

8.7.5 – 8.12

Ejemplos

angel@uc.cl

IEE2713 Sistemas Digitales

8.7.5 Ejemplo: un contador diferente

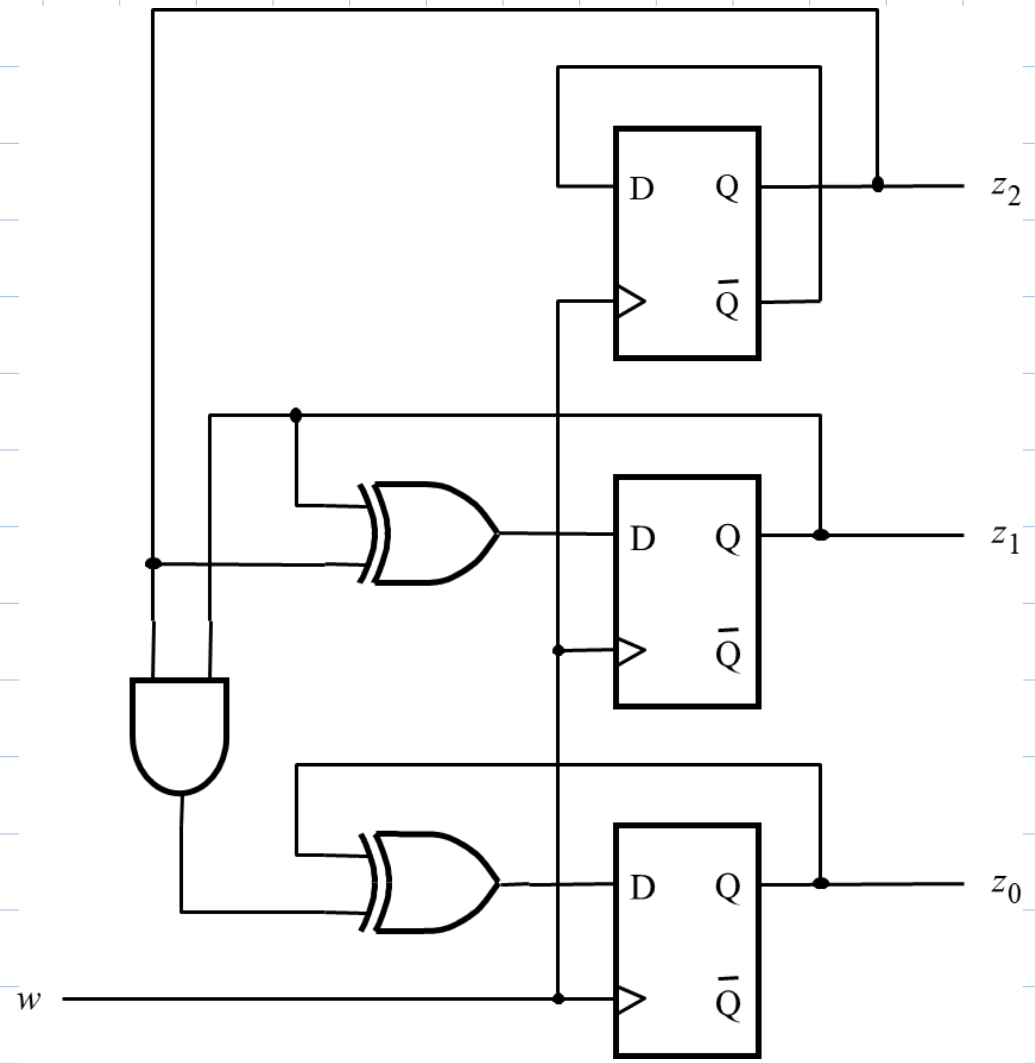
- Consideremos un contador que cuenta pulsos en entrada w, según la secuencia
  - 0, 4, 2, 6, 1, 5, 3, 7, 0, 4, …
  - Como queremos contar la cantidad de pulsos w, tiene sentido usarlo como clock, y el enable siempre en 1
- Usando las tablas, llegamos a las siguientes ecuaciones para el siguiente estado:

$$
\begin{aligned}
D_2 &= Y_2 = \bar{y}_2 \\
D_1 &= Y_1 = y_1 \oplus y_2 \\
D_0 &= Y_0 = y_0\bar{y}_1 + y_0\bar{y}_2 + \bar{y}_0 y_1 y_2 \\
&= y_0(\bar{y}_1 + \bar{y}_2) + \bar{y}_0 y_1 y_2 \\
&= y_0 \oplus y_1 y_2
\end{aligned}
$$

