Design de Computadores

Aula 6 – Extras 1

Insper

Máquinas de Estados Finitos

- Tópicos:
  - Máquinas de estados finitos (MEFs);
  - Conceito de “estado”;
  - Máquina de Moore;
  - Máquina de Mealy;
  - Comparativo Mealy versus Moore.

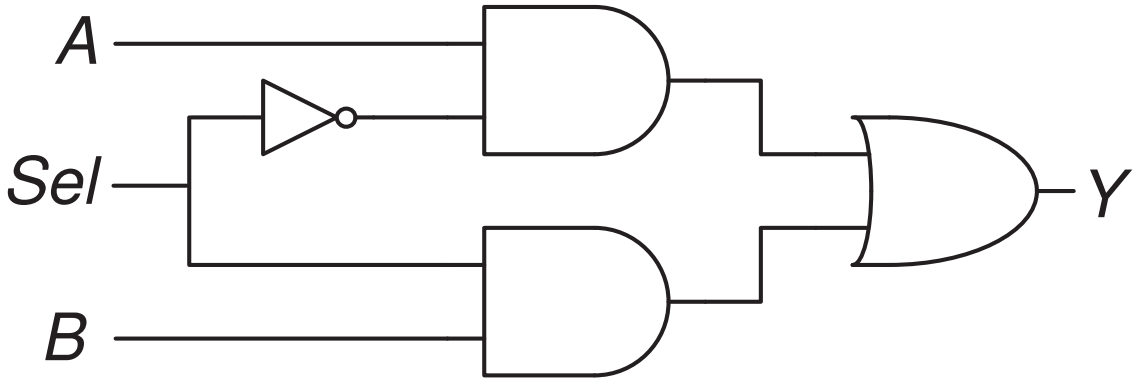
- Objetivos de aprendizado:
  - Entender o funcionamento das MEFs;
  - Análise e aplicação de MEFs na criação de circuitos sequenciais.

- Os circuitos digitais podem ser divididos em:
  - Circuitos combinacionais;
  - Circuitos sequenciais.

- Porém, é rara a aplicação que seja:
  - Puramente sequencial;
  - Ou puramente combinacional.

- Podemos caracterizar a grande maioria dos circuitos digitais como:
  - Possuindo uma parte sequencial;
  - E outra combinacional.

- Na lógica combinacional:
  - A saída depende somente do valor atual das suas entradas;
  - Então, possuindo-se somente os valores de entrada, é possível determinar o valor de saída.



- O seu funcionamento:
  - É completamente descrito pela sua tabela da verdade.

- O circuito sequencial:
  - Possui memória interna:
    - Na forma de um ou mais flip-flops (ou registradores).
  - É sincronizado por um sinal de clock.
  - E pode possuir alguma lógica combinacional.

- O conteúdo da memória interna:
  - Define o estado interno do sistema;
  - Para um determinado momento.

- O funcionamento do circuito:
  - É a evolução, no tempo, desses estados internos:
    - Que influenciam o valor de saída.

- Essa evolução, ou sequencia, de estados internos:
  - É definida pela função de próximo estado (transição):
    - Que consideram os valores de entrada;
    - E o estado interno.
- A saída é definida pela função de saída.
- Ela define o tipo de máquina de estados:
  - Máquina de estados finitos de Moore:
    - A saída é uma função combinacional do estado.
    - Possui um subtipo em que a saída é o próprio estado:
      - Máquina de estados finitos de Medvedev.
  - Máquina de estados finitos de Mealy:
    - A saída é uma função combinacional do estado e de um ou mais sinais de entrada.

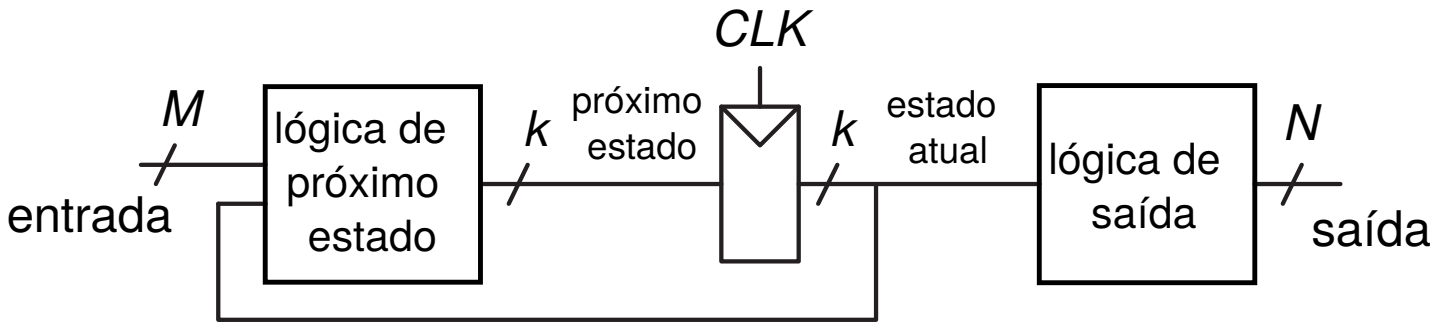
- O funcionamento desses circuitos:
  - Não podem ser descritos:
    - Somente com uma tabela da verdade;
    - Dado que a saída é função do estado atual e do valor de entrada.
- Devemos usar um conjunto de tabelas que especificam:
  - O estado atual;
  - Os valores da entrada;
  - O próximo estado resultante;
  - Para cada combinação de entrada e estado atual.
- Ou um diagrama de estados:
  - Que é um grafo cíclico.

