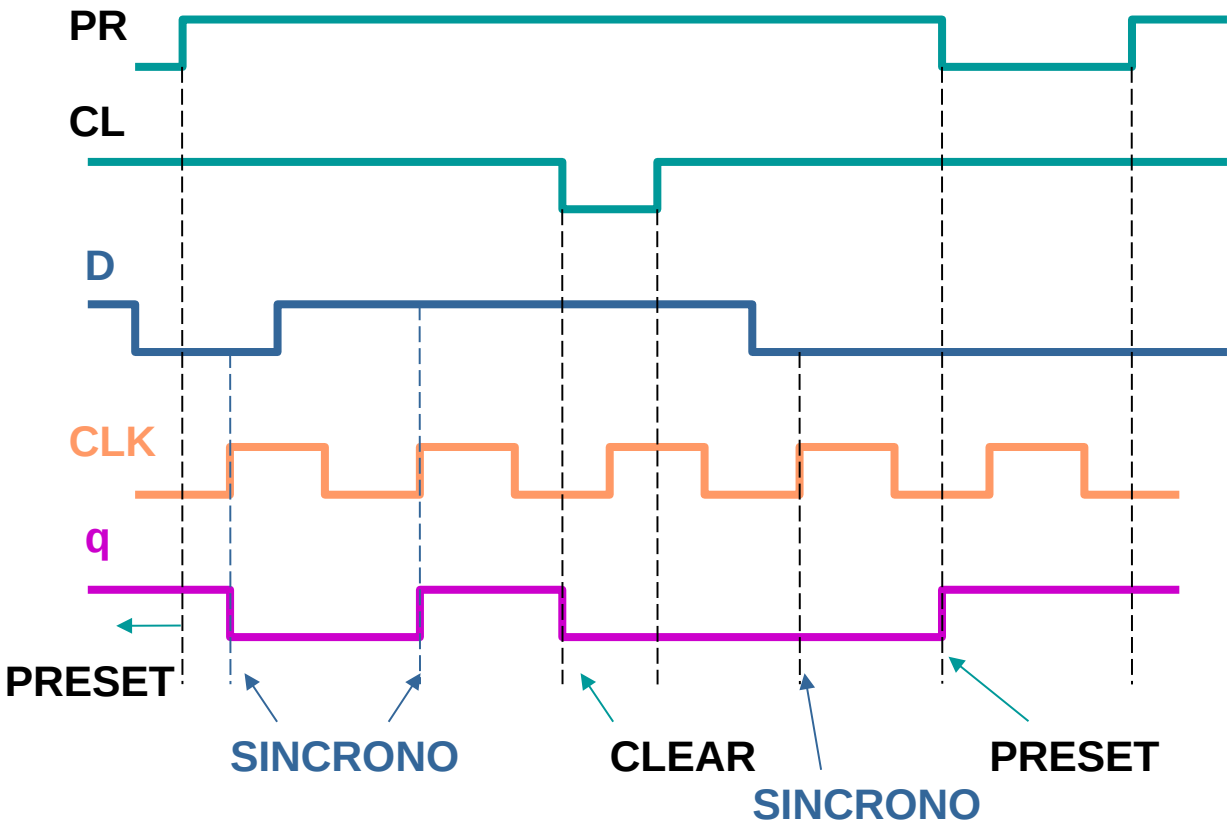






Entradas asíncronas de los biestables





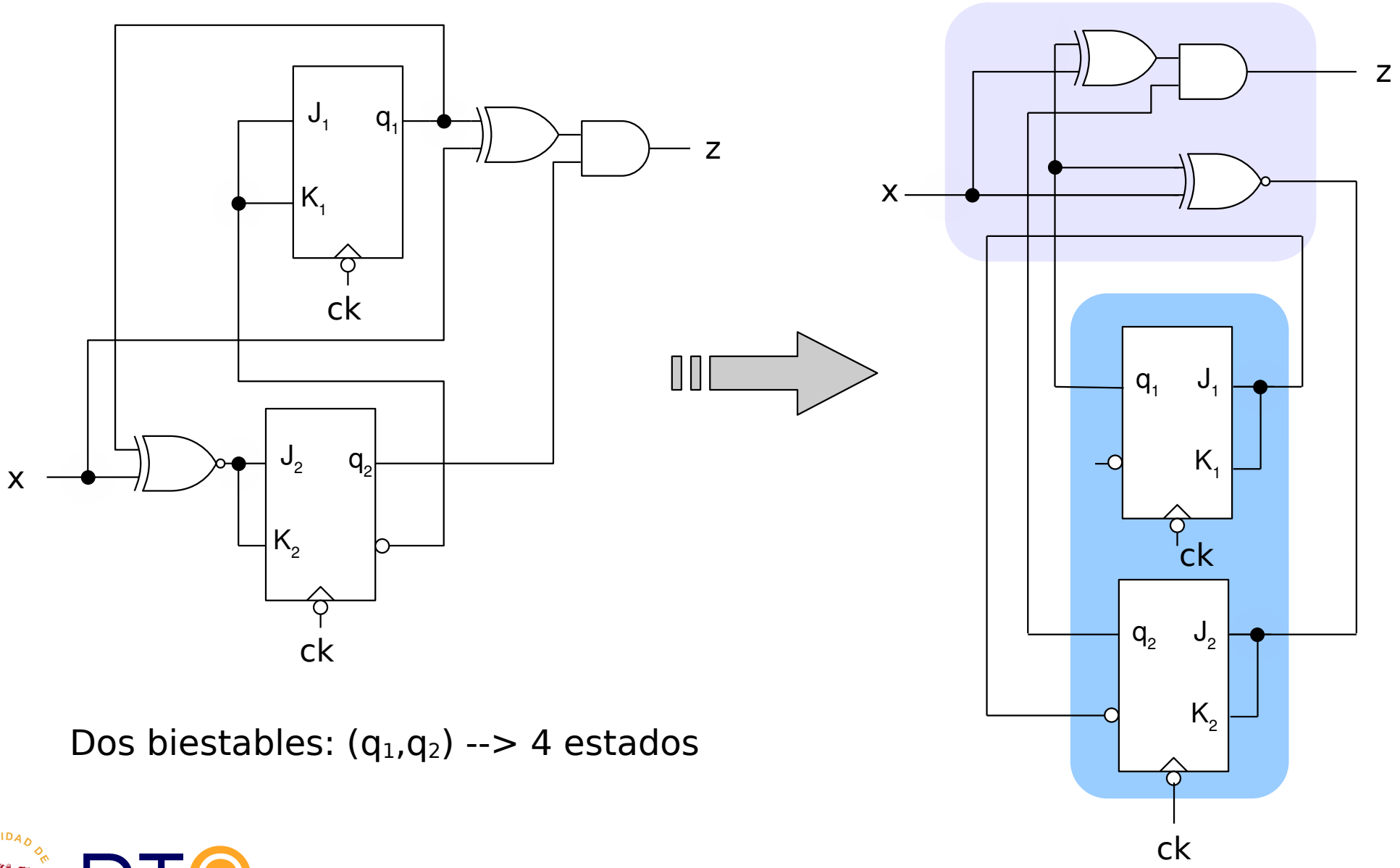
Máquinas de estados finitos y CSS

- Introducción
- Biestables
- Máquinas de estados finitos (FSM) y circuitos secuenciales síncronos (CSS)
  - Ejemplo de circuito secuencial
  - Generalización: modelo de máquina de estado finito
- Análisis de CSS
- Diseño de CSS

Circuitos secuenciales

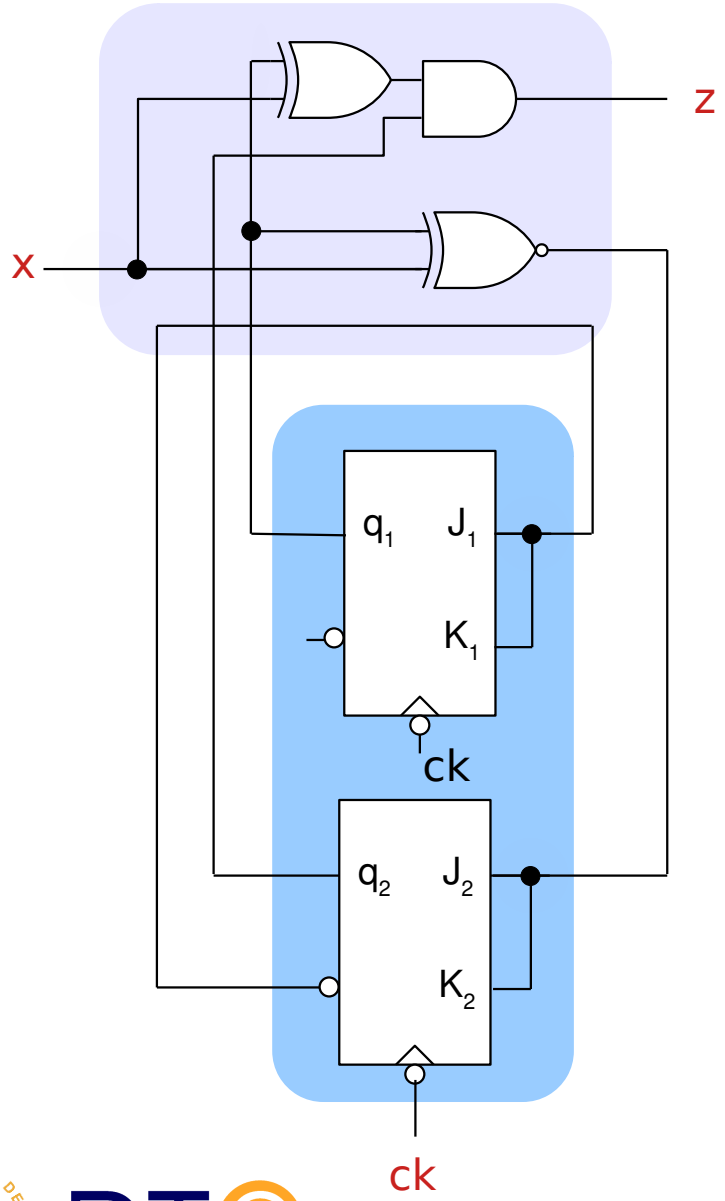
- Los biestables se usan para construir circuitos secuenciales.
- Un circuito secuencial es síncrono si y sólo si todos sus biestables son síncronos y controlados por las mismas señales de reloj.
- Los circuitos secuenciales síncronos tienen las siguientes ventajas:
  - Su proceso de diseño puede sistematizarse y ejecutarse automáticamente por herramientas software.
  - Son robustos frente a variaciones en los componentes y las señales.
- Las señales de reloj suelen ser periódicas. Su frecuencia es un factor esencial del rendimiento del circuito:
  - La frecuencia de reloj determina el número de operaciones por segundo que puede hacer el circuito.
  - La frecuencia máxima de reloj viene determinada por el retraso de los componentes y líneas de interconexión en el circuito así como por el tiempo de setup.
- Los circuitos secuenciales que diseñaremos serán síncronos cuyos biestables serán flip-flops disparados por el mismo flanco de la misma señal de reloj.

