Relatório de Labolatório de Sistemas Digitais Avançados

Alunos:

José David Sousa de Araújo– Matricula: 1020678

Judah Holanda Correia Lima – Matricula: 1010500/5

José Kilson – Matricula: XXXXXXX/X

Sumário

[Material Utilizado 3](#_Toc381387817)

[Prática 2- Numbers and Displays 4](#_Toc381387818)

[Parte 1 4](#_Toc381387819)

[Parte 2 5](#_Toc381387820)

[Parte 3 7](#_Toc381387821)

[Parte 4 10](#_Toc381387822)

[Parte 5 12](#_Toc381387823)

[Parte 6 13](#_Toc381387824)

# Material Utilizado

* Kit de desenvolvimento DE0
* Software Quartus fornecido pelo fabricante Altera

# Prática 2- Numbers and Displays

## Parte 1

Esta prática tem como finalidade fazer um BCD com dois displays usando 4 switchs para cada um:

Decodificador BCD de 0 a 9 para cada display:

module part1(SW,HEX0,HEX1,LEDG);

input [9:0] SW; // toggle switches

output [9:0] LEDG;

output [0:6] HEX0; // 7-seg displays

output [0:6] HEX1; // 7-seg displays

segmentDecoder7 s1(HEX0,SW[3:0]);

segmentDecoder7 s2(HEX1,SW[7:4]);

endmodule

Decodificador BCD de 0 a 9 para um display:

module segmentDecoder7(Display,D);

input [0:3] D;

output [0:6]Display;

assign Display[0] = ~(D[0]|D[2]|(D[1]&D[3])|~(D[1]|D[3])); // 3-to-1 multiplexer output

assign Display[1] = ~(~D[1] | (D[2]&D[3]) | ~(D[2]|D[3])); // 3-to-1 multiplexer output

assign Display[2] = ~(D[1] | ~D[2] | D[3]); // 3-to-1 multiplexer output

assign Display[3] = ~(D[0] |(D[2]&~D[1]) | (D[2]&~D[3]) | ~(D[1]|D[3]) | (D[1]&~D[2]&D[3])); // 3-to-1 multiplexer output

assign Display[4] = ~((D[2]&~D[3])| ~(D[1]|D[3])); // 3-to-1 multiplexer output

assign Display[5] = ~(D[0] | (D[1]&~D[2]) | (D[1]&~D[3]) | ~(D[2]|D[3])); // 3-to-1 multiplexer output

assign Display[6] = ~(D[0] | (D[1]&~D[2]) | (D[1]&~D[3]) | (~D[1]&D[2])); // 3-to-1 multiplexer output

//assign Display[7] = 1; // 3-to-1 multiplexer output

endmodule

## Parte 2

Esta prática tem como finalidade a implementação de um BCD de 4 bits para 2 display de 7 segmentos.

module part2(SW,HEX0,HEX1,LEDG);

input [9:0] SW;

output [0:6] HEX0;

output [0:6] HEX1;

output [9:0] LEDG;

wire Z;

wire [2:0] EQ;

wire [3:0] M;

comparatorMaior9 cm(SW[3:0],Z);

assign LEDG[0] = Z;

circuitA a(SW[3:0], EQ);

//Mutex usados na seleção em qual dos bits serão utilizados quando o valor for maior que 9

mutex2to1 b1(Z,SW[3],0,M[3] );

mutex2to1 b2(Z,SW[2],EQ[2],M[2] );

mutex2to1 b3(Z,SW[1],EQ[1],M[1] );

mutex2to1 b4(Z,SW[0],EQ[0],M[0] );

//chama o modulo circuitoB

circuitB c(HEX1[0:6],Z);

//Passa os valores de bits do mutes com a lógica do display de 7 segmentos.

segmentDecoder7 d(HEX0[0:6],M[3:0]);

endmodule

//Implementação de um mutex 2 para 1

module mutex2to1(S,C1,C2,M );

input S,C1,C2;

output M;

assign M = (C1 & ~S) | (C2 & S);

endmodule

//Implementação de um decodificador de 7 segmentos.

