Circuitos Seqüenciais

Circuitos Lógicos

DCC-IM/UFRJ

Prof. Gabriel P. Silva

Circuitos Seqüenciais

- Um circuito seqüencial síncrono consiste de um circuito combinacional e uma rede de memória formada por elementos de armazenamento (usualmente flip-flops);
- A rede de memória define o estado atual da máquina de estados, representado pelo conjunto de flip-flops;
- O circuito seqüencial difere de um circuito combinacional puro na medida em que o próximo estado será definido não só a partir das entradas atuais, como também do estado atual.
- Logo, as possibilidades de projeto aumentam enormente com uso de circuitos seqüenciais.

Circuitos Seqüenciais

- O funcionamento dos circuitos seqüenciais pode ser representado por uma máquina de estado.
- O conjunto dos valores armazenados em cada flip-flop define o estado atual dessa máquina de estado.
- Há diversas codificações possíveis para a representação do estado atual com o conjunto de flip-flops, com por exemplo:
  - Binária
  - Código Gray
  - One Hot
- Veremos mais detalhes de cada uma dessas possibilidades mais adiante.

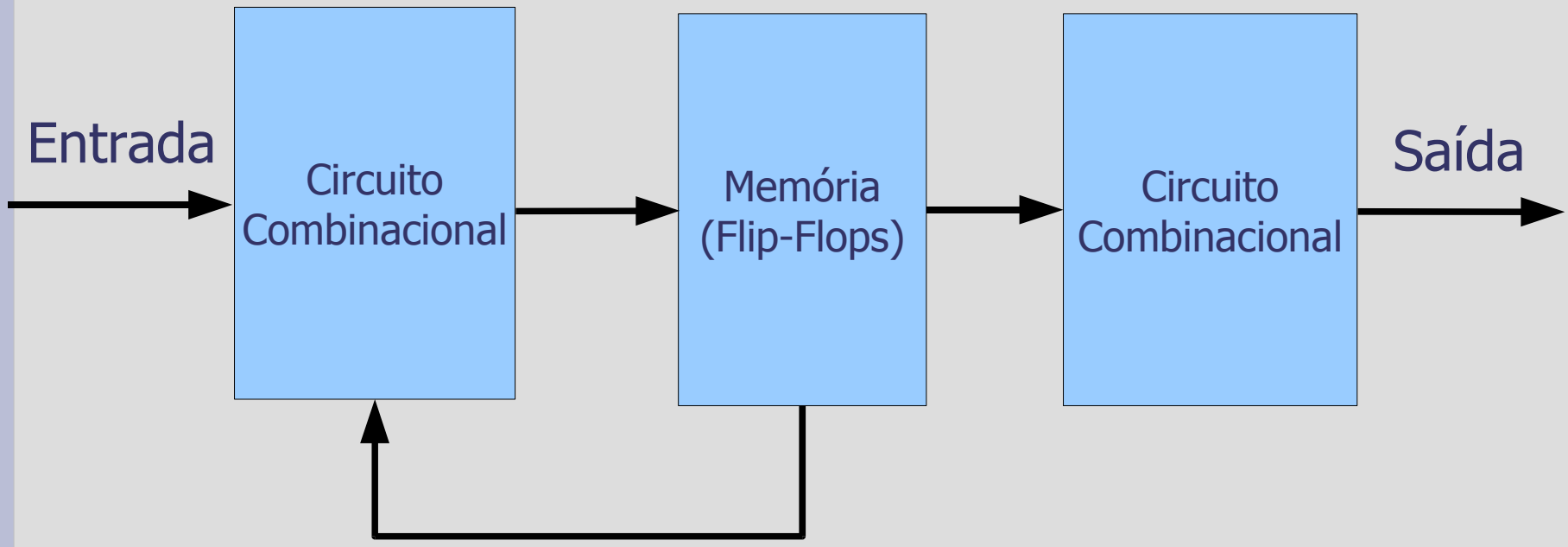
Circuitos Seqüenciais

• A máquina de estados representada pelo circuito seqüencial síncrono pode ser implementada de duas maneiras equivalentes entre si:

