



Universidade Federal de São Carlos - UFSCar

Lorhan Sohaky de Oliveira Duda Kondo 740951

Projeto Individual - Porta Giratória

São Carlos - SP

2017

Universidade Federal de São Carlos - UFSCar

Lorhan Sohaky de Oliveira Duda Kondo 740951

Projeto Individual - Porta Giratória

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

Figura 1 – Ilustração da máquina de estado do problema.	7
Figura 2 – Tabela das entradas das transições.	8
Figura 3 – Diagrama esquemático.	9
Figura 4 – Resultado da compilação do código da máquina de estado do problema.	12
Figura 5 – <i>Test bench Wave</i> do código da máquina.	14
Figura 6 – <i>Test bench Transcript</i> do código da máquina.	14
Figura 7 – Máquina no estado inicial.	15
Figura 8 – Transição do estado A para o B.	15
Figura 9 – Transição do estado B para o C.	16
Figura 10 – Transição do estado C para o D.	16
Figura 11 – Transição do estado D para o E.	17

Lista de quadros

Quadro 1 – Lista das entradas da máquina de estado do problema.	7
Quadro 2 – Significado dos estados relacionando com os estados reais do problema proposto.	8

Lista de abreviaturas e siglas

FPGA	<i>Field Programmable Gate Array</i> - Arranjo de Portas Programáveis em Campo
LED	<i>Light Emitting Diode</i> - Diodo emissor de luz

Sumário

1	RESUMO	6
2	DESCRIÇÃO DA EXECUÇÃO DO EXPERIMENTO	7
2.1	Passo 1 – Desenhar a máquina de estado e o diagrama esquemático	7
2.2	Passo 2 - Escrever um código Verilog para a máquina de estado . .	9
3	AVALIAÇÃO DOS RESULTADOS DO EXPERIMENTO	14
4	ANÁLISE CRÍTICA E DISCUSSÃO	18

1 Resumo

O experimento tem como objetivo implementar uma máquina de estado que corresponda ao funcionamento de uma porta giratório de banco. Conforme a seguinte descrição:

Considere uma porta giratória integrada com detector de metais, utilizada em bancos e desenvolva um projeto de um circuito Lógico Digital para Controle da porta utilizando FSM (máquina de estados finitos). A porta possui sensores de detecção de sentido de giro, detecção de presença de pessoas, sensor detecção de presença de metais para quem estiver entrando, sinalizador luminoso para luz verde (seguir em frente) e vermelho (retornar e colocar metais na janela lateral), além de mensagem sonora indicando a necessidade de voltar e retirar metais dos bolsos e colocar na janela lateral. Considere as seguintes características operacionais: A porta é bidirecional: 1. Quando uma pessoa adentrar a porta giratória para sair ou para entrar deve ter o sensor de presença detecta a presença de pessoa no lado correspondente e acende uma luz verde para sinalizar o “vá em frente”; 2. Se houver pessoas de ambos lados a preferência será para quem estiver saindo, assim deverá ser sinalizado verde de um lado (saída) e vermelho do lado entrada; 3. A detecção de metais ocorrerá somente para quem estiver entrando, e em caso positivo, a porta deverá acender a luz vermelha e disparar mensagem audível para que a pessoa retorne e coloque metais na janela lateral; 4. A porta deverá permanecer travada enquanto houver 2 pessoas simultâneas tentando entrar / sair, até que a pessoa entrando recue para aquela que estiver saindo sair antes dela.

Para desenvolver tal projeto, dividiu-se a tarefa em etapas:

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.

2 Descrição da execução do experimento

2.1 Passo 1 – Desenhar a máquina de estado e o diagrama esquemático

Com o problema em questão, tem-se em mente que trata-se de uma porta giratória que gira no sentido anti-horário. Assim, para a elaboração da máquina¹, considerou-se as entradas conforme o [Quadro 1](#).

