Universidade Federal de São Carlos - UFSCar

Lorhan Sohaky de Oliveira Duda Kondo 740951

Experimento 05 - Implementação de um circuito sequencial utilizando Verilog

São Carlos - SP

2017

Universidade Federal de São Carlos - UFSCar

Lorhan Sohaky de Oliveira Duda Kondo 740951

Experimento 05 - Implementação de um circuito sequencial utilizando Verilog

Orientador: Fredy João Valente

Universidade Federal de São Carlos - UFSCar

Departamento de Computação

Ciência da Computação

Laboratório de Circuitos Digitais

São Carlos - SP

2017

Lista de ilustrações

Lista de tabelas

Lista de quadros

Lista de abreviaturas e siglas

FPGA	*Field Programmable Gate Array* - Arranjo de Portas Programáveis em Campo
LED	*Light Emitting Diode* - Diodo emissor de luz

Sumário

1 Resumo

A ideia deste experimento é implementar uma máquina de estado utilizando, a linguagem de descrição de *hardware*, Verilog. A máquina tenta representar uma situação do mundo real de um portão de garagem, conforme o cenário proposto abaixo:

> Considere o cenário de um controle para um portão de garagem. Em um estado inicial, o portão está fechado. Caso um acionador externo seja selecionado, o portão abre. Caso o portão esteja aberto e o acionador externo seja selecionado, o portão fecha. O portão nunca abre e fecha ao mesmo tempo. O trilho no qual o portão se desloca é equipado com dois sensores que indicam quando o portão está completamente aberto e quando está completamente fechado. O motor não deve tentar abrir o portão quando esse estiver aberto e nem deve fechá-lo quando este já estiver fechado.
>
> Para maior segurança dos usuários, o motor está equipado com um aviso luminoso que deve ser acesso quando o portão se desloca.
>
> Deve-se assumir que não é possível parar o portão enquanto ele estiver abrindo ou fechando, mas é possível que o usuário aperte o acionador externo enquanto o portão estiver se deslocando. Nesse caso, se o portão estiver abrindo, ele deve passar a fechar e vice-versa.

Para solucionar tal problema e facilitar sua resolução, dividiu-se o processo em três passos:

1. Desenhar a máquina de estado para o cenário em questão;
2. Escrever um código Verilog para a máquina de estado no passo anterior;
3. Executar o código na *Field Programmable Gate Array* - Arranjo de Portas Programáveis em Campo (FPGA) e simulação.

Além disso, escolheu-se uma máquina de estado qualquer para estudar como implementá-la em Verilog.

2 Descrição da execução do experimento

2.1 Passo 1 – Desenhar a máquina de estado

Com o problema em questão, tem-se em mente que trata-se de um portão de garagem que move-se horizontalmente, ou seja, da esquerda para direita e vise e versa, deste modo o portão abriria para a esquerda e fecharia para a direita. Assim, para a elaboração da máquina[1], considerou-se as entradas conforme o Quadro 1.

Quadro 1 – Lista das entradas da máquina de estado do problema da garagem.

