UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
CAMPUS SAN JOAQUÍN

Arquitectura y Organización de Computadores

2023

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
CAMPUS SAN JOAQUÍN

Sistemas Secuenciales

Índice

- ✓ Introducción
- ✓ Latches & Flip Flop’s
- ✓ Diseño Sincrónico
- ✓ Máquinas de Estados Finitas (FSM)

Introducción

- ✓ Los sistemas digitales más simples corresponden a los sistemas combinacionales.
- ✓ En los SC la salida depende únicamente del valor de las entradas en el estado actual.
- ✓ Los SC no tienen memoria y solo realizan y/o modelan funciones lógicas.
- ✓ Los SC conforman en gran parte el contenido de un chip VLSI.

Introducción

- ✓ Los Sistemas Secuenciales son un poco más complejos, ya que permiten **memorizar información** a través de sus estados internos.

- ✓ La salida de un SS depende del estado actual y de un set de variables de entrada priorizadas según la lógica del problema que modela.

- ✓ El estado interno de un SS corresponde a un set de variables (bits) que contienen toda la información necesaria sobre el pasado para modelar el comportamiento del futuro.

Introducción

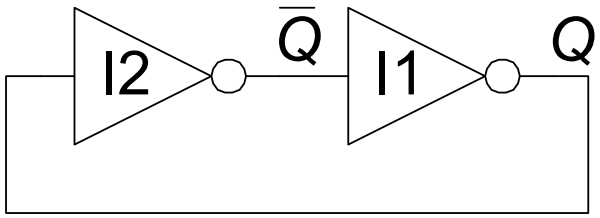
- ✓ Para construir los SS utilizaremos los componentes denominados Latches & Flip Flop’s. Ambos componentes corresponden a un circuito combinacional simple que permite almacenar el comportamiento de un bit del estado.

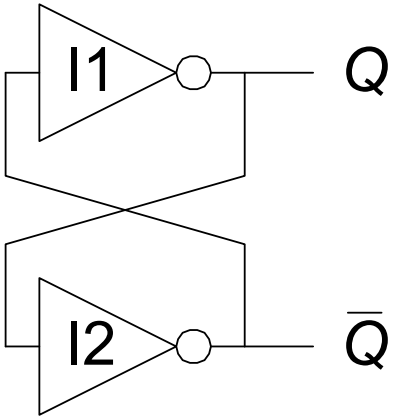
- ✓ Los SS finalmente corresponderán a una combinación de un circuito combinacional que realiza toda la lógica del problema y un conjunto de Latches y/o Flip Flop’s.

Latches & Flip Flop's

- ✓ El bloque fundamental de memoria corresponde al elemento biestable. Es un elemento con dos estados estables.

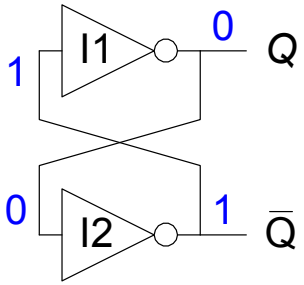
- ✓ No tiene entradas, pero si dos salidas.



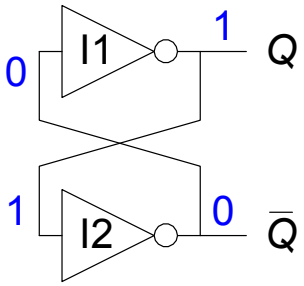


Latches & Flip Flop's

- ✓ El análisis de estos bloque es distinto al de un SC, debido a que son cíclicos. El estado Q depende de Not Q y viceversa.
- ✓ Consideremos dos posible casos:
  - ✓ Q = 0. -> Not Q = 1. -> Q = 0 (es consistente)



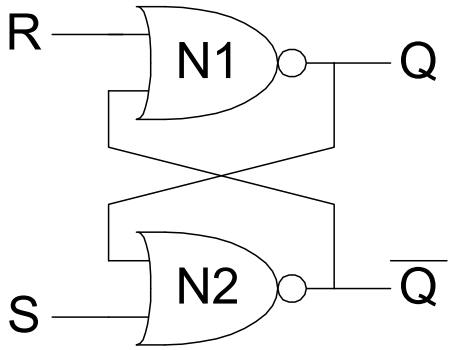
  - ✓ Q = 1. -> Not Q = 0. -> Q = 1 (es consistente)



- ✓ Almacena 1 bit en la variable de estado Q.
- ✓ El problema es que no hay líneas de control del estado.

Latches (S/R)

- ✓ Uno de los CS más simple corresponde al Latch S/R.
- ✓ Está conformado por dos compuertas NOR cruzadas. Posee dos entradas (S y R) y dos salidas (Q y Not Q).
- ✓ S y R permiten controlar los estados.



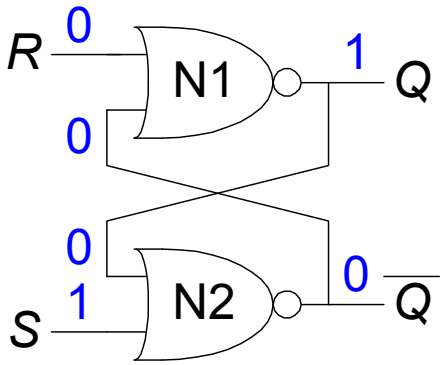
Latches (S/R)

- Revisemos su comportamiento:
  - Si R = 1, S = 0: N1 produce Q = 0. Con Q y S en 0, Not Q = 1.
  - Si R = 0, S = 1. Inicialmente no podemos determinar Q ¿?. Si S = 1, entonces N2 produce Not Q = 0. Con esto determinamos que Q = 1.
  - Si R = 1, S = 1. Q = Not Q = 0.
  - Si R = 0, S = 0. Inicialmente no se puede determinar. Hay que analizar los sub casos (ya que Q puede valer solo 0 o 1)

Latches (S/R)

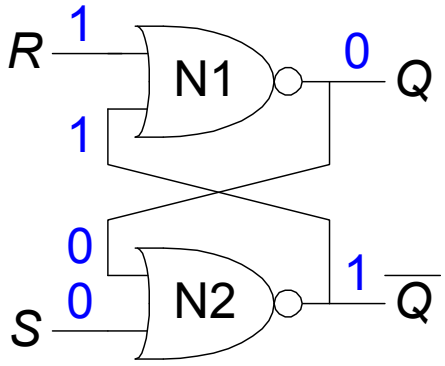
- ✓ Lo mismo pero gráficamente:
  - $S = 1, R = 0$:

    then $Q = 1$ and $\bar{Q} = 0$



  - $S = 0, R = 1$:

    then $\bar{Q} = 1$ and $Q = 0$

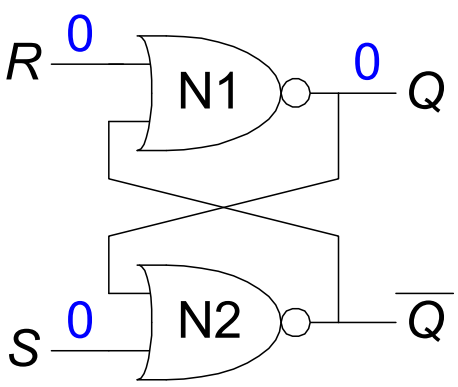


Latches (S/R)