Ejemplo de circuito secuencial

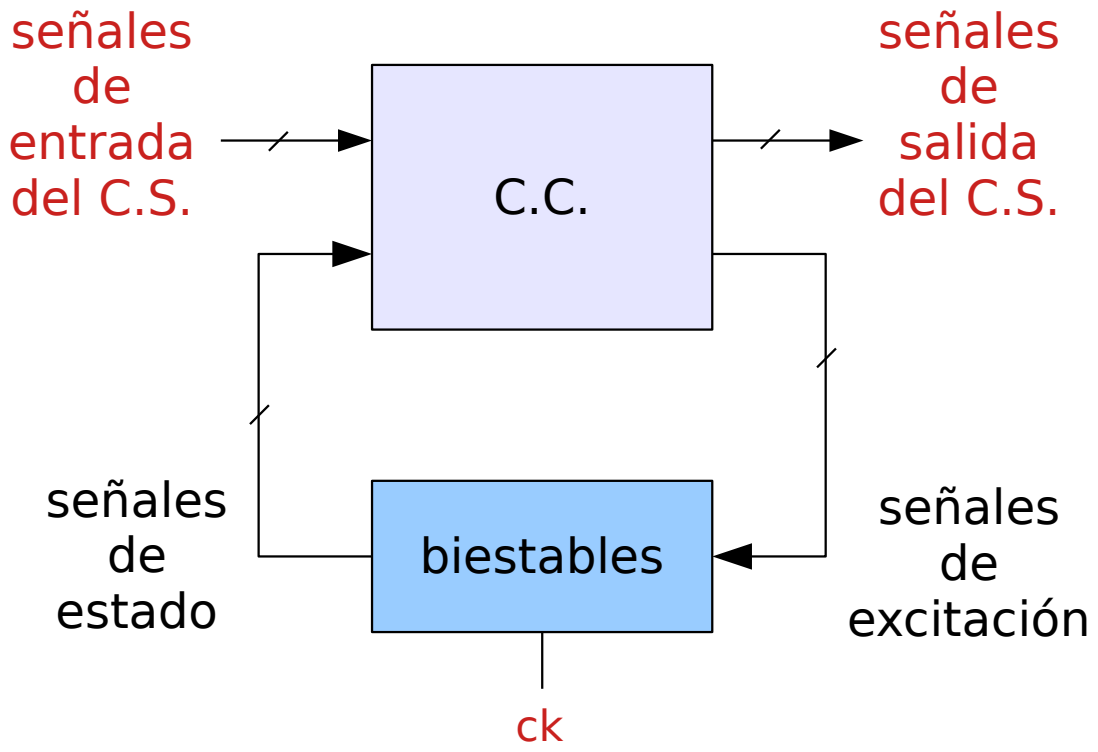


Dos biestables: (q_1,q_2) --> 4 estados

Ejemplo de circuito secuencial

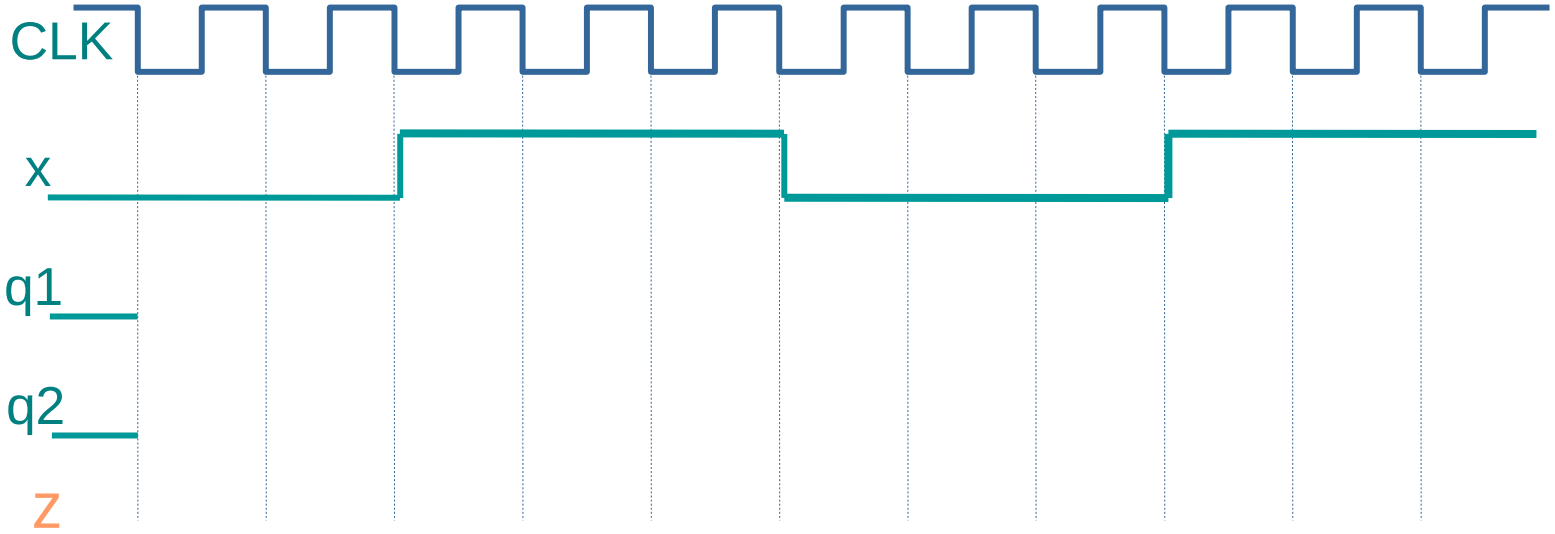


generalización



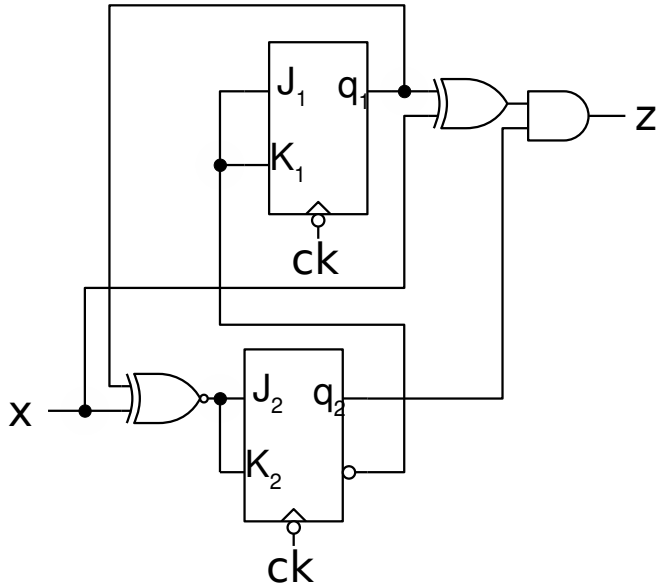
Análisis temporal

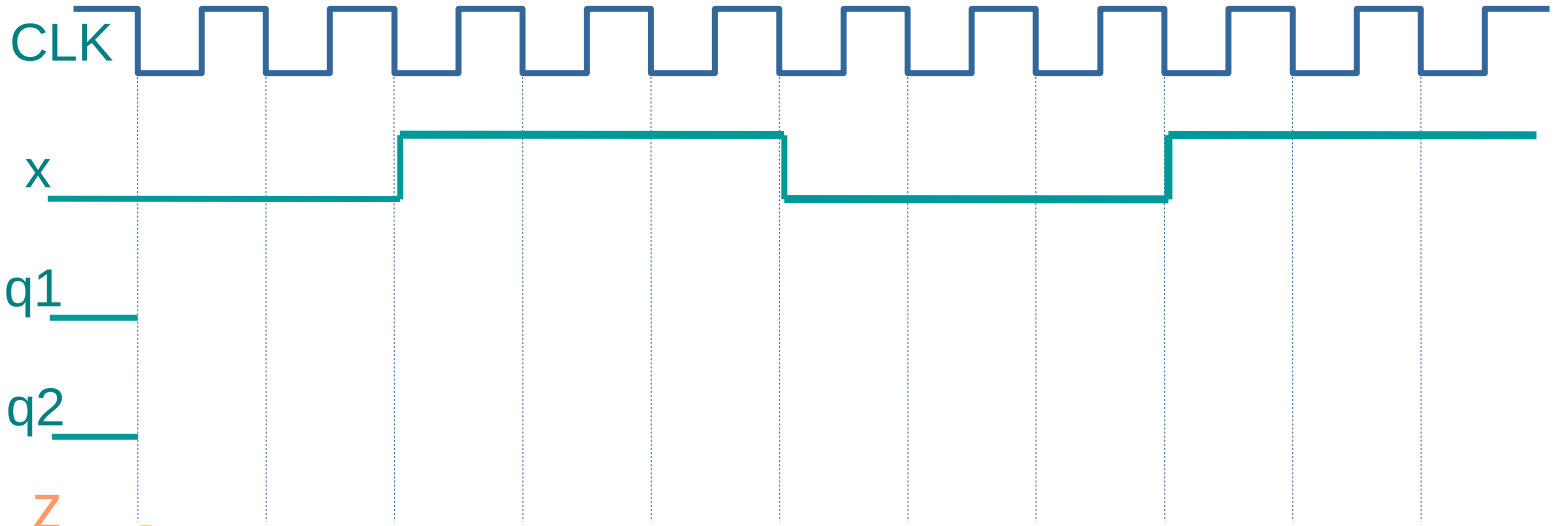
- Objetivo: Obtener el cronograma de las señales de salida del circuito para unas señales de entrada dadas.
- Procedimiento similar al de circuitos combinacionales
  - Parte combinacional: idéntica
  - Biestables (por flanco): se observan sus entradas y se calcula como cambian sus salidas (nuevo estado) a partir de las tablas de transición de los biestables. Los biestables también tienen retrasos (t_{ck-q}).



Análisis temporal

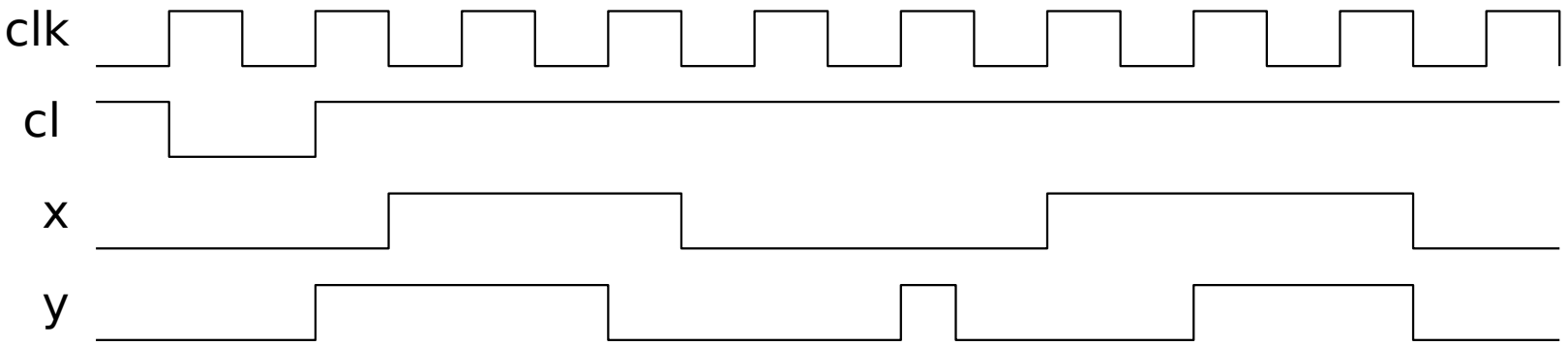
- Un análisis simplificado ignora el retraso de los componentes.

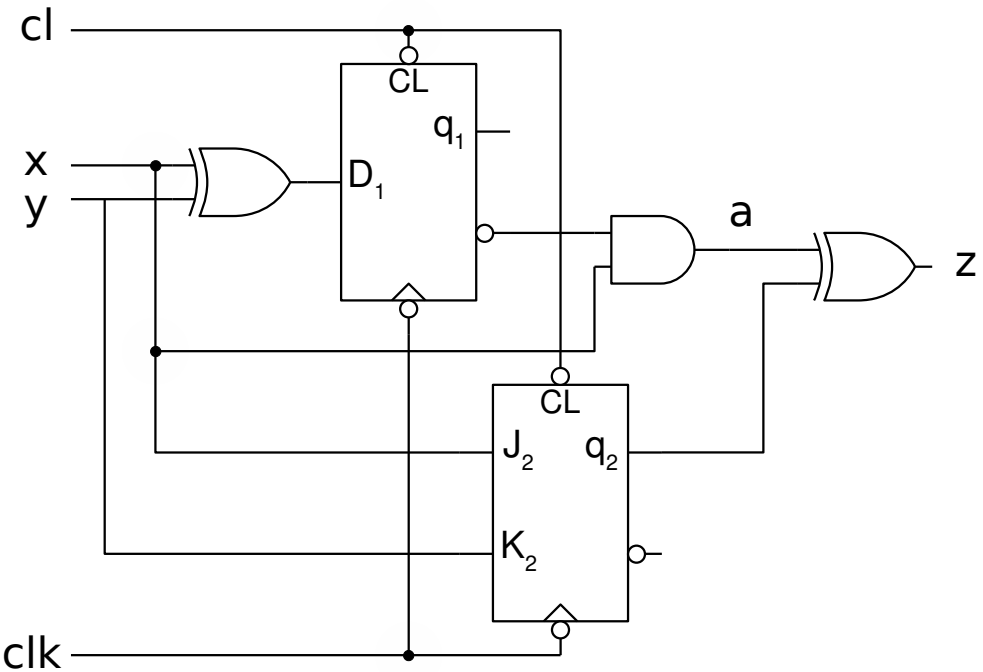




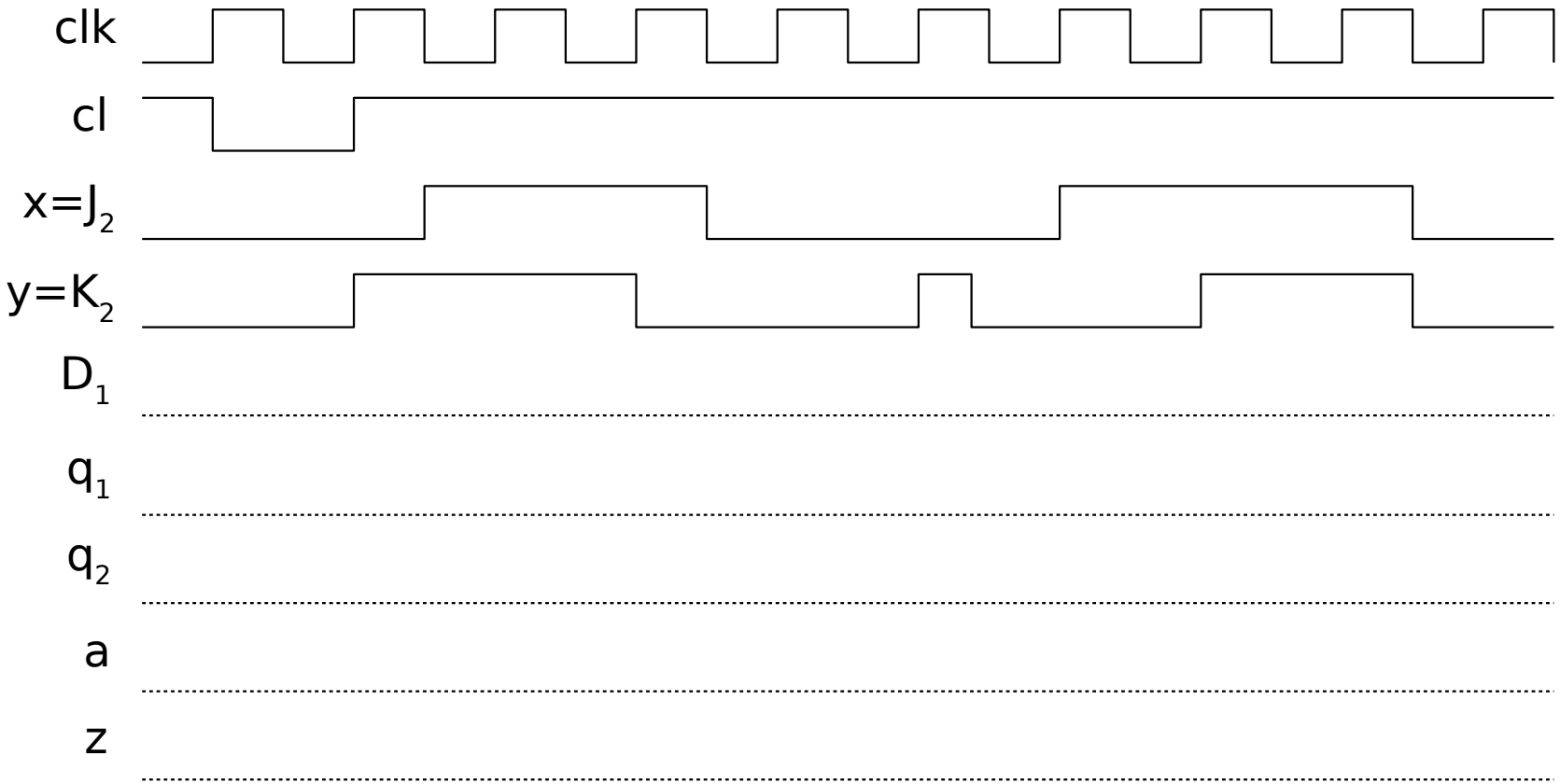
Análisis temporal

- Un análisis más realista tiene en cuenta los retrasos de los componentes. Los biestables también tienen retrasos (t_{ck-q}).





