Universidade Federal de São Carlos - UFSCar

Joao Vitor Azevedo Marciano 743554

Experimento de Avaliação Individual – Circuito Sequencial para Porta Giratória de um Banco

São Carlos - SP

2017

Universidade Federal de São Carlos - UFSCar

Joao Vitor Azevedo Marciano 743554

Experimento de Avaliação Individual – Circuito Sequencial para Porta Giratória de um Banco

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 uma porta giratória de um estabelecimento bancário.

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 saíndo sair antes dela.

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 uma porta giratória que geralmente gira em um único sentido, ou seja, permitindo entrada e saída. Assim, para a elaboração da máquina[1], considerou-se as entradas conforme código Verilog.

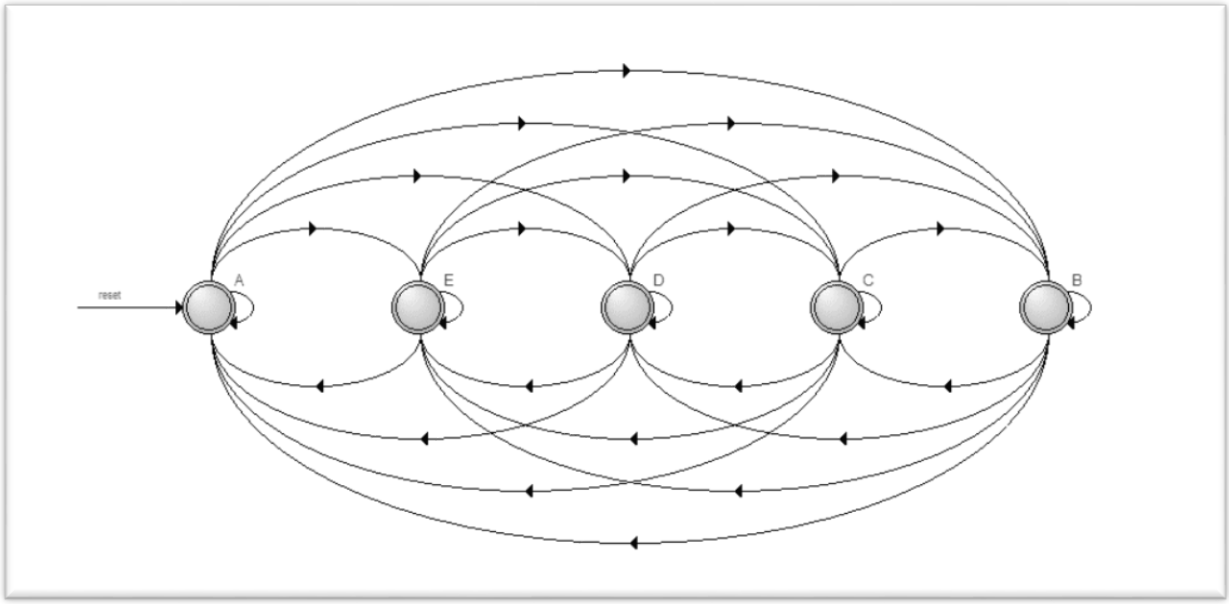
Quadro 1 – Lista das entradas da máquina de estado do problema.

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

Com as entradas descritas no código Verilog, 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 porta



É importante destacar que este desenho digital foi gerado por uma ferramenta externa ao Quartus, da Altera, e por este motivo encontra-se sem o detalhamento das entradas. Essa imagem é gerada a partir do código Verilog, e este sim detalha exatamente as entradas necessárias para a transição entre os estados. Os detalhes sobre os estados podem ser encontrados nas figuras abaixo.

Figura 2 – Tabela de Estados gerada a partir do código Verilog:

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

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

Houve uma inversão dos passos, e foi escolhido fazer primeiro o código Verilog, para a partir dele gerar a máquina de estados.

Abaixo segue o código Verilog desenvolvido para este Experimento. 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 seguinrw e o resultado da compilação na Figura 3.

```
module inicial ( giro, entrada, saida, decMetais, ledVerde, ledVermelho, display, clock );
input giro, entrada, saida, decMetais, 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, decMetais }; // Equivamente ao código comentado abaixo
/*tmp[3] = giro;
tmp[2] = entrada;
tmp[1] = saida;
tmp[0] = decMetais;*/

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 avaliacaoIndividual ( 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
```


Figura 3 – Resultado da compilação do código da máquina de estado do problema da porta.

Flow Summary	
Flow Status	Successful - Thu Dec 21 19:16:34 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.

```
module avaliacaoIndividual _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 );

//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
```

Figura 4 - Diagrama esquematico, desenhado usando outra funcionalidade interna do Quartus

