Relatório de Labolatório de Sistemas Digitais Avançados

Alunos:

Judah Holanda Correia Lima – Matricula: 1010500/5

José David – Matricula: XXXXXXX/X

José Kilson – Matricula: XXXXXXX/X

Sumário

[Material Utilizado 3](#_Toc380764288)

[Prática 1- Switches, Lights, and Multiplexers 4](#_Toc380764289)

[Parte 1 4](#_Toc380764290)

[Parte 2 5](#_Toc380764291)

[Parte 3 7](#_Toc380764292)

[Parte 4 9](#_Toc380764293)

[Parte 5 9](#_Toc380764294)

[Parte 6 10](#_Toc380764295)

# Material Utilizado

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

# Prática 1- Switches, Lights, and Multiplexers

## Parte 1

Esta prática tem como finalidade uma ligação direta entre os sinais de entrada e de saída. Para esta finalidade é proposto a seguinte código em verilog:

module **part1** (SW, LEDR);

input [17:0] SW; // toggle switches

output [17:0] LEDR; // red LEDs

assign LEDR = SW; //operation

endmodule

Em “module **part1** (SW, LEDR);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

A declaração abaixo, especifica a entrada e saída lógica dos sinais, usando uma declaração em vetor [MSB:LSB].

input [17:0] SW; // toggle switches

output [17:0] LEDR; // red LEDs

Em “assign LEDR = SW;”, especifica a inteligação entre os switches e os leds.

## Parte 2

Esta prática tem como finalidade a implementação de um mutex 2 para 1. Abaixo o código proposto em verilog:

module part2 (SW,LEDG);

input [9:0] SW;

output [9:0] LEDG;

wire [2:0] M;

mutex M0(SW[5:3],SW[8:6],SW[9],M);

assign LEDG[9:3] = SW[9:3];

assign LEDG[2:0] = M;

endmodule

module mutex (X,Y,S,M);

input [2:0] X, Y;

output [2:0] M;

input S;

assign M[0] = (!S&X[0]) | (S&Y[0]);

assign M[1] = (!S&X[1]) | (S&Y[1]);

assign M[2] = (!S&X[2]) | (S&Y[2]);

endmodule

Em “module **part2** (SW, LEDR);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

A declaração abaixo, especifica a entrada e saída lógica dos sinais, usando uma declaração em vetor [MSB:LSB].

input [17:0] SW; // toggle switches

output [17:0] LEDR; // red LEDs

Em “wire [2:0] M;”é declarada um interligação par facilitar a visibilidade, não prejudicando o desempenho do circuito final gerado.

Em “mutex M0(SW[5:3],SW[8:6],SW[9],M);”, cria um mutex referente ao módulo “mutex” implementado.

Abaixo apresenta a interligação dos resultado na saída:

assign LEDG[9:3] = SW[9:3];

assign LEDG[2:0] = M;

Em “module mutex (X,Y,S,M);” é definido o módulo que será implementado o multiplexador.

Abaixo, as definições do multiplexador:

input [2:0] X, Y;

output [2:0] M;

input S;

Cada operação abaixo seleciona um dos bits para sair como resultado, dependendo de S, que é o seletor, do multiplexador.

assign M[0] = (!S&X[0]) | (S&Y[0]);

assign M[1] = (!S&X[1]) | (S&Y[1]);

assign M[2] = (!S&X[2]) | (S&Y[2]);

## Parte 3

Esta prática tem como finalidade desenvolver um mutex 3 para 2. Abaixo o código em verilog:

module part3(SW,LEDG);

input [9:0] SW;

output [9:0] LEDG;

wire[1:0] M;

mutex3To2 M1(SW[9:8], SW[7:6], SW[5:4], SW[3:2], M);

assign LEDG[9:2] = SW[9:2];

assign LEDG[1:0] = M[1:0];

endmodule

module mutex3To2(U,V,W,S,M);

input [1:0] S, U, V, W;

output [1:0] M;

wire m0, m1;

assign m0 = (~S[0] & U[0]) | (S[0] & V[0]);

assign M[0] = (~S[1] & m0) | (S[1] & W[0]);

assign m\_1 = (~S[0] & U[1]) | (S[0] & V[1]);

assign M[1] = (~S[1] & m1) | (S[1] & W[1]);

endmodule

Em “module **part3** (SW, LEDR);” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