- Máquina de Moore
- Máquina de Mealy

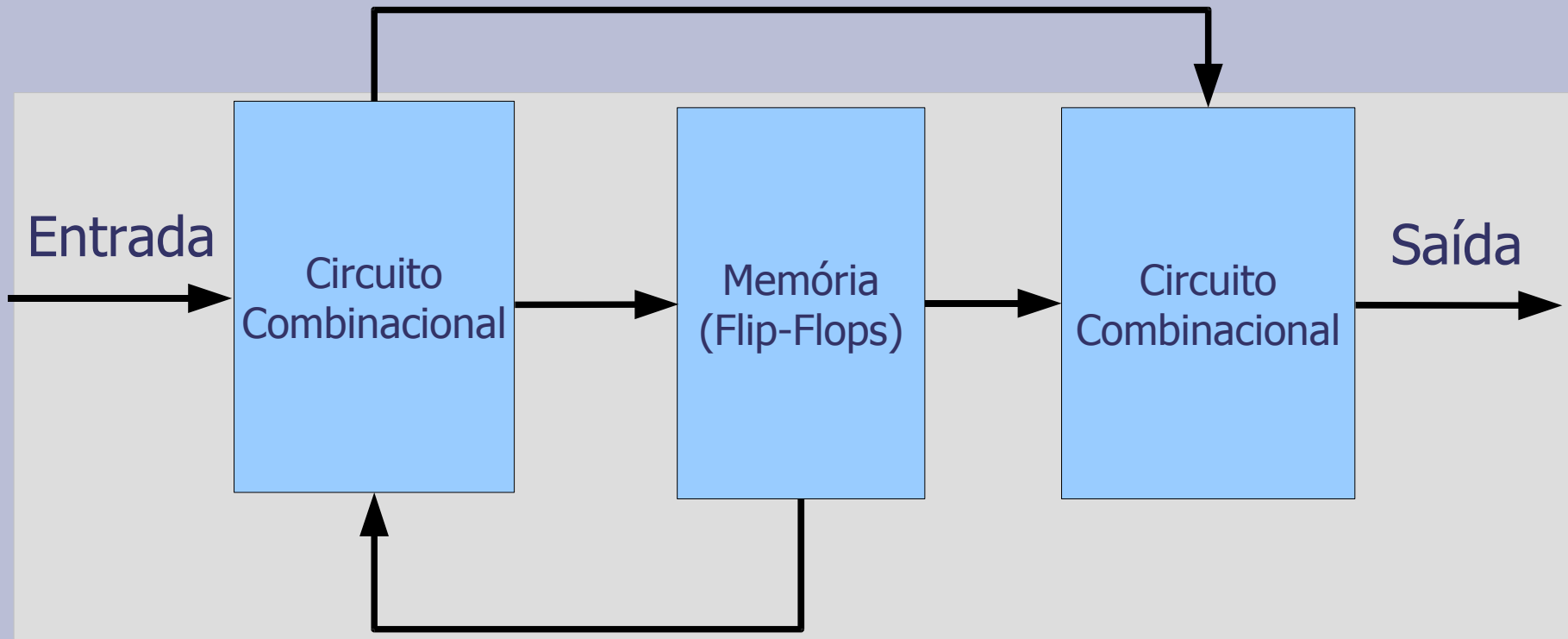
- Nos slides a seguir observamos diagramas representativos dessas implementações

Máquina de Moore



Na máquina de Moore a saída Y muda apenas na transição do relógio.

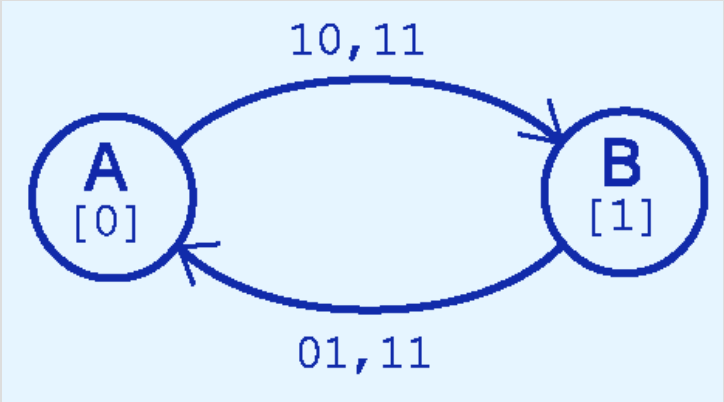
Máquina de Mealy



Na máquina de Mealy a saída Y pode mudar em qualquer instante, em função da entrada X.

