



IC-UNICAMP

# MC 602

## Circuitos Lógicos e Organização de Computadores

IC/Unicamp

Prof Mario Côrtes

### Capítulo 8

### Máquinas de estado

### FSM (*Finite State Machines*)

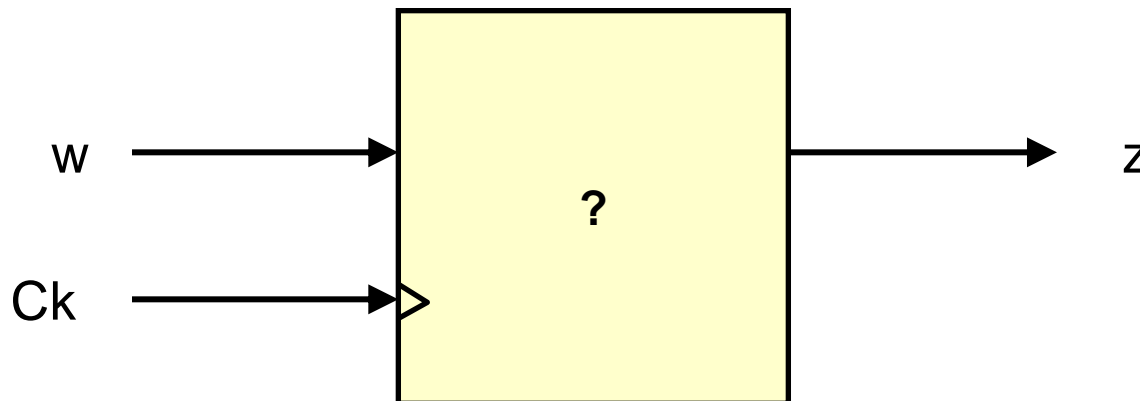
# Tópicos

- Sistematização do projeto de circuitos sequenciais síncronos
- Máquinas de Moore
- Máquinas de Mealy
- Atribuição de estados
- Visão geral de máquinas síncronas

# Problema

- Saída  $z$  igual a
  - “1” se  $w=1$  nos dois últimos ciclos de clock
  - “0” caso contrário
- Todas mudanças sincronizadas com o clock

Ciclo:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
$w$ :	0	1	0	1	1	0	1	1	1	0	1
$z$ :	0	0	0	0	0	1	0	0	1	1	0





## Problema (cont)

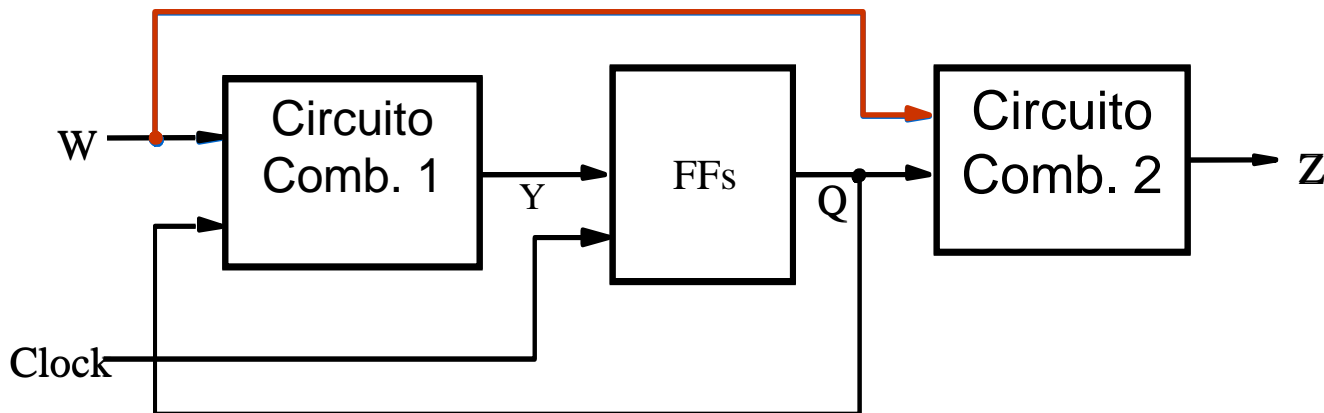
- Possível implementar como circuito combinacional?
- Possível implementar apenas com circuitos sequenciais básicos (FFs, latches, registradores, contadores)?
- Solução estruturada para circuitos sequenciais síncronos:
  - Máquinas de Estados
  - FSM: *Finite State Machines*

# Forma geral de um circuito síncrono

- Circuitos combinacionais: saídas dependem das entradas atuais
- Circuitos sequenciais : saídas dependem do conteúdo atual (estado) dos FFs e das entradas
  - circuitos simples (FF, latch, registrador, contador): há pouca quantidade de portas lógicas além dos elementos de memória
  - Circuito síncrono genérico com grande quantidade de portas lógicas interligando elementos de memória. Estado atual dos FFs e valor das entradas definem:
    - próximo conteúdo (estado) dos FFs
    - saídas atuais

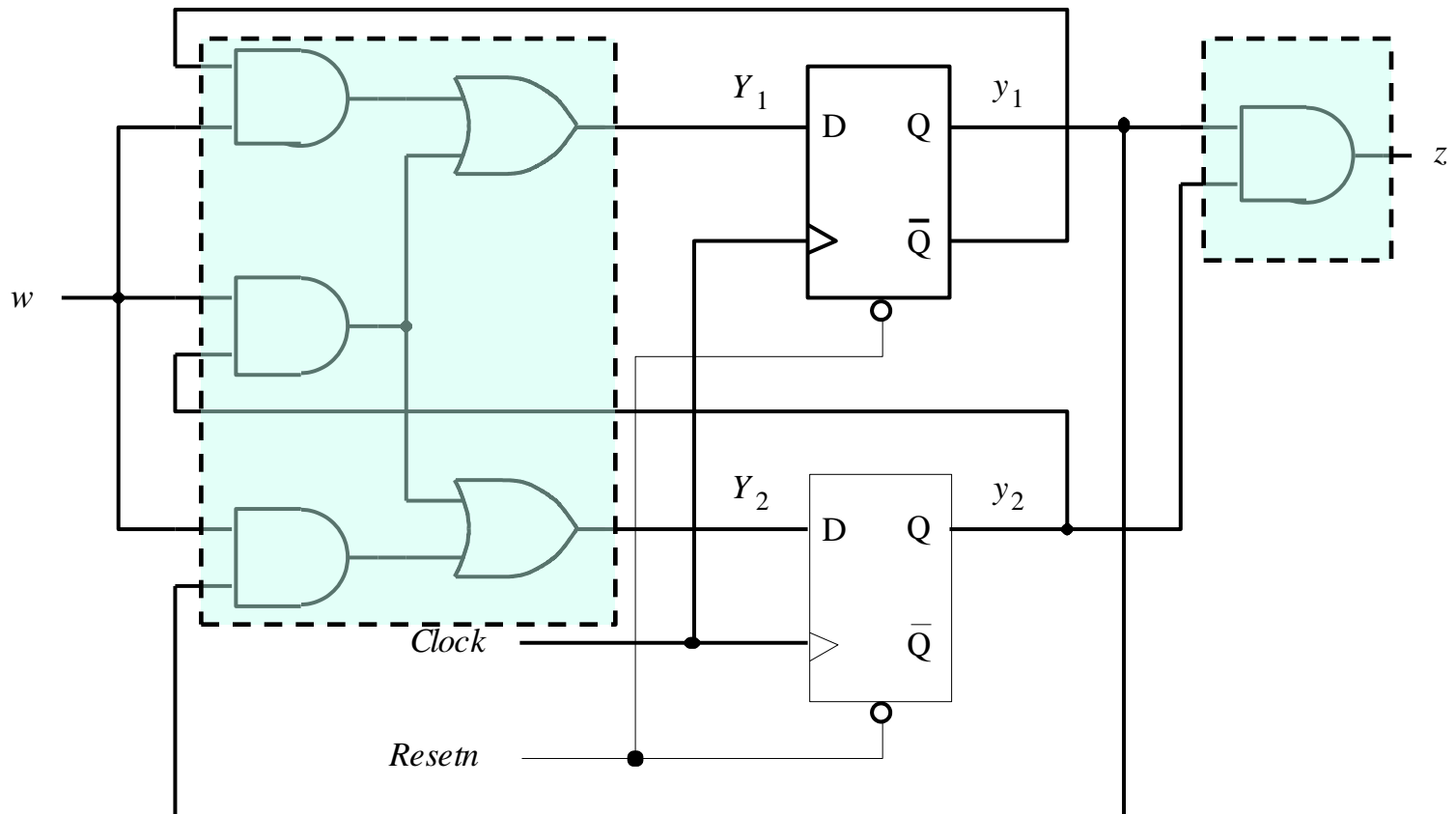
# Máquinas de Estado

- W: Entradas                      Z:Saídas
- Q: Estados internos (saídas de FFs)
- Y: Próximos estados dos FFs
- Circ comb 1:
  - calcula Y (entradas dos FFs) a partir de Q (estado atual = saídas dos FFs) e das entradas W
- Circ comb 2:
  - calcula z a partir de Q e (opcionalmente) W

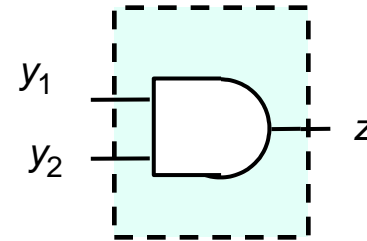
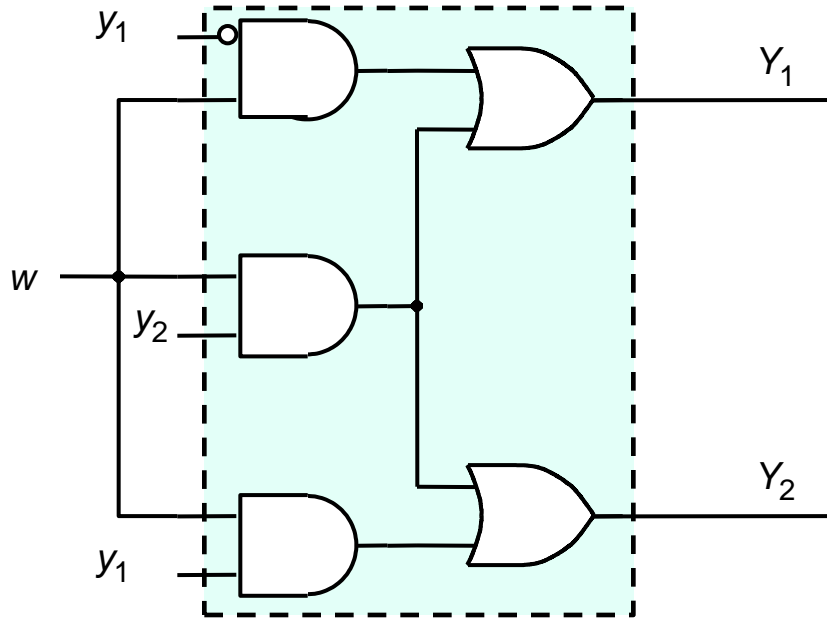


# Análise de uma máquina simples

- Dois FFs  $\rightarrow$  quatro estados possíveis
- Uma entrada  $w$  e uma saída  $z$



# Tabela verdade dos combinacionais



Entradas		Saídas	
w	y2y1	Y2Y1	z
0	00	00	0
0	01	00	0
0	10	00	0
0	11	00	1
1	00	01	0
1	01	10	0
1	10	11	0
1	11	11	1





# Tabela verdade em forma alternativa

Entradas		Saídas	
w	$Y_2Y_1$	$Y_2Y_1$	z
0	00	00	0
0	01	00	0
0	10	00	0
0	11	00	1
1	00	01	0
1	01	10	0
1	10	11	0
1	11	11	1

Present State	Next State		Output
	w=0	w=1	
$Y_2Y_1$	$Y_2Y_1$	$Y_2Y_1$	z
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

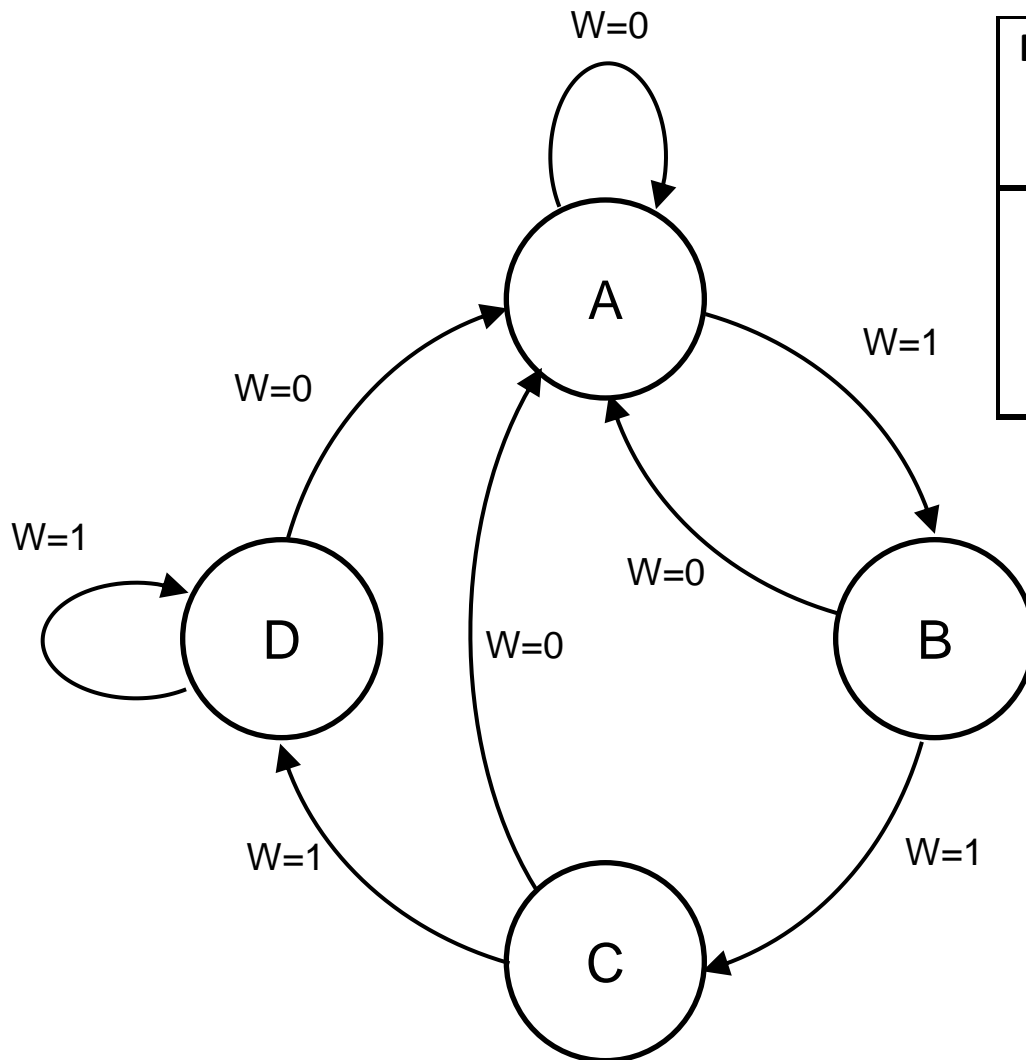
- novo formato:
  - explicita transições de estado

# Atribuição simbólica de estados

Present State	Next State		Output
	w=0	w=1	
$Y_2Y_1$	$Y_2Y_1$	$Y_2Y_1$	$z$
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

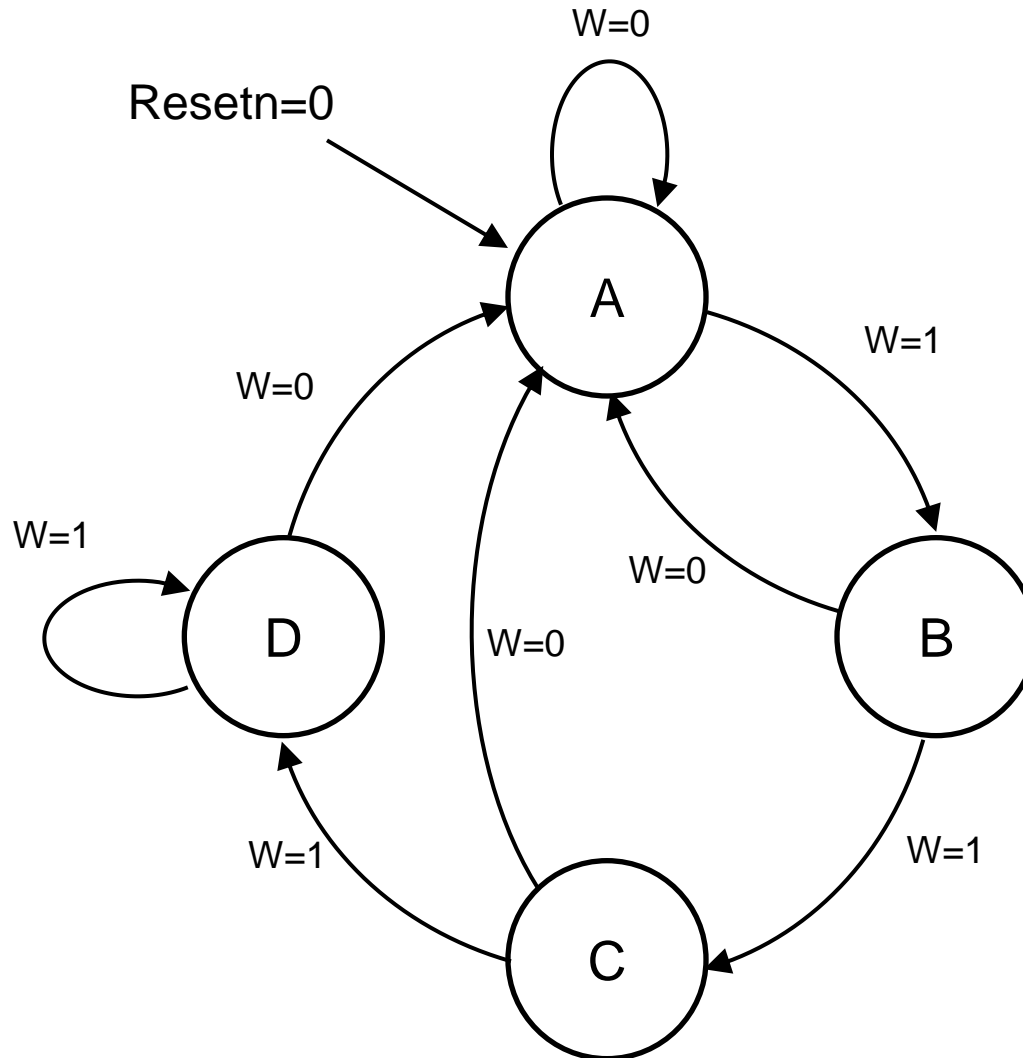
Present State	Next State		Output
	w=0	w=1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

# Diagrama de transição de estados



Present State	Next State		Output
	w=0	w=1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

