Tema 7. Análisis y diseño de circuitos secuenciales

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

Jorge Juan <jjchico@dte.us.es>

Contenidos

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

Introducción

- Muchos problemas prácticos no pueden resolverse sólo mediante el uso de funciones combinacionales.
- En este tipo de sistemas, el valor de las salidas en un instante de tiempo no puede determinarse a partir del valor de las entradas en ese mismo instante de tiempo
- Se necesita que la acción del sistema tenga en cuenta no sólo las entradas sino también la historia pasada del sistema (estado).
- Para almacenar un estado son necesarios nuevos elementos de circuito: biestables.

Biestables

- Los biestables son circuitos electrónicos con dos estados estables
- Son el elemento básico de los circuitos secuenciales
- Poseen una o más entradas que hacen que sea posible conmutar entre los dos estados estables
- Un circuito con *n* biestables puede almacenar hasta 2^n estados

Biestable SR asíncrono

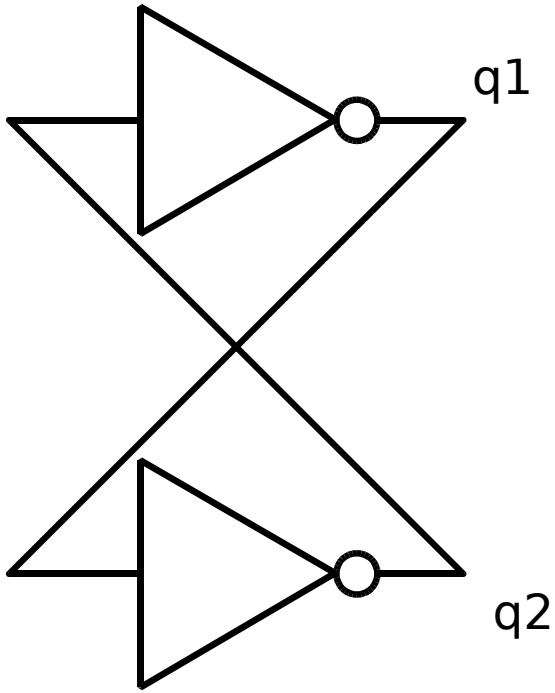
La capacidad de almacenar información se obtiene de la "realimentación" de las salidas hacia las entradas: el valor de la salida refuerza el de las entradas y viceversa.

Estados estables:

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

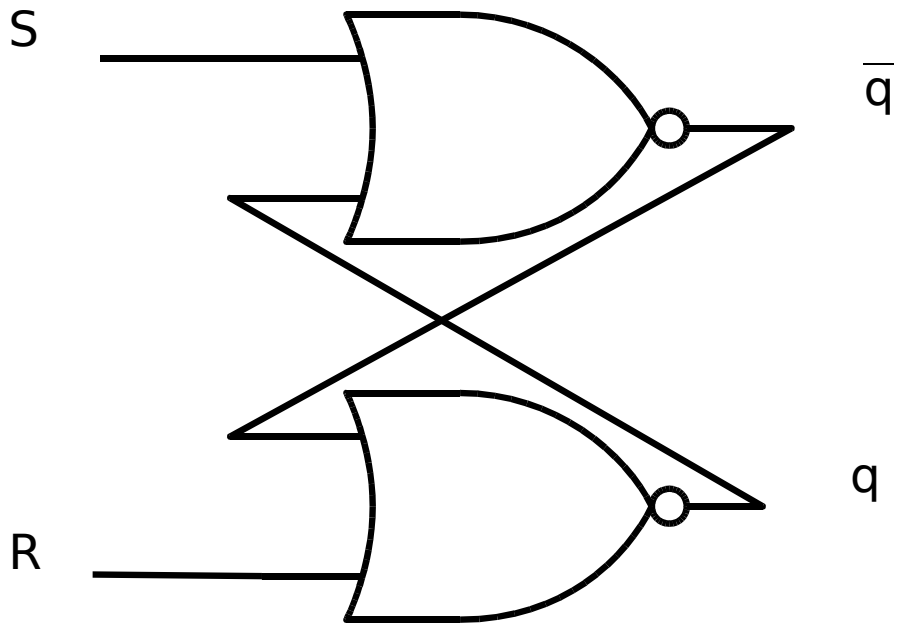
Convenio

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



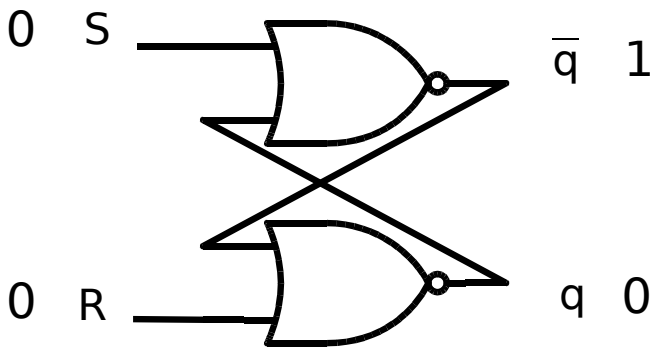
Biestable SR asíncrono

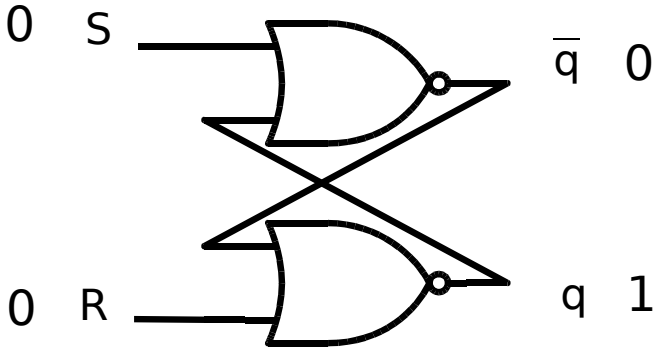
Necesitamos un circuito que permita almacenar dos estados y conmutar entre ellos.



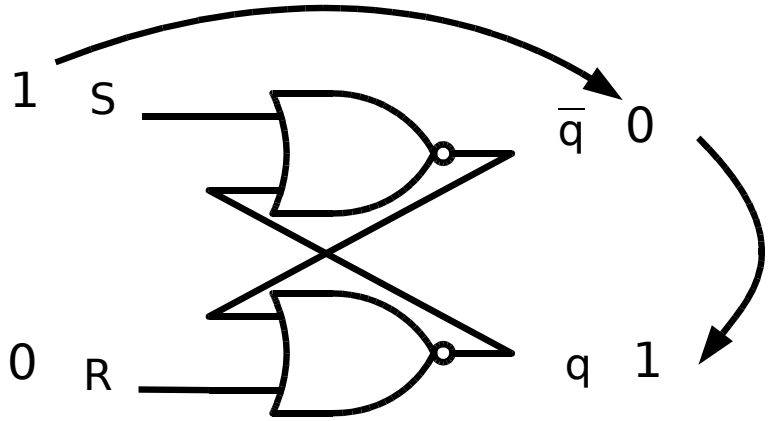
Biestable SR asíncrono

R=S=0 conserva el estado

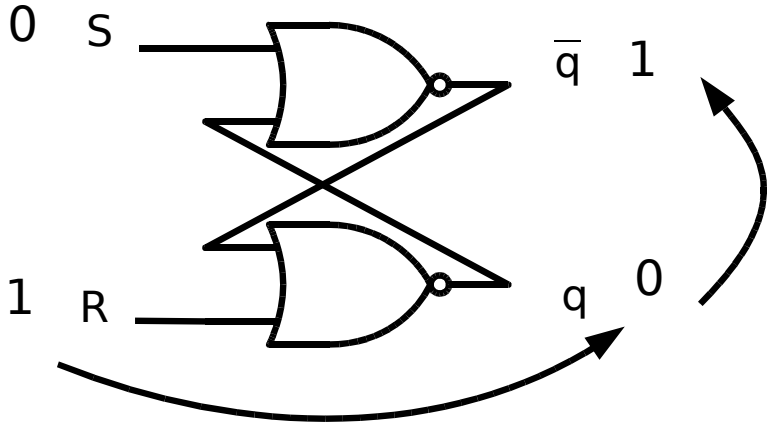




S=1, R=0 cambia a 1 (set)

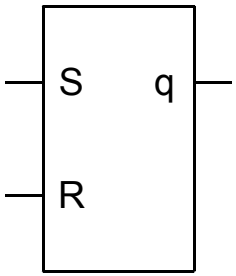


S=0, R=1 cambia a 0 (reset)



Biestable SR. Representación formal

Símbolos



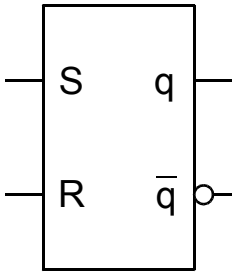


Tabla de estados

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

Q

Diagrama de estados

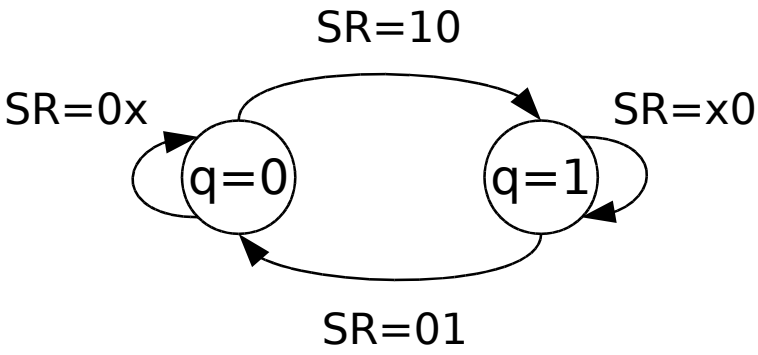


Tabla de excitación

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

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

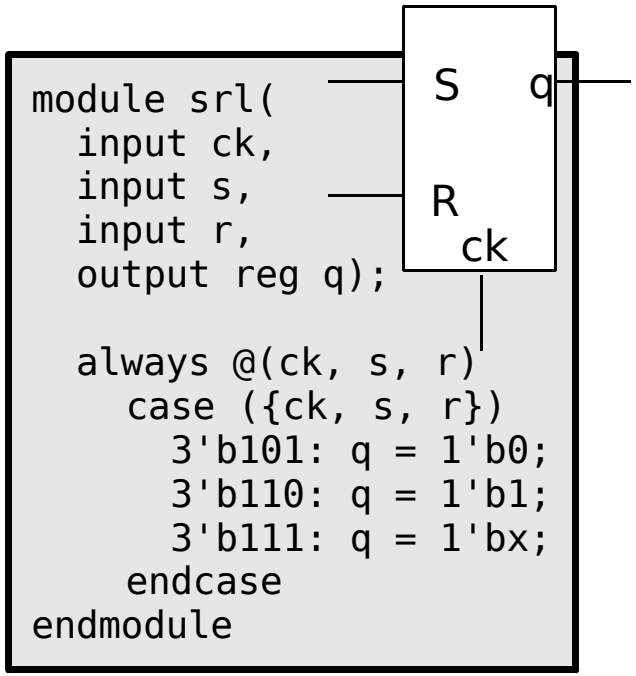
- En circuitos reales con miles (o millones) de biestables es muy útil que todos cambien de estado a la vez: esto simplificará el proceso de diseño.
- Los cambios de estado se producen “sincronizados” con una “señal de reloj” (CK)
- Tipos de sincronización:
  - Por nivel: cuando CK tiene un valor determinado, alto (1) o bajo (0).
  - Por flanco: cuando CK cambia de 0 a 1 (flanco de subida) o de 1 a 0 (flanco de bajada).
- Flanco: más conveniente.