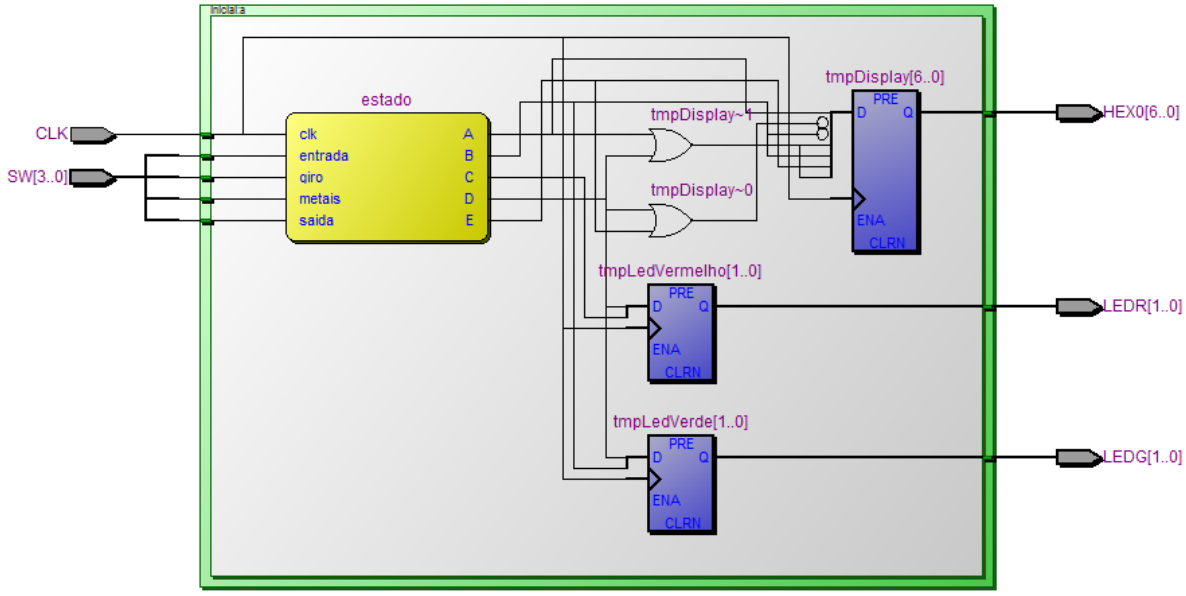
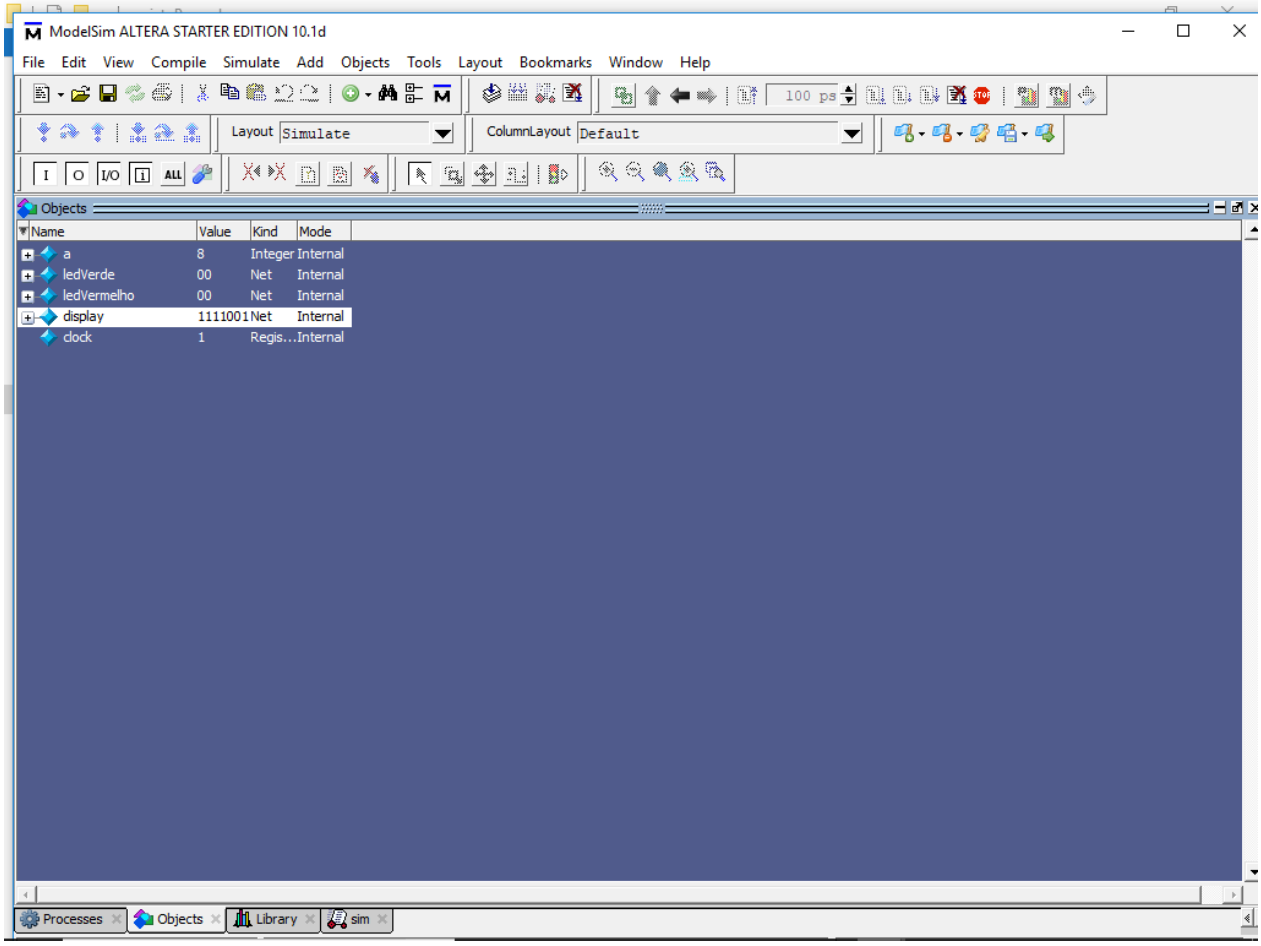


Figura 5 - Transcript

```
#       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
```

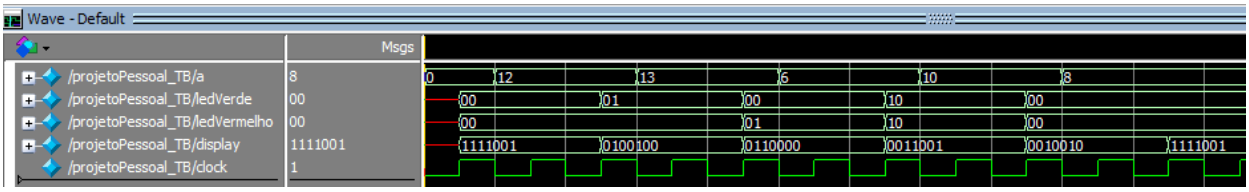
Figura 6 - Detalhamento dos objetos na simulação

