CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS –
CEFET/MG

Celso França Neto (20203018570)
Diogo Emanuel Antunes Santos (20213002091)

AULA 10: Implementação em Verilog dos componentes do nRisc que armazenam estado.

Laboratório de Arquitetura e Organização de Computadores

BELO HORIZONTE
2022

1) **Apresente o código fonte, em Verilog, dos módulos que descrevem os componentes do seu nRisc que armazenam valores.**

module PC:

```
1   module PC (
2     input wire clock,
3     input wire [7:0] in,
4     output wire [7:0] out,
5     input Encerra
6   );
7
8     reg [7:0] pc;
9
10    always @(posedge clock)
11      begin
12      if(~Encerra) begin
13        pc = in;
14      end
15    end
16
17    assign out = pc;
18
19  endmodule
```

module bacoInstrucoes:

```
51  module bancoInstrucoes (
52    input wire clock,
53    input wire [7:0] in,
54    output wire [7:0] out
55  );
56
57    reg [7:0] banco_instrucoes;
58
59    always @(posedge clock)
60      begin
61        case (in)
62          8'b00000000: banco_instrucoes = 8'b00010001;
63          8'b00000001: banco_instrucoes = 8'b00101101;
64          8'b00000010: banco_instrucoes = 8'b01110010;
65          8'b00000011: banco_instrucoes = 8'b01010010;
66          8'b00000100: banco_instrucoes = 8'b10010000;
67          8'b00000101: banco_instrucoes = 8'b10100001;
68          8'b00000110: banco_instrucoes = 8'b11000011;
69           8'b00000111: banco_instrucoes = 8'b11100000;
70          default: banco_instrucoes = 8'bxxxxxxxx;
71          endcase
72      end
73    assign out = banco_instrucoes;
74  endmodule
```

module bancoRegs:

```
 89  module bancoRegs (
 90    input wire clock,
 91    input wire [1:0] in1,
 92    input wire [1:0] in2,
 93    input wire inec,
 94    input wire [7:0] dado,
 95    output wire [7:0] out1,
 96    output wire [7:0] out2,
 97    input EscreveReg
 98  );
 99    reg [7:0] regs [1:0];
100    reg [7:0] valores [1:0];
101
102    always @(posedge clock)
103      begin
104        if(EscreveReg) begin
105          regs[inec] = dado;
106        end
107     end
108
109     always @(negedge clock)
110        begin
111          valores [0] = regs [in1];
112          valores [1] = regs [in2];
113        end
114
115      assign out1 = valores[0];
116      assign out2 = valores[1];
117
118  endmodule
```

module bancoMem:

```
120  module bancoMem(
121  input wire clock,
122    input wire [7:0] endereco,
123    input wire [7:0] dado,
124    output wire [7:0] out,
125    input lerMem,
126    input EscreverMem
127  );
128
129    reg[7:0] memoria [255:0];
130    reg[7:0] lido;
131
132    always @(posedge clock)
133      begin
134        if(EscreverMem) begin
135          memoria[endereco] = dado;
136        end
137    end
138
139    always @(negedge clock)
140      begin
141        if(lerMem) begin
142        lido = memoria[endereco];
143        end
144     end
145
146      assign out = lido;
147  endmodule
```

2) **Configure simulações que demonstrem o correto funcionamento de todos os componentes implementados. Apresente o código fonte desse(s) módulo(s) de simulação.**

test_PC:

```
21  module test_PC;
22    reg clock;
23    reg [7:0] in;
24    reg Encerra;
25    wire [7:0] out;
26    initial begin
27    clock = 0; in = 0; Encerra=0;
28
29     #1 clock = 1;
30     in = 1;
31     Encerra = 0;
32
33     #1 clock = 0;
34     #1 clock = 1;
35     in = 0;
36     Encerra = 0;
37
38     #1 clock = 0;
39     #1 clock = 1;
40     in = 1;
41     Encerra = 1;
42
43    end
44    initial begin
45    $monitor("Time=%0d clock %d entrada %d saida %d encerra %d",
46    $time, clock, in, out, Encerra);
47    end
48    PC gate1(clock, in, out, Encerra);
49  endmodule
```