  Determina de forma precisa el instante de cambio

  Minimiza errores en los circuitos

Biestables síncronos

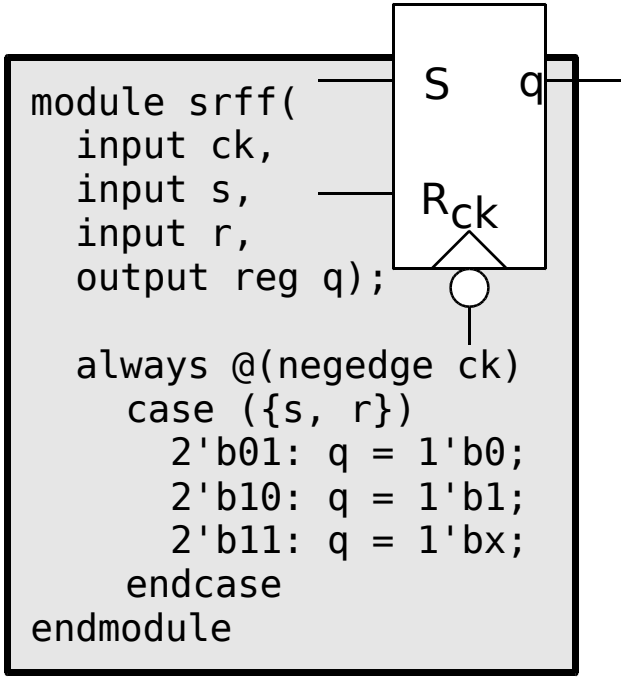
Disp. por nivel




```
module srl(
  input ck,
  input s,
  input 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 se produce cuando ck=1 (nivel alto) o ck=0 (nivel bajo)

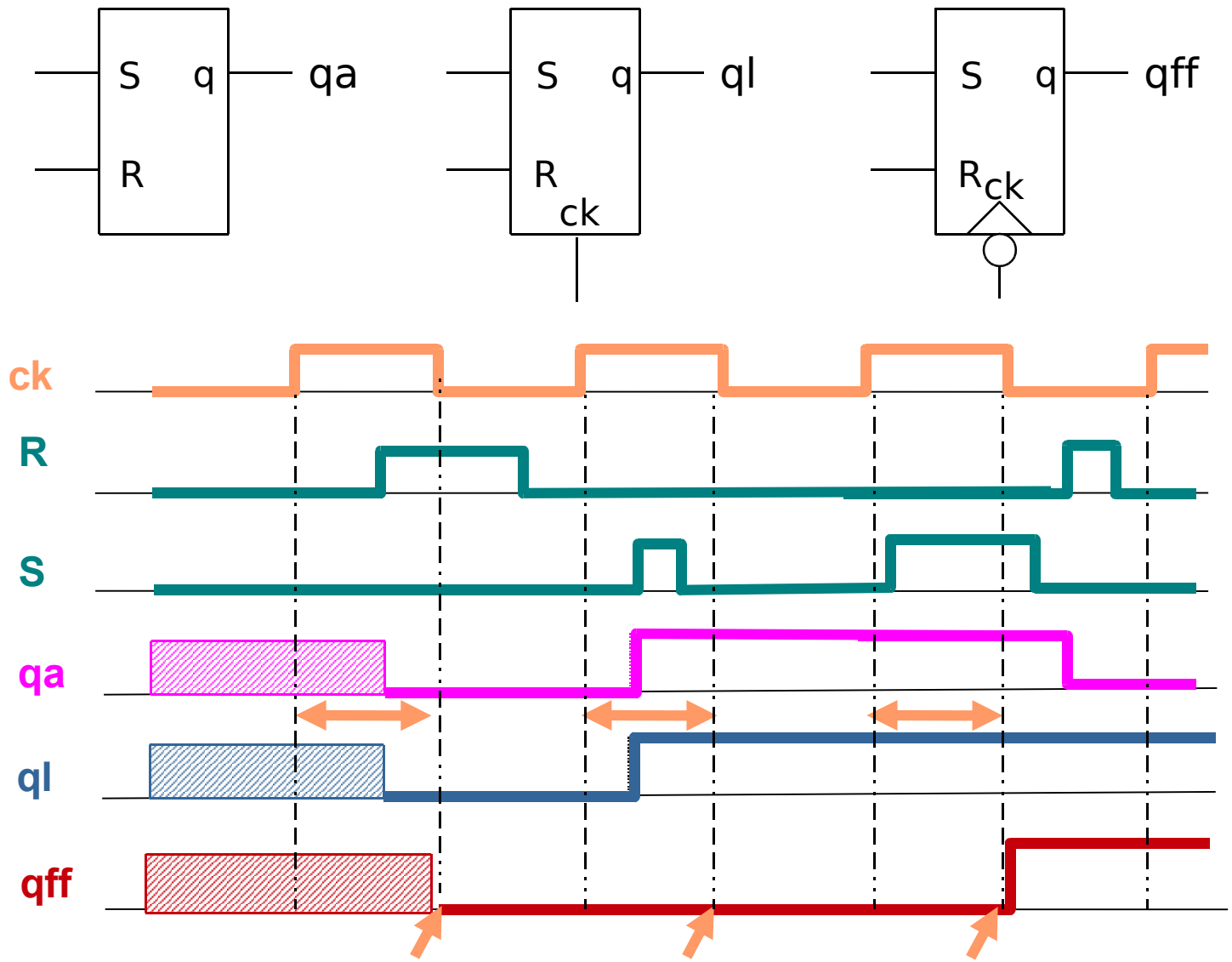
Disp. por flanco




```
module srff(
  input ck,
  input s,
  input 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 se produce 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

Biestables síncronos



Otros biestables síncronos

SR

JK

Similar a SR: J~S, K~R
Función de cambio de estado (toggle) para J=K=1

D

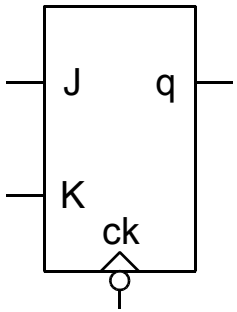
Una única entrada que indica el próximo estado.
Fácil de usar e implementar.

T

Una única entrada que permite complementar el estado.
Útil en aplicaciones especiales.

Biestable JK

Símbolos



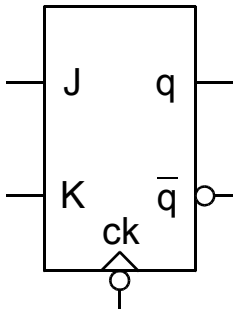


Tabla de estados

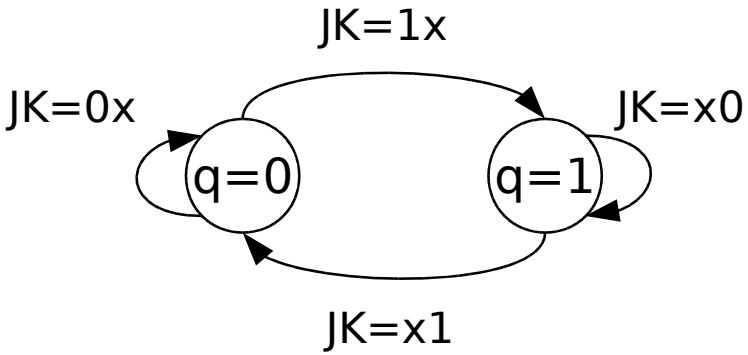
q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Q

Tabla de excitación

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

Diagrama de estados

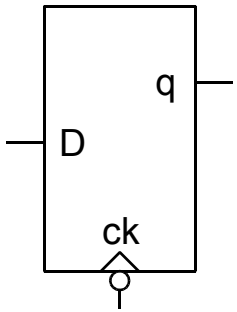


Verilog

```
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



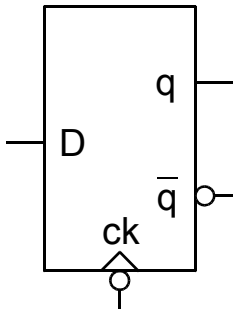


Tabla de estados

q \ D	0	1
0	0	1
1	0	1

Q

Diagrama de estados

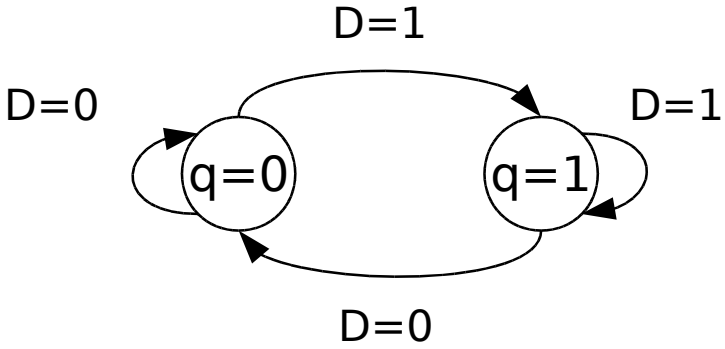


Tabla de excitación

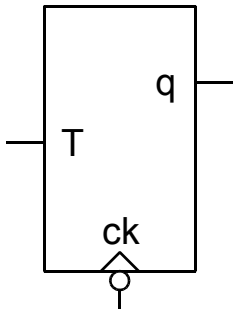
q → Q	D
0 → 0	0
0 → 1	1
1 → 0	0
1 → 1	1

Verilog

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

  always @(negedge ck)
    q <= d;

endmodule
```

Biestable T

Símbolos



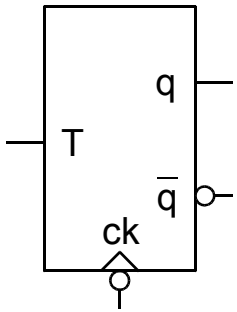


Tabla de estados

q \ T	0	1
0	0	1
1	1	0

Q

Diagrama de estados

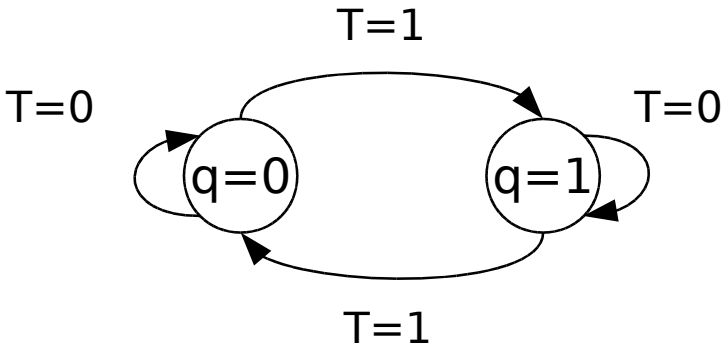


Tabla de excitación

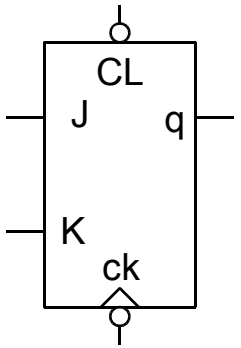
q → Q	T
0 → 0	0
0 → 1	1
1 → 0	1
1 → 1	0

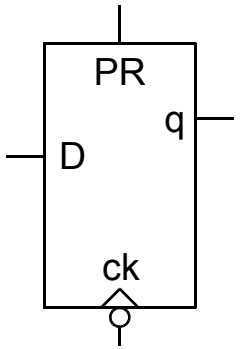
Verilog

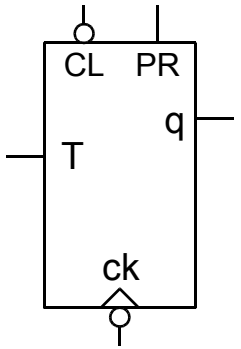
```
module tff(
  input ck,
  input t,
  output reg q);

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

endmodule
```

Entradas asíncronas de los biestables

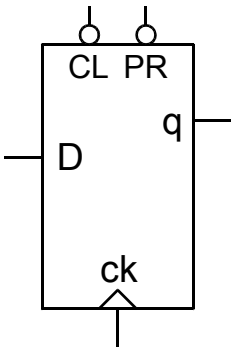
- Permiten cargar un estado determinado de forma sencilla
  - CL (clear): puesta a cero
  - PR (preset): puesta a uno
- Operan inmediatamente cuando se activan:
  - Activas en nivel bajo (0)
  - Activas en nivel alto (1)
- Las entradas asíncronas tienen prioridad sobre las síncronas (J, K, D, T, ...)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

