

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

ARQ1 _ Aula_11

Tema: Introdução à linguagem Verilog e simulação em Logisim (reconhecedores de Mealy e Moore)

Orientação geral:

Atividades previstas como parte da avaliação

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**,
a fim de que possam ser compilados e testados.

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.

Atividades extras e opcionais

Outras formas de solução serão **opcionais**; não servirão para substituir as atividades
a serem avaliadas. Caso entregues, poderão contar apenas como atividades extras.

Os *layouts* de circuitos deverão ser entregues no formato (.circ), identificados internamente.
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).

Arquivos em formato (.pdf), fotos, cópias de tela ou soluções manuscritas também serão aceitos
como recursos suplementares para visualização, e **não** terão validade para fins de avaliação.

01.) Projetar um circuito em Logisim para realizar a descrição em Verilog de um módulo para implementar uma máquina de estados finitos segundo a abordagem de Mealy.
O nome do arquivo deverá ser Mealy_1101.v, e poderá seguir o modelo descrito abaixo.

```
// -----  
// --- Mealy FSM  
// -----  
  
/*  
  
                Mealy FSM Diagram  
  
          /           \         \      |   // found  
        1       v     1             0 1 |  
[start] ----> [id1] ----> [id11] ----> [id110]  
    ^   \|            1 /   ^               0 | // not found  
    \|   \|              \|                 \|  
    \|___\|_____|_____ \|_____  
    \|_____\|_____|_____  
  
*/  
  
// constant definitions  
`define found      1  
`define notfound_0  
  
// FSM by Mealy  
module mealy_1101 ( y , x, clk, reset );  
output y;  
input  x;  
input  clk;  
input  reset;  
  
reg    y;  
  
parameter    // state identifiers  
    start = 2'b00,  
    id1   = 2'b01,  
    id11  = 2'b11,  
    id110 = 2'b10;
```

```

// next state logic
always @( x or E1 )
begin
y = `notfound;
case ( E1 )
start:
if ( x )
E2 = id1;
else
E2 = start;
id1:
if ( x )
E2 = id11;
else
E2 = start;
id11:
if ( x )
E2 = id11;
else
E2 = id110;
id110:
if ( x )
begin
E2 = id1;
y = `found;
end
else
begin
E2 = start;
y = `notfound;
end
default: // undefined state
E2 = 2'bxx;
endcase
end // always at signal or state changing

// state variables
always @( posedge clk or negedge reset )
begin
if ( reset )
E1 = E2; // updates current state
else
E1 = 0; // reset
end // always at signal changing

endmodule // mealy_1101

```

[illegible]

```

// next state logic
always @( x or E1 )
begin
case( E1 )
start:
if ( x )
E2 = id1;
else
E2 = start;
id1:
if ( x )
E2 = id11;
else
E2 = start;
id11:
if ( x )
E2 = id11;
else
E2 = id110;
id110:
if ( x )
E2 = id1101;
else
E2 = start;
id1101:
if ( x )
E2 = id11;
else
E2 = start;
default: // undefined state
E2 = 3'bxxx;
endcase
end // always at signal or state changing

// state variables
always @( posedge clk or negedge reset )
begin
if ( reset )
E1 = E2; // updates current state
else
E1 = 0; // reset
end // always at signal changing

// output logic
always @( E1 )
begin
y = E1[2]; // first bit of state value (MOORE indicator)
end // always at state changing

endmodule // moore_1101

```

- 03.) Projetar um circuito em Logisim para realizar a descrição em Verilog de um módulo para testar as máquinas de estados finitos segundo as abordagens de Mealy e de Moore. O nome do arquivo deverá ser Guia_1100.v, e poderá seguir o modelo descrito abaixo.

```
// -----  
// --- Mealy-Moore FSM  
// -----  
//  
  
`include "mealy_1101.v"  
`include "moore_1101.v"  
  
module Guia_1101;  
    reg  clk, reset, x;  
    wire m1, m2;  
  
    mealy_1101 mealy1 ( m1, x, clk, reset );  
    moore_1101 moore1 ( m2, x, clk, reset );  
  
    initial  
    begin  
        $display ( "Time    X   Mealy Moore" );  
  
        // initial values  
        clk  = 1;  
        reset = 0;  
        x    = 0;  
  
        // input signal changing  
        #5  reset = 1;  
        #10 x = 1;  
        #10 x = 0;  
        #10 x = 1;  
        #20 x = 0;  
        #10 x = 1;  
        #10 x = 1;  
        #10 x = 0;  
        #10 x = 1;  
  
        #30 $finish;  
    end // initial  
  
    always  
        #5 clk = ~clk;  
  
    always @( posedge clk )  
    begin  
        $display ( "%4d %4b %4b %5b", $time, x, m1, m2 );  
    end // always at positive edge clocking changing  
  
endmodule // Guia_1100
```

- 04.) Projetar um circuito em Logisim para realizar a
a descrição em Verilog de um módulo
para implementar uma máquina de estados finitos (FSM),
capaz de reconhecer apenas a primeira sequência
(1001) que aparecer e parar
(10010010 deverá ser reconhecida apenas uma vez).
O nome do arquivo deverá ser Guia_1101.v.
Incluir previsão de testes e verificação do circuito pelo Logisim.
- 05.) Projetar um circuito em Logisim para realizar a
a descrição em Verilog de um módulo
para implementar uma máquina de estados finitos,
segundo a abordagem de Mealy, para reconhecer
uma sequência (1001) sem interseção
(100100101 deverá ser reconhecida apenas uma vez).
O nome do arquivo deverá ser Guia_1102.v.
Incluir previsão de testes e verificação do circuito pelo Logisim.
- 06.) Projetar um circuito em Logisim para realizar a
a descrição em Verilog de um módulo
segundo a abordagem de Moore, para reconhecer
uma sequência (1001) com interseção
(10010010001001 deverá ser reconhecida três vezes).
O nome do arquivo deverá ser Guia_1103.v.
Incluir previsão de testes e verificação do circuito pelo Logisim.

Extra

- 07.) Projetar um circuito em Logisim para realizar a
a descrição em Verilog de um módulo
para implementar uma máquina de estados finitos,
capaz de reconhecer uma sequência de
quatro dígitos binários que termine com
três valores iguais a 000 (x000, por exemplo).
O nome do arquivo deverá ser Guia_1104.v.
Incluir previsão de testes e verificação do circuito pelo Logisim.
- 08.) Projetar um circuito em Logisim para realizar a
a descrição em Verilog de um módulo
para implementar uma máquina de estados finitos,
capaz de reconhecer uma sequência de
três dígitos binários alternados (010 ou 101).
O nome do arquivo deverá ser Guia_1105.v.
Incluir previsão de testes e verificação do circuito pelo Logisim.

Instruções para ver as cartas de tempo no GTKWave:

01.) Abrir o módulo de visualização (GTKWave)

02.) Selecionar a pasta de trabalho:

File

Open

Guia_1100 (.vcd) (por exemplo)

03.) Selecionar os sinais desejados:

clk (sinal a ser visto)

clock (outro sinal a ser visto)

(selecionar, arrastar e soltar na coluna à direita)

Modelo em Logisim para um detector de sequência 1101

Sequence detector