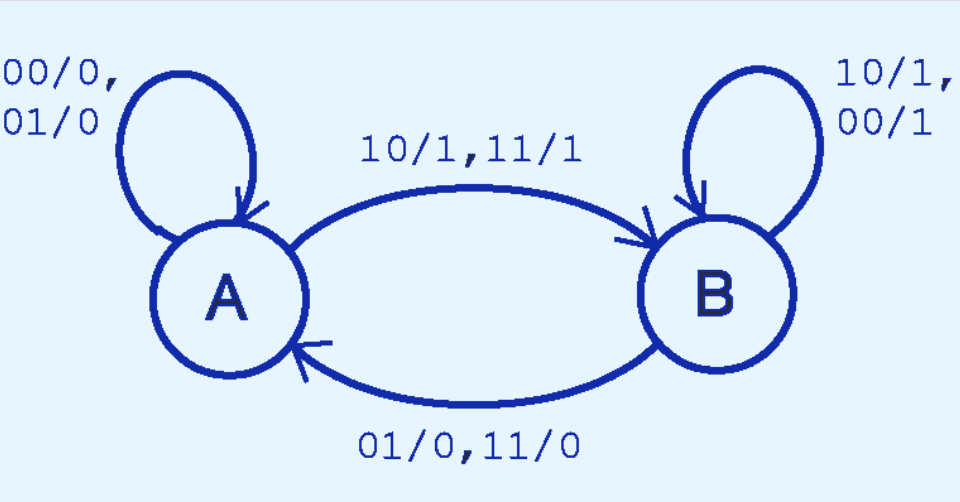
Máquina de Moore

- O comportamento das máquinas de Moore e Mealy machines é idêntico, mas suas implementações diferem.
- Na representação da máquina de Moore, nos arcos do grafo somente são representados os sinais de entrada causadores da transição de um estado para outro.



Máquina de Mealy

- Na representação da máquina de Mealy, nos arcos do grafo são representados os sinais de entrada causadores da transição de um estado para outro, com os respectivos valores para a saída.



Mealy x Moore

- As representações podem ser transformadas de uma para a outra.
- Vantagens e desvantagens:
  - Mealy:
    - glitches (ruídos)
    - problemas de amostragem
    - + fácil de projetar
    - menor número total de estados
  - Moore:

    + sem glitches

Síntese de Redes Seqüenciais

- A síntese de redes seqüenciais pode ser obtida seguindo-se uma série bem determinada de passos:
  - O primeiro passo consiste em elaborar um diagrama de estados, que seja uma interpretação fiel do problema original;
  - Opcionalmente pode-se minimizar o número de estados no diagrama de estados;
  - Escrever a tabela de estados, com os estados atuais, próximos estados e saídas.
  - Atribuir a cada estado uma combinação de variáveis de estado (flip-flops);
  - Contruir a tabela de excitação do tipo de flip-flop utilizado;

Síntese de Redes Seqüenciais

- A síntese de redes seqüencias pode ser obtida seguindo-se uma série bem determinada de passos:
  - Montar o mapa de Karnaugh para cada uma das entradas dos flip-flops do circuito, com o auxílo da tabela de excitação;
  - Obter a equação final de cada entrada de cada um dos flip-flops do circuito a partir da simplificação do mapa de Karnaugh;
  - Fazer o mesmo procedimento para as equações das variáveis de saída.
  - Finalmente, elaboração do diagrama lógico do circuito, lembrando que todos os elementos de memória (flip-flops) recebem o mesmo sinal de relógio.