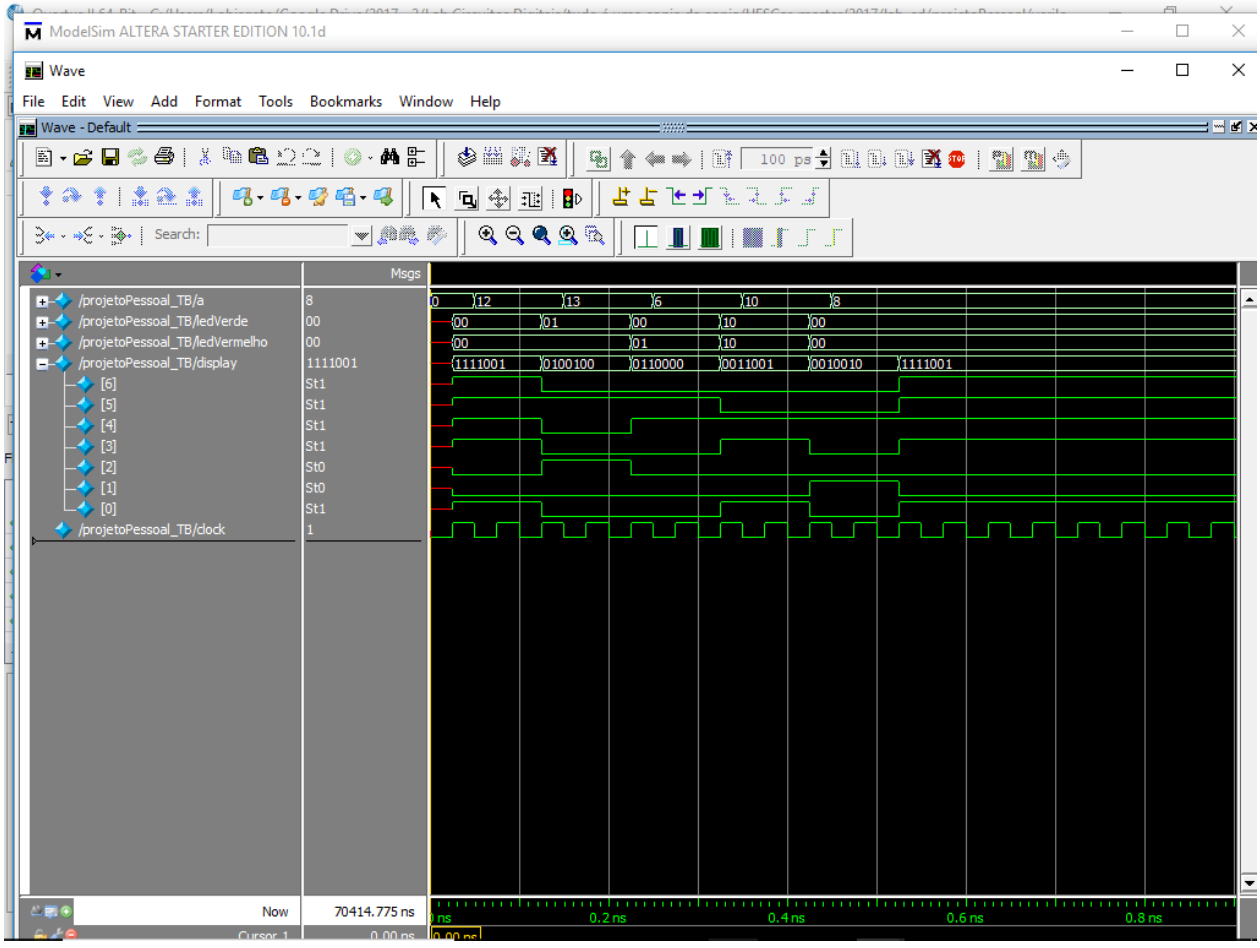


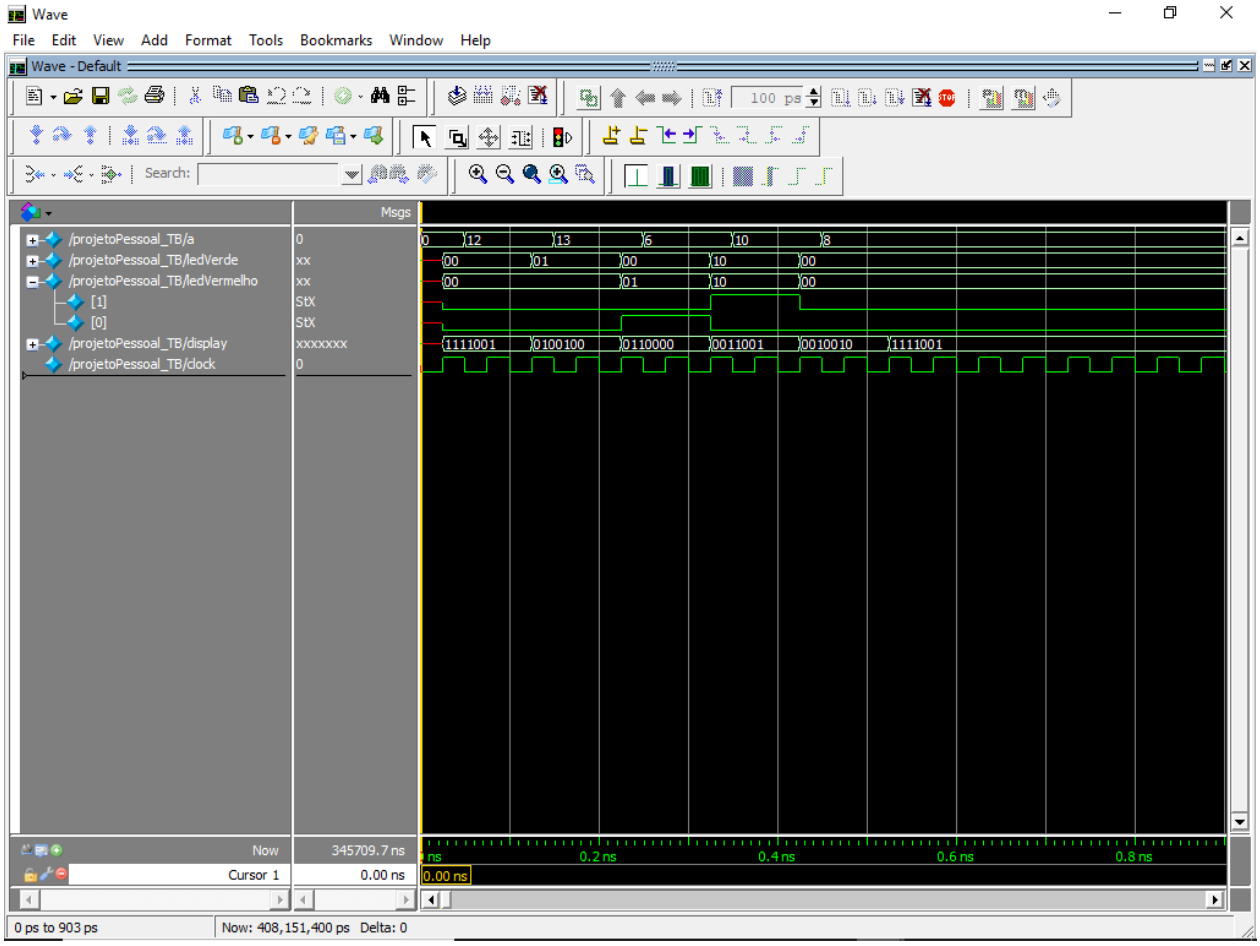
3 Avaliação dos resultados do experimento

