**Lab 4 Report Hardik J Patel**

**EEC 180B 999121498**

**Timing and Power Analysis Tutorial**

**Objective**

The purpose of this lab was to learn the procedures of performing **gate level simulation** using ModelSim and **power analysis** using Altera’s Quartus II **PowerePlay Power Analyser**, by anaysing the power and timing consumption of the Booth multiplier from lab 3.

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**Design and Test Procedure**

Part 1: In this part of the the lab, were perform gate level simulation of the Booth multiplier using ModelSim. The prelab asks to revise the testbench from lab 3 to work it with the synthesizable version of the code. Since in my case, the booth multiplier was done in two parts, first was the actual code and the second was the code that assigned the variables to switches and keys, I did not need to revise the testbench. *The testbench is attached at the end.* The steps were followed closely to complete this simulation. The delay from the rising edge of the clock to “Done” signal was minimal and difficult to understand, but after careful observation it turned out to be in the order of 2 ps. The functional simulation only checks the logic operation of the circuit and does not take into account the delay through the internal logic, but the timing simulation takes the delay associated with the logic elements.

Part 2: For this part a different approach was taken to realise the timing performance of the design, using the **TimeQuest Timing Analyser** tool. For this part too the steps in the manual were followed closely. **t**pd represents the propagational delay and **t**co is the combinational delay. Critical path of this circuit was determined to be from Mplier[4] to nxtB[5], 7.395 and the maximum frequency is 40 MHz. The critical path by intuitions seems consistent with the design. With the frequency at 80 MHz, critical path is Mplier[1] to nxtB[2], 7.087.

Part 3: For the PowerPlay Power Analyser, the booth multiplier is modified to a 16 bit multiplier and both the multipliers are compared in terms of power consumption. The 16 bit multiplier consumes more power that the 8 bit multiplier. *The 16 bit multiplier is attached at the end.*

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**Conclusion**

This lab taught the procedure and implementation of timing and power analysis in ModelSim and Quartus.

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**16 Bit Booth Multiplier**

module boothmul(Clk, Resetn, Start, Mplier, Mcand, Done, Product);

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

output [31:0] Product;

output Done;

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

input [15:0] Mcand, Mplier;

input Clk, Start, Resetn;

reg [1:0] state;

reg [1:0] nextstate;

localparam state3 = 3'd3;

localparam state2 = 3'd2;

localparam state1 = 3'd1;

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

reg [16:0] A, B, nxtA, nxtB;

reg [15:0] C, nxtC;

reg [4:0] count;

reg Done = 1'b0;//, a, b, c;

reg inc\_cnt;

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

always @(posedge Clk or negedge Resetn) begin

if (~Resetn)

state <= state1;

else

state <= nextstate;

end

//initial count = 'b0;

always @ (posedge Clk or negedge Resetn) begin

if (!Resetn)

count <= 5'b0;

else if (inc\_cnt)

count <= count + 1'b1;

else

count <= count;

end

// next state logic

always @ (\*)

begin : controller

inc\_cnt = 1'b0;

nextstate = state1;

case (state)

state1: begin

if (Start && ~Done)

nextstate = state2;

else

nextstate = state1;

end

state2: begin

if((B[0]==0 && B[1]==0) | (B[0]==1 && B[1]==1)) begin

if(count > 8)

nextstate = state1;

else begin

//count = count+ 1'b1;

inc\_cnt = 1'b1;

nextstate = state2;

end

end

if((B[0]==0 && B[1]==1))

nextstate = state3 ;

if((B[0]==1 && B[1]==0))

nextstate = state3 ;

end

state3:

if(count > 16)

nextstate = state1;

else begin

inc\_cnt = 1'b1;

nextstate = state2;

end

default:

nextstate = state1;

endcase

end

// registers update logic

always @ (posedge Clk or negedge Resetn)

begin

if (!Resetn) begin

A = 'b0;

B = 'b0;

C = 'b0;

end

else begin

A <= nxtA;

B <= nxtB;

C <= nxtC;

end

end

// Registers input logic

always @ (\*) begin

case(state)

state1: begin

nxtA = 'b0;

nxtC = Mcand;

nxtB = {Mplier[15:0], 1'b0};

if (count > 16)

Done = 1'b1;

nxtB[0] = 1'b0;

end

state2: begin

if((B[0]==0 && B[1]==0) | (B[0]==1 && B[1]==1))

begin

nxtA = {A[16], A[16:1]};

nxtB = {B[16], B[16:1]};

end

else if((B[0]==0 && B[1]==1))

nxtA = A + (~C + 1'b1) ;

else if((B[0]==1 && B[1]==0))

nxtA = A + C;

end

state3: begin

nxtA = {A[16], A[16:1]};

nxtB = {B[16], B[16:1]};

end

endcase

end

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

assign Product = {A, B};

endmodule

**Testbench**

`timescale 1 ns/ 1 ns

module tb\_booth;

parameter n=8; // n-bit Booth multiplier

parameter num\_vectors=8;

reg Clock, Resetn, Start;

wire Done;

reg [n-1:0] Mplier, Mcand;

wire [n+n-1:0] Product;

reg [n+n-1:0] vectors [0:num\_vectors-1];

integer i;

boothmul UUT (.Clk(Clock), .Resetn(Resetn), .Start(Start), .Mplier(Mplier), .Mcand(Mcand),

.Done(Done), .Product(Product));

initial // Clock generator

begin

Clock = 1'b0;

forever #20 Clock = ~Clock; // Clock period = 40 ns

end

initial // Test stimulus

begin

Resetn = 1'b0; // synchronous reset of state machine

Start = 1'b0; // set Start to �false�

#80 Resetn = 1'b1; // reset low for 2 Clock periods

$readmemb ("testvecs.txt", vectors); // read testvecs file

for (i=0; i<num\_vectors; i=i+1) begin

{Mplier, Mcand} = vectors[i]; // load Mplier, Mcand

#20 Start = 1'b1; // Start = �true�

#80 Start = 1'b0; // After 2 clock cycles, reset Start

wait (Done==1);

wait (Done==0);

$display("Mplier=%h, Mcand=%h, Product=%h",Mplier,Mcand,Product);

end

end

endmodule