Relatório de Labolatório de Sistemas Digitais Avançados

Alunos:

Judah Holanda Correia Lima – Matricula: 1010500/5

José David Sousa de Araújo– Matricula: 1020678

José Kilson de Souza Oliveira – Matricula: 0810847

Sumário

[Material Utilizado 3](#_Toc388649918)

[Prática 4-Counters 4](#_Toc388649919)

[INTRODUçÃO 4](#_Toc388649920)

[Códigos Auxiliares 5](#_Toc388649921)

[Módulos utilizado para converter binário em decimal 5](#_Toc388649922)

[Módulo responsável em converter binário em hexadecimal 6](#_Toc388649923)

[Módulo utilizado para tratar o bouncer das chaves. 7](#_Toc388649924)

[Módulo do contador de 16 bits criado no Mega Wizard 8](#_Toc388649925)

[Módulo do contador de 26 bits 9](#_Toc388649926)

[Módulo contador de 16 bits 12](#_Toc388649927)

[Parte 1 15](#_Toc388649928)

[Módulo Principal do Projeto 15](#_Toc388649929)

[Parte 2 18](#_Toc388649930)

[Módulo Principal do Projeto 18](#_Toc388649931)

[Parte 3 22](#_Toc388649932)

[Módulo Principal do Projeto 22](#_Toc388649933)

[Parte 4 24](#_Toc388649934)

[Módulo Principal do Projeto 24](#_Toc388649935)

[Parte 5 26](#_Toc388649936)

[Módulo Principal do Projeto 26](#_Toc388649937)

[Conclusão 30](#_Toc388649938)

# Material Utilizado

* Kit de desenvolvimento DE0
* Software Quartus2 13.1 fornecido pelo fabricante Altera
* Sistema operacional: Windows 7 64 bits Serve Pack 1 e Windows 8.1 64 bits Serve Pack 1
* Estação de trabalho: Notebooks Core i7 Segunda Geração 4 Gb de Ram (Windows 7) e Core i7 Segunda Geração 8 Gb de Ram (Windows 8.1)

# Prática 4-Counters

## INTRODUçÃO

Contadores são circuitos digitais que variam seus estados, sob comando de um clock, de acordo com uma sequência predeterminada, cujo objetivo é gerar uma sequência numérica. São divididos em duas categorias: assíncronos e síncronos.

Os contadores podem ser ASSÍNCRONOS, quando existe o sinal de clock aplicado apenas ao primeiro estágio. Os estágios seguintes utilizam como sinal de sincronismo a saída de cada estágio anterior. Estes contadores também são denominados Ripple Counters. Os contadores também podem ser SÍNCRONOS, quando existe um sinal de clock único externo aplicado a todos os estágios ao mesmo tempo.

Esta prática tem como finalidade a resolução do exercício 4 que impõem o projeto do uso de Contadores. Na sua descrição de hardware foi utilizado verilog 2001, utilizando portas lógicas, lógica booleana e estruturas definidas pela própria sintaxe do verilog, realizando a descrição do hardware em módulos, tornando-os reaproveitáveis para as outras práticas.

A descrição de hardware desta prática foi feita em Verilog 2001 no Kit de desenvolvimento da empresa DE0, fornecido pela Unifor.

## Códigos Auxiliares

### Módulos utilizado para converter binário em decimal

module hexD (BIN, D);

input [15:0] BIN;

output reg [0:6] D;

always begin

case(BIN)

0:D=7'b0000001;

1:D=7'b1001111;

2:D=7'b0010010;

3:D=7'b0000110;

4:D=7'b1001100;

5:D=7'b0100100;

6:D=7'b0100000;

7:D=7'b0001111;

8:D=7'b0000000;

9:D=7'b0001100;

10:D=7'b0001000;

11:D=7'b1100000;

12:D=7'b0110001;

13:D=7'b1000010;

14:D=7'b0110000;

15:D=7'b0111000;

default:D=7'b0000001;

endcase

end

endmodule

### Módulo responsável em converter binário em hexadecimal

module b2dD (Bin, D);

input [3:0] Bin;

output reg [0:6] D;

always begin

case(Bin)

0:D=7'b0000001;

1:D=7'b1001111;

2:D=7'b0010010;

3:D=7'b0000110;

4:D=7'b1001100;

5:D=7'b0100100;

6:D=7'b0100000;

7:D=7'b0001111;

8:D=7'b0000000;