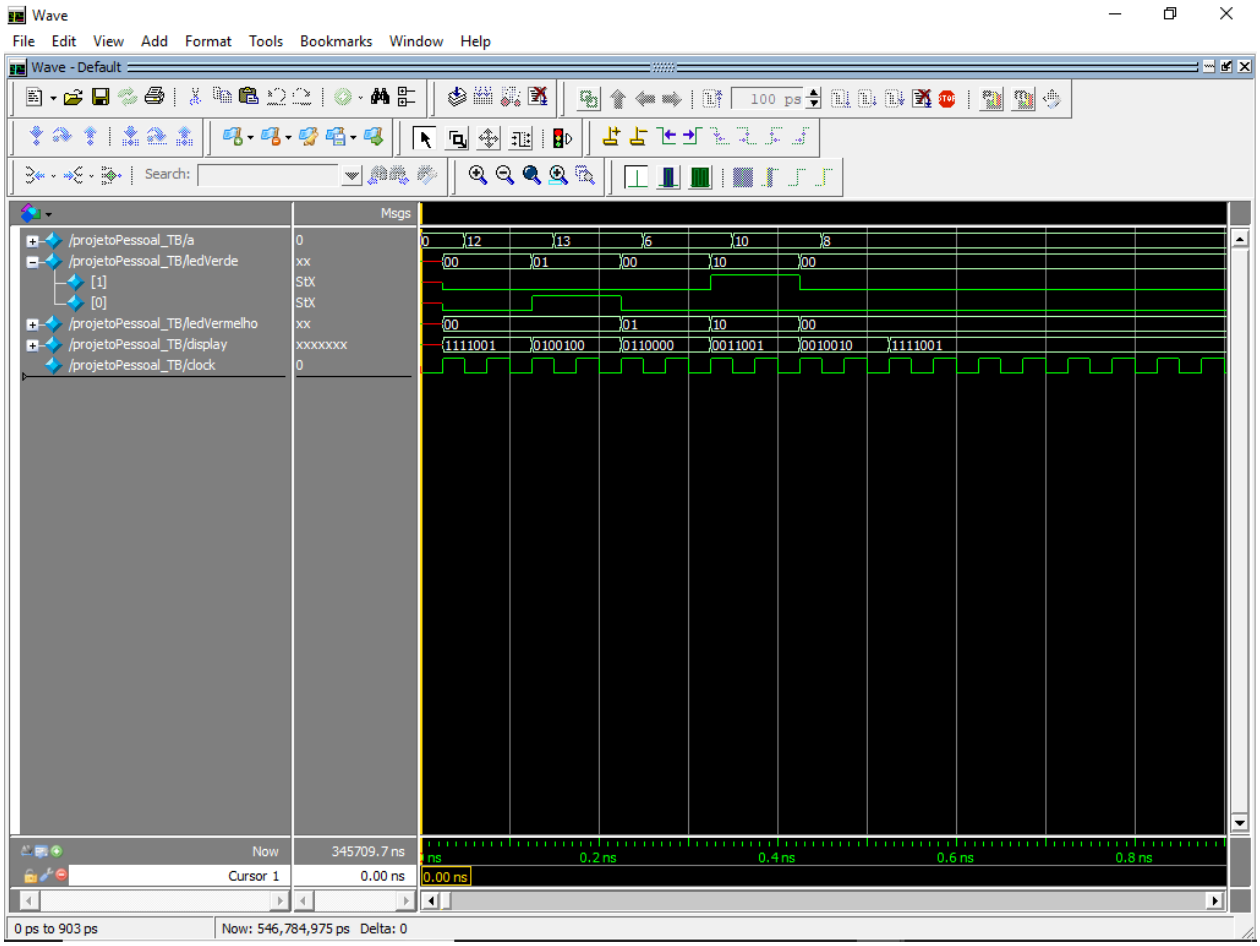
Resultados da Simulação RTL

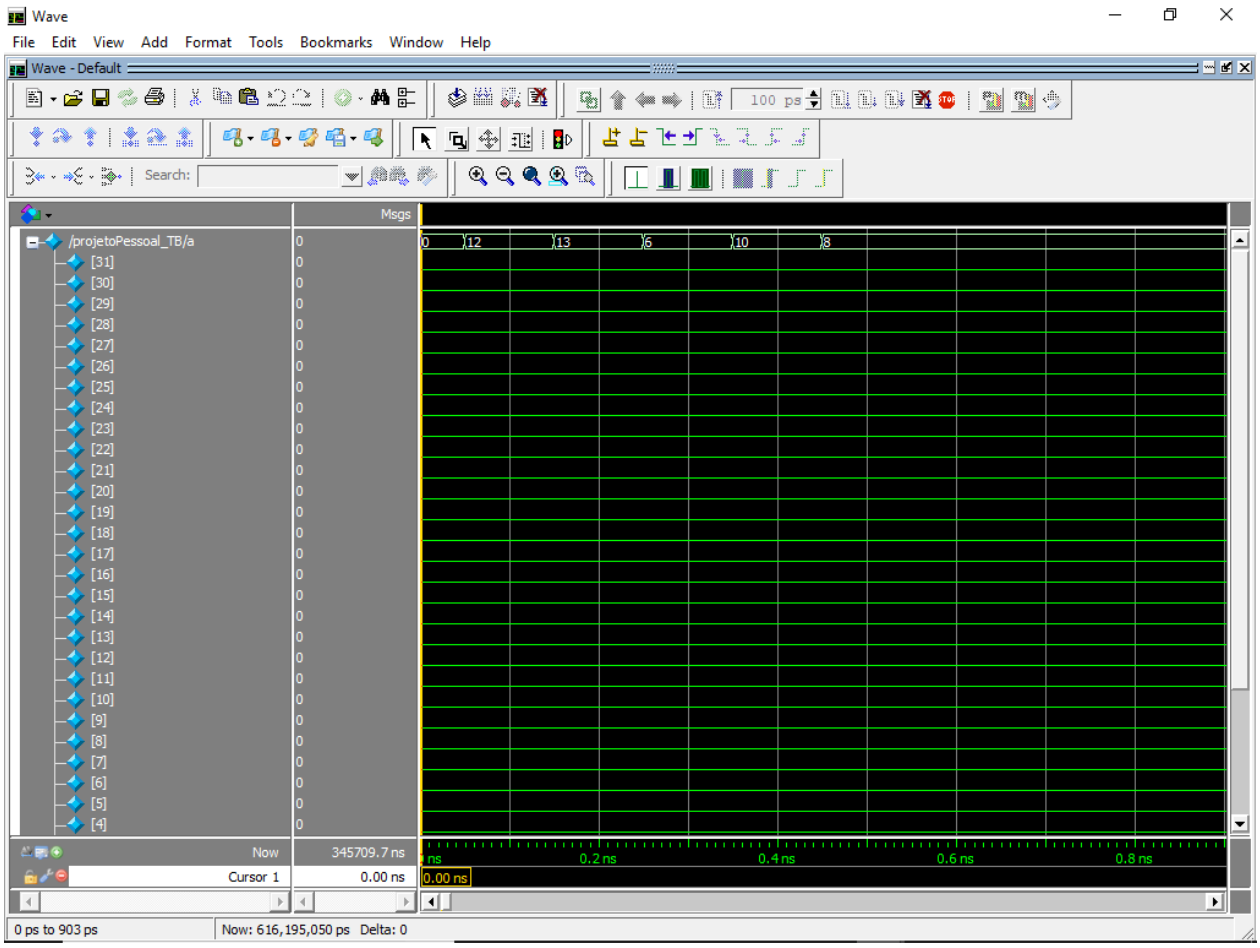
Figuras 7,8,9,10,11





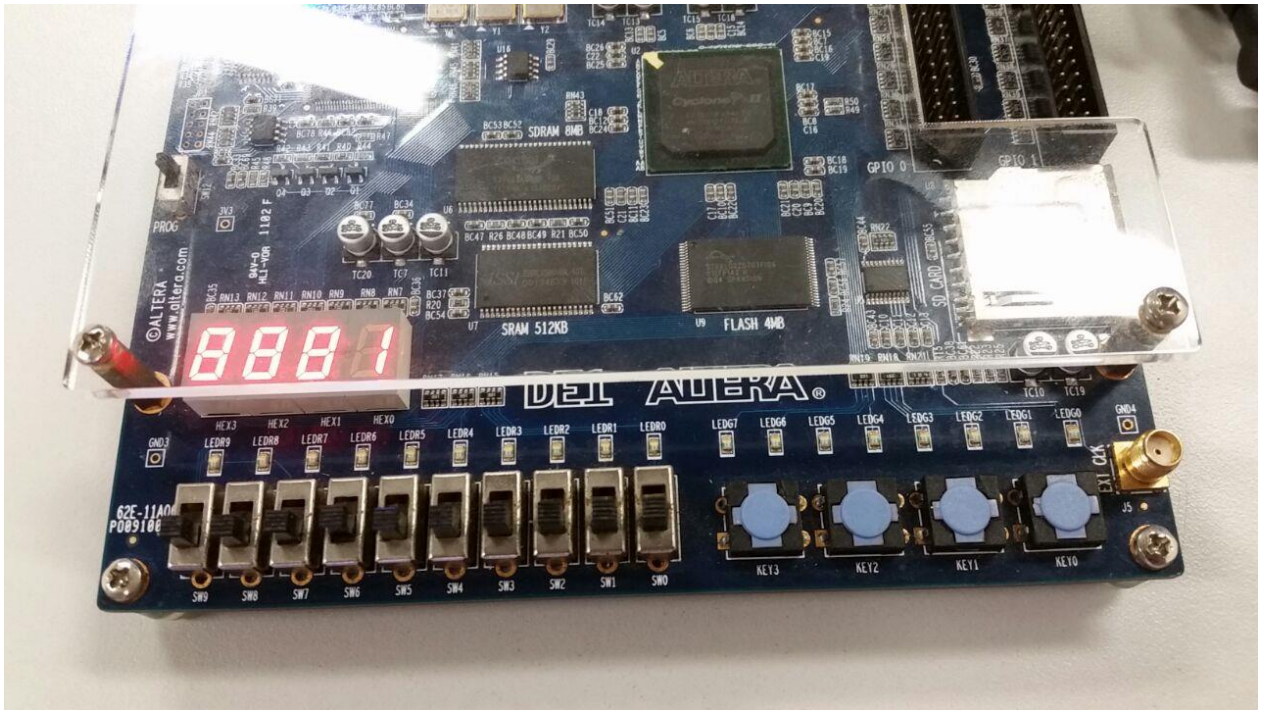

Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
/projetoPessoal_TB/a
/projetoPessoal_TB/ledVerde
/projetoPessoal_TB/ledVermelho
[1]
[0]
/projetoPessoal_TB/display