module segmentDecoder7(Display,D);

input [0:3] D;

output [0:6]Display;

assign Display[0] = ~(D[0]|D[2]|(D[1]&D[3])|~(D[1]|D[3])); // 3-to-1 multiplexer output

assign Display[1] = ~(~D[1] | (D[2]&D[3]) | ~(D[2]|D[3])); // 3-to-1 multiplexer output

assign Display[2] = ~(D[1] | ~D[2] | D[3]); // 3-to-1 multiplexer output

assign Display[3] = ~(D[0] |(D[2]&~D[1]) | (D[2]&~D[3]) | ~(D[1]|D[3]) | (D[1]&~D[2]&D[3]));

assign Display[4] = ~((D[2]&~D[3])| ~(D[1]|D[3])); // 3-to-1 multiplexer output

assign Display[5] = ~(D[0] | (D[1]&~D[2]) | (D[1]&~D[3]) | ~(D[2]|D[3])); // 3-to-1 multiplexer output

assign Display[6] = ~(D[0] | (D[1]&~D[2]) | (D[1]&~D[3]) | (~D[1]&D[2])); // 3-to-1 multiplexer output

endmodule

//Implemetação de um comparador maior que 9, EQ igual a 1 se for maior

module comparatorMaior9(B,EQ);

input [3:0] B;

output EQ;

assign EQ = B[3] & (B[2] | B[1]);

endmodule

//Implementação de um decodificador de 7 segmentos para mostrar 0 ou 1 no display, dependendo do módulo “comparatorMaior9(B,EQ)”.

module circuitB(Display,C);

input C;

output [0:6] Display;

assign Display [0] = C;

assign Display [1] = 0;

assign Display [2] = 0;

assign Display [3] = C;

assign Display [4] = C;

assign Display [5] = C;

assign Display [6] = 1;

endmodule

//Comverter os 3 bits em valores de 0 a 5 para ser selecionado pelo mutex

module circuitA(B, EQ);

input [2:0] B;

output [2:0] EQ;

assign EQ[2] = B[2] & B[1];

assign EQ[1] = ~B[1];

assign EQ[0] = B[0];

endmodule

## Parte 3

Esta prática tem como finalidade desenvolver um somador com carry.

module part3(SW,HEX0,HEX1,LEDG);

input [9:0] SW; // toggle switches

output [9:0] LEDG;

output [0:6] HEX0; // 7-seg displays

//Sinal auxilizares para visibilidade, não perde desempenho em sua utilização.

wire [3:0] c0, T;

//chamada dos módulos do somador

adder add\_1(SW[8], c0[0], SW[4],SW[0], T[0]);

adder add\_2(c0[0], c0[1], SW[5],SW[1], T[1]);

adder add\_3(c0[1], c0[2], SW[6],SW[2], T[2]);

adder add\_4(c0[2], c0[3], SW[7],SW[3], T[3]);

//valores de saída

assign LEDG[3:0] = T[3:0];

assign LEDG[7:4] = c0[3:0];

//utiliza um decodificador implementado anteriormente

segmentDecoder7 as(HEX0,T);

endmodule

//Módulo do somador com carry

module adder(ci, co, a, b,T);

input a, b, ci;

output T, co;

wire s1;

assign s1 = (a & ~b) | (~a & b);

assign T = (ci & ~s1) | (~ci & s1);

//Chamada do modulo do mutex implementado anteriormente

mutex2to1 mute(s1, b, ci, co);

endmodule

## Parte 4

Esta prática tem como finalidade fazer um somador de dois BCDs mais carry:

Decodificador BCD de 0 a 9 para cada display, depois a soma destes e ao final coloca-se nos dois outros displays:

module part4(SW,HEX0,HEX1,HEX2,HEX3,LEDG);

input [9:0] SW; // toggle switches

output [9:0] LEDG;

output [0:6] HEX0; // 7-seg displays

output [0:6] HEX1; // 7-seg displays

output [0:6] HEX2; // 7-seg displays

output [0:6] HEX3; // 7-seg displays

wire cv;

wire [0:3]T0;

wire [0:3]co;

segmentDecoder7 SD70(HEX0,SW[3:0]);

segmentDecoder7 SD71(HEX1,SW[7:4]);

adder4 a4(SW[8], co[0:3], SW[3:0], SW[7:4],T0[0:3]);

//segmentDecoder7 SD72(HEX2,T0[0:3]);

//segmentDecoder7 SD72(HEX3,T0[0:3]);

sub9(co[0], cv, T0[0:3], HEX2, HEX3);

endmodule

Somador de 4 bits:

module adder4(ci, co, a, b,T);

input [3:0]a;

input [3:0]b;

input ci;

output [3:0]T, co;

adder(ci, co[0], a[0], b[0],T[0]);

adder(co[0], co[1], a[1], b[1],T[1]);

adder(co[1], co[2], a[2], b[2],T[2]);

adder(co[2], co[3], a[3], b[3],T[3]);

endmodule

Módulo que divide a soma em dois displays:

module sub9(ci, co, a, HEX2, HEX3);

input [3:0]a;

input ci;

output [0:6] HEX2; // 7-seg displays

output [0:6] HEX3; // 7-seg displays

output co;

wire [3:0]T1;

wire cf;