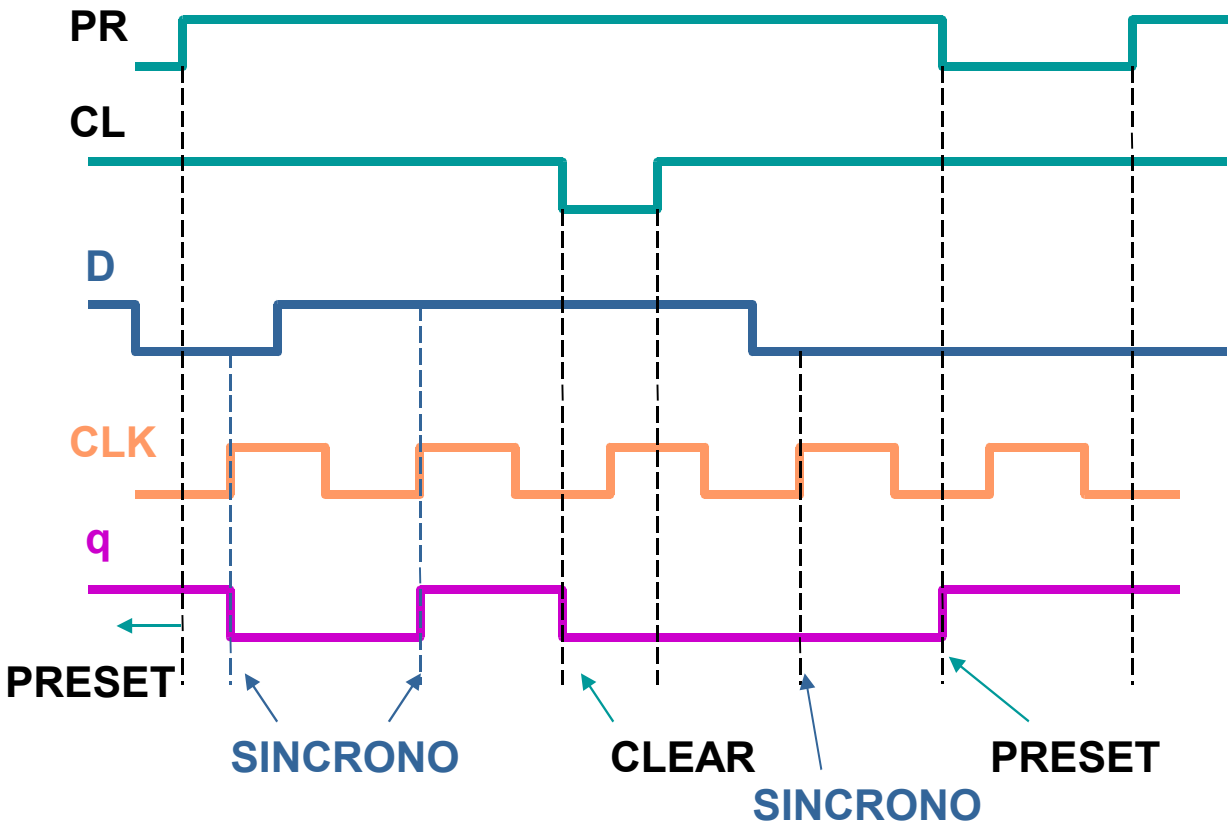






Entradas asíncronas de los biestables

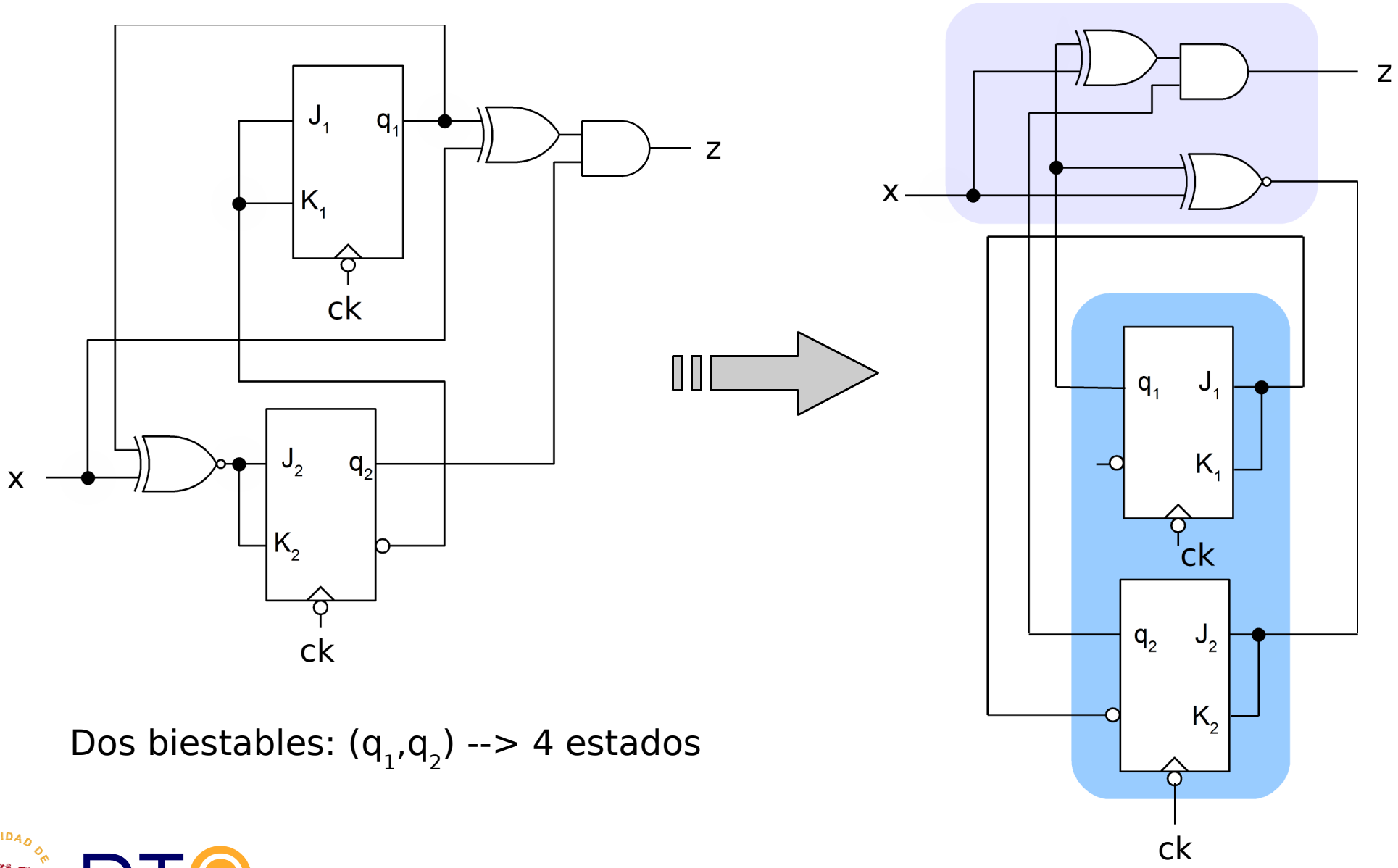




Máquinas de estados finitos y CSS

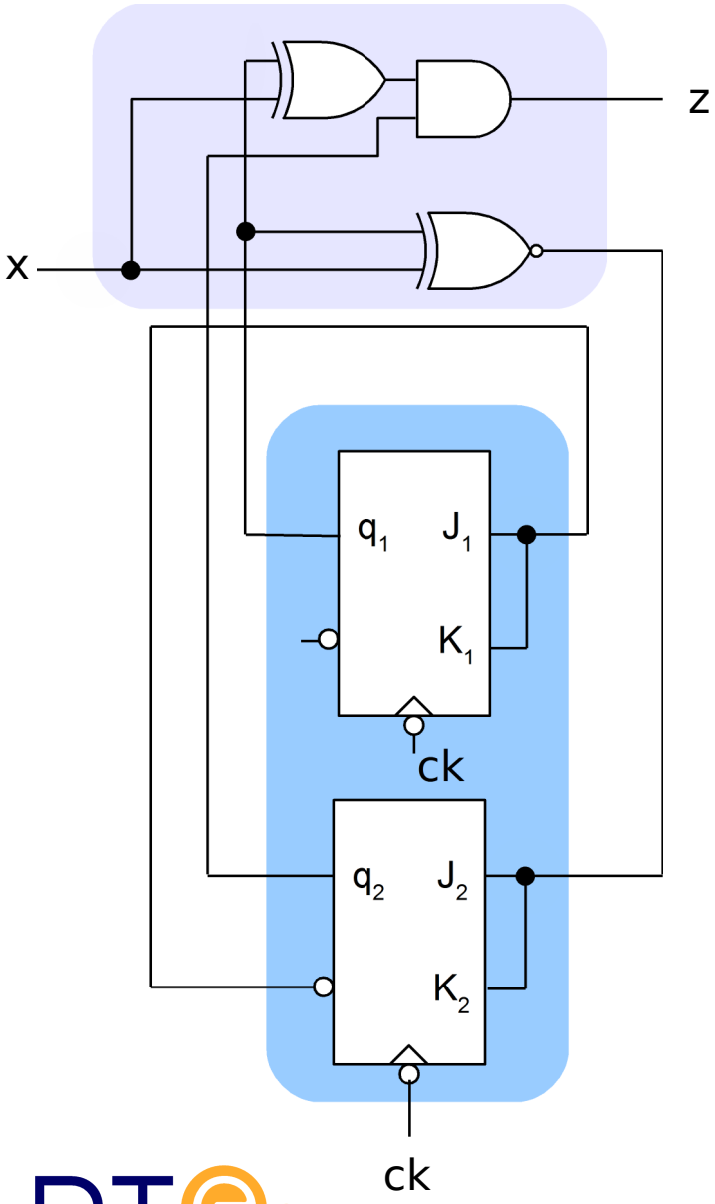
- Introducción
- Biestables
- Máquinas de estados finitos (FSM) y circuitos secuenciales síncronos (CSS)
  - Ejemplo de circuito secuencial
  - Generalización: modelo de máquina de estado finito
- Análisis de CSS
- Diseño de CSS

Ejemplo de circuito secuencial

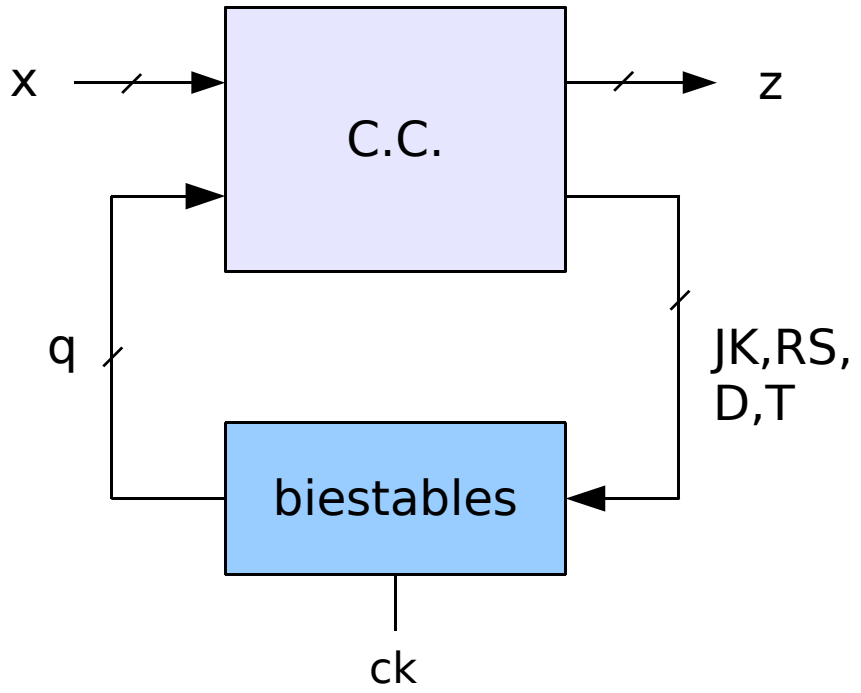


Dos biestables: (q_1,q_2) --> 4 estados

Ejemplo de circuito secuencial



generalización



Concepto de máquinas de estados

Una máquina de estados finitos es una 5-tupla: M(I,O,S,δ,λ) donde

I: Conjunto finito de entradas
($x \in I$)

O: Conjunto finito de salidas
($z \in O$)

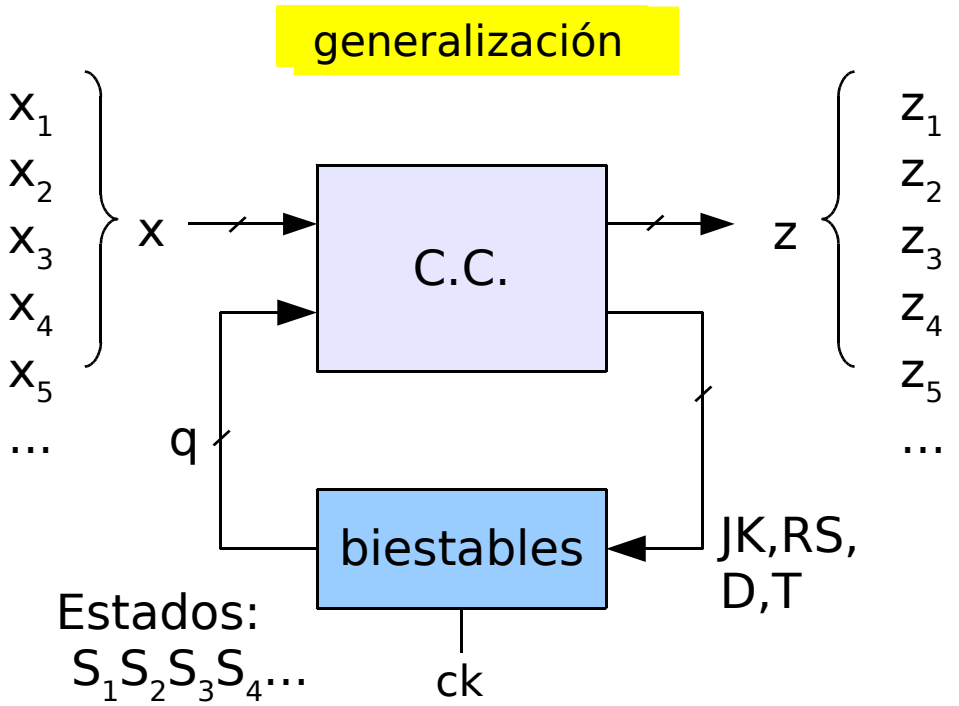
S: Conjunto finito de estados
($S_i \in S$)

δ: Función de próximo estado
($\delta: I \times S \rightarrow S$)): $Q = \delta(q, x)$

λ: Función de salida (λ)

Mealy ($\lambda: I \times S \rightarrow O$): $z = \lambda(q, x)$

Moore ($\lambda: S \rightarrow O$): $z = \lambda(q)$



Concepto de máquinas de estados. Propiedades

Dos máquinas de estados son equivalentes si generan las mismas secuencias de salida para las mismas secuencias de entrada.

Las máquinas de estados se pueden optimizar: máquinas equivalentes con menor número de estados.

Las máquinas de estados pueden ser incompletamente especificadas: próximo estado no definido para un estado actual y entrada dados.

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.

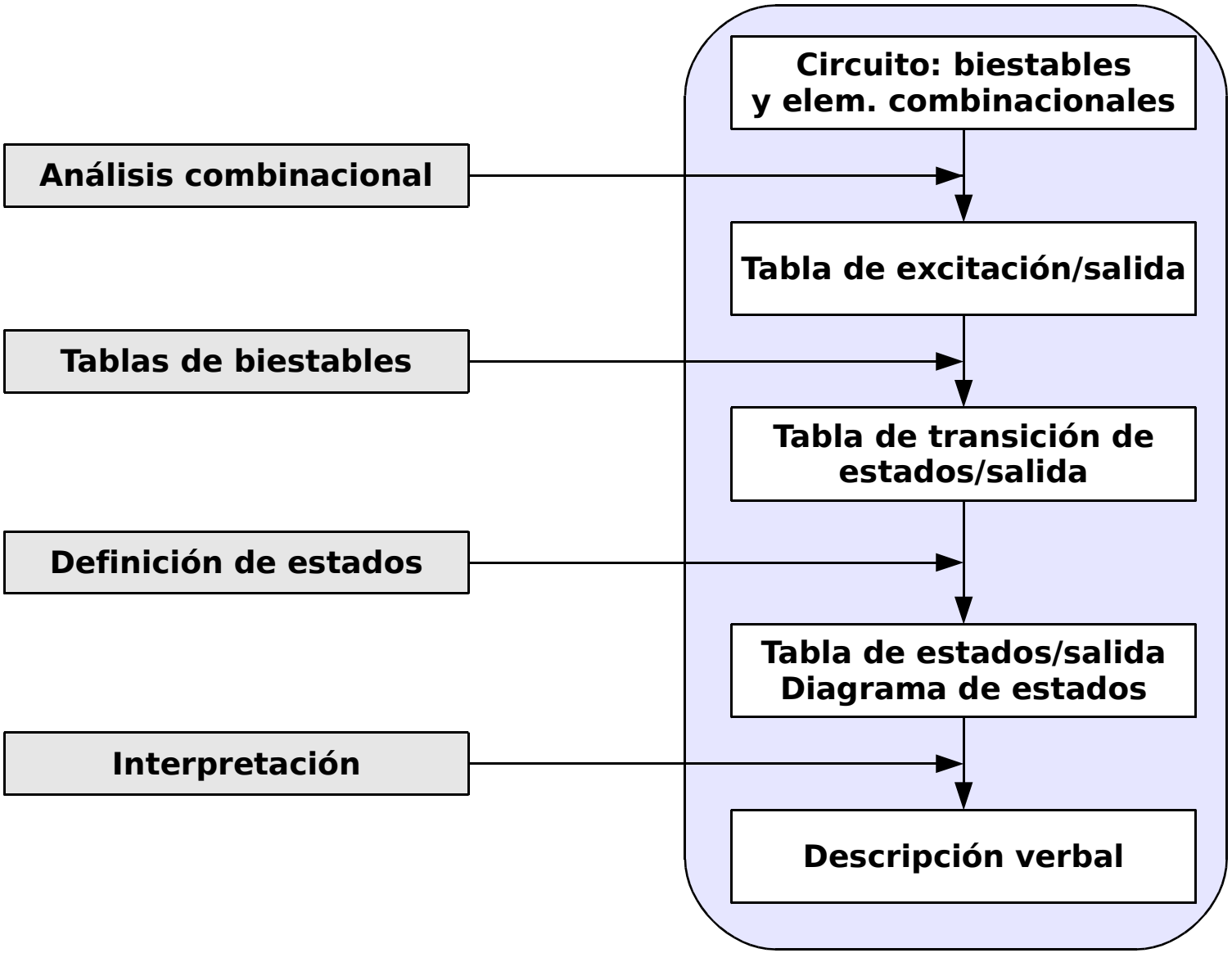
El proceso hasta obtener el diagrama de estados es sistemático.

