Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática – ICEI
Arquitetura de Computadores I

ARQ1 _ Aula_10
Tema: Introdução aos autômatos

Preparação

Como preparação para o início das atividades, recomendam-se

a.) leitura prévia do resumo teórico, do detalhamento na apostila e referências recomendadas
b.) estudo e testes dos exemplos
c.) assistir aos seguintes vídeos:

https://www.youtube.com/watch?v=YebQtg-tTfI
https://www.youtube.com/watch?v=xKoIdX6XBps
https://www.youtube.com/watch?v=SA2o7UnYqbw

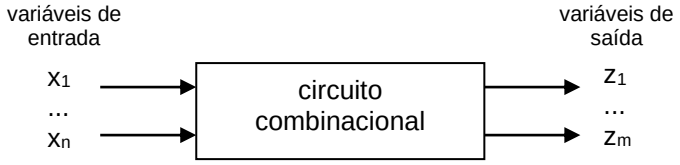
Orientação geral:

Apresentar todas as soluções em apenas um arquivo com formato texto (.txt).
Sugere-se usar como nome Guia_xx.txt, onde xx indicará o guia, exemplo Guia_01.txt.
Todos os arquivos deverão conter identificações iniciais com o nome e matrícula,
no caso de programas, usar comentários.
As implementações e testes dos exemplos em Verilog (**.v**) fornecidos como pontos de partida,
também fazem parte da atividade e deverão ter os códigos fontes entregues **separadamente**,
com o código fonte, a fim de que possam ser compilados e testados. Entregar os módulos de testes.
Sugere-se usar como nomes Guia_01yy.v, onde yy indicará a questão, exemplo Guia_0101.v
As saídas de resultados, opcionalmente, poderão ser copiadas ao final do código, em comentários.
Quaisquer outras anotações, observações ou comentários poderão ser colocadas em arquivo texto (README.txt) acompanhando a entrega.

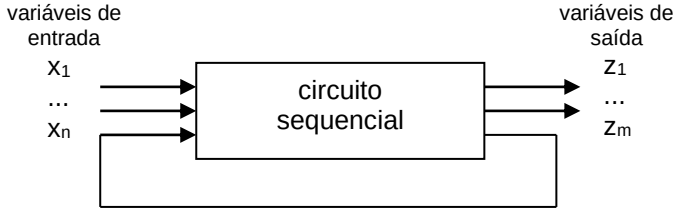
Outras formas de solução serão **opcionais**; não servirão para substituir as atividades
a serem avaliadas. Caso entregues, poderão contar apenas como atividades extras.
Os programas com funções desenvolvidas em C, Java ou Python (c, .java, py),
como os modelos usados para verificação automática de testes das respostas;
caso entregues, também deverão estar em arquivos **separados**, com o código fonte,
a fim de serem compilados e testados.
As execuções deverão, preferencialmente, serão testadas mediante uso de redirecionamento de entradas e saídas padrões, cujos dados/resultados deverão ser armazenados em arquivos textos.
Os resultados poderão ser anexados ao código, ao final, como comentários.
Planilhas, caso venham a ser solicitadas, deverão ser **programadas** e/ou usar funções nativas.
Serão suplementares e opcionais, e deverão ser entregues em formato texto, preferencialmente,
com colunas separadas por tabulações ou no formato (**.csv**), acompanhando a solução em texto.
Arquivos em formato (**.pdf**), fotos, cópias de tela ou soluções manuscritas também poderão ser aceitos como recursos suplementares para visualização, mas não servirão como substitutos e **não** terão validade para fins de avaliação.
Os *layouts* de circuitos deverão ser entregues no formato (**.circ**), identificados internamente.
Os *layouts* de diagramas deverão ser entregues no formato (**.jff**), identificados externamente.
Figuras exportadas pela ferramenta serão aceitas apenas como arquivos para visualização,
mas **não** terão validade para fins de avaliação. Separar versões completas (a) e simplificadas (b).