test_bancoInstrucoes:

```
76  module test_bancoInstrucoes;
77    reg clock;
78    reg [7:0] in;
79    wire [7:0] out;
80    initial begin
81    clock = 0; in = 0;
82
83     #1 clock = 1;
84     in = 1;
85
86     #1 clock = 0;
87     #1 clock = 1;
88     in = 0;
89
90    end
91    initial begin
92    $monitor("Time=%0d clock %d entrada %d saida %b",
93    $time, clock, in, out);
94    end
95    bancoInstrucoes gate1(clock, in, out);
96  endmodule
```

test_bancoRegs

```
module test_bancoRegs;
  reg clock;
  reg [1:0] in1;
  reg [1:0] in2;
  reg inec;
  reg [7:0] dado;
  reg EscreveReg;
  wire [7:0] out1;
  wire [7:0] out2;
  initial begin
  clock = 0; in1 = 0; in2=0; inec=0; dado=0; EscreveReg = 0;

   #1 clock = 1;
   in1 = 0;
   in2 = 1;
   inec = 1;
   dado = 0;
   EscreveReg = 1;

   #1 clock = 0;
   #1 clock = 1;
   in1 = 0;
   in2 = 1;
   inec = 0;
   dado = 1;
   EscreveReg = 1;

   #1 clock = 0;

  end
  initial begin
  $monitor("Time=%0d clock %d entrada1 %d entrada2 %d regs escrito %d dado %d EscreveReg %d, out1 %d, out2 %d",
  $time, clock, in1, in2, inec, dado, EscreveReg, out1, out2);
  end
  bancoRegs gate1(clock, in1, in2, inec, dado, out1, out2, EscreveReg);
endmodule
```

test_bancoMem

```
module test_bancoMem;
  reg clock;
  reg [7:0] endereco;
  reg [7:0] dado;
  reg lerMem, EscreverMem;
  wire [7:0] out;
  initial begin
  clock = 0; endereco = 0; dado = 0; lerMem = 0; EscreverMem = 0;

   #1 clock = 1;
   endereco = 1;
   dado = 1;
   EscreverMem = 1;
   lerMem = 0;

   #1 clock = 0;
   #1 clock = 1;
   endereco = 1;
   dado = 1;
   EscreverMem = 0;
   lerMem = 1;

   #1 clock = 0;

  end
  initial begin
  $monitor("Time=%0d clock %d endereco %d dado %d EscreveMem %d, lerMem %d, out %d",
  $time, clock, endereco, dado, EscreverMem, lerMem, out);
  end
  bancoMem gate1(clock, endereco, dado, out, lerMem, EscreverMem);
endmodule
```

3) **Demonstre e explique o funcionamento das simulações, usando fotos da tela (screenshots) da aplicação ModelSim em execução.**

teste_PC:

```
ModelSim> vsim -gui work.test_PC
# vsim -gui work.test_PC
# Loading work.test_PC
# Loading work.PC
VSIM 70> run
# Time=0 clock 0 entrada   0 saida   x encerra 0
# Time=1 clock 1 entrada   1 saida   1 encerra 0
# Time=2 clock 0 entrada   1 saida   1 encerra 0
# Time=3 clock 1 entrada   0 saida   0 encerra 0
# Time=4 clock 0 entrada   0 saida   0 encerra 0
# Time=5 clock 1 entrada   1 saida   0 encerra 1
```

Inicialmente, o *clock*, o *in* (entrada*)* e o *encerra* são iniciados com 0. O módulo PC está configurado para executar assim que houver uma virada de clock de 0 para 1 (borda de subida) e se o valor de *encerra* for igual a 0. Nesse sentido, na simulação, quando o clock passa a ser 1 pela primeira vez, a *entrada* é configurada como 1 e o *encerra* como 0, o resultado esperado para *out* (saída) é 1 (o que pode ser interpretado como a escrita de 1 em PC). Logo após, o clock reinicia, e é atribuído o valor de 0 para *in* (entrada*)* e *encerra*, o resultado esperado para *out* (saída) é 0 (o que pode ser interpretado como a escrita de 0 em PC). Por fim, para testar o *encerra*, na próxima virada de clock, o *in* (entrada) e o *encerra* são configurados como 1, assim, o resultado esperado para *out* (saída) é 0, visto que o *encerra = 1* pausa a escrita em PC, logo, o valor de PC continua com o valor anterior. Com isso, conseguimos perceber o funcionamento correto do *modulePC* no que se diz respeito à escrita em PC e ao encerramento (pausa) do módulo.