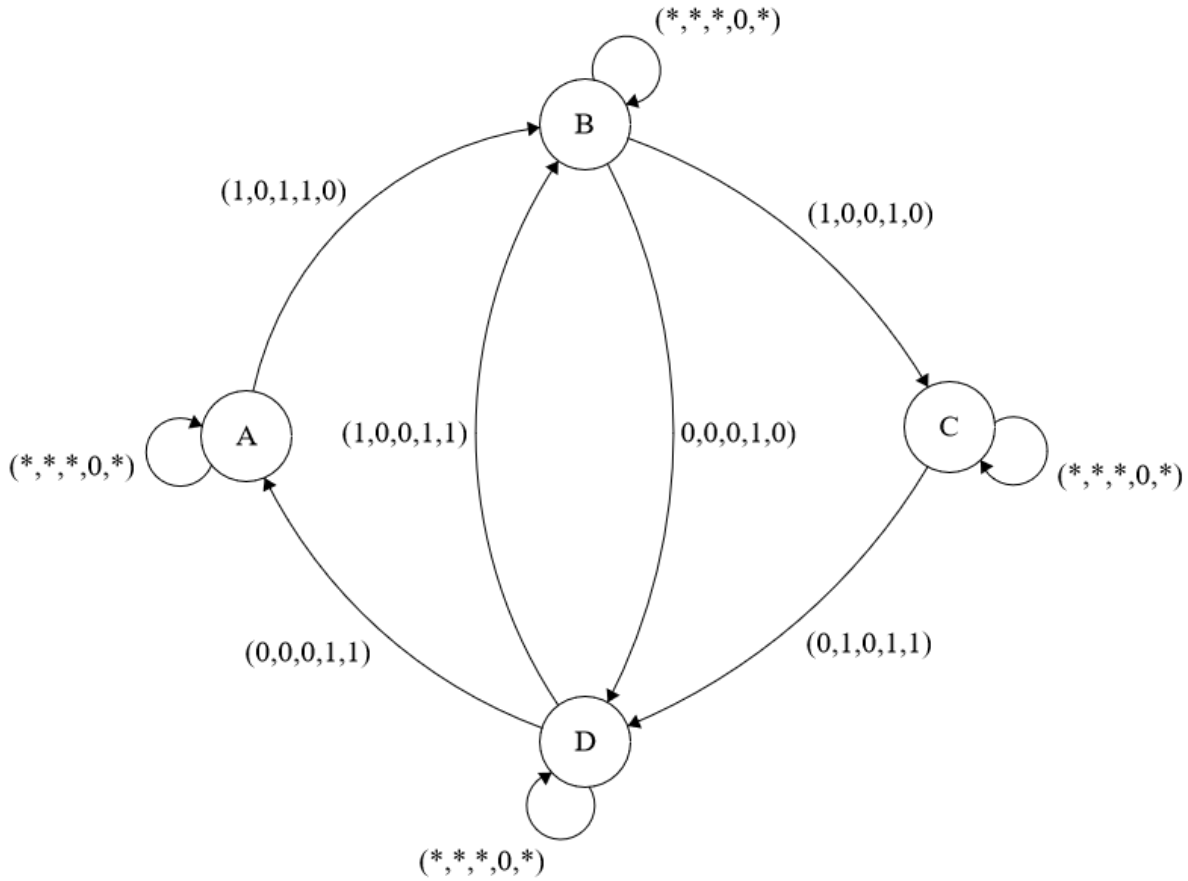
Entrada	**Valor lógico 0**	**Valor lógico 1**
Botao (b)	Abrir	Fechar
Aberto (a)	Não aberto	Totalmente aberto
Fechado (f)	Não fechado	Totalmente fechado
Motor (m)	Motor desligado	Motor ligado
Sentido (s)	Esquerda	Direita

Fonte: Próprio Autor

Com as entradas descritas no Quadro 1 elaborou-se a máquina conforme a Figura 1.

[1] Preferiu-se a utilização de uma máquina de Moore por haver conhecimento prévio deste tipo de máquina.

Figura 1 – Ilustração da máquina de estado do problema da garagem.



Nota: A entrada está está no formato (b, a, f, m, s). Os significados dos estados estão no Quadro 2.

Quadro 2 – Significados dos estados relacionando com os estados reais do problema proposto.

Estado	**Significado**
A	Totalmente fechado
B	Abrindo
C	Totalmente aberto
D	Fechando

Fonte: Próprio Autor

2.2 Passo 2 - Escrever um código Verilog para a máquina de estado

Para a realização deste passo, foram utilizados o programa Quartus 13.0 SP 1 e a placa FPGA Cyclone II - EP2C20F484C7.