La interpretación no es sistemática, intervienen:

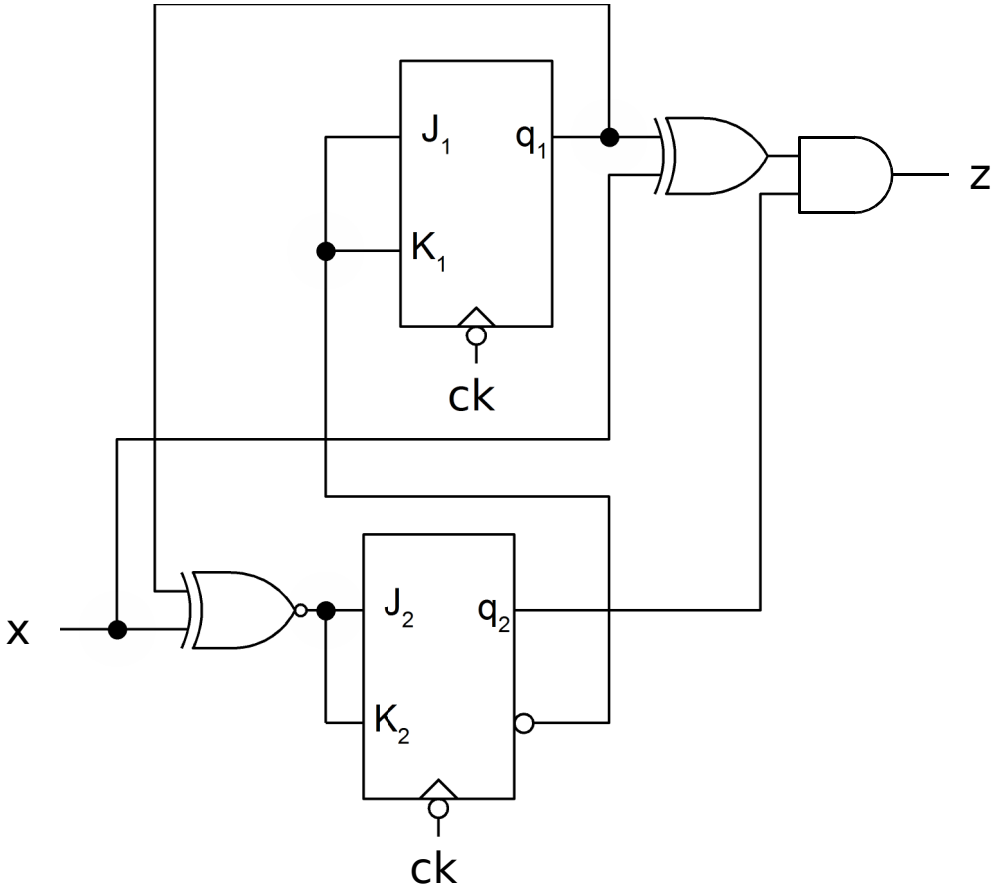
Experiencia

Información adicional

Análisis de CSS: Procedimiento



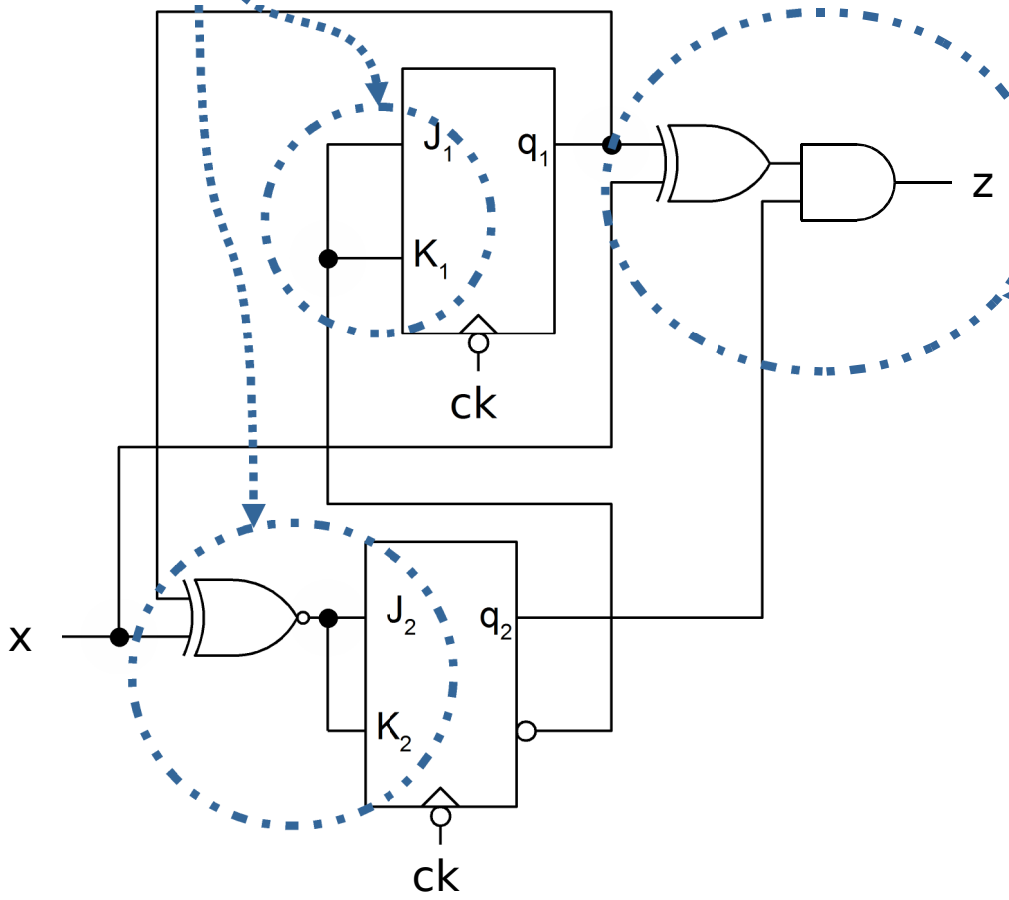
Análisis de CSS: Ejemplo



Análisis de CSS: Ejemplo

$J_2=K_2=(x \oplus q_1)'$

$J_1=K_1=q_2'$



Circuito: biestables y elem. combinacionales

Análisis combinacional

Tabla de excitación/salida

$z= q_2(x \oplus q_1)$

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

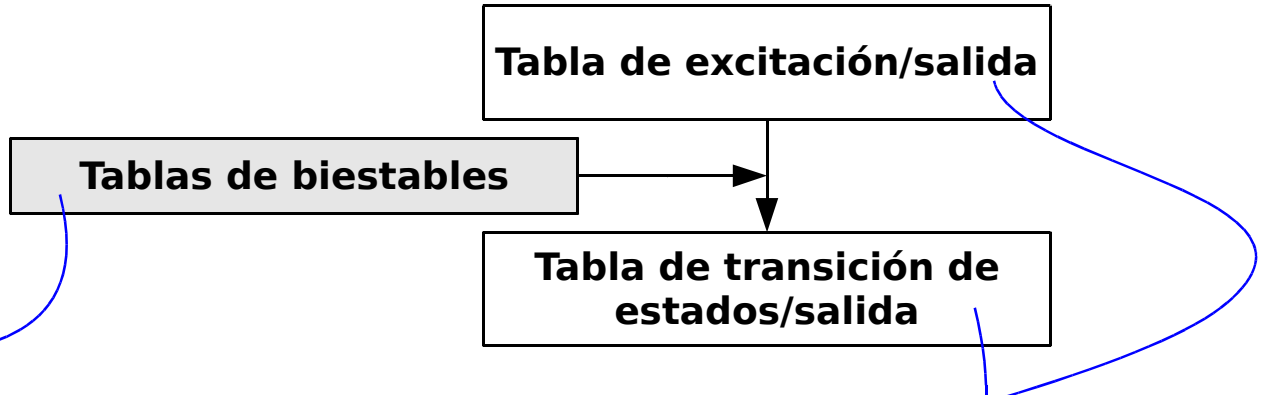
$J_2\ K_2,\ J_1\ K_1, z$

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



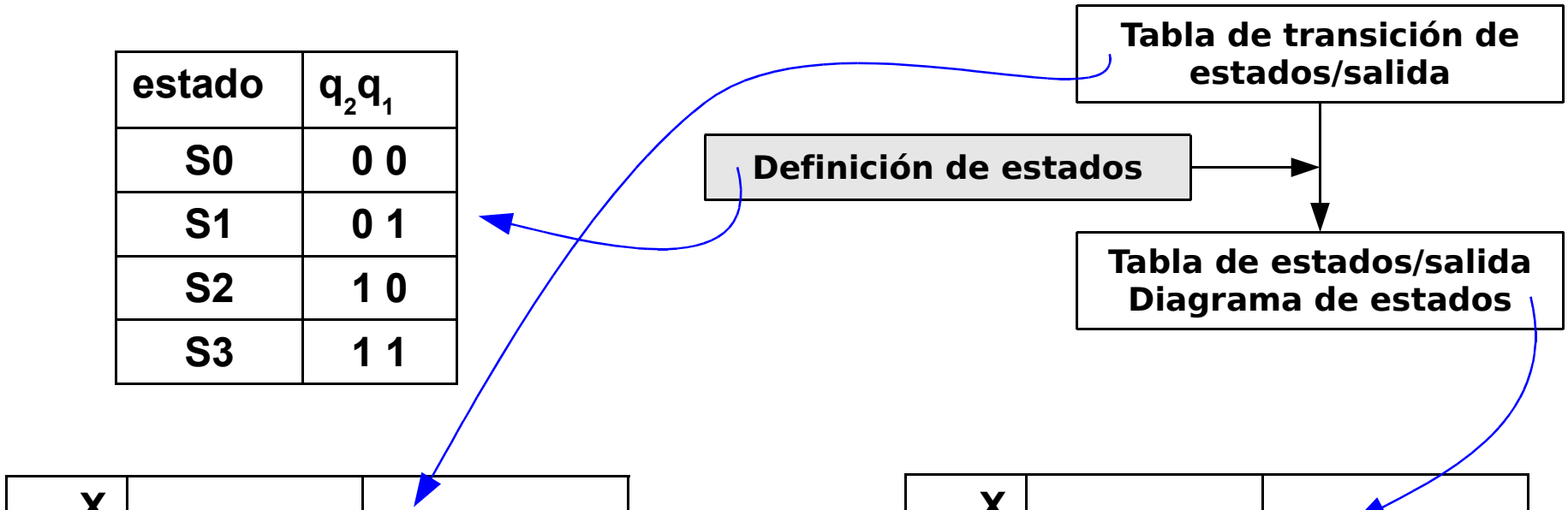
q_2q_1 \ X	0	1
0 0	1 1, 1 1, 0	0 0, 1 1, 0
0 1	0 0, 1 1, 0	1 1, 1 1, 0
1 0	1 1, 0 0, 0	0 0, 0 0, 1
1 1	0 0, 0 0, 1	1 1, 0 0 , 0

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

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

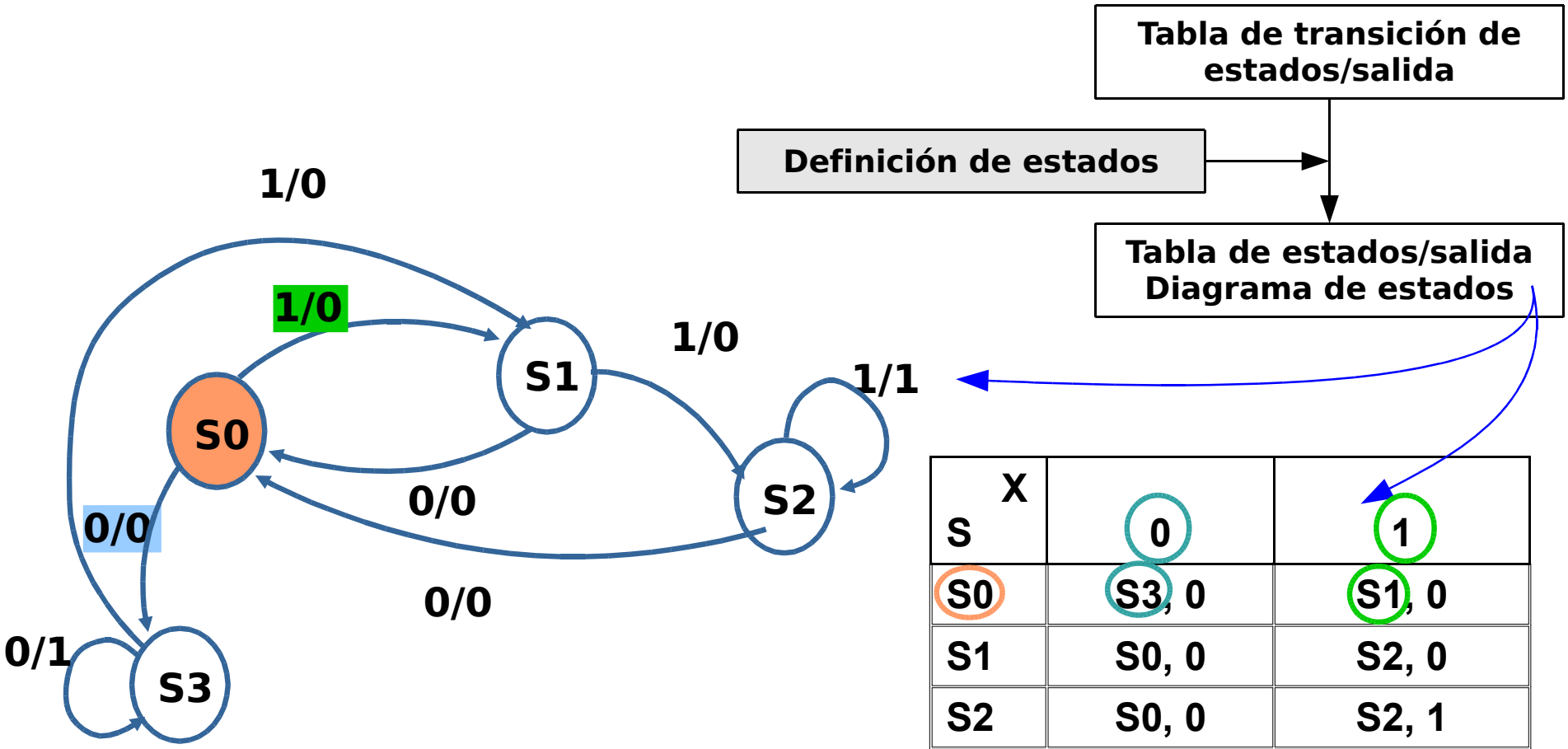
X / q_2q_1	0	1
0 0	1 1, 0	0 1, 0
0 1	0 0, 0	1 0, 0
1 0	0 0, 0	1 0, 1
1 1	1 1, 1	0 1 , 0