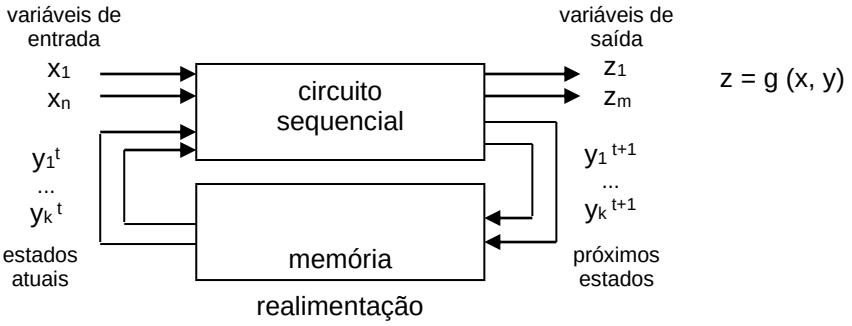
Circuitos sequenciais

Um *circuito combinacional* é aquele em que a(s) saída(s) depende(m) de uma combinação das entradas.




Um *circuito sequencial*, além de uma combinação das entradas, depende de uma combinação de outras variáveis que definem o estado em que o sistema se encontrava. Isto significa que um sistema deverá ter *memória*; para passar ao próximo estado, precisará guardar informações sobre o estado atual.

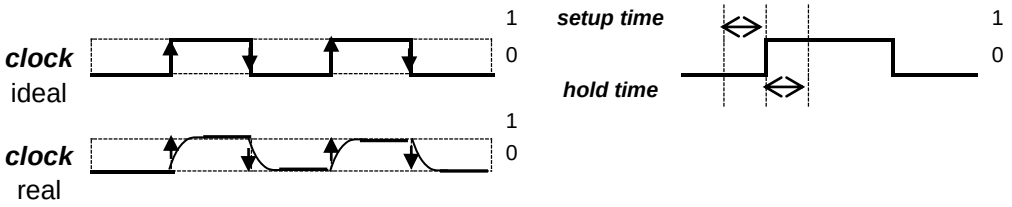



Basicamente, há dois tipos de circuitos sequenciais:

- assíncronos - em que os estados podem mudar a qualquer instante
- síncronos - em que os estados mudam em instantes bem determinados

As mudanças de estados que ocorrerão em instantes determinados serão orientadas por um sinal de temporização (***clock***). Se ocorrerem as transições ocorrerem durante uma variação de 0 para 1 ( ↑ - borda de subida), o sistema será dito de *nível alto*; caso contrário, durante uma variação de 1 para 0 ( ↓ - borda de descida), o sistema será dito de *nível baixo*. As especificações de tempo para circuitos sequenciais também incluirão o tempo para a transição se estabilizar (***setup time***) e o tempo, após a transição, em que o sinal deve se mantiver constante (***hold time***).



Máquinas de estados finitos (***Finite State Machines***)

Uma *máquina de estados finitos*, ou simplesmente *autômato finito*, é um modelo de comportamento composto de estados, transições e ações. Um estado armazena uma informação sobre a história de um sistema (reflete como as mudanças nas entradas trouxeram o sistema até o estado atual). Uma transição indica uma mudança de estado e é descrita por uma condição que a permite. Uma ação é a descrição de uma atividade executada em certo instante.

Máquinas de estados finitos podem ser usadas para descrever circuitos sequenciais pois suas saídas e seus novos estados são funções de suas entradas e de seus estados atuais.

Diagrama de estados

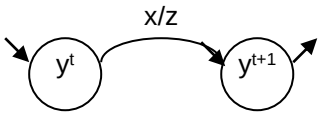



Tabela de estados

| entrada / estado\ atual | x | |
|---|---|---|
| y | | |
| | $y^{t+1}/z$ | |
| | | |

Exemplo:

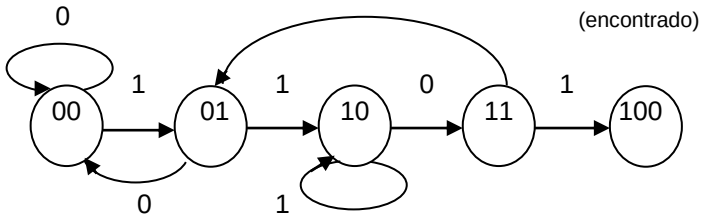
Considerar um circuito capaz de identificar a sequência binária (abcd=1101).