- Alguns conceitos:
  - Estado: é a situação interna do sistema que aguarda para executar uma transição;
  - Transição: são as ações a serem executadas quando uma condição é preenchida ou um evento é recebido;
  - Ação de Entrada: é a ação executada quando se entra no estado;
  - Ação de Saída: é a ação executada quando se sai do estado;
  - Estado inicial: é o estado em que a máquina inicia, normalmente representado com uma seta.

Máquina de Moore

- Nesta máquina, a saída (y):
  - Depende do estado atual interno: s(t);
  - E de uma transformação H (lógica de saída).
- Podemos dizer que:

$$y(t) = H(s(t))$$



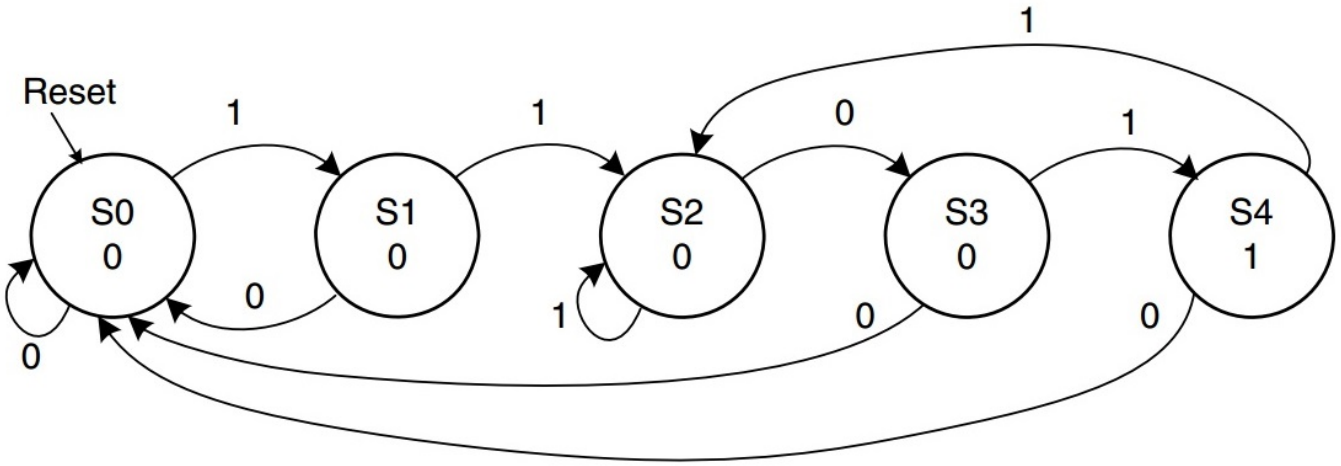
Máquina de estados do tipo: Moore

Insper

- A máquina de Moore utiliza somente ações de entrada, ou seja, quando se entra no estado:
  - Dessa forma, a sua saída depende somente do estado atual;
  - Simplificando o comportamento da máquina;
  - Simplificando o seu projeto;
  - E evitando glitchs na saída, pois todos os sinais estão sincronizados pelo Clock.

- Porém, em geral, uma máquina de Moore:
  - Possui mais estados, para um mesmo problema:
    - Que a máquina de Mealy.

- O diagrama de estados da máquina de Moore:
  - Representa os estados (círculos com nome e valor de saída);
  - A transição entre os estados, através dos arcos com flechas;
  - A entrada que gera a transição, marcada no arco;
  - O destino da transição, de acordo com a entrada, indicado pelo círculo (estado e seu valor de saída).
  - O estado inicial (S0) possui a saída 0 após o Reset (seta indicadora).

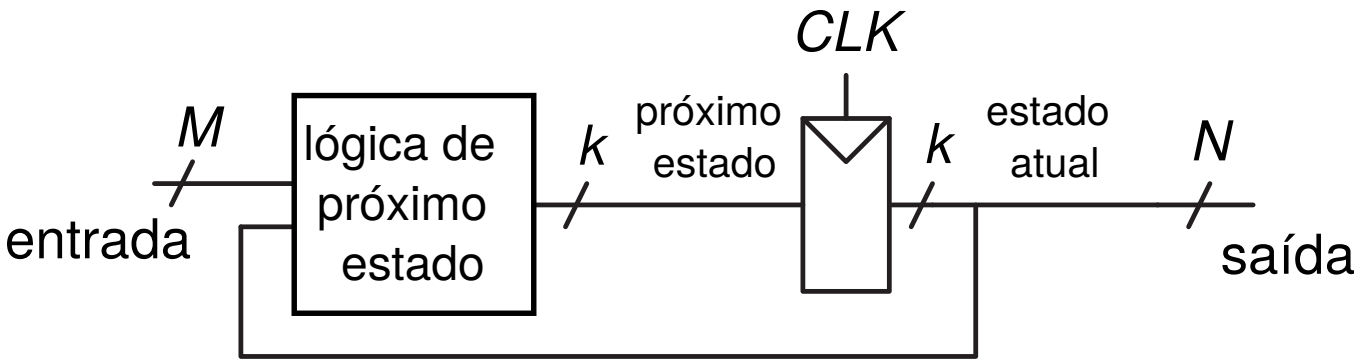


Insper

Moore, subtipo Medvedev (curiosidade)