Present state	Next state	Output $z_2z_1z_0$
A	B	000
B	C	100
C	D	010
D	E	110
E	F	001
F	G	101
G	H	011
H	A	111

Present state $y_2y_1y_0$	Next state $Y_2Y_1Y_0$	Output $z_2z_1z_0$
000	100	000
100	010	100
010	110	010
110	001	110
001	101	001
101	011	101
011	111	011
111	000	111

8.7.5 Ejemplo: un contador diferente



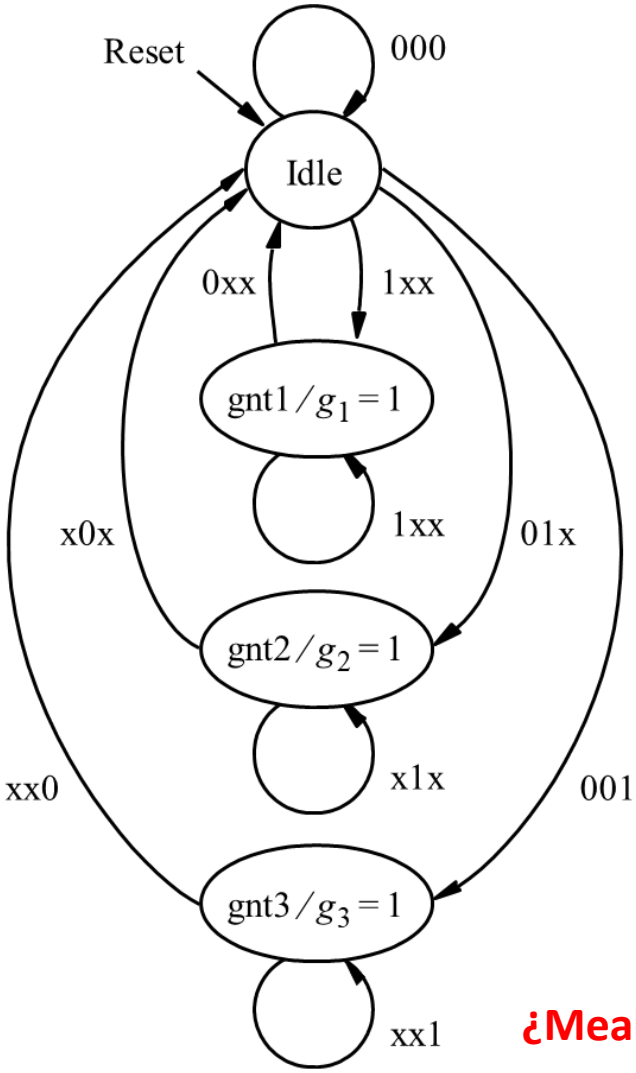
Circuito resultante

- Este circuito cuenta en orden incorrecto porque expresa la cuenta desde LSB hasta MSB (al revés de lo usual)
- Por eso es similar a uno de los contadores vistos en el capítulo anterior, pero con el orden de los bits de salida alterado