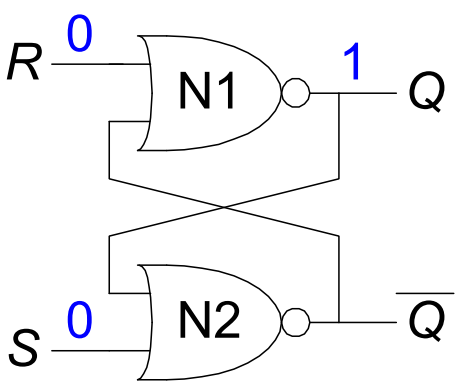
- ✓ Lo mismo pero gráficamente:
  - $S = 0, R = 0$:

    then $Q = \bar{Q}_{prev}$
  - **Memory!**

$Q_{prev} = 0$

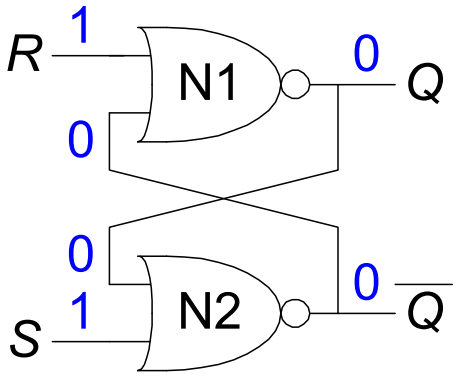


$Q_{prev} = 1$



- $S = 1, R = 1$:

  then $Q = 0, \bar{Q} = 0$

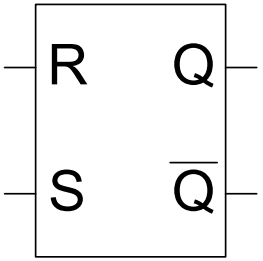


Latches (S/R)

- ✓ La gracia de esto es que podemos controlar el estado utilizando las entradas S y R.

- ✓ Set: S = 1, R = 0, Q = 1.
- ✓ Reset: S = 0, R = 1, Q = 0.

SR Latch Symbol

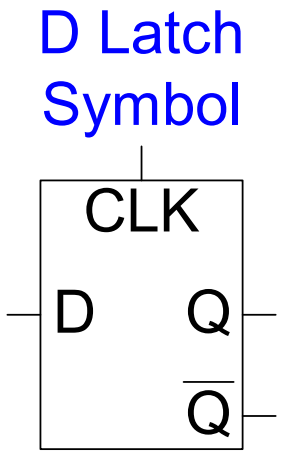


Case	S	R	Q	$\overline{Q}$
IV	0	0	Q_{prev}	$\overline{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

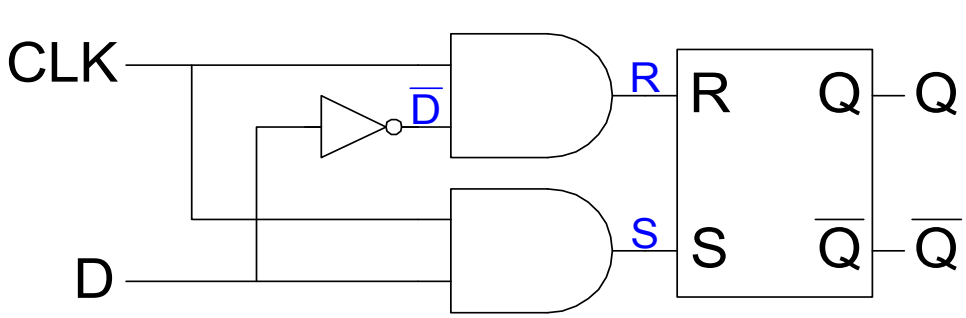
Latches (D)

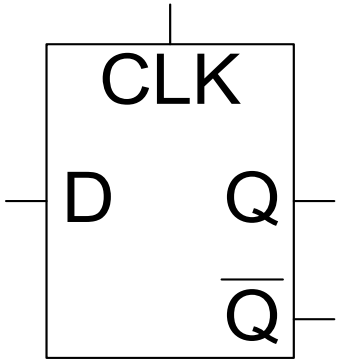
- ✓ Latch tipo D también es un elemento biestable con dos entradas:
  - ✓ CLK: Controla el “cuando” se produce el cambio.
  - ✓ D: Controla que es lo que se debe cambiar (dato).

- ✓ Como funciona:
  - ✓ CLK = 1, se produce el cambio. D pasa a Q.
  - ✓ CLK = 0, Q mantiene su valor anterior (no hay cambio).



Latches (D)

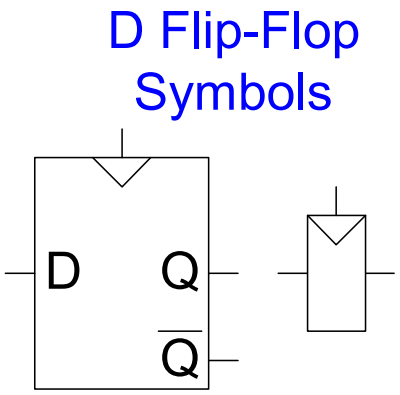




CLK	*D*	$\overline{D}$	*S*	*R*	*Q*	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

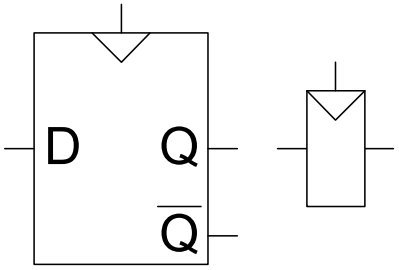
Flip Flop (D)

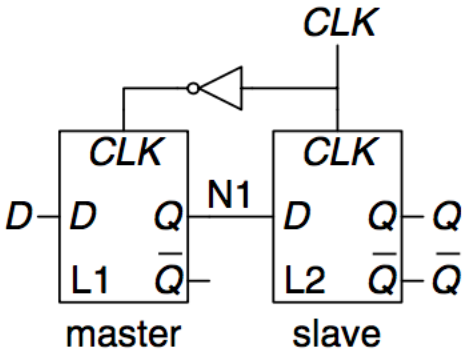
- ✓ Un FF tipo D será construido a partir de dos Latch tipo D controlados por una línea complementaria del reloj.
- ✓ La gracia de esto es intentar solucionar el problema del ancho del pulso de los circuitos a través de la configuración maestro esclavo.
- ✓ El cambio de estado se sincroniza con el canto de bajada de la señal del reloj.
- ✓ Tiene dos entradas: CLK y D. Cuando CLK cambia de 0 a 1. D pasa al valor de Q. Caso contrario se mantiene.