- Na máquina de Medvedev, a saída (y):
  - Depende somente do estado interno atual: s(t);
  - Não existe a transformação para a saída.
- Podemos dizer que:

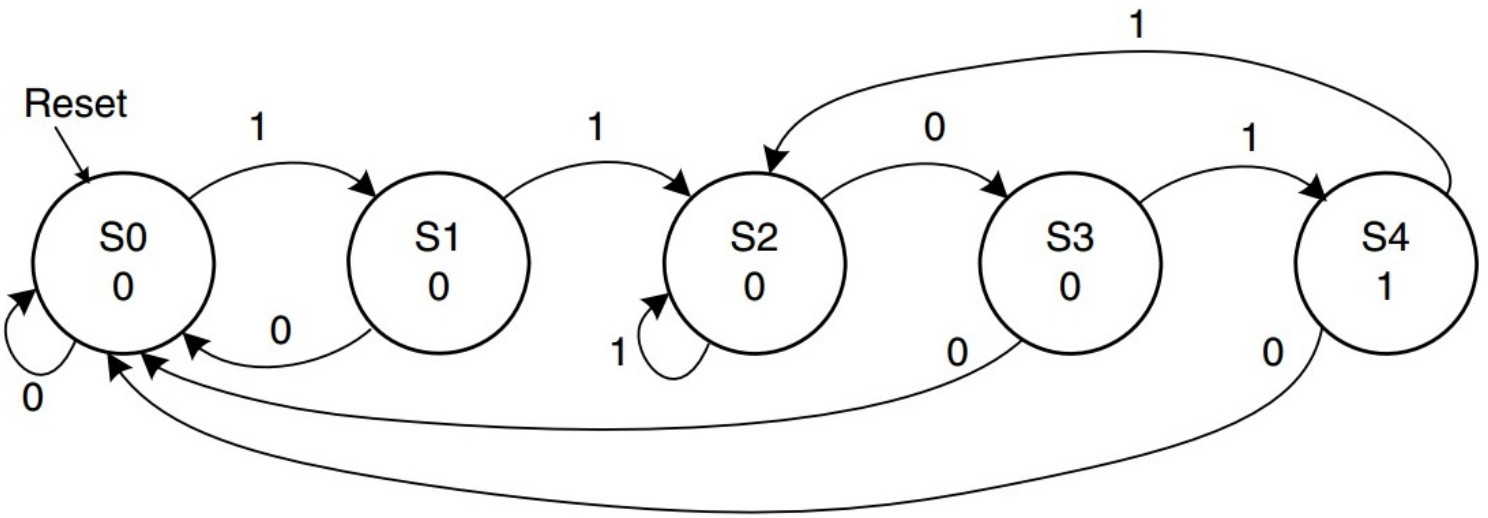
$$y(t) = s(t)$$



Máquina de estados do tipo: Medvedev

- Roteiro para a criação de máquinas de Moore:
  - Identifique as entradas e saídas;
  - Esboce o diagrama de estados;
  - Monte a tabela de transição de estados;
  - Monte a tabela de saída;
  - Escolha a codificação dos estados (binária, código de Gray, one-hot, etc...);
  - Monte as novas tabelas de transição e saída de acordo com a codificação e o Flip Flop escolhido (D, JK, etc...);
  - Monte os mapas de Karnaugh para as novas tabelas e obtenha as equações do circuito:
    - Para o próximo estado;
    - Para a saída.
  - Desenhe o resultado final.

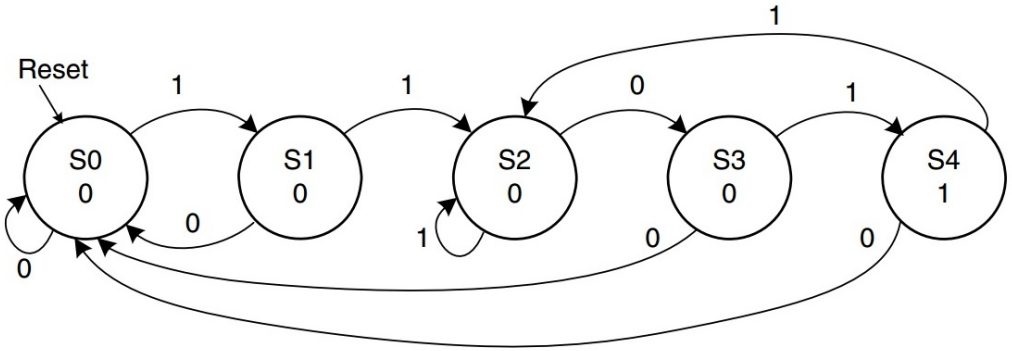
- No exemplo abaixo, temos um detector da sequência binária 1101:
  - Quando recebe a sequência 1101, ativa a saída (com nível 1);
  - Qualquer outra sequência mantém a saída em nível 0.
- São necessários cinco estados (S0 a S4):
  - Um "erro" na sequência pode reiniciar para estados diferentes, dependendo em qual estágio o erro ocorreu:
    - Transição de S1 para S0, S4 para S2, S2 para S2 e S3 para S0.



