Relatório de Labolatório de Sistemas Digitais Avançados

Prática 9

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](#_Toc387753691)

[Prática 9 4](#_Toc387753692)

[INTRODUÇÃO 4](#_Toc387753693)

[Códigos Auxiliares 4](#_Toc387753694)

[Módulo Removedor de Bounce 4](#_Toc387753695)

[Módulo Codificador para o Display de 7 Segmentos BCD (B2D) 5](#_Toc387753696)

[Parte 1 (Exercício 7) 5](#_Toc387753697)

[Módulo Principal do Projeto 6](#_Toc387753698)

[Parte 2 (Exercício 7) 7](#_Toc387753699)

[Módulo Principal do Projeto 7](#_Toc387753700)

[Parte 1 (Exercício 8) 9](#_Toc387753701)

[Módulo Principal do Projeto 9](#_Toc387753702)

[Parte 2 (Exercício 8) 11](#_Toc387753703)

[Módulo Principal do Projeto 11](#_Toc387753704)

[Conclusão 12](#_Toc387753705)

# 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 9

## INTRODUÇÃO

As tarefas executadas em um computador precisam realizar o tratamento das entradas e fornecer saídas de acordo com a entrada fornecida, sendo necessário desenvolver uma lógica por circuitos para fazer as operações de cálculo e tomar as divisões no computador.

Para a implementação de um computador é necessário a inclusão de registradores, ULA (Unidade Lógica e Aritmética), entre outros, isto constitui a arquitetura do processador e a principal razão para torna-lo mais robusto, pois dependendo dos sinais e suas instruções são implementados, o seu desempenho pode ser variar.

Para esta prática é proposto o exercício 9 fornecido pela altera. Esta prática tem como finalidade o estudo e implementação de um processador simples, baseada nas instruções da prática.

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

## Códigos Auxiliares

### Módulo do processador simples

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

//declaração das variáveis de controle

reg IRin, DINout, Ain, Gout, Gin, AddSub;

reg [7:0] Rout, Rin;

wire [7:0] Xreg, Yreg;

wire [1:9] IR;

wire [1:3] I;

reg [9:0] MUXsel;

wire [8:0] R0, R1, R2, R3, R4, R5, R6, R7, result;

wire [8:0] A, G;

wire [1:0] Tstep\_Q;

//sinais para resetar a máquina de estado

wire Clear = Done || ~Resetn;

//incrementa ou reseta o contador com base

//no sinal de Clear e Clock

upcount Tstep (Clear, Clock, Tstep\_Q);

//Decodificação dos sinais para o registrador

assign I = IR[1:3];

dec3to8 decX (IR[4:6], 1'b1, Xreg);

dec3to8 decY (IR[7:9], 1'b1, Yreg);

/\*Tabela referente aos sinais para a geração das informações

T1 T2 T3

==========================================================

(mv):I0 | RYout,RXin, | |

| DONE | |

===========================================================

(mvi):I1|DINout,RXin, | |

DONE | |

===========================================================

(add):I2|RXout,Ain |RYout,Gin | Gout,RXin, | | Done

===========================================================

(sub):I3|RXout,Ain |RYout,Gin | Gout,RXin, |AddSub | Done

===========================================================

\*/

//Maquina de estado para as operações de atribuição, soma e subtração

always @(Tstep\_Q or I or Xreg or Yreg)

begin

//especifica os valores iniciais

IRin = 1'b0;

Rout[7:0] = 8'b00000000;

Rin[7:0] = 8'b00000000;

DINout = 1'b0;

Ain = 1'b0;

Gout = 1'b0;

Gin = 1'b0;

AddSub = 1'b0;

Done = 1'b0;

case (Tstep\_Q)

2'b00: // carrega o IR no DIN na etapa 0

begin

IRin = 1'b1;

end

2'b01: //define sinais na etapa 1

case (I)

3'b000:

begin

Rout = Yreg;

Rin = Xreg;

Done = 1'b1;

end

3'b001:

begin

DINout = 1'b1;

Rin = Xreg;

Done = 1'b1;

end

3'b010:

begin

Rout = Xreg;

Ain = 1'b1;

end

3'b011:

begin

Rout = Xreg;

Ain = 1'b1;

end

endcase

2'b10: //define sinais na etapa 2

case (I)

3'b010:

begin

Rout = Yreg;

Gin = 1'b1;

end

3'b011:

begin

Rout = Yreg;

Gin = 1'b1;