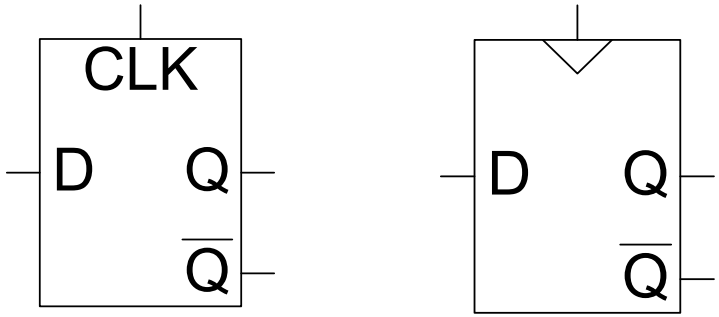
Flip Flop (D)

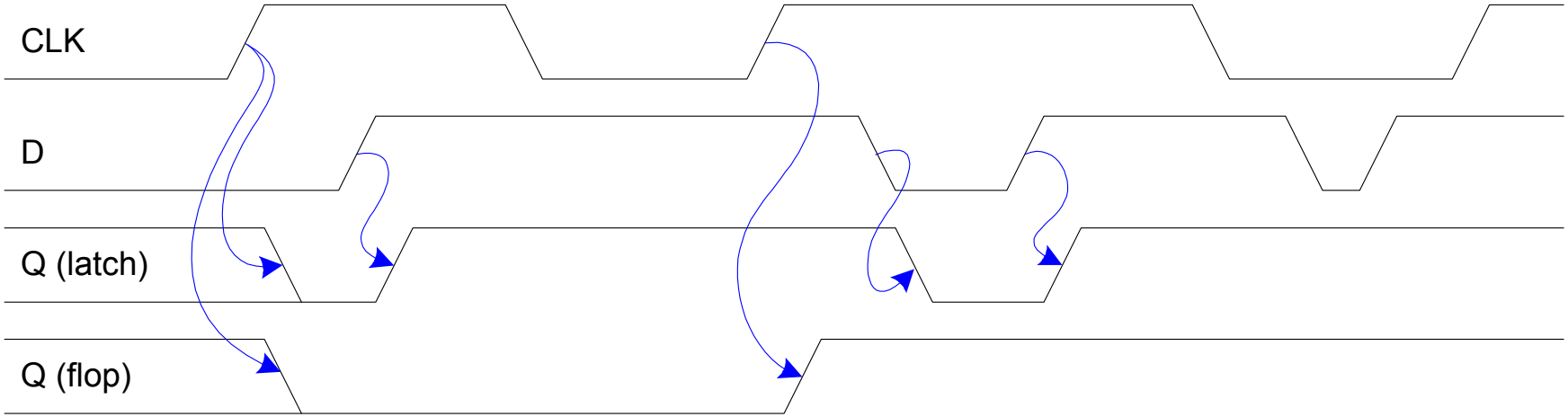
D Flip-Flop
Symbols



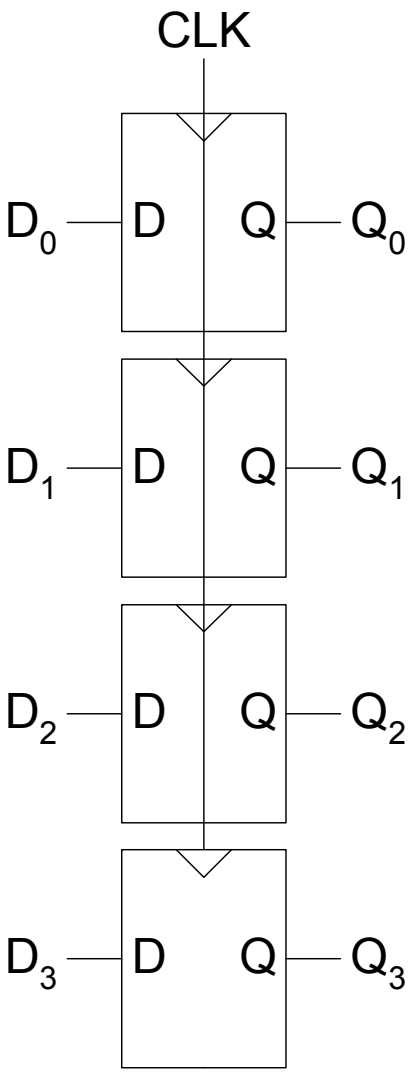


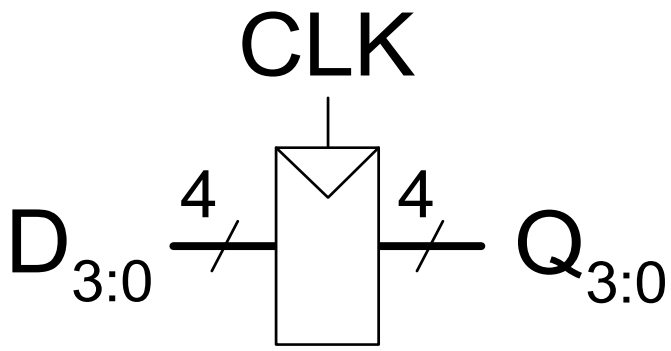
Flip Flop vs Latch (D)



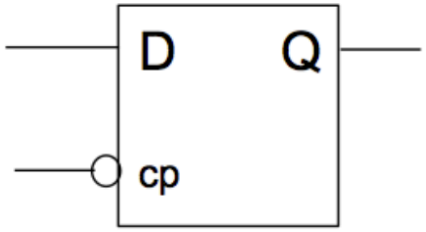


Registros





Flip Flop (D)



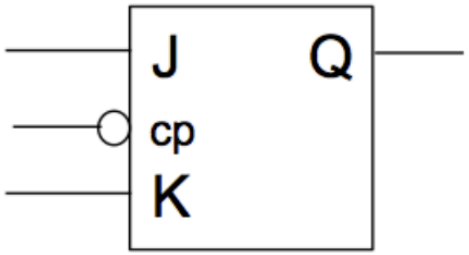
D	Q^{n+1}
0	0
1	1

Tabla característica

Q^n	Q^{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Tabla de excitación

Flip Flop (JK)



J	K	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	$\overline{Q^n}$

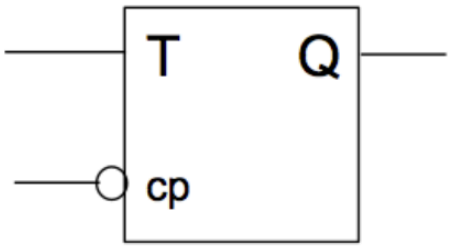
Tabla característica

Q^n	Q^{n+1}	J	K
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

Tabla de excitación

OJO: Estos no son parte del alcance de la versión 2022 del curso

Flip Flop (T)



