DESIGNING MAC WITH VERILOG.

MACHINE LEARNING ARCHITECTURE

Name:Shreya Keshari
Roll no.: 234102207
Submitted To:Dr Gourav Trivedi
Professor, Electrical and Electronics Engineering Department

Indian Institute Of Technology,Guwahati

05,February 2024

Contents

List of Figures

1 OBJECTIVE

Design and implement an IEEE754 floating-point MAC module using verilog.

2 Theory:

2.1 IEEE 754 floating point number representation:

In this experiment, the objective is to design an IEEE754 floating point multiplier. Floating point number mean the decimal point can be shifted to right or left of the fixed number, hence the name floating point. The floating point representation gives greater precision . The floating point numbers can be expressed using the IEEE-754 standard which defines a set of floating point data formats, single precision consisting of 32 bits and double precision consisting of 64 bits. The Single Precision Floating Point Multiplier consists of 32 bits in which the sign bit is represented by 1 bit, the exponent bit is represented by 8 bits, and the mantissa bit is of 23 bits. The Double Precision Floating Point Multiplier consists of 8 bytes in which the sign bit is represented by MSB bit, the exponent bit is represented by 11bits, and the mantissa hits are of 52 bits. Commonly used format is the single precision format of IEEE 754:

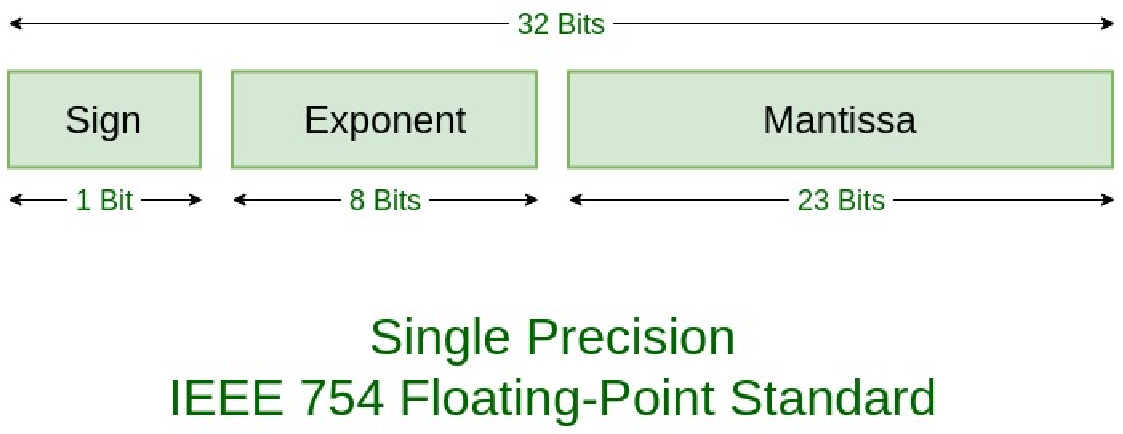


Figure 1: IEEE 754 single precision format

Commonly used format is the double precision format of IEEE 754:

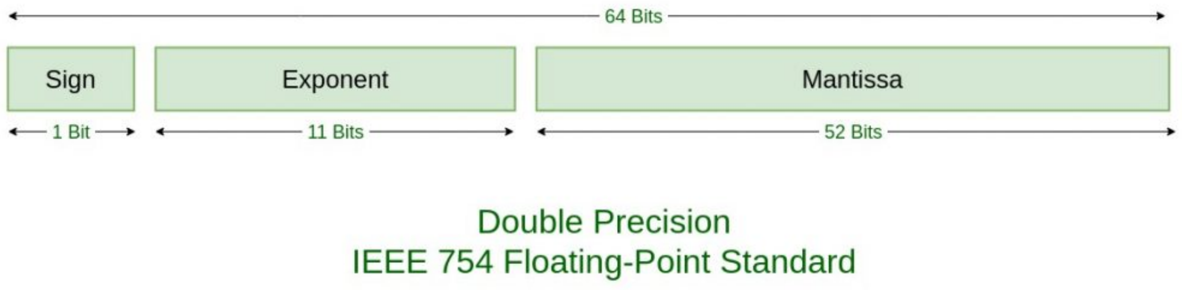


Figure 2: IEEE 754 double precision format

3 MAC Unit:

In computing, especially digital signal processing, the multiply–accumulate (MAC) or multiply-add (MAD) operation is a common step that computes the product of two numbers and adds that product to an accumulator. The hardware unit that performs the operation is known as a multiplier–accumulator (MAC unit); the operation itself is also often called a MAC or a MAD operation. The MAC operation modifies an accumulator a.The block diagram of MAC Unit is given as:

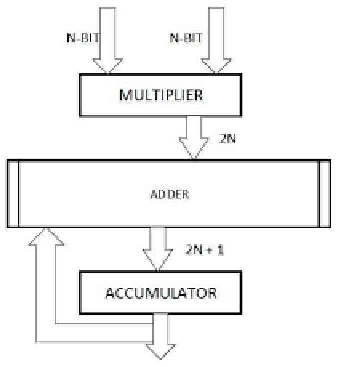


Figure 3: MAC Unit block diagram

4 Design approach:

The design is IEEE 754 MAC unit. The numbers has Sign is always 1 bit. Exponent is 8 bit part and matissaa is 23bit.

5 Verilog Code:

module MAC$_final(clk, A, B, out); inputclk; input[31:0]A, B; output[31:0]out;$

wire [31:0]res1,res2;

float$_point_mulm1(clk, A, B, res1); accum_rega1(out, res2); float_point_addf1(clk, res1, res2, out);$

endmodule

module float$_point_mul(clk, A, B, out); inputclk; input[31:0]A, B; outputreg[31:0]out; regs1, s2, s_out; reg[7:0]e1, e2, e_out; reg[22:0]m1, m2, m_out; regzero_out; reg[47:0]mul; regx;$

always@(posedge clk) begin s1,e1,m1=A[31],A[30:23],A[22:0]; s2,e2,m2=B[31],B[30:23],B[22:0]; zero$_out = (e1, m1 == 0)|(e2, m2 == 0); s_out = s1^s2; mul = 1'b1, m1 * 1'b1, m2; m_out = (mul[47] == 1)?mul[46 : 24] : mul[45 : 23]; x = (mul[47] == 1)?1'b1 : 1'b0; e_out = e1 + e2 + x - 127;$

out = (zero$_out == 1)?0 : s_out, e_out, m_out; endendmodule$

module float$_point_add(clk, A, B, out); inputclk; input[31:0]A, B; outputreg[31:0]out = 0; regs1, s2, s_final; reg[7:0]e1, e2, e_final; reg[22:0]m1, m2; reg[7:0]exp_diff = 0; reg[24:0]m_final = 0;$

reg count=0; reg [4:0]index=23; reg [4:0]i=23;

always@(posedge clk) begin s1=A[31];s2=B[31]; e1=A[30:23];e2=B[30:23]; m1=A[22:0];m2=B[22:0];

if(e1==e2) begin e$_final = e1; if(s1 == s2)beginm_final = 1'b1, m1 + 1'b1, m2; s_final = s1; out[30 : 23] = (m_final[24] == 1'b1)?e_final+1 : e_final; out[22 : 0] = (m_final[24] == 1'b1)?m_final[23 : 1] : m_final[22 : 0]; endelsebeginm_final = (m1 > m2)?1'b1, m1 - 1'b1, m2 : 1'b1, m2 - 1'b1, m1; s_final = (m1 > m2)?s1 : s2; count = 0; index = 23; for(i = 23; i > 0; i = i - 1)beginif(m_final[i] == 1count == 0)beginindex = i; count = count + 1; endendm_final[23 : 0] = m_final[23 : 0] << (23 - index); out[22 : 0] = m_final[22 : 0]; out[30 : 23] = e_final - (23 - index); endendelsebeginexp_diff = (e1 > e2)?(e1 - e2) : (e2 - e1); e_final = (e1 > e2)?e1 : e2; if(s1 == s2)beginif(A[30 : 0] > B[30 : 0])beginm2 = 1'b1, m2[22 : 0] >> exp_diff; m_final = 1'b1, m1 + 1'b0, m2; endelsebeginm1 = 1'b1, m1[22 : 0] >> exp_diff; m_final = 1'b0, m1 + 1'b1, m2; ends_final = s1; out[30 : 23] = (m_final[24] == 1'b1)?e_final + 1 : e_final; out[22 : 0] = (m_final[24] == 1'b1)?m_final[23 : 1] : m_final[22 : 0]; endelsebeginif(A[30 : 0] > B[30 :$

