Relatório de Labolatório de Sistemas Digitais Avançados

Prática 6 – Adders, Subtractors and Multipliers

Alunos:

José David Sousa de Araújo– Matricula: 1020678

Judah Holanda Correia Lima – Matricula: 1010500/5

José Kilson de Souza Oliveira – Matricula: 0810847/1

Sumário

[Material Utilizado 3](#_Toc388649757)

[Prática 6 – Adders, Subtractors and Multipliers 4](#_Toc388649758)

[INTRODUÇÃO 4](#_Toc388649759)

[Códigos Auxiliares 4](#_Toc388649760)

[Módulo Removedor de Bounce 4](#_Toc388649761)

[Módulo Codificador para o Display de 7 Segmentos BCD (B2D) 4](#_Toc388649762)

[Parte 1 6](#_Toc388649763)

[Módulo Principal do Projeto 6](#_Toc388649764)

[Parte 2 8](#_Toc388649765)

[Módulo Principal do Projeto 8](#_Toc388649766)

[Parte 3 10](#_Toc388649767)

[Módulo Principal do Projeto 10](#_Toc388649768)

[Parte 4 12](#_Toc388649769)

[Módulo Principal do Projeto 12](#_Toc388649770)

[parte 5 13](#_Toc388649771)

[módulo principal do projeto 13](#_Toc388649772)

[Conclusão 16](#_Toc388649773)

# 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 6 – Adders, Subtractors and Multipliers

## INTRODUÇÃO

Esta prática tem como finalidade o estudo do uso de circuitos combinacionais que somam, subtraem e multiplicam números. Cada circuito será descrito em Verilog e implementado em uma placa DE0 da Altera.

Um circuito combinacional é constituído por um conjunto de portas lógicas as quais determinam os valores das saídas diretamente a partir dos valores atuais das entradas. Pode-se dizer que um circuito combinacional realiza uma operação de processamento de informação a qual pode ser especificada por meio de um conjunto de equações Booleanas. No caso, cada combinação de valores de entrada pode ser vista como uma informação diferente e cada conjunto de valores de saída representa o resultado da operação.

## Códigos Auxiliares

### Módulo Removedor de Bounce

Esse módulo tem como finalidade a retirada do bounce de alguma chave do circuito.

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 Codificador para o Display de 7 Segmentos BCD (B2D)

module b2d\_ssd (X, SSD);

input [4:0] X;

output reg [0:6] SSD;

always begin

case(X)

0:SSD=7'b0000001;

1:SSD=7'b1001111;

2:SSD=7'b0010010;

3:SSD=7'b0000110;

4:SSD=7'b1001100;

5:SSD=7'b0100100;

6:SSD=7'b0100000;

7:SSD=7'b0001111;

8:SSD=7'b0000000;

9:SSD=7'b0001100;

endcase

end

endmodule

## Parte 1

Este exercício tem como finalidade criar um circuito acumulador de 8 – bits, será usado como referência o circuito somador de 4 – bits ripple-carry feito na prática 2;

### Módulo Principal do Projeto

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

Os conteúdos do contador vão ser exibidos nos LEDs verdes.

module part1v1(SW, LEDG, KEY);

input [9:0] SW; // if this is 1, add; else subtract

output [9:0] LEDG;

input [2:0] KEY;

reg [6:0] result;

reg carry;

wire [6:0] dataa, datab;

assign dataa = SW [6:0];

//assign datab = SW [7:4];

DeBounce D0(1, SW[9], SW[8], Bt);

always @ (posedge Bt)// or negedge SW[7] )//or posedge KEY[2] or posedge KEY[1])

begin

if(Bt == 1)

begin

result <= dataa + datab;

if( result >= 127)

carry = 1'b1;

else

carry = 1'b0;

end

if(SW[7] == 0)

begin

result <= 0;

carry <=0;

end

end

assign LEDG[6:0] = result;

assign LEDG[9] = Bt;

assign LEDG[8] = carry;

assign datab = result;

endmodule

Os testes referente a parte 1 desse exercício foram realizados aplicando sinais produzidos pelas chaves que representavam números que iam ser somados e as saídas verificadas através dos displays de 7 segmentos que exibiam o resultado da soma, quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.

## Parte 2

Esta prática tem como finalidade estender a parte 1, o circuito irá ser capaz de além de somar, também subtrair números.

### Módulo Principal do Projeto

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

Os conteúdos do contador vão ser exibidos nos LEDs verdes e nos Displays.

module part2(SW, LEDG, KEY);

input [9:0] SW; // if this is 1, add; else subtract

output [9:0] LEDG;

input [2:0] KEY;

reg [6:0] result;

reg carry;

wire [6:0] dataa, datab;

assign dataa = SW [6:0];

//assign datab = SW [7:4];

DeBounce D0(1, SW[9], SW[8], Bt);

always @ (posedge Bt)// or negedge SW[7] )//or posedge KEY[2] or posedge KEY[1])

begin

if(Bt == 1)

begin

result <= dataa - datab;

if( result >= 127)

carry = 1'b1;

else

carry = 1'b0;

end

if(SW[7] == 0)

begin

result <= 0;

carry <=0;

end

end

assign LEDG[6:0] = result;

assign LEDG[9] = Bt;

assign LEDG[8] = carry;

assign datab = result;

endmodule

Os testes referente a parte 2 desse exercício foram realizados aplicando sinais produzidos pelas chaves que representavam números que iam ser somados ou subtraídos e as saídas verificadas através dos displays de 7 segmentos que exibiam o resultado da soma ou da subtração, quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.

## Parte 3

Esta prática tem como finalidade implementar um circuito que atua como um multiplicador de números binários.