T	Q^{n+1}
0	Q^n
1	$\overline{Q^n}$

Tabla característica

Q^n	Q^{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Tabla de excitación

OJO: Estos no son parte del alcance de la versión 2022 del curso

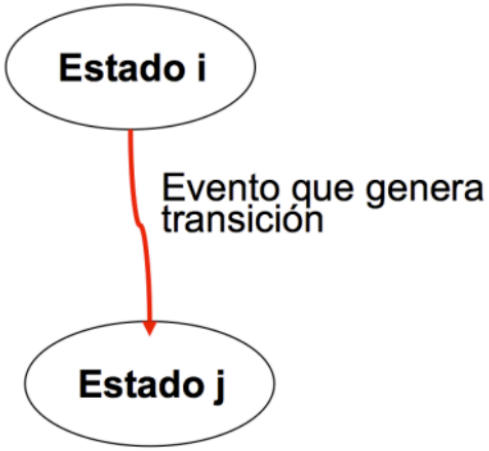
Diagrama de Estados

- ✓ Los sistemas secuenciales se caracterizan por tener estados internos.
- ✓ La salida no depende solo de las entradas (como los SC) sino también del llamado vector de estado.
- ✓ Para cambiar de estado es necesario que ante el evento llamado canto de bajada de CP, cambie por lo menos el estado de un Flip Flop.
- ✓ Los estados internos se describen por un grafo llamado diagrama de estados.

Diagrama de Estados

- ✓ Los nodos representan valores internos de FF's y los arcos representan las transiciones. Los rótulos de los arcos indican los eventos que generan cambio de estado.

- ✓ Estos eventos pueden ser combinaciones de entrada, pero siempre, la transición está sincronizada con el reloj del sistema.



Nuestro Ejemplo

- ✓ Otro Ejemplo: La universidad quiere poner un letrero luminoso en el techo del edificio más alto del campus. El letrero se compone de las letras U, T, F, S y M. Cada letra se ilumina cuando su entrada toma el valor 1 y se apaga cuando su entrada toma el valor 0.
- ✓ El letrero tiene una entrada de control que permite controlar con un número de secuencia si cada una de las letras está iluminada o apagada. El letrero tiene la siguiente secuencia de iluminación:

Etapa 0
Todas las letras están apagadas.

Etapa 1
U y M están prendidas

Etapa 2
T y S están prendidas

Etapa 3
F está prendida

Etapa 4
Todas las letras están prendidas

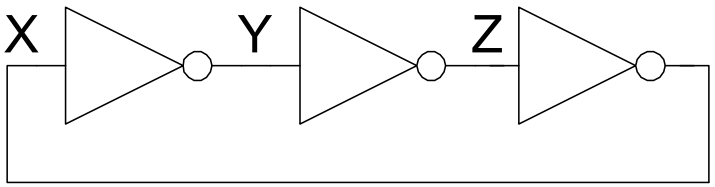
Ejemplo

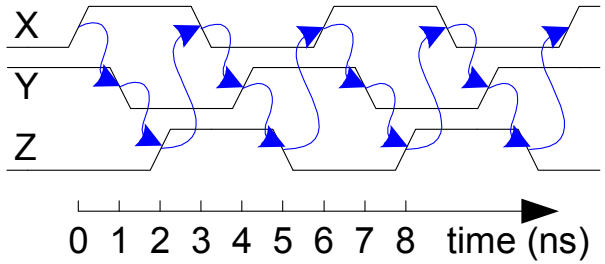
✓ Preguntas:

1. ¿Cuántos bits son necesarios para controlar el número de etapa de la secuencia de iluminación del letrero UTFSM?
2. ¿Cuáles son las combinaciones superfluas?
3. Determinar la tabla de iluminación de cada letrero en función de los bits de control del número de etapa.
4. Deducir de las tablas las ecuaciones booleanas que corresponden a cada letra.
5. Determinar la tabla de estado Qn/Qn+1 correspondiente a la secuencia de iluminación del letrero.
6. Usando FF del tipo J-K construir el circuito digital de control de la secuencia de animación del letrero.

Diseño Secuencial

- ✓ Los circuitos secuenciales corresponden a todos aquellos que no son combinacionales.
- ✓ Un circuito problemático:





- ✓ sin entradas y de 1 a 3 salidas.
- ✓ el tiempo depende de los delay's del inversor.
- ✓ tiene un camino cíclico: La salida alimenta la entrada.

Diseño Secuencial

- ✓ Debemos eliminar los caminos cíclicos insertando registros.
- ✓ Los registros contendrán los estados del sistema.
- ✓ Los estados cambian en en canto de bajada. El sistema está sincronizado con el reloj.
- ✓ Reglas de diseño:
  - ✓ Todo elemento del circuito corresponde a un registro o a un circuito combinacional.
  - ✓ Debe haber al menos 1 registro.
  - ✓ Todos los registros reciben la misma señal de reloj.
  - ✓ Todo camino cíclico contiene un registro.

Máquinas de Estados

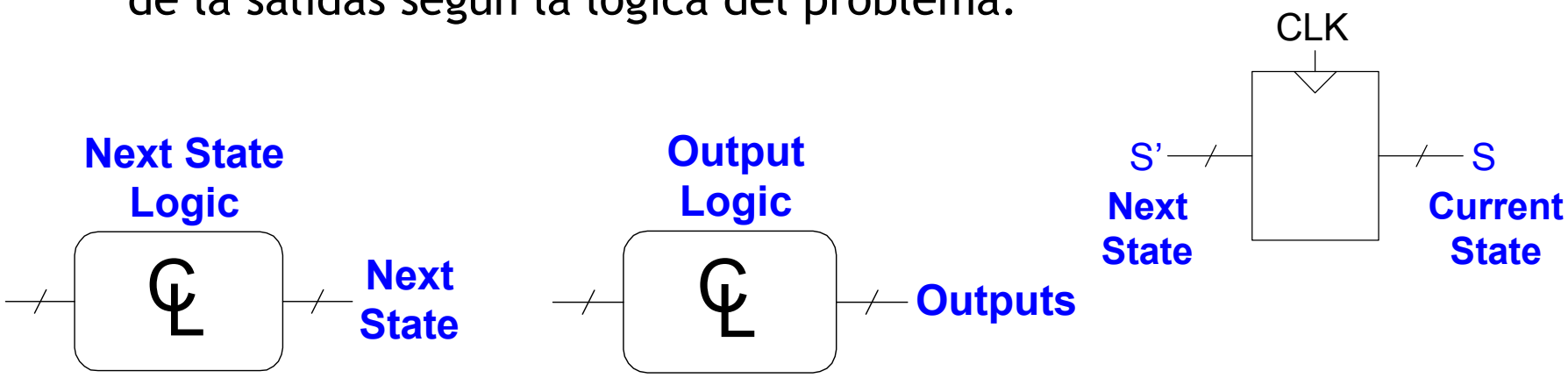
- ✓ Se denomina máquinas de estado a un modelo de comportamiento de un sistema con entradas y salidas.