AddSub = 1'b1;

end

endcase

2'b11: //define sinais na etapa 3

case (I)

3'b010:

begin

Gout = 1'b1;

Rin = Xreg;

Done = 1'b1;

end

3'b011:

begin

Gout = 1'b1;

Rin = Xreg;

Done = 1'b1;

end

endcase

endcase

end

//instancia registradore, somadores e subtratores

regn reg\_0 (BusWires, Rin[0], Clock, R0);

regn reg\_1 (BusWires, Rin[1], Clock, R1);

regn reg\_2 (BusWires, Rin[2], Clock, R2);

regn reg\_3 (BusWires, Rin[3], Clock, R3);

regn reg\_4 (BusWires, Rin[4], Clock, R4);

regn reg\_5 (BusWires, Rin[5], Clock, R5);

regn reg\_6 (BusWires, Rin[6], Clock, R6);

regn reg\_7 (BusWires, Rin[7], Clock, R7);

//módulo para o registrador IR

regn reg\_IR (DIN, IRin, Clock, IR);

defparam reg\_IR.n = 9;

//modulos para os registradores A e G

regn reg\_A (BusWires, Ain, Clock, A);

regn reg\_G (result, Gin, Clock, G);

addsub AS (~AddSub, A, BusWires, result);

//adiciona os valores dos registradores a saída

always @ (MUXsel or Rout or Gout or DINout)

begin

MUXsel[9:2] = Rout;

MUXsel[1] = Gout;

MUXsel[0] = DINout;

case (MUXsel)

10'b0000000001: BusWires = DIN;

10'b0000000010: BusWires = G;

10'b0000000100: BusWires = R0;

10'b0000001000: BusWires = R1;

10'b0000010000: BusWires = R2;

10'b0000100000: BusWires = R3;

10'b0001000000: BusWires = R4;

10'b0010000000: BusWires = R5;

10'b0100000000: BusWires = R6;

10'b1000000000: BusWires = R7;

endcase

end

endmodule

//Módulo do contador incrementado a cada pulso na

//borda de subida do clock e zerado quando o clear é setado

module upcount(Clear, Clock, Q);

input Clear, Clock;

output [1:0] Q;

reg [1:0] Q;

always @(posedge Clock)

if (Clear)

Q <= 2'b0;

else

Q <= Q + 1'b1;

endmodule

//Decodifica os sinais de entrada baseada na tabela do exercício 9, prática 1

module dec3to8(W, En, Y);

input [2:0] W;

input En;

output [0:7] Y;

reg [0:7] Y;

always @(W or En)

begin

if (En == 1)

case (W)

3'b000: Y = 8'b10000000;

3'b001: Y = 8'b01000000;

3'b010: Y = 8'b00100000;

3'b011: Y = 8'b00010000;

3'b100: Y = 8'b00001000;

3'b101: Y = 8'b00000100;

3'b110: Y = 8'b00000010;

3'b111: Y = 8'b00000001;

endcase

else

Y = 8'b00000000;

end

endmodule

//módulo para os registradores

module regn(R, Rin, Clock, Q);

parameter n = 16;

input [n-1:0] R;

input Rin, Clock;

output [n-1:0] Q;

reg [n-1:0] Q;

always @(posedge Clock)

if (Rin)

Q <= R;

Endmodule

### Módulo da memória rom

module Rom1Port (

address,

clock,

q);

input [4:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.addressstall\_a (1'b0),

.addressstall\_b (1'b0),

.byteena\_a (1'b1),

.byteena\_b (1'b1),

.clock1 (1'b1),

.clocken0 (1'b1),

.clocken1 (1'b1),

.clocken2 (1'b1),

.clocken3 (1'b1),

.data\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

altsyncram\_component.address\_aclr\_a = "NONE",

altsyncram\_component.clock\_enable\_input\_a = "BYPASS",

altsyncram\_component.clock\_enable\_output\_a = "BYPASS",

altsyncram\_component.init\_file = "inst\_mem.mif",

altsyncram\_component.intended\_device\_family = "Cyclone III",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

altsyncram\_component.lpm\_type = "altsyncram",

altsyncram\_component.numwords\_a = 32,

altsyncram\_component.operation\_mode = "ROM",

altsyncram\_component.outdata\_aclr\_a = "NONE",

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 5,

altsyncram\_component.width\_a = 9,

altsyncram\_component.width\_byteena\_a = 1;

endmodule

## Parte 1

Este exercício tem como finalidade a implementação de um processador simples seguindo a arquitetura mostrada na Figura 1:



Figura 1: Arquitetura do processador proposto

### Módulo Principal do Projeto

//módulo principal

module part1 (DIN, Resetn, Clock, Run, Done, BusWires);

//declaração dos sinais de entrada e saída

input [8:0] DIN;

input Resetn, Clock, Run;

output reg Done;

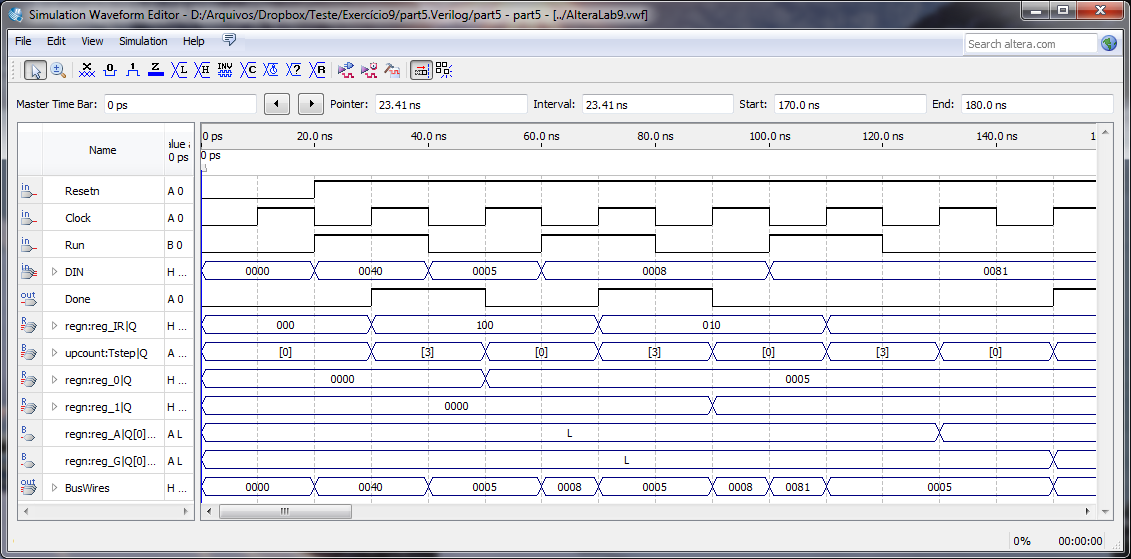
output reg BusWires;

//instancia o modulo do processador

proc P0 (DIN, Resetn, PClock, Run, Done, BusWires);

endmodule

Os testes foram realizados através da ferramenta Simulation WaveForm Editor, valores foram atribuídas as entradas e as saídas setadas ao começar a simulação. Os resultados das operações atribuição, adição e subtração foram executados corretamente como observado abaixo:



## Parte 2

Este exercício tem como finalidade a implementação de uma memória ROM no MegaWizard para a integração com o processador simples.

### Módulo Principal do Projeto

//módulo principal

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

//sinais de entrada e saída

input [9:0] SW;

input [3:0] KEY;

output reg [9:0] LEDG;

output [0:6] HEX1, HEX0;

//sinais auxiliares

wire [8:0] DIN, BusWires;

wire Resetn, Clock, Run, Done;

wire [4:0] addr;

//adiciona os sinais da chave

assign MClock = KEY[1];

assign PClock = KEY[2];

assign Resetn = KEY[0];

assign Run = SW[9];

assign addr = SW[3:0];

always

if (SW[8])

LEDG[8:0] = DIN;

else

//módulo do contador

counterModK C0 (MClock, Resetn, addr);

defparam C0.n = 5;

defparam C0.k = 32;

//módulo gerado pelo MegaWizard

Rom1Port R0 (addr, MClock, DIN);

//instancia módulo do processador da parte 1

proc P0 (DIN, Resetn, PClock, Run, Done, BusWires);

//atribuição da saída do processador ao led

LEDG = BusWires;

endmodule

Os testes foram realizados setando as chaves, contudo não foi obter os resultados esperados, pois houve dificuldade na forma como foram realizados.

# Conclusão

Analisando os resultados descritos em cada prática, percebe-se que as práticas foram feitas de acordo com as restrições presentes em sua descrição, obtendo as respostas esperadas dentro de seus parâmetros. As práticas fornecerem um aprendizado básico de implementação em Verilog de processador e memória, contudo alguns testes não foram devidamente feitos por causa da dificuldade da geração de alguns sinais.