# Último passo: reset



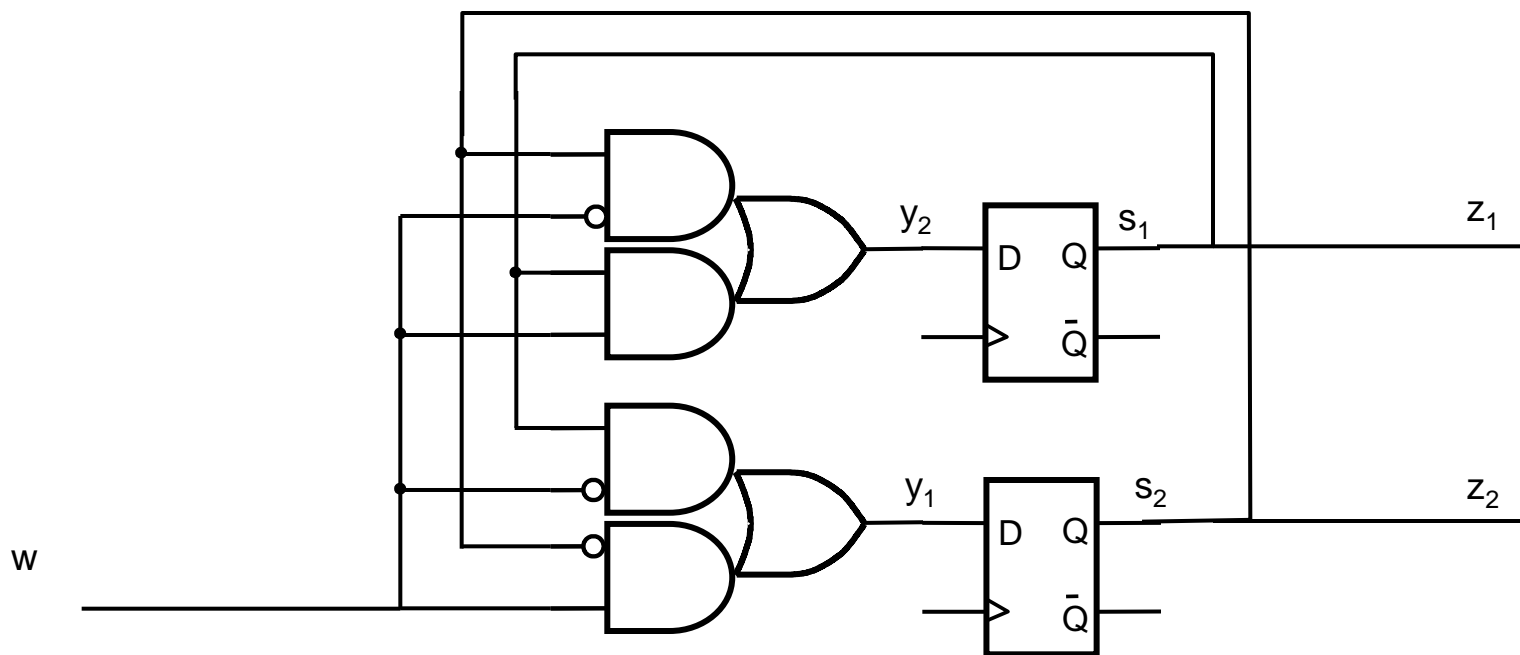
# Análise de uma FSM: resumo

1. Identificar os dois circuitos combinacionais (re-arranjar desenho?)
  - próximo estado
  - saída
2. Tabela verdade convencional
3. Tabela verdade: forma alternativa
4. Atribuição de estados: binário → simbólico
5. Tabela de transição de estados
6. Diagrama de transição de estados
  - Representa de forma compacta a funcionalidade da FSM



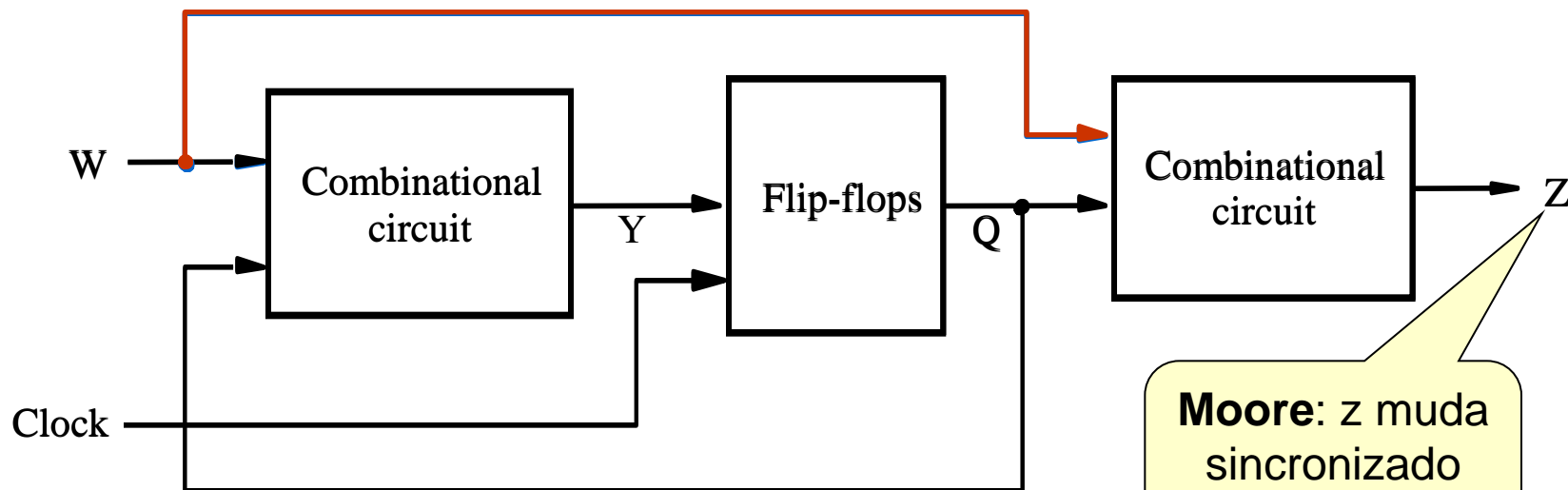
# Exercício de análise de uma FSM simples

- Dois FFs  $\rightarrow$  quatro estados possíveis
- Uma entrada e duas saídas



# Máquinas de Moore e de Mealy

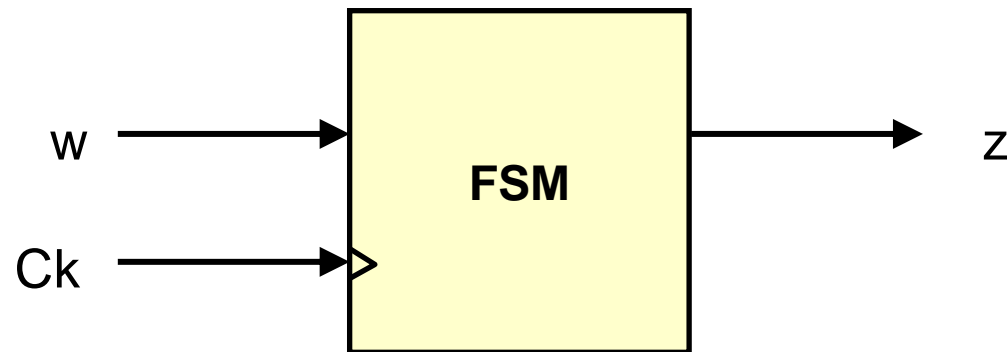
- Máquinas de Moore
  - próximo conteúdo dos flip-flops (Y) depende das entradas (W) e do estado atual (Q)
  - saída depende do estado atual (Q)
- Máquina de Mealy
  - próximo conteúdo dos flip-flops (Y) depende das entradas (W) e do estado atual (Q)
  - saída depende do estado atual (Q) **e das entradas (W)**



**Moore:** z muda sincronizado com o clock

# Síntese de uma FSM simples (Moore)

- O circuito tem uma entrada  $w$  e uma saída  $z$
- Todas as mudanças ocorrem na borda de subida do clock
- $z=1$  se  $w=1$  nos dois últimos ciclos de clock
- $z=0$  caso contrário



Ciclo:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
$w$ :	0	1	0	1	1	0	1	1	1	0	1
$z$ :	0	0	0	0	0	1	0	0	1	1	0

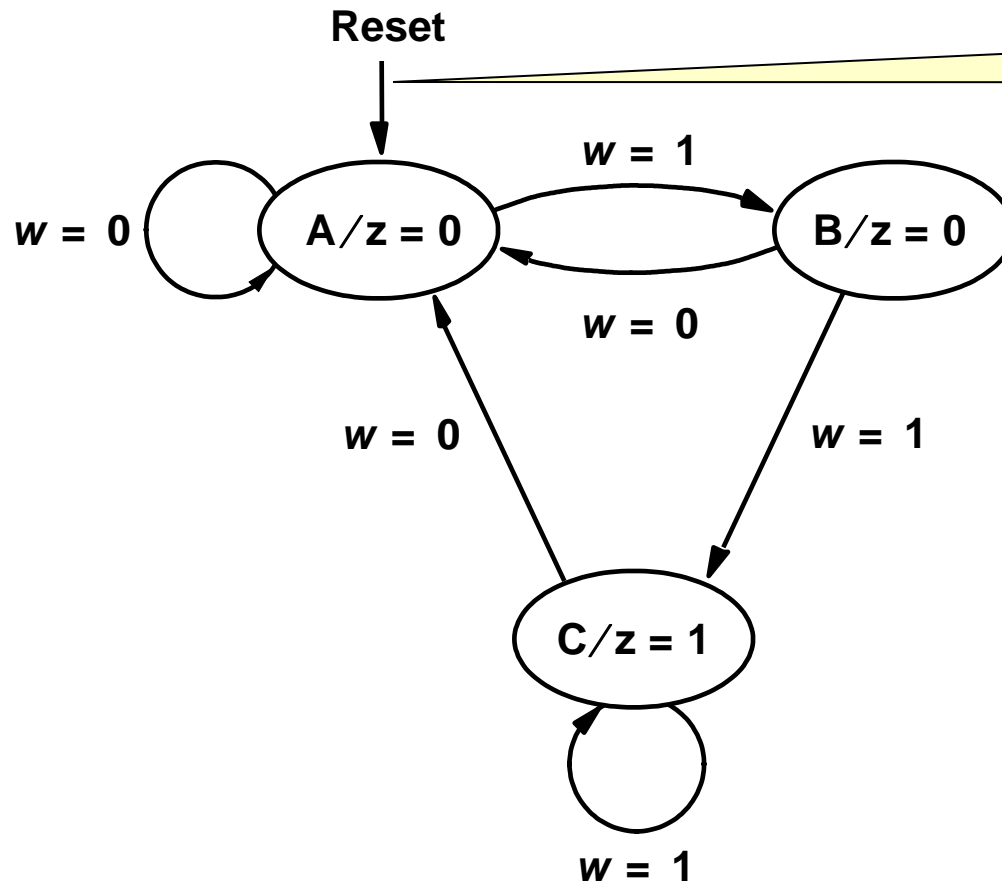




# Definição dos estados

- Primeiro passo: quantos estados são necessários para representar o histórico?
- Não há método sistemático
- Imaginemos estado inicial (power-up ou reset) A com  $z = 0$ 
  - desde que  $w=0$ , FSM mantém-se em A
- Se  $w=1$  por um clock  $\rightarrow$  estado B
  - significa histórico = um clock apenas com  $w=1$
- Em B
  - se  $w=1 \rightarrow$  estado C e  $z = 1$
  - se  $w=0 \rightarrow$  volta para A
- Três estados são suficientes

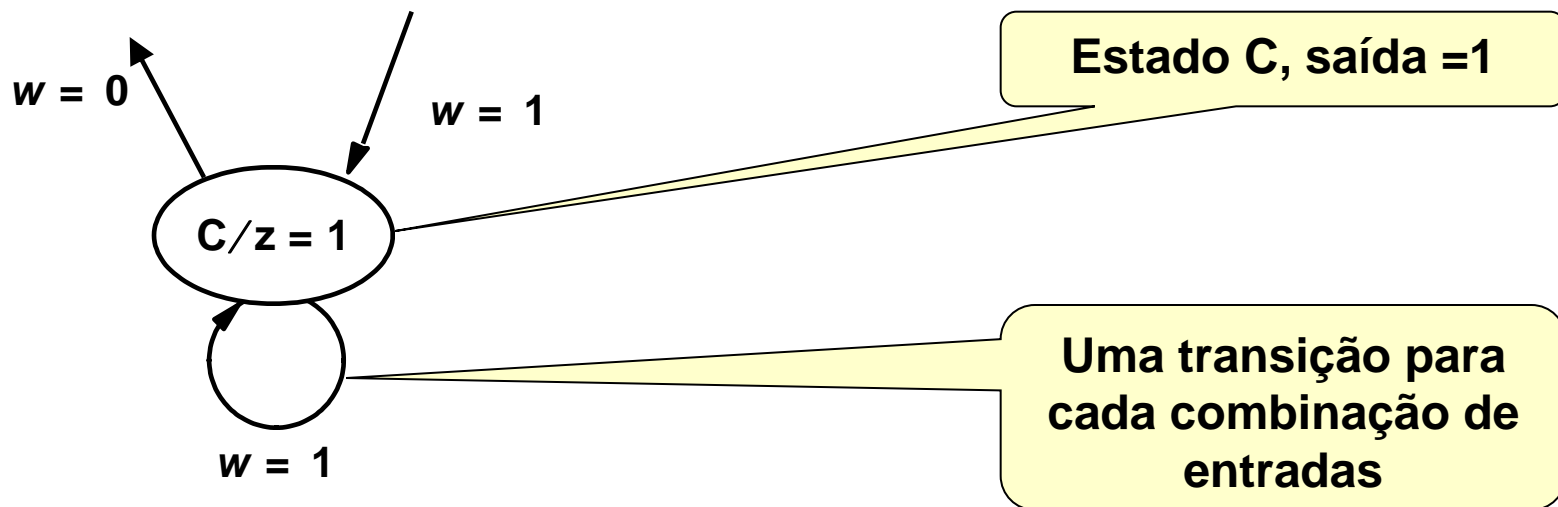
# Diagrama de transição de estados



Reset pode ser clear dos FFs

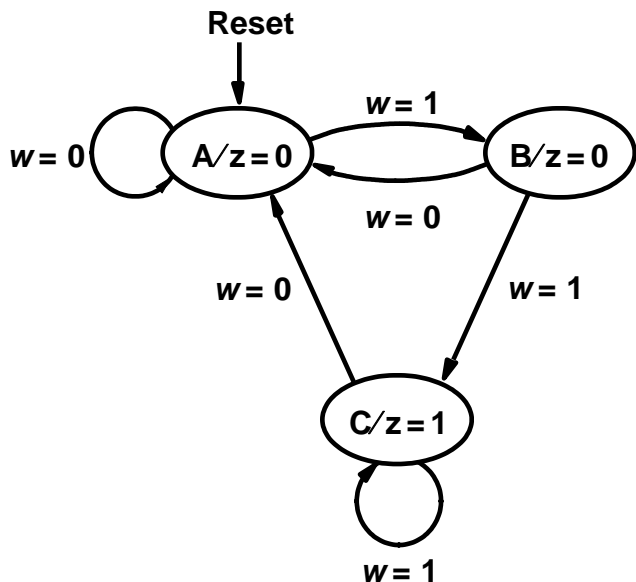
Ciclo:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
$w$ :	0	1	0	1	1	0	1	1	1	0	1
$z$ :	0	0	0	0	0	1	0	0	1	1	0

# Notação do diagrama



- Nesta modalidade: há uma saída determinada para cada estado (C/z=1)
- Transições de estado
  - saindo de C: uma para cada combinação de entradas
  - entrando em C: arbitrário (mas deve haver alguma, senão o estado nunca é alcançado)

# Tabela de transição de estados



Estado atual	Próx estado		saída $z$
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	C	1

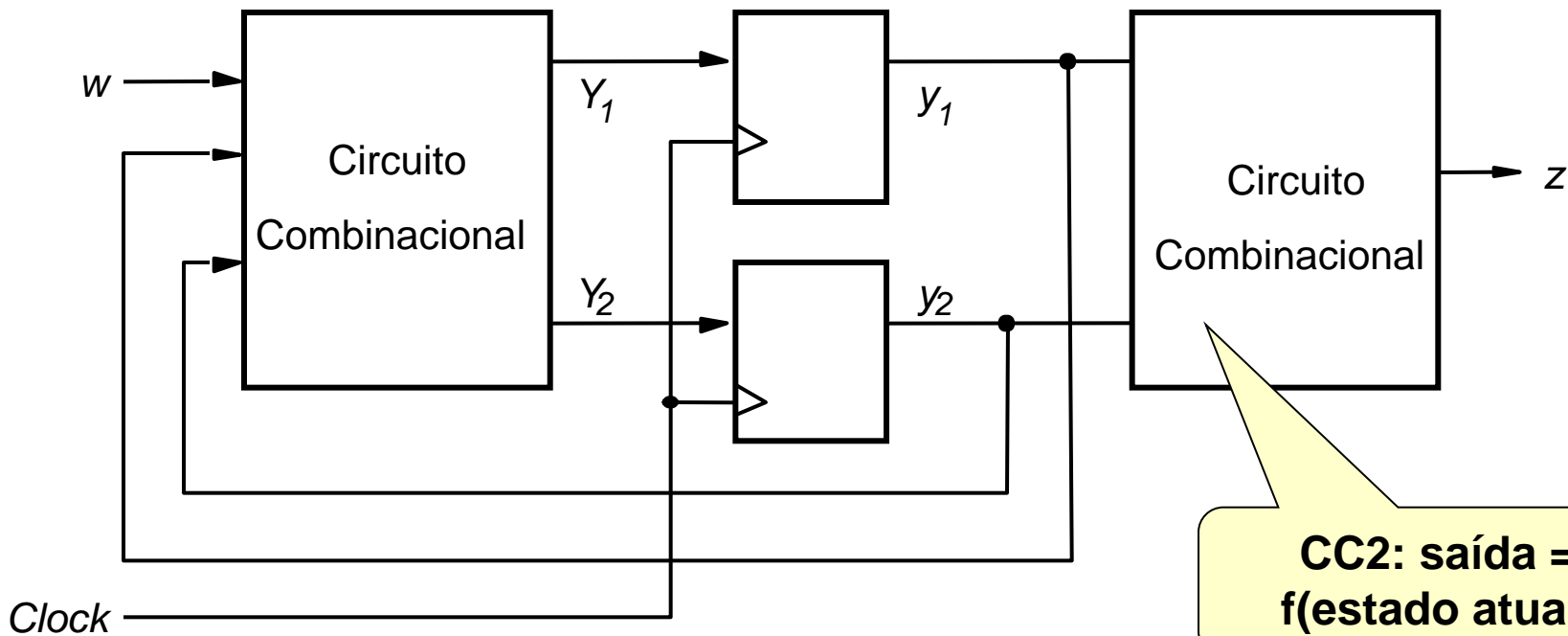
- Notar a ausência de Reset na tabela
  - clear do FF



# Estrutura da FSM

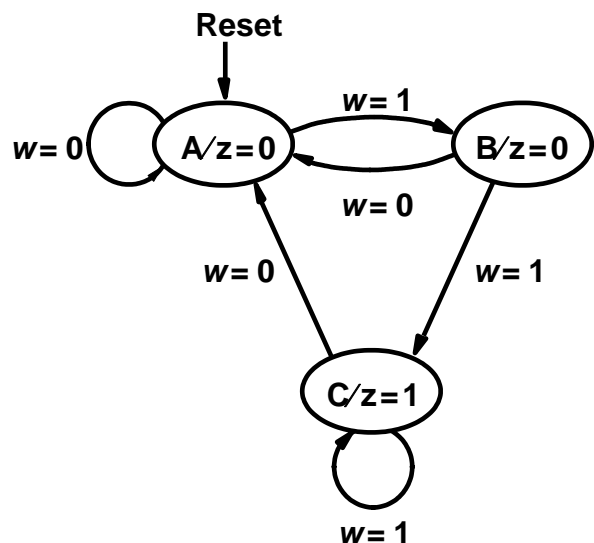
- Há três estados  $\rightarrow$  2 bits são suficientes
- Variáveis de estado:
  - Estado atual  $y_1$  e  $y_2$
  - Próximo estado  $Y_1$  e  $Y_2$