- ✓ Las máquinas de estados son un conjunto de estados que guardan estrecha relación con las entradas y salidas, haciendo que la historia de señales de entrada determinen, para cada instante, un estado de la máquina, de tal forma que la salida depende únicamente de la entrada y del estado actual de la máquina.

- ✓ Las máquinas de estados finitos (FSM) son aquellas máquinas que tienen una cantidad finita de estados.

Máquinas de Estados

- ✓ Están formadas por:
  - ✓ Registro de Estados: Almacena el estado actual y actualiza al nuevo estado en el canto de bajada.
  - ✓ Circuito combinacional: Computa el siguiente estado y el valor de la salidas según la lógica del problema.



Máquinas de Estados

- ✓ Estas son las únicas máquinas que se pueden modelar en la actualidad con los computadores modernos.
- ✓ En este curso, estudiaremos al menos dos, las máquinas de Moore y Mealy.
- ✓ Las máquinas de Moore y Mealy son circuitos sincrónicos (circuito digital en el cual sus partes están sincronizadas por una señal de reloj).
- ✓ Una máquina de Mealy es una FMS, donde las salidas están determinadas por el estado actual y la entrada.
- ✓ Por otro lado, una máquina de Moore depende sólo del estado actual y no depende de la entrada.

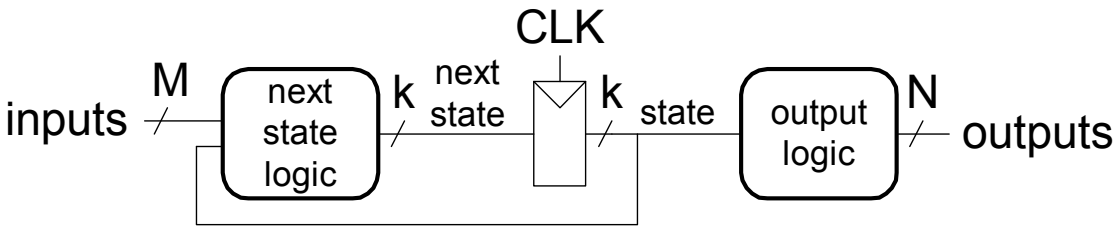
Máquinas de Estados

- ✓ Generalizando, en una máquina de Mealy:
  - ✓ La salida depende del estado actual y de las entradas.
  - ✓ En general, tiene menos cantidad de estados.
  - ✓ Las salidas se encuentran en la transición.

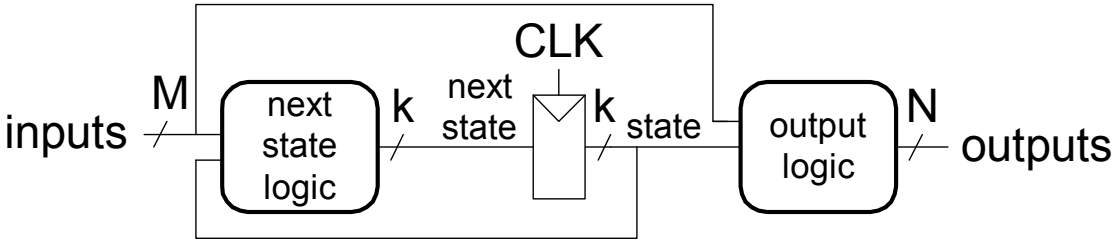
- ✓ Mientras que en una máquina de Moore:
  - ✓ La salida depende del estado actual.
  - ✓ El número de estados es mayor o igual que una de Mealy.
  - ✓ La salidas se encuentran dentro del estado.