Tabela de Transição

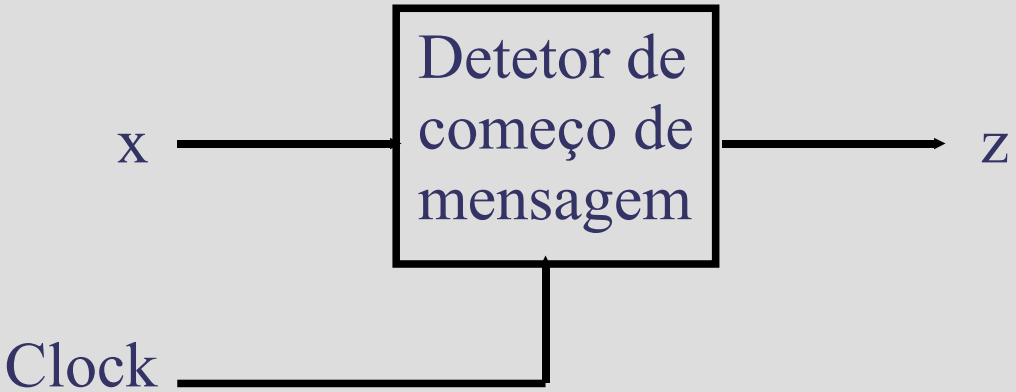
Estado Atual		Próximo Estado				Função de Saída		
Vetor de Entrada		X_1	X_2	X_3		X_1	X_2	X_3
A		C	B	A		Z_1	Z_2	-
B		A	C	B		Z_1	Z_1Z_2	-
C		D	B	D		Z_1	Z_1	Z_2
D		C	D	A		Z_2	Z_2	Z_1

Ex. Um estado atual C e um vetor de entrada atual X_2 produzem uma variável de saída Z_1 e próximo estado B.

Síntese de Redes Seqüenciais

- Exemplo
  - O começo de uma mensagem em um sistema de comunicação é indicado pela ocorrência de três ‘1’ consecutivos em uma linha *x*. Projete um circuito que forneça em sua saída o valor ‘1’ apenas durante o período de relógio coincidente com o terceiro ‘1’ consecutivo na linha *x*. Supor que um mecanismo externo inicialize o detetor, após o término da mensagem, no estado de “reset”.

