**Lab 6 Report Hardik J Patel**

**EEC 180B 999121498**

**Cramer’s Rule in Hardware**

**Objective**

The purpose of this lab was to implement Cramer’s rule to solve a system of equations by calculating the determinants of the coefficients. This Implementation is done on an FPGA for a 2\*2 and a 3\*3 matrix.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Design and Test Procedure**

Part 1: For this part, a determinant calculator module for a 2\*2 matrix was designed and an lpm divide module was utilized from the library to do the necessary division required in the calculation. A module was then created that used parameters as inputs to the system and the parameters were initialized in an always block. The two modules for determinant calculations and division were then called and the determinants and unknowns were saved in a register file. A different module was written to be downloaded onto the DE2 board. The unknowns were then displayed on the HEX display.

*The codes are all attached at the end.*

Part 2: This part dealt with expanding the code for the 2\*2 matrix and make it applicable for a 3\*3 matrix.

*The codes are all attached at the end.*

Extra Credit: The design in extended for 4\*4 matrices.

*The codes are all attached at the end.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Results and Conclusions**

For the 2\*2 system: Clock Frequency = 40.00MHz

Total Power Dissipation = 117.10 mW

For the 3\*3 system: Clock Frequency = 1000.00MHz

Total Power Dissipation = 116.34 mW

For the 2\*2 system: Clock Frequency = 1000.00MHz

Total Power Dissipation = 133.35 mW

This lab taught us the trade offs in power and timing and also taught the implementation of a large design using a modular approach.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2\*2 Design :**

module Lab61 (int, out, Clock, Reset);

input signed [47:0] int;

input Clock, Reset;

output signed [23:0] out;

reg signed [23:0] out;

reg signed [11:0] A,B,C,D;

reg signed [23:0] temp1, temp2, temp3;

always @ (posedge Clock or negedge Reset) begin

if (~Reset) begin

out = 'b0;

end

else begin

A = int[11:0];

B = int[35:24];

C = int[23:12];

D = int[47:36];

temp1 = A \* D;

temp2 = C \* B;

temp3 = temp1 - temp2;

out = temp3;

end

end

endmodule

//Example of how to use the integer divide block from Altera

//synopsys translate\_off

`timescale 1 ps / 1 ps

//synopsys translate\_on

module divide (Clock, denom, numer, quotient, remain);

input Clock;

input [23:0] denom;

input [23:0] numer;

output [23:0] quotient;

output [23:0] remain;

wire [23:0] sub\_wire0;

wire [23:0] sub\_wire1;

wire [23:0] quotient = sub\_wire0[23:0];

wire [23:0] remain = sub\_wire1[23:0];

lpm\_divide lpm\_divide\_component (.denom (denom), .Clock (Clock), .numer (numer),

.quotient (sub\_wire0), .remain (sub\_wire1), .aclr (1'b0), .clken (1'b1));

defparam

lpm\_divide\_component.lpm\_drepresentation = "SIGNED",

lpm\_divide\_component.lpm\_hint = "LPM\_REMAINDERPOSITIVE=FALSE",

lpm\_divide\_component.lpm\_nrepresentation = "SIGNED",

lpm\_divide\_component.lpm\_pipeline = 1,

lpm\_divide\_component.lpm\_type = "LPM\_DIVIDE",

lpm\_divide\_component.lpm\_widthd = 24,

lpm\_divide\_component.lpm\_widthn = 24;

endmodule

module main ( outy1, outy2, Clock, Reset);

////////////////////////////////////////////////

input Clock, Reset;

output wire signed [23:0] outy1, outy2;

//input signed [11:0] intx1, intx2, intx3, intx4, intx5, intx6;

////////////////////////////////////////////////

reg [47:0] M1, M2, M0;

wire signed [23:0] Det0, Det1, Det2, R1, R2;

//reg [11:0] x1, x2, x3, x4, x5, x6;

reg signed [11:0] intx1, intx2, intx3, intx4, intx5, intx6;

/////////////////////////////////////////////////

always @ (posedge Clock or negedge Reset) begin

if (~Reset) begin

M0 = 'b0;

M1 = 'b0;

M2 = 'b0;

intx1 = 12'd1;

intx2 = 12'd2;

intx3 = 12'd3;

intx4 = 12'd4;

intx5 = 12'd5;

intx6 = 12'd6;

end

else begin

M0 = {intx1, intx4, intx2, intx5};

