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

ARQ1 _ Aula_09
Tema: Introdução à linguagem Verilog

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=o6kS7izbM7o

https://www.asic-world.com/verilog/art_testbench_writing2.html
https://referencedesigner.com/tutorials/verilogexamples/verilog_ex_06.php
https://www.testbench.in/TB_08_CLOCK_GENERATOR.html

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.
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.
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).

Atividade: Circuitos sequenciais

Exemplo

Projetar e descrever em Verilog um módulo gerador de ***clock***.
O nome do arquivo deverá ser Guia_0900.v, e poderá seguir o modelo descrito abaixo.
Incluir previsão de testes e verificação da carta de tempo usando GTKWave.

```
// ---------------------------
// -- test clock generator (1)
// ---------------------------

module clock ( output clk );
reg      clk;

 initial
  begin
   clk = 1'b0;
  end

 always
  begin
   #12 clk = ~clk;
  end

endmodule // clock ( )


module Guia_0900;

 wire  clk;
 clock CLK1 ( clk );

 initial begin
  $dumpfile ( "Guia_0900.vcd" );
  $dumpvars;

  #120 $finish;
 end

endmodule // Guia_0901 ( )
```

01.) Projetar e descrever em Verilog módulos geradores de pulso (***pulse***) e gatilho (***trigger***). O nome do arquivo deverá ser Guia_0901.v, e poderá seguir o modelo descrito abaixo. Incluir previsão de testes e verificação da carta de tempo usando GTKWave.

```
// ----------------------------
// -- test clock generator (2)
// ----------------------------

module clock ( output clk );
 reg      clk;

 initial
  begin
   clk = 1'b0;
  end

 always
  begin
   #12 clk = ~clk;
  end
endmodule

module pulse ( signal, clock );
 input  clock;
 output signal;
 reg      signal;

 always @ ( clock )
  begin
        signal = 1'b1;
  #3  signal = 1'b0;
  #3  signal = 1'b1;
  #3  signal = 1'b0;
  end
endmodule // pulse

module trigger ( signal, on, clock );
 input  on, clock;
 output signal;
 reg      signal;

 always @ ( posedge clock & on )
  begin
  #60 signal = 1'b1;
  #60 signal = 1'b0;
  end
endmodule // trigger
```

```
module Guia_0901;

 wire   clock;
 clock clk ( clock );

 reg   p;

 wire  p1,t1;

 pulse   pulse1  ( p1, clock );
 trigger trigger1 ( t1, p, clock );

 initial begin
   p = 1'b0;
 end

 initial begin
  $dumpfile ( "Guia_0901.vcd" );
  $dumpvars ( 1, clock, p1, p, t1 );

  #060 p = 1'b1;
  #120 p = 1'b0;
  #180 p = 1'b1;
  #240 p = 1'b0;
  #300 p = 1'b1;
  #360 p = 1'b0;
  #376 $finish;
 end

endmodule // Guia_0901
```

02.) Projetar e descrever em Verilog módulos
geradores de pulso (***pulse***) com períodos diferentes.
O nome do arquivo deverá ser Guia_0902.v,
e poderá seguir o modelo descrito a seguir.
O gerador de ***clock*** do Guia_0900.v deverá
ser previamente isolado em um arquivo único
cujo nome deverá ser **clock.v**, para uso posterior.
Incluir previsão de testes e verificação da
carta de tempo usando GTKWave.

```
// ----------------------------
// -- test clock generator (3)
// ----------------------------

`include "clock.v"

module pulse1 ( signal, clock );
 input   clock;
 output signal;
 reg      signal;

 always @ ( posedge clock )
  begin
       signal = 1'b1;
  #4  signal = 1'b0;
  #4  signal = 1'b1;
  #4  signal = 1'b0;
  #4  signal = 1'b1;
  #4  signal = 1'b0;
  end
endmodule // pulse

module pulse2 ( signal, clock );
 input  clock;
 output signal;
 reg      signal;

 always @ ( posedge clock )
  begin
       signal = 1'b1;
  #5  signal = 1'b0;
  end
endmodule // pulse

module pulse3 ( signal, clock );
 input  clock;
 output signal;
 reg    signal;

 always @ ( negedge clock )
  begin
       signal = 1'b1;
  #15 signal = 1'b0;
  #15 signal = 1'b1;
  end
endmodule // pulse

module pulse4 ( signal, clock );
 input  clock;
 output signal;
 reg    signal;

 always @ ( negedge clock )
  begin
       signal = 1'b1;
  #20 signal = 1'b0;
  #20 signal = 1'b1;
  #20 signal = 1'b0;
  end
endmodule // pulse
```

```
module Guia_0902;

 wire  clock;
 clock clk ( clock );

 wire  p1,p2,p3,p4;

 pulse1   pls1   ( p1, clock );
 pulse2   pls2   ( p2, clock );
 pulse3   pls3   ( p3, clock );
 pulse4   pls4   ( p4, clock );

 initial begin
  $dumpfile ( " Guia_0902.vcd" );
  $dumpvars ( 1, clock, p1, p2, p3, p4 );

  #480 $finish;
 end

endmodule // Guia_0902
```

03.) Projetar e descrever em Verilog um módulo gerador de pulso (***pulse***) com frequência igual a um terço da frequência (um terço do período) do gerador do Guia_0900.v.
O nome do arquivo deverá ser Guia_0903.v.
Incluir previsão de testes e verificação da carta de tempo usando GTKWave.

04.) Projetar e descrever em Verilog um módulo gerador de pulso (***pulse***) com frequência igual a quatro vezes a frequência (quatro vezes o período) do gerador do Guia_0900.v.
O nome do arquivo deverá ser Guia_0904.v.
Incluir previsão de testes e verificação da carta de tempo usando GTKWave.

05.) Projetar e descrever em Verilog um módulo gerador de pulso (***pulse***) com marcação igual a 3 unidades de tempo, sincronizado com a borda de subida do gerador do Guia_09001.v.
O nome do arquivo deverá ser Guia_0905.v.
Incluir previsão de testes e verificação da carta de tempo usando GTKWave.
DICA: Usar *always* @(posedge clk).

Extra

06.) Projetar e descrever em Verilog um módulo gerador de pulso (***pulse***) com marcação igual a 4 unidades de tempo, sincronizado com a borda de descida do gerador do Guia_0900.v.
O nome do arquivo deverá ser Guia_0906.v.
Incluir previsão de testes e verificação da carta de tempo usando GTKWave.
DICA: Usar *always* @(negedge clk).

07.) Projetar e descrever em Verilog um módulo gerador de pulso (***pulse***) com marcação igual a 5 unidades de tempo, sincronizado com o nível alto e estável do gerador do Guia_0900.v.
O nome do arquivo deverá ser Guia_0907.v.
Incluir previsão de testes e verificação da carta de tempo usando GTKWave.
DICA: Usar *always* @(clk).

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_0900 (.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)