**CC1: prox estado = f(entrada e estado atual)**



**CC2: saída = f(estado atual)**

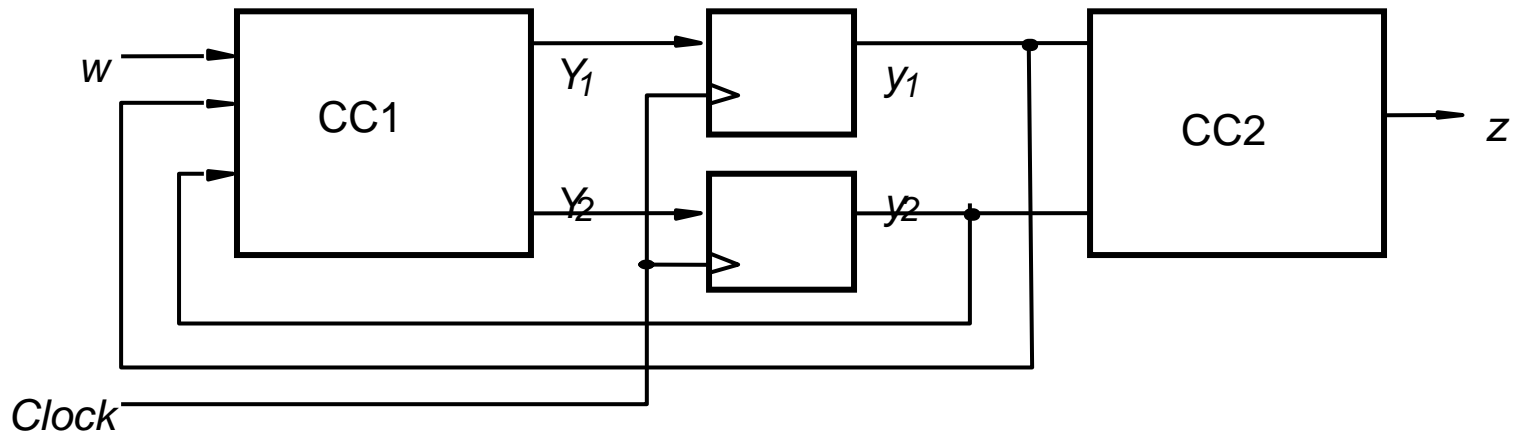
# Atribuição de Estado



Estado atual	Próx estado		saída $z$
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	C	1

Entrada	Estado atual	Próximo estado	Saída
$w$	$y_2y_1$	$Y_2Y_1$	$z$
0	A=00	A=00	0
0	B=01	A=00	0
0	C=10	A=00	1
0	11	dd	d
1	A=00	B=01	0
1	B=01	C=10	0
1	C=10	C=10	1
1	11	dd	d

# Tabelas verdade de CC1 e CC2 (assumindo FF-D)



w	y2	y1	Y2	Y1
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	d	d
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	d	d

w	y2	y1	z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	d
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	d

# Síntese de CC1 e CC2

		$y_2y_1$			
		00	01	11	10
$w$	0	0	0	d	0
	1	1	0	d	0

Sem don't cares

$$Y_1 = w\bar{y}_1\bar{y}_2$$

Com don't cares

$$Y_1 = w\bar{y}_1\bar{y}_2$$

		$y_2y_1$			
		00	01	11	10
$w$	0	0	0	d	0
	1	0	1	d	1

$$Y_2 = wy_1\bar{y}_2 + w\bar{y}_1y_2$$

$$Y_2 = wy_1 + wy_2$$

$$= w(y_1 + y_2)$$

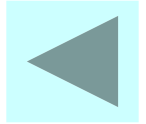
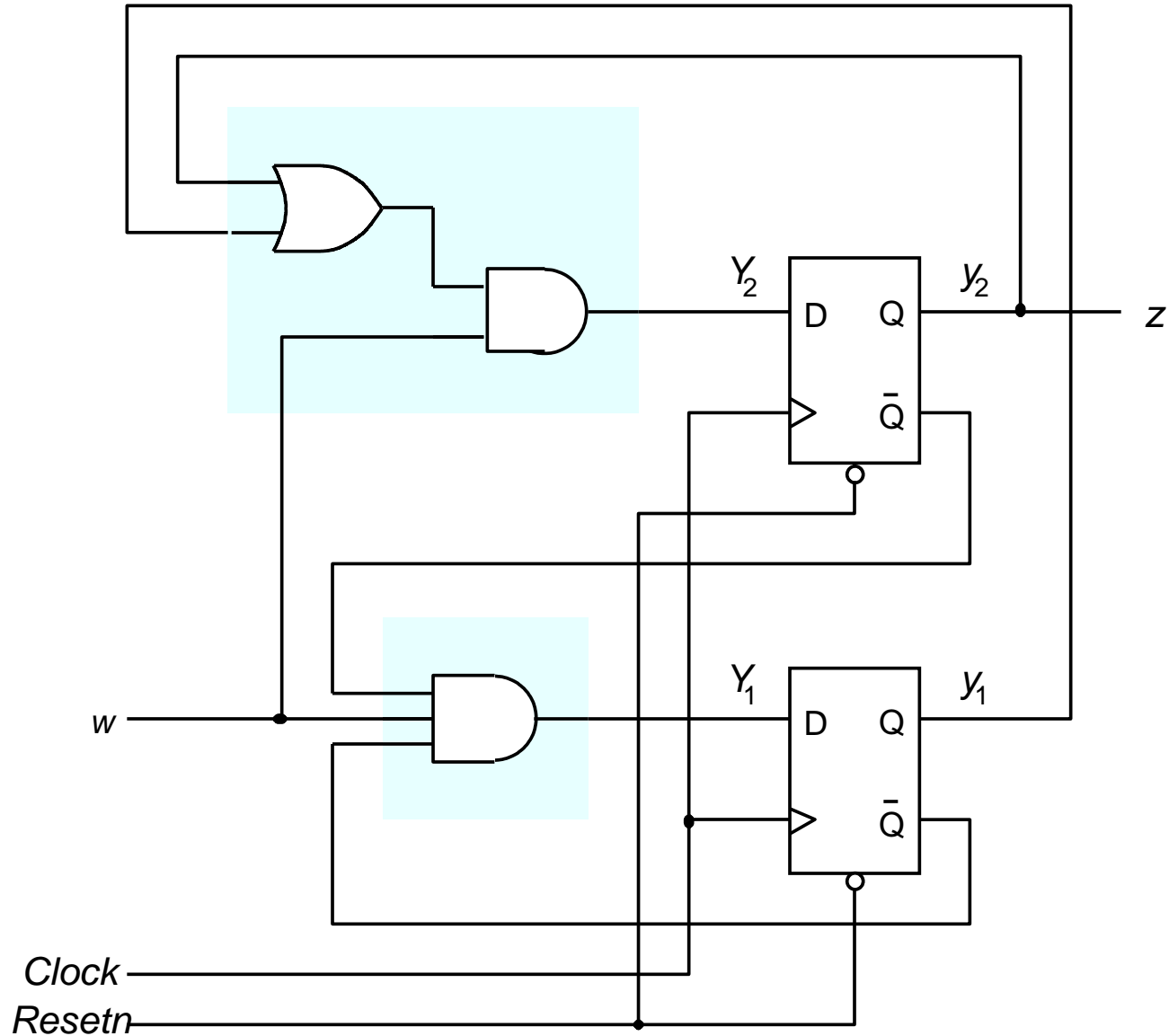
		$y_1$	
		0	1
$y_2$	0	0	0
	1	1	d

$$z = \bar{y}_1y_2$$

$$z = y_2$$



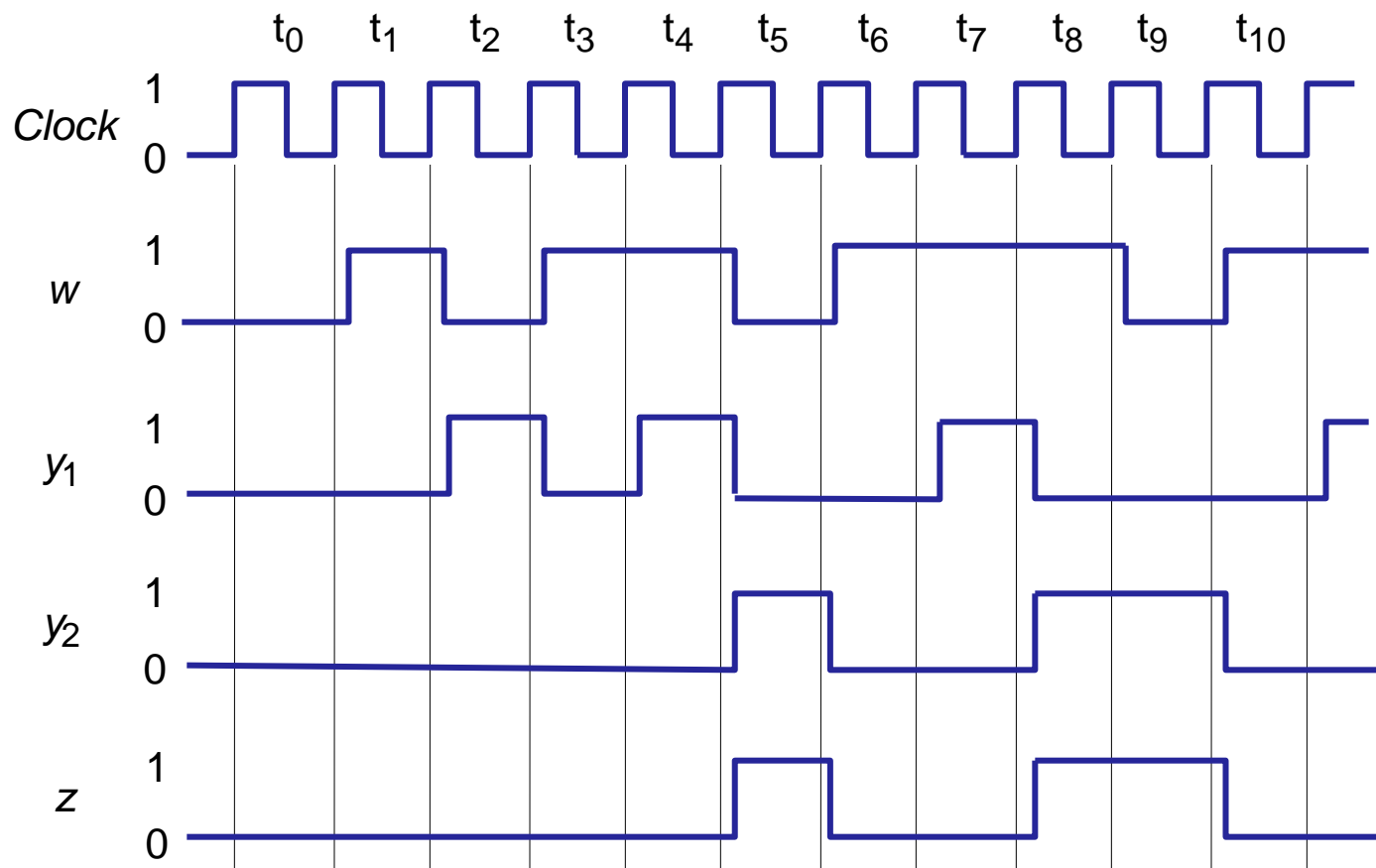
# Circuito final com FF tipo D





# Diagrama de tempo da FSM

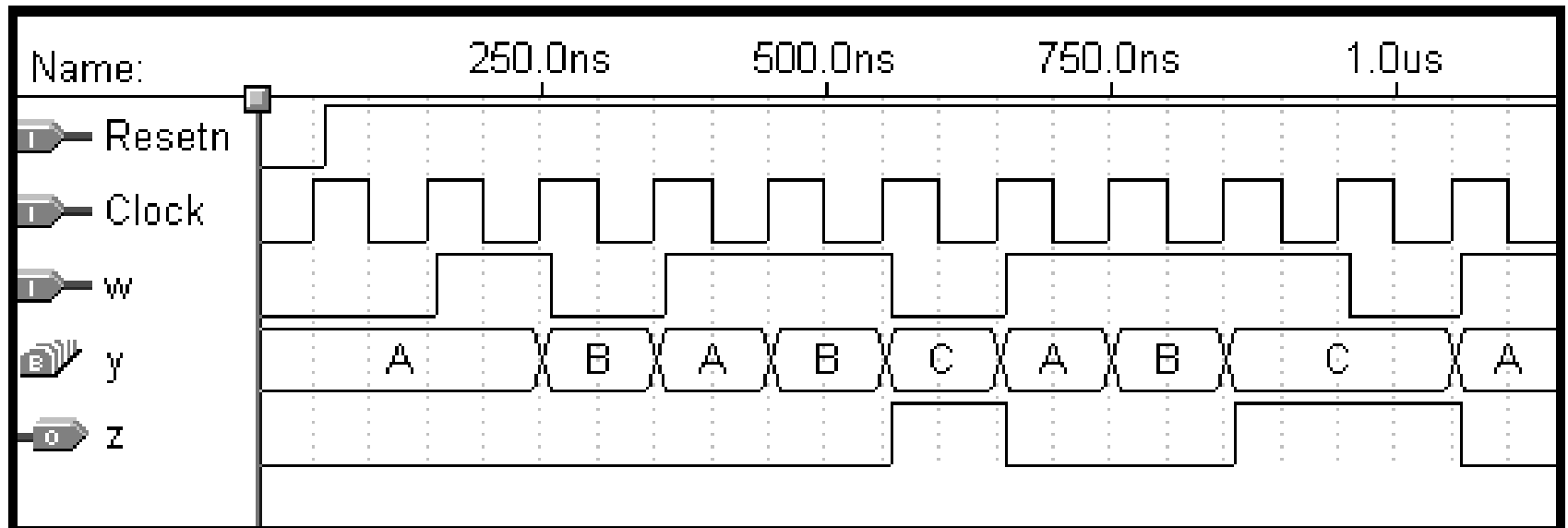
Ciclo:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0



Observar sinais síncronos com borda de subida do clock

# Simulação no Quartus

- Observar
  - agrupamento de sinais das variáveis de estado (tipo enumerado)
  - identificação pelo Quartus da máquina de estado e geração automática do diagrama de transição

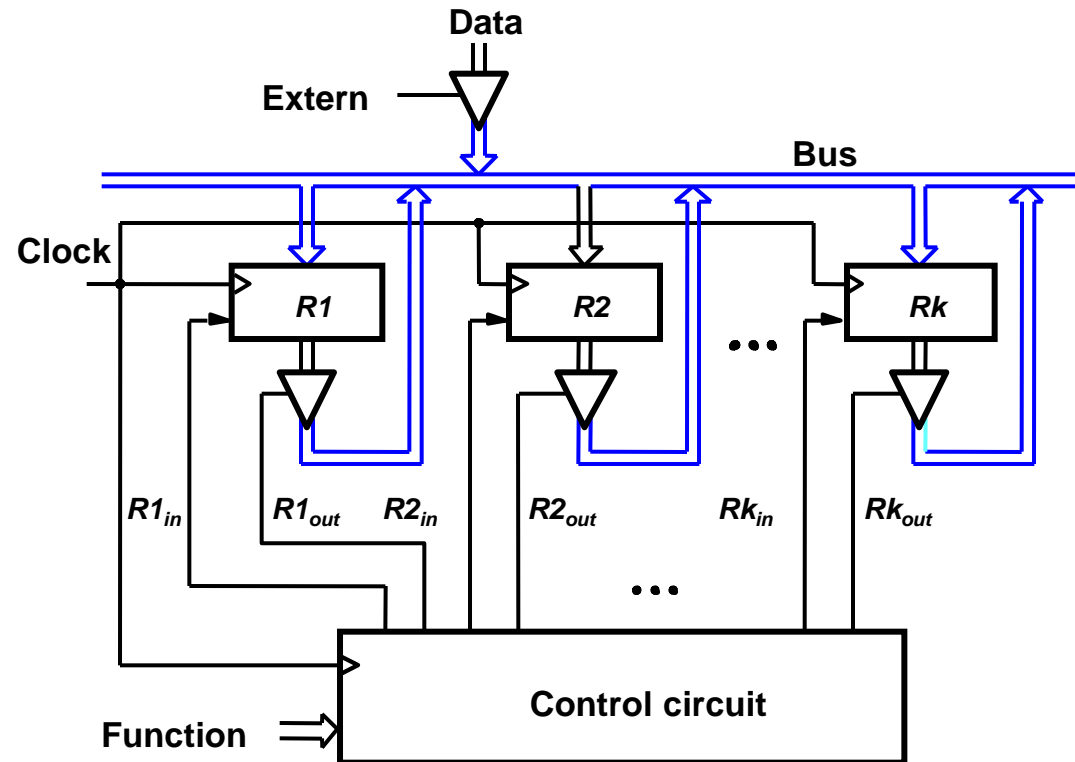
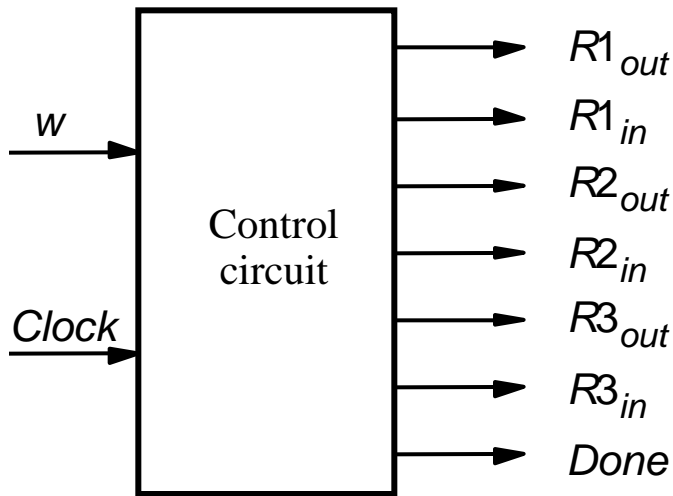


# Resumo do procedimento

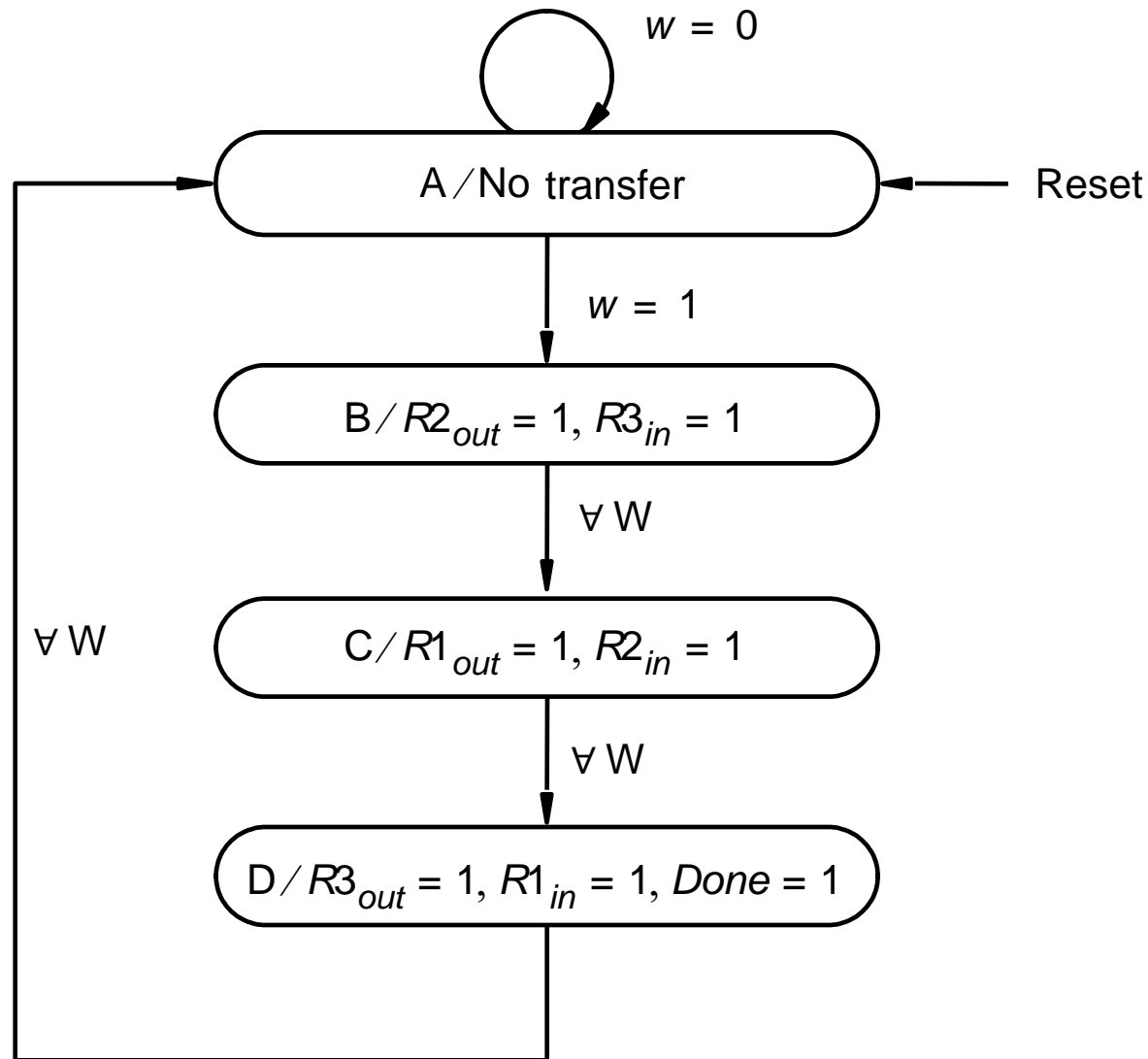
- Especificar o circuito
- Identificar o número de estados necessário e suas transições
- Desenhar o diagrama de transição de estados
- Fazer a tabela de transição de estados
- Fazer a atribuição de estados e completar a tabela
- (decidir o tipo de FF a ser usado e completar a tabela)
- Implementar os circuitos combinacionais CC1 e CC2

# Exemplo de projeto: Expl 8.1

- Implementação da unidade de controle da fig 7.55
- Tarefa: após pulso de start em  $w$ , trocar o conteúdo de  $R1$  e  $R2$ , usando  $R3$  como temporário. Ao final, ativar sinal “done” (?? possível trocar diretamente??)