8.8 Una FSM como circuito arbitrador

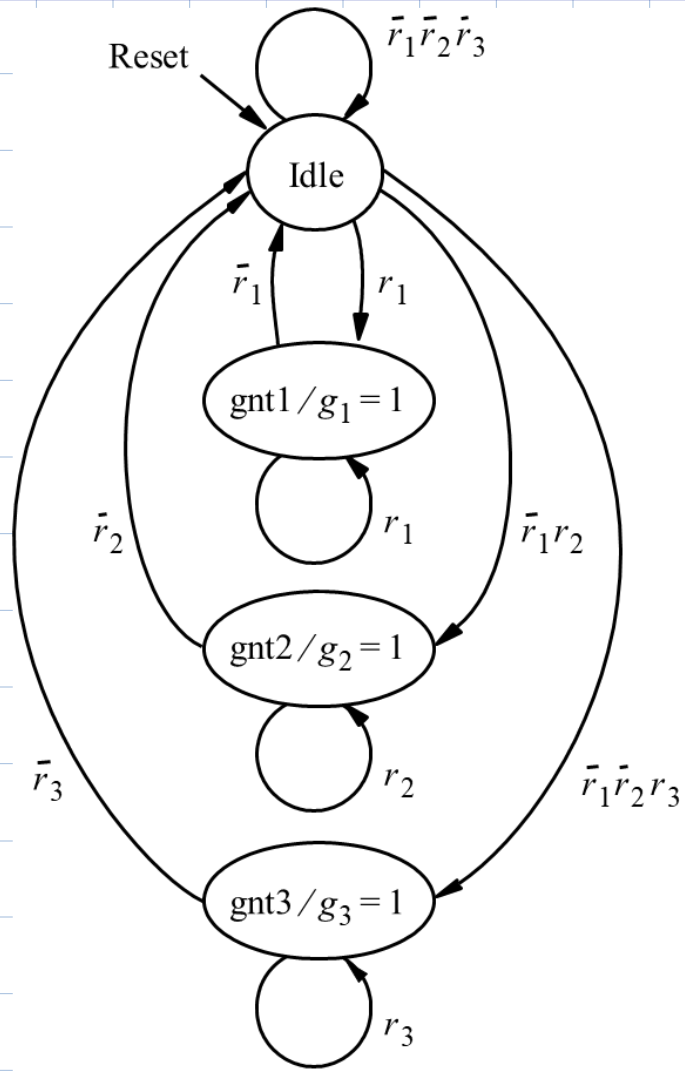
- Vamos a diseñar un circuito que administra el acceso a un recurso que puede ser solicitado por **tres** dispositivos
- Supuestos:
  - Cada dispositivo indica con una señal cuándo quiere utilizar el recurso
  - Se concede la solicitud siempre y cuando el recurso no esté siendo utilizado por otro dispositivo
  - El uso del recurso por un dispositivo se mantiene mientras su señal se mantenga activa
- Vamos a llamar a las entradas **requests** (r1, r2, r3), y a las salidas, **grants** (g1, g2, g3)
- El dispositivo 1 tiene la mayor prioridad en caso de hacer solicitudes al mismo tiempo

8.8 Una FSM como circuito arbitrador



¿Mealy o Moore?

8.8 Una FSM como circuito arbitrador



Este estilo de etiquetado es más intuitivo de entender y nos lleva más rápidamente a expresiones de álgebra booleana

8.8.1 Implementación del circuito arbitrador

```
module arbiter (r, Resetn, Clock, g);
    input [1:3] r;
    input Resetn, Clock;
    output wire [1:3] g;
    reg [2:1] y, Y;
    parameter Idle = 2'b00, gnt1 = 2'b01, gnt2 = 2'b10, gnt3 = 2'b11;

    // Next state combinational circuit
    always @(r, y)
        case (y)
            Idle: casex (r)
                        3'b000: Y = Idle;
                        3'b1xx: Y = gnt1;
                        3'b01x: Y = gnt2;
                        3'b001: Y = gnt3;
                        default: Y = Idle;
                    endcase
            gnt1: if (r[1])  Y = gnt1;
                    else  Y = Idle;
            gnt2: if (r[2])  Y = gnt2;
                    else  Y = Idle;
            gnt3: if (r[3])  Y = gnt3;
                    else  Y = Idle;
            default: Y = Idle;
        endcase

    // Sequential block
    always @(posedge Clock)
        if (Resetn == 0)    y <= Idle;
        else    y <= Y;

    //  Define output
    assign g[1] = (y == gnt1);
    assign g[2] = (y == gnt2);
    assign g[3] = (y == gnt3);

endmodule
```