9:D=7'b0001100;

default:D=7'b0000001;

endcase

end

endmodule

### Módulo utilizado para tratar o bouncer das chaves.

module DeBounce (En, Clk, Clr, Q);

input En, Clk, Clr;

output reg Q;

always @ (posedge Clk)

begin

if (~Clr)

Q = 0;

else if (En)

Q = ~Q;

end

endmodule

//Módulo do flip-flop tipo D

module flipFlopD (En, Clk, Clr, Q);

input En, Clk, Clr;

output reg Q;

always @ (posedge Clk)

begin

if (~Clr)

Q = 0;

else if (En)

Q = ~Q;

end

endmodule

### Módulo do contador de 16 bits criado no Mega Wizard

module cont16 ( clock, cnt\_en, sclr, q);

input clock;

input cnt\_en;

input sclr;

output [15:0] q;

wire [15:0] sub\_wire0;

wire [15:0] q = sub\_wire0[15:0];

lpm\_counter LPM\_COUNTER\_component (

.clock (clock),

.cnt\_en (cnt\_en),

.sclr (sclr),

.q (sub\_wire0),

.aclr (1'b0),

.aload (1'b0),

.aset (1'b0),

.cin (1'b1),

.clk\_en (1'b1),

.cout (),

.data ({16{1'b0}}),

.eq (),

.sload (1'b0),

.sset (1'b0),

.updown (1'b1));

defparam

LPM\_COUNTER\_component.lpm\_direction = "UP",

LPM\_COUNTER\_component.lpm\_port\_updown = "PORT\_UNUSED",

LPM\_COUNTER\_component.lpm\_type = "LPM\_COUNTER",

LPM\_COUNTER\_component.lpm\_width = 16;

Endmodule

### Módulo do contador de 26 bits

module counter\_26bit (En, Clk, Clr, Q);

input En, Clk, Clr;

output [25:0] Q;

wire [25:0] T, Qs;

flipFlopD T0 (En, Clk, Clr, Qs[0]);

assign T[0] = En & Qs[0];

flipFlopD T1 (T[0], Clk, Clr, Qs[1]);

assign T[1] = T[0] & Qs[1];

flipFlopD T2 (T[1], Clk, Clr, Qs[2]);

assign T[2] = T[1] & Qs[2];

flipFlopD T3 (T[2], Clk, Clr, Qs[3]);

assign T[3] = T[2] & Qs[3];

flipFlopD T4 (T[3], Clk, Clr, Qs[4]);

assign T[4] = T[3] & Qs[4];

flipFlopD T5 (T[4], Clk, Clr, Qs[5]);

assign T[5] = T[4] & Qs[5];

flipFlopD T6 (T[5], Clk, Clr, Qs[6]);

assign T[6] = T[5] & Qs[6];

flipFlopD T7 (T[6], Clk, Clr, Qs[7]);

assign T[7] = T[6] & Qs[7];

flipFlopD T8 (T[7], Clk, Clr, Qs[8]);

assign T[8] = T[7] & Qs[8];

flipFlopD T9 (T[8], Clk, Clr, Qs[9]);

assign T[9] = T[8] & Qs[9];

flipFlopD T10 (T[9], Clk, Clr, Qs[10]);

assign T[10] = T[9] & Qs[10];

flipFlopD T11 (T[10], Clk, Clr, Qs[11]);

assign T[11] = T[10] & Qs[11];

flipFlopD T12 (T[11], Clk, Clr, Qs[12]);

assign T[12] = T[11] & Qs[12];

flipFlopD T13 (T[12], Clk, Clr, Qs[13]);

assign T[13] = T[12] & Qs[13];

flipFlopD T14 (T[13], Clk, Clr, Qs[14]);

assign T[14] = T[13] & Qs[14];

flipFlopD T15 (T[14], Clk, Clr, Qs[15]);

assign T[15] = T[14] & Qs[15];

flipFlopD T16 (T[15], Clk, Clr, Qs[16]);

assign T[16] = T[15] & Qs[16];

flipFlopD T17 (T[16], Clk, Clr, Qs[17]);

assign T[17] = T[16] & Qs[17];

flipFlopD T18 (T[17], Clk, Clr, Qs[18]);

assign T[18] = T[17] & Qs[18];

flipFlopD T19 (T[18], Clk, Clr, Qs[19]);

assign T[19] = T[18] & Qs[19];

flipFlopD T20 (T[19], Clk, Clr, Qs[20]);