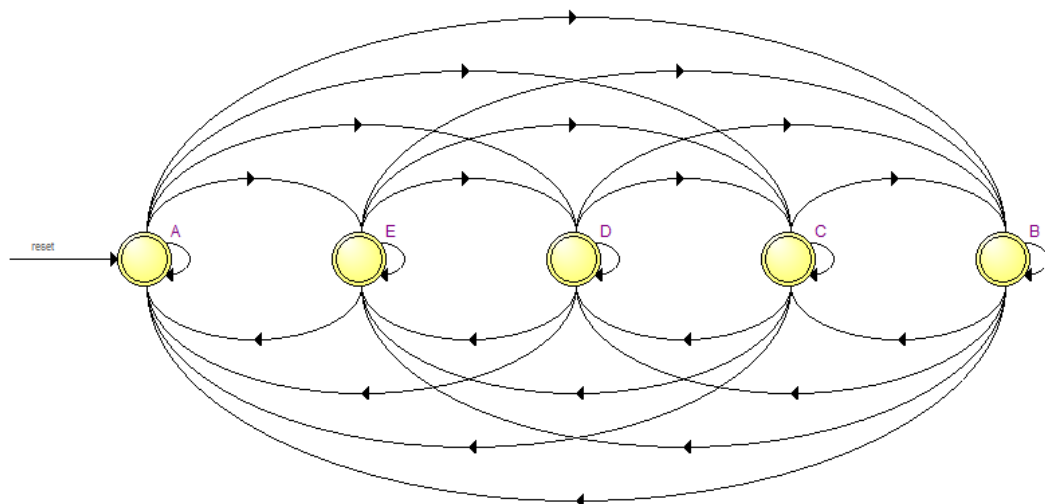
Quadro 1 – Lista das entradas da máquina de estado do problema.

Entrada	Valor lógico 0	Valor lógico 1
Giro	Parado ou sentido horário	Sentido Anti-horário
Entrada	Não há ninguém entrando	Há alguém entrando
Saida	Não há ninguém saindo	Há alguém entrando
Metais	Não há metais na entrada	Há metais na entrada

Fonte: Próprio Autor

Com as entradas descritas no [Quadro 1](#) elaborou-se a máquina conforme a [Figura 1](#).

Figura 1 – Ilustração da máquina de estado do problema.



Nota: A entrada está descrita na [Figura 2](#). O significado dos estados estão no [Quadro 2](#).

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

Figura 2 – Tabela das entradas das transições.

State Table			
	Source State	Destination State	Condition
1	A	B	(!metais).(!saida).(entrada).(giro)
2	A	C	(metais).(entrada).(giro)
3	A	A	(!metais).(!saida).(!entrada) + (!metais).(!saida).(entrada).(!giro) + (!metais).(saida).(!giro) + (metais).(!entrada) + (metais).(entrada).(!giro)
4	A	D	(!metais).(saida).(entrada).(giro)
5	A	E	(!metais).(saida).(!entrada).(giro)
6	B	B	(!metais).(!saida).(entrada) + (!metais).(saida).(!giro) + (metais).(!saida).(!entrada) + (metais).(!saida).(entrada).(!giro) + (metais).(saida).(!giro)
7	B	C	(metais).(!saida).(entrada).(giro)
8	B	A	(!metais).(!saida).(!entrada)
9	B	D	(saida).(entrada).(giro)
10	B	E	(saida).(!entrada).(giro)
11	C	B	(!metais).(!saida).(entrada)
12	C	C	(!metais).(saida).(!entrada).(!giro) + (metais)
13	C	A	(!metais).(!saida).(!entrada)
14	C	D	(!metais).(saida).(entrada)
15	C	E	(!metais).(saida).(!entrada).(giro)
16	D	B	(!metais).(!saida).(entrada).(giro)
17	D	C	(metais).(saida).(entrada)
18	D	A	(!metais).(!saida).(!entrada)
19	D	D	(!metais).(!saida).(entrada).(!giro) + (!metais).(saida).(!entrada).(!giro) + (!metais).(saida).(entrada) + (metais).(!saida) + (metais).(saida).(!entrada)
20	D	E	(!metais).(saida).(!entrada).(giro)
21	E	B	(!metais).(!saida).(entrada).(giro)
22	E	C	(metais).(!saida).(entrada).(giro)
23	E	A	(!metais).(!saida).(!entrada)
24	E	D	(!metais).(saida).(entrada)
25	E	E	(!metais).(!saida).(entrada).(!giro) + (!metais).(saida).(!entrada) + (metais).(!saida).(!entrada) + (metais).(!saida).(entrada).(!giro) + (metais).(saida)