# Diagrama de transição de estados



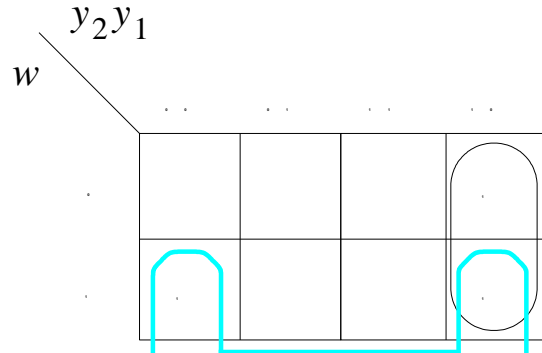


# Tabelas de transição de estado

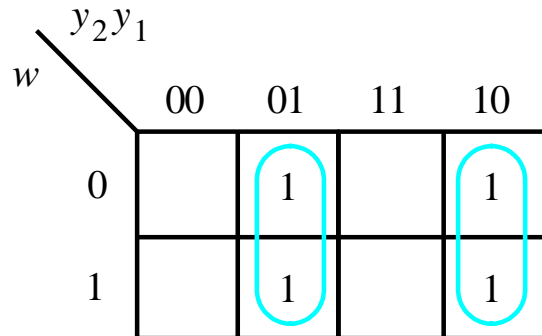
Present state	Next state		Outputs						
	$w = 0$	$w = 1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	A	B	0	0	0	0	0	0	0
B	C	C	0	0	1	0	0	1	0
C	D	D	1	0	0	1	0	0	0
D	A	A	0	1	0	0	1	0	1

	Present state	Nextstate		Outputs						
		$w = 0$	$w = 1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
	$y_2y_1$	$Y_2Y_1$	$Y_2Y_1$							
A	00	00	01	0	0	0	0	0	0	0
B	01	10	10	0	0	1	0	0	1	0
C	10	11	11	1	0	0	1	0	0	0
D	11	00	00	0	1	0	0	1	0	1

# Mapas de Karnaugh



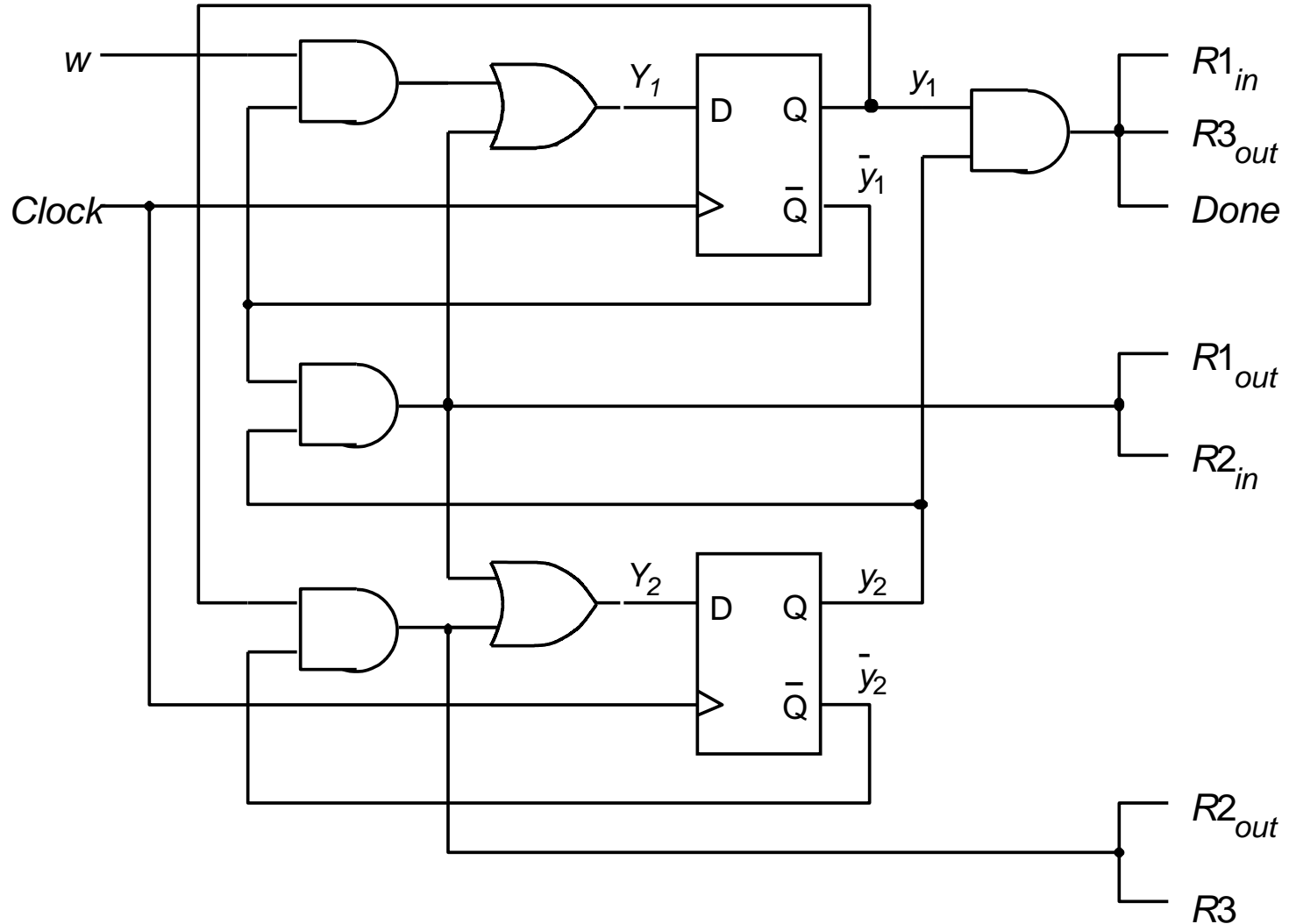
$$Y_1 = w\bar{y}_1 + \bar{y}_1y_2$$



$$Y_2 = y_1\bar{y}_2 + \bar{y}_1y_2$$

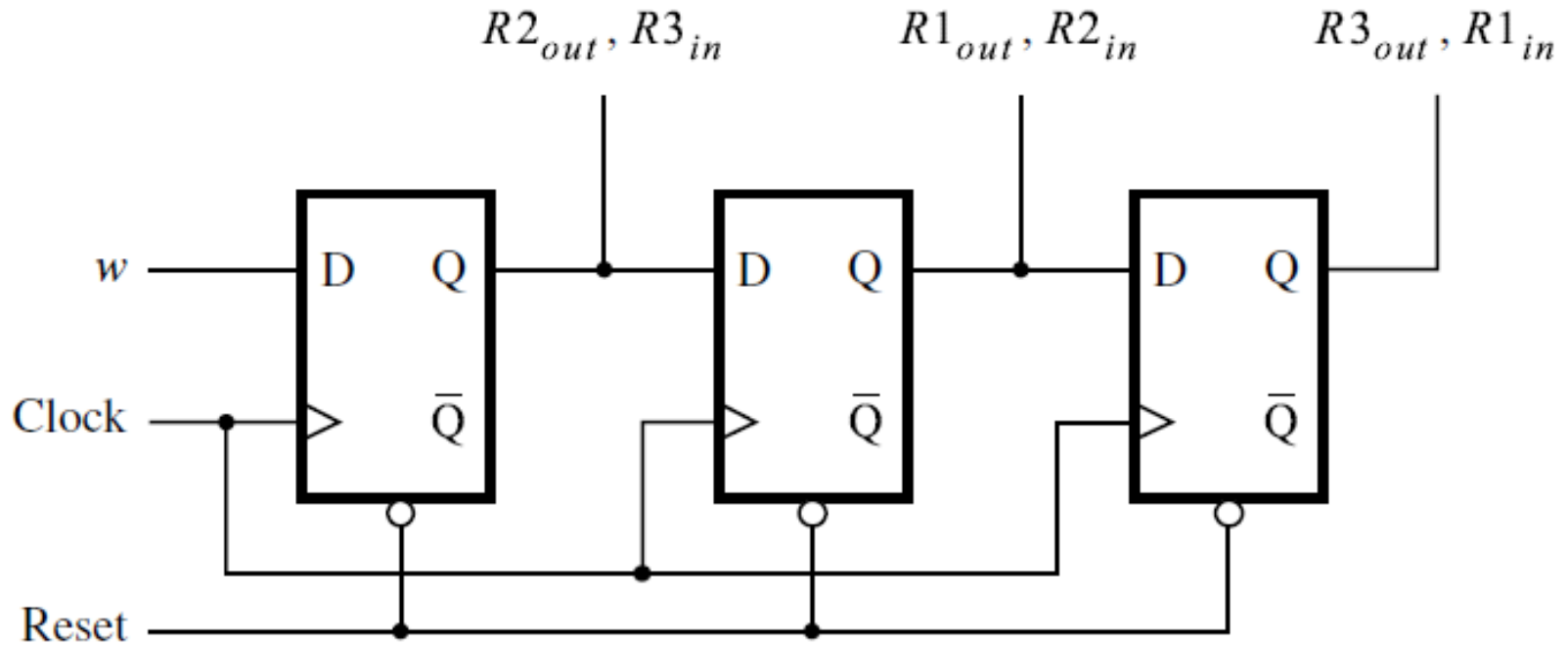


# Circuito final



- (ver outra solução na fig. 7.57, com o uso de registrador de deslocamento)

# Circuito final: solução c/ Shift

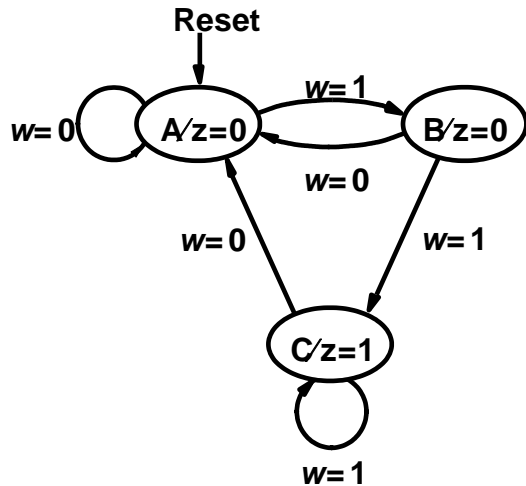


**Figure 7.57** A shift-register control circuit.

# O problema da atribuição de estados

- Nos exemplos anteriores → codificação dos estados foi arbitrária
- É possível escolher atribuição visando minimizar o circuito final
  - solução ótima é difícil
- Vejamos os dois últimos exemplos em outra codificação

# Exemplo: detector de seq de dois 1s



Estado atual	Próx estado		saída $z$
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	C	1

atribuição original

	Present state $y_2y_1$	Next state		Output $z$
		$w = 0$	$w = 1$	
		$Y_2Y_1$	$Y_2Y_1$	
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	<i>dd</i>	<i>dd</i>	<i>d</i>

atribuição melhorada

	Present state $y_2y_1$	Next state		Output $z$
		$w = 0$	$w = 1$	
		$Y_2Y_1$	$Y_2Y_1$	
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	<i>dd</i>	<i>dd</i>	<i>d</i>

# Síntese do novo circuito

- Possível escrever equações diretamente
- $Y_1 = w$
- $Y_2 = w \cdot y_1$
- $z = y_2$

atribuição melhorada

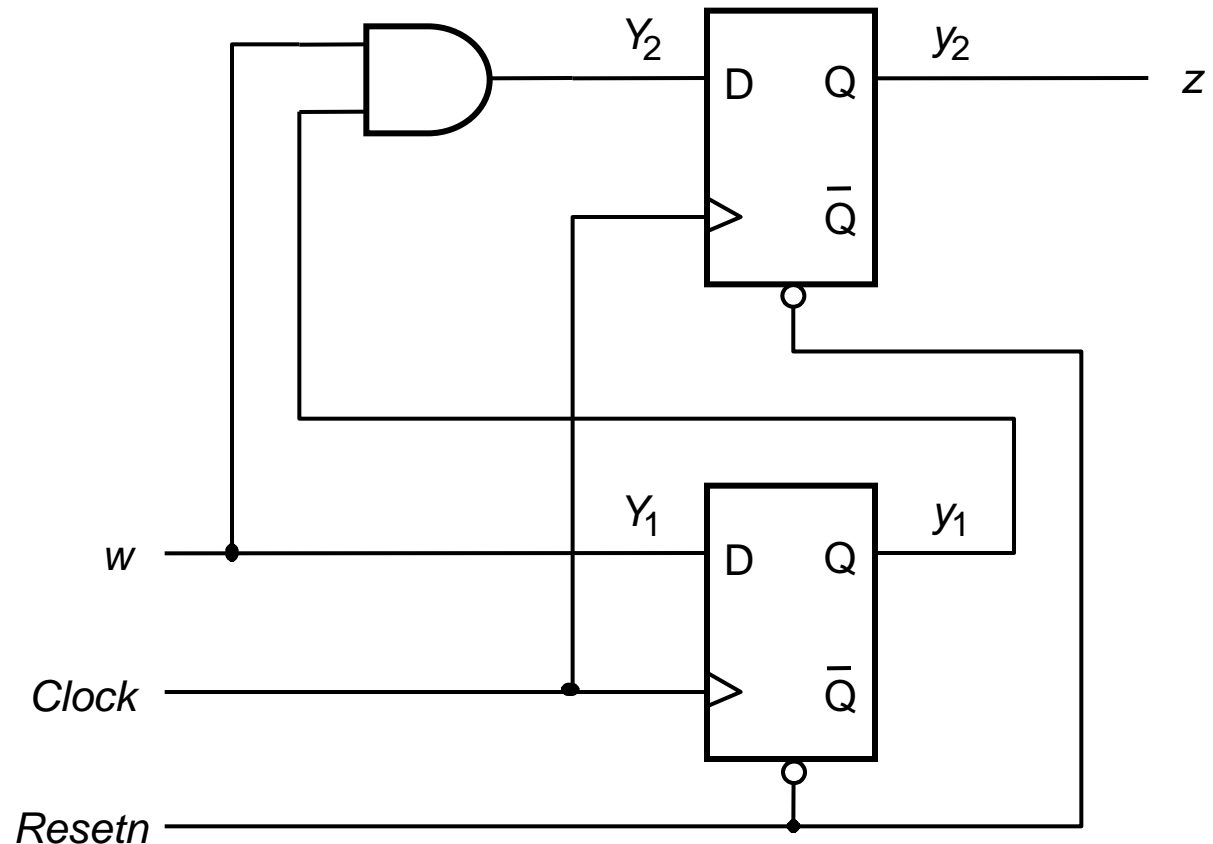
	Present state	Next state		Output $z$
		$w = 0$	$w = 1$	
	$y_2y_1$	$Y_2Y_1$	$Y_2Y_1$	
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	<i>dd</i>	<i>dd</i>	<i>d</i>



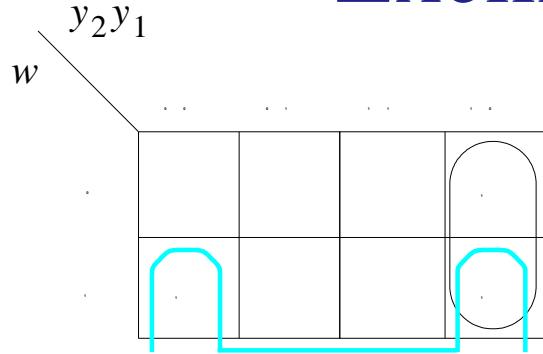
# Circuito final com atribuição melhorada

- $Y_1 = w$
- $Y_2 = w \cdot y_1$
- $z = y_2$

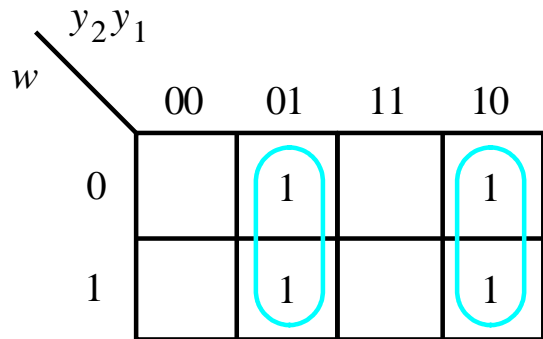
circuito antigo



# Exemplo 8.1 antigo



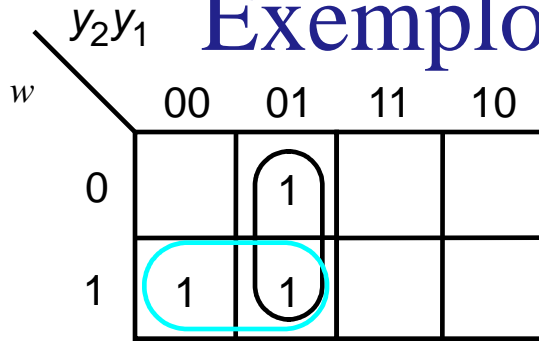
$$Y_1 = wy\bar{1} + \bar{y}_1y_2$$



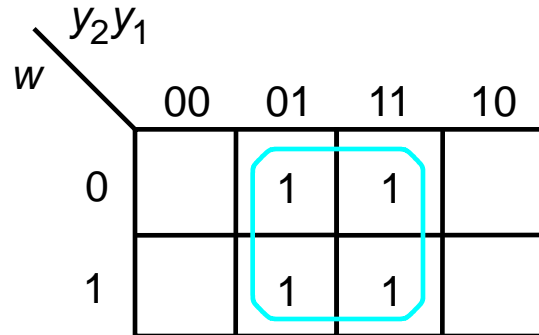
$$Y_2 = y_1\bar{y}_2 + \bar{y}_1y_2$$

	Present state	Nextstate		Outputs						
		$w = 0$	$w = 1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	Done
		$y_2y_1$	$Y_2Y_1$							
A	00	00	01	0	0	0	0	0	0	0
B	01	10	10	0	0	1	0	0	1	0
C	10	11	11	1	0	0	1	0	0	0
D	11	00	00	0	1	0	0	1	0	1