Autômato finito:

Tabela de Estados

| estado atual | \ código y | entradas nome | entrada (x) x=0 | entrada (x) x=1 |
|---|---|---|---|---|
| 0 | 000 | início | 000 (início) | 001 ( id1 ) |
| 1 | 001 | id1 | 000 (início) | 010 (id11) |
| 2 | 010 | id11 | 011 (id110) | 010 (id11) |
| 3 | 011 | id110 | 001 ( id1 ) | 100 (fim) |
| 4 | 100 | fim | 100 (fim) | 100 (fim) |

Diagrama de estados

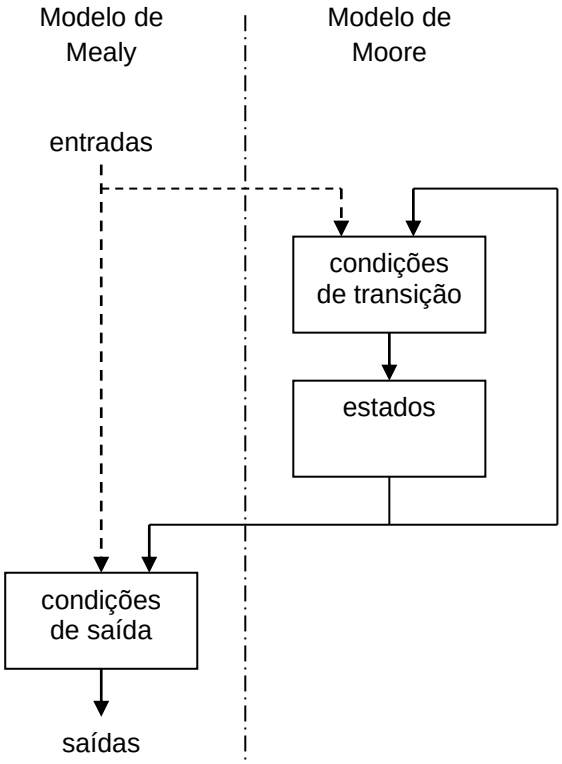


Os modelos de Mealy e Moore são comumente usados para descrever máquinas de estados finitos.

Caso a saída seja função do estado da máquina e de suas entradas, então o modelo de Mealy será melhor empregado, pois reage mais rapidamente às variações das entradas.

Se a saída for função apenas do estado, o modelo de Moore será melhor empregado, pois garante a transição completa entre estados, antes de emitir alguma saída.

Na prática, esses dois modelos poderão ser combinados para oferecer uma descrição melhor do funcionamento de uma máquina de estados finitos.



Exemplo:

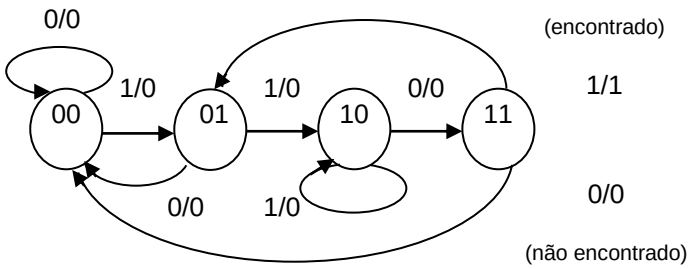
Considerar um circuito capaz de identificar a sequência binária (abcd=1101).