Circuito Detector de Seqüência Inicial



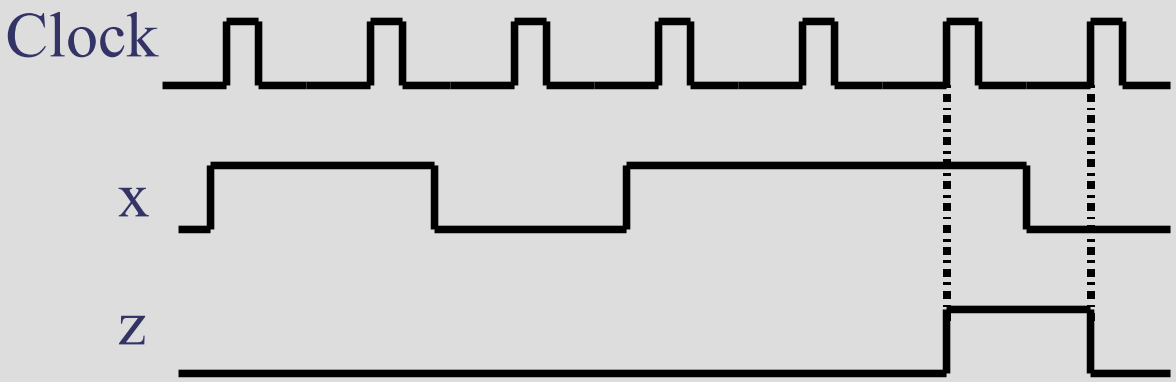


Diagrama de Estados

Diagrama de Estados

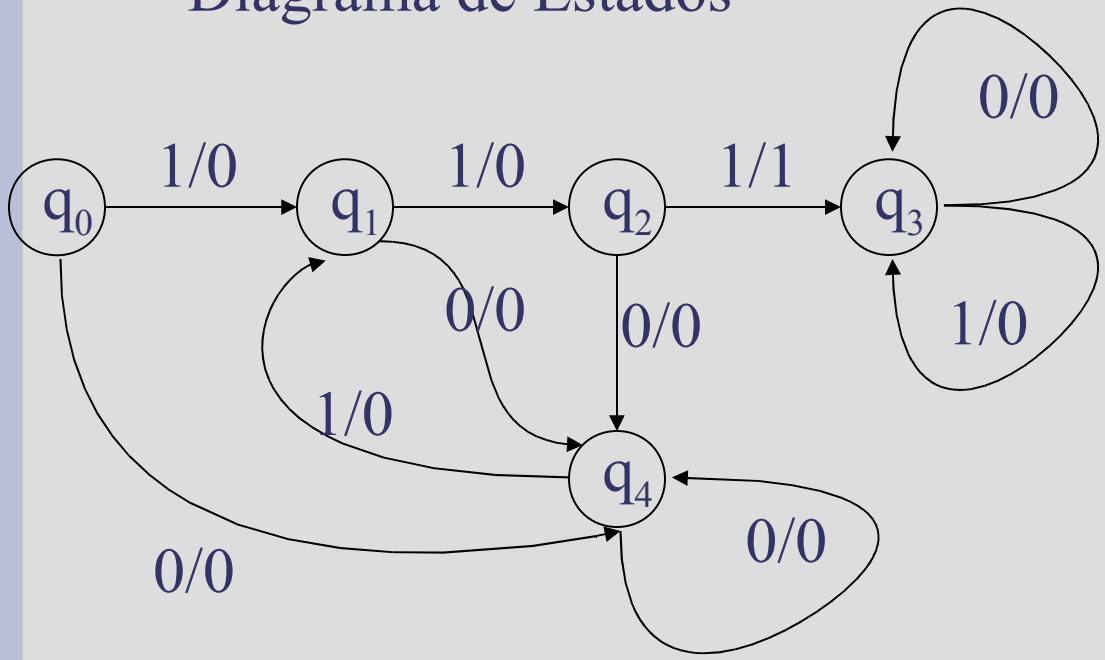


Tabela de Estados

q^t \ x^t	0	1
q_0	q_4,0	q_1,0
q_1	q_4,0	q_2,0
q_2	q_4,0	q_3,1
q_3	q_3,0	q_3,0
q_4	q_4,0	q_1,0

q^{t+1}, z^t

Diagrama de Estados Mínimo

Diagrama de Estados

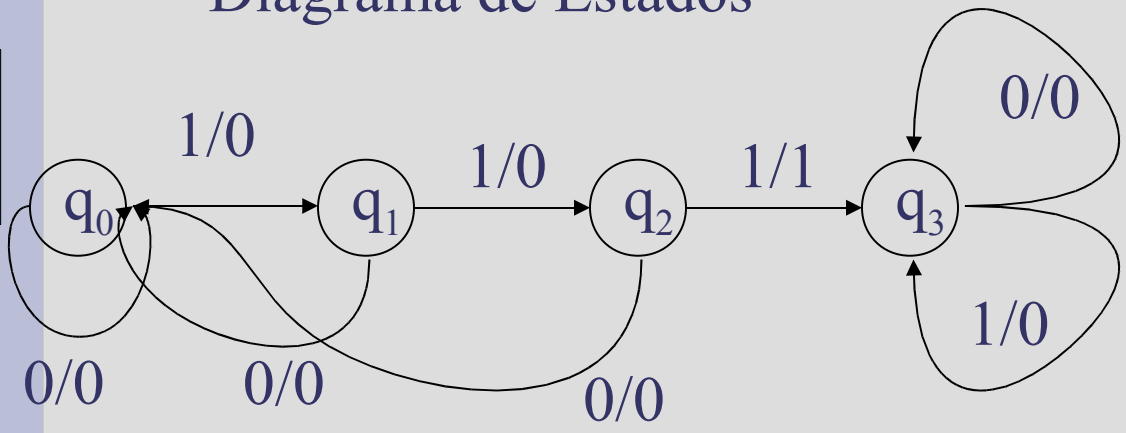


Tabela de Estados MÍNIMA

q^t \ x^t	0	1
q_0	q_0,0	q_1,0
q_1	q_0,0	q_2,0
q_2	q_0,0	q_3,1
q_3	q_3,0	q_3,0

q^{t+1}, z^t

Variáveis para os Estados

- **Uma** variável binária é necessária para codificar dois estados
- **Duas** variáveis binárias são necessárias para codificar três ou quatro estados.
- **Três** variáveis binárias são necessárias para codificar de cinco a oito estados.
- **Quatro** variáveis binárias são necessárias para codificar de nove a dezesseis estados.
- **Cinco** variáveis binárias são necessárias para codificar de dezessete a trinta e dois estados.
- Neste exemplo de detetor de sequências

  $\Rightarrow$São necessárias 2 variáveis para representar 4 estados.