Insper

- Utilizando o diagrama de estados:
  - Preenche-se a tabela de transição.

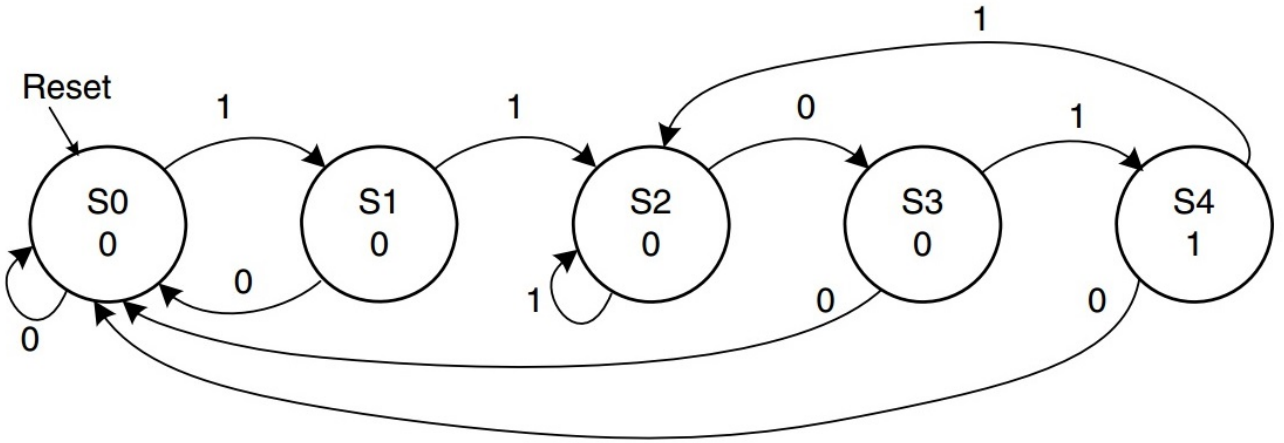
Estado Atual	Entrada	Próximo Estado
S0	0	S0
S0	1	S1
S1	0	S0
S1	1	S2
S2	0	S3
S2	1	S2
S3	0	S0
S3	1	S4
S4	0	S0
S4	1	S2



Insper

- Para a função de saída:
  - A saída terá valor 1 somente em S4:

Estado Atual	Saída: Y
S0	0
S1	0
S2	0
S3	0
S4	1



Insper

- As tabelas de transição e saída utilizando:
  - Codificação binária para o número dos estados;
  - E flip-flops tipo D.

Estado Atual			Entrada	Próximo Estado		
bit 2_{EA}	bit 1_{EA}	bit 0_{EA}		bit 2_{pe}	bit 1_{pe}	bit 0_{pe}
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0

Estado Atual			Saída
bit 2_{EA}	bit 1_{EA}	bit 0_{EA}	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1

- A equação de próximo estado, para cada bit, pode ser feita por Karnaugh ou inspeção direta:

Para $bit2_{pe}=1$, temos:

$bit2_{pe} = bit2'_{EA}*bit1_{EA}*bit0_{EA}*A$

Não existe a condição:

$bit2_{EA}*bit1_{EA}*bit0_{EA}*A$

Minimizamos para:

$bit2_{pe} = bit1_{EA}*bit0_{EA}*A$

Estado Atual			Entrada	Próximo Estado		
bit 2_{EA}	bit 1_{EA}	bit 0_{EA}	A	bit 2_{pe}	bit 1_{pe}	bit 0_{pe}
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0

- Para $bit1_{pe}=1$, temos três condições:
  - $bit1_{pe} = bit2'_{EA} * bit1'_{EA} * bit0_{EA} * A$
  - $bit1_{pe} = bit2'_{EA} * bit1_{EA} * bit0'_{EA}$
  - $bit1_{pe} = bit2_{EA} * bit1'_{EA} * bit'0_{EA} * A$

Minimizamos para:

$$bit1_{pe} = bit1'_{EA} * bit0_{EA} * A + bit1_{EA} * bit0'_{EA} + bit2_{EA} * A$$

Estado Atual			Entrada	Próximo Estado		
$bit2_{EA}$	$bit1_{EA}$	$bit0_{EA}$	A	$bit2_{pe}$	$bit1_{pe}$	$bit0_{pe}$
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0