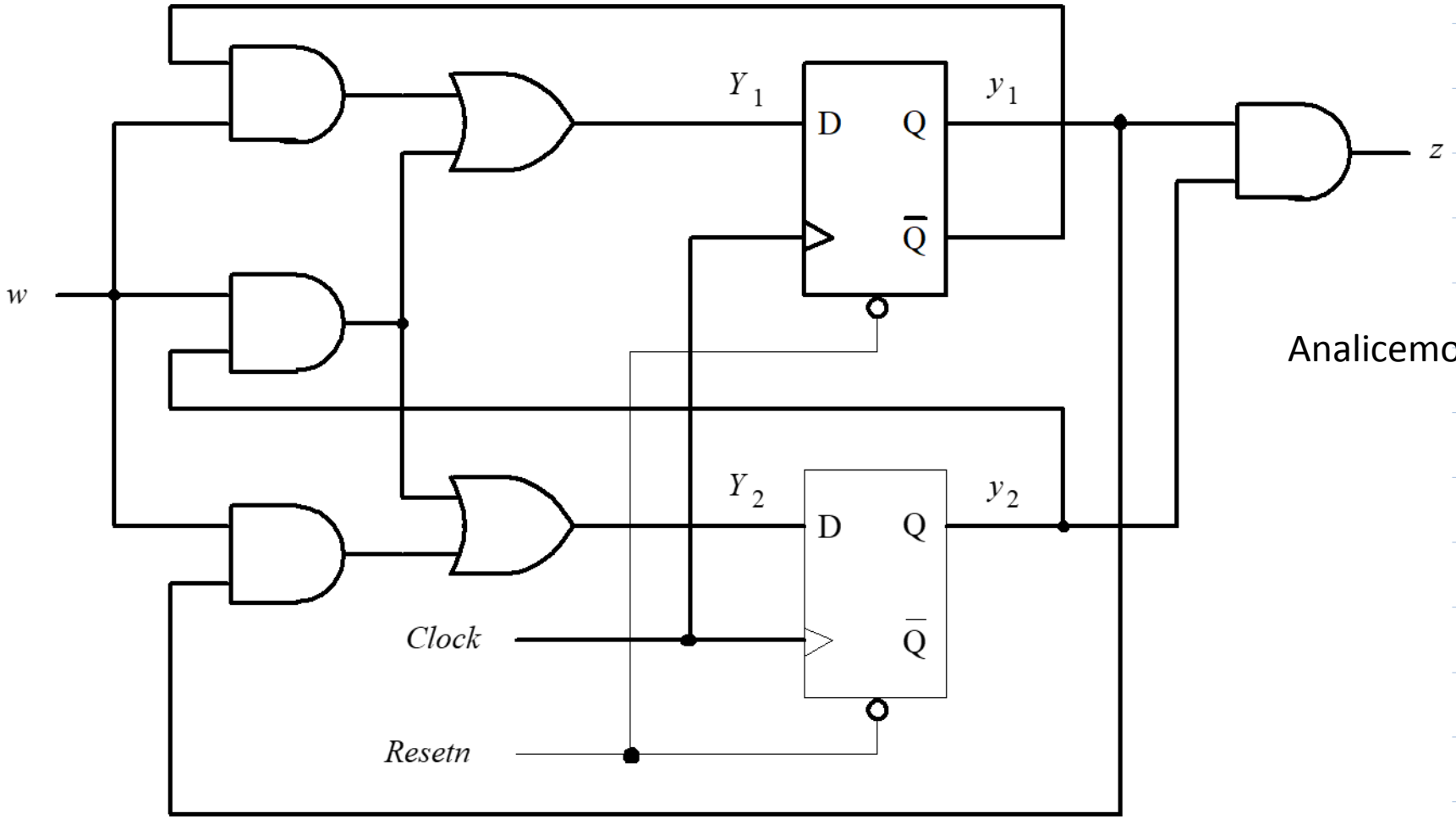
8.8.2 Minimización del retardo de salida de una FSM

- Una FSM puede tener diferentes retardos dependiendo de su implementación
  - Por ejemplo, una implementación one-hot del arbitrador requiere más flip-flops, pero menos lógica combinacional después de los flip-flops
  - Esto hace que una implementación one-hot pueda ser **más rápida** que una implementación codificada en forma binaria
- Los diseñadores de circuitos secuenciales realizan este tipo de optimización para cumplir con las especificaciones de diseño