Modelo de Mealy:

Tabela de Estados

| estado atual | \ código y | entradas nome | x x=0 | x=1 |
|---|---|---|---|---|
| | 0 0 | início | início / s=0 | id1 / s=0 |
| | 0 1 | id1 | início / s=0 | id11 / s=0 |
| | 1 0 | id11 | id110 / s=0 | id11 / s=0 |
| | 1 1 | id110 | início / s=0 | id1 / s=1 |

Diagrama de estados (Mealy)

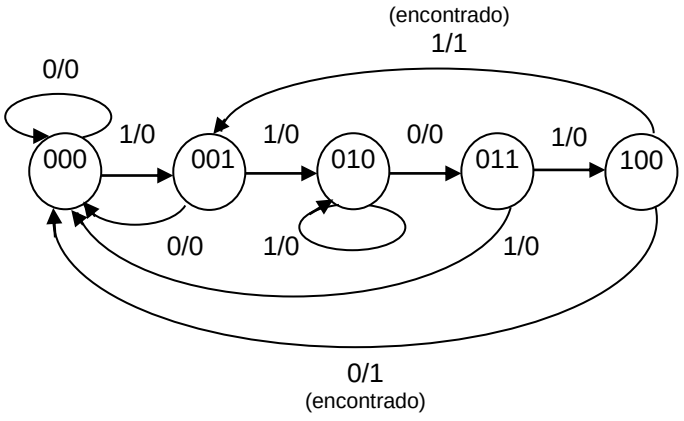



Modelo de Moore:

Tabela de Estados

| estado atual | \ código y | entradas nome | x x=0 | x=1 |
|---|---|---|---|---|
| | 0 0 0 | início | início / s=0 | id1 / s=0 |
| | 0 0 1 | id1 | início / s=0 | id11 / s=0 |
| | 0 1 0 | id11 | id110 / s=0 | id11 / s=0 |
| | 0 1 1 | id110 | início / s=0 | id1101 / s=0 |
| | 1 0 0 | id1101 | início / s=0 | id1 / s=1 |

Diagrama de estados (Moore)



Atividade: Circuitos sequenciais -Máquinas de Estados Finitos (*Finite State Machines* - FSM)

Todos os circuitos deverão ser simulados no JFLAP e testados com as respectivas indicações.

01.) Projetar e descrever em JFLAP uma máquina de Mealy,
para implementar um comportamento semelhante ao descrito na tabela abaixo.

| estado atual | entrada (x) / (próximo estado, saída) x=0 | x=1 |
|---|---|---|
| > 0 | (1, 0) | (0, 0) |
| 1 | (2, 0) | (0, 0) |
| 2 | (2, 0) | (3, 1) |
| 3 | (2, 0) | (0, 0) |

Testar:
a.) 001101110
b.) 0000101101

02.) Projetar e descrever em JFLAP uma máquina de Moore
para implementar um comportamento semelhante ao descrito na tabela abaixo.
DICA: Ver modelo de Moore.

| estado atual | entrada (x) / próximo estado x=0 | x=1 | saída |
|---|---|---|---|
| > 0 | 0 | 3 | 0 |
| 1 | 2 | 0 | 0 |
| 2 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 |

Testar:
a.) 0011100110
b.) 000110011101

03.) Projetar e descrever em JFLAP uma máquina de Turing,
para complementar todos os bits de uma sequência, exceto o último bit à direita.
DICA: Escrever o complemento de 1 de todo o que for lido, menos o último.

Testar:
a.) 1101
b.) 0100

04.) Projetar e descrever em JFLAP uma máquina de estados finitos (FSM),
para identificar sequências de duplas com três valores alternados 010 ou 101,
com interseção.
DICA: Usar os estados para contar.

Testar:
a.) 00110100110
b.) 000110100101

05.) Projetar e descrever em JFLAP um autômato de pilha (PDA),
para implementar um reconhecedor de uma sequência igual a 0110,
sem interseção.
DICA: Estado final deverá ter apenas o valor 1 no topo da pilha.

Testar:
a.) 10110110
b.) 1011001100

Extras

06.) Projetar e descrever em JFLAP uma máquina de Mealy,
para implementar um reconhecedor de sequência igual a 111,
com interseção.
Caso seja simulado por módulo no Logisim,
apresentar *layout* do circuito e subcircuitos.

07.) Projetar e descrever em JFLAP uma máquina de Mealy,
para implementar um reconhecedor de sequência igual a 0001,
sem interseção.
Caso seja simulado por módulo no Logisim,
apresentar *layout* do circuito e subcircuitos.

Modelo em Logisim para um detector de sequência 1101