Análisis temporal



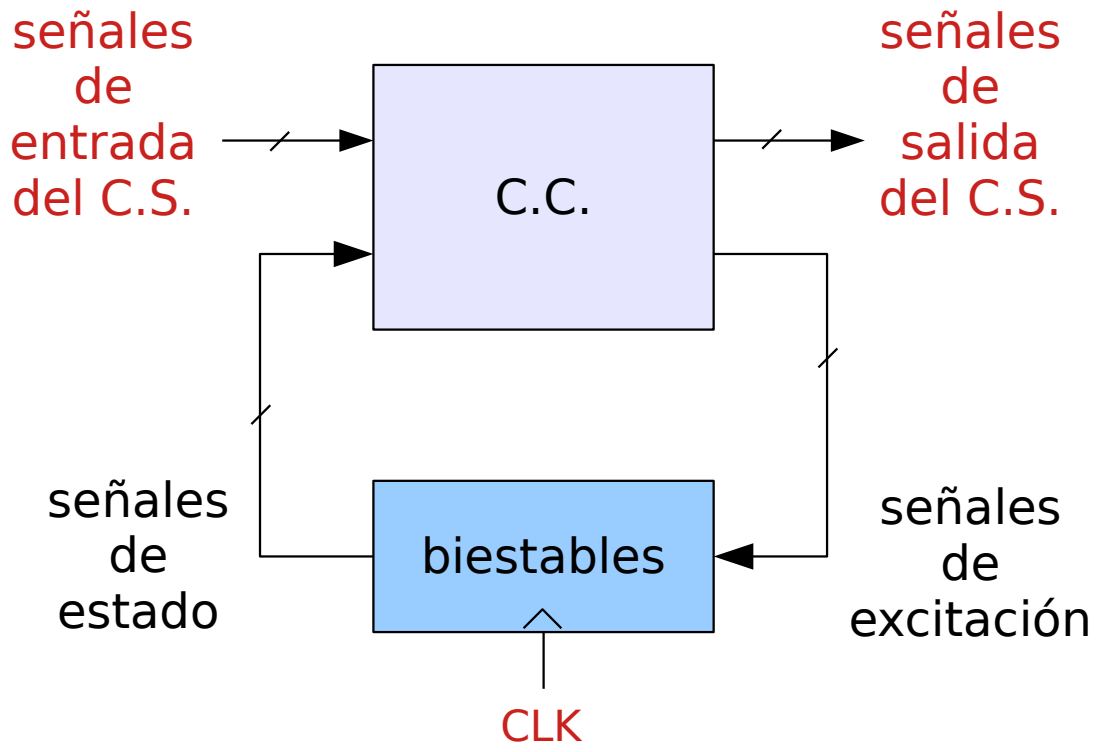
$D_1 = x \oplus y$

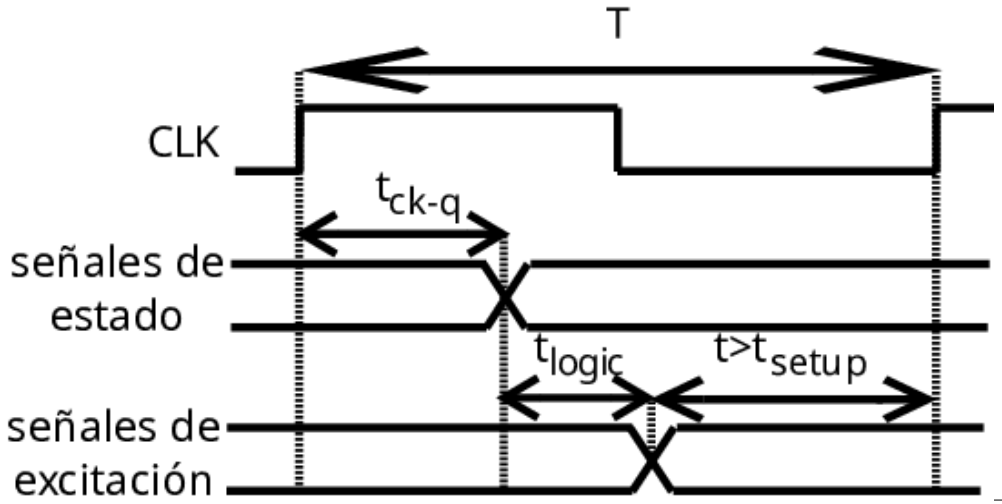
$J_2 = x$; $K_2 = y$

$a = q_1\, x$

$z = a \oplus q_2$

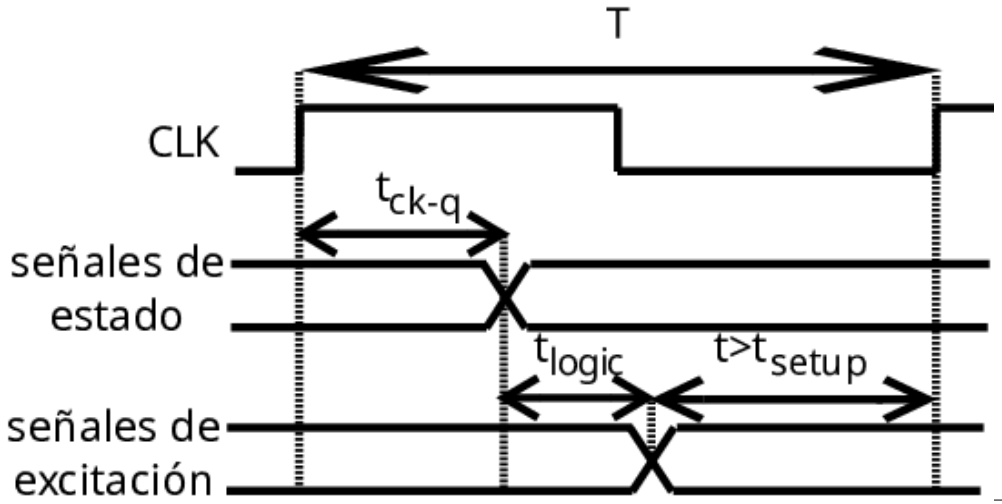
Frecuencia máxima de operación





Frecuencia máxima de operación

- $t=T-t_{ck-q}-t_{logic}$
- Debe cumplirse, $t>t_{setup}$ es decir, $T>t_{setup}+t_{ck-q}+t_{logic}$, es decir, $f<1/(t_{setup}+t_{ck-q}+t_{logic})$
- Este es un análisis idealizado en el que la señal de reloj llega simultáneamente a todos los biestables. Un análisis más realista puede consultarse en el apéndice del tema.



Concepto de máquinas de estados

Una máquina de estados finitos es una 6-tupla: $M(I,O,E,S_0,\delta,\lambda)$ donde I,O y E son conjuntos finitos no vacíos.

I: alfabeto de entrada.

O: alfabeto de salida .

E: estados.

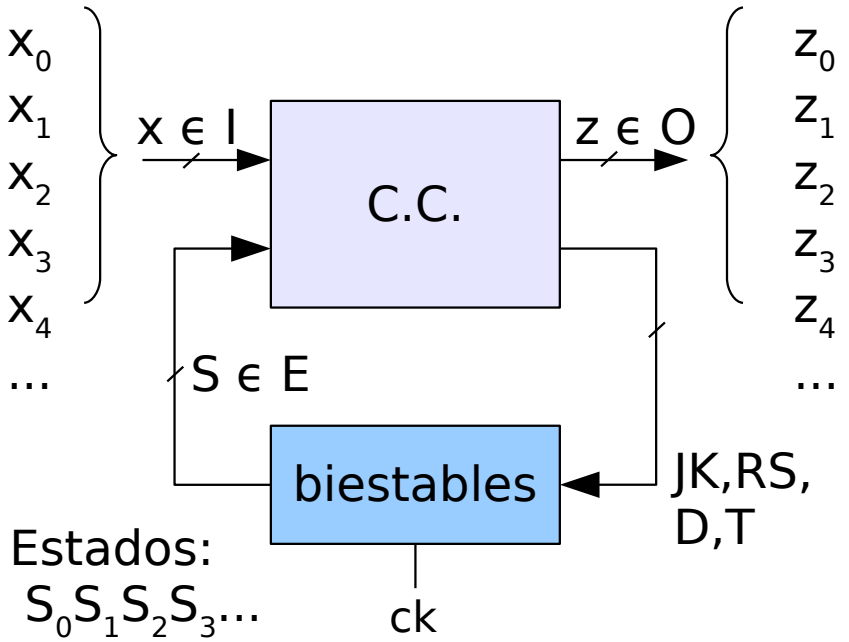
S_0: estado inicial (elemento de E)

δ: Función de próximo estado (δ: IxE → E)): $Q = \delta(q, x)$

λ: Función de salida (λ)

Mealy (λ:IxE→ O): $z = \lambda(q, x)$

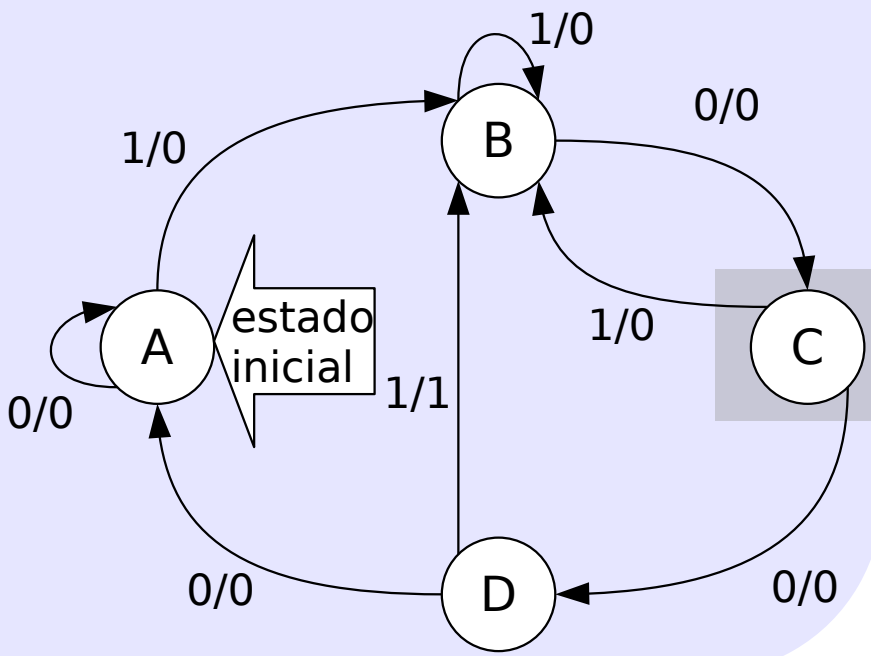
Moore (λ:E→ O): $z = \lambda(q)$



Diagrama/grafo de estados. Mealy

Cada arco del diagrama muestra x/z:

- x: valor de entrada que provoca la transición desde el estado S.
- z: valor de salida generado en el estado S cuando la entrada vale x.



La tabla muestra la misma información:

- Posibles estados en filas
- Posibles valores de entradas en columnas
- Próximo estado y salida en cada celda.

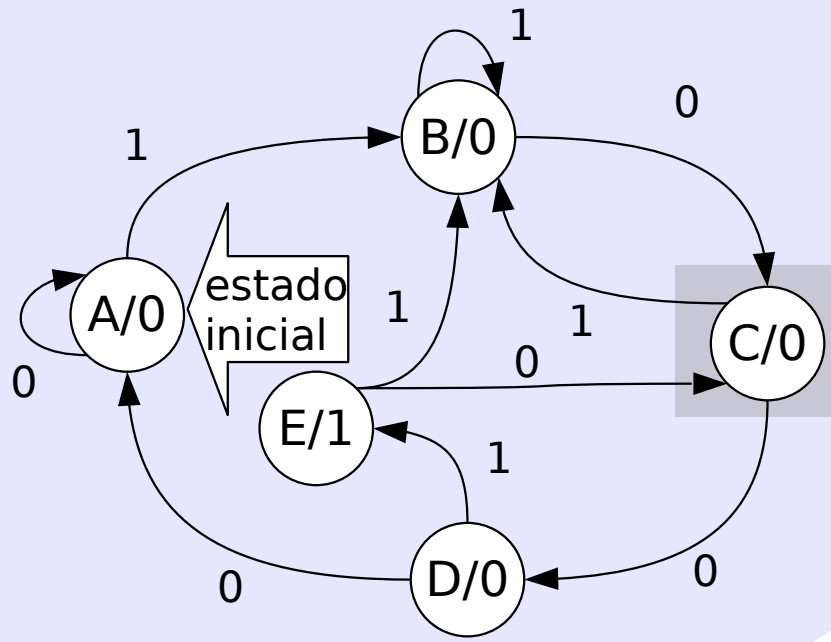
S \ x	0	1
A	A,0	B,0
B	C,0	B,0
C	D,0	B,0
D	A,0	B,1

$NS=\delta(S, x)$,
$z=\lambda(S, x)$

Diagrama/grafo de estados. Moore

Cada estado lleva asociado un valor de salida (z).

Los arcos indican las posibles transiciones desde cada estado (S) según el valor de entrada (x).



La tabla muestra la misma información:

- Posibles estados en filas
- Posibles valores de entradas en columnas
- Salida asociada al estado en la última columna.

S \ x	0	1	z=λ(S)
A	A	B	0
B	C	B	0
C	D	B	0
D	A	E	0
E	C	B	1

NS=δ(S, x)

Análisis de CSS

- Introducción
- Biestables
- Máquinas de estados finitos (FSM) y circuitos secuenciales síncronos (CSS)
- Análisis de CSS
  - Análisis lógico: procedimiento y ejemplo
  - Análisis temporal
- Diseño de CSS

Análisis de CSS

Objetivo:

Partiendo del circuito construido (esquema del circuito), obtener el diagrama de estados de la máquina que implementa e interpretar su operación/utilidad.