8.9 Análisis de circuitos secuenciales síncronos

- Además de diseñar circuitos síncronos, como en los ejemplos anteriores, también es importante aprender a analizar circuitos existentes
  - Esto es mucho más sencillo que el proceso de diseño
- La manera más fácil es realizar el proceso inverso al diseño
  - Las salidas de los flip-flops representan las variables del estado **actual**
  - Las entradas a los flip-flops determinan el estado **siguiente**
- Usando esta información podemos construir la tabla de estados
  - Luego, a partir de la tabla llegamos al diagrama de estados, lo que nos permite conocer el comportamiento complete del ciruito

8.9 Análisis de circuitos secuenciales síncronos



Analicemos este ejemplo

8.9 Análisis de circuitos secuenciales síncronos

$$Y_1 = w\bar{y}_1 + wy_2$$
$$Y_2 = wy_1 + wy_2$$
$$z = y_1 y_2$$

- Como hay dos flip-flops, tenemos como máximo 4 estados en la FSM
- Es buena idea es asumir un estado inicial para los flip-flops $y_1 = y_2 = 0$
- De las ecuaciones derivamos las siguientes tablas
  - La interpretación de éstas es el de un circuito que "detecta" un patrón de tres 1s consecutivos, ya que al tercer 1, la salida se vuelve z = 1 y no se reinicia hasta que la entrada se vuelva cero

Present state y_2y_1	Next State Y_2Y_1 w = 0	Next State Y_2Y_1 w = 1	Output z
0 0	0 0	01	0
0 1	0 0	10	0
1 0	0 0	11	0
1 1	0 0	11	1

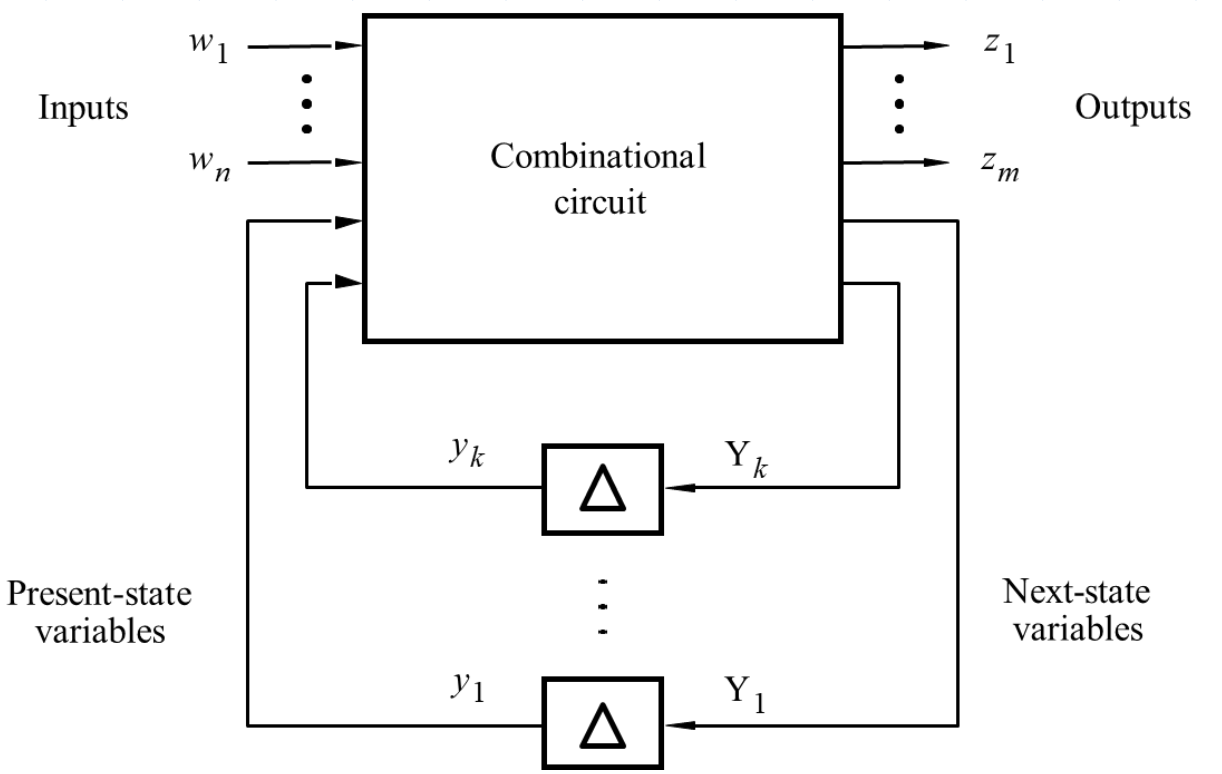
Present state	Next state w = 0	Next state w = 1	Output z
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