- Para $bit0_{pe}=1$, temos duas condições:
  - $bit0_{pe} = bit2'_{EA}*bit1'_{EA}*bit0'_{EA}*A$
  - $bit0_{pe} = bit2'_{EA}*bit1_{EA}*bit0'_{EA}*A'$

Minimizamos para:

$bit0_{pe} =$

$bit2'_{EA}*bit1'_{EA}*bit0'_{EA}*A$

$+ bit1_{EA}*bit0'_{EA}*A'$

Estado Atual			Entrada	Próximo Estado		
bit 2_{EA}	bit 1_{EA}	bit 0_{EA}	A	bit 2_{pe}	bit 1_{pe}	bit 0_{pe}
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0

- A lógica de saída é retirada da tabela de saída:

$$Y = bit2_{EA} * bit1'_{EA} * bit0'_{EA}$$

- Como não existe outra ocorrência de $bit2_{EA}$:
  - Minimizamos para: $Y = bit2_{EA}$

Estado Atual			Saída
bit 2_{EA}	bit 1_{EA}	bit 0_{EA}	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1

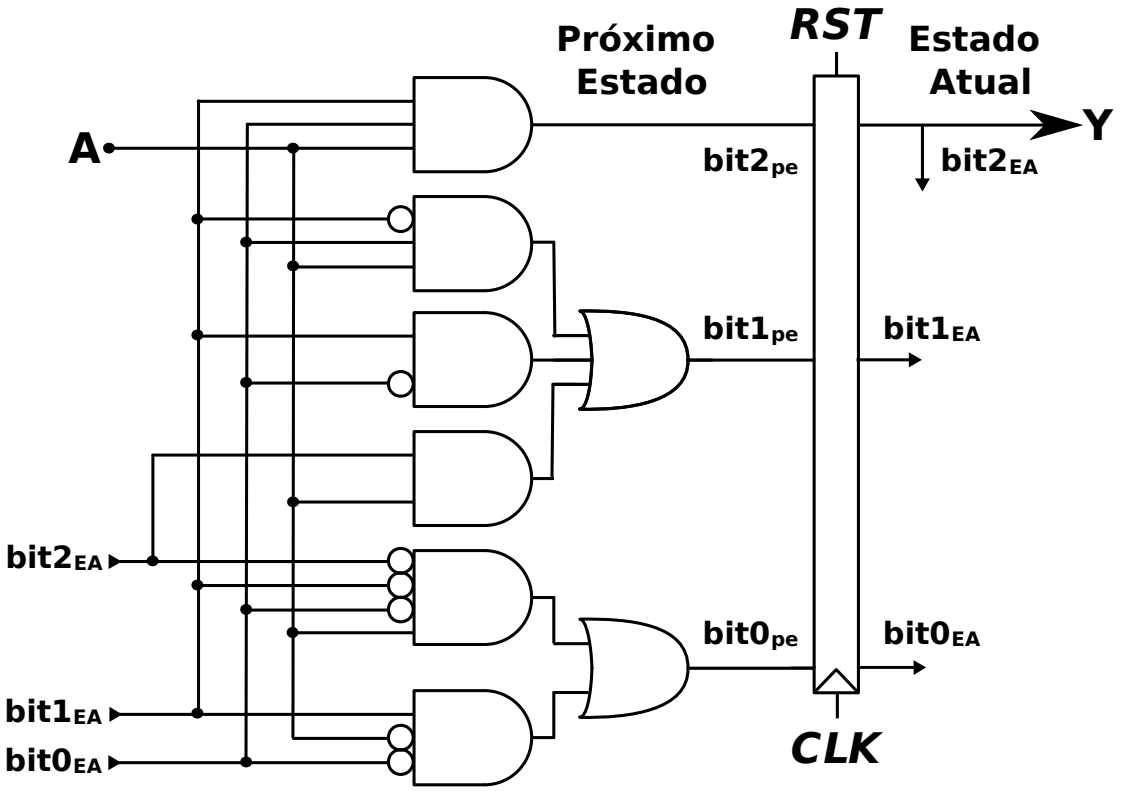
- Resumindo:

$bit2_{pe} = bit1_{EA} * bit0_{EA} * A$

$bit1_{pe} = bit1'_{EA} * bit0_{EA} * A + bit1_{EA} * bit0'_{EA} + bit2_{EA} * A$

$bit0_{pe} = bit2'_{EA} * bit1'_{EA} * bit0'_{EA} * A + bit1_{EA} * bit0'_{EA} * A'$