test_bancoInstrucoes:

```
VSIM 127> vsim -gui work.test_bancoInstrucoes
# vsim -gui work.test_bancoInstrucoes
# Loading work.test_bancoInstrucoes
# Loading work.bancoInstrucoes
VSIM 128> run
# Time=0 clock 0 entrada   0 saida xxxxxxxx
# Time=1 clock 1 entrada   1 saida 00101101
# Time=2 clock 0 entrada   1 saida 00101101
# Time=3 clock 1 entrada   0 saida 00010001
```

Inicialmente, o *clock* e o *in* (entrada) são iniciados com 0. O módulo *bancoInstrucoes* está configurado para executar assim que houver uma virada de *clock* de 0 para 1 (borda de subida). Nesse sentido, quando o *clock* passa a ser 1, no primeiro teste, o *in* (entrada) é configurado com o valor 1, assim, o resultado esperado em *out* (saída) é o valor da instrução 1 (00101101), que está previamente armazenado no banco de instruções. Logo após a virada do clock, o valor de *in* (entrada) é configurado como 0, o valor esperado em *out* (saída), é o valor da instrução 0 (00010001), que está previamente armazenada no banco de instruções. Com esses testes, conseguimos perceber o bom funcionamento do módulo *bancoInstrucoes*, que tem como saída os valores das instruções desejadas. Note que a alteração do valor em *out* (saída) só ocorre quando clock é igual a 1, isso acontece, como foi dito acima, porque o módulo do banco de instruções está configurado para executar assim que detectar uma borda de subida no clock.

test_bancoRegs:

```
# vsim -gui work.test_bancoRegs
# Loading work.test_bancoRegs
# Loading work.bancoRegs
VSIM 189> run
# Time=0 clock 0 entrada1 0 entrada2 0 regs escrito 0 dado   0 EscreveReg 0, out1   x, out2   x
# Time=1 clock 1 entrada1 0 entrada2 1 regs escrito 1 dado   0 EscreveReg 1, out1   x, out2   x
# Time=2 clock 0 entrada1 0 entrada2 1 regs escrito 1 dado   0 EscreveReg 1, out1   x, out2   0
# Time=3 clock 1 entrada1 0 entrada2 1 regs escrito 0 dado   1 EscreveReg 1, out1   x, out2   0
# Time=4 clock 0 entrada1 0 entrada2 1 regs escrito 0 dado   1 EscreveReg 1, out1   1, out2   0
```

Inicialmente, o *clock*, o *in1* (entrada 1), o *in2* (entrada 2), o *inec* (regs escrito), o *dado* (a ser escrito) e o *EscreveReg* são iniciados como 0. O módulo *bancoRegs* está configurado para escrever no registrador a ser escrito (*inec*) assim que o *clock* mudar de 0 para 1 (borda de subida) e o *EscreveReg* estiver configurado como 1. Nesse sentido, na primeira virada de clock (quando *clock* = 1), o *inec* (regs escrito) como 1, o *dado* continua como 0, e o *EscreveReg* é configurado como 1, o resultado esperado, para esse caso é a escrita do *dado* (0) na saída 1 (pela configuração do inec), só conseguimos visualizar esse resultado quando o *clock* volta a ser 0, porque o módulo *bancoRegs* foi configurado para levar os registradores para as saídas (*out1* e *out2*) na borda de descida, logo, quando o *clock* volta para 0, é possível perceber a alteração do valor de *out2* (saída 2), como 0. No próximo teste, quando o *clock* volta a ser 1, o *inec* (registrador a ser escrito) passa a ser 0, o *dado* passa a ser 1 e o *EscreveReg* continua como 1. Nesse caso, o resultado esperado é a escrita de 1 na saída 0, o que só pode ser observado quando o *clock* volta a ser 0, momento em que os registradores vão para a saída do módulo, percebe-se que, quando isso acontece, o valor de *out1* (saída 1), passa a ser 1, que é o valor do dado que foi escrito nele. Com isso, conseguimos visualizar o funcionamento correto do módulo *bancoRegs* , no que se diz respeito à escrita de dados nos registradores escolhidos e a saída dos dados dos registradores do módulo.