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

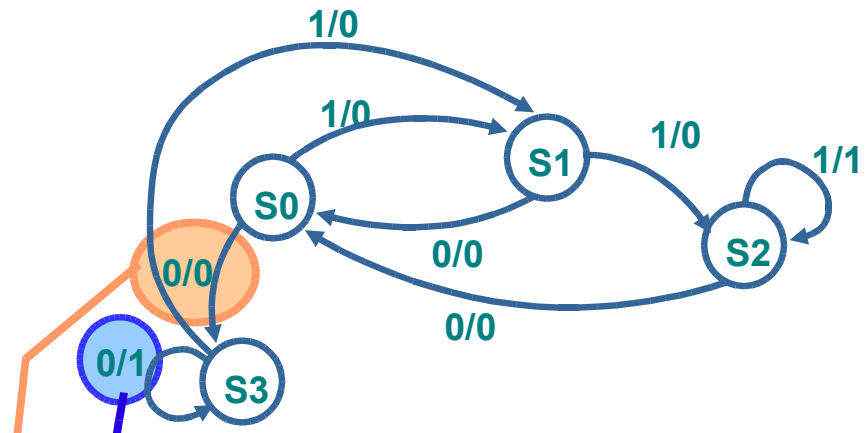
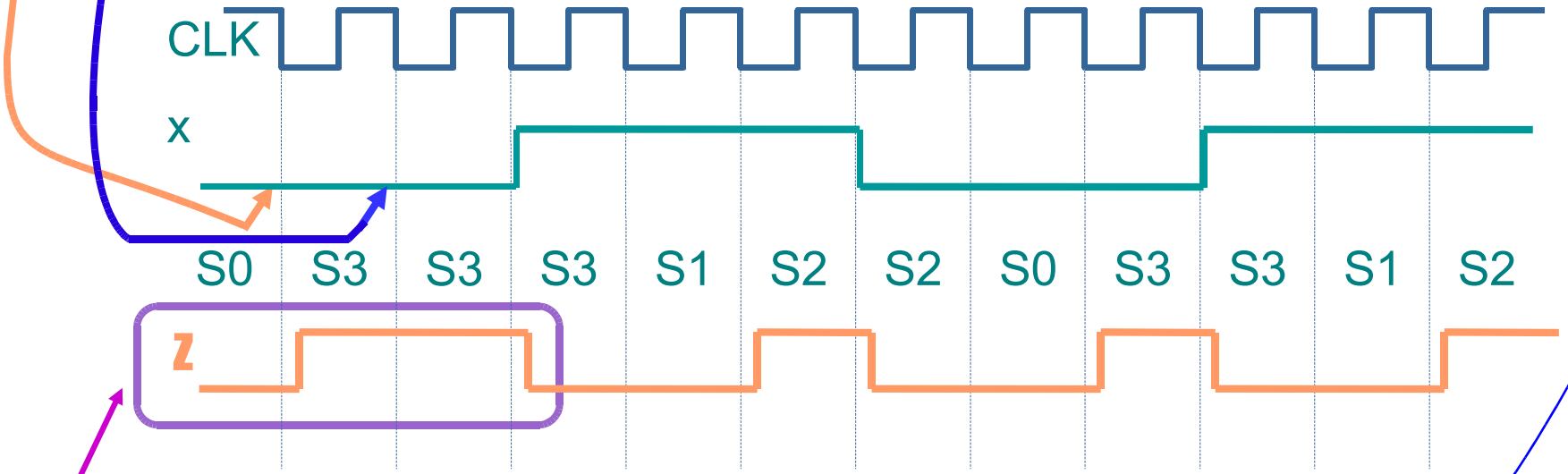


Tabla de estados/salida
Diagrama de estados

Interpretación

Descripción verbal



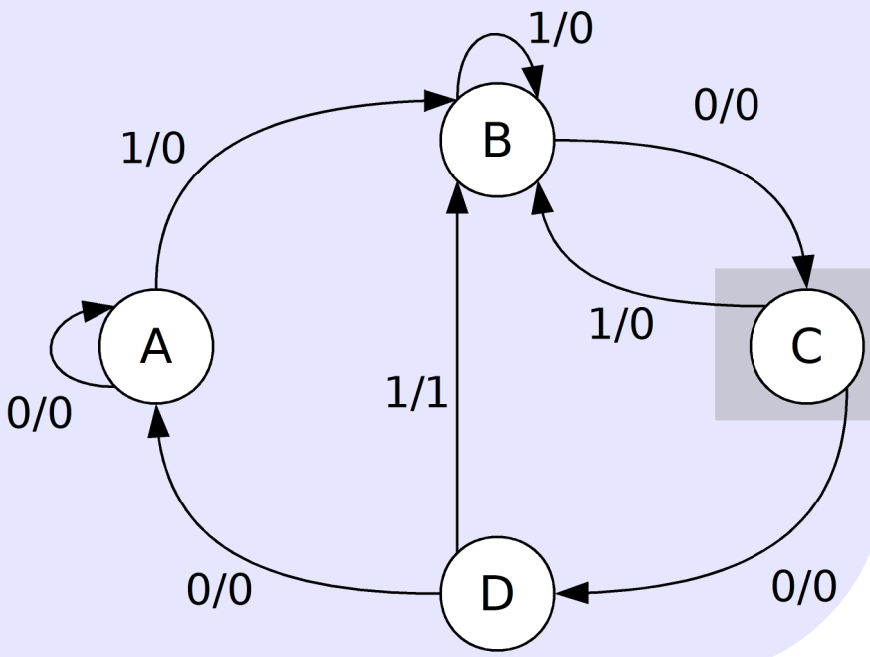
Problemas de inicialización

Función:
detecta 3 ceros o tres unos consecutivos en la entrada

Diagrama/Tabla 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:

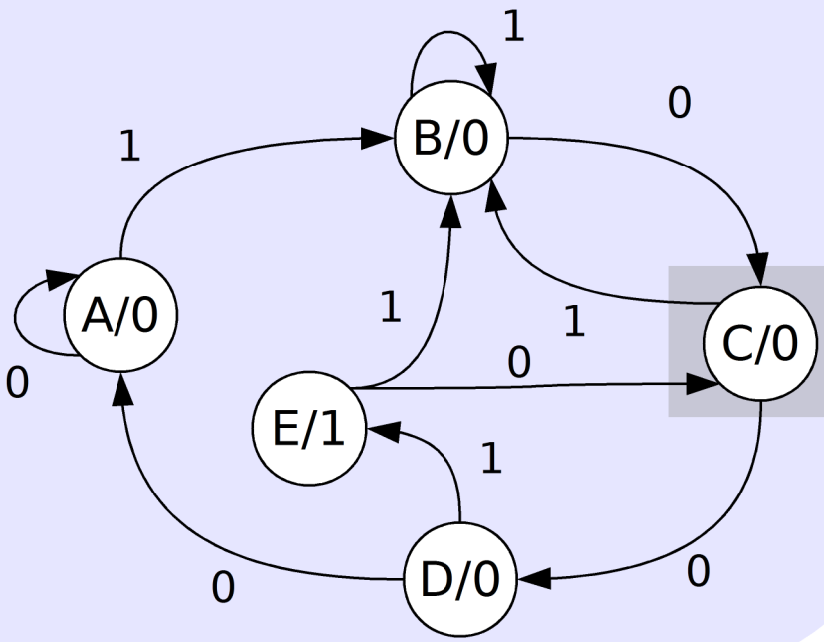
- Posibles estados en filas
- Posibles valores de entradas en columnas
- Próximo estado y salida en cada celda.

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

NS,z

Diagrama/Tabla 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
A	A	B	0
B	C	A	0
C	D	B	0
D	A	E	0
E	C	B	1

NS

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 genera una secuencia fija o variable en función de la entrada.

Unidades de control

Las entradas modifican el estado y el estado define 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, suma de una constante, producto por una constante, codificación/decodificación secuencial en general).

Análisis temporal

Objetivo

- Dado un circuito diseñado (biestables, puertas, etc.), obtener el cronograma de las señales de salida para unas señales de entrada dadas.

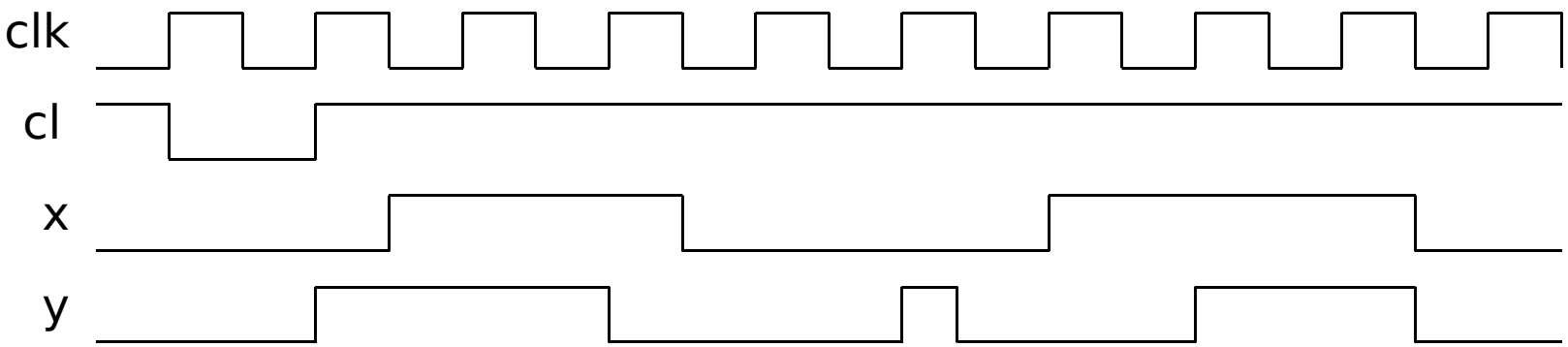
Consideraciones

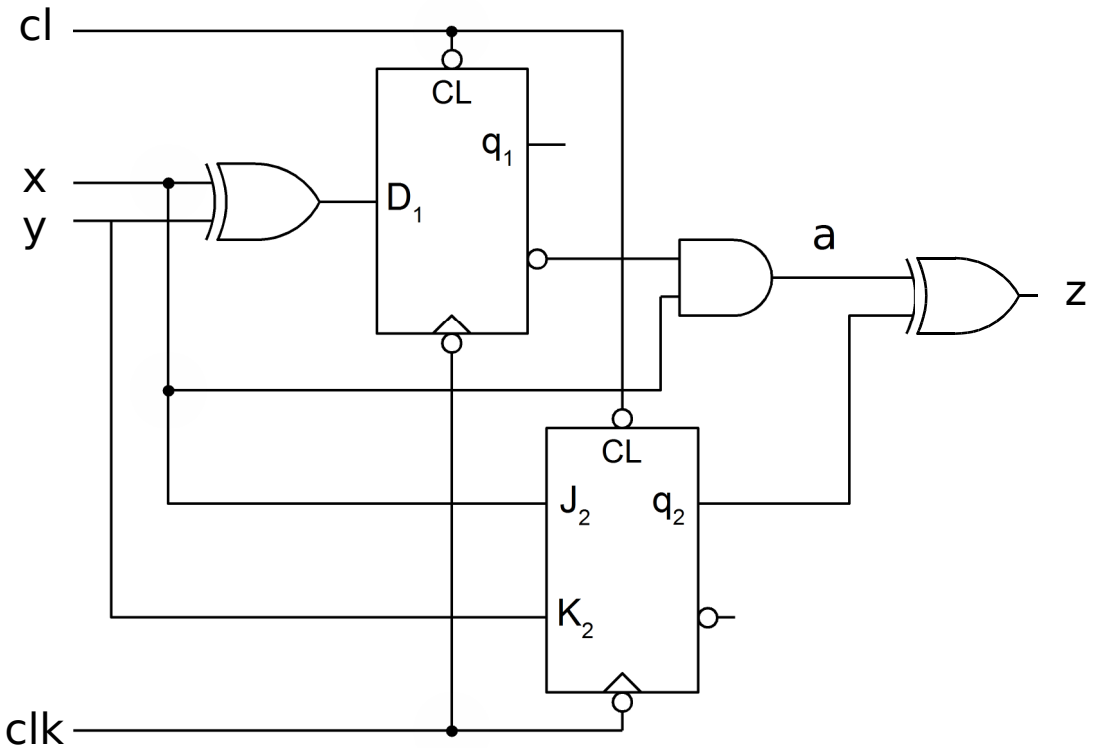
- Es posible analizar circuitos con biestables aunque no sean CSS.
- Si se trata de un CSS, el análisis temporal debe corresponder con la máquina de estados que implementa.