8.11 Modelo formal de circuitos secuenciales

- Por el momento hemos hecho un análisis descriptivo y más intuitivo
- Sin embargo, también es posible realizar un análisis formal
- Consideremos:
  - N entradas: $W = \{w_1, w_2, \ldots, w_n\}$
  - M salidas: $Z = \{z_1, z_2, \ldots, z_m\}$
  - K Variables del estado actual: $y = \{y_1, y_2, \ldots, y_k\}$
  - K Variables del estado siguiente: $Y = \{Y_1, Y_2, \ldots, Y_k\}$
  - 2^k Estados: $S = \{S_1, S_2, \ldots, S_{2^k}\}$

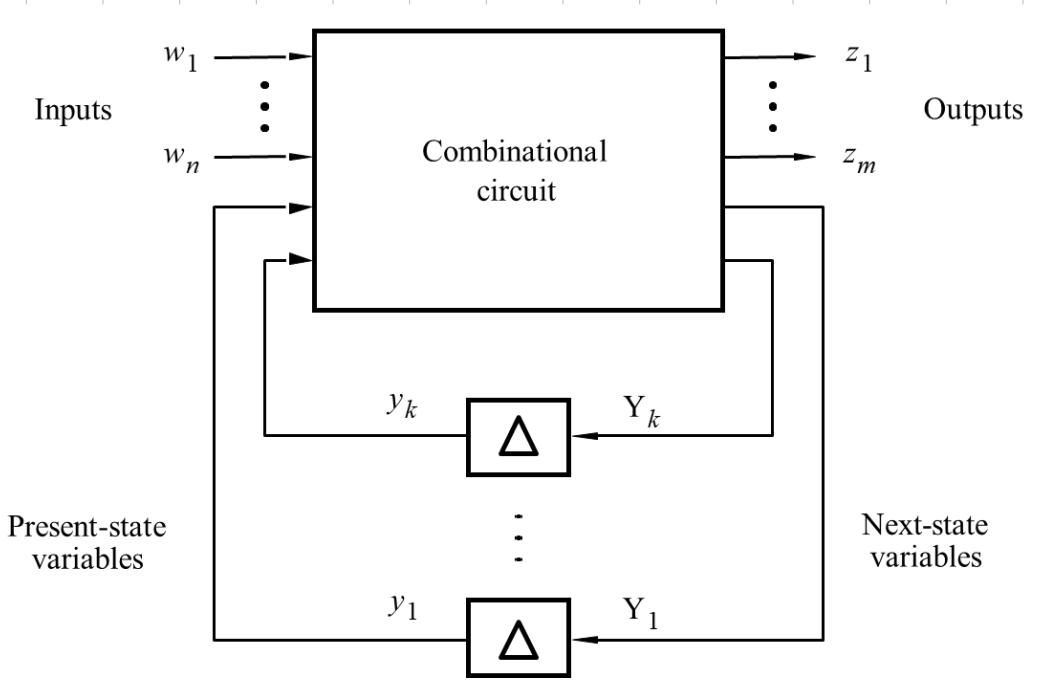
8.11 Modelo formal de circuitos secuenciales

- Un circuito secuencial se puede representar de manera general según la imagen
- Los elementos Δ representan un retardo (delay) que asegura que *y* tomará los valores de *Y* luego de un retardo Δ
- En circuitos secuenciales estos retardos corresponden a los flip-flops