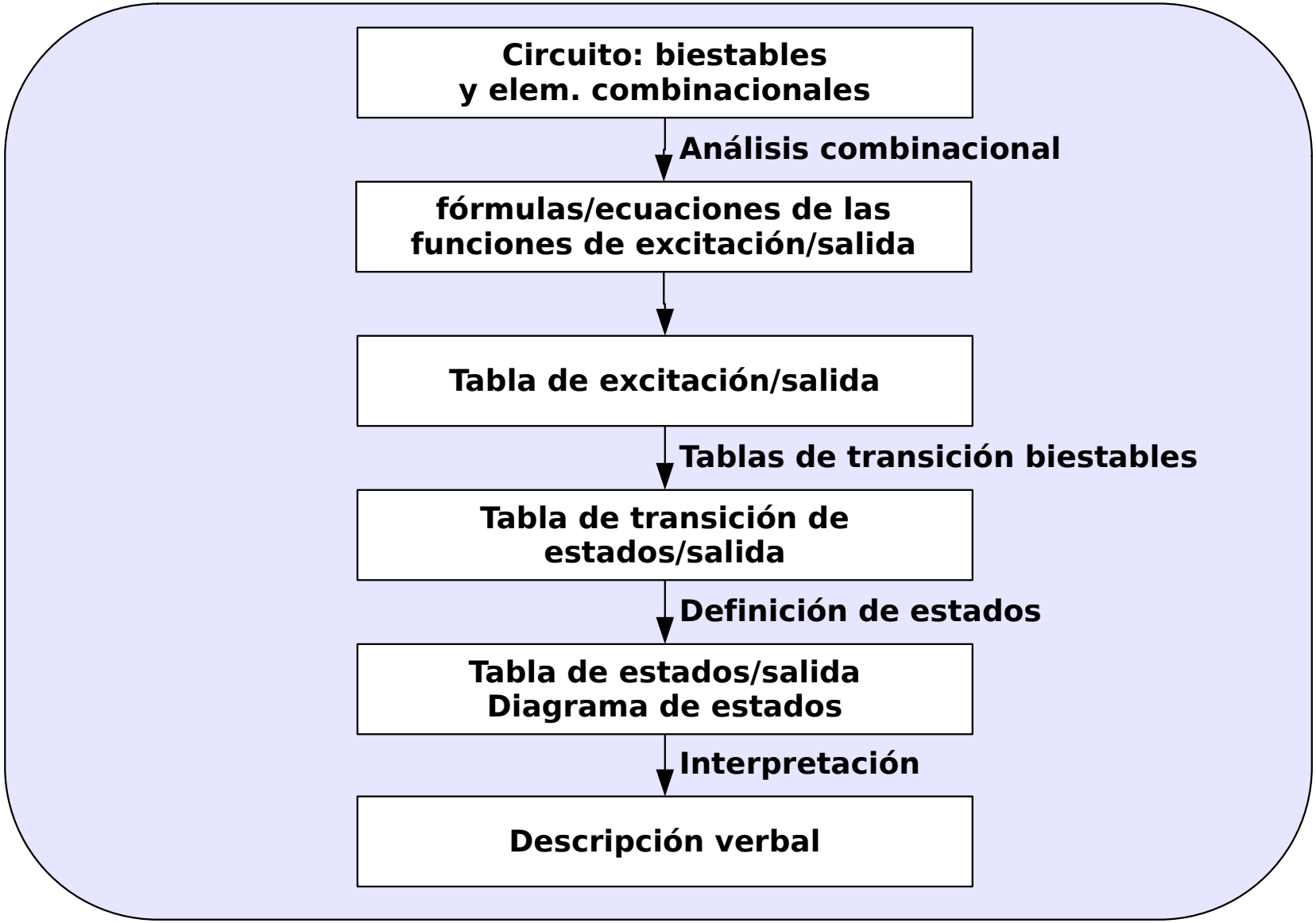
El proceso hasta obtener el diagrama de estados es sistemático.

La interpretación no es sistemática, intervienen:

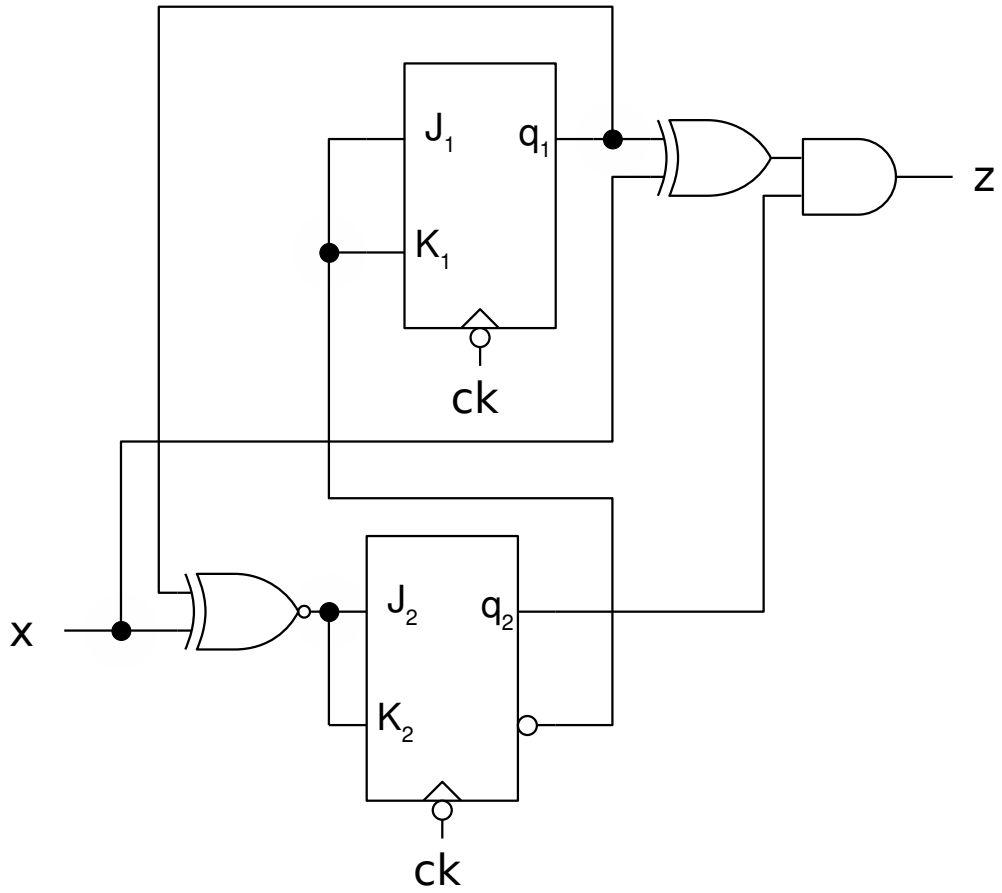
Experiencia

Información adicional

Análisis de CSS: Procedimiento



Análisis de CSS: Ejemplo

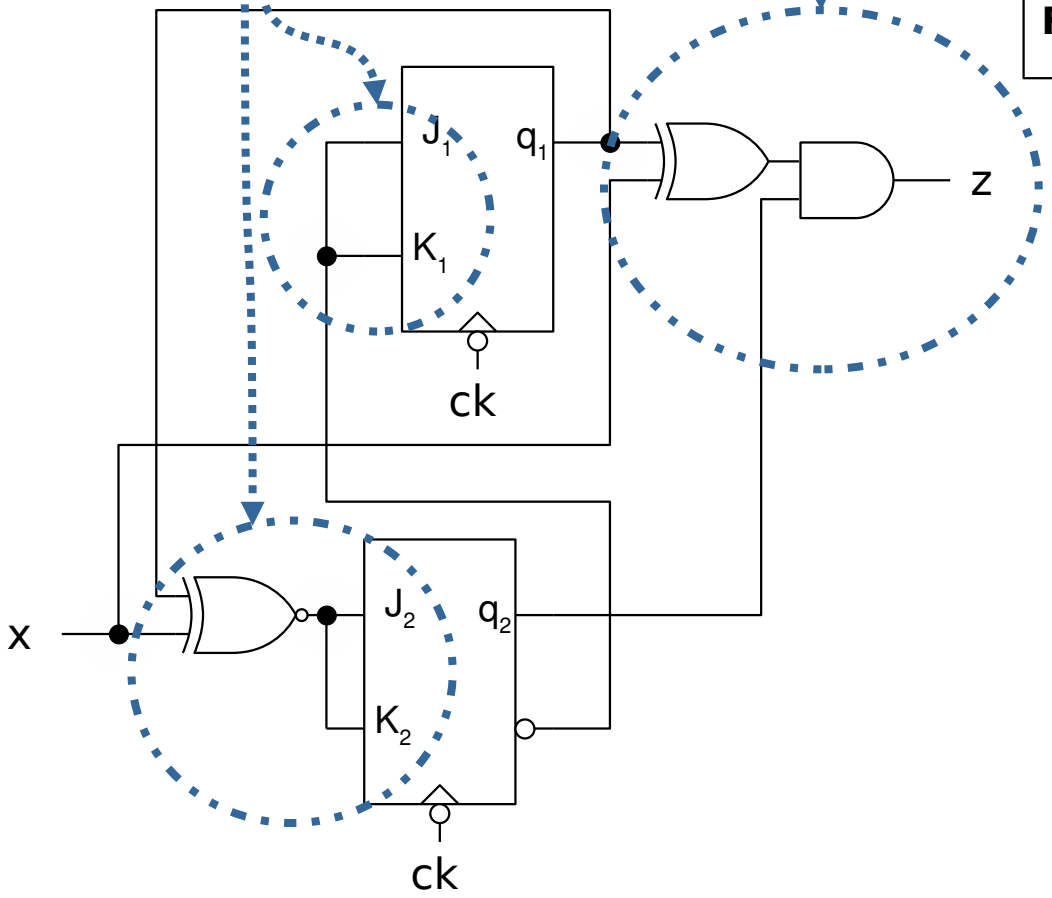


Análisis de CSS: Ejemplo

$J_1=K_1=\overline{q}_2$

$J_2=K_2=NOT(x \oplus q_1)$

$z= q_2(x \oplus q_1)$



Circuito: biestables y elem. combinacionales

↓ **Análisis combinacional**

Funciones de excitación/salida

↓

Tabla de excitación/salida

X q_2q_1	0	1
0 0	1 1, 1 1, 0	0 0, 1 1, 0
0 1	0 0, 1 1, 0	1 1, 1 1, 0
1 0	1 1, 0 0, 0	0 0, 0 0, 1
1 1	0 0, 0 0, 1	1 1, 0 0 , 0

$J_2\ K_2,\ J_1\ K_1, z$

Análisis de CSS: Ejemplo

Tabla del biestable JK

q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Q

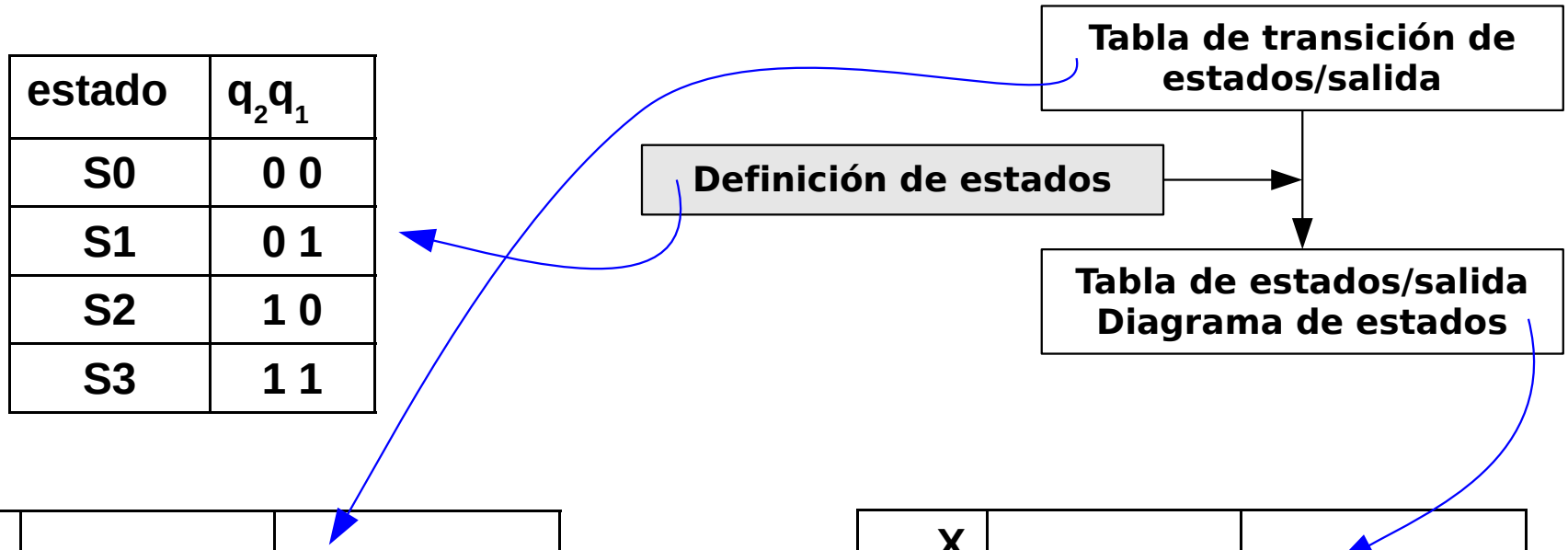
q_2q_1 \ X	0	1
0 0	1 1, 1 1, 0	0 0, 1 1, 0
0 1	0 0, 1 1, 0	1 1, 1 1, 0
1 0	1 1, 0 0, 0	0 0, 0 0, 1
1 1	0 0, 0 0, 1	1 1, 0 0 , 0

$J_2\ K_2,\ J_1\ K_1, z$

q_2q_1 \ X	0	1
0 0	1 1, 0	0 1, 0
0 1	0 0, 0	1 0, 0
1 0	0 0, 0	1 0, 1
1 1	1 1, 1	0 1 , 0