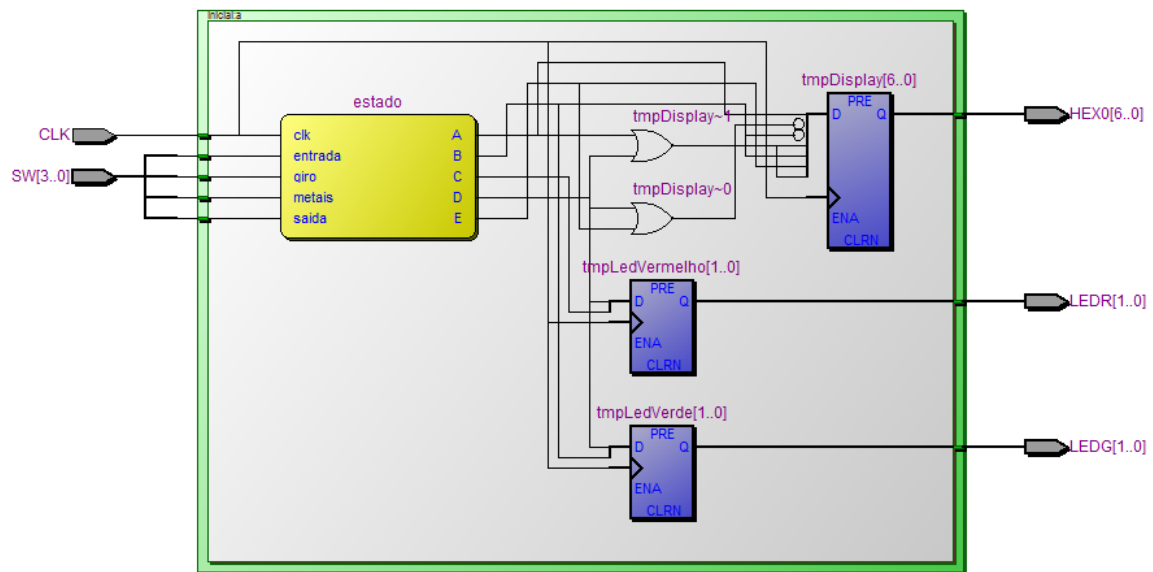
Quadro 2 – Significado dos estados relacionando com os estados reais do problema proposto.

Estado	Significado
A	Inicial
B	Entrando
C	Detectou metal
D	Duas pessoas
E	Saindo

Fonte: Próprio Autor

Por fim desenh-se o diagrama esquemático conforme a [Figura 3](#).

Figura 3 – Diagrama esquemático.



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 pronta, criou-se o código em Verilog, que encontra-se no [Código 2.1](#) e o resultado da compilação na [Figura 4](#).