test_bancoMem:

```
VSIM 191> vsim -gui work.test_bancoMem
# vsim -gui work.test_bancoMem
# Loading work.test_bancoMem
# Loading work.bancoMem
VSIM 192> run
# Time=0 clock 0 endereco   0 dado   0 EscreveMem 0, lerMem 0, out   x
# Time=1 clock 1 endereco   1 dado   1 EscreveMem 1, lerMem 0, out   x
# Time=2 clock 0 endereco   1 dado   1 EscreveMem 1, lerMem 0, out   x
# Time=3 clock 1 endereco   1 dado   1 EscreveMem 0, lerMem 1, out   x
# Time=4 clock 0 endereco   1 dado   1 EscreveMem 0, lerMem 1, out   1
```

Inicialmente, o *clock,* o *endereco* , o *dado*, o *lerMem* e o *escreveMem* são iniciados com 0. O módulo *bancoMem* é configurado para escrever na memória assim que identificar uma virada de clock de 0 para 1 (borda de subida) e se o valor de *EscreveMem* for configurado como 1. Nesse sentido, no primeiro teste, assim que o *clock* assume o valor de 1, o *endereco*, o *dado*, o *EscreveMem* são configurados como 1 e o *LerMem* como 0, o resultado esperado para esse teste, é a escrita do *dado* (1) no *endereco* (1). Para validar se esse dado foi escrito, no próximo teste, o valor de *endereco* e *LerMem* foram configurados como 1, assim, o resultado esperado para *out* (saída) é o valor referente ao *endereco* (1) do *bancoMem*. Note que, só podemos observar essa saída quando o *clock* voltar a ser 0, porque o *bancoMem* foi configurado para enviar ao *out* (saída) assim que identificar uma virada de *clock* de 1 para 0 (borda de descida) e se o valor de *LerMem* for configurado como 1, logo, quando o *clock* volta a ser 0, podemos perceber a alteração do *out* (saída) para 1, validando a escrita e leitura corretas no *endereco* (1) do *bancoMem*. Com isso, podemos notar o funcionamento correto do módulo *bancoMem* no que se diz respeito à escrita e leitura de dados na memória.