### Módulo Principal do Projeto

Em “module **part3** (SW, LEDG, KEY, HEX0, HEX1, HEX2, HEX3)” é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

Os conteúdos do relógio vão ser exibidos nos LEDs verdes e nos Displays.

module part3(SW, LEDG, KEY, HEX0, HEX1, HEX2, HEX3);

input [9:0] SW; // if this is 1, add; else subtract

output [9:0] LEDG;

input [2:0] KEY;

reg [9:0] result;

reg carry;

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

wire [4:0] dataa, datab;

assign dataa = SW [3:0];

assign datab = SW [7:4];

//assign datab = SW [7:4];

DeBounce D0(1, SW[9], SW[8], Bt);

always @ (posedge Bt)// or negedge SW[7] )//or posedge KEY[2] or posedge KEY[1])

begin

if(Bt == 1)

begin

result <= dataa \* datab;

if( result >= 127)

carry = 1'b1;

else

carry = 1'b0;

end

/\*if(SW[7] == 0)

begin

result <= 0;

carry <=0;

end\*/

end

hexD H1 (dataa, HEX0);

hexD H2 (datab, HEX1);

hexD H3 (result[3:0], HEX2);

hexD H4 (result[7:4],HEX3);

endmodule

Os testes referente a parte 3 desse exercício foram realizados aplicando sinais produzidos pelas chaves que representavam números que iam ser multiplicados e as saídas verificadas através dos displays de 7 segmentos que exibiam o resultado da multiplicação, quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.

## Parte 4

Esta prática irá estender a prática 3 e tem como finalidade implementar um circuito multiplicador 5x5 com entradas e saídas registradas, utilizando somadores de n-bits.

### Módulo Principal do Projeto

Em “module **part4** (SW, LEDG, KEY, HEX0, HEX1, HEX2, HEX3) é feita a declaração do módulo da lógica do circuito, como os sinais nele existentes.

Os conteúdos do código Morse vão ser exibidos no LED verde.

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

input [9:0] SW; // if this is 1, add; else subtract

output [9:0] LEDG;

input [2:0] KEY;

reg [15:0] result;

reg carry;

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

wire [4:0] dataa, datab;

assign dataa = SW [3:0];

assign datab = SW [7:4];

DeBounce D0(1, SW[9], SW[8], Bt);

always @ (posedge Bt)

begin

if(Bt == 1)

begin

result <= dataa \* datab;

end

end

hexD H1 (result[3:0], HEX0);

hexD H2 (result[7:4], HEX1);

hexD H3 (result[11:8], HEX2);

hexD H4 (result[15:12],HEX3);

endmodule

Os testes referente a parte4 desse exercício foram realizados aplicando sinais produzidos pelas chaves que representavam números que iam ser multiplicados e as saídas verificadas através dos displays de 7 segmentos que exibiam o resultado da multiplicação, quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.

## parte 5

Está prática e tem como finalidade implementar um circuito multiplicador AxB utilizando uma árvore de somas.

### módulo principal do projeto

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

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

input [9:0] SW;

output [9:0] LEDG;

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

//DeBounce D0(1, SW[9], SW[8], Bt);

//always @ (posedge Bt)

//begin

// if(Bt == 1)

// begin

multiplier M0 (SW [7:4], SW[3:0], LEDG[9:0]);

hexD H1 (LEDG[3:0], HEX0);

hexD H2 (LEDG[7:4], HEX1);

// end

//end

endmodule

// implements a 4-bit by 4-bit multiplier with 8-bit result

module multiplier (A, B, P);

input [3:0] A, B;

output [7:0] P;

wire c01, c02, c03, c04;

wire s02, s03, s04;

wire c12, c13, c14, c15;

wire s13, s14, s15;

wire c23, c24, c25, c26;

assign P[0] = A[0] & B[0];

fulladder F01 (A[1] & B[0], A[0] & B[1], 0, P[1], c01);

fulladder F02 (A[2] & B[0], A[1] & B[1], c01, s02, c02);

fulladder F03 (A[3] & B[0], A[2] & B[1], c02, s03, c03);

fulladder F04 (0, A[3] & B[1], c03, s04, c04);

fulladder F12 (s02, A[0] & B[2], 0, P[2], c12);

fulladder F13 (s03, A[1] & B[2], c12, s13, c13);

fulladder F14 (s04, A[2] & B[2], c13, s14, c14);

fulladder F15 (c04, A[3] & B[2], c14, s15, c15);

fulladder F23 (s13, A[0] & B[3], 0, P[3], c23);

fulladder F24 (s14, A[1] & B[3], c23, P[4], c24);

fulladder F25 (s15, A[2] & B[3], c24, P[5], c25);

fulladder F26 (c15, A[3] & B[3], c25, P[6], P[7]);

endmodule

module fulladder (a, b, ci, s, co);

input a, b, ci;

output co, s;

wire d;

assign d = a ^ b;

assign s = d ^ ci;

assign co = (b & ~d) | (d & ci);

endmodule

Os testes referente a parte 5 desse exercício foram realizados aplicando sinais produzidos pelas chaves que representavam números que iam ser multiplicados e as saídas verificadas através dos displays de 7 segmentos que exibiam o resultado da multiplicação, quando aplicado o referido sinal a saída apresentou-se como esperado, então pode-se concluir que a implementação foi bem sucedida.

# Conclusão

De acordo com a prática realizada pode-se aprender o comportamento de circuitos somadores, subtratores e multiplicadores em projetos, com isso podemos afirmar que esses circuitos são muito úteis em vários tipos de aplicações. A maior parte da implementação é modularizado, apresentando os resultados esperados da prática, sendo todas as tarefas executadas perfeitamente.