M1 = {intx3, intx6, intx2, intx5};

M2 = {intx1, intx4, intx3, intx6};

end

end

Lab61 det0 (.int(M0), .out(Det0), .Clock(Clock), .Reset(Reset));

Lab61 det1 (.int(M1), .out(Det1), .Clock(Clock), .Reset(Reset));

Lab61 det2 (.int(M2), .out(Det2), .Clock(Clock), .Reset(Reset));

divide first ( .Clock(Clock), .denom(Det0), .numer(Det1), .quotient(outy1), .remain(R1) );

divide second ( .Clock(Clock), .denom(Det0), .numer(Det2), .quotient(outy2), .remain(R2) );

endmodule

module abd (KEY, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7);

main cramer( .outy1(outy1), .outy2(outy2), .Clock(Clock), .Reset(Reset));

input [1:0] KEY;

output [0:6] HEX0;

output [0:6] HEX1;

output [0:6] HEX2;

output [0:6] HEX3;

output [0:6] HEX4;

output [0:6] HEX5;

output [0:6] HEX6;

output [0:6] HEX7;

wire [23:0] outy1, outy2;

//wire [12:0] intx1, intx2, intx3, intx4, intx5, intx6;

wire Clock, Reset;

assign Clock = KEY[0];

assign Reset = KEY[1];

hex\_7seg ha0(.w(outy1[3:0]), .seg(HEX0));

hex\_7seg ha1(.w(outy1[7:4]), .seg(HEX1));

hex\_7seg ha2(.w(outy1[11:8]), .seg(HEX2));

hex\_7seg ha3(.w(outy1[15:12]), .seg(HEX3));

hex\_7seg ha4(.w(outy2[3:0]), .seg(HEX4));

hex\_7seg ha5(.w(outy2[7:4]), .seg(HEX5));

hex\_7seg ha6(.w(outy2[11:8]), .seg(HEX6));

hex\_7seg ha7(.w(outy2[15:12]), .seg(HEX7));

endmodule

**Testbench:**

module testbech1;

reg Clock, Reset;

wire [23:0] outy1, outy2;

reg [11:0] intx1, intx2, intx3, intx4, intx5, intx6;

main baba( .intx1(intx1), .intx2(intx2), .intx3(intx3), .intx4(intx4), .intx5(intx5), .intx6(intx6), .outy1(outy1), .outy2(outy2), .Clock(Clock), .Reset(Reset));

initial

begin

Clock = 1'b0;

forever #20 Clock = ~Clock;

end

initial

begin

intx1 = 12'd1;

intx2 = 12'd2;

intx3 = 12'd3;

intx4 = 12'd4;

intx5 = 12'd5;

intx6 = 12'd6;

end

reg [47:0] M = {intx1, intx4, intx2, intx5};

wire signed [23:0] B;

Lab61 MM (.int(M), .out(B), .Clock(Clock), .Reset(Reset))

initial

begin

if( B == 24'b0)

$display("Error: Determinant = 0");

else

$display("y1 = %h, y2 = %h", outy1, outy2);

end

endmodule

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**3\*3 Design:**

module Lab62 (int, out, Clock, Reset);

input signed [107:0] int;

input Clock, Reset;

output signed [35:0] out;

reg signed [35:0] out;

reg signed [11:0] A, B, C, D, E, F, G, H, I;

reg signed [23:0] temp1, temp2, temp3, temp5, temp6, temp7, temp9, temp10, temp11;

reg signed [35:0] temp4, temp8, temp12, temp13;

always @ (posedge Clock or negedge Reset) begin

if (~Reset) begin

out = 'b0;

end

else begin

A = int[11:0];

B = int[47:36];

C = int[83:72];

D = int[23:12];

E = int[59:48];

F = int[95:84];

G = int[35:24];

H = int[71:60];

I = int[107:96];

temp1 = E \* I;

temp2 = H \* F;

temp3 = temp1 - temp2;

temp4 = A \* temp3;

temp5 = D \* I;

temp6 = G \* F;

temp7 = temp6 - temp5;

temp8 = B \* temp7;

temp9 = D \* H;

temp10 = G \* E;

temp11 = temp9 - temp10;

temp12 = C \* temp11;

temp13 = temp4 + temp8 + temp12;

out = temp13;

end

end

endmodule

module main ( outy1, outy2, outy3, Clock, Reset);