assign T[20] = T[19] & Qs[20];

flipFlopD T21 (T[20], Clk, Clr, Qs[21]);

assign T[21] = T[20] & Qs[21];

flipFlopD T22 (T[21], Clk, Clr, Qs[22]);

assign T[22] = T[21] & Qs[22];

flipFlopD T23 (T[22], Clk, Clr, Qs[23]);

assign T[23] = T[22] & Qs[23];

flipFlopD T24 (T[23], Clk, Clr, Qs[24]);

assign T[24] = T[23] & Qs[24];

flipFlopD T25 (T[24], Clk, Clr, Qs[25]);

assign T[25] = T[24] & Qs[25];

assign Q = Qs;

endmodule

### Módulo contador de 16 bits

module counter\_16bit (En, Clk, Clr, Q);

input En, Clk, Clr;

output [15:0] Q;

wire [15:0] T, Qs;

flipFlopD T0 (En, Clk, Clr, Qs[0]);

assign T[0] = En & Qs[0];

flipFlopD T1 (T[0], Clk, Clr, Qs[1]);

assign T[1] = T[0] & Qs[1];

flipFlopD T2 (T[1], Clk, Clr, Qs[2]);

assign T[2] = T[1] & Qs[2];

flipFlopD T3 (T[2], Clk, Clr, Qs[3]);

assign T[3] = T[2] & Qs[3];

flipFlopD T4 (T[3], Clk, Clr, Qs[4]);

assign T[4] = T[3] & Qs[4];

flipFlopD T5 (T[4], Clk, Clr, Qs[5]);

assign T[5] = T[4] & Qs[5];

flipFlopD T6 (T[5], Clk, Clr, Qs[6]);

assign T[6] = T[5] & Qs[6];

flipFlopD T7 (T[6], Clk, Clr, Qs[7]);

assign T[7] = T[6] & Qs[7];

flipFlopD T8 (T[7], Clk, Clr, Qs[8]);

assign T[8] = T[7] & Qs[8];

flipFlopD T9 (T[8], Clk, Clr, Qs[9]);

assign T[9] = T[8] & Qs[9];

flipFlopD T10 (T[9], Clk, Clr, Qs[10]);

assign T[10] = T[9] & Qs[10];

flipFlopD T11 (T[10], Clk, Clr, Qs[11]);

assign T[11] = T[10] & Qs[11];

flipFlopD T12 (T[11], Clk, Clr, Qs[12]);

assign T[12] = T[11] & Qs[12];

flipFlopD T13 (T[12], Clk, Clr, Qs[13]);

assign T[13] = T[12] & Qs[13];

flipFlopD T14 (T[13], Clk, Clr, Qs[14]);

assign T[14] = T[13] & Qs[14];

flipFlopD T15 (T[14], Clk, Clr, Qs[15]);

//assign T[15] = T[14] & Qs[15];

assign Q = Qs;

endmodule

## Parte 1

Esta prática tem como finalidade criar um 4-bit counter:

### Módulo Principal do Projeto

module part1(SW,LEDG, BUTTON, HEX0, HEX1, CLOCK\_50);

//Declaração dos sinais de entrada e saída

input [9:0] SW;

input [2:0] BUTTON;

output [9:0] LEDG;

output [0:6] HEX0;

output [0:6] HEX1;

//Sinais auxiliaries de interligação

wire [7:0] Q;

wire Bt;

//Chamada do modulo para controle do Bounce gerado pelas chaves

DeBounce D0(1, SW[4], SW[3], Bt);

//Módulo do contador de 8 bits

counter8bits(SW[1], Bt, SW[0], Q);

//Chamada do modulo dos display

hexD H0 (Q[3:0], HEX0);

hexD H1 (Q[7:4], HEX1);

assign LEDG = Q;

endmodule

//Módulo contador de 8 bits utilizando flipFlop tipo D.

module counter8bits(En, Clk, Clr, Q);

//Declaração dos sinais de entrada e saída

input En, Clk, Clr;

output [7:0] Q;

//Sinais auxiliares

wire [7:0] T, Qs;

wire En;