Código 2.1 – Código da máquina de estado do problema.

```
module inicial ( giro, entrada, saida, metais, ledVerde, ledVermelho,
    display, clock );
input  giro, entrada, saida, metais, clock;
output [1:0] ledVerde, ledVermelho;
output [6:0] display;

reg [2:0] estado;
reg [3:0] tmp;
reg [1:0] tmpLedVerde, tmpLedVermelho;
reg [6:0] tmpDisplay;

parameter A = 3'b000, B = 3'b001, C = 3'b010, D = 3'b011, E = 3'b100;

initial estado = A;

always @( posedge clock ) begin
    tmp = { giro, entrada, saida, metais }; // Equivalente ao código
    comentado abaixo
    /*tmp[3] = giro;
```

```
tmp[2] = entrada;
tmp[1] = saida;
tmp[0] = metaís;*/

case( estado )
  A: begin
    tmpLedVerde = 2'b00;
    tmpLedVermelho = 2'b00;
    tmpDisplay = 7'b1111001;

    if( tmp == 4'b1100 )
      estado = B;
    if( tmp == 4'b1101 | tmp == 4'b1111 )
      estado = C;
    if( tmp == 4'b1110 )
      estado = D;
    if( tmp == 4'b1010 )
      estado = E;

  end

  B: begin
    tmpLedVerde = 2'b01;
    tmpLedVermelho = 2'b00;
    tmpDisplay = 7'b0100100;

    if( tmp == 4'b0000 | tmp == 4'b1000 )
      estado = A;
    if( tmp == 4'b1101 )
      estado = C;
    if( tmp == 4'b1110 | tmp == 4'b1111 )
      estado = D;
    if( tmp == 4'b1010 | tmp == 4'b1011 )
      estado = E;

  end

  C: begin
    tmpLedVerde = 2'b00;
    tmpLedVermelho = 2'b01;
    tmpDisplay = 7'b0110000;

    if( tmp == 4'b0100 | tmp == 4'b1100 )
      estado = B;
    if( tmp == 4'b0000 | tmp == 4'b1000 )
      estado = A;
    if( tmp == 4'b0110 | tmp == 4'b1110 )
```

```
        estado = D;
    if( tmp == 4'b1010 )
        estado = E;

end

D: begin
    tmpLedVerde = 2'b10;
    tmpLedVermelho = 2'b10;
    tmpDisplay = 7'b0011001;

    if( tmp == 4'b0111 | tmp == 4'b1111 )
        estado = C;
    if( tmp == 4'b1100 )
        estado = B;
    if( tmp == 4'b1010 )
        estado = E;
    if( tmp == 4'b0000 | tmp == 4'b1000 )
        estado = A;

end

E: begin
    tmpLedVerde = 2'b00;
    tmpLedVermelho = 2'b00;
    tmpDisplay = 7'b0010010;

    if( tmp == 4'b0000 | tmp == 4'b1000 )
        estado = A;
    if( tmp == 4'b0110 | tmp == 4'b1110 )
        estado = D;
    if( tmp == 4'b1100 )
        estado = B;
    if( tmp == 4'b1101 )
        estado = C;

end

    default: estado = A;

endcase

end

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

```

endmodule

module projetoPessoal ( SW,LEDG, LEDR, HEX0, CLK );
  input [3:0] SW;
  input CLK;
  output [1:0] LEDG, LEDR;
  output [6:0] HEX0;

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

```

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

Figura 4 – Resultado da compilação do código da máquina de estado do problema.

Flow Summary	
Flow Status	Successful - Tue Dec 19 00:38:00 2017
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	projetoPessoal
Top-level Entity Name	projetoPessoal
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	39 / 18,752 (< 1 %)
Total combinational functions	37 / 18,752 (< 1 %)
Dedicated logic registers	16 / 18,752 (< 1 %)
Total registers	16
Total pins	16 / 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 FPGA².

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

```

module projetoPessoal_TB;

  integer a;
  wire [1:0] ledVerde, ledVermelho;
  wire [6:0] display;
  reg clock;

  inicial i( a[3], a[2], a[1], a[0], ledVerde, ledVermelho, display,
    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("\tPorta Giratorias\n");
    $display(" Estado | Entrada | LG | LR");
    $display("-----");

    //Estado A
    a = 0; #50 // a = 0000
    $display( "      A      |   %x%x%x%x   | %x%x |   %x%x", a[3], a[2], a[1],
a[0], ledVerde[1], ledVerde[0], ledVermelho[1], ledVermelho[0] );

    //Estado B
    a = 12; #100 // a = 1100
    $display( "      B      |   %x%x%x%x   | %x%x |   %x%x", a[3], a[2], a[1],
a[0], ledVerde[1], ledVerde[0], ledVermelho[1], ledVermelho[0] );

    //Estado C
    a = 13; #100 // a = 1101
    $display( "      C      |   %x%x%x%x   | %x%x |   %x%x", a[3], a[2], a[1],
a[0], ledVerde[1], ledVerde[0], ledVermelho[1], ledVermelho[0] );

    //Estado D
    a = 6; #100 // a = 0110
    $display( "      D      |   %x%x%x%x   | %x%x |   %x%x", a[3], a[2], a[1],
a[0], ledVerde[1], ledVerde[0], ledVermelho[1], ledVermelho[0] );

    //Estado E
    a = 10; #100 // a = 1010
    $display( "      E      |   %x%x%x%x   | %x%x |   %x%x", a[3], a[2], a[1],
a[0], ledVerde[1], ledVerde[0], ledVermelho[1], ledVermelho[0] );