Máquinas de Estados

Moore FSM

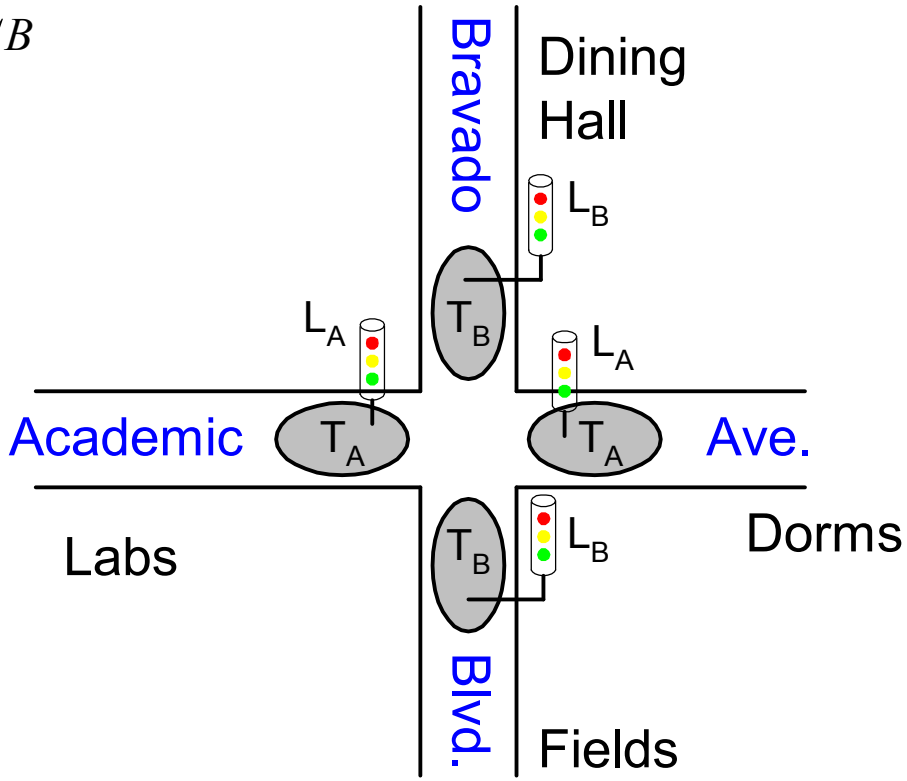


Mealy FSM



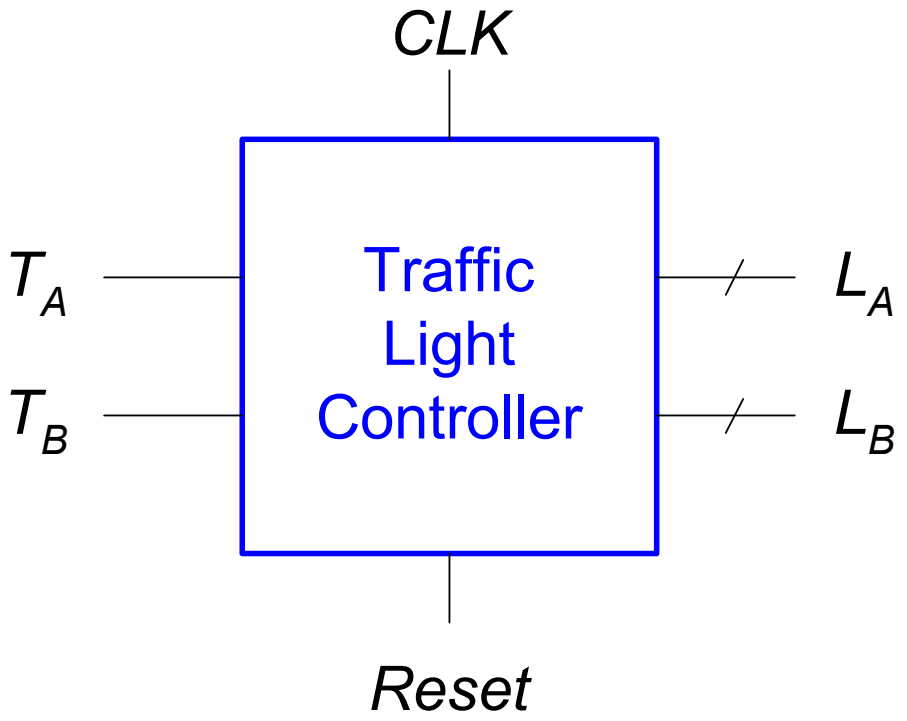
Ejemplo

- Traffic light controller
  - Traffic sensors: T_A, T_B (TRUE when there's traffic)
  - Lights: L_A, L_B



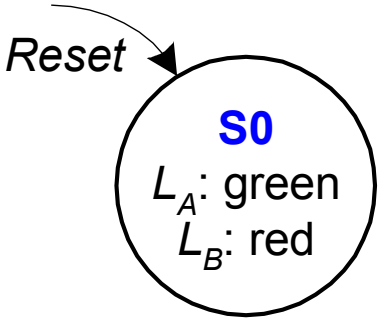
Ejemplo

- Inputs: *CLK*, *Reset*, T_A, T_B
- Outputs: L_A, L_B



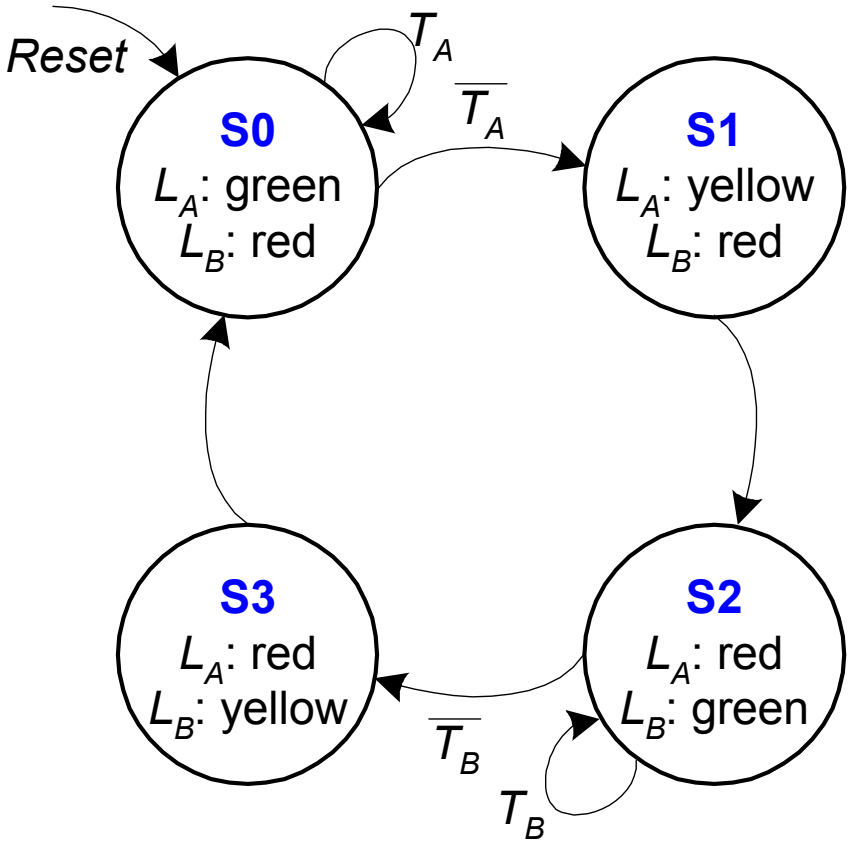
Ejemplo

- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs



Ejemplo

- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs



Ejemplo

Current State S	Inputs T_A	T_B	Next State S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