# Exemplo 8.1 com nova atribuição



$$Y_1 = w\bar{y}_2 + y_1\bar{y}_2$$



$$Y_2 = y_1$$

	Present state	Nextstate		Outputs						
		$w = 0$	$w = 1$							
	$y_2y_1$	$Y_2Y_1$	$Y_2Y_1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	01	0	0	0	0	0	0	0
B	01	11	11	0	0	1	0	0	1	0
C	11	10	10	1	0	0	1	0	0	0
D	10	00	00	0	1	0	0	1	0	1





# One-hot encoding

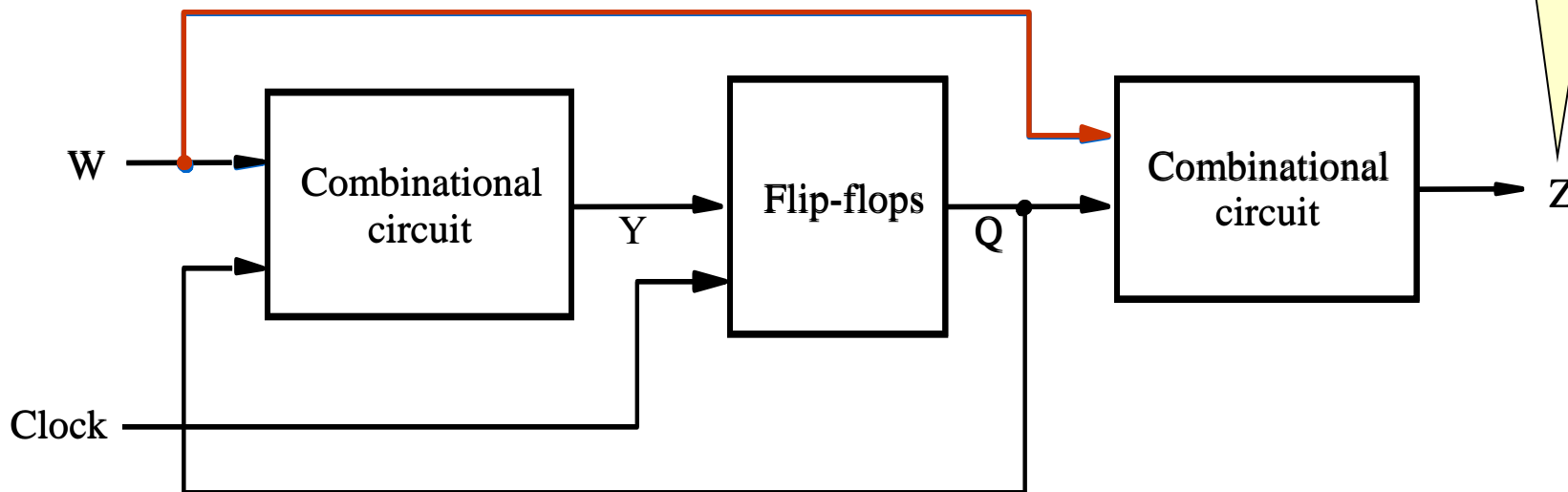
- É possível codificar os estados com  $n$  bits para  $n$  estados
  - apenas um bit fica ligado (one-hot)
- Grande número de don't care (combinações de estados não utilizados)
- Vantagem: circuito combinacional pode ficar mais simples
- Desvantagem: usa maior número de FFs
- Ver detalhes na seção 8.2.1 do livro texto



# Máquinas de Mealy

- Máquinas de Moore
  - próximo conteúdo dos flip-flops (Y) depende das entradas (W) e do estado atual (Q)
  - saída depende do estado atual (Q)
- Máquina de Mealy
  - próximo conteúdo dos flip-flops (Y) depende das entradas (W) e do estado atual (Q)
  - saída depende do estado atual (Q) **e das entradas (W)**

**Mealy:** z pode mudar a qquer instante, indep. do clock

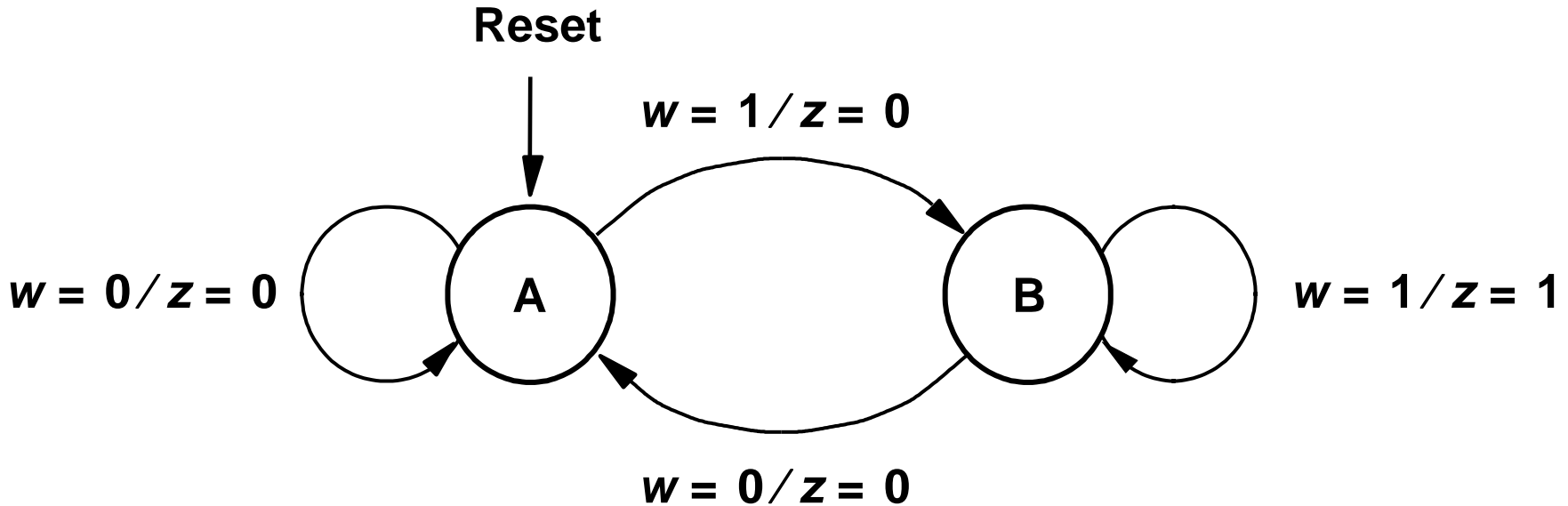


# Implementação Mealy do mesmo exemplo

- Especificações alteradas:
  - O circuito tem uma entrada  $w$  e uma saída  $z$
  - ~~Todas as mudanças ocorrem na borda de subida do clock~~
  - $z=1$  se  $w=1$  ~~nos dois últimos ciclos de clock~~ no último clock e no clock atual
  - $z=0$  caso contrário

Clock cycle:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
$w$ :	0	1	0	1	1	0	1	1	1	0	1
$z$ :	0	0	0	0	1	0	0	1	1	0	0

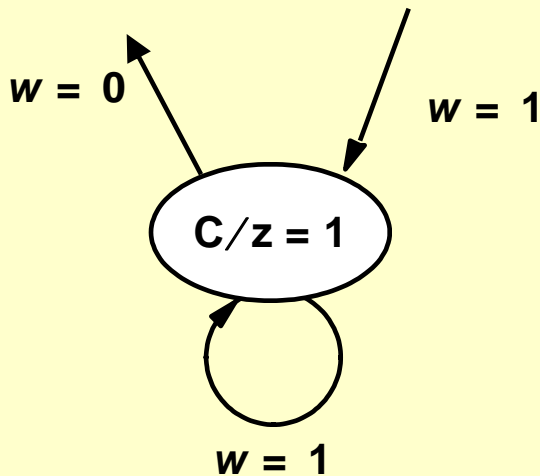
# Diagrama de transição de estados da Máquina de Mealy



Clock cycle:	$t_0$	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$t_7$	$t_8$	$t_9$	$t_{10}$
$w$ :	0	1	0	1	1	0	1	1	1	0	1
$z$ :	0	0	0	0	1	0	0	1	1	0	0



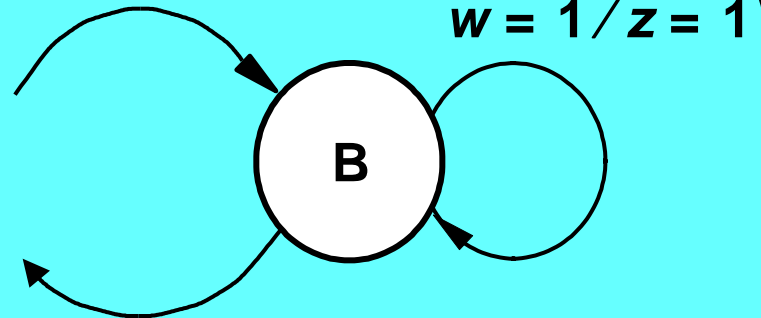
# Notação Moore e Mealy



## Moore

- Saída definida pelo estado atual
- Arcos de transição de estados não mostram saídas

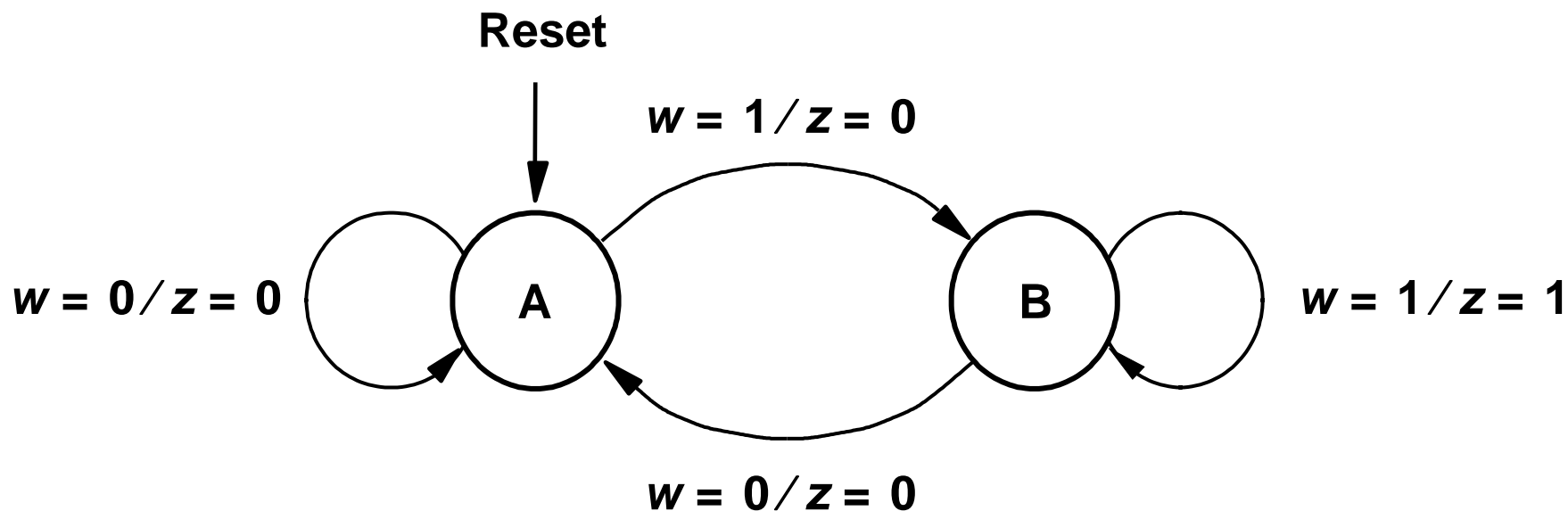
$w = 1 / z = 0$



## Mealy

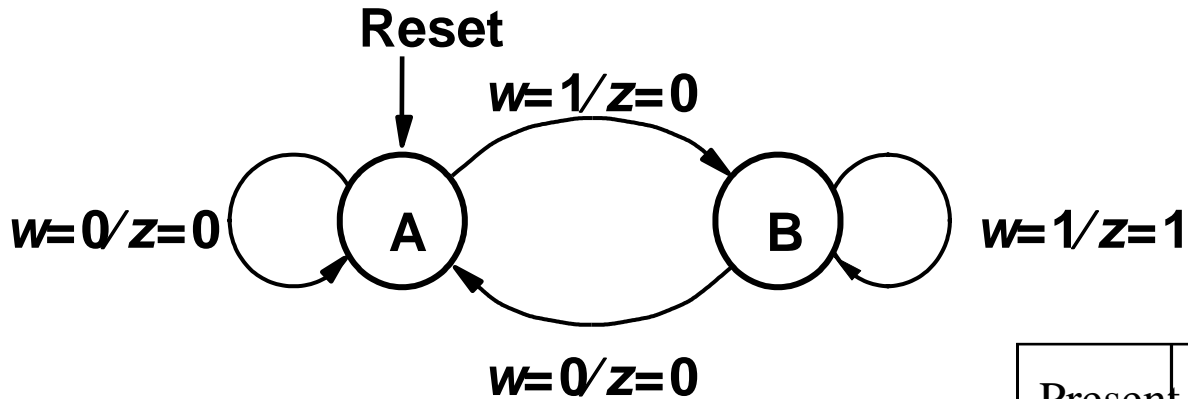
- O estado atual pode ter várias saídas, em função de  $w$
- Arcos de transição mostram  $w$  e  $z$

# Tabela de transição de estados



Present state	Next state		Output $z$	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	A	B	0	1

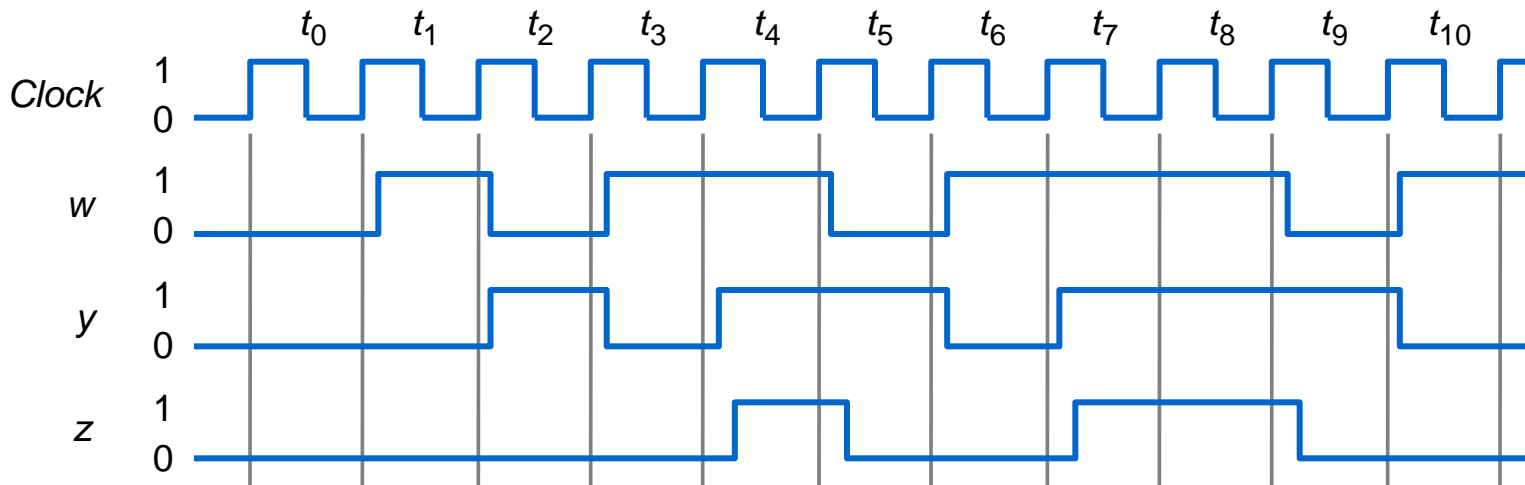
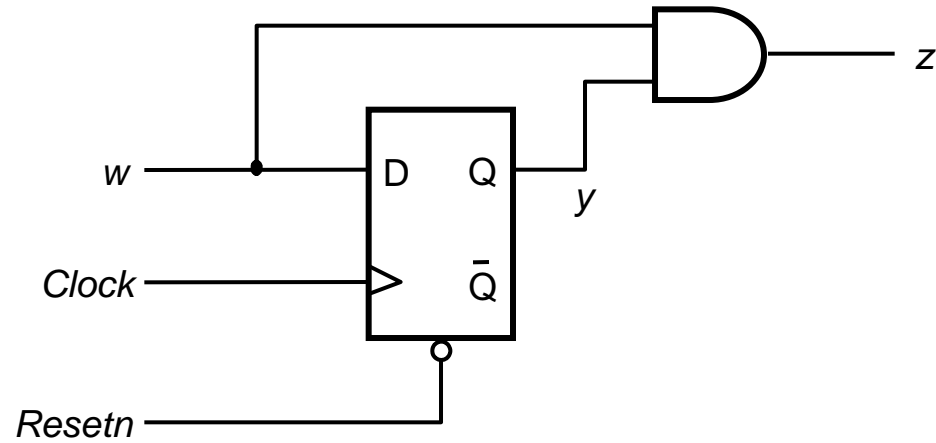
# Tabela de transição de estados



Present state	Next state		Output	
	w = 0	w = 1	w = 0	w = 1
A	A	B	0	0
B	A	B	0	1

	Present state	Next state		Output	
		w = 0	w = 1	w = 0	w = 1
	y	Y	Y	z	z
A	0	0	1	0	0
B	1	0	1	0	1