$Y = bit2_{EA}$



Codificação do Estado Atual = $bit2_{EA}$ $bit1_{EA}$ $bit0_{EA}$

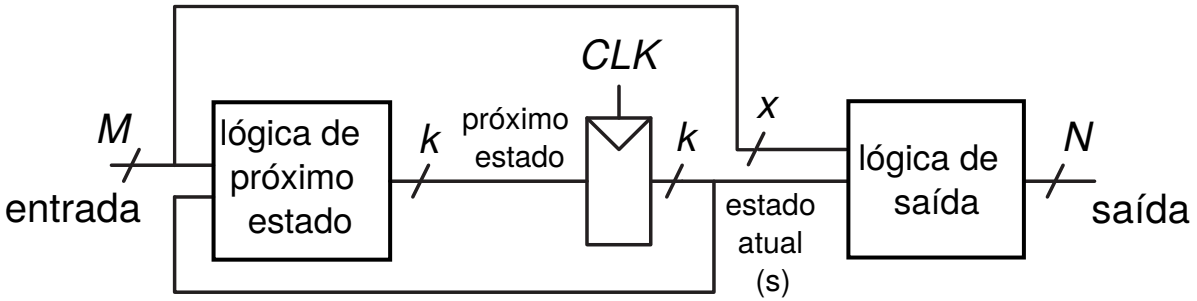
Codificação do Próximo Estado = $bit2_{pe}$ $bit1_{pe}$ $bit0_{pe}$

Insper

Máquina de Mealy

- Nesta máquina, a saída (y):
  - Depende do estado atual interno (s(t));
  - Da entrada atual (x(t));
  - E de uma transformação H (lógica de saída).
- Podemos dizer que:

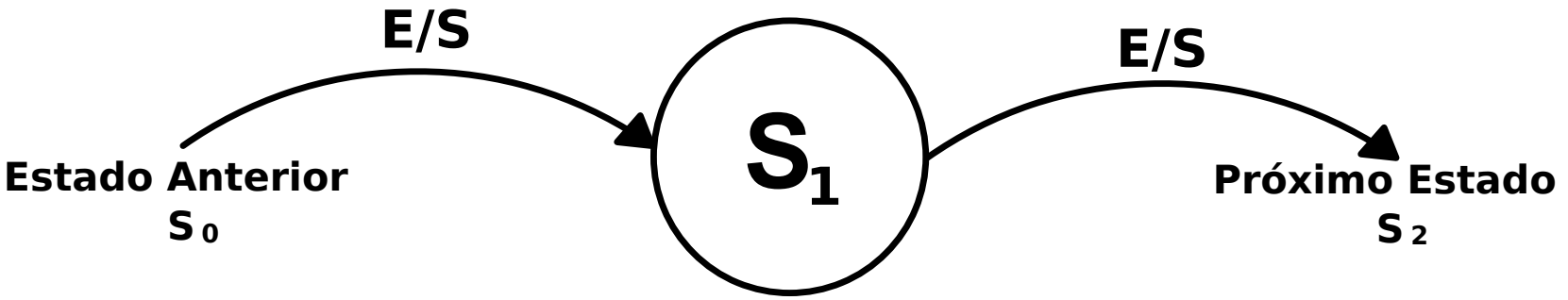
$$y(t) = H(s(t), x(t))$$



Máquina de estados do tipo: Mealy

Insper

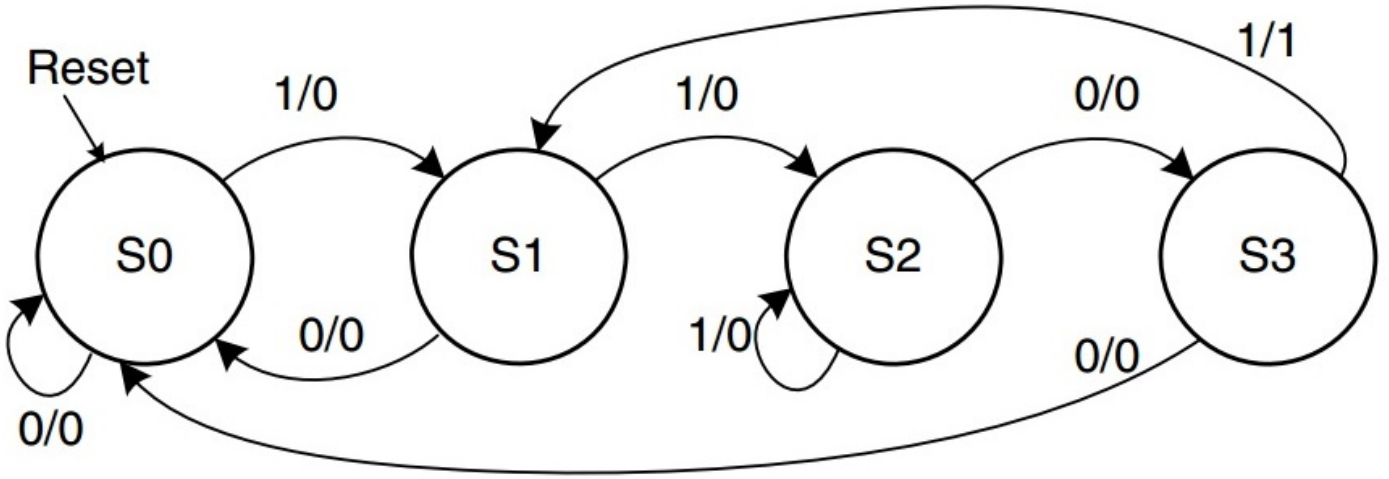
- Na maquina de Mealy:
  - A transição de saída pode ocorrer de forma assíncrona (independente do Clock):
    - Sinais de entrada conectados com a lógica de saída.
- Além disso, o diagrama de estados:
  - É diferente do diagrama da máquina de Moore.