Com a máquina de estado pronto, criou-se a código em Verilog, que no estado "totalmente aberto" apresenta A no *display*, no estado "totalmente fechado" apresenta

F no *display*, no estado "abrindo" acende um *Light Emitting Diode* - Diodo emissor de luz (LED) verde e no estado "fechando" acente um LED vermelho. Nos estados "abrindo" e "fechando" o *display* apresenta 0. O código da máquina de estado encontra-se no Código 2.1 e o resultado da compilação na Figura 2.

Código 2.1 – Código da máquina de estado do problema da garagem.

```
module inicial ( botao, aberto, fechado, motor, sentido, ledVerde,
   ledVermelho, display, clock );
  input botao, aberto, fechado, motor, sentido, clock;
  output ledVerde, ledVermelho;
  output [6:0] display;

  reg [1:0] estado;
  reg [4:0] entrada;

  reg [6:0] tmpDisplay;
  reg tmpLedVerde, tmpLedVermelho;

  parameter Fechado = 2'b00, Abrindo = 2'b01, Aberto = 2'b10, Fechando =
    2'b11;

  initial estado = Fechado;

  always @(posedge clock)begin
            entrada[4] = botao;
            entrada[3] = aberto;
            entrada[2] = fechado;
            entrada[1] = motor;
            entrada[0] = sentido;

            case( estado )
                Fechado: begin
                    tmpDisplay = 7'b0001110;
                    tmpLedVerde = 0;
                    tmpLedVermelho = 0;

                    if( entrada == 5'b10110 ) // botao = 1 & aberto = 0
  & fechado = 1 & motor = 1 & sentido = 0
                     estado = Abrindo;
                  end

                Abrindo: begin
                    tmpDisplay = 7'b1000000;
                    tmpLedVerde = 1;
                    tmpLedVermelho = 0;

                    if( entrada == 5'b10010 ) // botao = 1 & aberto = 0
```

```
    & fechado = 0 && motor = 1 & sentido = 0
                        estado = Aberto;
                      if( entrada == 5'b00010 ) // botao = 0 & aberto = 0
    & fechado = 0 & motor = 1 & sentido == 0
                        estado = Fechando;
                    end

                  Aberto: begin
                      tmpDisplay = 7'b0001000;
                      tmpLedVerde = 0;
                      tmpLedVermelho = 0;

                      if( entrada == 5'b01011 ) // botao = 0 & aberto = 1
    & fechado = 0 & motor = 1 & sentido = 1
                       estado = Fechando;
                    end

                  Fechando: begin
                      tmpDisplay = 7'b1000000;
                      tmpLedVerde = 0;
                      tmpLedVermelho = 1;

                      if( entrada == 5'b10011 ) // botao = 1 & aberto = 0
    & fechado = 0 & motor = 1 & sentido = 1
                        estado = Abrindo;
                      if( entrada == 5'b00011 ) // botao = 0 & aberto = 0
    & fechado = 0 & motor = 1 & sentido = 1
                        estado = Fechado;
                     end

                  default: estado = Fechado;

              endcase
  end

  assign display= tmpDisplay;
  assign ledVerde = tmpLedVerde;
  assign ledVermelho = tmpLedVermelho;

endmodule

module maquina( SW, LEDG, LEDR, HEX0, CLK );
  input [4:0] SW;
  input CLK;
  output [0:0] LEDG, LEDR;
  output [6:0] HEX0;
```

```
  inicial a( SW[4], SW[3], SW[2], SW[1], SW[0], LEDG[0], LEDR[0], HEX0,
   CLK);
endmodule
```

Nota: Para as entradas não mapeadas, a máquina mantêm-se no estado atual.

Figura 2 – Resultado da compilação do código da máquina de estado do problema da garagem.