/projetoPessoal_TB/clock
0
xx
xx
StX
StX
xxxxxxx
0
0 12 13 6 10 8
00 01 00 10 00
00 01 10 00
1111001 0100100 0110000 0011001 0010010 1111001
Now
345709.7 ns
Cursor 1
0.00 ns
0.2 ns
0.4 ns
0.6 ns
0.8 ns
0 ps to 903 ps
Now: 408,151,400 ps Delta: 0

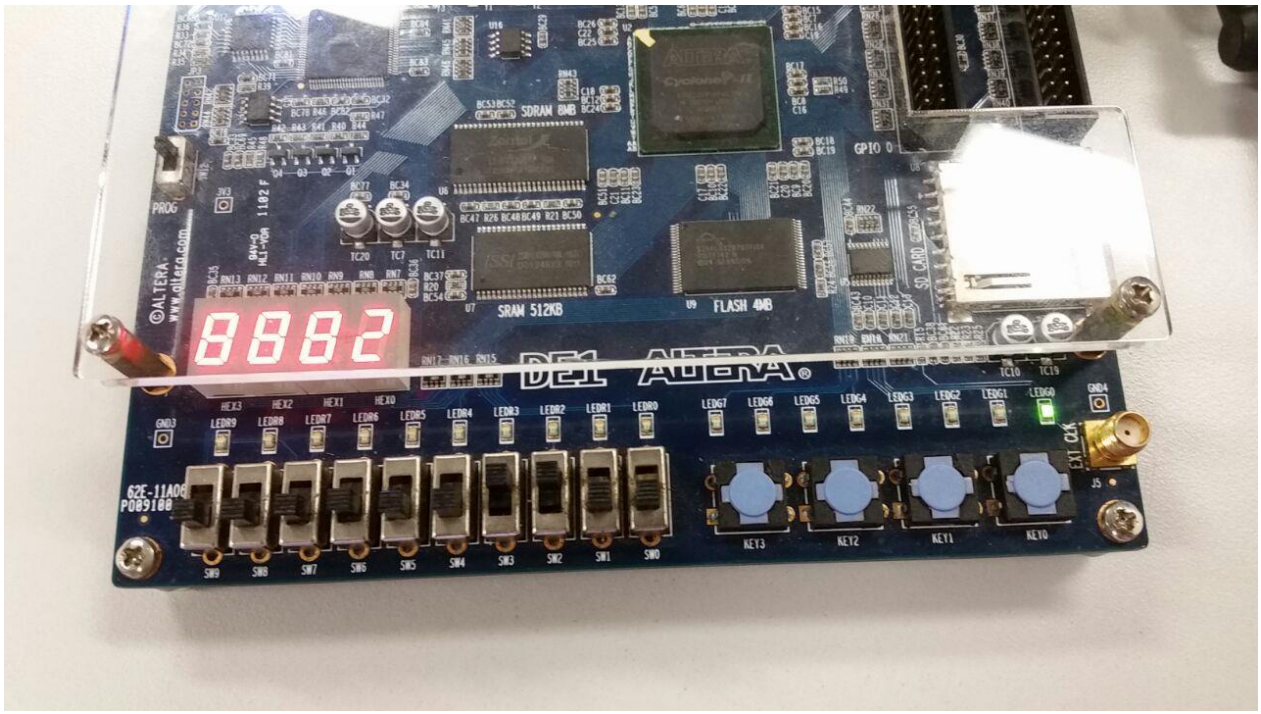

Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
/projetoPessoal_TB/a
/projetoPessoal_TB/ledVerde
/projetoPessoal_TB/ledVermelho
[1]
[0]
/projetoPessoal_TB/display
/projetoPessoal_TB/clock
0
xx
xx
StX
StX
xxxxxxx
0
0 12 13 6 10 8
00 01 00 10 00
00 01 10 00
1111001 0100100 0110000 0011001 0010010 1111001
Now
345709.7 ns
Cursor 1
0.00 ns
0.2 ns
0.4 ns
0.6 ns
0.8 ns
0 ps to 903 ps
Now: 546,784,975 ps Delta: 0