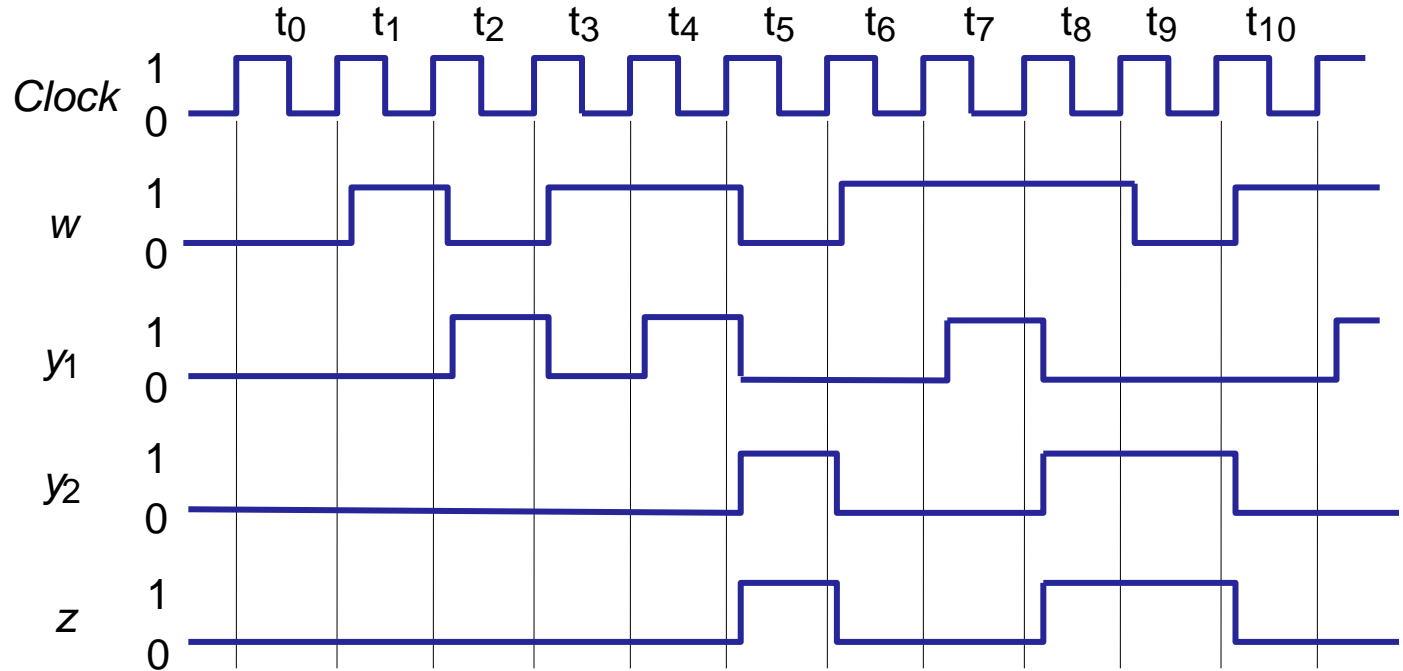
# Circuito e seu comportamento



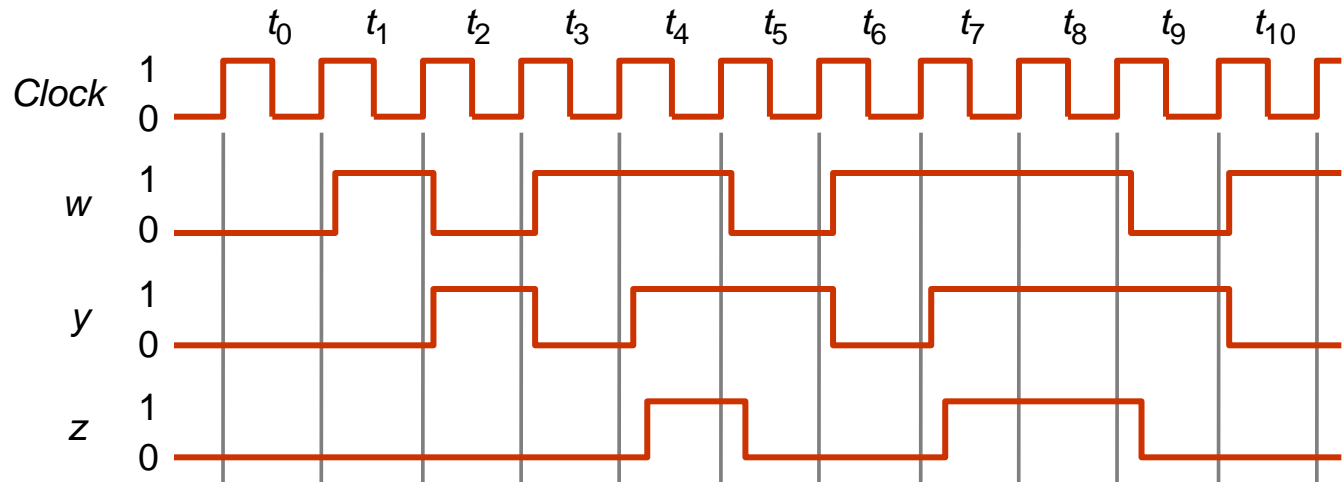


# Comportamento comparado

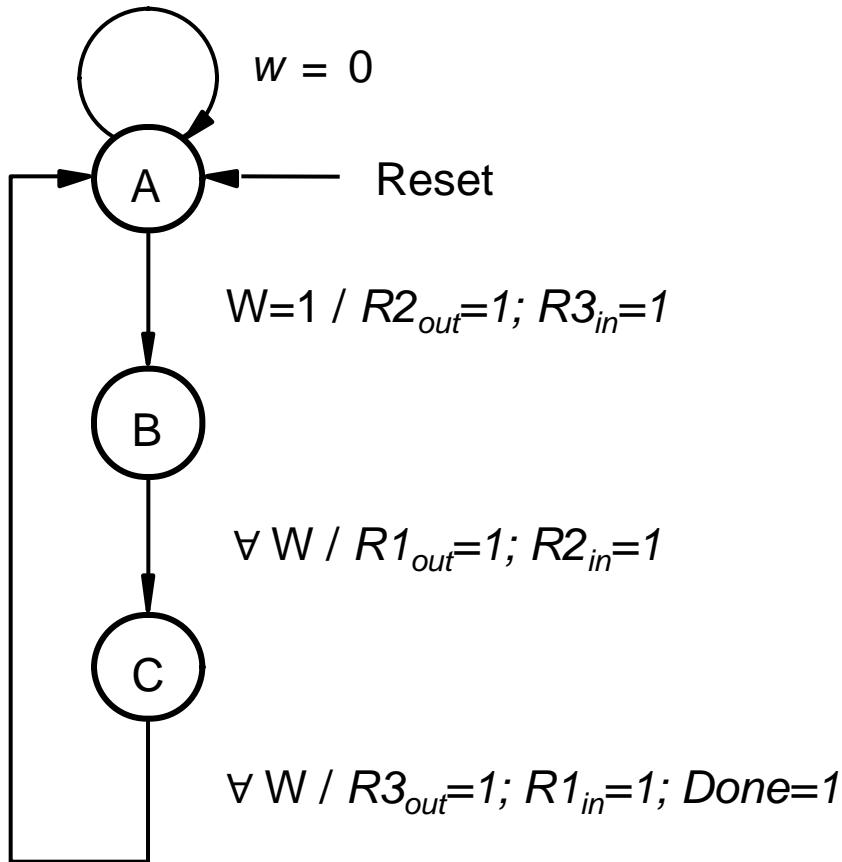
Moore



Mealy



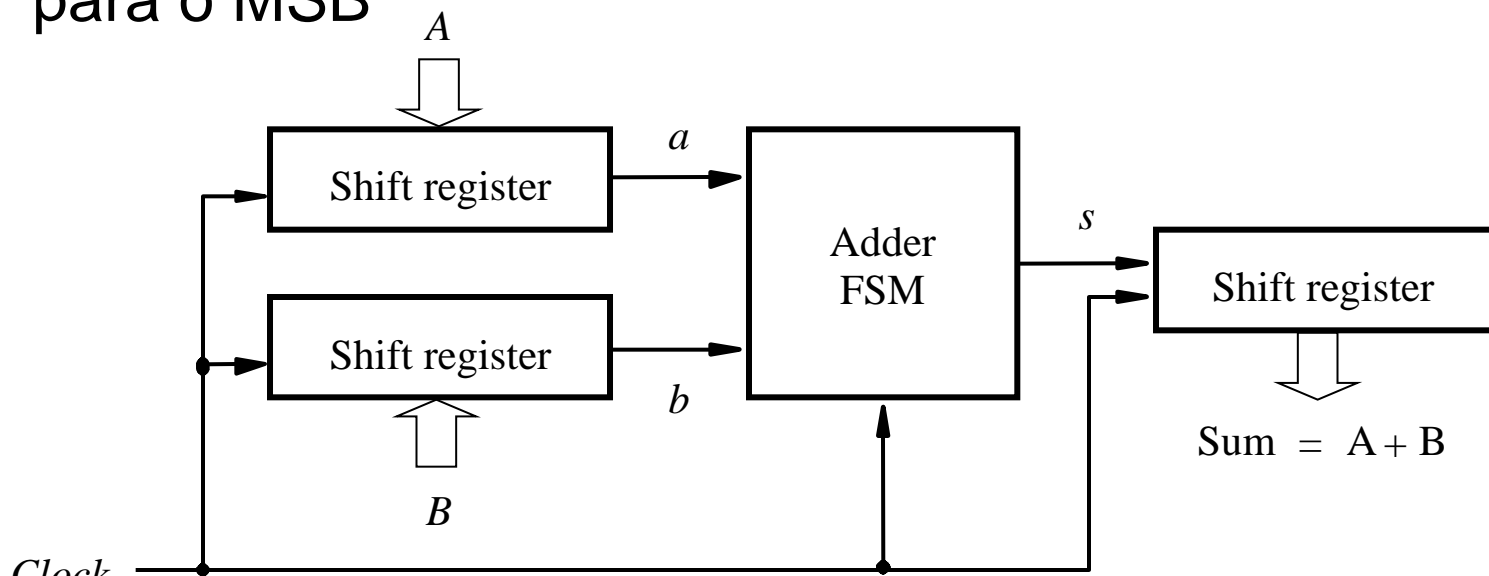
# Exemplo 8.4 (Mealy em expl 8.1)



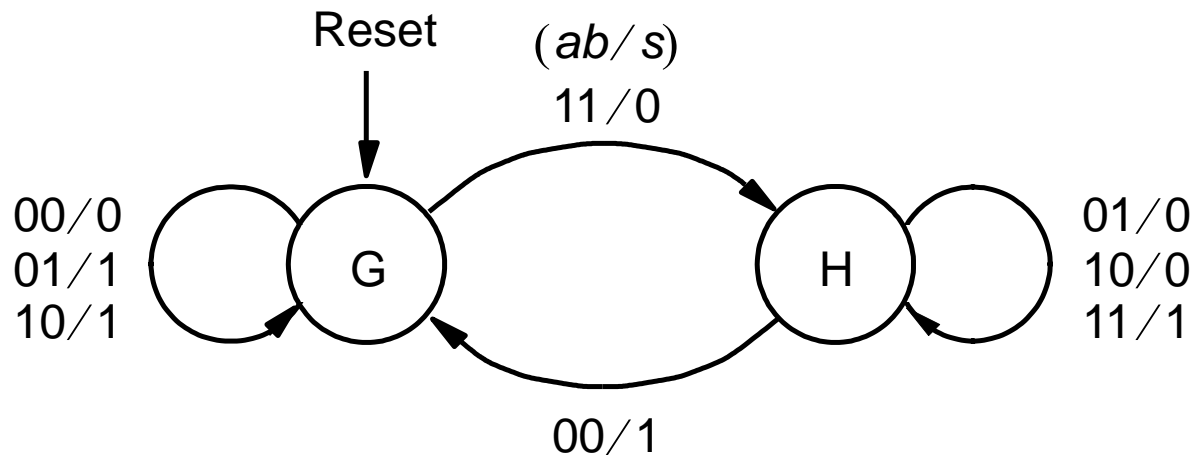
- troca de conteúdo dos registradores R1 e R2, usando R3 como temporário
- diagrama de transição de estados

## 8.5 Exemplo: somador serial

- Projetar um somador serial
  - operandos de  $n$  bits armazenados nos shifts A e B
  - resultado ficará armazenado no shift  $A+B$
  - “Adder FSM” cuida da soma e do carry do bit anterior
  - bits são apresentados ao “Adder FSM” do LSB para o MSB

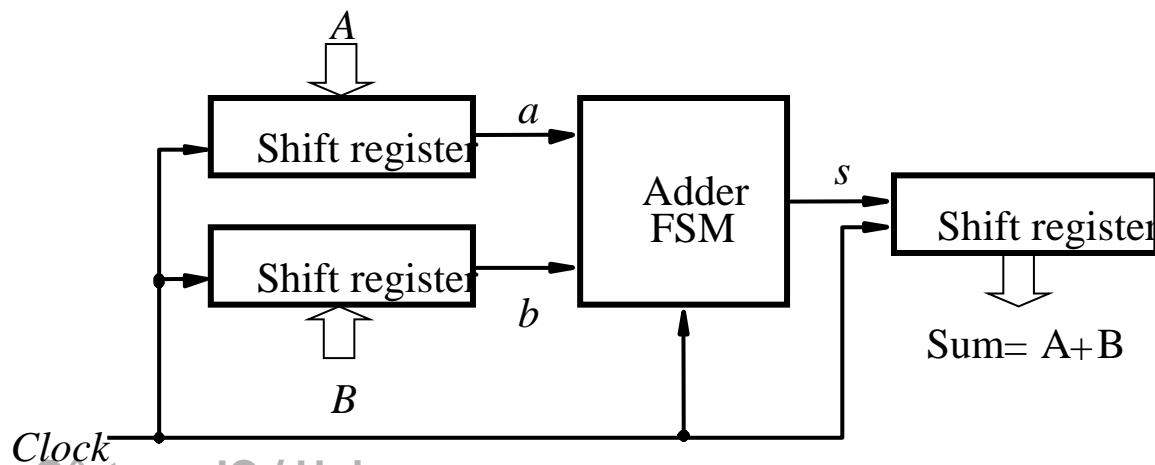


# Projeto somador serial Mealy



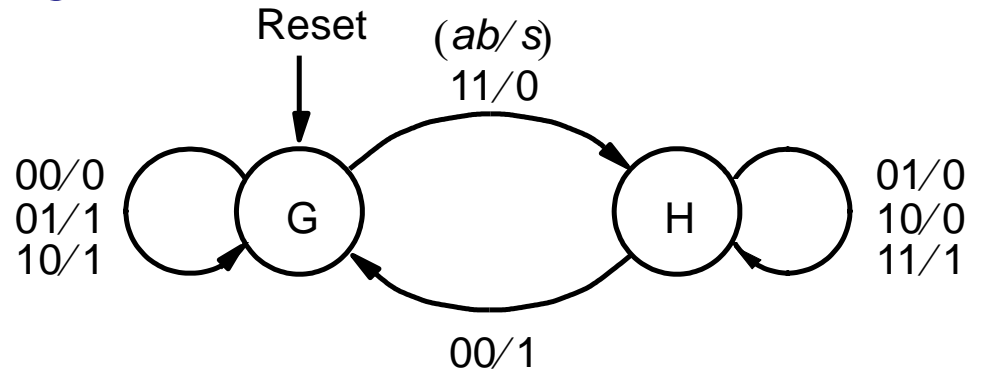
G: carry-in = 0

H: carry-in = 1



# Tabelas de transição de estados

G: carry-in = 0  
 H: carry-in = 1



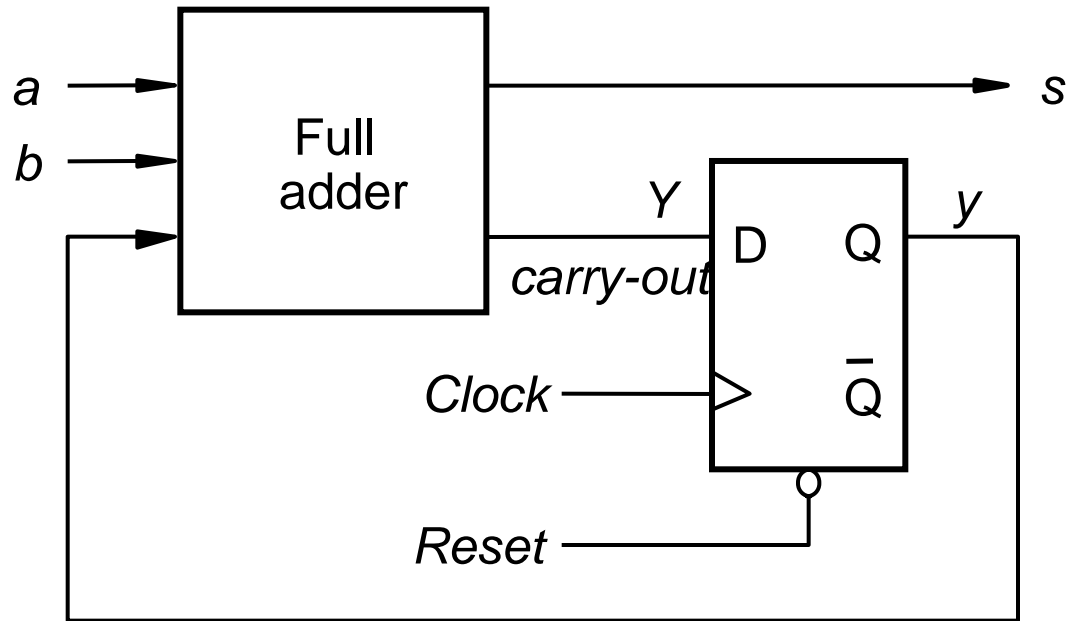
Estados genéricos

Present state	Next state				Output <i>s</i>			
	<i>ab</i> = 00	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

Estados atribuídos

Present state	Next state				Output			
	<i>ab</i> = 00	01	10	11	00	01	10	11
<i>y</i>	<i>Y</i>				<i>s</i>			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

# Circuito do Somador Serial Mealy



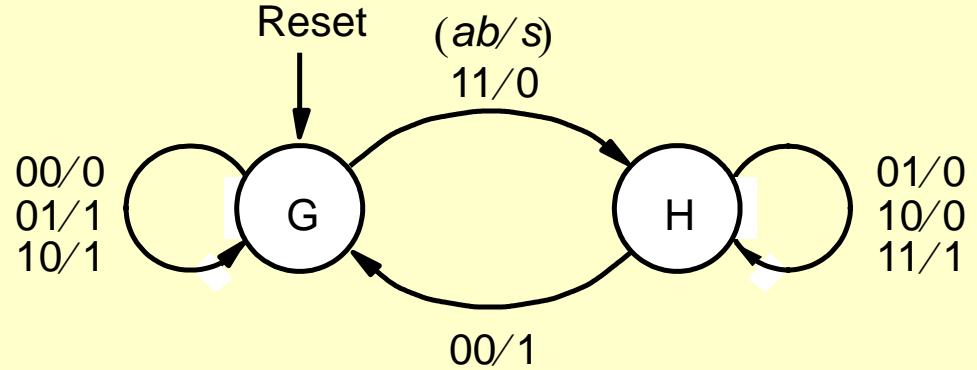
- Poderia ser projetado por inspeção (e tentativa e erro)