$Q_2Q_1,\ z$

Análisis de CSS: Ejemplo



estado	q_2q_1
S0	0 0
S1	0 1
S2	1 0
S3	1 1

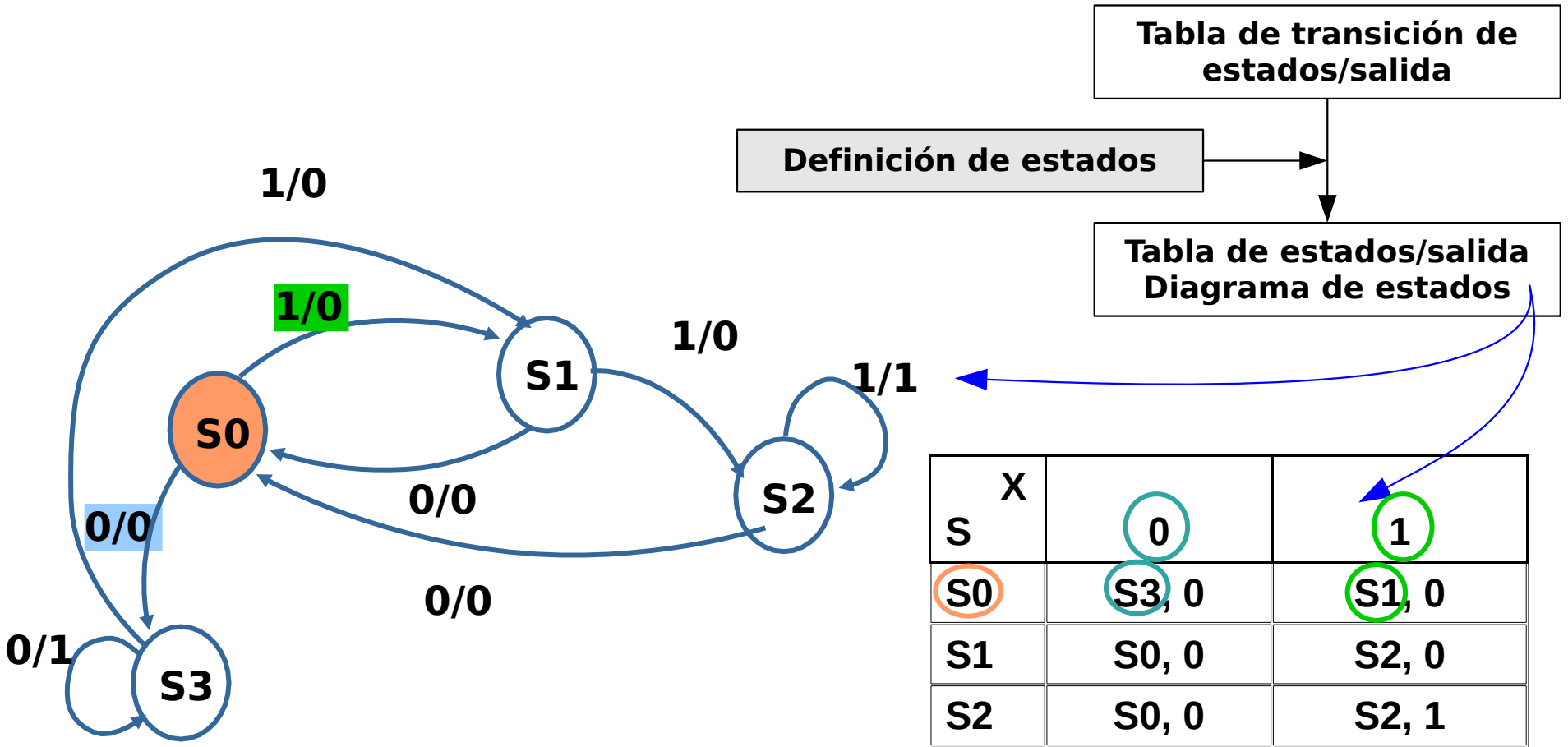
X q_2q_1	0	1
0 0	1 1, 0	0 1, 0
0 1	0 0, 0	1 0, 0
1 0	0 0, 0	1 0, 1
1 1	1 1, 1	0 1 , 0

Q_2Q_1, z

X S	0	1
S0	S3, 0	S1, 0
S1	S0, 0	S2, 0
S2	S0, 0	S2, 1
S3	S3, 1	S1, 0

NS, z

Análisis de CSS: Ejemplo



S \ X	0	1
S0	S3, 0	S1, 0
S1	S0, 0	S2, 0
S2	S0, 0	S2, 1
S3	S3, 1	S1, 0

NS, z

Análisis de CSS: Ejemplo

El análisis temporal de un CSS es sencillo una vez descrita la máquina de estados correspondiente:

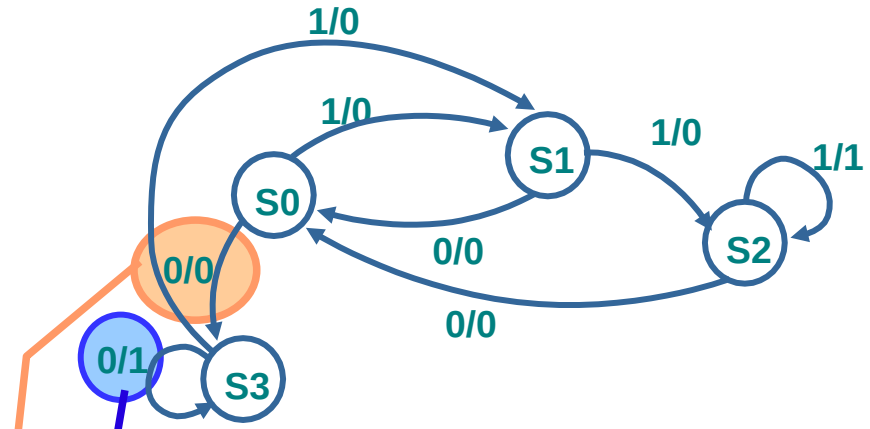
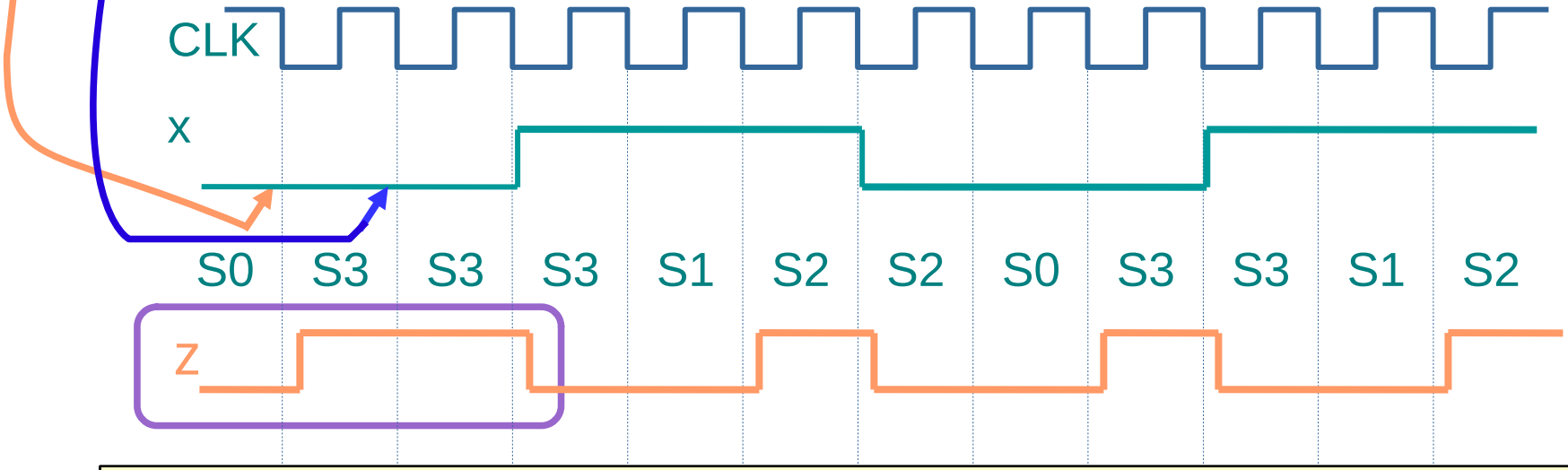


Tabla de estados/salida
Diagrama de estados

Interpretación

Descripción verbal



Función: detecta tres ceros o tres unos consecutivos en la entrada

Concepto de máquinas de estados. Propiedades

- Dos máquinas de estados son equivalentes si y sólo si generan las mismas secuencias de salida para toda secuencia de entrada.
- Algunas máquinas de estados se pueden optimizar: Hay máquinas equivalentes con menor número de estados.
- Las máquinas de estados pueden ser incompletamente especificadas: el próximo estado o la salida puede no estar definida para cierta combinación de estado actual y entrada dados.

Aplicaciones de los circuitos secuenciales síncronos

Detectores de secuencia

La salida se activa sólo en caso de que aparezca una determinada secuencia a la entrada.

Generadores de secuencia

La salida es una secuencia fija o variable en función de la entrada.

Unidades de control

Las entradas y el estado definen la actuación sobre un sistema externo (control de una barrera, control de temperatura, control de presencia, control de nivel de líquidos, etc.)

Procesamiento secuencial

La secuencia de salida es el resultado de aplicar alguna operación a la secuencia de entrada (cálculo de la paridad u otros códigos de comprobación de errores, suma, codificación/decodificación secuencial en general).

Diseño de CSS

Introducción

Biestables

Máquinas de estados finitos (FSM) y circuitos secuenciales síncronos (CSS)

Análisis de CSS

Diseño de CSS

- Objetivos
- Procedimiento y ejemplo

Objetivo

- Objetivo:
  - Definir una máquina de estados que resuelva un problema dado.
  - Implementar la máquina de estados mediante un circuito secuencial síncrono.
- Habitualmente, el proceso de diseño sigue criterios de minimización de coste y recursos, por ejemplo:
  - Minimizar el número de elementos de memoria
  - Minimizar los componentes combinacionales
  - Maximizar la velocidad (frecuencia de operación)
  - Reducción del consumo de energía
  - Reducción del tiempo y esfuerzo de diseño
- Suele haber un compromiso entre diferentes criterios.

Procedimientos

Procedimiento manual

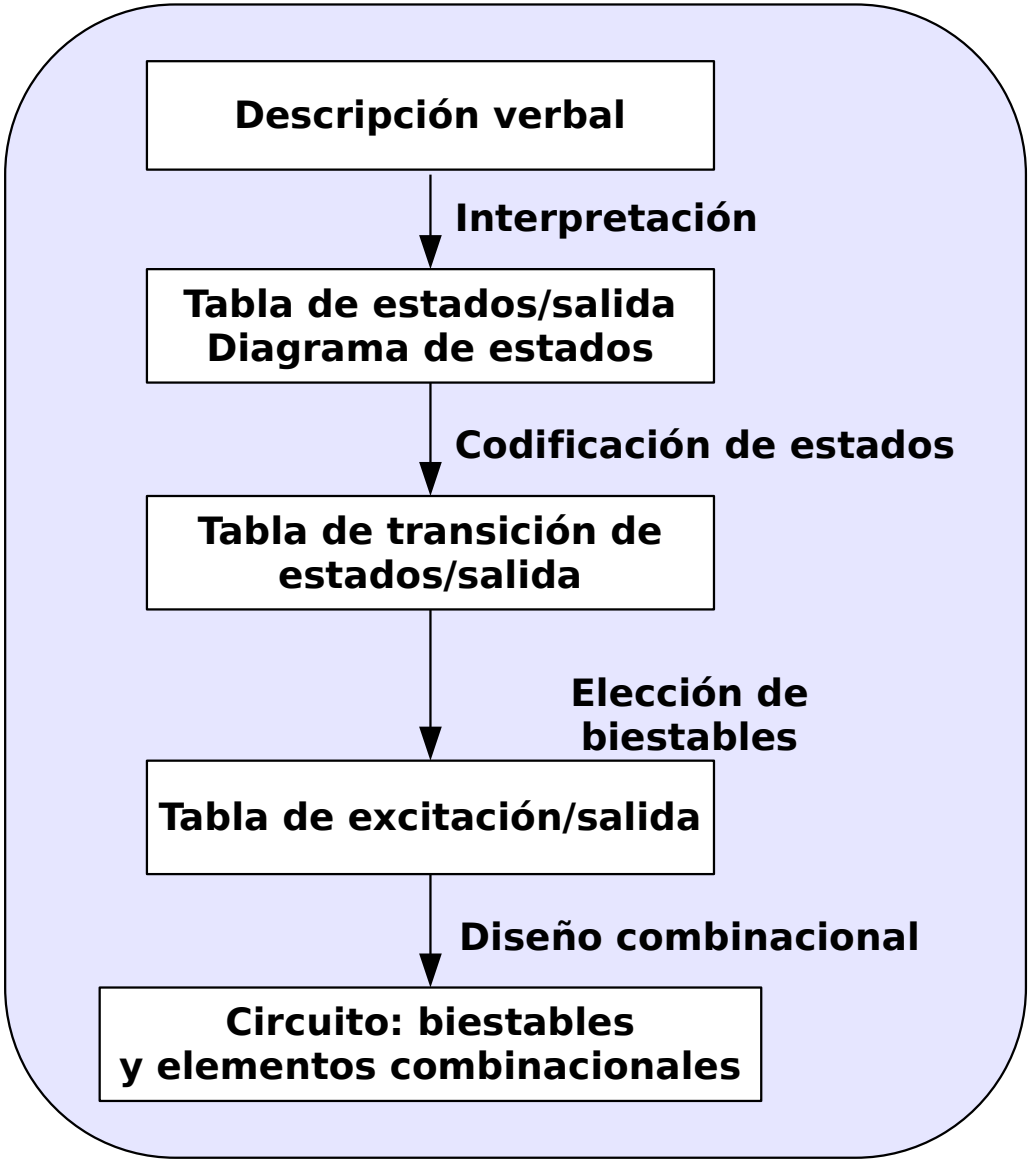
- Realizable con lápiz y papel.
- Describe la máquina que resuelve el problema mediante un diagrama o tabla de estados.
- A partir del diagrama de estados se van obteniendo diversas representaciones hasta llegar al circuito digital.

Procedimiento con herramientas de diseño (CAD)

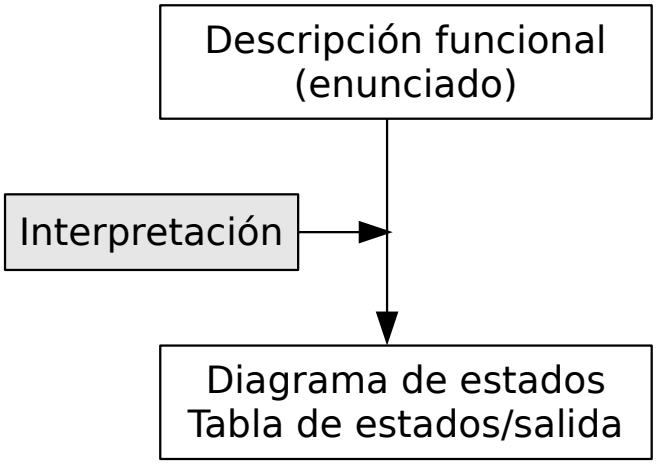
- Emplea herramientas informáticas.
- A partir del enunciado del problema o el diagrama de estados, se hace una descripción formal en un LDH.
- Se emplean herramientas de simulación para comprobar que la descripción del sistema es correcta.
- Se emplean herramientas de síntesis automática para obtener el circuito final.