Variáveis para os Estados

- Eventualmente pode-se utilizar a codificação “one hot”, onde a cada estado é associado um flip-flop diferente.
- Assim sendo, por exemplo, se precisarmos representar cinco estados diferentes utilizando uma codificação “one hot”, poderíamos usar cinco flip-flops distintos com a codificação a seguir:
  - 0 0 0 0 1
  - 0 0 0 1 0
  - 0 0 1 0 0
  - 0 1 0 0 0
  - 1 0 0 0 0

Designação dos Estados

Designação de estados:

Tabela de Transição:

q^t	$(y_1y_0)^t$	x^t = 0	x^t = 1	x^t = 0	x^t = 1
		$(y_1y_0)^{t+1}$		z^t	
q_0	0 0	0 0	1 1	0	0
q_1	1 1	0 0	0 1	0	0
q_2	0 1	0 0	1 0	0	1
q_3	1 0	1 0	1 0	0	0

Designação dos Estados

- Os nomes dos estados (letras) são substituídos na tabela de estados pela codificação escolhida para eles.
- Escolhe-se um determinado tipo de “flip-flop” para representar a codificação escolhida.
- A tabela de estados mostra, a partir do estado atual, qual a codificação para o próximo estado que desejamos alcançar.
- Precisamos montar um “mapa de karnaugh” para determinar as equações de entrada de cada flip-flop do circuito.

Designação dos Estados

- Como queremos saber o “próximo estado”, devemos utilizar a “tabela de excitação” relativa ao flip-flop escolhido para determinar essas esquações.
- A tabela de excitação diz quais são os valores que deve haver na entrada do flip-flop para que um determinado valor seja obtido na sua saída.

Tabela de Excitação
FF JK

Tabela de Excitação para o flip-flop JK

Q^t	$\rightarrow$	Q^{t+1}	J^t	K^t
0	$\rightarrow$	0	0	X
0	$\rightarrow$	1	1	X
1	$\rightarrow$	0	X	1
1	$\rightarrow$	1	X	0

J^t	K^t	Q^{t+1}
0	0	Q^t
0	1	0
1	0	1
1	1	$(Q^t)'$

Mapa de Karnaugh
Flip-Flop 1

q^t	$(y_1y_0)^t$	x^t = 0	x^t = 1	x^t = 0	x^t = 1
q_0	0 0	0 0	1 1	0	0
q_1	1 1	0 0	0 1	0	0
q_2	0 1	0 0	1 0	0	1
q_3	1 0	1 0	1 0	0	0
		$(y_1y_0)^{t+1}$		z^t	

$Q^t \rightarrow Q^{t+1}$	J^t	K^t
0 → 0	0	X
0 → 1	1	X
1 → 0	X	1
1 → 1	X	0

$y_1^t y_0^t$ / x^t

x^t \ $y_1^t y_0^t$	00	01	11	10
0	0	0	X	X
1	1	1	X	X

$$J_{y_1} = x$$

$y_1^t y_0^t$ / x^t

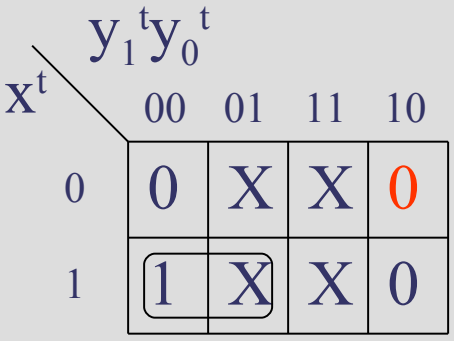
x^t \ $y_1^t y_0^t$	00	01	11	10
0	X	X	1	0
1	X	X	1	0