    //Estado A
    a = 8; #100 // a = 1000
    $display( "      A      |   %x%x%x%x   | %x%x |   %x%x", a[3], a[2], a[1],
a[0], ledVerde[1], ledVerde[0], ledVermelho[1], ledVermelho[0] );
end
endmodule

```

3 Avaliação dos resultados do experimento

A entrada formada por (giro, entrada, saída, metais) é correspondida respectivamente pelas entradas SW[3], SW[2], SW[1] e SW[0]. Para o estado B (entrando) acende o *Light Emitting Diode* - Diodo emissor de luz (LED) verde (LEDG[0]), para o estado de C (Detectou metal) acende o LED vermelho (LEDR[0]) e para o estado de D (Duas pessoas) acende o LED verde (LEDG[1]) e LED vermelho (LEDR[1]). Assim, ao executar o *test bench* obteve-se os resultados conforme as Figura 5 e Figura 6.

Figura 5 – *Test bench Wave* do código da máquina.

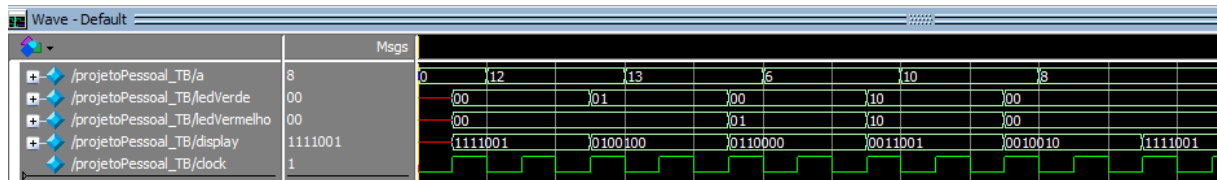


Figura 6 – *Test bench Transcript* do código da máquina.

