Nem Negash

Prof. Mohsenin

CMPE415

3/14/2021

Homework 4: Phase 1 Report

Statement of Completeness  
 My project works to 95% completion. The only issue I have is with the divider generator. My output for divider generator function doesn’t actually compute the divide but rather just take the value of ‘B’ with a 200ns delay. I still don’t know what is causing that.

ALU functions

**c\_addsub\_0**: This Adder takes in 2 16-bit values and a clk signal. It completes binary addition and produces a single 16-bit output(S).

**c\_addsub\_1**: This subtractor takes in 2 16-bit values and a clk signal. It completes binary subtraction and produces a single 16-bit output(S) mapped to D port.

**multi\_gen\_0**: This multiplier takes in 2 16-bit values and a clk signal. It completes binary multiplication arithmetic by implementing 15 16-bit adders internally and produces a 32-bit output P.

**and16\_0**: This is a simple 16-bit and logic gate. It takes 2 16-bit values, ands the values with each other and produces a single 16-bit output (AND\_OUT).

**or16\_0**: This is a simple 16-bit or logic gate. It takes 2 16-bit values, ors the values with each other and produces a single 16-bit output(OR\_OUT).

**xor16\_0**: This is a simple 16-bit or logic gate. It takes 2 16-bit values, xors the values with each other and produces a single 16-bit output(XOR\_OUT). It probably just inverts the or operation.

**div\_gen\_0**: Takes in two 16-bit inputs and two 1-bit input valid inputs. Supposedly performs binary division with the given inputs and outputs the value and a 1-bit output valid output.

Testbench Simulation

**Testbench:**

`timescale 1ns / 1ps

module ALU\_tb;

reg [15:0]A;

reg [15:0]B;

reg AVALID;

reg BVALID;

reg clk;

wire [15:0] S; //c\_add\_S;

wire [15:0] D; //c\_sub\_S;

wire [15:0] P;

wire [15:0] AND\_OUT; //and

wire [15:0] OR\_OUT; //or

wire [15:0] XOR\_OUT; //xor

wire [31:0] Q;

wire QVALID;

ALU alu\_uut(.A(A),

.AND\_OUT(AND\_OUT),

.AVALID(AVALID),

.B(B),

.BVALID(BVALID),

.D(D),

.OR\_OUT(OR\_OUT),

.P(P),

.Q(Q),

.QVALID(QVALID),

.S(S),

.XOR\_OUT(XOR\_OUT),

.clk(clk));

always #5 clk = (clk ^ 1'b1);

initial begin

clk = 0;

#200

AVALID = 1;

BVALID = 0;

A = 16'b0000000001001111; // 5

B = 16'b0000000000100101;

#20

BVALID = 0;

#200

BVALID = 0;

#20

A = 16'b0000000000101101; // 5

B = 16'b0000000000011001;

#20

BVALID = 0;

#200

BVALID = 0;

#20

A = 16'b0000000100101111; // 5

B = 16'b0000000000000101;

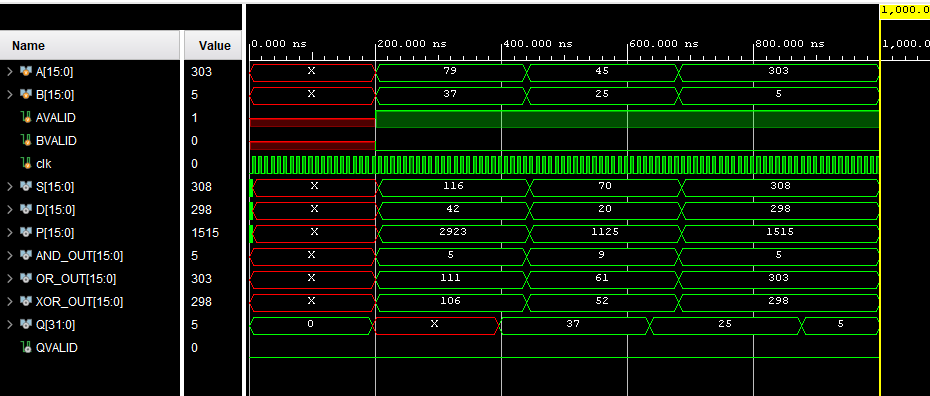
#20

BVALID = 0;

end

endmodule

**Simulation waveform:**



Block Diagram