$0])beginm2 = 1'b1, m2[22:0] >> exp_diff; m_final = 1'b1, m1 - 1'b0, m2; s_final = s1; endelsebeginm1 = 1'b1, m1[22:0] >> exp_diff; m_final = 1'b1, m2 - 1'b0, m1; s_final = s2; endcount = 0; index = 23; for(i = 23; i > 0; i = i - 1)beginif(m_final[i] == 1count == 0)beginindex = i; count = count + 1; endendm_final[23:0] = m_final[23:0] << (23 - index); out[22:0] = m_final[22:0]; out[30:23] = e_final - (23 - index); endend$

out[31]=s$_final; endendmodule$

module accum$_reg(A, out); input[31:0]A; outputreg[31:0]out = 0; always@(A)beginout = A; endendmodule$

5.1 Testbench:

module MAC$_final_tb(); regclk; reg[31:0]A, B; wire[31:0]out;$

MAC$_finalmac1(clk, A, B, out);$

initial clk=1; always 5 clk= clk;

initial begin A=32'b0$_1$0000000$_0$10000000000000000000000$; B = 32'b0_0 1111111_1 00000000000000000000000; //10A = 32'b0_0 1111111_1 00000000000000000000000; B = 32'b1_0 1111111_1 00000000000000000000000; 10A = 32'b1_1 0000001_0 11000000000000000000000; B = 32'b0_1 0000001_0 01000000000000000000000; 10A = 0; B = 0;$ 20finish; end