Procedimiento similar al de circuitos combinacionales

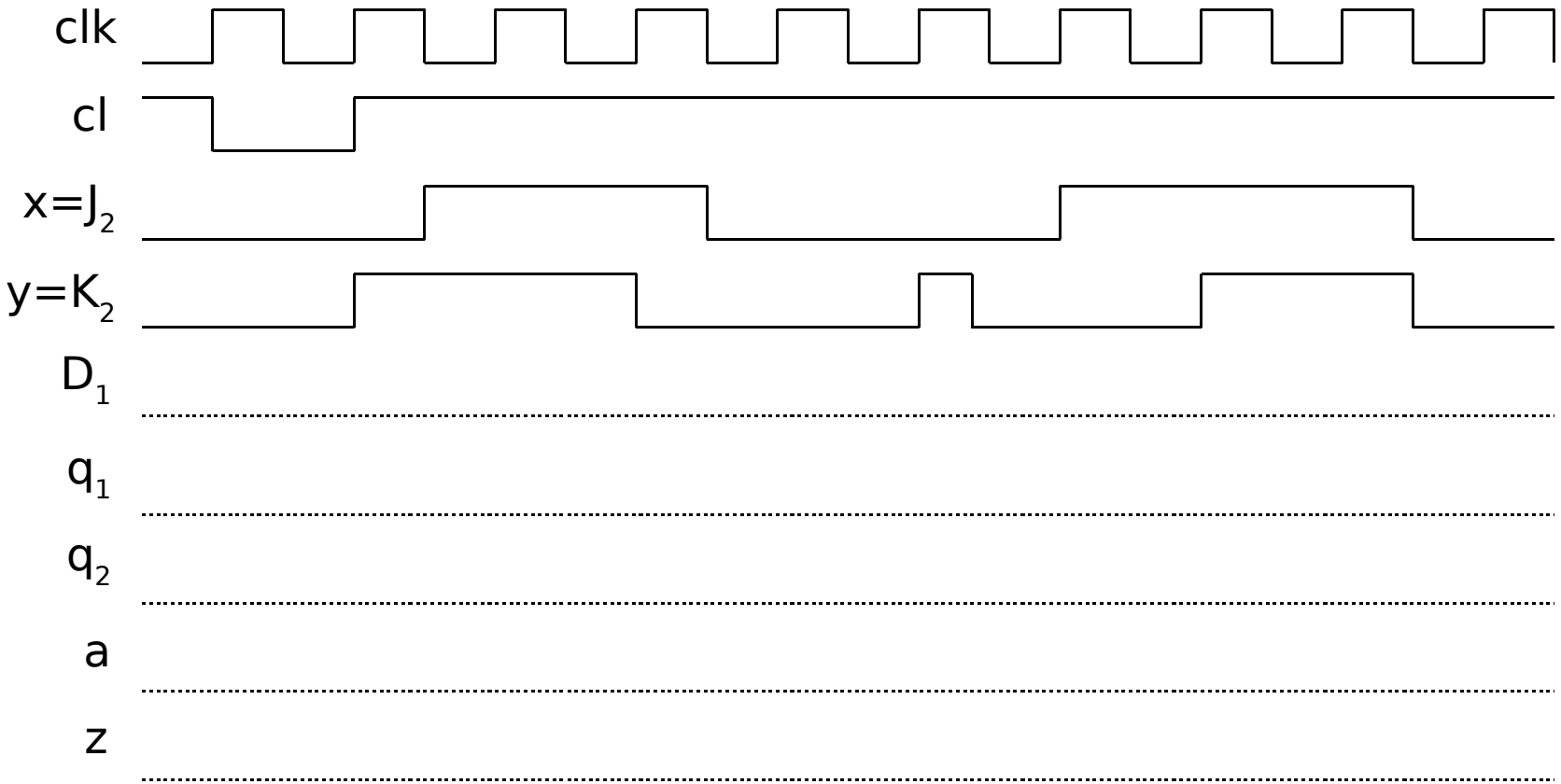
- Parte combinacional: idéntica
- Biestables (por flanco): observando el flanco activo del reloj y calculando la salida (nuevo estado) a partir de la tabla de estados del biestable
- La salida cambia con el retraso definido desde el cambio en el reloj hasta el cambio en el estado (t_{ck-q})

Análisis temporal. Ejemplo





Análisis temporal. Ejemplo



$D_1 = x \oplus y$

$J_2 = x$; $K_2 = y$

$a = q_1\, x$

$z = a \oplus q_2$

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.

Coste

- Habitualmente, el proceso de diseño va dirigido por consideraciones de coste y de optimización de recursos.
- Ejemplo de criterios
  - Minimización del número de elementos de memoria
  - Minimización de componentes
  - Frecuencia de operación
  - Consumo de energía
- Compromiso entre diferentes criterios

Procedimientos

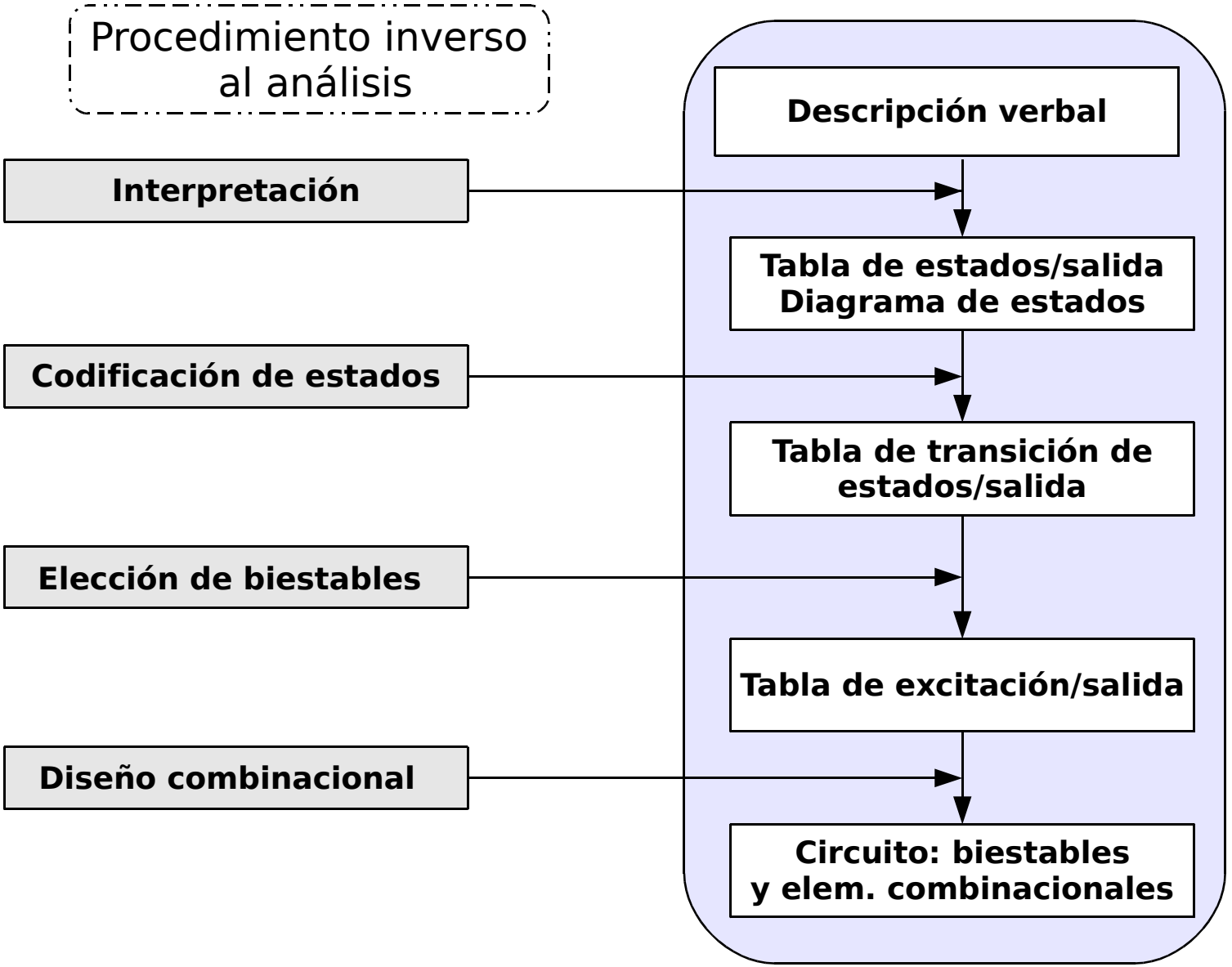
Procedimiento manual

- Realizable con lápiz y papel.
- Comienza describiendo el problema formalmente mediante un diagrama o tabla de estados.
- A partir del diagrama de estados se van obteniendo diversas representaciones hasta llegar al circuito digital.

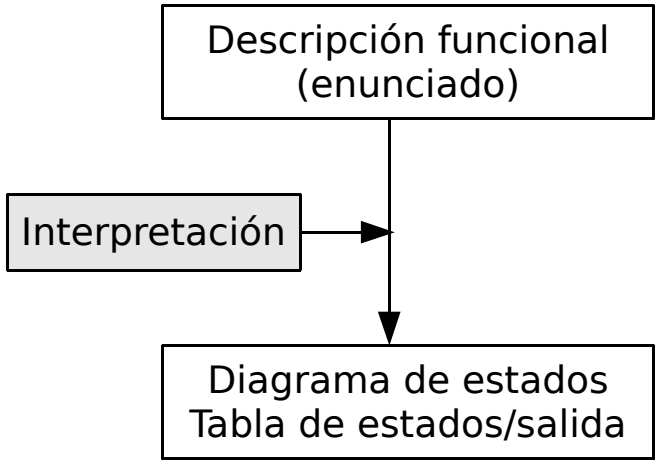
Procedimiento con herramientas de diseño

- 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



Interpretación



Es la fase más importante del diseño

Es la fase menos sistemática

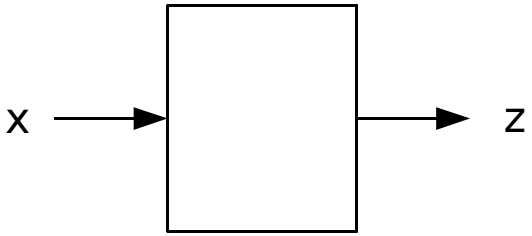
Procedimiento/consejos

- Definir claramente entradas y salidas.
- Elegir Mealy o Moore según características del problema (sincronización de la salida)
- Identificar y definir los estados adecuados de la forma más general posible
- Establecer las transiciones y salidas necesarias
- Capturar todos los detalles del problema en la máquina de estados
- Comprobar el diagrama con una secuencia de entrada típica

Interpretación

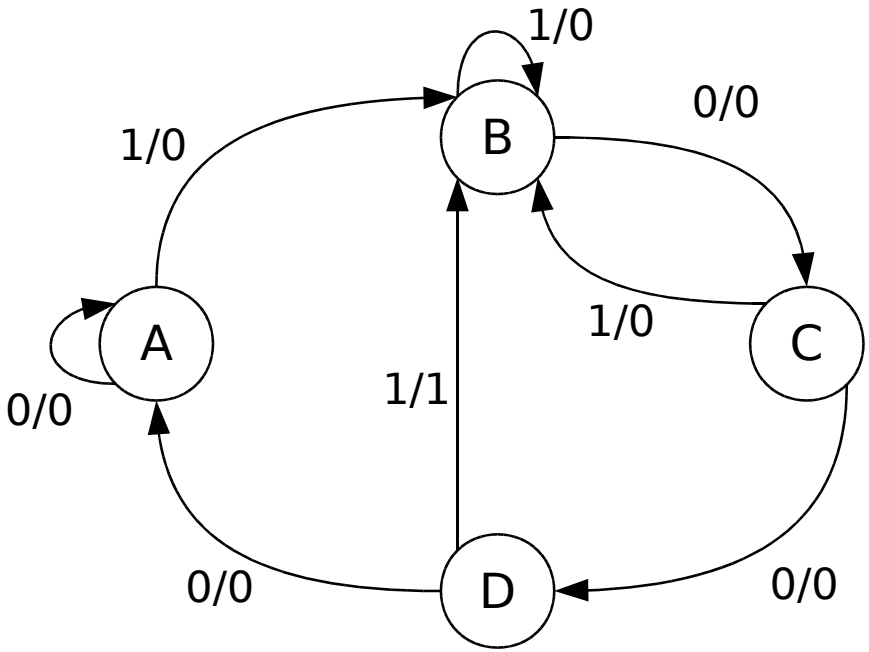
Ejemplo

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

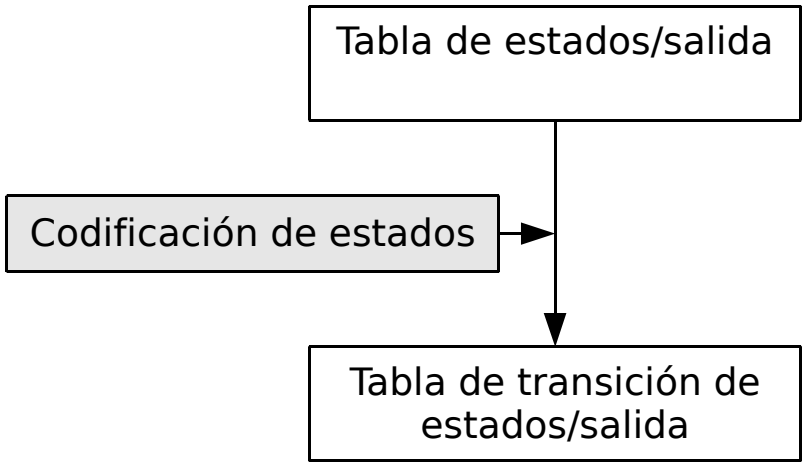


S \ x	0	1
A	A,0	B,0
B	C,0	A,0
C	D,0	B,0
D	A,0	B,1

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 valores binarios a los estados (codificación de estados) para su almacenamiento en biestables.