$$K_{y_1} = y_0$$

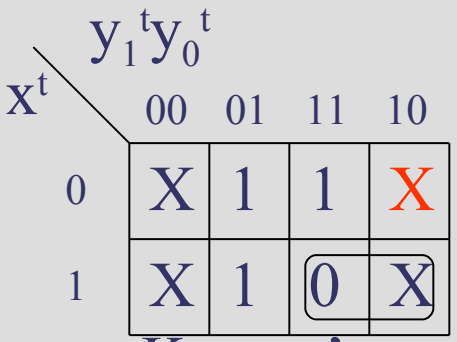
Mapa de Karnaugh
Flip-Flop 0

q^t	$(y_1y_0)^t$	x^t = 0	x^t = 1	x^t = 0	x^t = 1
q_0	0 0	0 0	1 1	0	0
q_1	1 1	0 0	0 1	0	0
q_2	0 1	0 0	1 0	0	1
q_3	1 0	1 0	1 0	0	0
		$(y_1y_0)^{t+1}$		z^t	

$Q^t \rightarrow Q^{t+1}$	J^t	K^t
0 → 0	0	X
0 → 1	1	X
1 → 0	X	1
1 → 1	X	0



$J_{y_0} = x \,.\, y_1'$



$K_{y_0} = x' + y_1'$

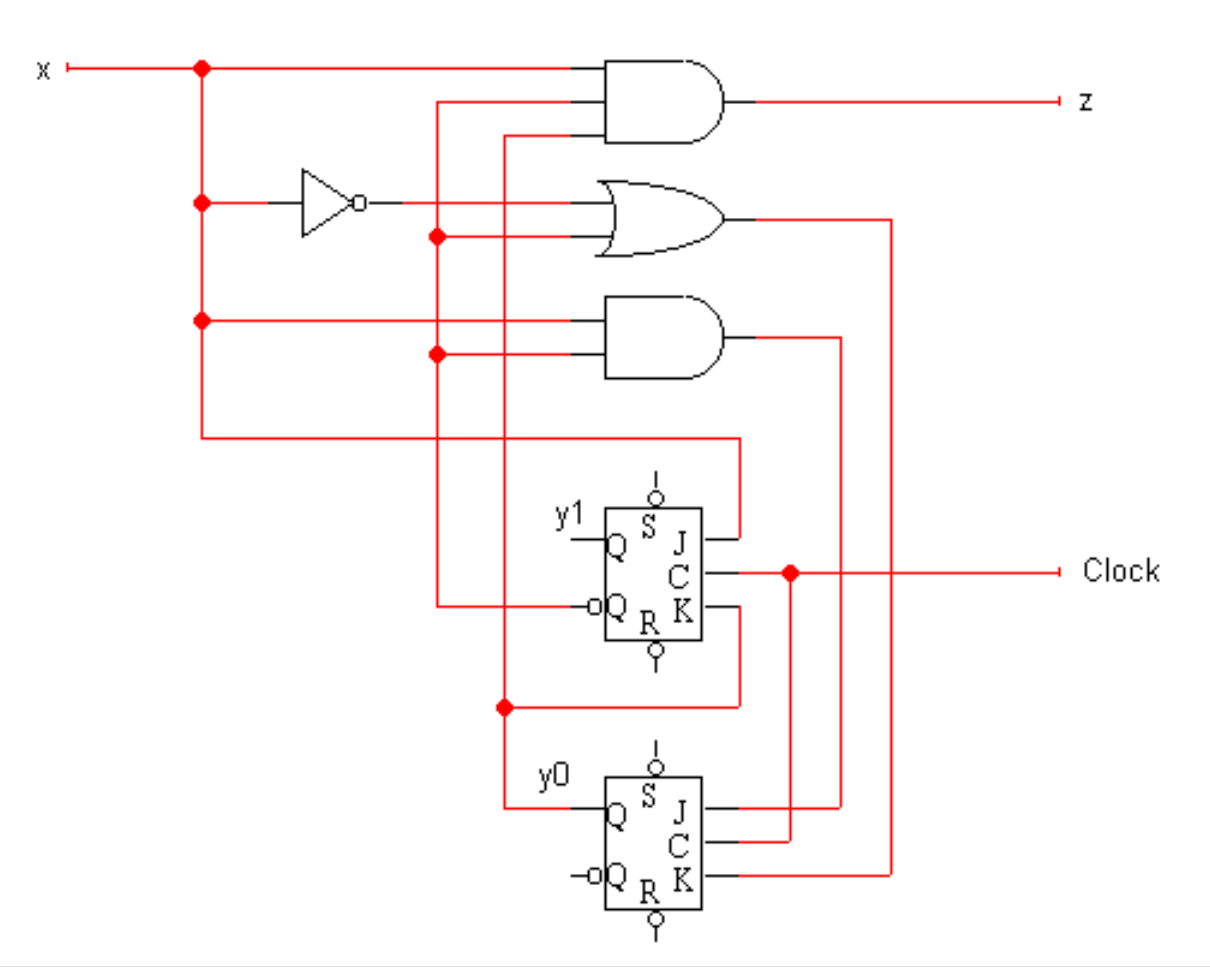
Mapa de Karnaugh
Saída Z

		x^t		x^t	
q^t	$(y_1y_0)^t$	0	1	0	1
q_0	0 0	0 0	1 1	0	0
q_1	1 1	0 0	0 1	0	0
q_2	0 1	0 0	1 0	0	1
q_3	1 0	1 0	1 0	0	0
			$(y_1y_0)^{t+1}$		z^t

x^t \ $y_1^t y_0^t$	00	01	11	10
0	0	0	0	0
1	0	1	0	0

$$z = x \,.y_1\text{'}.y_0$$

Síntese de Redes Seqüenciais
Circuito Final



Resumo

- O projeto de redes seqüenciais requer atenção e experiência.
  - A maioria das aplicações úteis requer o uso de tais sistemas.
  - Há necessidade de um cuidado especial com as especificações iniciais do circuito, principalmente na codificação dos estados e na escolha do tipo de flip-flop a ser utilizado.

Exercício

- Projete um sistema com uma entrada e duas saídas que detecte a ocorrência de duas seqüências de 4 bits.

  - O valor inicial da saída é 00.
  - Se a seqüência 1010 for detectada $\Rightarrow$ a saída 01 é gerada.
  - Se a seqüência 0111 for detectada $\Rightarrow$ a saída 10 é gerada.

  - A saídas devem ser geradas quando o último bit da seqüência de entrada for detectado e o detetor deve retornar para o estado inicial.
  - Nenhuma sobreposição das seqüências deve ser assumida.

Exercício

- Projete um sistema com uma entrada e duas saídas que detecte a ocorrência de duas seqüências de 4 bits.

  - O valor inicial da saída é 00.