```
#          Porta Giratorias
#
# Estado | Entrada | LG | LR
# -----
#      A | 0000 | 00 | 00
#      B | 1100 | 01 | 00
#      C | 1101 | 00 | 01
#      D | 0110 | 10 | 10
#      E | 1010 | 00 | 00
#      A | 1000 | 00 | 00
```

Como resultado do *deploy* na placa, obteve o resultado conforme as Figura 7, Figura 8, Figura 9, Figura 10 e Figura 11.

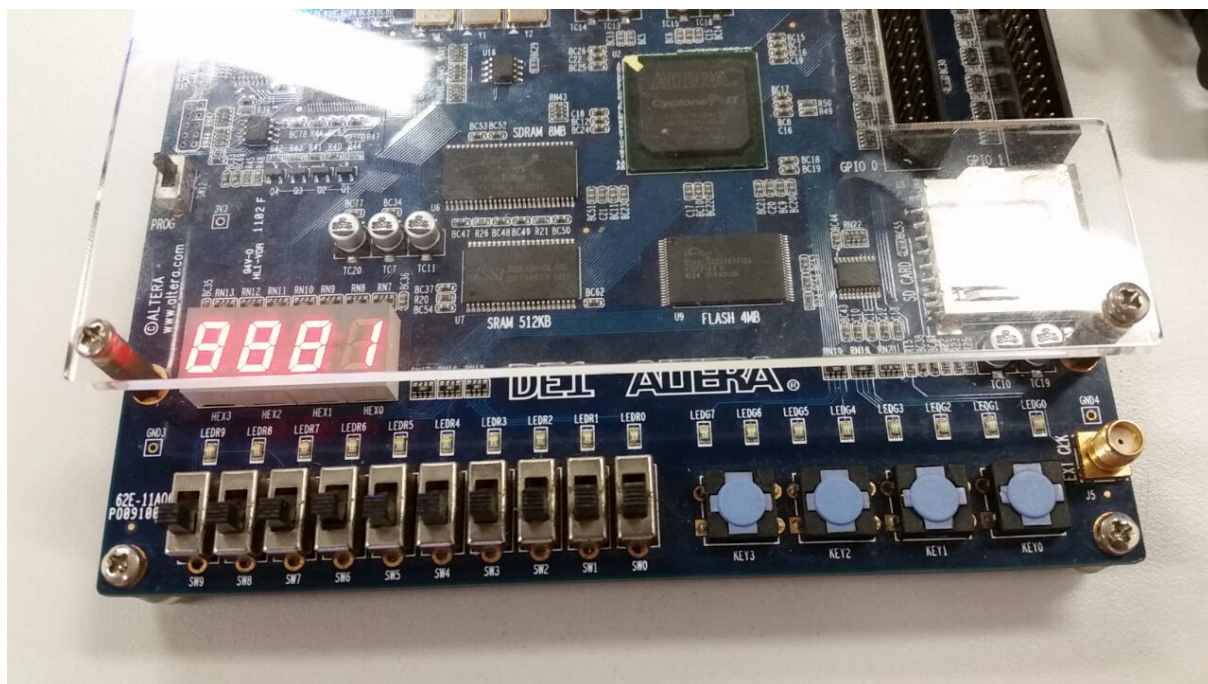


Figura 7 – Máquina no estado inicial.

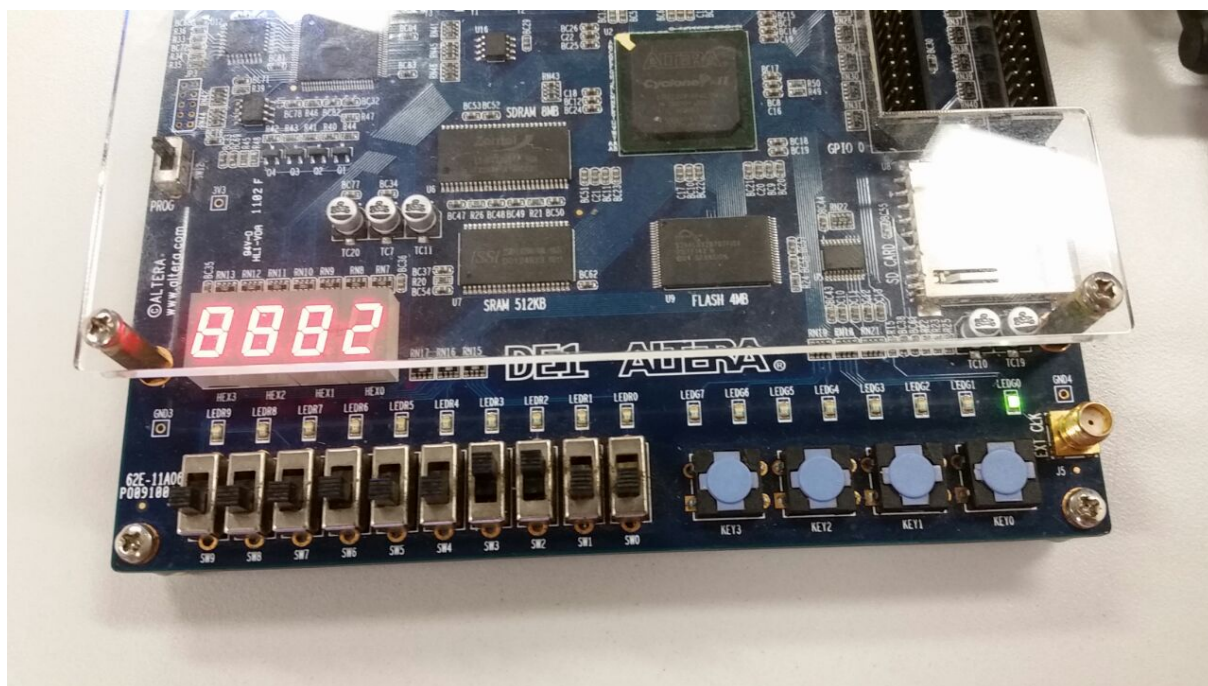


Figura 8 – Transição do estado A para o B.

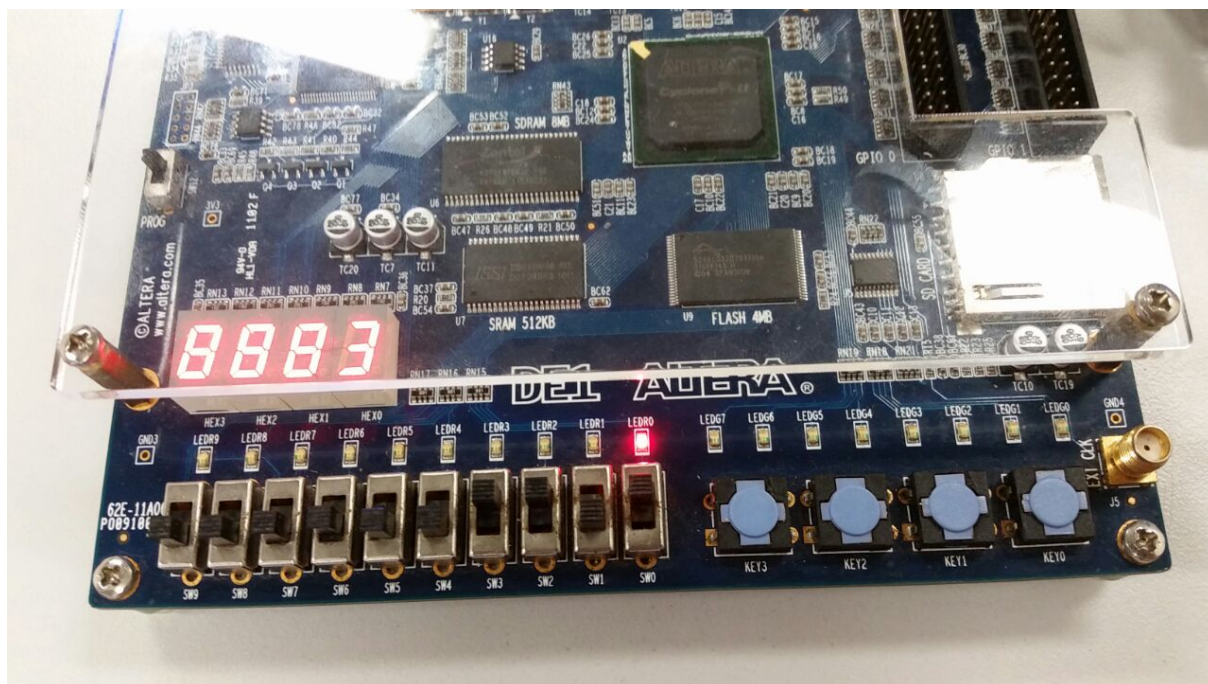