////////////////////////////////////////////////

input Clock, Reset;

output wire signed [35:0] outy1, outy2, outy3;

////////////////////////////////////////////////

reg [107:0] M1, M2, M0, M3;

wire signed [35:0] Det0, Det1, Det2, Det3, R1, R2, R3;

//reg [11:0] x1, x2, x3, x4, x5, x6;

reg signed [11:0] intx1, intx2, intx3, intx4, intx5, intx6, intx7, intx8, intx9, intx10, intx11, intx12;

/////////////////////////////////////////////////

/////////////////////////////////////////////////

always @ (posedge Clock or negedge Reset) begin

if (~Reset) begin

M0 = 'b0;

M1 = 'b0;

M2 = 'b0;

M3 = 'b0;

intx1 = 12'd1;

intx2 = 12'd4;

intx3 = 12'd4;

intx4 = 12'd7;

intx5 = 12'd2;

intx6 = 12'd4;

intx7 = 12'd7;

intx8 = 12'd3;

intx9 = 12'd8;

intx10 = 12'd1;

intx11 = 12'd13;

intx12 = 12'd11;

end

else begin

M0 = {intx1, intx5, intx9, intx2, intx6, intx10, intx3, intx7, intx11};

M1 = {intx4, intx8, intx12, intx2, intx6, intx10, intx3, intx7, intx11};

M2 = {intx1, intx5, intx9, intx4, intx8, intx12, intx3, intx7, intx11};

M3 = {intx1, intx5, intx9, intx2, intx6, intx10, intx4, intx8, intx12};

end

end

Lab62 det0 (.int(M0), .out(Det0), .Clock(Clock), .Reset(Reset));

Lab62 det1 (.int(M1), .out(Det1), .Clock(Clock), .Reset(Reset));

Lab62 det2 (.int(M2), .out(Det2), .Clock(Clock), .Reset(Reset));

Lab62 det3 (.int(M3), .out(Det3), .Clock(Clock), .Reset(Reset));

divide first ( .Clock(Clock), .denom(Det0), .numer(Det1), .quotient(outy1), .remain(R1) );

divide second ( .Clock(Clock), .denom(Det0), .numer(Det2), .quotient(outy2), .remain(R2) );

divide third ( .Clock(Clock), .denom(Det0), .numer(Det3), .quotient(outy3), .remain(R3) );

endmodule

module abd (KEY, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7);

main cramer( .outy1(outy1), .outy2(outy2), .outy3(outy3), .Clock(Clock), .Reset(Reset));

input [1:0] KEY;

output [0:6] HEX0;

output [0:6] HEX1;

output [0:6] HEX2;

output [0:6] HEX3;

output [0:6] HEX4;

output [0:6] HEX5;

output [0:6] HEX6;

output [0:6] HEX7;

wire [23:0] outy1, outy2, outy3;

//wire [12:0] intx1, intx2, intx3, intx4, intx5, intx6;

wire Clock, Reset;

assign Clock = KEY[0];

assign Reset = KEY[1];

hex\_7seg ha0(.w(outy1[3:0]), .seg(HEX0));

hex\_7seg ha1(.w(outy1[7:4]), .seg(HEX1));

hex\_7seg ha2(.w(outy1[11:8]), .seg(HEX2));

hex\_7seg ha3(.w(outy2[3:0]), .seg(HEX3));

hex\_7seg ha4(.w(outy2[7:4]), .seg(HEX4));

hex\_7seg ha5(.w(outy2[11:8]), .seg(HEX5));

hex\_7seg ha6(.w(outy3[3:0]), .seg(HEX6));

hex\_7seg ha7(.w(outy3[7:4]), .seg(HEX7));

endmodule

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4\*4 Design:**

module La6EC (int, out, Clock, Reset);

input signed [191:0] int;

input Clock, Reset;

output reg signed [47:0] out;

reg signed [11:0] A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P;

reg signed [47:0] temp1;

reg signed [107:0] temp4, temp8, temp12, temp16;

wire signed [35:0] temp5, temp9, temp13, temp17;

always @ (posedge Clock or negedge Reset) begin

if (~Reset) begin

out = 'b0;

end

else begin

A = int[11:0];

B = int[59:48];

C = int[107:96];

D = int[155:144];

E = int[23:12];

F = int[71:60];

G = int[119:108];

H = int[167:156];

I = int[35:24];