A declaração abaixo, especifica a entrada e saída lógica dos sinais, usando uma declaração em vetor [MSB:LSB].

input [17:0] SW; // toggle switches

output [17:0] LEDR; // red LEDs

Em “wire [2:0] M;”é declarada uma interligação par facilitar a visibilidade, não prejudicando o desempenho do circuito final gerado.

Em “module mutex3To2 (X,Y,S,M);” é definido o módulo que será implementado o multiplexador de 3 entradas casa um contendo 2 bitis e a saída de 2 bits.

A declaração abaixo, especifica a entrada e saída lógica dos sinais, usando uma declaração em vetor [MSB:LSB].

input [1:0] S, U, V, W;

output [1:0] M;

wire m0, m1;

Em “wire m0, m1;” é declarado uma inter ligação par facilitar a visibilidade, não prejudicando o desempenho do circuito final gerado.

Na operaçao abaixo, o primeiro bit do seletor[0], serve para selecionar entre U[0] e V[0], enquanto o segundo bit de seleção S[1], seleciona o resultado anteiror com o W[0].

assign m0 = (~S[0] & U[0]) | (S[0] & V[0]);

assign M[0] = (~S[1] & m0) | (S[1] & W[0]);

Na operaçao abaixo, o primeiro bit do seletor[0], serve para selecionar entre U[1] e V[1], enquanto o segundo bit de seleção S[1], seleciona o resultado anteiror com o W[1].

assign m1 = (~S[0] & U[1]) | (S[0] & V[1]);

assign M[1] = (~S[1] & m1) | (S[1] & W[1]);

## Parte 4

## Parte 5

## Parte 6

O Objetivo desta ultima parte é rotacionar a palavra dE0 nos 4 displays utilizando duas chaves.

Tabela a sugerida:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SW[9] | SW[8] | Display[3] | Display[2] | Display[1] | Display[0] |
| 0 | 0 |  | d | E | 0 |
| 0 | 1 | d | E | 0 |  |
| 1 | 0 | E | 0 |  | d |
| 1 | 1 | 0 |  | d | E |

* Este modulo cria um shifter para cada display e informa que display é.

module part6 (SW, HEX0, HEX1, HEX2, HEX3);

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

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

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

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

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

shifter M0 (0, 0, SW[9], SW[8], HEX0);

shifter M1 (0, 1, SW[9], SW[8], HEX1);

shifter M2 (1, 0, SW[9], SW[8], HEX2);

shifter M3 (1, 1, SW[9], SW[8], HEX3);

endmodule

* Este modulo “shifteia” as entradas do módulo do display para que o texto possa “correr”.

module shifter (S1, S0, SW1, SW0, HEX);

input S1, S0, SW1, SW0;

output [0:7] HEX;

assign M0 = (~SW0&S0)|(SW0&~S0);

assign M1 = ((~SW1&~SW0)&S1)|((~SW1&SW0)&(S1^S0))|((SW1&~SW0)&~S1)|((SW1&SW0)&~(S1^S0));

char\_7seg H0 (M1, M0, HEX);

endmodule

* Este módulo imprime no display selecionado pelas chaves S1 e S0, fazendo, caso não haja tratamento nas chaves, o inicio da rotação do display (XdE0).

module char\_7seg (S1, S0, Display);

input S0, S1;

output [0:7] Display; // output 7-seg code

assign Display[0] = S1;

assign Display[1] = S0;

assign Display[2] = S0;

assign Display[3] = (S0&S1);

assign Display[4] = (S0&S1);

assign Display[5] = S1;

assign Display[6] = ~(S0^S1);

assign Display[7] = 1;

endmodule