Figura 9 – Transição do estado B para o C.

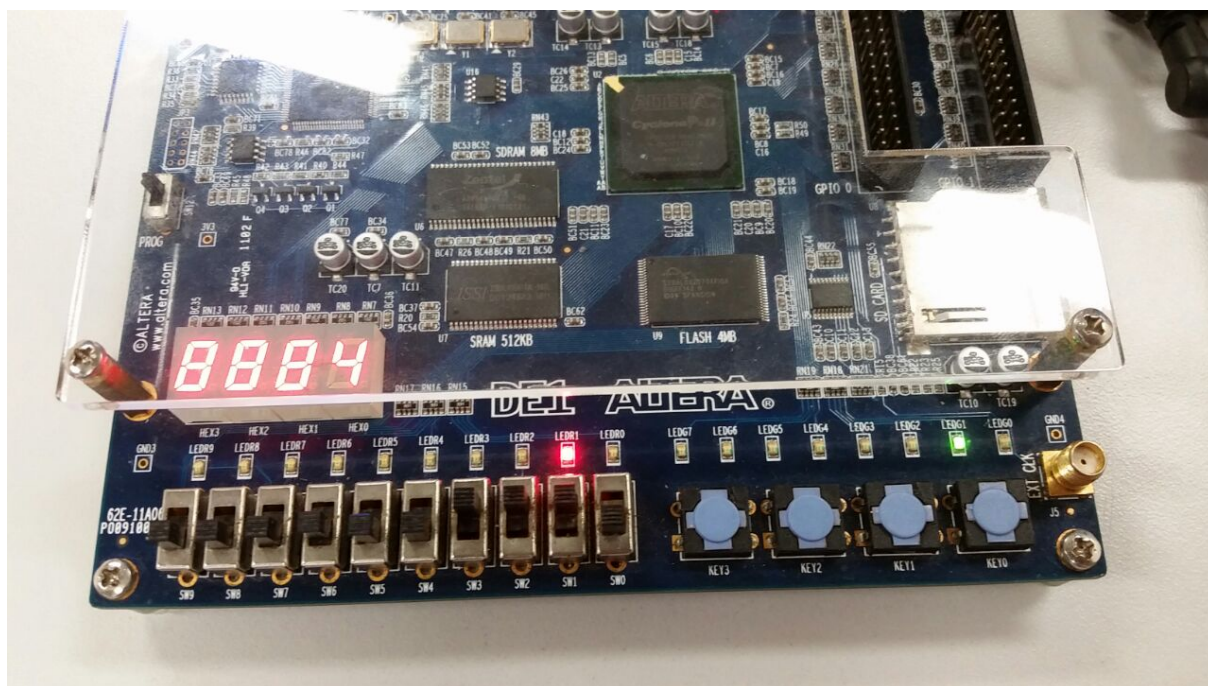


Figura 10 – Transição do estado C para o D.

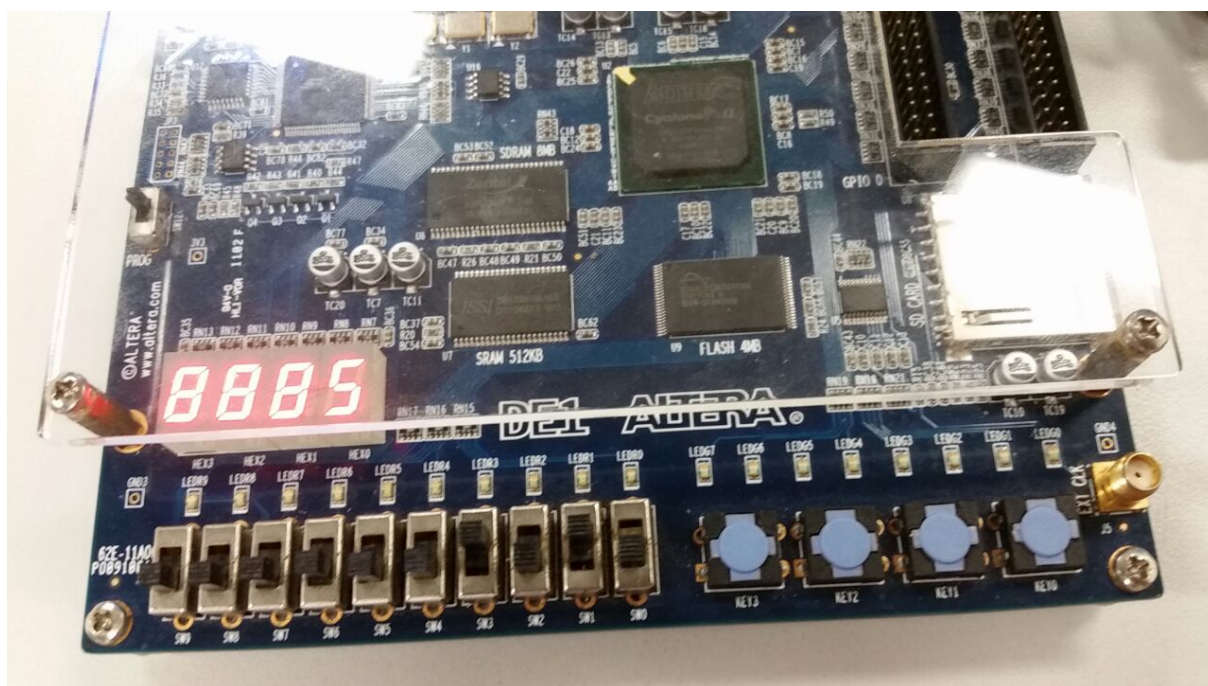


Figura 11 – Transição do estado D para o E.

4 Análise crítica e discussão

O [Código 2.1](#) engloba todas as entradas, todos os estados e suas saídas, assim, se fosse um circuito sequencial, englobaria o circuito de excitação, circuito de memória e o circuito de saída, ou seja, não sendo necessário criar um módulo para cada parte da máquina.

Constatou-se que para as entradas que não aparecem no [Código 2.1](#), a máquina mantém-se no estado atual, isso era o esperado para uma máquina deste tipo.

Teve-se dificuldade de executar o código na placa, pois mesmo com os *test bench* funcionando corretamente, os LEDs não acendiam. Notou-se que quando modificava a saída do display, os LEDs funcionavam, então exibiu-se números no *display* de acordo com o estado da máquina, o que também facilitou a identificação do estado em que a máquina se encontrava. Sento 1 para o estado A, 2 para B, 3 para C, 4 para D e 5 para E.