J = int[83:72];

K = int[131:120];

L = int[179:168];

M = int[47:36];

N = int[95:84];

O = int[143:132];

P = int[191:180];

temp4 = {F, J, N, G, K, O, H, L, P};

temp8 = {E, I, M, G, K, O, H, L, P};

temp12 = {E, I, M, F, J, N, H, L, P};

temp16 = {E, I, M, F, J, N, G, K, O};

temp1 = (A \* temp5) - (B \* temp9) + (C \* temp13) - (D \* temp17);

out = temp1;

end

end

La62 lab621(.int(temp4), .out(temp5), .Clock(Clock), .Reset(Reset));

La62 lab622(.int(temp8), .out(temp9), .Clock(Clock), .Reset(Reset));

La62 lab623(.int(temp12), .out(temp13), .Clock(Clock), .Reset(Reset));

La62 lab624(.int(temp16), .out(temp17), .Clock(Clock), .Reset(Reset));

endmodule

module main ( outy1, outy2, outy3, out4, Clock, Reset);

////////////////////////////////////////////////

input Clock, Reset;

output wire signed [47:0] outy1, outy2, outy3, out4;

////////////////////////////////////////////////

reg [191:0] M1, M2, M0, M3, M4;

wire signed [47:0] Det0, Det1, Det2, Det3, Det4, R1, R2, R3, R4;

reg signed [11:0] intx1, intx2, intx3, intx4, intx5, intx6, intx7, intx8, intx9, intx10, intx11, intx12, intx13, intx14, intx15, intx16;

reg signed [11:0] intx17, intx18, intx19, intx20;

////////////////////////////////////////////////

always @ (posedge Clock or negedge Reset) begin

if (~Reset) begin

M0 = 'b0;

M1 = 'b0;

M2 = 'b0;

M3 = 'b0;

M4 = 'b0;

intx1 = 12'd1;

intx2 = 12'd4;

intx3 = 12'd4;

intx4 = 12'd7;

intx5 = 12'd2;

intx6 = 12'd4;

intx7 = 12'd7;

intx8 = 12'd3;

intx9 = 12'd8;

intx10 = 12'd1;

intx11 = 12'd13;

intx12 = 12'd16;

intx13 = 12'd24;

intx14 = 12'd34;

intx15 = 12'd3;

intx16 = 12'd8;

intx17 = 12'd6;

intx18 = 12'd3;

intx19 = 12'd2;

intx20 = 12'd20;

end

else begin

M0 = {intx1, intx6, intx11, intx16, intx2, intx7, intx12, intx17, intx3, intx8, intx13, intx18, intx4, intx9, intx14, intx19 };

M1 = {intx5, intx10, intx15, intx20, intx2, intx7, intx12, intx17, intx3, intx8, intx13, intx18, intx4, intx9, intx14, intx19 };

M2 = {intx1, intx6, intx11, intx16, intx5, intx10, intx15, intx20, intx3, intx8, intx13, intx18, intx4, intx9, intx14, intx19 };

M3 = {intx1, intx6, intx11, intx16, intx2, intx7, intx12, intx17, intx5, intx10, intx15, intx20, intx4, intx9, intx14, intx19 };

M4 = {intx1, intx6, intx11, intx16, intx2, intx7, intx12, intx17, intx3, intx8, intx13, intx18, intx5, intx10, intx15, intx20 };

end

end

La6EC det0 (.int(M0), .out(Det0), .Clock(Clock), .Reset(Reset));

La6EC det1 (.int(M1), .out(Det1), .Clock(Clock), .Reset(Reset));

La6EC det2 (.int(M2), .out(Det2), .Clock(Clock), .Reset(Reset));

La6EC det3 (.int(M3), .out(Det3), .Clock(Clock), .Reset(Reset));

La6EC det4 (.int(M4), .out(Det4), .Clock(Clock), .Reset(Reset));

divide first ( .Clock(Clock), .denom(Det0), .numer(Det1), .quotient(outy1), .remain(R1) );

divide second ( .Clock(Clock), .denom(Det0), .numer(Det2), .quotient(outy2), .remain(R2) );

divide third ( .Clock(Clock), .denom(Det0), .numer(Det3), .quotient(outy3), .remain(R3) );

divide fourth ( .Clock(Clock), .denom(Det0), .numer(Det4), .quotient(outy4), .remain(R4) );

endmodule

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_