  - Se a seqüência 1010 for detectada $\Rightarrow$ a saída 01 é gerada.
  - Se a seqüência 0111 for detectada $\Rightarrow$ a saída 10 é gerada.

  - A saídas devem ser geradas quando o último bit da seqüência de entrada for detectado e o detetor deve retornar para o estado inicial.
  - Nenhuma sobreposição das seqüências deve ser assumida.

Máquina de Estados em VHDL

```
ENTITY simple_seq IS
  PORT(x    : IN  BIT ;
       z    : OUT Bit_Vector(1 DOWNTO 0);
       clk  : IN  BIT);
END simple_seq;

ARCHITECTURE behavioral OF simple_seq IS
  TYPE   stateT is (S0, S1, S2, S3);
  SIGNAL state: stateT;          -- state
```

Máquina de Estados em VHDL

```
BEGIN
 PROCESS (clk)                  -- processo ativado por clk
 BEGIN
  IF (clk'EVENT AND clk='1') THEN
   CASE state IS                 -- determina o novo estado
    WHEN S0 => IF (x = '0') THEN state <= S0;
               ELSE             state <= S1;
               END IF;
    WHEN S1 => IF (x = '0') THEN state <= S2;
               ELSE             state <= S3;
               END IF;
    WHEN S2 => IF (x = '0') THEN state <= S0;
               ELSE             state <= S1;
               END IF;
    WHEN S3 => IF (x = '0') THEN state <= S2;
               ELSE             state <= S3;
               END IF;
   END CASE;
  END IF;
 END PROCESS;
```

Máquina de Estados em VHDL

```
PROCESS (state,x)        -- processo ativado por state ou x
BEGIN
  CASE state IS
    WHEN S0 =>
                          z <= "00";
    WHEN S1 =>
                         z <= x & NOT(x);
    WHEN S2 =>
                         z <= NOT(x) & x;
    WHEN S3 =>
                         z <= "11";
  END CASE;
 END PROCESS;
END behavioral;
```