Ejemplo

Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

State	Encoding
S0	00
S1	01
S2	10
S3	11

$$S'_1 = S_1 \oplus S_0$$

$$S'_0 = S_1 S_0 T_A + S_1 S_0 T_B$$

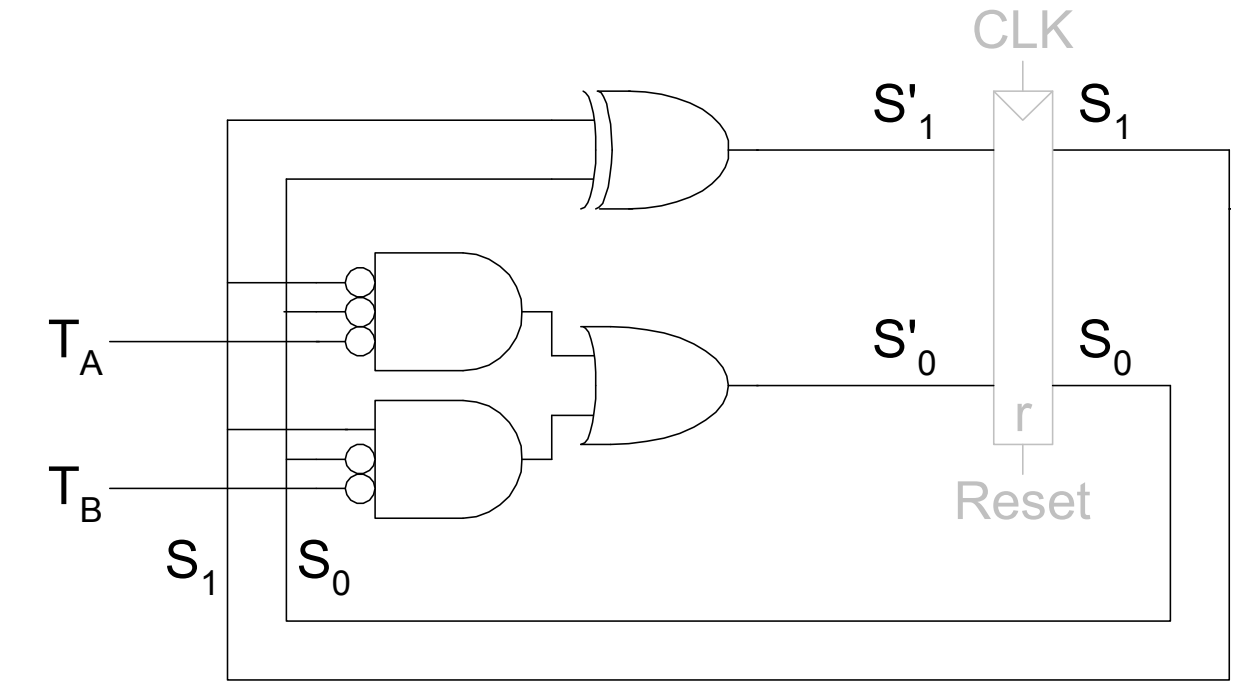
Ejemplo

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

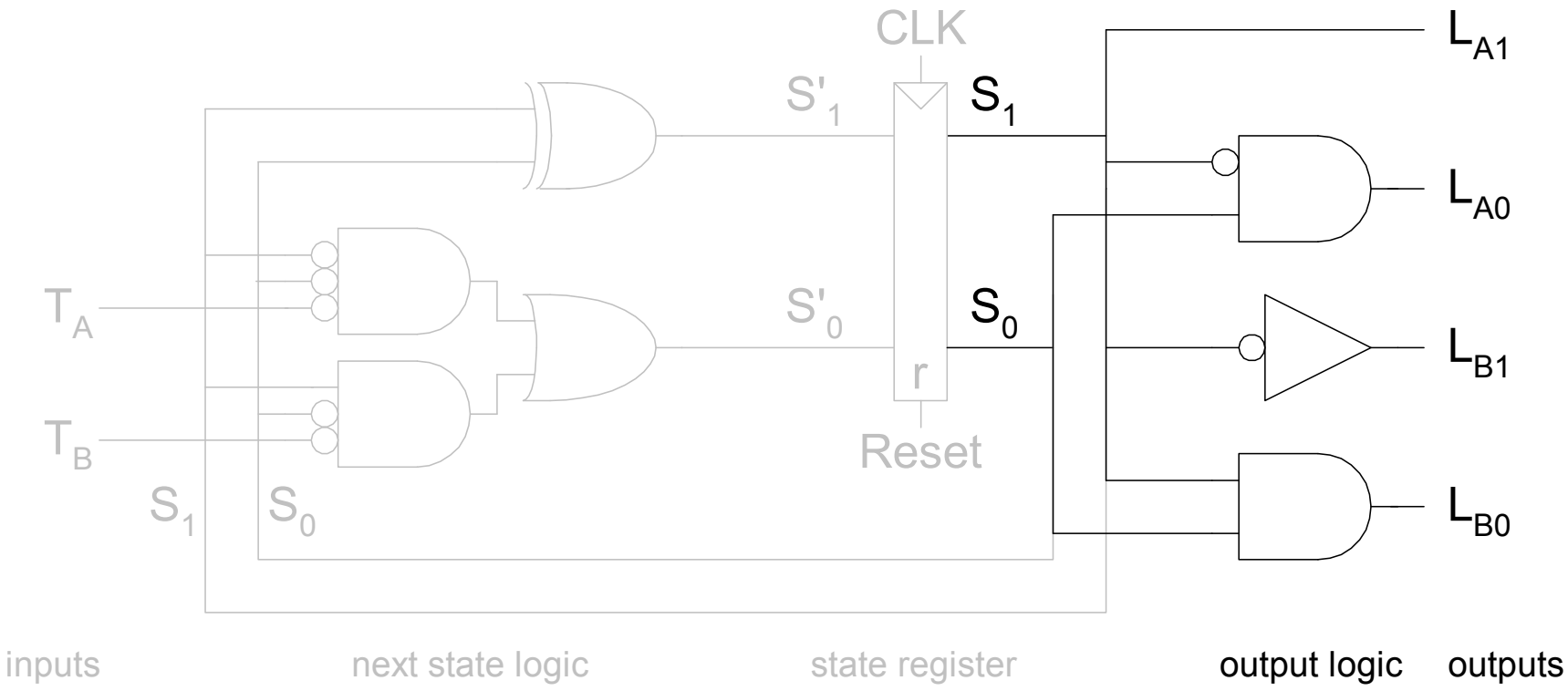
Output	Encoding
green	00
yellow	01
red	10

$$L_{A1} = S_1$$
$$L_{A0} = \overline{S_1} S_0$$
$$L_{B1} = S_1$$
$$L_{B0} = S_1 S_0$$