Flow Summary

Flow Status	Successful - Thu Dec 07 16:56:53 2017
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	maquina
Top-level Entity Name	maquina
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	20 / 18,752 (< 1 %)
Total combinational functions	20 / 18,752 (< 1 %)
Dedicated logic registers	10 / 18,752 (< 1 %)
Total registers	10
Total pins	15 / 315 (5 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Para a realização dos testes, criou-se um *test bench* conforme o Código 2.2. Além disto, fez-se o *deploy* na FPGA[2].

Código 2.2 – Código de *test bench* da máquina do problema da garagem.

```
module maquina_TB;
  integer a;

  wire ledVerde, ledVermelho;
  wire [6:0] HEX;
  reg [4:0] val;
  reg clock;

  inicial i( a[4], a[3], a[2], a[1], a[0], ledVerde, ledVermelho, HEX,
   clock);
```

[2] Os resultados dos testes e simulações encontram-se na Avaliação dos resultados.

```
//Alterando clock
 always begin
    clock <= 0;
    #25;
    clock <= 1;
    #25;
 end

initial begin
  $display("\tSistema Porta\n");
  $display(" Estado  | Entrada | LG | LR | HEX");
  $display("-----------------------------------");

  //Estado Fechado
  #50
  a=0;#100 // a = 00000
  $write( " Fechado |  %x%x%x%x%x  |  %x |  %x |", a[4], a[3], a[2], a
 [1], a[0], ledVerde, ledVermelho);
  if( HEX == 7'b0001110 ) $display( "  F" ); if( HEX == 7'b1000000 )
 $display ("  0" );  if( HEX == 7'b0001000 ) $display( "  A" );

  //Estado Abrindo
  a=22;#100 // a = 10110
  $write( " Abrindo |  %x%x%x%x%x  |  %x |  %x |", a[4], a[3], a[2], a
 [1], a[0], ledVerde, ledVermelho);
  if( HEX == 7'b0001110 ) $display( "  F" ); if( HEX == 7'b1000000 )
 $display ("  0" ); if( HEX == 7'b0001000 ) $display( "  A" );

  //Estado Aberto
  a=18;#100 // a = 10010
  $write( "  Aberto |  %x%x%x%x%x  |  %x |  %x |", a[4], a[3], a[2], a
 [1], a[0], ledVerde, ledVermelho);
  if( HEX == 7'b0001110 ) $display( "  F" ); if( HEX == 7'b1000000 )
 $display ("  0" ); if( HEX == 7'b0001000 ) $display( "  A" );

  //Estado Fechando
  a=11;#100 // a = 01011
  $write( "Fechando |  %x%x%x%x%x  |  %x |  %x |", a[4], a[3], a[2], a
 [1], a[0], ledVerde, ledVermelho);
  if( HEX == 7'b0001110 ) $display( "  F" ); if( HEX == 7'b1000000 )
 $display ("  0" ); if( HEX == 7'b0001000 ) $display( "  A" );

  //Estado Fechado
  #50
  a=3;#100 // a = 00011
  $write( " Fechado |  %x%x%x%x%x  |  %x |  %x |", a[4], a[3], a[2], a
```

```
    [1], a[0], ledVerde, ledVermelho);
     if( HEX == 7'b0001110 ) $display( "  F" ); if( HEX == 7'b1000000 )
    $display ("  0" );  if( HEX == 7'b0001000 ) $display( "  A" );
  end
endmodule
```

3 Avaliação dos resultados do experimento

Apresentar os resultados da simulação em software e da utilização do Kit DE1 e/ou protoboard. Utilizar figuras, descrevê-las e discuti-las.

4 Análise crítica e discussão

O Código 2.1 engloba todos as entradas, todos os estados e suas saídas, assim, se fosse um circuito sequêncial, englobaria o circuito de excitação, circuito de memória e o circuito de saída.

Apresentar a visão do grupo sobre o experimento, apresentando pontos fáceis e de dificuldades para a realização do mesmo. Comente se os resultados obtidos representam o comportamento esperado do grupo para o circuito, fazendo relação com o conteúdo teórico.