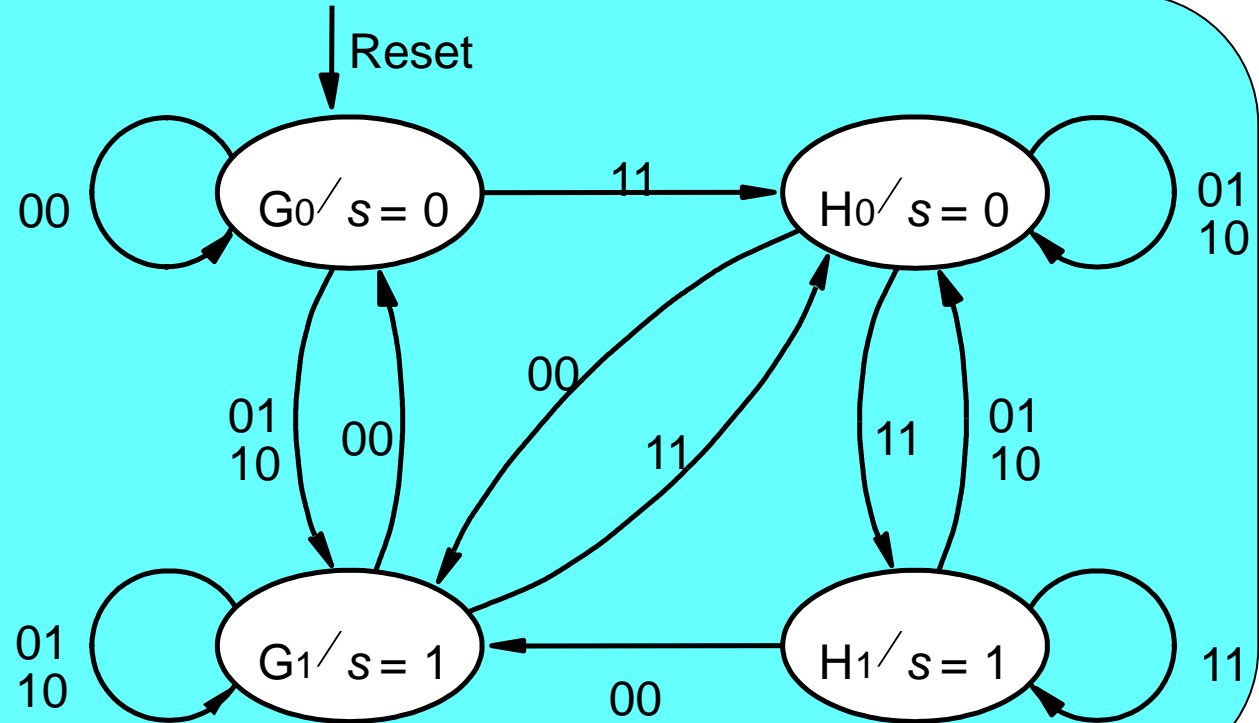
# Projeto somador serial Moore

## Mealy

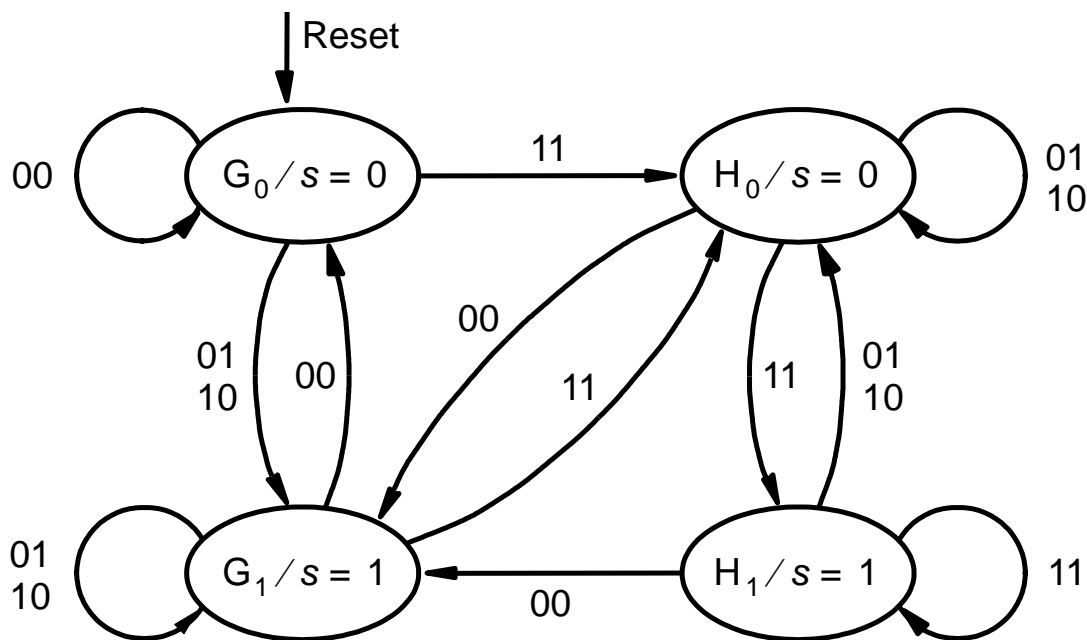
G: carry-in = 0  
H: carry-in = 1



## Moore



# Tabelas de transição de estados

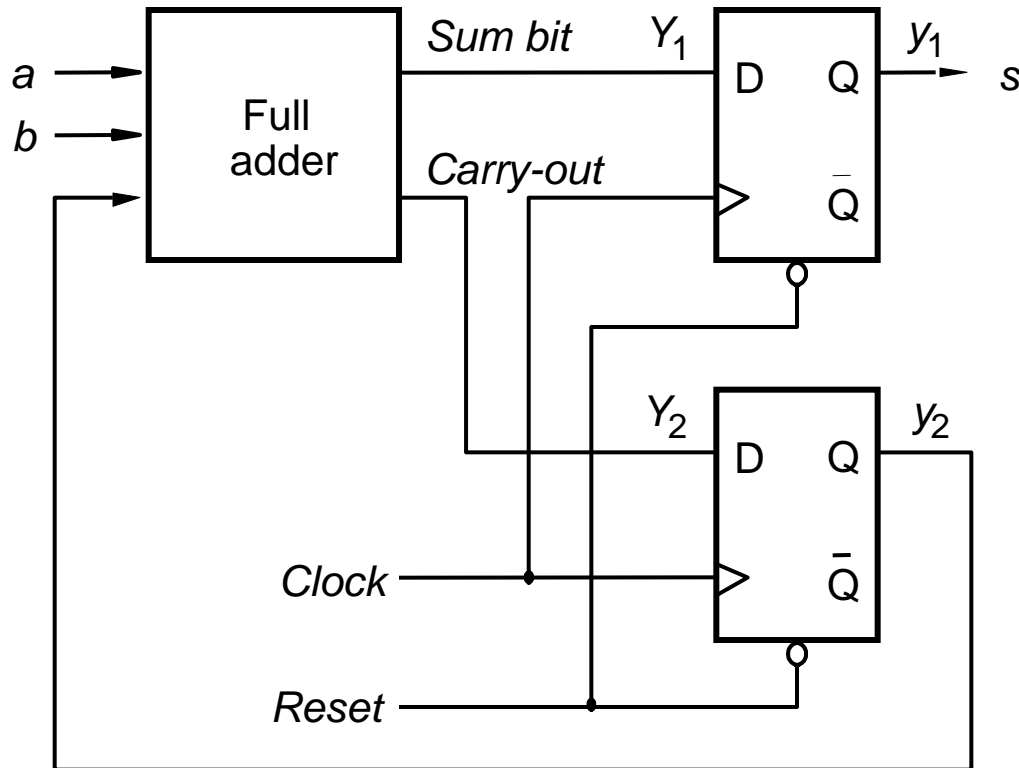


Present state	Nextstate				Output $s$
	$ab=00$	01	10	11	
$G_0$	$G_0$	$G_1$	$G_1$	$H_0$	0
$G_1$	$G_0$	$G_1$	$G_1$	$H_0$	1
$H_0$	$G_1$	$H_0$	$H_0$	$H_1$	0
$H_1$	$G_1$	$H_0$	$H_0$	$H_1$	1

Present state $y_2y_1$	Nextstate				Output $s$
	$ab=00$	01	10	11	
	$Y_2Y_1$				
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1



# Circuito do Somador Serial Moore



- Nos dois casos, apesar de mais complicada, a máquina Moore é mais robusta:
  - mudanças na entrada não afetam imediatamente a saída e só no próximo clock

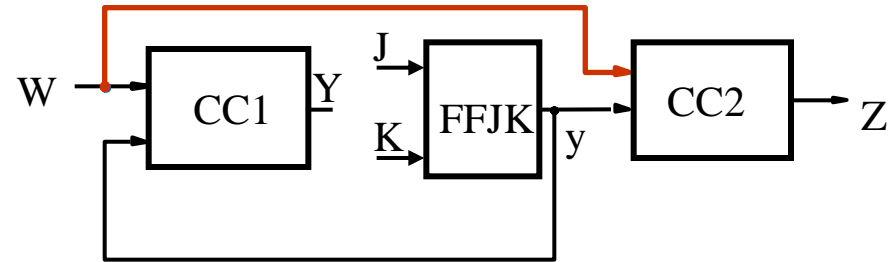
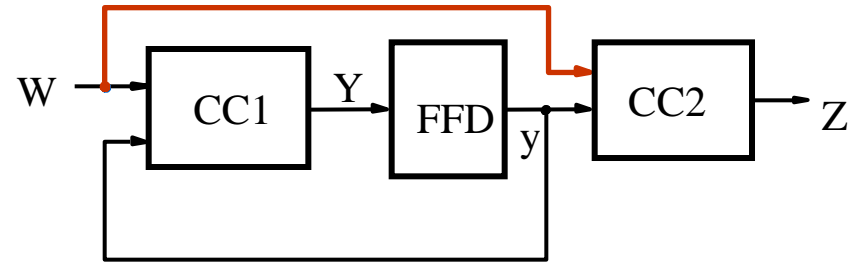
# Projeto de FSM com FF-D e FF-JK

- Visto até agora:  
FSM com FF-D

$$- y_{i+1} = Y_i$$

- E se FF-JK?

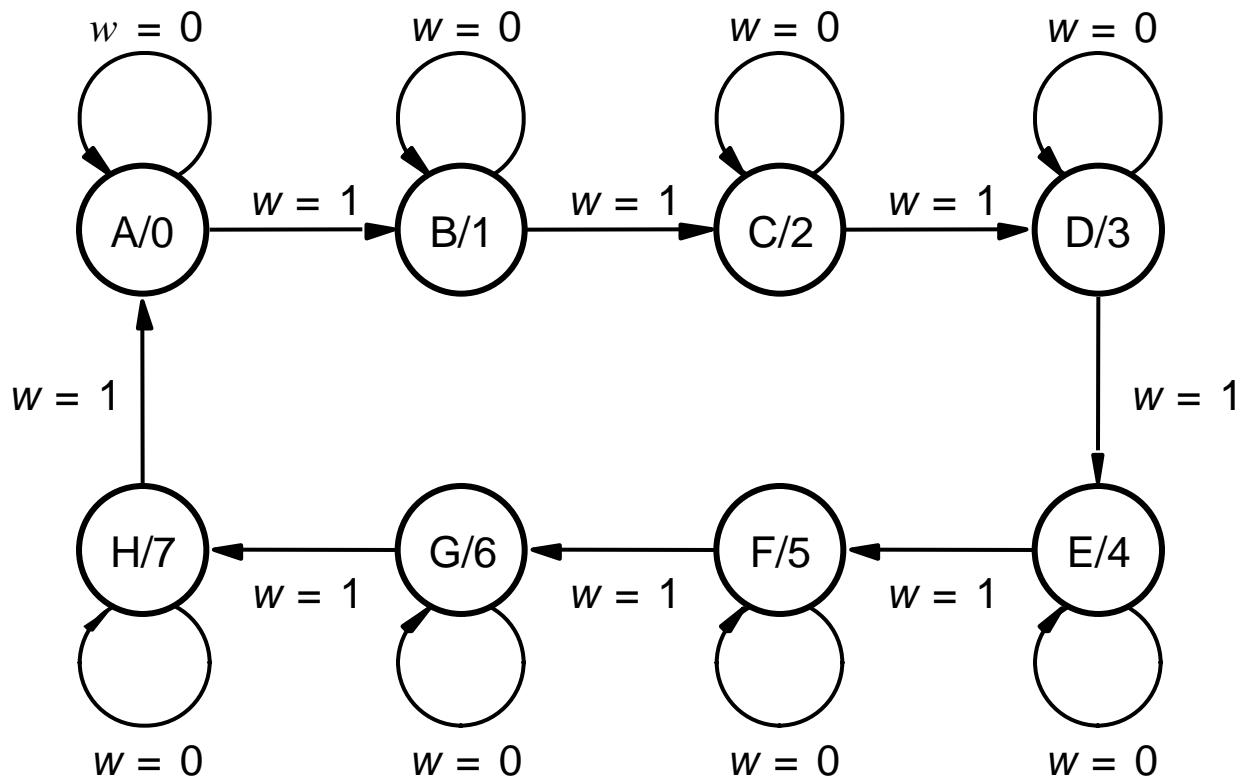
- Derivar de Y  
(próximo estado)  $\rightarrow$  J e K



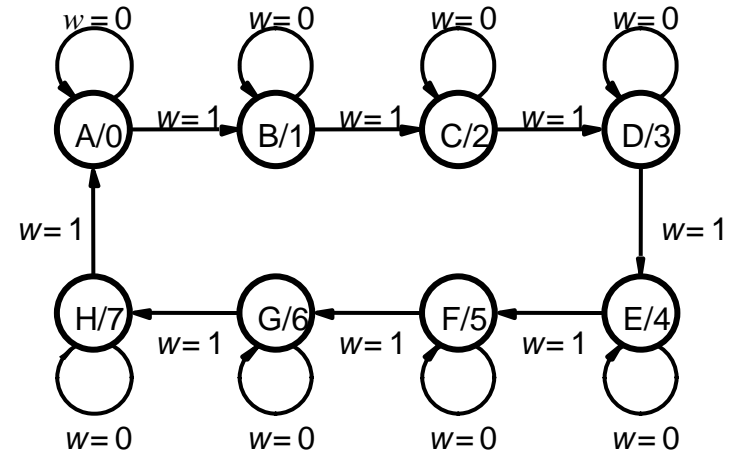
Estado atual $\rightarrow$ Próximo estado	J	K
0 $\rightarrow$ 0	0	d
0 $\rightarrow$ 1	1	d
1 $\rightarrow$ 1	d	0
1 $\rightarrow$ 0	d	1

# Exemplo: contador como FSM (JK)

- Contador síncrono mod 8: 0 1 2 3 4 5 6 7 0 ...



# Tabelas de transição de estados



Present state	Next state		Output
	$w = 0$	$w = 1$	
A	A	B	0
B	B	C	1
C	C	D	2
D	D	E	3
E	E	F	4
F	F	G	5
G	G	H	6
H	H	A	7

	Present state $y_2y_1y_0$	Next state		Count $z_2z_1z_0$
		$w = 0$	$w = 1$	
		$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	
A	000	000	001	000
B	001	001	010	001
C	010	010	011	010
D	011	011	100	011
E	100	100	101	100
F	101	101	110	101
G	110	110	111	110
H	111	111	000	111

# Derivação de J K do FF

Transição	J	K
0 → 0	0	d
0 → 1	1	d
1 → 1	d	0
1 → 0	d	1

	Present state $y_2y_1y_0$	Next state		Count $z_2z_1z_0$
		$w = 0$	$w = 1$	
		$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	
A	000	000	001	000
B	001	001	010	001
C	010	010	011	010
D	011	011	100	011
E	100	100	101	100
F	101	101	110	101
G	110	110	111	110
H	111	111	000	111

	Present state $y_2y_1y_0$	Flip-flop inputs							Count $z_2z_1z_0$	
		$w = 0$			$w = 1$					
		$Y_2Y_1Y_0$	$J^2K^2$	$J^1K^1$	$J^0K^0$	$Y_2Y_1Y_0$	$J^2K^2$	$J^1K^1$		$J^0K^0$
A	000	000	0d	0d	0d	001	0d	0d	1d	000
B	001	001	0d	0d	d0	010	0d	1d	d1	001
C	010	010	0d	d0	0d	011	0d	d0	1d	010
D	011	011	0d	d0	d0	100	1d	d1	d1	011
E	100	100	d0	0d	0d	101	d0	0d	1d	100
F	101	101	d0	0d	d0	110	d0	1d	d1	101
G	110	110	d0	d0	0d	111	d0	d0	1d	110
H	111	111	d0	d0	d0	000	d1	d1	d1	111

# Mapas de Karnaugh para Js e Ks

$wy_2$	$y_1y_0$	00	01	11	10
00	0	0	0	0	0
01	d	d	d	d	d
11	d	d	d	d	d
10	0	0	1	0	

$$J_2 = wy_0y_1$$

$wy_2$	$y_1y_0$	00	01	11	10
00	0	0	d	d	
01	0	0	d	d	
11	0	1	d	d	
10	0	1	d	d	

$$J_1 = wy_0$$

$wy_2$	$y_1y_0$	00	01	11	10
00	0	d	d	0	
01	0	d	d	0	
11	1	d	d	1	
10	1	d	d	1	

$$J_0 = w$$

$wy_2$	$y_1y_0$	00	01	11	10
00	d	d	d	d	
01	0	0	0	0	
11	0	0	1	0	
10	d	d	d	d	

$$K_2 = wy_0y_1$$

$wy_2$	$y_1y_0$	00	01	11	10
00	d	d	0	0	
01	d	d	0	0	
11	d	d	1	0	
10	d	d	1	0	

$$K_1 = wy_0$$

$wy_2$	$y_1y_0$	00	01	11	10
00	d	0	0	d	
01	d	0	0	d	
11	d	1	1	d	
10	d	1	1	d	

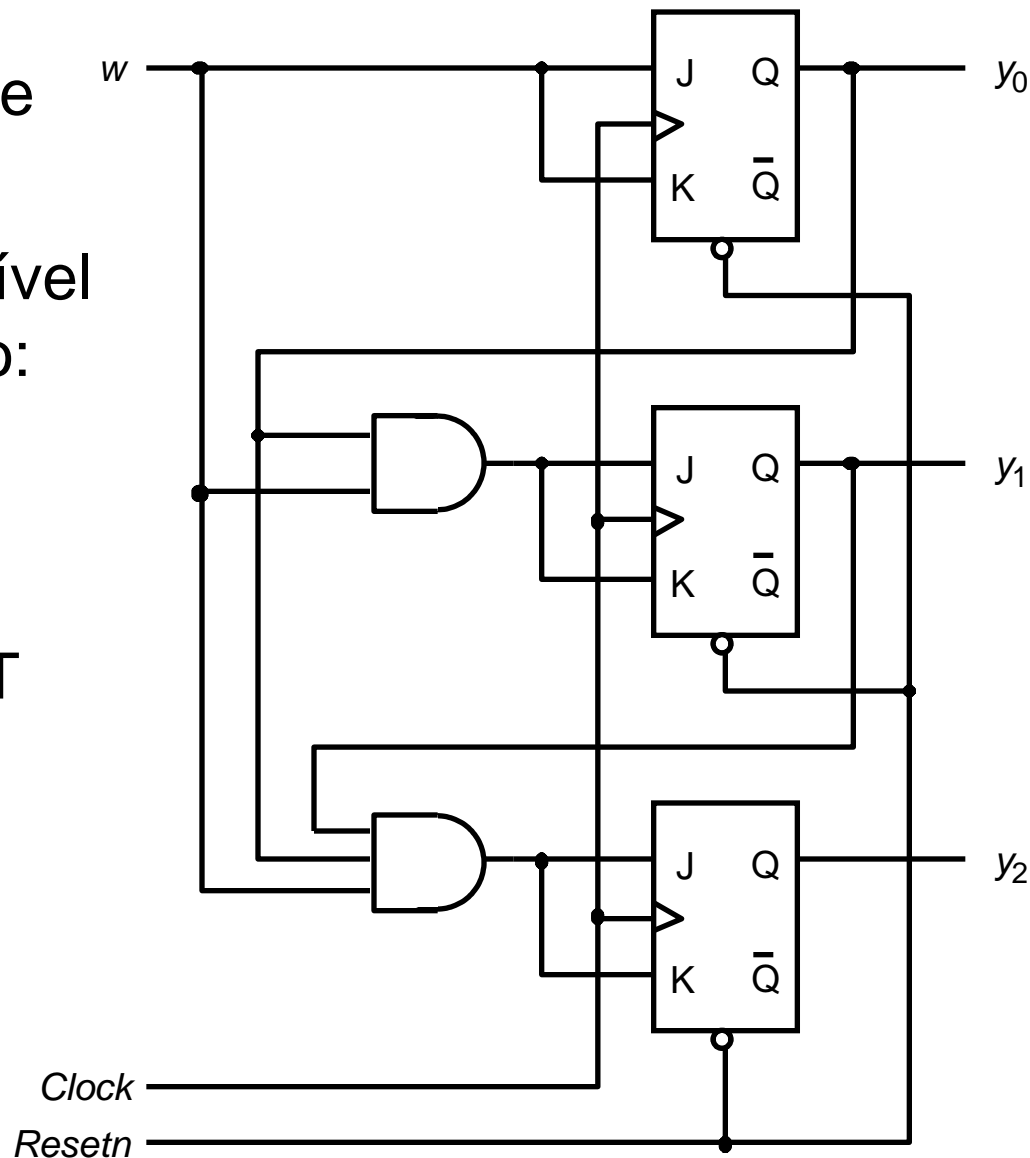
$$K_0 = w$$

# Circuito final com JKs

- Observar regularidade
- $J=K= w y_0 y_1 \dots y_{n-1}$
- Observar que é possível fatorar termo repetido:

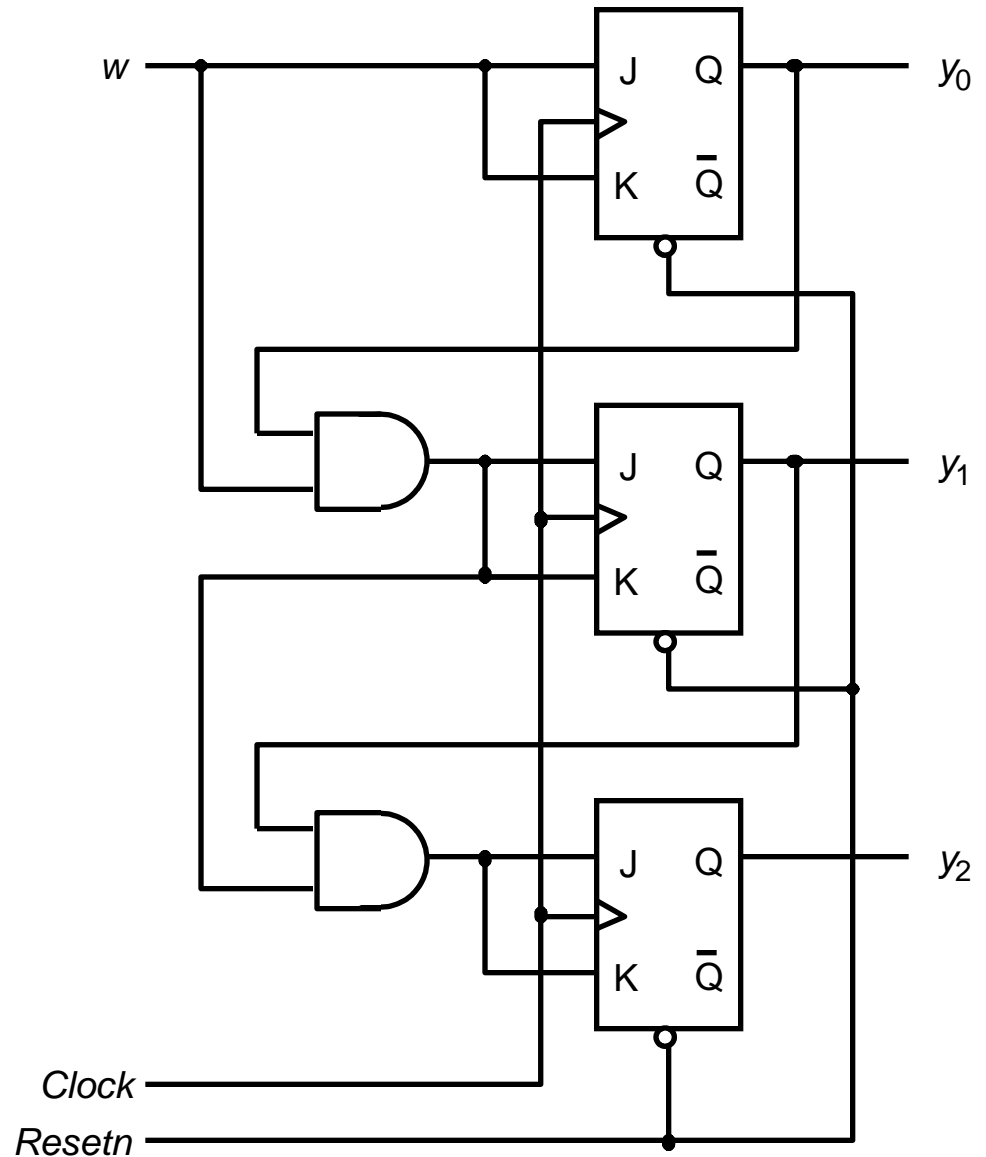
$$J_i = K_i = J_{n-1} \cdot y_{n-1}$$

- Essa é a forma conhecida com FF - T



# Contador com JK e termos fatorados

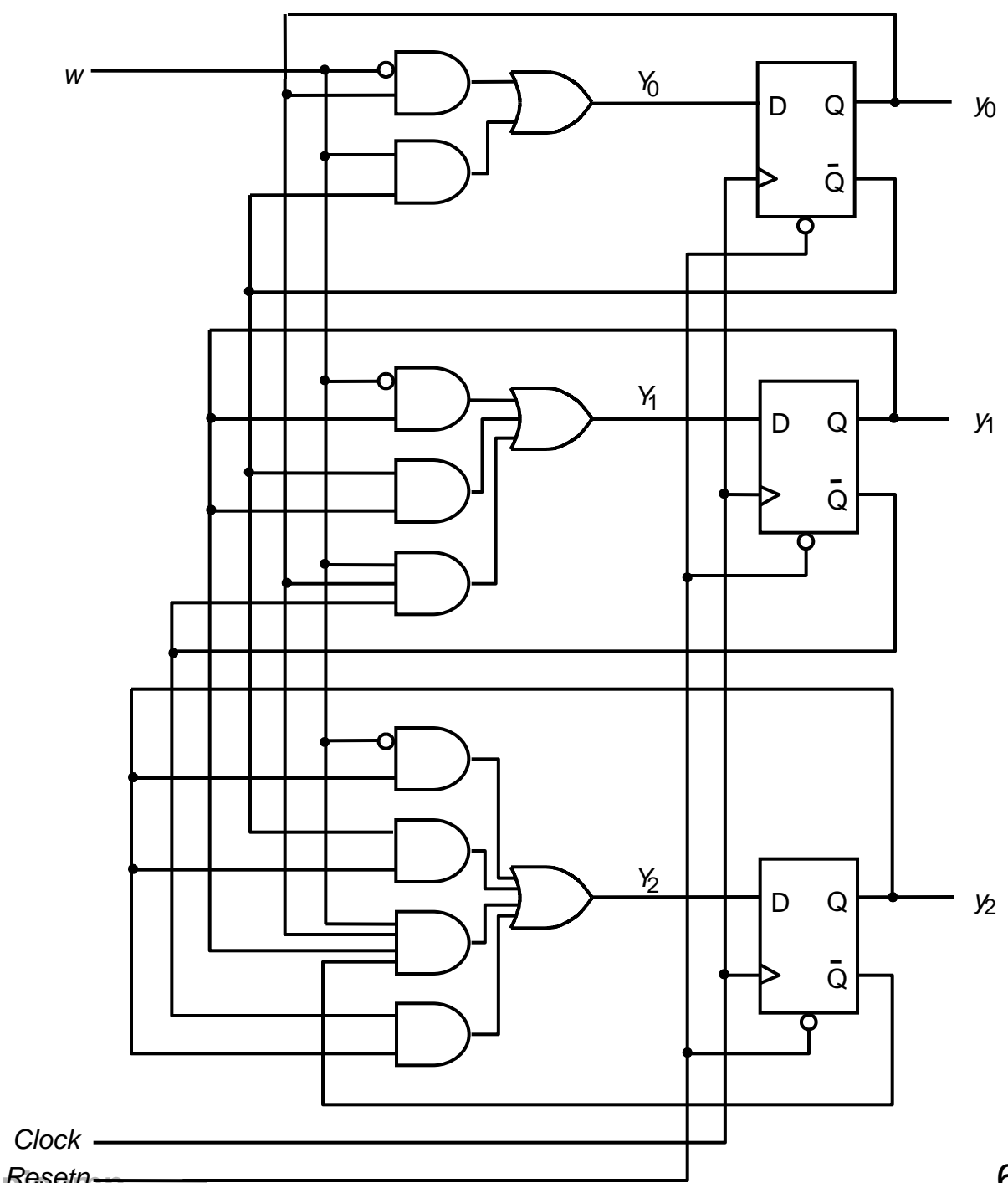
- Exatamente o contador com FFT
- Comparar com contador FFD





# Contador com FFD

- Em geral, implementação com FFD tem maior custo



## 8.6 Minimização de estados

- Definições:
  - Estados  $S_i$  e  $S_j$  são equivalentes: para qualquer sequencia de entradas, as saídas serão as mesmas se  $S_i$  ou  $S_j$  for o estado inicial
  - 1-sucessor de  $S_v$ : Estado que se alcança aplicando-se a entrada 1 a partir do estado  $S_v$
  - Partição de um ou mais blocos: estados em blocos diferentes SÃO não-equivalentes

# Exemplo de minimização

- Partição inicial:  $P_1 = (ABCDEFGG)$
- Saídas diferentes:  
 $P_2 = (ABD)(CEFG)$
- Ver sucessores (0 e 1) dos estados em cada bloco
  - (ABD) 0-sucessores: (BDB) mesmo bloco em  $P_2 \rightarrow ok$
  - (ABD) 1-sucessores: (CFG) mesmo bloco em  $P_2 \rightarrow ok$
  - (CEFG) 0-sucessores: (FFEF) mesmo bloco em  $P_2 \rightarrow ok$
  - (CEFG) 1-sucessores: (ECDG) dif. bloco em  $P_2 \rightarrow não ok$ 
    - Partição (CEG) $\rightarrow$ (ECG) e (F) $\rightarrow$ (D) bloco diferente
- $P_3 = (ABD)(CEG)(F)$  ..... Repetindo
- $P_4 = (AD)(B)(CEG)(F)$  .....fim

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

# Exemplo de minimização(2)

- Original  
(ABCDEFGG)

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

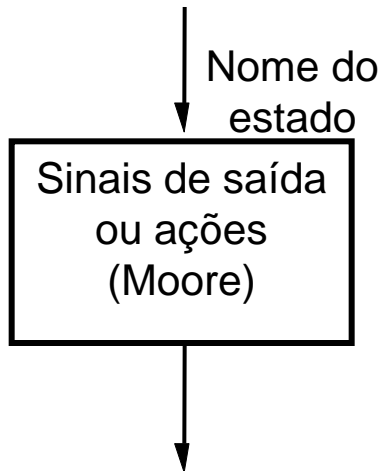
- Minimizado  
(AD)(B)(CEG)(F)

Present state	Nextstate		Output z
	w = 0	w = 1	
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

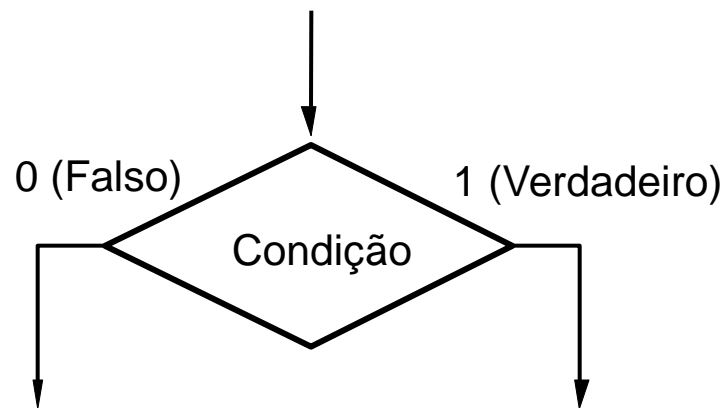
# ASM – Algorithmic State Machine

- Representação alternativa (e equivalente) ao diagrama de estados
  - Semelhante ao fluxograma
  - Elementos principais:

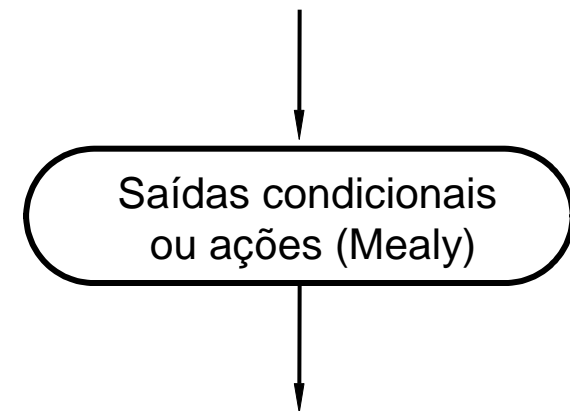
a) Caixa de estado



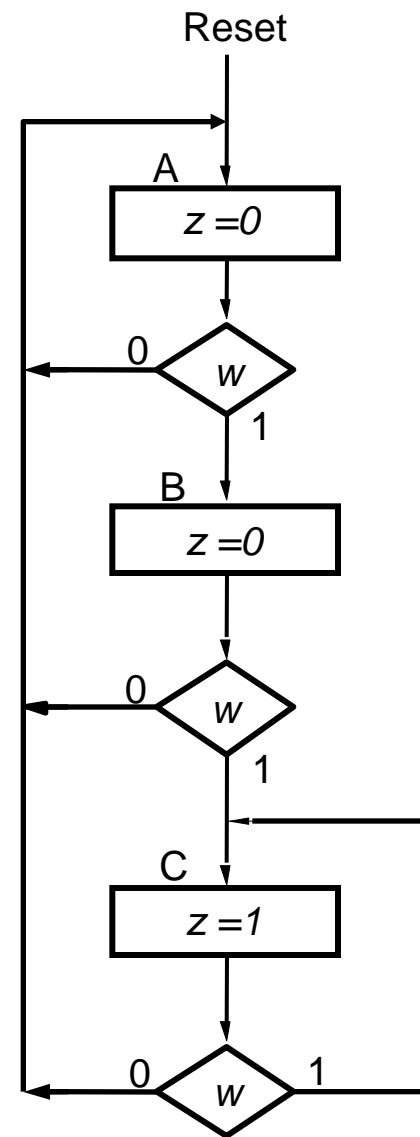
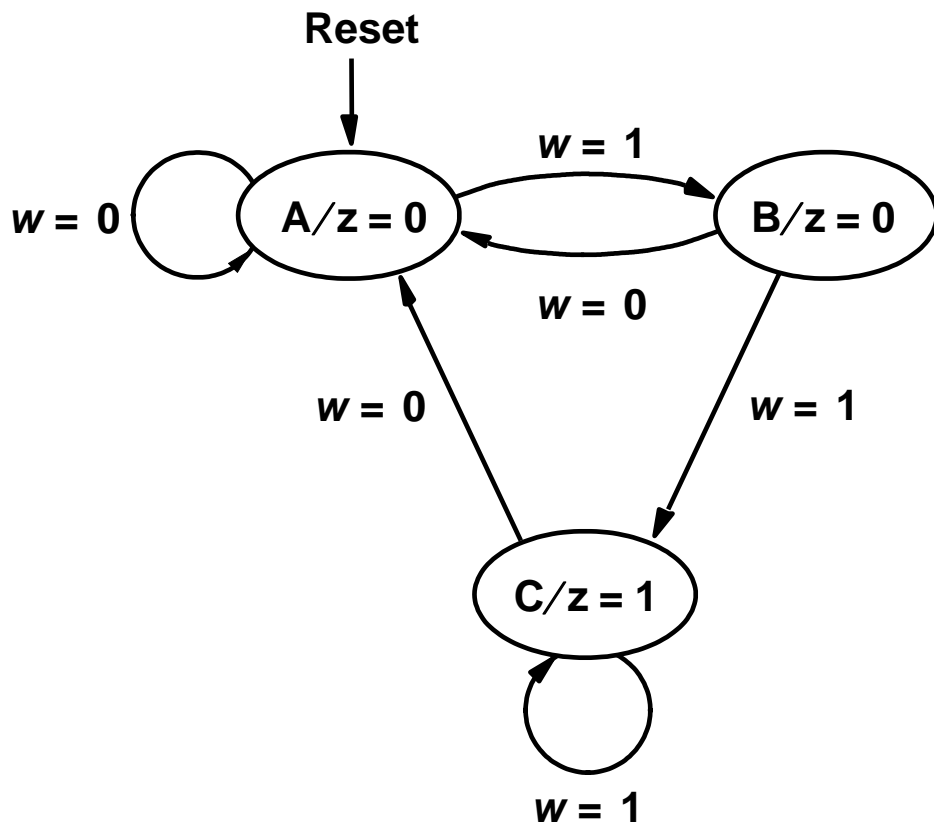
b) Decisão



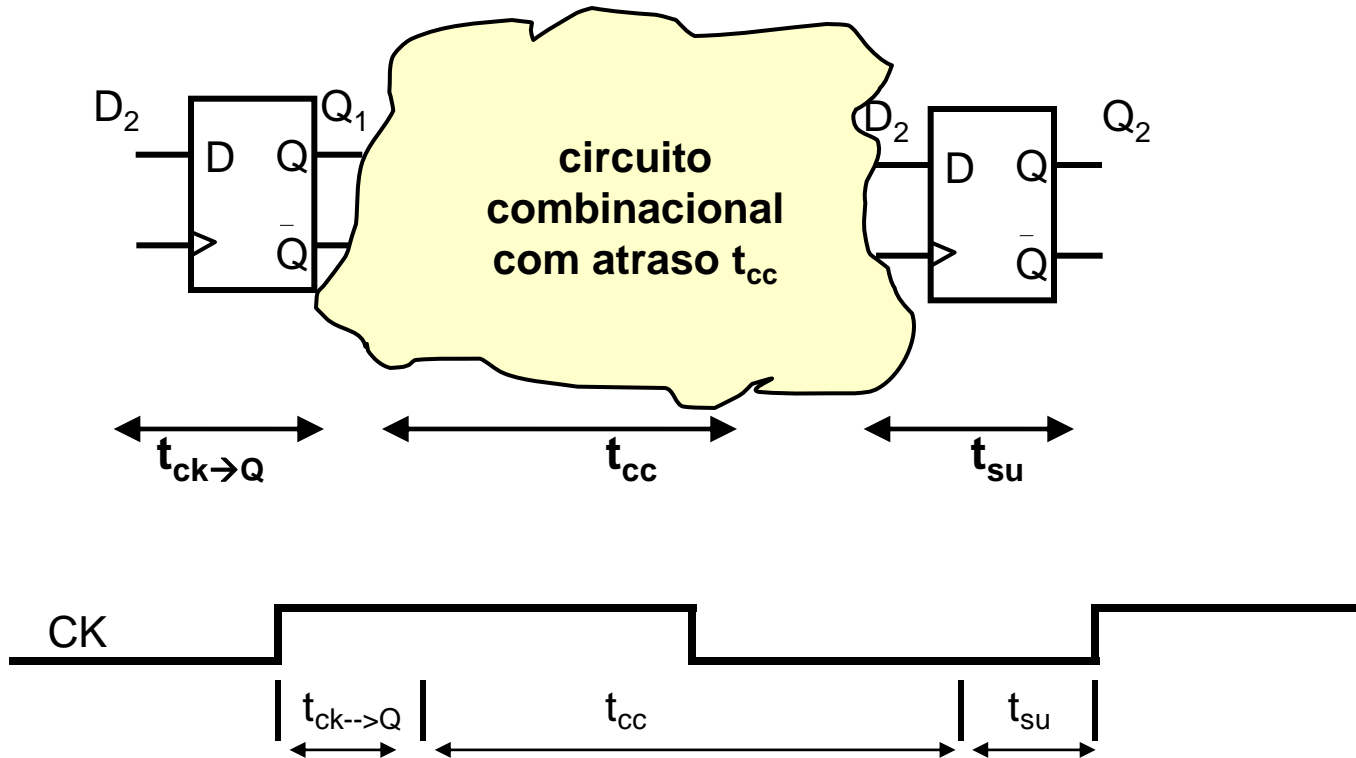
c) Saída condicional



# Exemplo 8.3: FSM e ASM



# Visão geral de máquinas síncronas



- Em circuitos síncronos, frequência máxima do clock limitada pelo maior atraso combinacional
- $T_{ck} \geq t_{ck \rightarrow Q} + t_{cc} + t_{su}$
- $f_{ck} \leq 1 / (t_{ck \rightarrow Q} + t_{cc} + t_{su})$