Procedimiento manual

Procedimiento inverso al análisis



Interpretación



Es la fase más importante del diseño

Es la fase menos sistemática

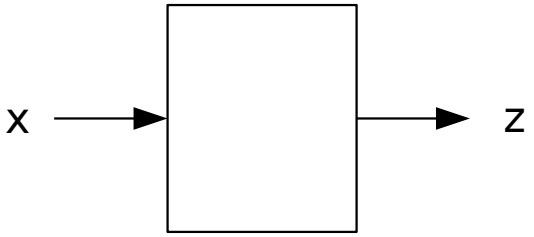
Procedimiento/consejos

- Definir claramente entradas y salidas.
- Elegir Mealy o Moore según características del problema (sincronización de la salida)
- Identificar y definir los estados adecuados
- Detallar la máquina de estados
- Comprobar el diagrama con una secuencia de entrada típica

Interpretación

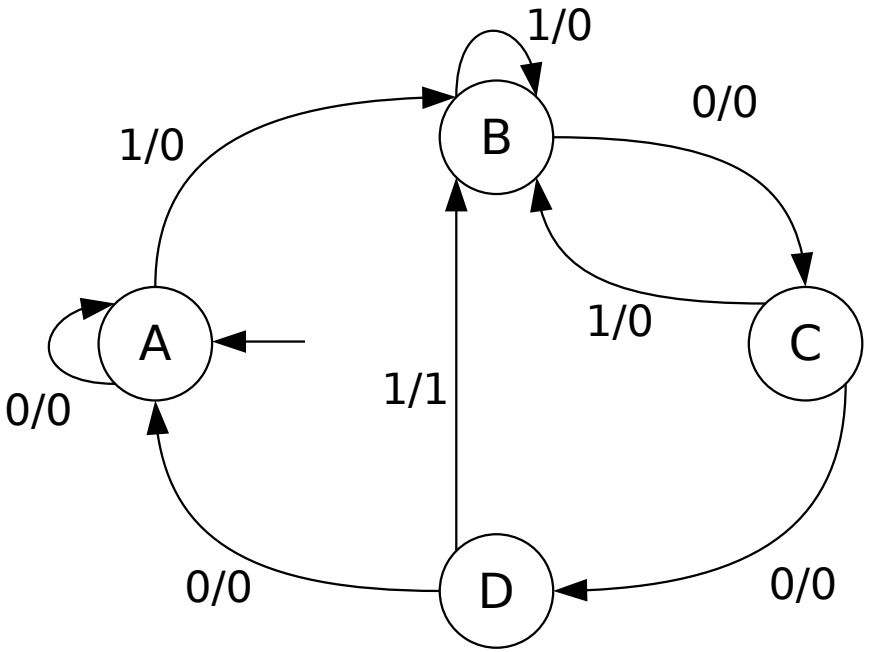
Ejemplo

Diseñe un circuito con una entrada x y una salida z que detecte la aparición de la secuencia "1001" en la entrada. Cuando esto ocurre se activará la salida (z=1). El último "1" de una secuencia puede considerarse también el primer "1" de una secuencia posterior (detector con solapamiento).




```
x: 00100111000011101001001001010011...
z: 00000100000000000000100100100010...
```

Interpretación

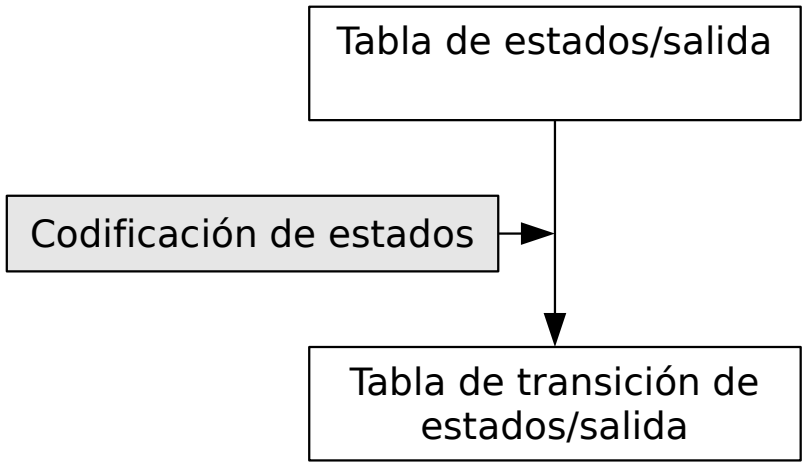


S \ x	0	1
A	A,0	B,0
B	C,0	B,0
C	D,0	B,0
D	A,0	B,1

NS,z

A: ningún bit de la secuencia se ha recibido aún, esperando "1"
B: 1er bit de la secuencia recibido, esperando "0"
C: 2 bits de la secuencia recibidos, esperando "0"
D: 3 bits de la secuencia recibidos, esperando "1"

Codificación de estados



Objetivo:

- Asignar palabras binarias a los estados (codificación de estados) para su almacenamiento en biestables.
- Para codificar n estatos se requieren palabras de al menos $\lceil \log_2(n) \rceil$ bits.

Opciones de asignación:

- Algoritmos complejos
- Asignación arbitraria
- Un biestable por estado (codificación one-hot)

Elección de biestables:

- Afecta al resultado final: número de componentes, tamaño, velocidad de operación, consumo de energía.
- Elección diferente según el objetivo (criterio de coste).

Codificación de estados

Tabla de estados/salida

S \ x	0	1
A	A,0	B,0
B	C,0	B,0
C	D,0	B,0
D	A,0	B,1

NS,z

Codificación de estados

S	q_1q_2
A	00
B	01
C	11
D	10

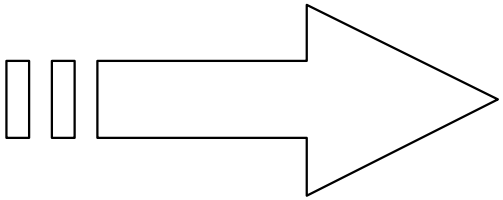
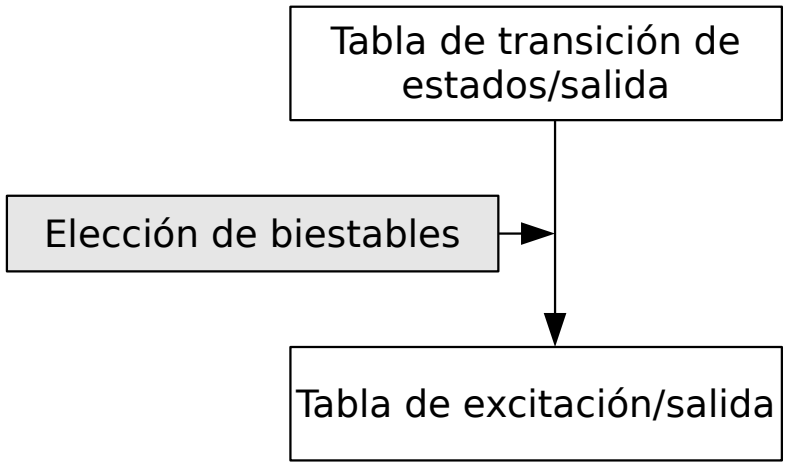


Tabla de transición de estados/salida

q_1q_2 \ x	0	1
00	00,0	01,0
01	11,0	01,0
11	10,0	01,0
10	00,0	01,1

Q_1Q_2,z

Elección de biestables



Objetivo

Seleccionar qué tipo de biestables almacenarán los bits del estado codificado.

Opciones

JK: reduce el coste de la parte combinacional.

RS: más simple que el JK pero menos flexible.

D: facilita el diseño, reduce el número de conexiones.

T: más conveniente en aplicaciones específicas (contadores)

Elección de biestables. Ejemplo: JK

Tabla de transición de estados/salida

q_1q_2 \ x	0	1
00	00,0	01,0
01	11,0	01,0
11	10,0	01,0
10	00,0	01,1

Q_1Q_2,z

Tabla de excitación

q → Q	JK
0 → 0	0x
0 → 1	1x
1 → 0	x1
1 → 1	x0

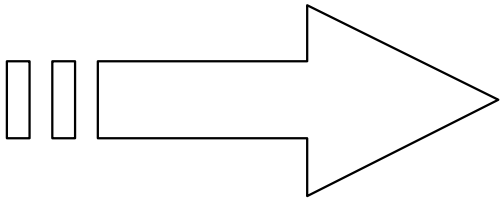


Tabla de excitación/salida

q_1q_2 \ x	0	1
00	0x,0x,0	0x,1x,0
01	1x,x0,0	0x,x0,0
11	x0,x1,0	x1,x0,0
10	x1,0x,0	x1,1x,1

J_1K_1,J_2K_2,z

Elección de biestable. Ejemplo: D

En el biestable D:

Q = D

D = Q

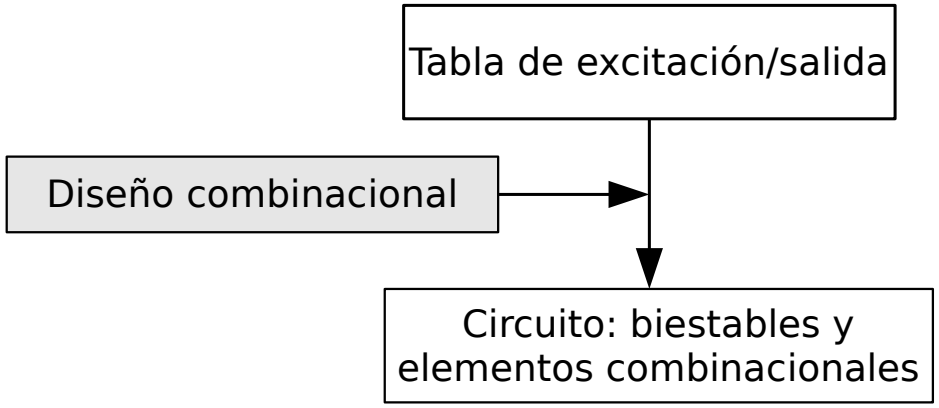
Tabla de transición de estados/salida ⟺ Tabla de excitación/salida

q_1q_2 \ x	0	1
00	00,0	01,0
01	11,0	01,0
11	10,0	01,0
10	00,0	01,1

Q_1Q_2,z

D_1,D_2,z

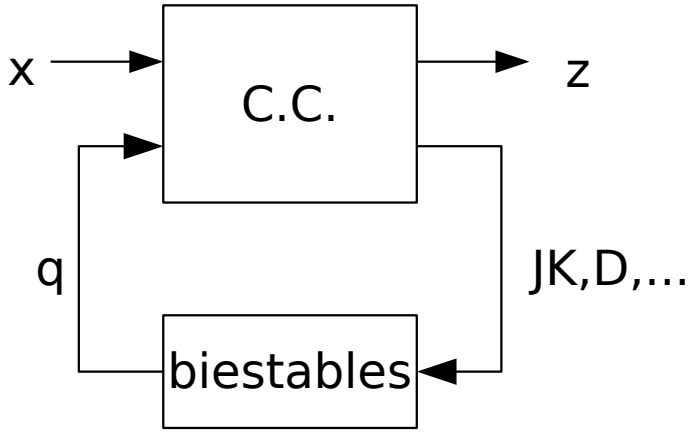
Diseño de la parte combinacional



La tabla de excitación/salida es una especificación de la parte combinacional.

La implementación se realiza mediante cualquiera de las técnicas de diseño de C.C.

- Dos niveles de puertas
- Subsistemas: multiplexores, decodificadores, etc.
- Etc.



Parte combinacional. Ejemplo

q_1q_2 \ x	0	1
00	0x,0x,0	0x,1x,0
01	1x,x0,0	0x,x0,0
11	x0,x1,0	x1,x0,0
10	x1,0x,0	x1,1x,1

J_1K_1,J_2K_2,z

q_1q_2 \ x	0	1
00	0	0
01	1	0
11	x	x
10	x	x

J_1

q_1q_2 \ x	0	1
00	x	x
01	x	x
11	0	1
10	1	1

K_1

q_1q_2 \ x	0	1
00	0	0
01	0	0
11	0	0
10	0	1

z

q_1q_2 \ x	0	1
00	0	1
01	x	x
11	x	x
10	0	1

J_2

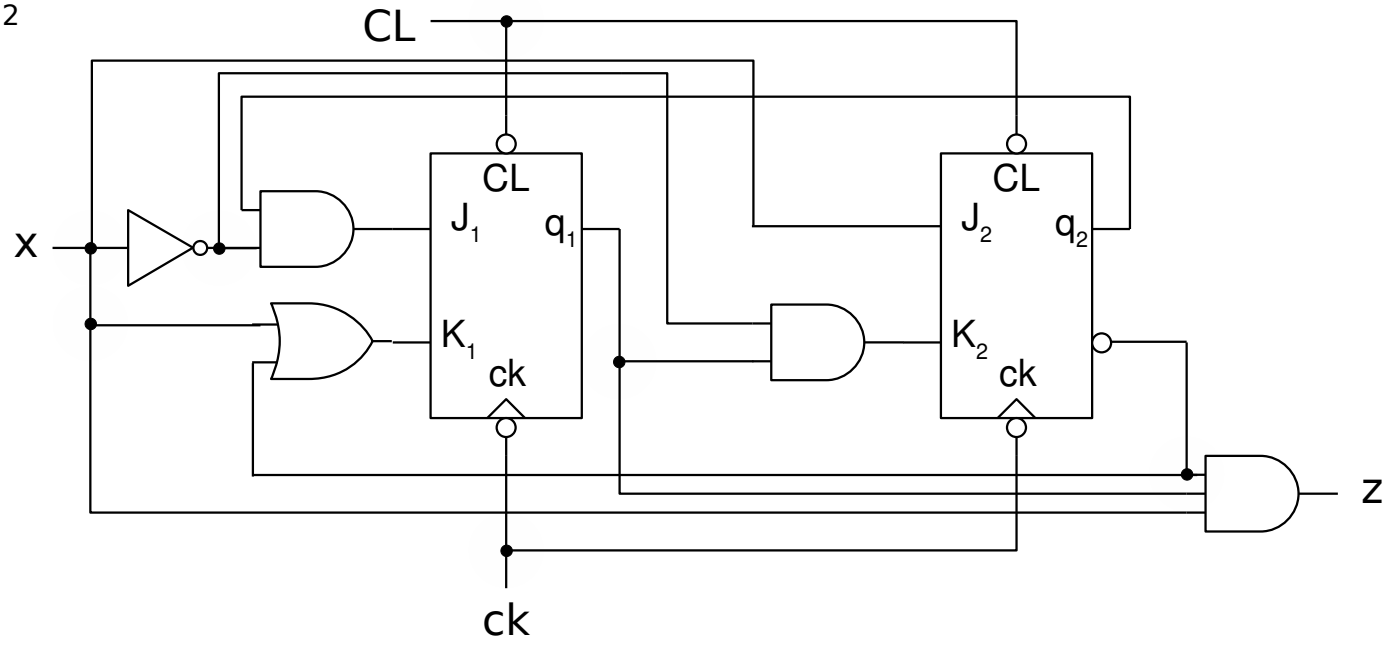
q_1q_2 \ x	0	1
00	x	x
01	0	0
11	1	0
10	x	x