endmodule

6 RTL Schematic:

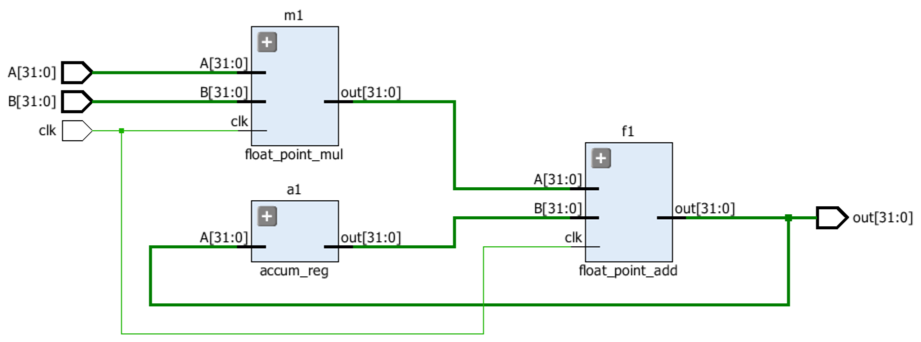


Figure 4: Schematic

7 Simulation:

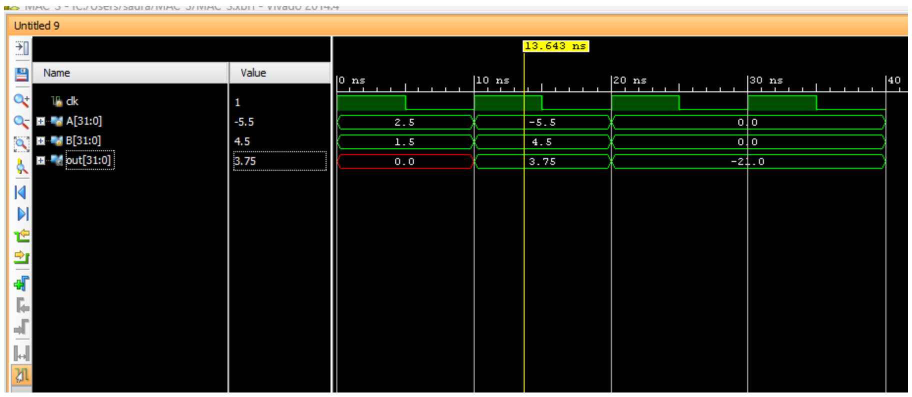


Figure 5: behavioral simulation

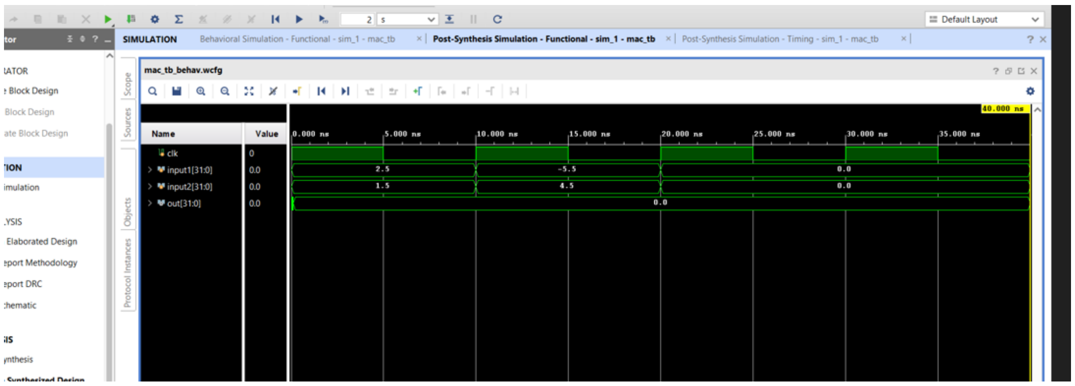


Figure 6: functional simulation

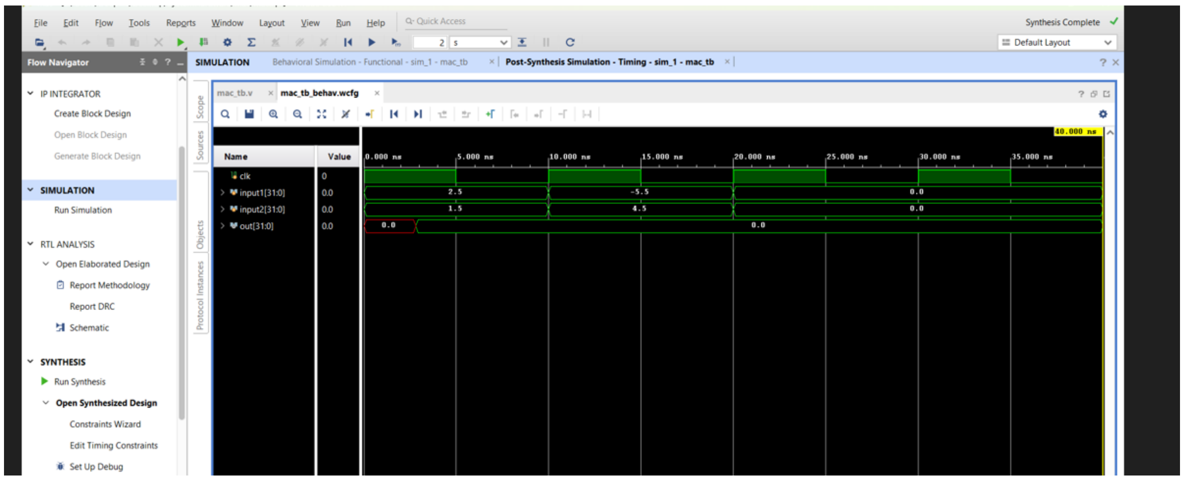
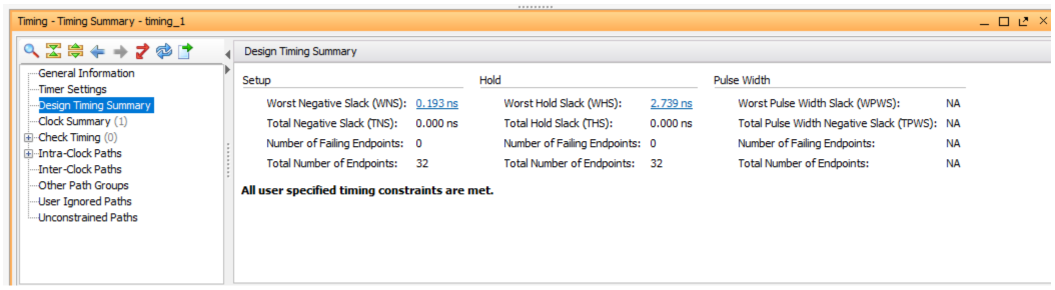