8.11 Modelo formal de circuitos secuenciales

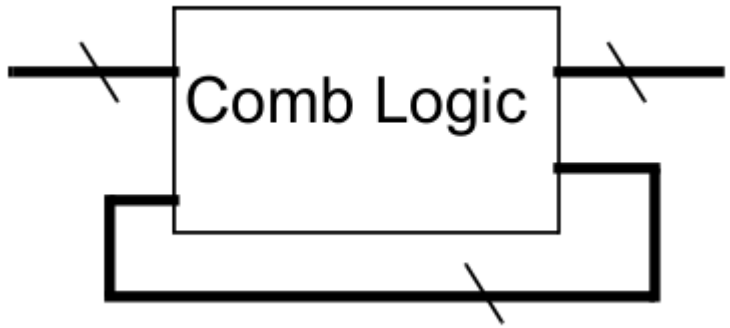
- Un circuito secuencial M se puede definir formalmente como:
  - M = (W, Z, S, φ, λ)
- En donde:
  - W, Z, S son conjuntos finitos, no vacíos de entradas, salidas y estados, respectivamente
  - φ es la función de transición de estados: S(t+1) = φ[W(t), S(t)]
  - λ es la función de los outputs.
  - Para una máquina de Moore: λ(t) = λ[S(t)]
  - Para una máquina de Mealy: λ(t) = λ[W(t), S(t)]
  - Para nuestro caso asumimos que el tiempo entre t y t+1 es un ciclo de clock



¿Con reloj o sin reloj?

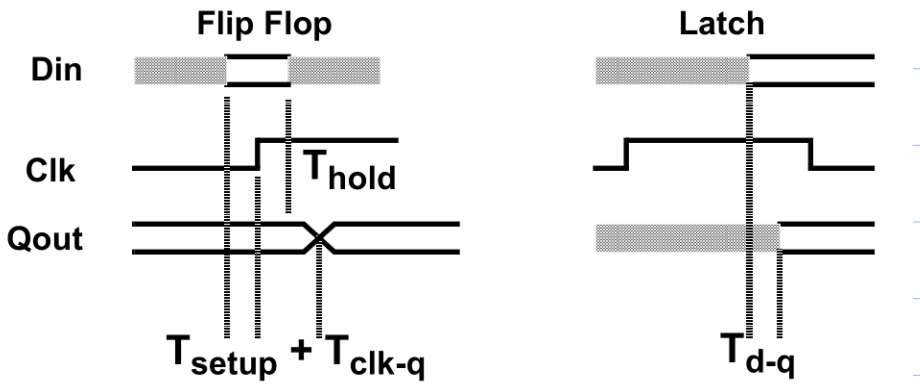
(esto no está en el libro)

- ¿Por qué necesitamos un reloj en los circuitos?
  - Para que las salidas no sólo dependan de las entrdas,sino también de las entradas anteriores
  - En una FSM es importante decir qué señales vienen antes y qué señales vienen después, y esto se logra con un reloj
  - Es importante que señales anteriores y nuevas no se mezclen al mismo tiempo
- Si todos los retardos de todas las señales fueran iguales, no necesitaríamos relojes



Tiempos de setup y hold

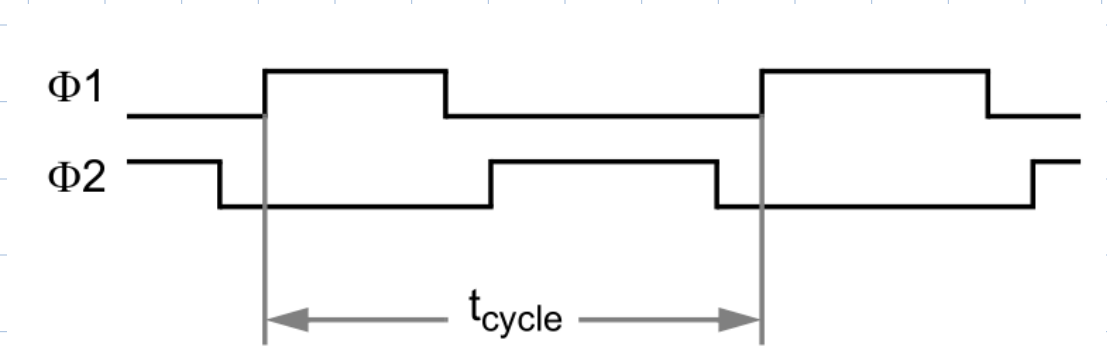
- Recordemos que un flip-flop y latches tienen un tiempos de setup y hold, además del retardo