K_2

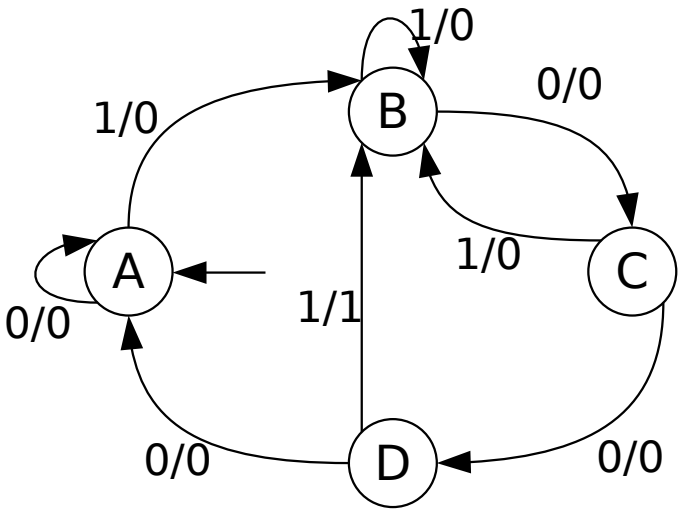
$$J_1 = \bar{x}\, q_2$$
$$K_1 = x + \bar{q}_2$$
$$J_2 = x$$
$$K_2 = \bar{x}\, q_1$$
$$z = x\, q_1\, \bar{q}_2$$

Circuito. Ejemplo

$J_1 = \overline{x}\, q_2$
$K_1 = x + \overline{q}_2$
$J_2 = x$
$K_2 = x\, \overline{q}_1$
$z = x\, q_1\, \overline{q}_2$



Ejemplo. Resumen



S \ x	0	1
A	A,0	B,0
B	C,0	B,0
C	D,0	B,0
D	A,0	B,1

Q,z

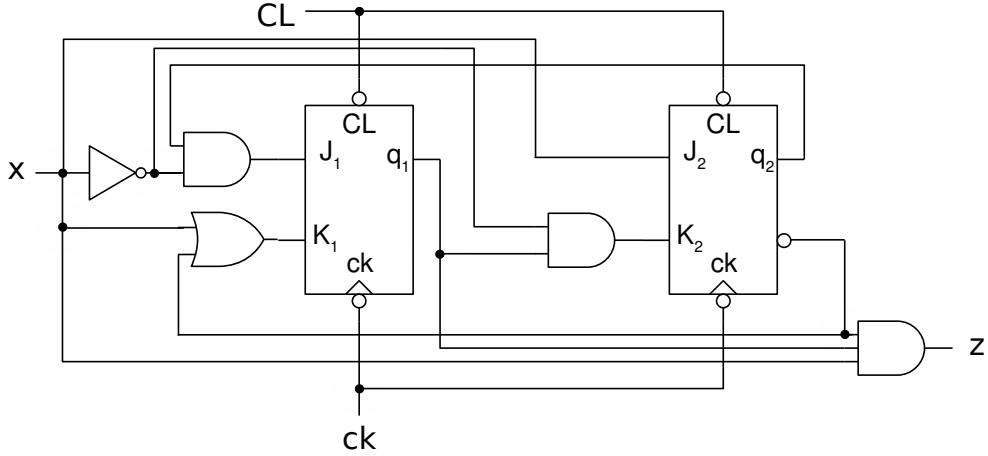
q_1q_2 \ x	0	1
00	00,0	01,0
01	11,0	01,0
11	10,0	01,0
10	00,0	01,1

Q,z

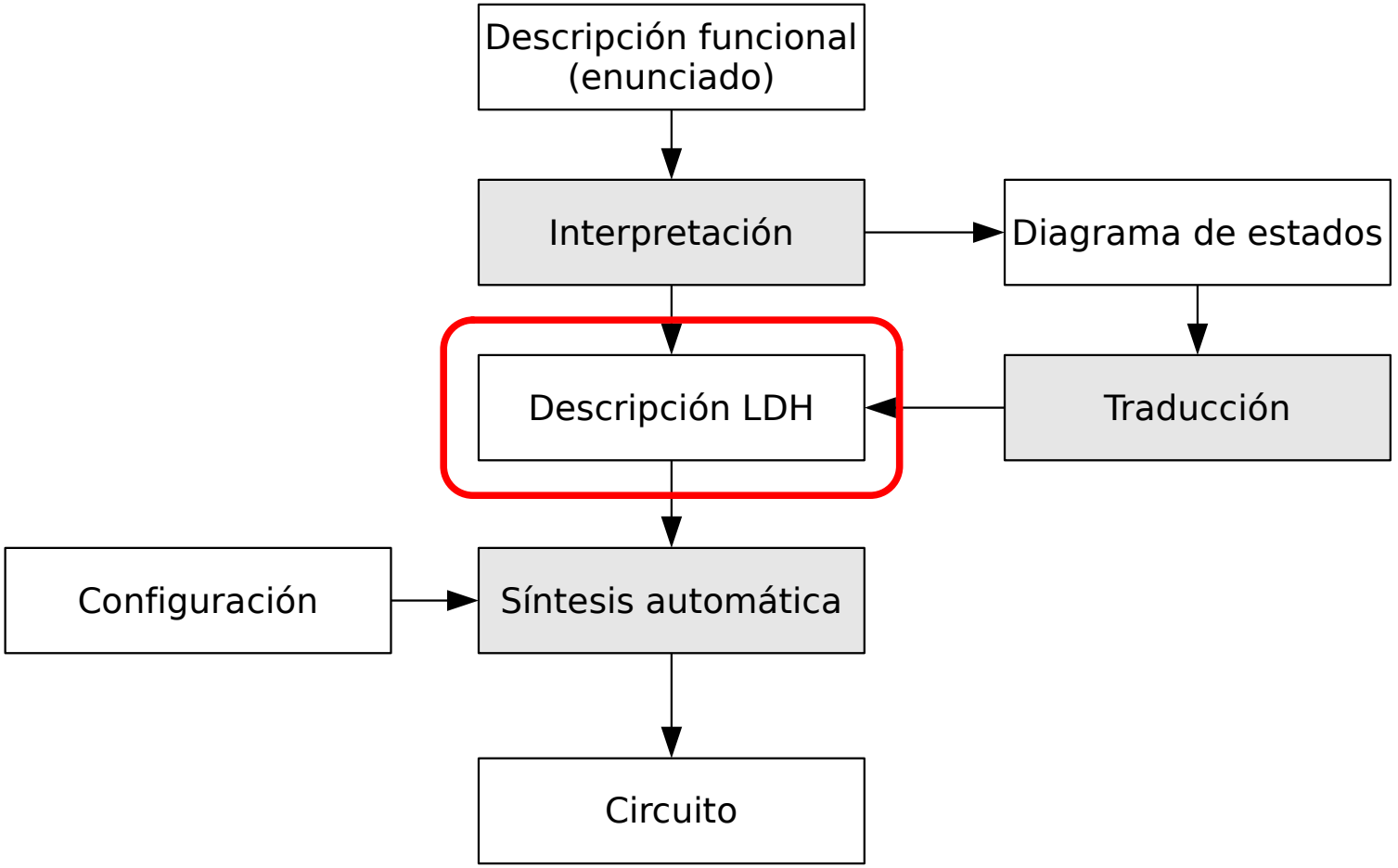
q_1q_2 \ x	0	1
00	0x,0x,0	0x,1x,0
01	1x,x0,0	0x,x0,0
11	x0,x1,0	x1,x0,0
10	x1,0x,0	x1,1x,1

J_1K_1,J_2K_2,z

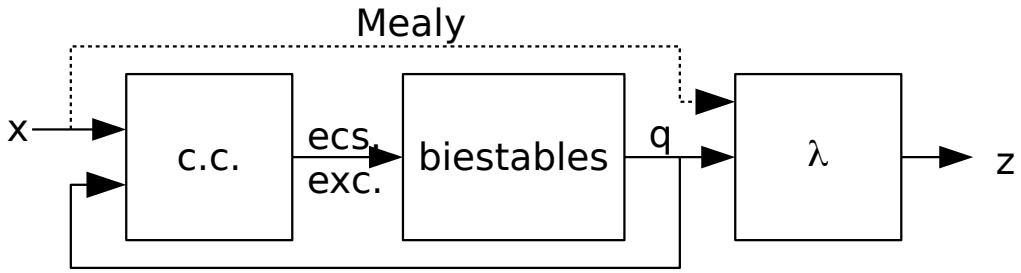
$$J_1 = \bar{x}\, q_2$$
$$K_1 = x + \bar{q}_2$$
$$J_2 = x$$
$$K_2 = x\, \bar{q}_1$$
$$z = x\, q_1\, \bar{q}_2$$



Procedimiento con herramientas de diseño



Descripción de FSM en Verilog



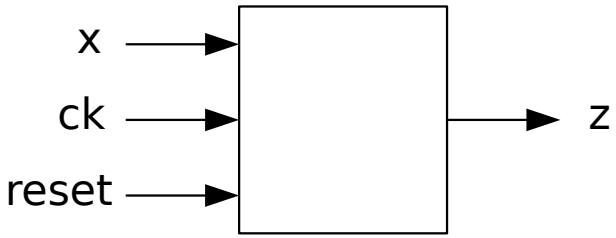
Dos procesos:

1) **Cambio de estado-** representa el bloque de biestables
2) **Cálculo del próximo estado y salida-** representa el circuito combinacional que genera las señales de excitación y de salida

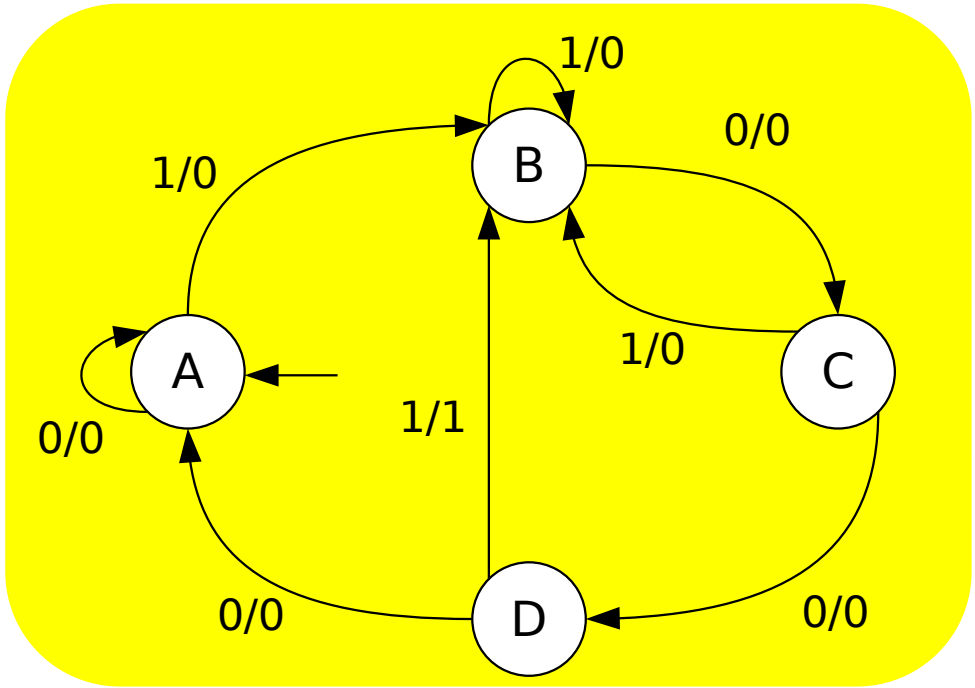
Sólo el proceso de cambio de estado es secuencial

```
// Proceso de cambio de estado
// (secuencial)
always @(posedge ck, posedge reset)
    if (reset)
        state <= A;
    else
        state <= next_state;


// Proceso de cálculo del nuevo estado
// y salidas
// (combinacional)
always @* begin
    z= .......;
    case (state)
    A:
        next_state = . . .;
    B:
        Next_state = . . .;
    endcase
end
```

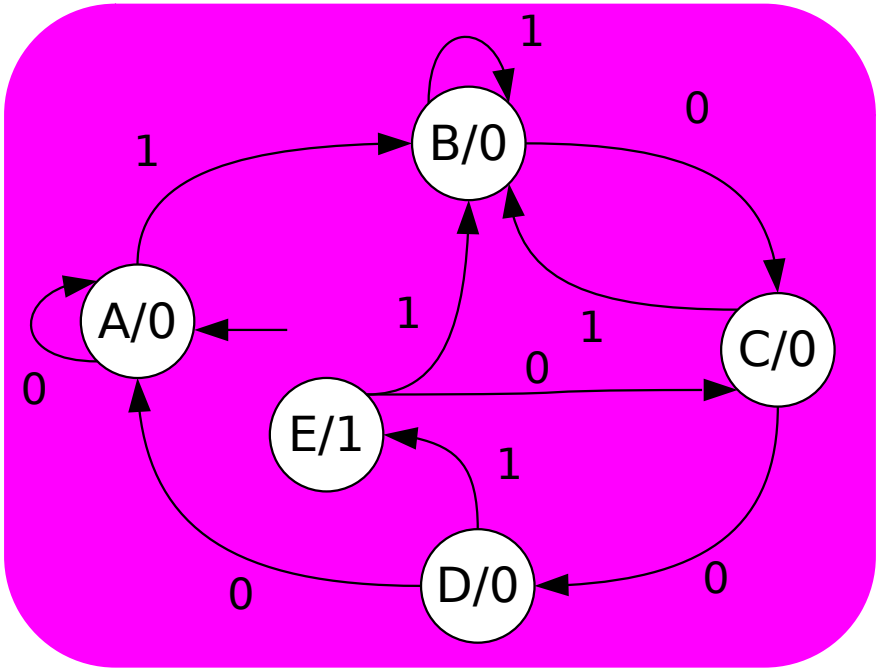
FSM en Verilog. Ejemplo



Consideramos de nuevo el ejemplo del detector de la secuencia 1001 con solapamiento



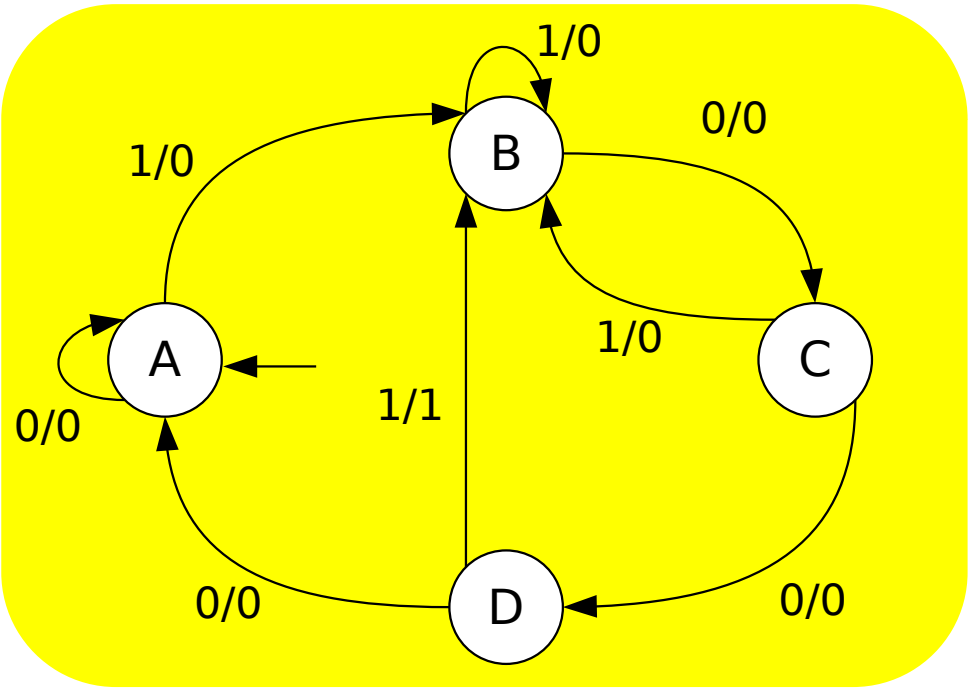
Solución con máquina de Mealy



Solución con máquina de Moore

FSM en Verilog. Ejemplo Mealy.