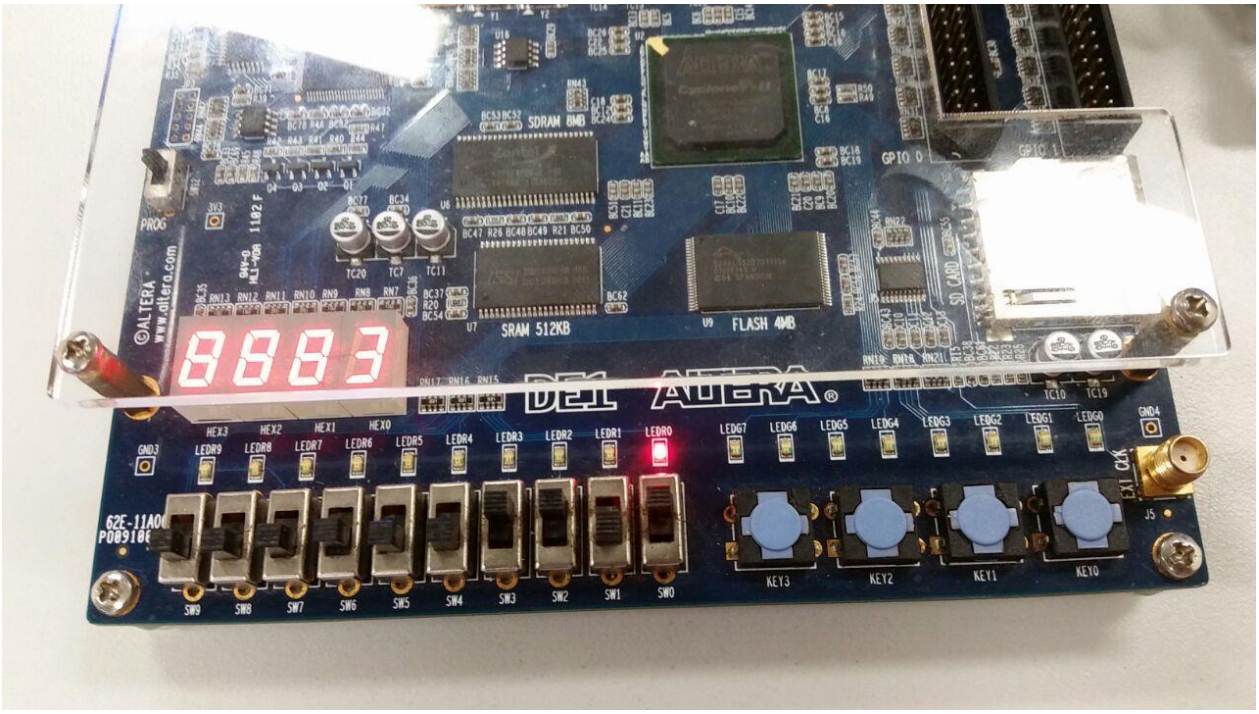


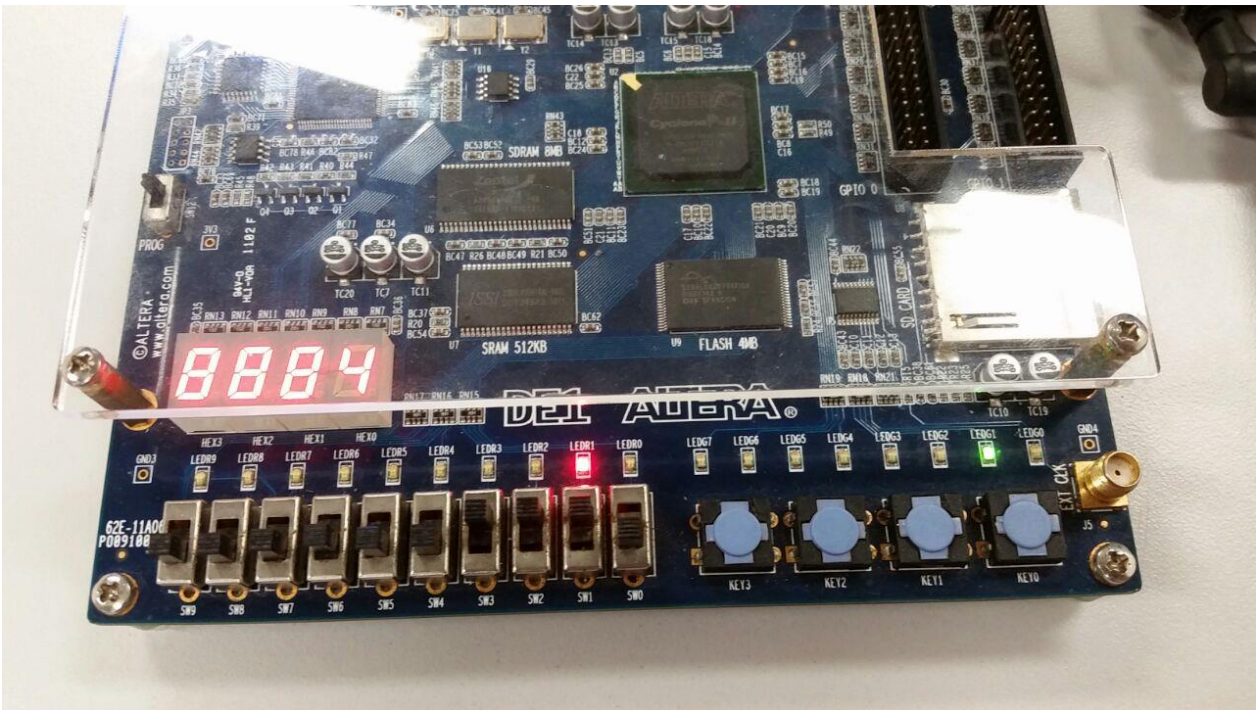
Com base nas “Waves” da simulação RTL, conclui-se que o circuito programado em Verilog corresponde ao requisitado na máquina de estados, migrando entre os estados “A”, “B”, “C” e “D” e “E” conforme as entradas são alteradas.