- Los tiempos de setup y hold deben ser respetados para que las señales no se corrompan
- Los flip-flops retardan las señales en tiempo de setup + retardo entre reloj y Q (T_{clk-q})

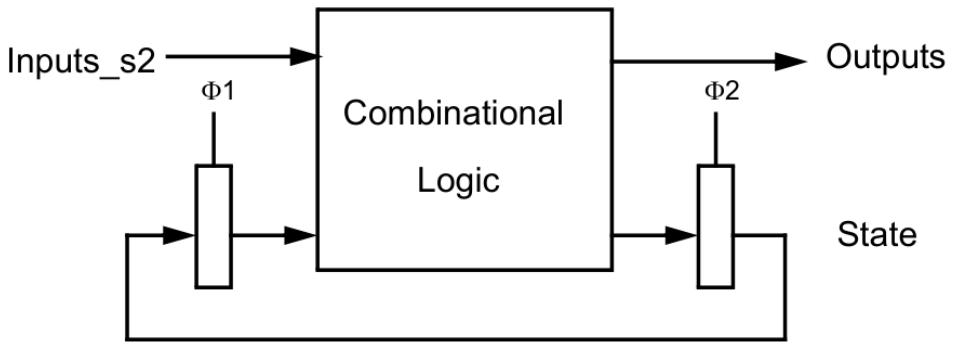
Race condition

- Cuando una señal demora mucho y otra demora muy poco, es posible que no respetemos estos tiempos de setup y hold y llevemos el circuito a un race condition
- Race conditions no se resuelven bajando la velocidad del reloj
- Race conditions se resuelven por diseño: 2-phase clocking

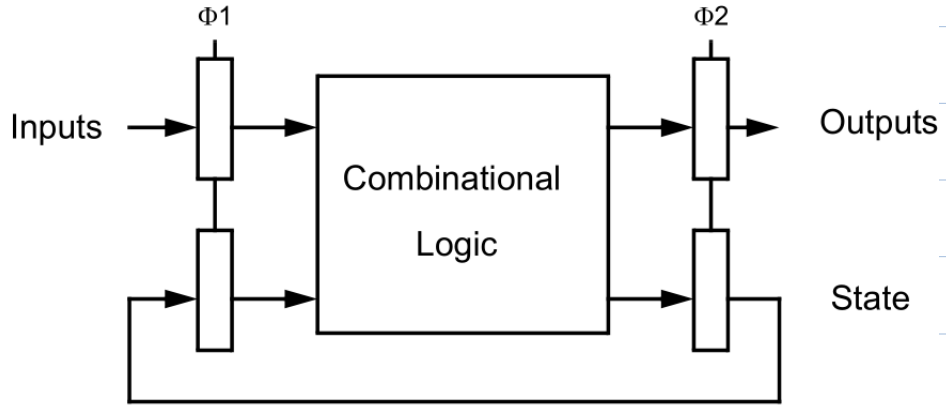


Circuitos con dos fases de reloj

- En circuitos con dos fases de reloj, tenemos
  - Latches comandados por ϕ_1 y latches comandados por ϕ_2
  - Las señales del circuito también se clasifican según su reloj
    - Las salidas de los latches ϕ_1 son buenas entradas para los latches comandados por ϕ_2
    - Las salidas de los latches ϕ_2 son buenas entradas para los latches comandados por ϕ_1
  - También podemos aplicar este esquema de dos fases de reloj a circuitos con flip-flops



Máquina de Mealy con 2 fases de reloj



Máquina de Moore con 2 fases de reloj

Les entrego esta información para que sepan que existe. En ningún caso pretendan que es suficiente para aprender circuitos con relojes de 2 fases

8.12 Resumen

- La existencia de loops cerrados y delays en un circuito secuencial permite un comportamiento caracterizado por un conjunto de estados
- Los valores presentes de las entradas ya no son el único factor determinante de este comportamiento, ya que una secuencia diferente de entradas puede causar un comportamiento diferente al pasar por distintos estados
- Los retardos de propagación son importantes de considerar
- El proceso de diseño asume que el período de reloj es más largo que el tiempo de propagación más largo del circuito para que éste funcione
- Estos circuitos síncronos son **muy** utilizados en diseños prácticos y son diseñados con herramientas CAD