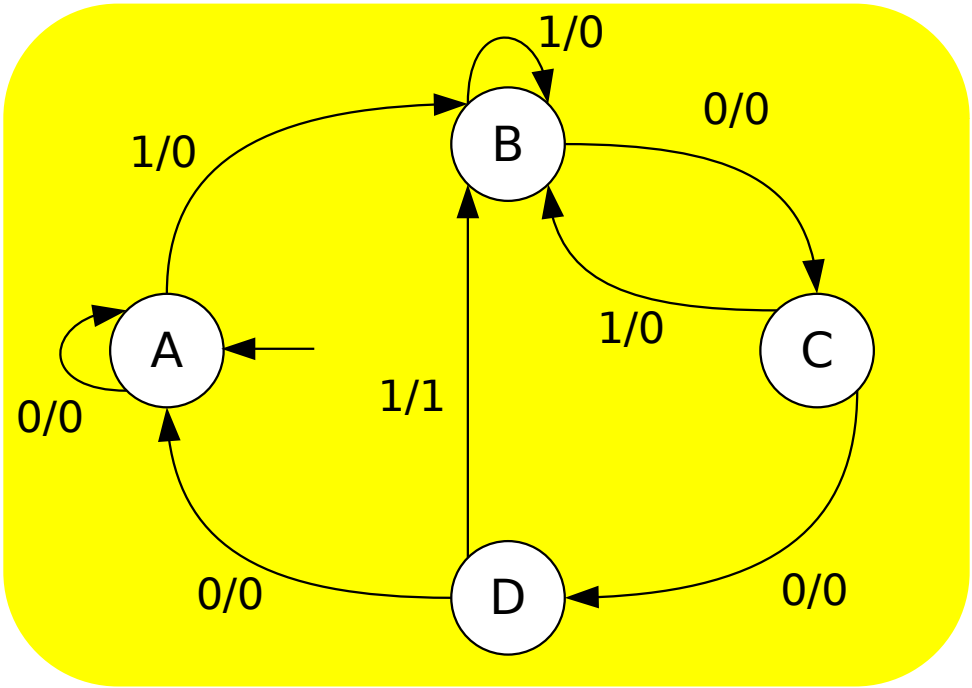
```
module seq_mealy(
    input wire ck,        // reloj
    input wire reset,     // reset
    input wire x,         // entrada
    output reg z      // salida
    );

// Codificación de estados
    parameter [1:0]
        A = 2'b00,
        B = 2'b01,
        C = 2'b11,
        D = 2'b10;

// Variables de estado y próximo estado
    reg [1:0] state, next_state;

// Proceso de cambio de estado
//(secuencial)

    always @(posedge ck, posedge reset)
        if (reset)
            state <= A;
        else
            state <= next_state;
```



FSM en Verilog. Ejemplo Mealy.

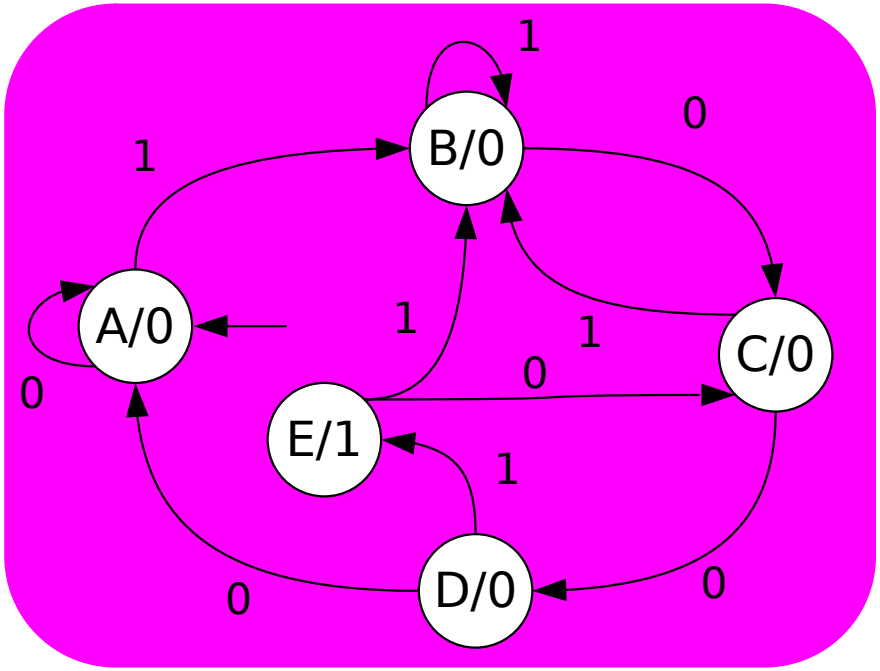
```
// Proceso de cálculo del nuevo estado
// (combinacional)

     always @* begin
          next_state = 2'bxx;
          case (state)
          A:   if (x == 0)
                    next_state = A;
               else
                    next_state = B;
          B:   if (x == 0)
                    next_state = C;
               else
                    next_state = B;
          C:   if (x == 0)
                    next_state = D;
               else
                    next_state = B;
          D:   if (x == 0)
                    next_state = A;
               else
                    next_state = B;
          endcase
     end
```

```
// Proceso de cálculo de la salida
// (combinacional)

     always @* begin
          if (state == D && x ==1)
               z = 1;
          else
               z = 0;
     end
endmodule
```



FSM en Verilog. Ejemplo Moore.

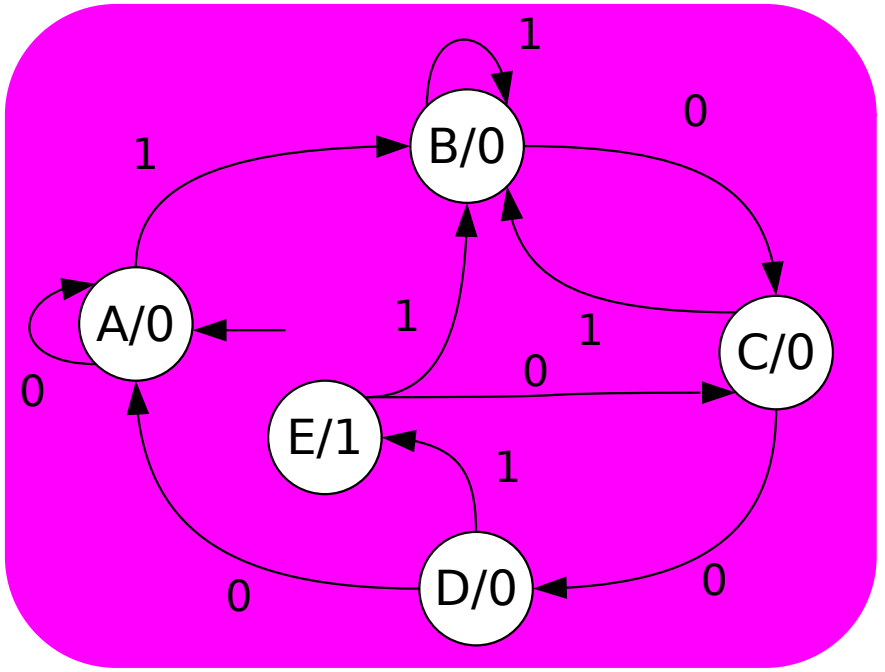
```
module seq_moore(
    input wire ck,        // reloj
    input wire reset,     // reset
    input wire x,         // entrada
    output reg z      // salida
    );
// Codificación de estados
    parameter [2:0]
        A = 3'b000,
        B = 3'b001,
        C = 3'b010,
        D = 3'b011,
        E = 3'b100;

// Variables de estado y próximo estado
    reg [2:0] state, next_state;

// Proceso de cambio de estado
//(secuencial)

    always @(posedge ck, posedge reset)
        if (reset)
            state <= A;
        else
            state <= next_state;
```



FSM en Verilog. Ejemplo Moore.

```
// Proceso de cálculo del nuevo estado
// (combinacional)
    always @* begin
        next_state = 3'bxxx;
        case (state)
        A:  if (x == 0)
                next_state = A;
            else
                next_state = B;
        B:  if (x == 0)
                next_state = C;
            else
                next_state = B;
        C:  if (x == 0)
                next_state = D;
            else
                next_state = B;
        D:  if (x == 0)
                next_state = A;
            else
                next_state = E;
        E:  if (x == 0)
                next_state = C;
            else
                next_state = B;
        endcase
    end
```

```
// Proceso de cálculo de la salida
// (combinacional)

    always @* begin
        if (state == E)
            z = 1;
        else
            z = 0;
    end
endmodule
```

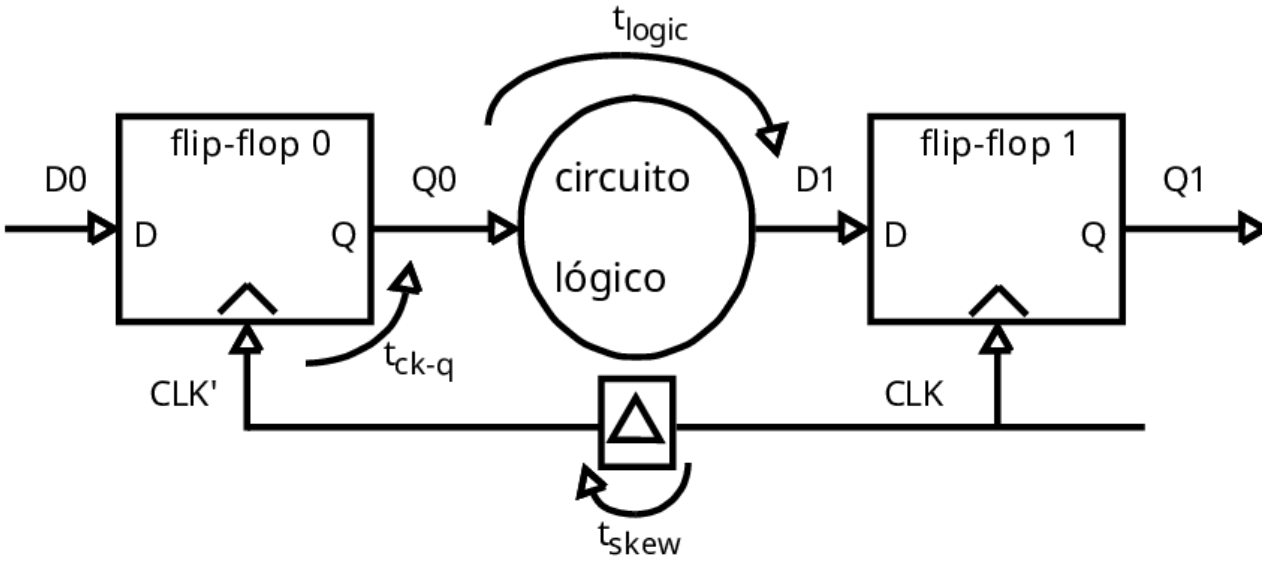


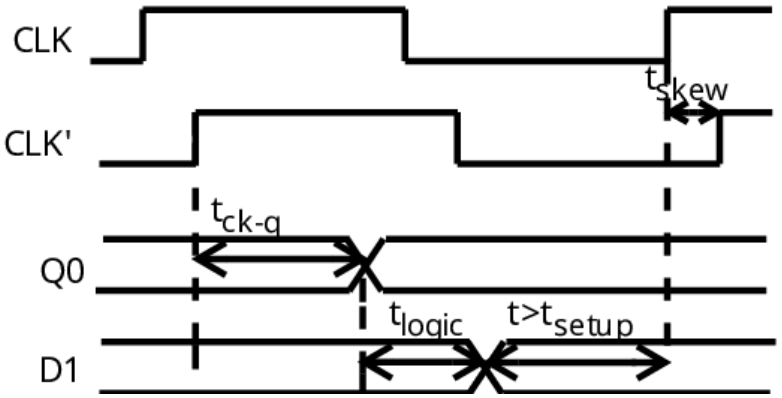
Apéndices

Desajuste de reloj

- Cuando en un circuito secuencial síncrono los flancos de la o las señales de reloj no llegan simultáneamente a todos los biestables se dice que hay *desajuste de reloj* (clock skew).
- Una de sus causas es el tiempo de propagación desigual en las líneas que transportan las señales de reloj.
- Afecta a la frecuencia máxima de operación de los circuitos.

Desajuste de reloj





$$T > t_{setup} + t_{ck\text{-}q} + t_{logic} + t_{skew}$$