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)

Opciones

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

Codificación de estados

Tabla de estados/salida

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

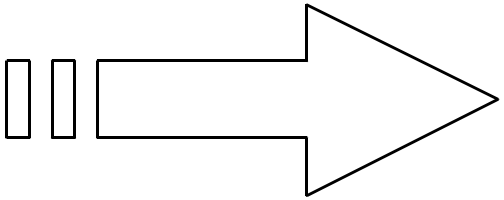
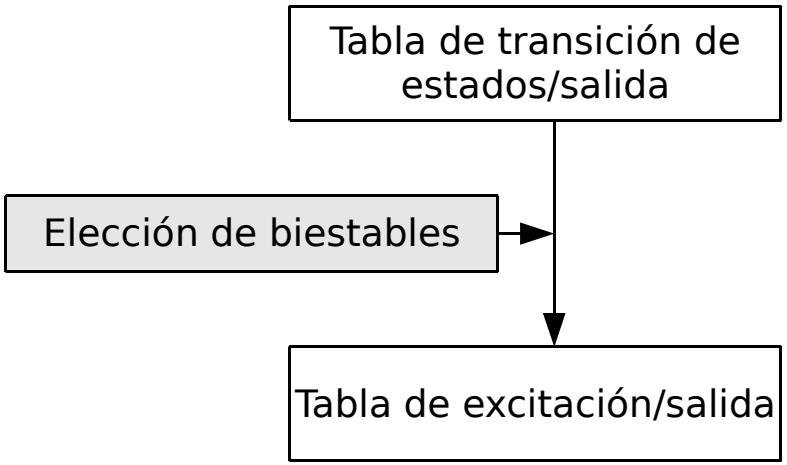


Tabla de transición de estados/salida

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

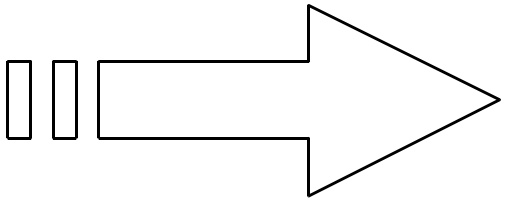


Tabla de excitación/salida

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

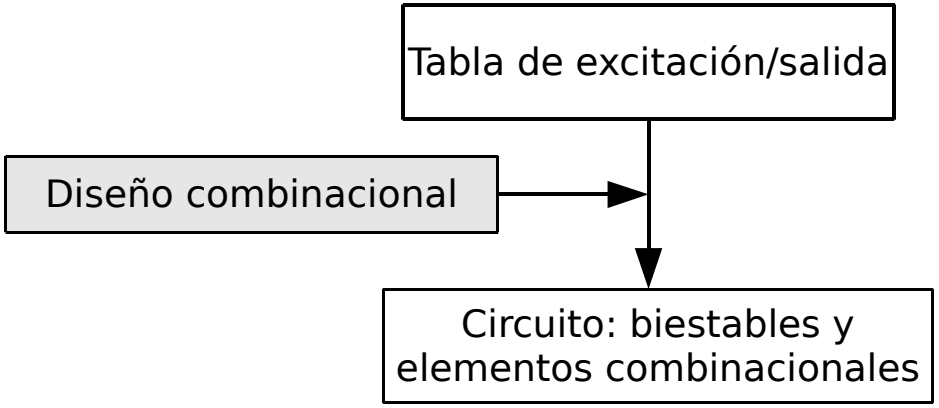
Tabla de transición de estados/salida ⟺ Tabla de excitación/salida

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

Q_1Q_2,z

D_1,D_2,z

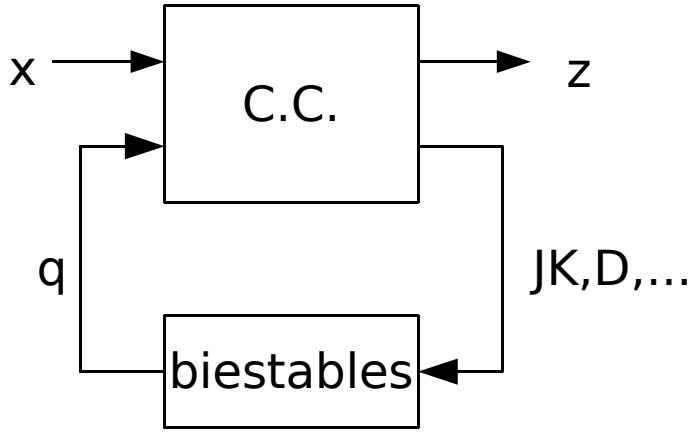
Diseño de la parte combinacional



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,x1,0
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	1
11	1	0
10	x	x

K_2

$J_1 = x'q_2$

$K_1 = x + q_2'$

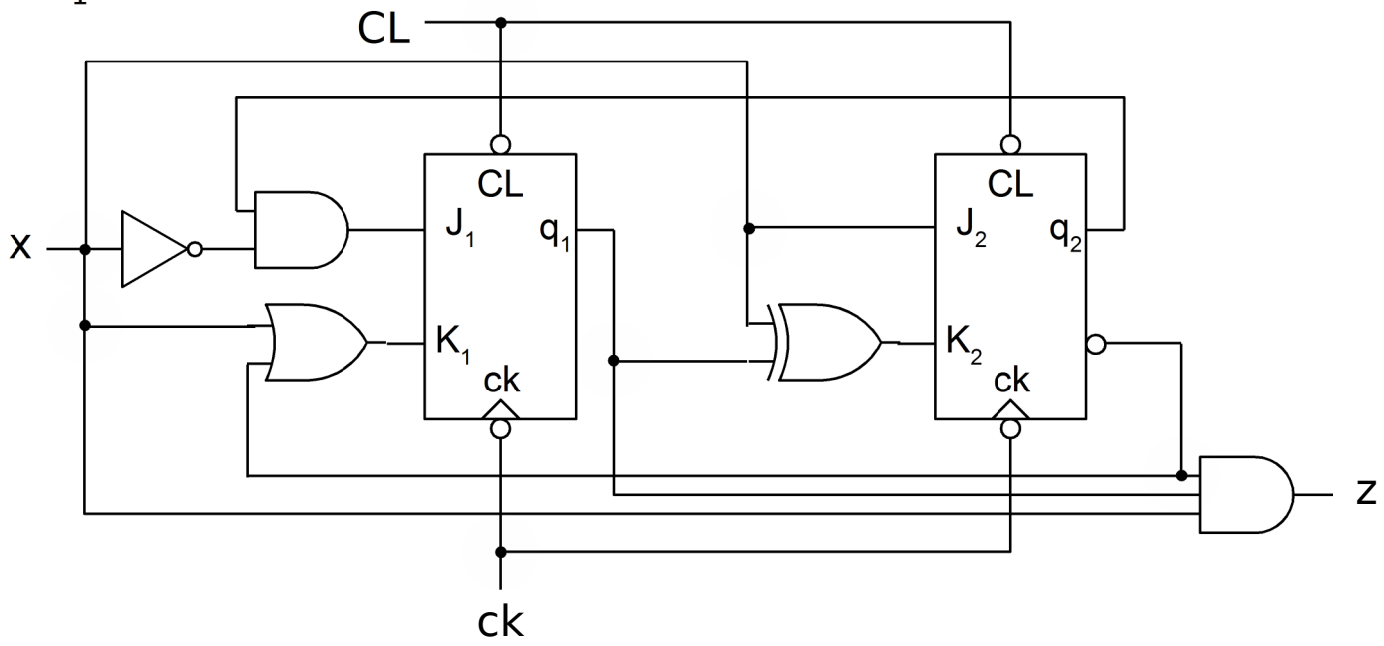
$J_2 = x$

$K_2 = x\,q_1' + x'\,q_1$

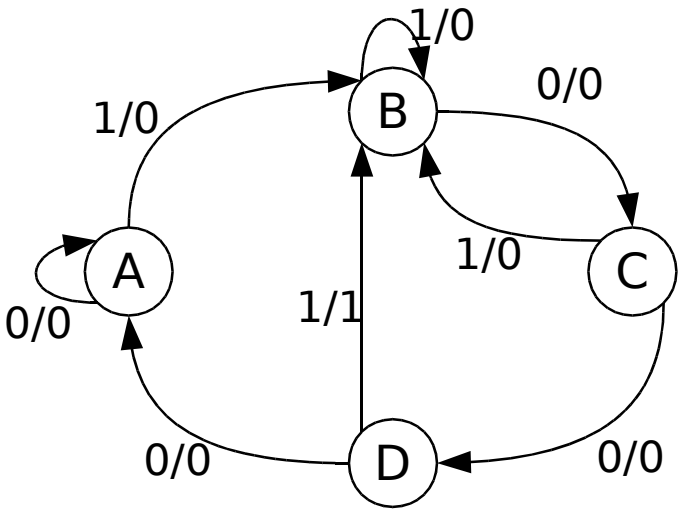
$z = xq_1q_2'$

Circuito. Ejemplo

$J_1 = x'q_2$
$K_1 = x + q_2'$
$J_2 = x$
$K_2 = x\,q_1' + x'\,q_1$
$z = xq_1q_2'$



Ejemplo. Resumen



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

Q,z

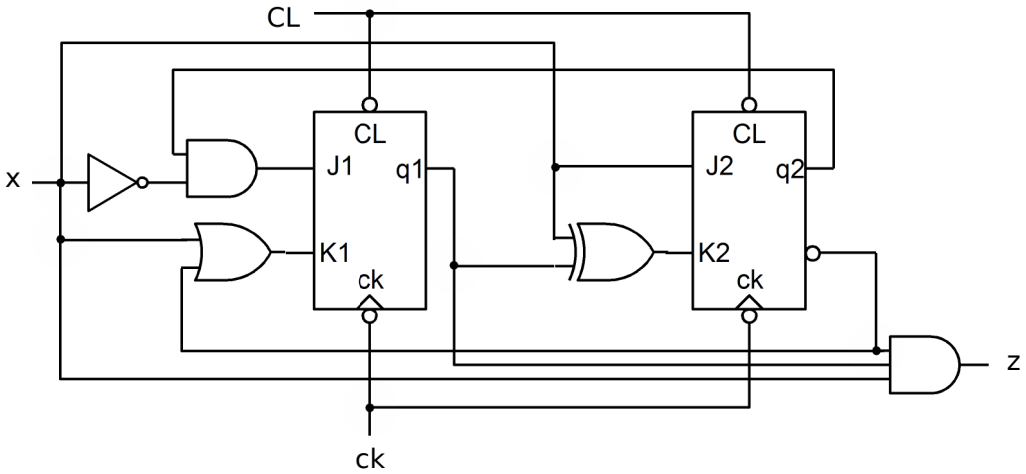
q_1q_2 \ x	0	1
00	00,0	01,0
01	11,0	00,0
11	10,0	01,0
10	00,0	01,1

Q,z

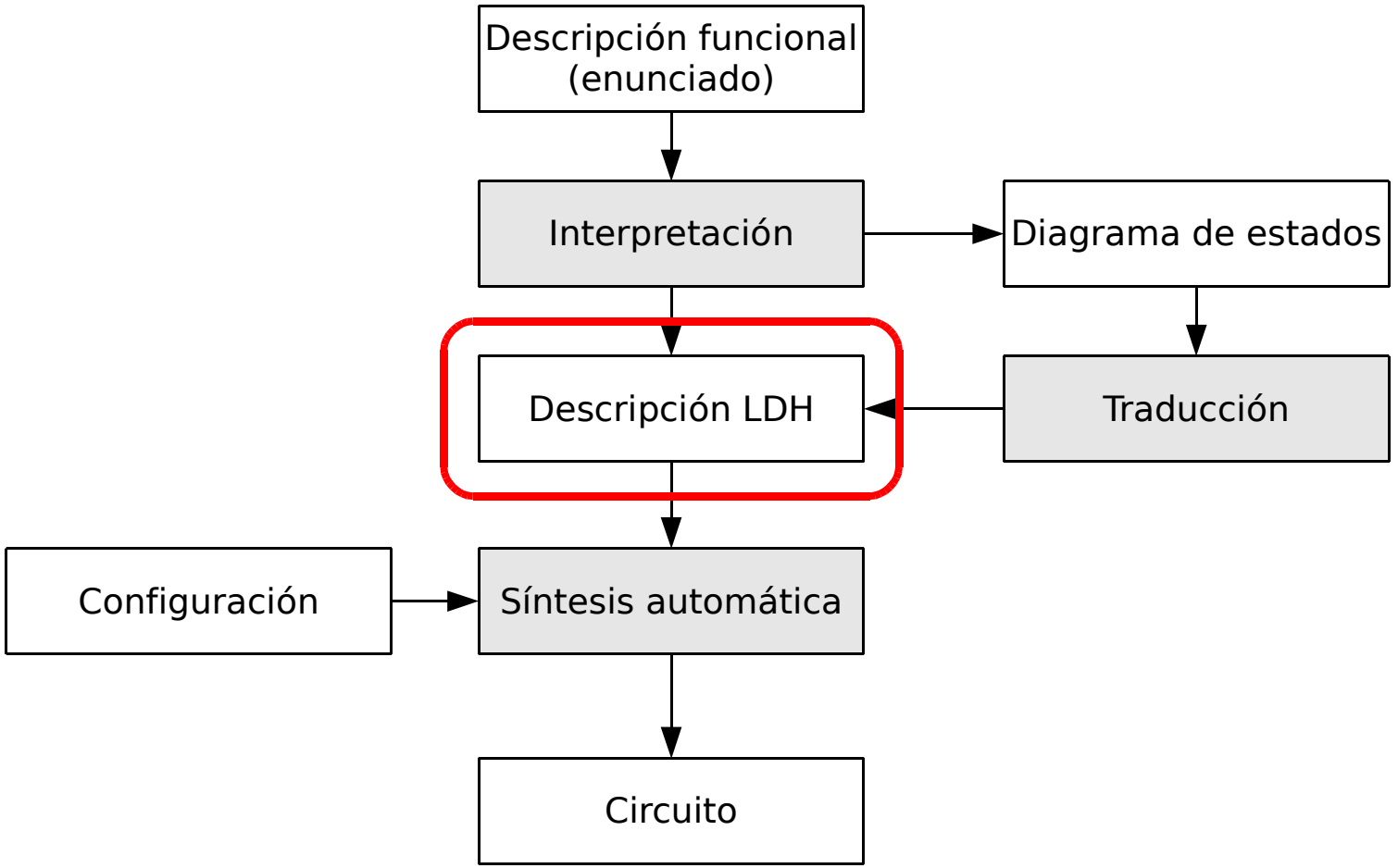
q_1q_2 \ x	0	1
00	0x,0x,0	0x,1x,0
01	1x,x0,0	0x,x1,0
11	x0,x1,0	x1,x0,0
10	x1,0x,0	x1,1x,1

