**LAB 9**

**-Daksh Thapar 2018137**

**- Tanmaya Gupta 2018200**

module top\_module(

input aclk,

input aresetn,

input xa\_valid,

input [31:0] xa,

output xa\_ready,

input ta\_valid,

input [31:0] ta,

output ta\_ready,

input na\_valid,

input [31:0] na,

output na\_ready,

// output [31:0] out1,

// input out\_ready,

output [31:0] out,

output out\_valid

// input n\_ready,

// output n\_valid

);

// wire out\_valid;

wire out\_ready;

wire [31:0] x;

wire x\_ready,x\_valid;

floating\_point\_float first (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(xa\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(xa\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(xa), // input wire [31 : 0] s\_axis\_a\_tdata

.m\_axis\_result\_tvalid(x\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(x\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(x) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] t;

wire t\_ready,t\_valid;

floating\_point\_float second (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(ta\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(ta\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(ta), // input wire [31 : 0] s\_axis\_a\_tdata

.m\_axis\_result\_tvalid(t\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(t\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(t) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] n;

wire n\_ready,n\_valid;

floating\_point\_float fourth (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(na\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(na\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(na), // input wire [31 : 0] s\_axis\_a\_tdata

.m\_axis\_result\_tvalid(n\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(n\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(n) // output wire [31 : 0] m\_axis\_result\_tdata

);

reg [31:0] two = 32'b01000000000000000000000000000000;

reg [31:0] three =32'b01000000010000000000000000000000;

wire [31:0] t1;

wire t1\_ready,t1\_valid;

assign t1 = t;

wire [31:0] x\_by\_t;

wire x\_by\_t\_valid,x\_by\_t\_ready;

floating\_point\_divide1 div\_1st (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(x\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(x\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(x), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(t\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(t\_ready), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(t1), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(x\_by\_t\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(x\_by\_t\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(x\_by\_t) // output wire [31 : 0] m\_axis\_result\_tdata

);

//wire [31:0] out1;

////wire out1\_valid,out1\_ready;

//assign out1 = x\_by\_t;

//assign out1\_valid = x\_by\_t\_valid;

//assign out1\_ready = x\_by\_t\_ready;

wire m1\_valid;

wire m1\_ready;

wire [31:0] m1;

assign m1 = x\_by\_t;

wire mt\_valid;

wire mt\_ready;

wire [31:0] mt;

assign mt = x\_by\_t;

wire m\_c\_valid;

wire m\_c\_ready;

wire [31:0] m\_c;

assign m\_c = x\_by\_t;

wire m\_3\_valid;

wire m\_3\_ready;

wire [31:0] m\_3;

assign m\_3 = x\_by\_t;

wire [31:0] m2;

wire m2\_ready,m2\_valid;

floating\_point\_mul mul1 (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(x\_by\_t\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(mt\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(mt), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(x\_by\_t\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(x\_by\_t\_ready), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(m1), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(m2\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(m2\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(m2) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] sub;

wire sub\_ready,sub\_valid;

floating\_point\_sub subing (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(x\_by\_t\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(m\_c\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(m\_c), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(m2\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(m2\_ready), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(m2), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(sub\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(sub\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(sub) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] logn;

wire logn\_ready,logn\_valid;

floating\_point\_log log (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(n\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(n\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(n), // input wire [31 : 0] s\_axis\_a\_tdata

.m\_axis\_result\_tvalid(logn\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(logn\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(logn) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] logn\_t;

wire logn\_t\_ready,logn\_t\_valid;

floating\_point\_divide1 div\_2nd (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(logn\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(logn\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(logn), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(t\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(t1\_ready), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(t), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(logn\_t\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(logn\_t\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(logn\_t) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire logn\_t1\_valid;

assign logn\_t1\_valid = logn\_t\_valid;

wire logn\_t1\_ready;

wire [31:0] logn\_t1;

assign logn\_t1 = logn\_t;

wire [31:0] two1;

assign two1 = two;

wire [31:0] three1;

assign three1 = three;

wire [31:0] twologn\_t;

wire twologn\_t\_valid,twologn\_t\_ready;

wire logging;

floating\_point\_mul mul2 (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(logn\_t1\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(logn\_t1\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(logn\_t1), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(logn\_t1\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(logging), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(two1), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(twologn\_t\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(twologn\_t\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(twologn\_t) // output wire [31 : 0] m\_axis\_result\_tdata

);

//assign out1=logn\_t1;

wire logn\_t2\_valid;

assign logn\_t2\_valid = logn\_t\_valid;

wire logn\_t2\_ready;

wire [31:0] logn\_t2;

assign logn\_t2 = logn\_t;

wire [31:0] threelogn\_t;

wire threelogn\_t\_valid,threelogn\_t\_ready;

wire logging1;

floating\_point\_mul mul3 (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(logn\_t2\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(logn\_t2\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(logn\_t2), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(logn\_t2\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(logging1), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(three1), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(threelogn\_t\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(threelogn\_t\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(threelogn\_t) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] t2;

wire t2\_ready,t2\_valid;

assign t2 = t;

assign t2\_valid=t\_valid;

wire [31:0] twologn\_tv;

wire twologn\_tv\_valid,twologn\_tv\_ready;

floating\_point\_mul mul4 (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(twologn\_t\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(twologn\_t\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(twologn\_t), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(sub\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(sub\_ready), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(sub), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(twologn\_tv\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(twologn\_tv\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(twologn\_tv) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] sqrt;

wire sqrt\_ready,sqrt\_valid;

floating\_point\_sqrt2 sqrt2 (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(twologn\_tv\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(twologn\_tv\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(twologn\_tv), // input wire [31 : 0] s\_axis\_a\_tdata