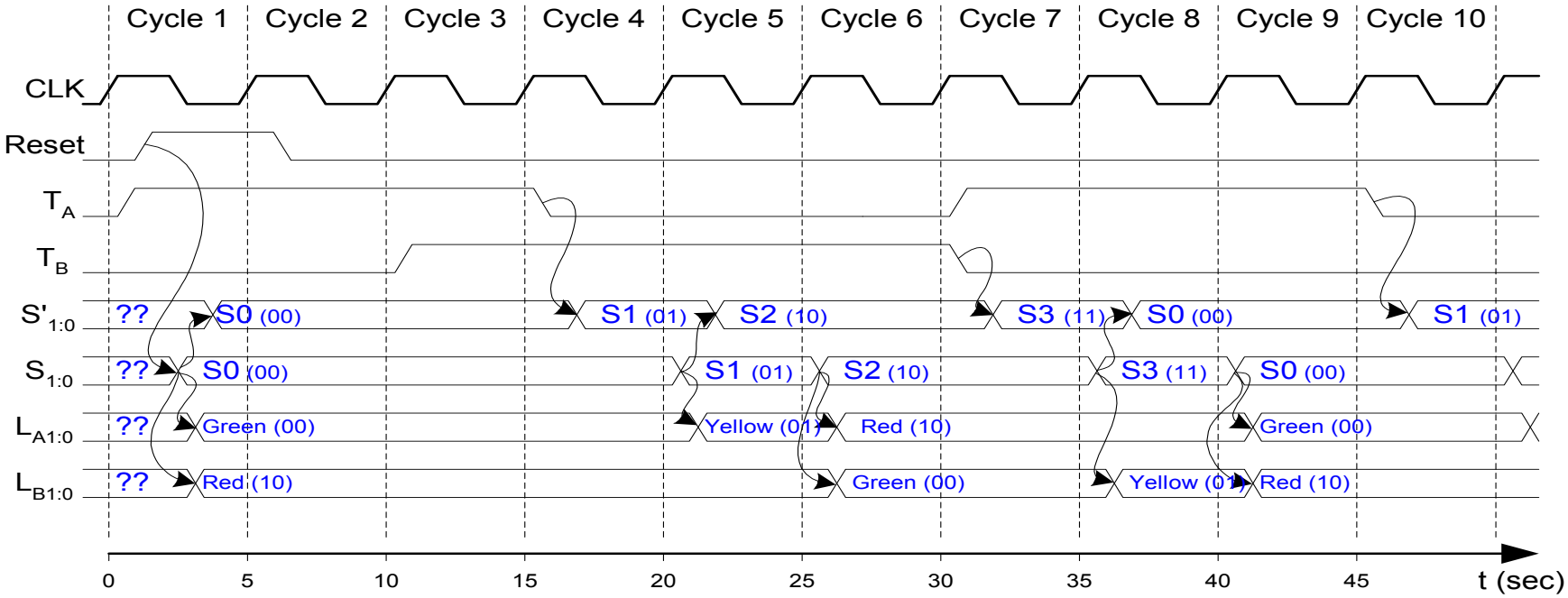
Ejemplo

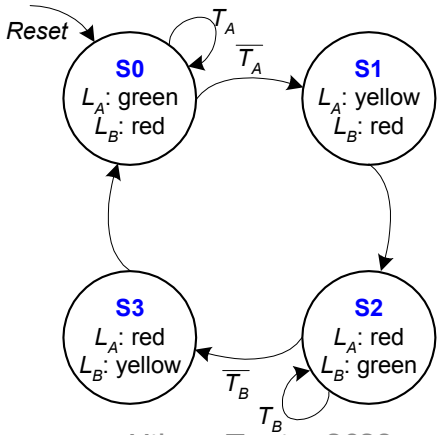

CLK
S'1
S1
TA
S'0
S0
r
TB
Reset
S1
S0
inputs
next state logic
state register

Ejemplo


CLK
S'1
S1
LA1
LA0
TA
S'0
S0
LB1
TB
r
Reset
S1
S0
LB0
inputs
next state logic
state register
output logic
outputs

Ejemplo

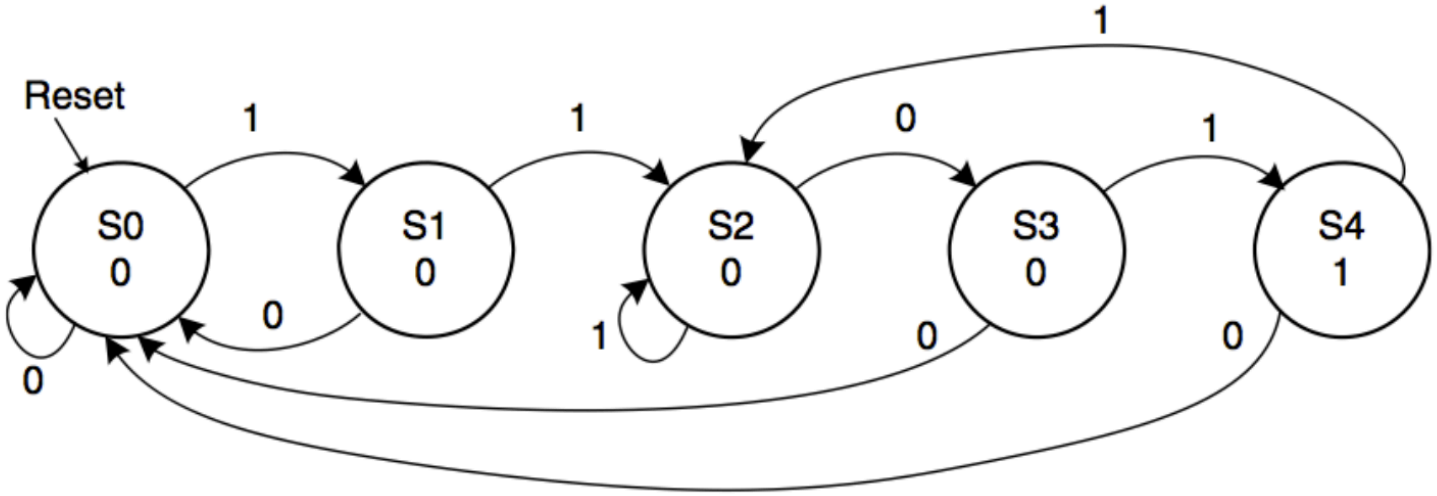

Cycle 1
Cycle 2
Cycle 3
Cycle 4
Cycle 5
Cycle 6
Cycle 7
Cycle 8
Cycle 9
Cycle 10
CLK
Reset
T_A
T_B
$S'_{1:0}$
$S_{1:0}$
$L_{A1:0}$
$L_{B1:0}$
?? S0 (00) S1 (01) S2 (10) S3 (11) S0 (00) S1 (01)
?? S0 (00) S1 (01) S2 (10) S3 (11) S0 (00)
?? Green (00) Yellow (01) Red (10) Green (00)
?? Red (10) Green (00) Yellow (01) Red (10)
0 5 10 15 20 25 30 35 40 45 t (sec)


Reset
T_A
$\overline{T_A}$
S0
L_A: green
L_B: red
S1
L_A: yellow
L_B: red
S3
L_A: red
L_B: yellow
S2
L_A: red
L_B: green
$\overline{T_B}$
T_B

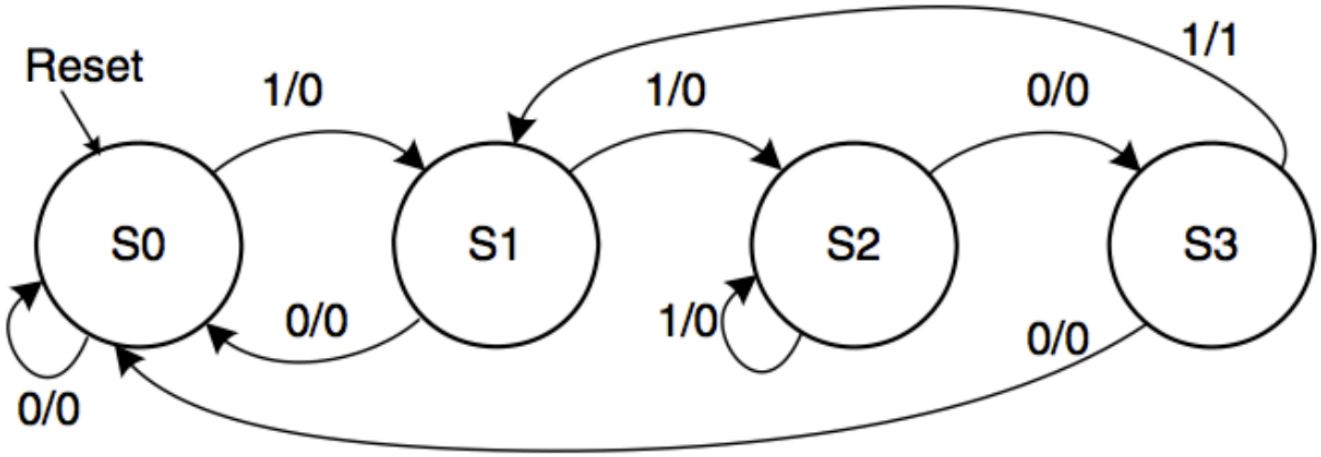
Máquinas de Estados

- ✓ Construir el sistema secuencial que permite detectar en un flujo de bits la secuencia 111. Moore con FF D y Mealy con JK.

Máquinas de Estados



Máquinas de Estados



UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
CAMPUS SAN JOAQUÍN

Arquitectura y Organización de Computadores

Primer Semestre 2023