J_1K_1,J_2K_2,z

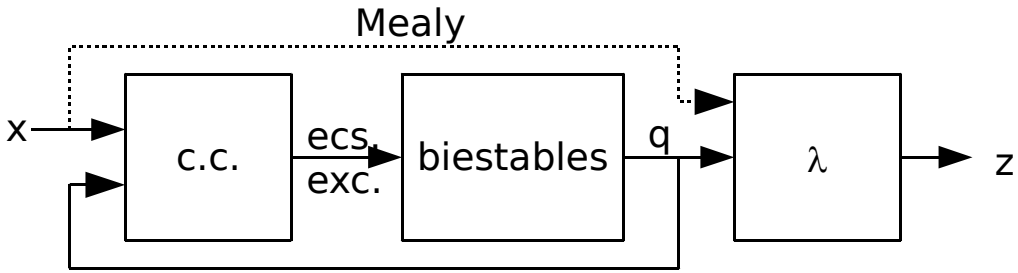
$J_1 = x'q_2$
$K_1 = x + q_2'$
$J_2 = x$
$K_2 = x\,q_1' + x'\,q_1$
$z = xq_1q_2'$



Procedimiento con herramientas de diseño



Descripción de FSM en Verilog

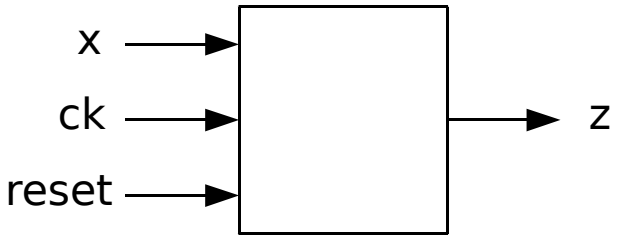


Dos procesos

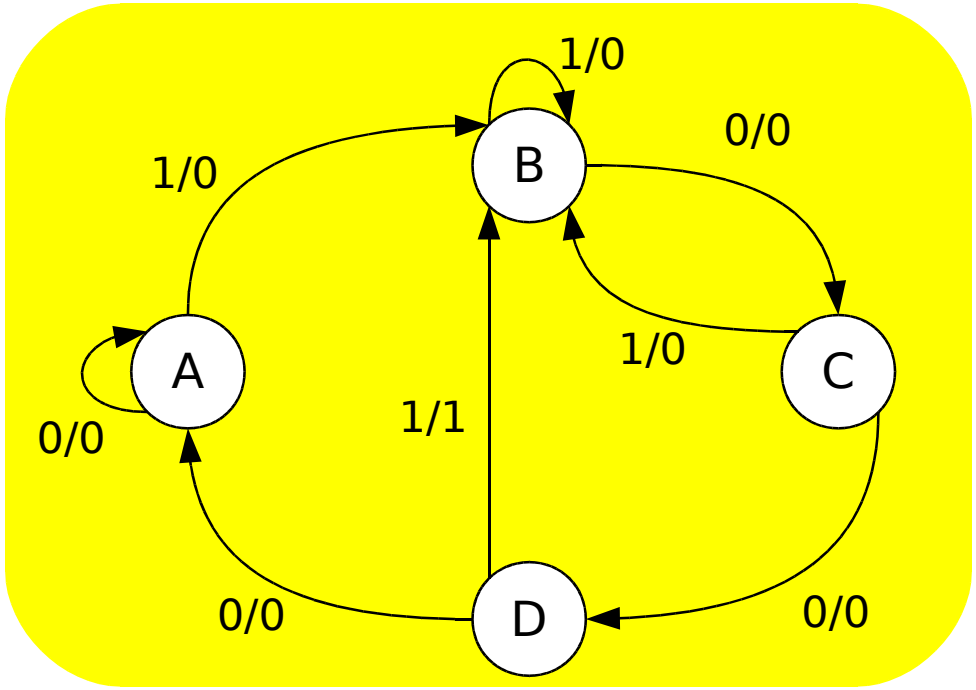
- **Cambio de estado:** representa el bloque de biestables
- **Cálculo del próximo estado** y salida(ecuaciones de excitación y salida)
- **Sólo el proceso de cambio de estado es secuencial**

```
// 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
```

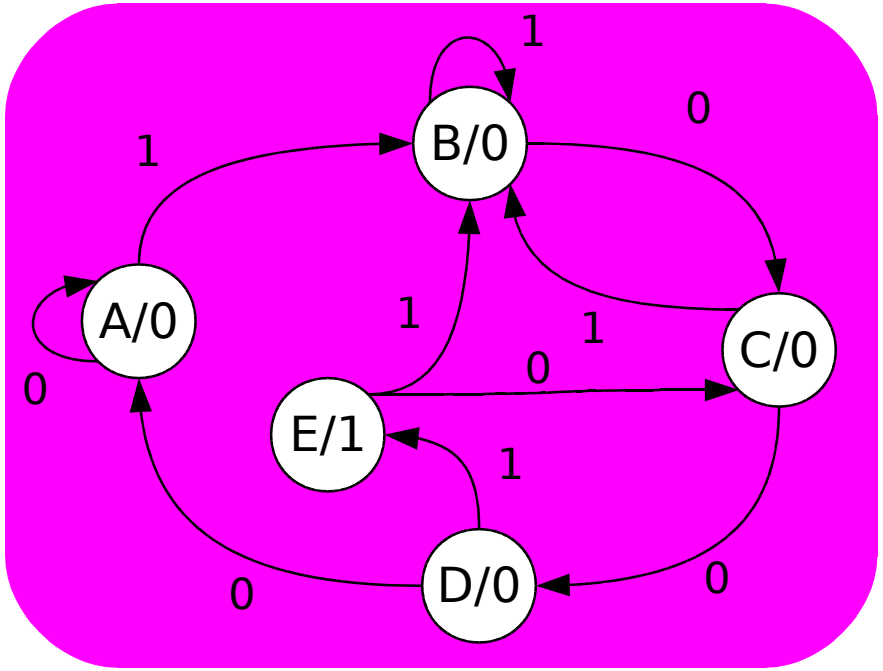
FSM en Verilog. Ejemplo



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



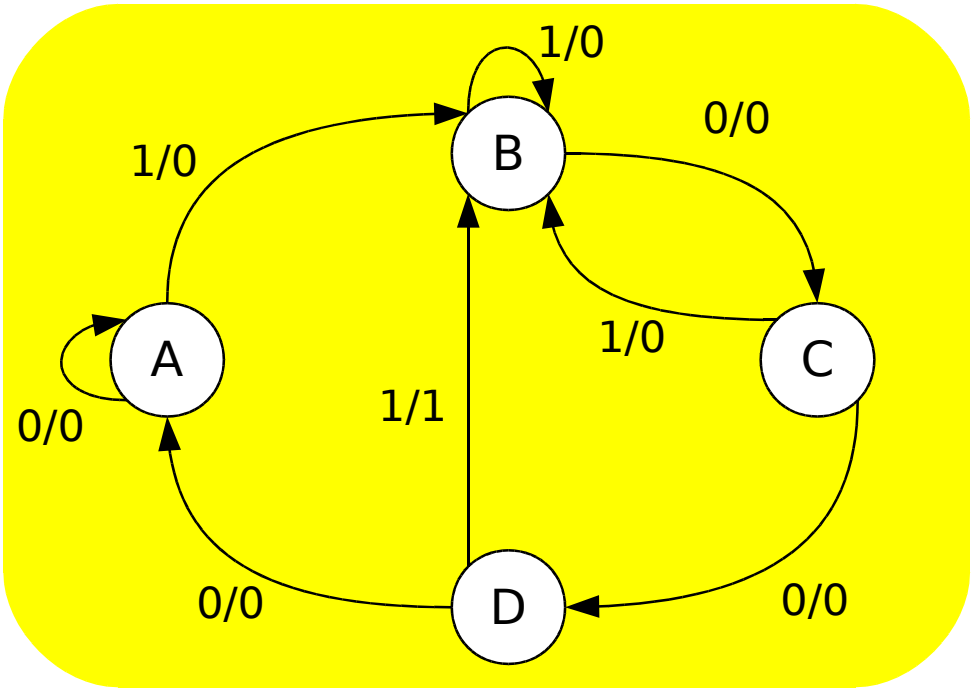
Solución con máquina de Mealy



Solución con máquina de Moore

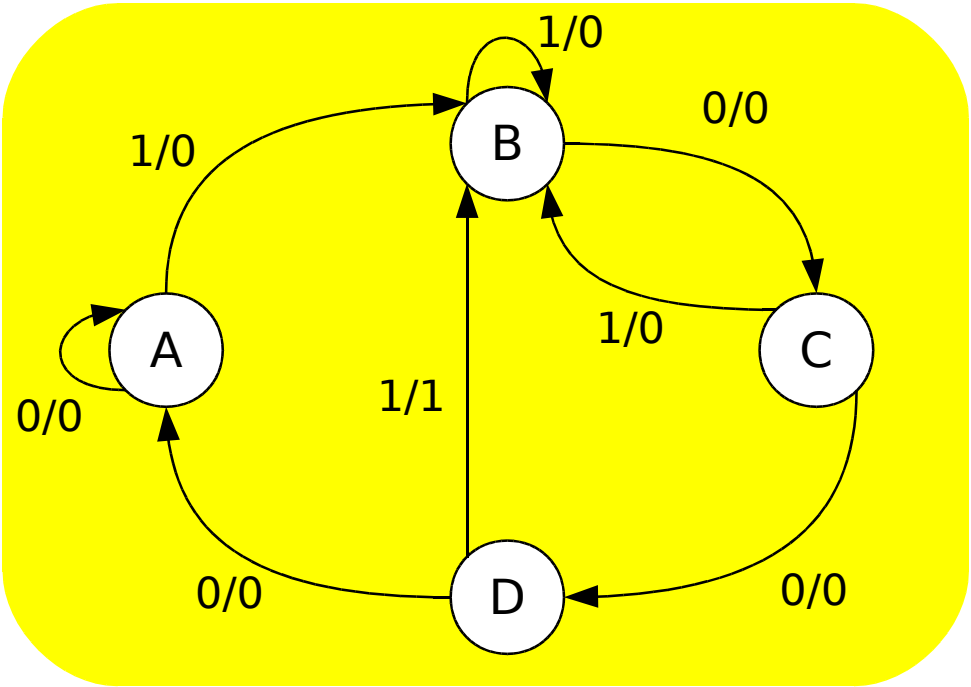
FSM en Verilog. Ejemplo Mealy.

```
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;
```



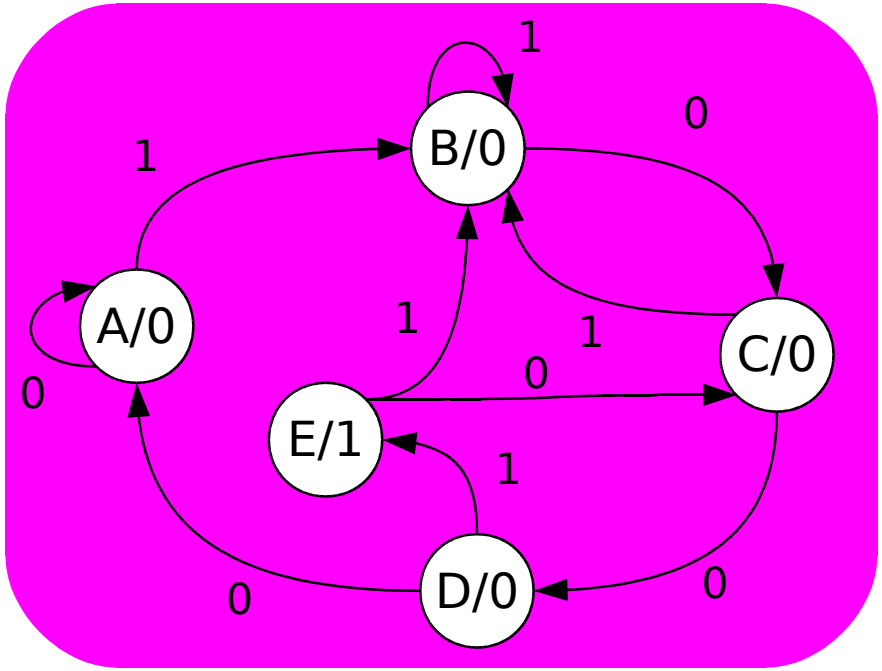
FSM en Verilog. Ejemplo Mealy.

```
// Proceso de cálculo del nuevo estado  y la
salida (combinacional)

     always @* begin
          z=0;
          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
                    begin
                         next_state = B;
                         z=1;
                    end
          endcase
     end
```



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
        z=0;
        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 = E;
        E:  z=1;
            if (x == 0)
                next_state = C;
            else
                next_state = B;
        endcase
    end
```