.m\_axis\_result\_tvalid(sqrt\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(sqrt\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(sqrt) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] r1;

wire r1\_valid,r1\_ready;

wire temp;

floating\_point\_1 add (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(sqrt\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(sqrt\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(sqrt), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(x\_by\_t\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(temp), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(x\_by\_t), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(r1\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(r1\_ready), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(r1) // output wire [31 : 0] m\_axis\_result\_tdata

);

wire [31:0] r2;

wire r2\_valid,r2\_ready;

floating\_point\_1 add11 (

.aclk(aclk), // input wire aclk

.aresetn(aresetn), // input wire aresetn

.s\_axis\_a\_tvalid(r1\_valid), // input wire s\_axis\_a\_tvalid

.s\_axis\_a\_tready(r1\_ready), // output wire s\_axis\_a\_tready

.s\_axis\_a\_tdata(r1), // input wire [31 : 0] s\_axis\_a\_tdata

.s\_axis\_b\_tvalid(threelogn\_t\_valid), // input wire s\_axis\_b\_tvalid

.s\_axis\_b\_tready(threelogn\_t\_ready), // output wire s\_axis\_b\_tready

.s\_axis\_b\_tdata(threelogn\_t), // input wire [31 : 0] s\_axis\_b\_tdata

.m\_axis\_result\_tvalid(out\_valid), // output wire m\_axis\_result\_tvalid

.m\_axis\_result\_tready(1'b1), // input wire m\_axis\_result\_tready

.m\_axis\_result\_tdata(out) // output wire [31 : 0] m\_axis\_result\_tdata

);

Endmodule

module tester(

);

reg clk = 0;

reg s\_aresetn = 0;

reg [3:0] inform = 0;

reg data\_valid = 0;

wire [31:0] Q,X,T,N;

wire q\_valid;

always #2 clk = !clk ;

ready\_valid u1(

.clk(clk),

.s\_aresetn( s\_aresetn),

.inform(inform),

.data\_valid(data\_valid),

// .out\_valid(data\_valid),

.X(X),

.T(T),

.N(N),

.Q(Q),

.q\_valid(q\_valid)

);

initial

begin

#100 s\_aresetn <= 1;

#10 inform <= 4'd0;

data\_valid <= 1;

#4 data\_valid <= 0;

end

endmodule

module ready\_valid(

input clk,

input s\_aresetn,

input [3:0] inform,

input data\_valid,

output reg [31:0] X=0,

output reg [31:0] T=0,

output reg [31:0] N=0,

output wire [31:0] Q,

output wire q\_valid

);

reg data\_valid\_prev=0 ;

always@(posedge clk) begin

if (data\_valid\_prev == 0&& data\_valid==1 ) begin

N=N+1;

if (inform[3]==1'b0) begin

X=32'd1;

T=32'd1;

N=32'd4;

end

else if (inform[0]==1'b1) begin

T = T+1;

X = X + inform[1];

end

end

end

always@(posedge clk) begin

data\_valid\_prev <= data\_valid;

end

reg valid\_x = 0;

reg valid\_t = 0;

reg valid\_n = 0;

wire ready\_x, ready\_t, ready\_n;

reg [1:0] count\_valid\_x = 0;

reg [1:0] count\_valid\_t = 0;

reg [1:0] count\_valid\_n = 0 ;

reg [1:0] count\_valid\_x\_reg =0;

reg [1:0] count\_valid\_t\_reg =0;

reg [1:0]count\_valid\_n\_reg=0;

always@(posedge clk) begin

count\_valid\_x\_reg = count\_valid\_x;

end

always@(posedge clk) begin

count\_valid\_t\_reg = count\_valid\_t;

end

always@(posedge clk) begin

count\_valid\_n\_reg = count\_valid\_n;

end

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

always@(\*) begin

if (ready\_x == 1'b1 && valid\_x== 1'b1)

count\_valid\_x = 2;

else if (count\_valid\_x\_reg==2)

count\_valid\_x=3;

end

always@(\*) begin

if (data\_valid\_prev==1'b0 && data\_valid == 1'b1)

valid\_x=1;

else if (count\_valid\_x\_reg==2)

valid\_x=0;

end

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

always@(\*) begin

if (ready\_t == 1'b1 && valid\_t== 1'b1)

count\_valid\_t = 2;

else if (count\_valid\_t\_reg==2)

count\_valid\_t=3;

end

always@(\*) begin

if (data\_valid\_prev==1'b0 && data\_valid == 1'b1)

valid\_t=1;

else if (count\_valid\_t\_reg==2)

valid\_t=0;

end

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

always@(\*) begin

if (ready\_n == 1'b1 && valid\_n== 1'b1)

count\_valid\_n = 2;

else if (count\_valid\_n\_reg==2)

count\_valid\_n=3;

end

always@(\*) begin

if (data\_valid\_prev==1'b0 && data\_valid == 1'b1)

valid\_n=1;

else if (count\_valid\_n\_reg==2)

valid\_n=0;

end

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

top\_module a1(

.aclk(clk),

.aresetn(s\_aresetn),

.xa\_valid(valid\_x),

.xa(X),

.xa\_ready(ready\_x),

.ta\_valid(valid\_t),

.ta(T),

.ta\_ready(ready\_t),

.na\_valid(valid\_n),

.na(N),

.na\_ready(ready\_n),

.out\_valid(q\_valid),

.out(Q)

);

endmodule