//Verificador se o valor em 4 bits é superior a 9 (valor máximo do display):

comparatorMaior9 (a, co);

assign cf=co|ci;

assign T1[0]=a[0];

assign T1[1]=(a[1]&~cf)|(~a[1]&cf);

assign T1[2]=(a[2]&~cf)|(((a[1]&a[2])|ci)&cf);

assign T1[3]=(a[3]&~cf);

segmentDecoder7 SD72(HEX2,T1[3:0]);

segmentDecoder7 SD73(HEX3,cf);

endmodule

Somador de 1 bit:

module adder(ci, co, a, b,T);

input a, b, ci;

output T, co;

wire s1;

assign s1 = (a & ~b) | (~a & b);

assign T = (ci & ~s1) | (~ci & s1) | (ci & a & b);

assign co = (ci & a) | (ci & b) | (a & b);

endmodule

## Parte 5

Esta prática tem como finalidade implementar o pseudo código fornecido, no link baixo na parte V:

ftp://ftp.altera.com/up/pub/Altera\_Material/12.0/Laboratory\_Exercises/Digital\_Logic/DE0/verilog/lab2\_Verilog.pdf

module part5(SW,HEX0,HEX1,HEX2,HEX3,LEDG);

input [9:0] SW; // toggle switches

output [0:6] HEX0; // 7-seg displays

output [0:6] HEX1; // 7-seg displays

output [0:6] HEX2; // 7-seg displays

output [0:6] HEX3; // 7-seg displays

output [9:0] LEDG;

//sinais utilizados para facilitar a visibilidade.

wire [3:0] c0, T, z1,S0,S1,c2;

reg [3:0]z0,c1;

wire EQ;

//chama o modulo do somador

adder add1(SW[8], c0[0], SW[4],SW[0], T[0]);

adder add2(c0[0], c0[1], SW[5],SW[1], T[1]);

adder add3(c0[1], c0[2], SW[6],SW[2], T[2]);

adder add4(c0[2], c0[3], SW[7],SW[3], T[3]);

//chama o modulo do comparador maior do que 9.

comparatorMaior9 cp2(T,EQ);

//verifica o resultado da comparação e atribui valores aos sinais com base no pseudo código.

always@(EQ)

if(EQ == 1'b1)

begin

z0=4'b1010;

c1 = 4'b0001;

end

else

begin

z0=4'b0000;

c1 = 4'b0000;

end

//Atribui 4 XOR ao circuito para converter o somador num subtrator.

assign z1[0] = (1 & ~z0[0]) | (0 & z0[0]);

assign z1[1] = (1 & ~z0[1]) | (0 & z0[1]);

assign z1[2] = (1& ~z0[2]) | (0 & z0[2]);

assign z1[3] = (1 & ~z0[3]) | (0 & z0[3]);

//chama o modulo do somador

adder add5(1, c2[0], T[0],z1[0], S0[0]);

adder add6(c2[0], c2[1], T[1],z1[1], S0[1]);

adder add7(c2[1], c2[2], T[2],z1[2], S0[2]);

adder add8(c2[2], c2[3], T[3],z1[3], S0[3]);

assign S1[3:0] = c1;

//chamda o modulo do decodificador e imprimi os valores de A, B, S) e S1.

segmentDecoder7(HEX0,S0);

segmentDecoder7(HEX1,S1);

segmentDecoder7(HEX2,SW[3:0]);

segmentDecoder7(HEX3,SW[7:4]);

//atribui valores para leds.

assign LEDG[3:0] = S0;

assign LEDG[7:4] = z1;

endmodule

## Parte 6

Esta prática tem como finalidade fazer um BCD com dois displays usando 4 switchs para cada um:

Decodificador BCD de 0 a 9 para cada display a partir de 2 e 4 switchs, como foram feitos para 4 o segundo tem os 2 ultimos bits zerados:

module part6(SW,HEX0,HEX1,LEDG);

input [9:0] SW; // toggle switches

output [9:0] LEDG;

output [0:6] HEX0; // 7-seg displays

output [0:6] HEX1; // 7-seg displays

wire [3:0]T0;

assign T0[0]=SW[4];

assign T0[1]=SW[5];

assign T0[2]=0;

assign T0[3]=0;

segmentDecoder7 (HEX0,SW[3:0]);

segmentDecoder7 (HEX1,T0[3:0]);

endmodule