Figure 7: post synthesis timing simulation

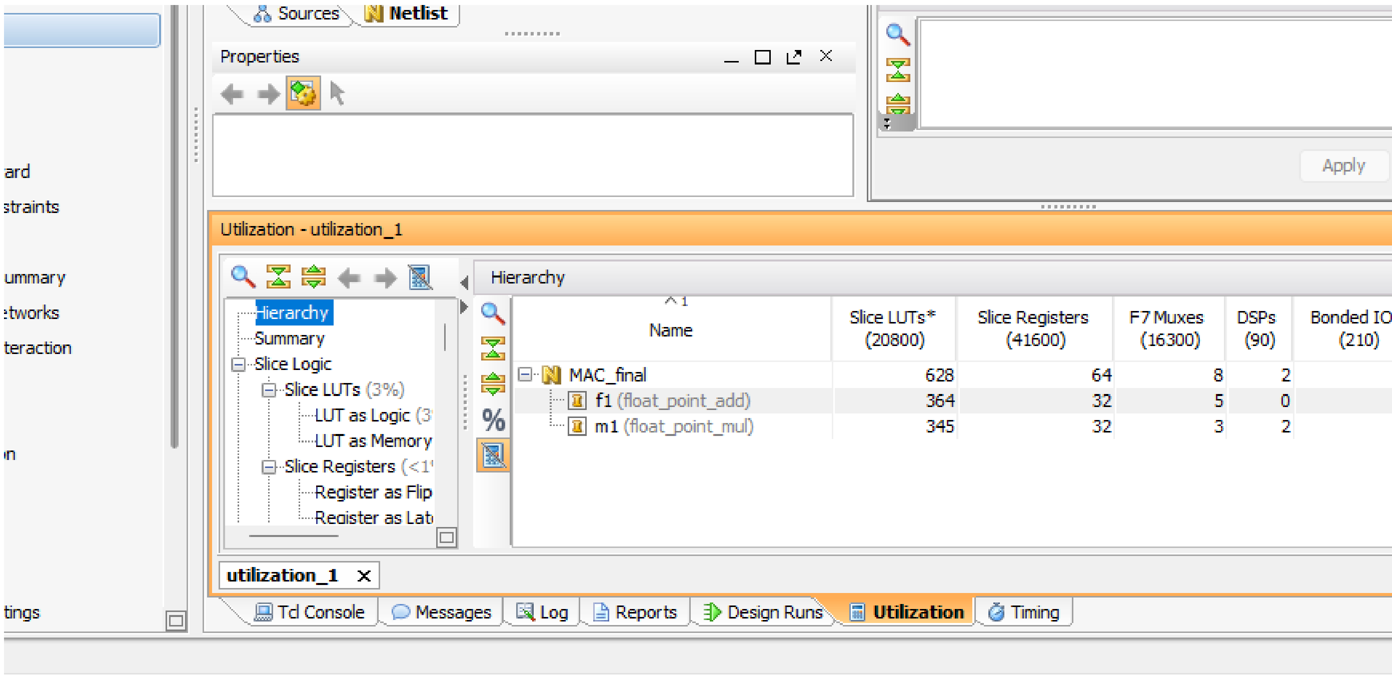
8 Timing summary and Utilization:



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.193 ns	Worst Hold Slack (WHS):	2.739 ns	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	32	Total Number of Endpoints:	32	Total Number of Endpoints:	NA

All user specified timing constraints are met.

Figure 8: timing summary



Name	Slice LUTs* (20800)	Slice Registers (41600)	F7 Muxes (16300)	DSPs (90)	Bonded IO (210)
MAC_final	628	64	8	2	
f1 (float_point_add)	364	32	5	0	
m1 (float_point_mul)	345	32	3	2	

Figure 9: Utilization

9 Observation:

After synthesis of the corresponding designs, The LUTs and Flops have been found from the utilization report. The delay has been found from the timing report and the power has been found from the power report. The proper constraints have been added and the synthesis results are shown below.

	LUTs	Flipflops	Power (W)	WNS (ns)
IEEE 754 32bit MAC	628	64	0.089	0.193

10 Results

1. MAC unit is designed as per mentioned and design is verified and the design is showing functionality as expected.
2. Different design parameters such as LUTs, Power and WNS is been calculated from synthesis.