E : Entrada que gera a transição.

S : Saída após a transição.

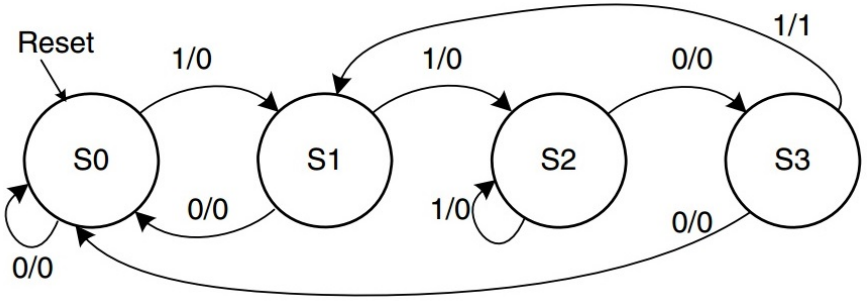
- Vamos usar o mesmo exemplo do detector de sequência 1101, utilizado no estudo da máquina de Moore.
- O diagrama de estados possui uma quantidade menor de estados:
  - Somente 4 contra 5 da máquina de Moore;
  - Com isso, se economiza um Flip Flop;
  - No arco temos a indicação de A/Y (entrada/saída).



Insper

- Roteiro para a criação de máquinas de Mealy:
  - Identifique as entradas e saídas;
  - Esboce o diagrama de estados;
  - Monte a tabela que combina a transição de estados e a saída;
  - Escolha a codificação dos estados (binária, código de Gray, one-hot, etc...);
  - Monte a nova tabela de transição e saída de acordo com a codificação e o Flip Flop escolhidos;
  - Monte os mapas de Karnaugh para a nova tabela e obtenha as equações do circuito:
    - Para o próximo estado;
    - Para a saída.
  - Desenhe o resultado final.

- Utilizando o diagrama de estados:



  - Preenche-se a tabela de transição e os valores de saída.

Estado Atual	Entrada (A)	Próximo Estado	Saída (Y)
S0	0	S0	0
S0	1	S1	0
S1	0	S0	0
S1	1	S2	0
S2	0	S3	0
S2	1	S2	0
S3	0	S0	0
S3	1	S1	1

Insper

- Codificamos os estados em binário:
  - E preenchemos a tabela de transição e saída:

Estado Atual		Entrada	Próximo Estado		Saída
bit 1_{EA}	bit 0_{EA}	A	bit 1_{pe}	bit 0_{pe}	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

- A saída e dada por:

$$Y = bit1_{EA} * bit0_{EA} * A$$

Estado Atual		Entrada	Próximo Estado		Saída
$bit1_{EA}$	$bit0_{EA}$	A	$bit1_{pe}$	$bit0_{pe}$	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

- Para $bit1_{pe}=1$, temos duas condições:
  - $bit1_{pe} = bit1'_{EA}*bit0_{EA}*A$ e $bit1_{pe} = bit1_{EA}*bit0'_{EA}$

$bit1_{pe} = bit1'_{EA}*bit0_{EA}*A + bit1_{EA}*bit0'_{EA}$

Estado Atual		Entrada	Próximo Estado		Saída
$bit1_{EA}$	$bit0_{EA}$	A	$bit1_{pe}$	$bit0_{pe}$	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

- Para $bit0_{pe}=1$, temos três condições:

$bit0_{pe} = bit1'_{EA}*bit0'_{EA}*A$

$bit0_{pe} = bit1_{EA}*S'0*A'$

$bit0_{pe} = bit1_{EA}*bit0_{EA}*A$

$bit0_{pe}$ =

$bit1'_{EA}*bit0'_{EA}*A$

+ $bit1_{EA}*bit0'_{EA}*A'$

+ $bit1_{EA}*bit0_{EA}*A$

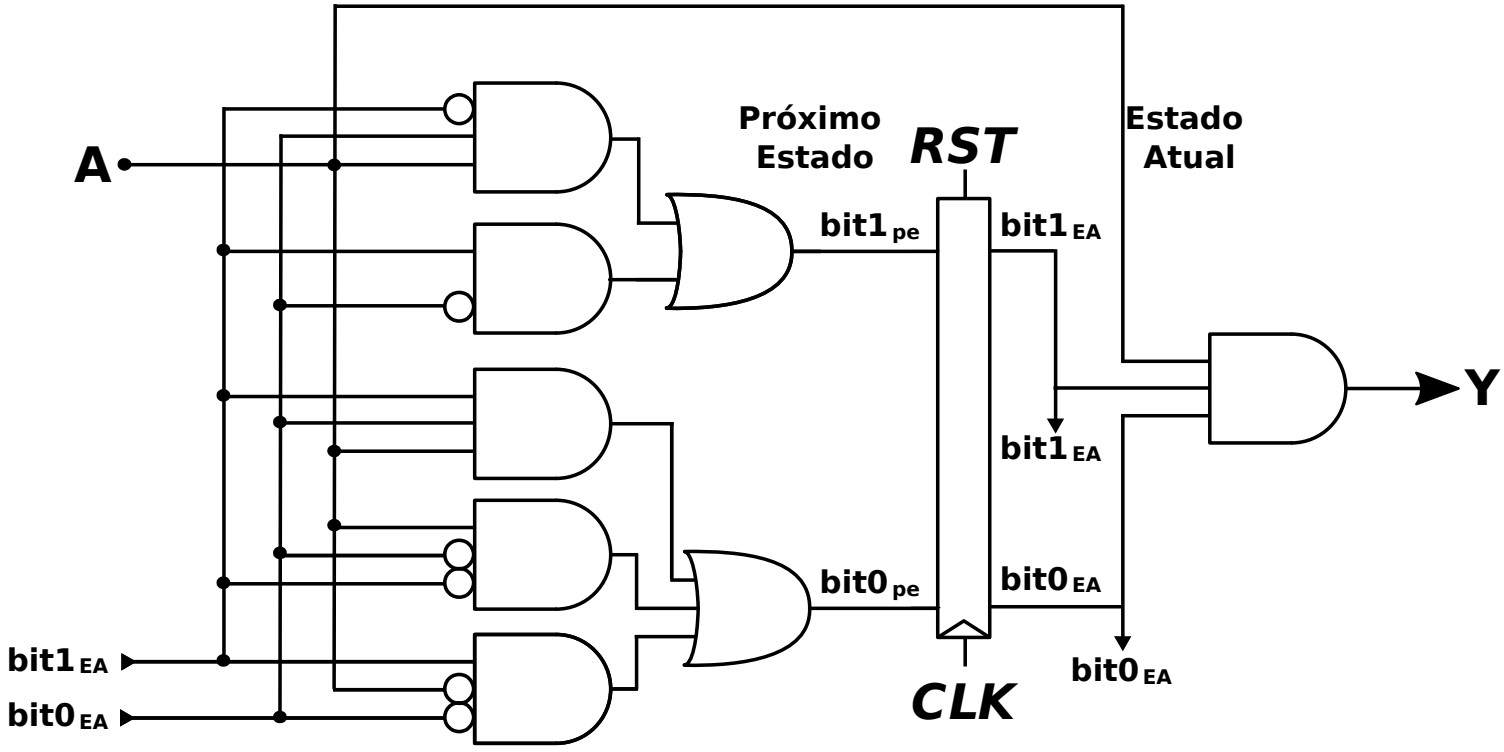
Estado Atual		Entrada	Próximo Estado		Saída
$bit1_{EA}$	$bit0_{EA}$	A	$bit1_{pe}$	$bit0_{pe}$	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

- Resumindo:

$bit1_{pe} = bit1'_{EA} * bit0_{EA} * A + bit1_{EA} * bit0'_{EA}$

$bit0_{pe} = bit1'_{EA} * bit0'_{EA} * A + bit1_{EA} * bit0'_{EA} * A + bit1_{EA} * bit0_{EA} * A$

$Y = bit1_{EA} * bit0_{EA} * A$



Codificação do Estado Atual = bit1 $_{EA}$ bit0$_{EA}$

Codificação do Próximo Estado = bit1 $_{pe}$ bit0$_{pe}$

Insper

Comparativo Mealy x Moore

- Velocidade de resposta:
  - Mealy: a saída pode responder a variações nas entradas sem esperar pelo sincronismo do Clock (mais rápido);
  - Moore: a saída depende do sincronismo do Clock para mudar seu estado (mais lento).

- Glitchs na entrada:
  - Mealy: transfere os glitchs das entradas para as saídas;
  - Moore: filtra os glitchs de entrada (sincronismo).

- Duração dos sinais de saída:
  - Mealy: pode ser curta, dependendo do sinal de entrada;
  - Moore: possui pelo menos um ciclo de Clock.

- Complexidade do circuito:
  - Mealy: pode eliminar um Flip Flop e toda a lógica relativa a ele, o que pode ser significativo;
  - Moore: em geral utiliza um estado a mais.

- Resumo geral:
  - As máquinas de Mealy, pelo seu circuito menor e resposta imediata à entrada, podem ser mais rápidas.
  - As máquinas de Moore tratam melhor os atrasos de propagação, sincronizando com o Clock, o que é bem adequado a circuitos com pipeline implementados em FPGAs.
  - Os glitchs podem não ser um problema, por si só, ou se o resto do circuito possuir etapas sincronizadas.

- Para auxiliar na escolha entre Moore e Mealy.
- Moore – se o seu problema define que:
  - A saída seja ativada no estado subsequente à entrada que finaliza a sequência;
  - Não apareçam glitchs nas saídas, mesmo que existam nas entradas.
- Mealy – se o seu problema define que:
  - A saída seja ativada assim que a última entrada, que finaliza a sequência, inicie;
  - A saída não precise esperar pelo sincronismo do Clock;
  - Se minimize o circuito (geralmente);
  - Possam existir glitchs na saída.

Insper

www.insper.edu.br