//Chamadas do módulos de flipFlop tipo D

//Senda a entrada do próximo flipFlop realimentado pela saída anterior

flipFlopD T0 (En, Clk, Clr, Qs[0]);

assign T[0] = En & Qs[0];

flipFlopD T1 (T[0], Clk, Clr, Qs[1]);

assign T[1] = T[0] & Qs[1];

flipFlopD T2 (T[1], Clk, Clr, Qs[2]);

assign T[2] = T[1] & Qs[2];

flipFlopD T3 (T[2], Clk, Clr, Qs[3]);

assign T[3] = T[2] & Qs[3];

flipFlopD T4 (T[3], Clk, Clr, Qs[4]);

assign T[4] = T[3] & Qs[4];

flipFlopD T5 (T[4], Clk, Clr, Qs[5]);

assign T[5] = T[4] & Qs[5];

flipFlopD T6 (T[5], Clk, Clr, Qs[6]);

assign T[6] = T[5] & Qs[6];

flipFlopD T7 (T[6], Clk, Clr, Qs[7]);

//Sinais gerados repassados para a saída do módulo

assign Q = Qs;

endmodule

Os testes da parte 1 foram realizados por sinais produzidos pelas chaves, e as saídas através dos displays de 7 segmentos. Quando aplicado o referido sinal a saída apresentou-se como esperado, mostrando a contagem nos displays de 7 segmentos, então pode-se concluir que a implementação foi bem sucedida.

## Parte 2

Esta prática tem como finalidade criar um circuito 16-bit counter:

### Módulo Principal do Projeto

Em “module **part2** (SW,LEDG, BUTTON, HEX0, HEX1, HEX2, HEX3);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

module part2(SW,LEDG, BUTTON, HEX0, HEX1, HEX2, HEX3);

input [9:0] SW;

input [2:0] BUTTON;

output [9:0] LEDG;

output [0:6] HEX0;

output [0:6] HEX1;

output [0:6] HEX2;

output [0:6] HEX3;

//Sinais auxiliares de interligação

wire [15:0] Q;

wire Bt;

//Chamada do modulo para controle do Bounce gerado pelas chaves

DeBounce D0(1, SW[4], SW[3], Bt);

//Chamada do modulo contator de 16 bits

FlipFlip16Bits b(SW[1], Bt, SW[0], Q);

//Chamada do módulodo display

hexD H0 (Q[3:0], HEX0);

hexD H1 (Q[7:4], HEX1);

hexD H2 (Q[11:8], HEX2);

hexD H3 (Q[15:12], HEX3);

endmodule

//Módulo FlipFlop de 16 Bits utilizando flipFlop tipo D.

module FlipFlip16Bits (En, Clk, Clr, Q);

//Sinais de saída e entrada

input En, Clk, Clr;

output [15:0] Q;

//Sinais auxiliares

wire [15:0] T, Qs;

//Chamada do FlipFlop modificado referente à Parte 2 do exercício

//Senda a entrada do próximo flipFlop realimentado pela saída anterior

FlipFlipDMod T0 (En, Clk, Clr, Qs[0]);

assign T[0] = En & Qs[0];

FlipFlipDMod T1 (T[0], Clk, Clr, Qs[1]);

assign T[1] = T[0] & Qs[1];

FlipFlipDMod T2 (T[1], Clk, Clr, Qs[2]);

assign T[2] = T[1] & Qs[2];

FlipFlipDMod T3 (T[2], Clk, Clr, Qs[3]);

assign T[3] = T[2] & Qs[3];

FlipFlipDMod T4 (T[3], Clk, Clr, Qs[4]);

assign T[4] = T[3] & Qs[4];

FlipFlipDMod T5 (T[4], Clk, Clr, Qs[5]);

assign T[5] = T[4] & Qs[5];

FlipFlipDMod T6 (T[5], Clk, Clr, Qs[6]);

assign T[6] = T[5] & Qs[6];

FlipFlipDMod T7 (T[6], Clk, Clr, Qs[7]);

assign T[7] = T[6] & Qs[7];

FlipFlipDMod T8 (T[7], Clk, Clr, Qs[8]);

assign T[8] = T[7] & Qs[8];

FlipFlipDMod T9 (T[7], Clk, Clr, Qs[9]);

assign T[9] = T[7] & Qs[9];

FlipFlipDMod T10 (T[9], Clk, Clr, Qs[10]);

assign T[10] = T[9] & Qs[10];

FlipFlipDMod T11 (T[10], Clk, Clr, Qs[11]);

assign T[11] = T[10] & Qs[11];

FlipFlipDMod T12 (T[11], Clk, Clr, Qs[12]);

assign T[12] = T[11] & Qs[12];

FlipFlipDMod T13 (T[12], Clk, Clr, Qs[13]);

assign T[13] = T[12] & Qs[13];