Apêndice:

```
module PC (
  input wire clock,
  input wire [7:0] in,
  output wire [7:0] out,
  input Encerra
);

  reg [7:0] pc;

  always @(posedge clock)
    begin
    if(~Encerra) begin
      pc = in;
    end
  end

  assign out = pc;

endmodule

module test_PC;
  reg clock;
  reg [7:0] in;
  reg Encerra;
  wire [7:0] out;
  initial begin
  clock = 0; in = 0; Encerra=0;

   #1 clock = 1;
   in = 1;
   Encerra = 0;

   #1 clock = 0;
   #1 clock = 1;
   in = 0;
   Encerra = 0;

   #1 clock = 0;
   #1 clock = 1;
   in = 1;
   Encerra = 1;
```

```
  end
  initial begin
  $monitor("Time=%0d clock %d entrada %d saida %d encerra %d",
  $time, clock, in, out, Encerra);
  end
  PC gate1(clock, in, out, Encerra);
endmodule

module bancoInstrucoes (
  input wire clock,
  input wire [7:0] in,
  output wire [7:0] out
);

  reg [7:0] banco_instrucoes;

  always @(posedge clock)
    begin
      case (in)
      8'b00000000: banco_instrucoes = 8'b00010001;
      8'b00000001: banco_instrucoes = 8'b00101101;
      8'b00000010: banco_instrucoes = 8'b01110010;
      8'b00000011: banco_instrucoes = 8'b01010010;
      8'b00000100: banco_instrucoes = 8'b10010000;
      8'b00000101: banco_instrucoes = 8'b10100001;
      8'b00000110: banco_instrucoes = 8'b11000011;
              8'b00000111: banco_instrucoes = 8'b11100000;
      default: banco_instrucoes = 8'bxxxxxxxx;
      endcase
    end
  assign out = banco_instrucoes;
endmodule

module test_bancoInstrucoes;
  reg clock;
  reg [7:0] in;
  wire [7:0] out;
  initial begin
  clock = 0; in = 0;

    #1 clock = 1;
    in = 1;
```

```
  #1 clock = 0;
  #1 clock = 1;
  in = 0;

 end
 initial begin
 $monitor("Time=%0d clock %d entrada %d saida %b",
 $time, clock, in, out);
 end
 bancoInstrucoes gate1(clock, in, out);
endmodule

module bancoRegs (
 input wire clock,
 input wire [1:0] in1,
 input wire [1:0] in2,
 input wire inec,
 input wire [7:0] dado,
 output wire [7:0] out1,
 output wire [7:0] out2,
 input EscreveReg
);
 reg [7:0] regs [1:0];
 reg [7:0] valores [1:0];

 always @(posedge clock)
   begin
     if(EscreveReg) begin
       regs[inec] = dado;
     end
   end

  always @(negedge clock)
     begin
       valores [0] = regs [in1];
       valores [1] = regs [in2];
     end

   assign out1 = valores [0];
   assign out2 = valores [1];

endmodule

module test_bancoRegs;
```

```
  reg clock;
  reg [1:0] in1;
  reg [1:0] in2;
  reg inec;
  reg [7:0] dado;
  reg EscreveReg;
  wire [7:0] out1;
  wire [7:0] out2;
  initial begin
  clock = 0; in1 = 0; in2=0; inec=0; dado=0; EscreveReg = 0;

   #1 clock = 1;
   in1 = 0;
   in2 = 1;
   inec = 1;
   dado = 0;
   EscreveReg = 1;

   #1 clock = 0;
   #1 clock = 1;
   in1 = 0;
   in2 = 1;
   inec = 0;
   dado = 1;
   EscreveReg = 1;

   #1 clock = 0;

  end
  initial begin
   $monitor("Time=%0d clock %d entrada1 %d entrada2 %d regs escrito %d dado %d
EscreveReg %d, out1 %d, out2 %d",
  $time, clock, in1, in2, inec, dado, EscreveReg, out1, out2);
  end
  bancoRegs gate1(clock, in1, in2, inec, dado, out1, out2, EscreveReg);
endmodule

module bancoMem(
input wire clock,
  input wire [7:0] endereco,
  input wire [7:0] dado,
  output wire [7:0] out,
  input lerMem,
  input EscreverMem
```

```
);

  reg[7:0] memoria [255:0];
  reg[7:0] lido;

  always @(posedge clock)
    begin
      if(EscreverMem) begin
        memoria[endereco] = dado;
      end
  end

  always @(negedge clock)
    begin
      if(lerMem) begin
      lido = memoria[endereco];
      end
    end

    assign out = lido;
endmodule

module test_bancoMem;
  reg clock;
  reg [7:0] endereco;
  reg [7:0] dado;
  reg lerMem, EscreverMem;
  wire [7:0] out;
  initial begin
  clock = 0; endereco = 0; dado = 0; lerMem = 0; EscreverMem = 0;

    #1 clock = 1;
    endereco = 1;
    dado = 1;
    EscreverMem = 1;
    lerMem = 0;

    #1 clock = 0;
    #1 clock = 1;
    endereco = 1;
    dado = 1;
    EscreverMem = 0;
    lerMem = 1;
```

```
    #1 clock = 0;

  end
  initial begin
    $monitor("Time=%0d clock %d endereco %d dado %d EscreveMem %d, lerMem
%d, out %d",
  $time, clock, endereco, dado, EscreverMem, lerMem, out);
  end
  bancoMem gate1(clock, endereco, dado, out, lerMem, EscreverMem);
endmodule
```