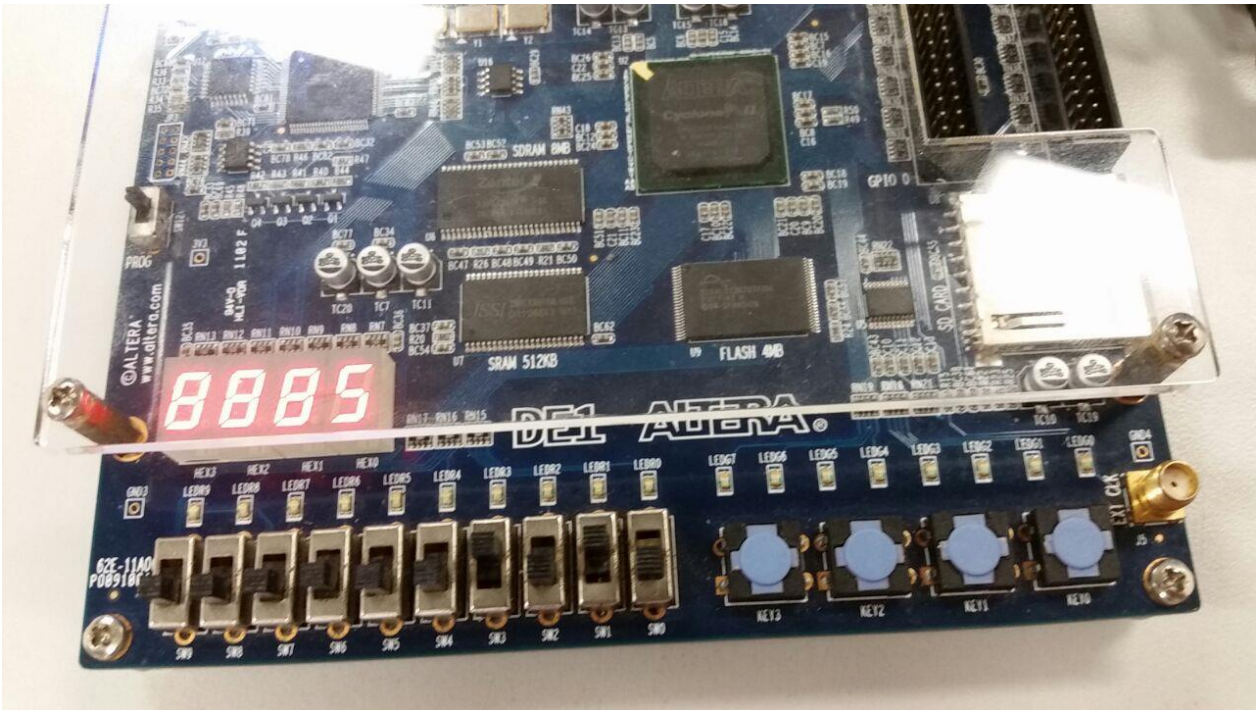
Resultado rodando na placa, após deploy






8883
SDRAM 8MB
SRAM 512KB
FLASH 4MB
GPIO 0
GPIO 1
SD CARD
PROG
©ALTERA
www.altera.com
HEX3 HEX2 HEX1 HEX0
LEDR9 LEDR8 LEDR7 LEDR6 LEDR5 LEDR4 LEDR3 LEDR2 LEDR1 LEDR0
LEDG7 LEDG6 LEDG5 LEDG4 LEDG3 LEDG2 LEDG1 LEDG0
SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
KEY3 KEY2 KEY1 KEY0
EXT CLK
GND3
GND4
J5


8884
SDRAM 8MB
SRAM 512KB
FLASH 4MB
GPIO 0
GPIO 1
SD CARD
PROG
©ALTERA
www.altera.com
HEX3 HEX2 HEX1 HEX0
LEDR9 LEDR8 LEDR7 LEDR6 LEDR5 LEDR4 LEDR3 LEDR2 LEDR1 LEDR0
LEDG7 LEDG6 LEDG5 LEDG4 LEDG3 LEDG2 LEDG1 LEDG0
SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
KEY3 KEY2 KEY1 KEY0
EXT CLK
GND3
GND4
J5


DE1
ALTERA
SDRAM 8MB
SRAM 512KB
FLASH 4MB
SD CARD
GPIO 0
PROG
©ALTERA
www.altera.com
EXT CLK
SW9
SW8
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2
KEY1
KEY0

Utilizados numeros para demonstrar em qual estado o circuito se encontra(1 à 5, A à E) , e os respectivos Leds vermelho e verde, sinalizando quem deve esperar/retornar e guardar os metais, e quem já pode atravessar.

4 Análise crítica e discussão

O Código 2.2 engloba todos 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.

Os resultados obtidos representam o comportamento ideal esperado, como definido no projeto. Das dificuldades encontradas, destacam-se as dificuldades em controlar as mudanças de estados, já que as entradas eram “pegas” automaticamente pela placa da Altera, podendo pular 2 estados de uma vez dependendo da entrada, e assim desorientar e dificultar a vida dos observadores.

Houve relativa facilidade para a idealização e desenvolvimento da máquina de estados, conteúdo já estudado previamente, principalmente devido a facilidades recentemente descobertas da ferramenta Quartus.