FlipFlipDMod T14 (T[13], Clk, Clr, Qs[14]);

assign T[14] = T[13] & Qs[14];

FlipFlipDMod T15 (T[14], Clk, Clr, Qs[15]);

assign Q = Qs;

endmodule

//modulo flipFlop tipo D modificado para a Parte 2.

module FlipFlipDMod (En, Clk, Clr, Q);

input En, Clk, Clr;

output reg Q;

always @ (posedge Clk)

if (~Clr)

Q = 0;

else if (En)

Q <= Q + 1;

Endmodule

Os testes da parte 2 foram realizados por sinais produzidos pelas chaves, e as saídas através dos displays de 7 segmentos. Quando aplicado o referido sinal a saída apresentou-se como esperado, mostrando a contagem nos displays de 7 segmentos, então pode-se concluir que a implementação foi bem sucedida.

## Parte 3

Esta prática tem como finalidade implementar um 16-bit counter utilizando um LPM encontrado na Library of Parameterized modules:

### Módulo Principal do Projeto

Em “module **part3** (SW,LEDG, BUTTON, HEX0, HEX1,CLOCK\_50);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

module part3(SW,LEDG, BUTTON, HEX0, HEX1,CLOCK\_50);

//Sinal de entrada e saída

input [9:0] SW;

input [2:0] BUTTON;

output [9:0] LEDG;

output [0:6] HEX0;

output [0:6] HEX1;

//Sinais auxiliares

wire [7:0] Q;

wire Bt;

//Chamada do modulo para controle do Bounce gerado pelas chaves

DeBounce D0(1, SW[4], SW[3], Bt);

//Módulo criado pelo Mega Wizard para o contador de 16 bits

cont16 (Bt, SW[1], SW[0], Q);

//Chamda dos módulos do Display

hexD H0 (Q[3:0], HEX0);

hexD H1 (Q[7:4], HEX1);

endmodule

Os testes da parte 3 foram realizados por sinais produzidos pelas chaves, e as saídas através dos displays de 7 segmentos. Quando aplicado o referido sinal a saída apresentou-se como esperado, mostrando a contagem nos displays de 7 segmentos, então pode-se concluir que a implementação foi bem sucedida.

## Parte 4

Esta prática tem como finalidade implementar um circuito que fará piscar sucessivamente os dígitos de 0 a 9, no display de 7 segmentos HEX0. Cada digito será exibido por 1 segundo e o contador que irá determinar esse 1 segundo. Será usado um clock de 50 MHz no contador.

### Módulo Principal do Projeto

Em “module **part4** (SW,HEX0,CLOCK\_50);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

module part4 (SW,HEX0,CLOCK\_50);

//Sinais de entrada e saída

input [9:0] SW;

output [0:6] HEX0;

input CLOCK\_50;

//Sinais auxiliares

wire [25:0] Q1;

wire [15:0] Q2;

reg Clr, Clr2,Clk;

//Utiliza o modulo do contador para converter a frequencoa de 50 Mhz em 1.6s

counter\_26bit count1(1, CLOCK\_50, Clr, Q1);

//a cada saída do contador de 26 bits aciona o contador para o display

counter\_16bit count2 (1, Clk, Clr2, Q2);

b2dD display(HEX0,Q);

//Executa um clear no contador de 26 bits após contar 50000000

always @ (negedge CLOCK\_50) begin

if (Q1 >= 50000000) begin

Clr = 0;

end else begin

Clr = 1;

end

Clk =~Clr;

end

//Fornece um clear no contandor do display se o valor for maior que 9

always @ (posedge Clr) begin

if (Q2 >= 9) begin

Clr2 = 0;

end else begin

Clr2 = 1;

end

end

//Chama o modulo do display

b2dD H0(Q2[3:0], HEX0);

endmodule

Os testes da parte 4 foram realizados por sinais produzidos a través de um clock de 50MHz, pelas chaves e as saídas através dos displays de 7 segmentos. Quando aplicado o referido sinal a saída apresentou-se como esperado, mostrando a contagem nos displays de 7 segmentos, então pode-se concluir que a implementação foi bem sucedida.

## Parte 5

Esta prática tem como finalidade de implementar um circuito que exibe a palavra “de0” utilizando os 4 displays de 7 segmentos HEX0 – 3. As letras vão se mover da direita pra esquerda em intervalos de 1 segundo, utilizando o clock.

### Módulo Principal do Projeto

Em “module **part5** (SW ,HEX0,HEX1, HEX2, HEX3,CLOCK\_50, LEDG);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

part5(SW ,HEX0,HEX1, HEX2, HEX3,CLOCK\_50, LEDG);

//Sinais de entrada e saída

output [0:6] HEX0,HEX1, HEX2, HEX3;

input CLOCK\_50;

output [9:0]LEDG;//Sinais auxiliares

wire [25:0] Q1;

wire [15:0] Q2;

reg Clr, Clr2,Clk;

//Utiliza o modulo do contador para converter a frequência de 50 Mhz em 1.6s

counter\_26bit count1(1, CLOCK\_50, Clr, Q1);

//a cada saída do contador de 26 bits aciona o contador para o display

counter\_16bit count2 (1, Clk, Clr2, Q2);

b2dD display(HEX0,Q);

//Executa um clear no contador de 26 bits após contar 50000000

always @ (negedge CLOCK\_50) begin

if (Q1 >= 50000000) begin

Clr = 0;

end else begin

Clr = 1;

end

Clk =~Clr;

end

//Fornece um clear no contador do display se o valor for maior que 9

always @ (posedge Clr) begin

if (Q2 >= 9) begin

Clr2 = 0;

end else begin

Clr2 = 1;

end

end

//Chama os módulos do display

bin2DEO0 H0(Q4[3:0], HEX0);

bin2DEO1 H1(Q4[3:0], HEX1);

bin2DEO2 H2(Q4[3:0], HEX2);

bin2DEO3 H3(Q4[3:0], HEX3);

endmodule

//Os módulos abaixo são referentes a cada Display.

module bin2DEO3 (BIN, Display);

input [3:0] BIN;

output reg [0:6] Display;

always begin

case (BIN)

8:Display=7'b1111111;

0:Display=7'b1111111;

1:Display=7'b1000010;

2:Display=7'b0110000;

3:Display=7'b0000001;

4:Display=7'b1111111;

5:Display=7'b1000010;

6:Display=7'b0110000;

7:Display=7'b0000001;

endcase

end

endmodule

module bin2DEO2 (BIN, Display);

input [3:0] BIN;

output reg [0:6] Display;

always begin

case (BIN)

7:Display=7'b1111111;

8:Display=7'b1000010;

0:Display=7'b1000010;

1:Display=7'b0110000;

2:Display=7'b0000001;

3:Display=7'b1111111;

4:Display=7'b1000010;

5:Display=7'b0110000;

6:Display=7'b0000001;

endcase

end

endmodule

module bin2DEO1 (BIN, Display);

input [3:0] BIN;

output reg [0:6] Display;

always begin

case (BIN)

6:Display=7'b1111111;

7:Display=7'b1000010;

8:Display=7'b0110000;

0:Display=7'b0110000;

1:Display=7'b0000001;

2:Display=7'b1111111;

3:Display=7'b1000010;

4:Display=7'b0110000;

5:Display=7'b0000001;

endcase

end

endmodule

module bin2DEO0 (BIN, Display);

input [3:0] BIN;

output reg [0:6] Display;

always begin

case (BIN)

0:Display=7'b0000001;

1:Display=7'b1111111;

2:Display=7'b1000010;

3:Display=7'b0110000;

4:Display=7'b0000001;

5:Display=7'b1111111;

6:Display=7'b1000010;

7:Display=7'b0110000;

8:Display=7'b0000001;

endcase

end

endmodule

Os testes da parte 5 foram realizados por sinais produzidos a través de um clock de 50MHz, pelas chaves e as saídas verificadas através da tabela fornecida na prática, o qual é mostrado abaixo, e mostrados através dos displays de 7 segmentos. Quando aplicado o referido sinal a saída apresentou-se como esperado, mostrando a contagem nos displays de 7 segmentos, então pode-se concluir que a implementação foi bem sucedida.



# Conclusão

De acordo com a realização da prática em questão, pode – se analisar o comportamento dos circuitos contadores. Todas as práticas se comportaram de acordo com o previsto. Nesta prática foi introduzido o uso de mais um registrador para retirada do bounce das chaves, utilizadas na prática, pois ao utilizar as chaves no contador, este apresentou uma contagem incorreta ocasionada pelas bouncer das chaves.

A maior parte da implementação é modularizado, apresentando os resultados esperados da prática, sendo todas as tarefas executadas perfeitamente.