



INDRAPRASTHA INSTITUTE *of*  
INFORMATION TECHNOLOGY  
DELHI

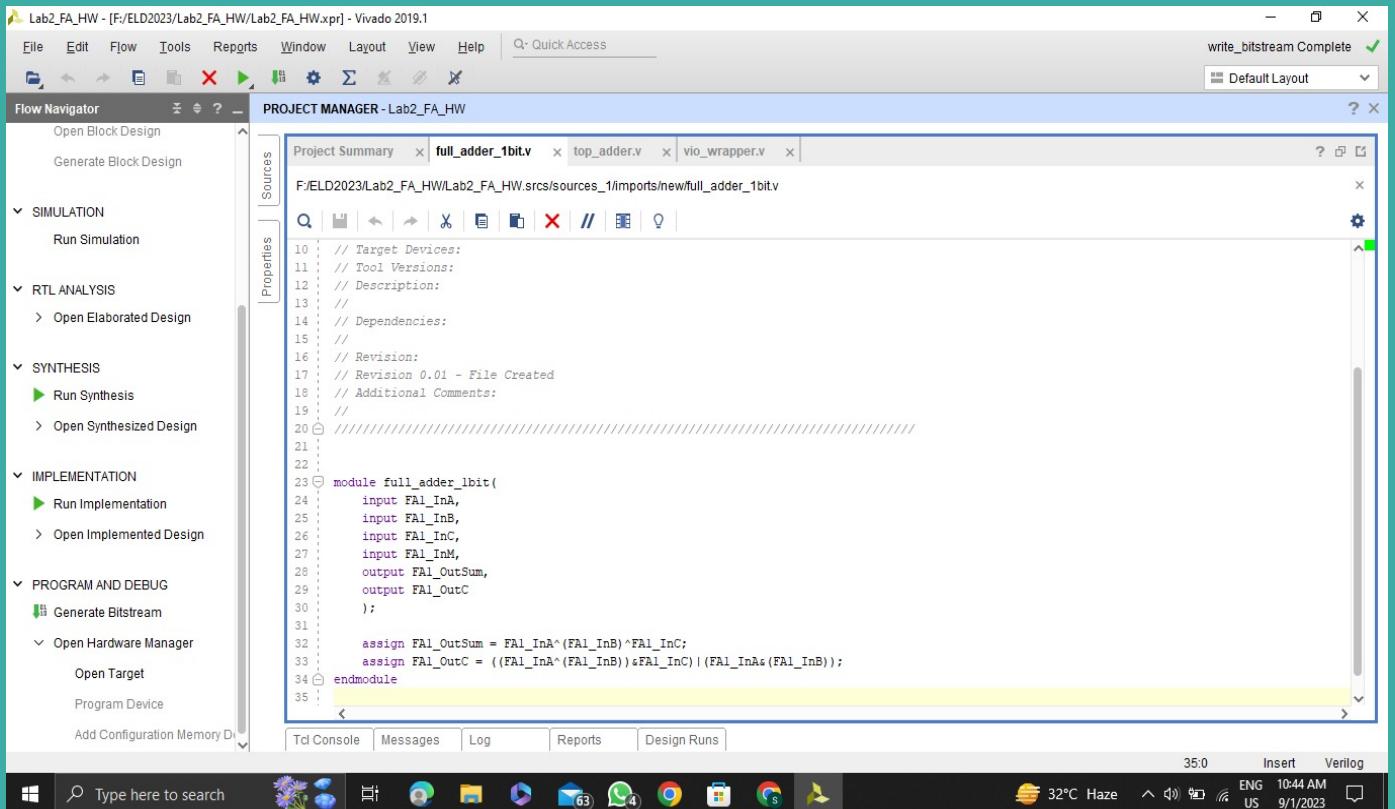
Department  
of  
Electronics & Communication Engineering

Embedded Logic Design

Lab 2 Submission

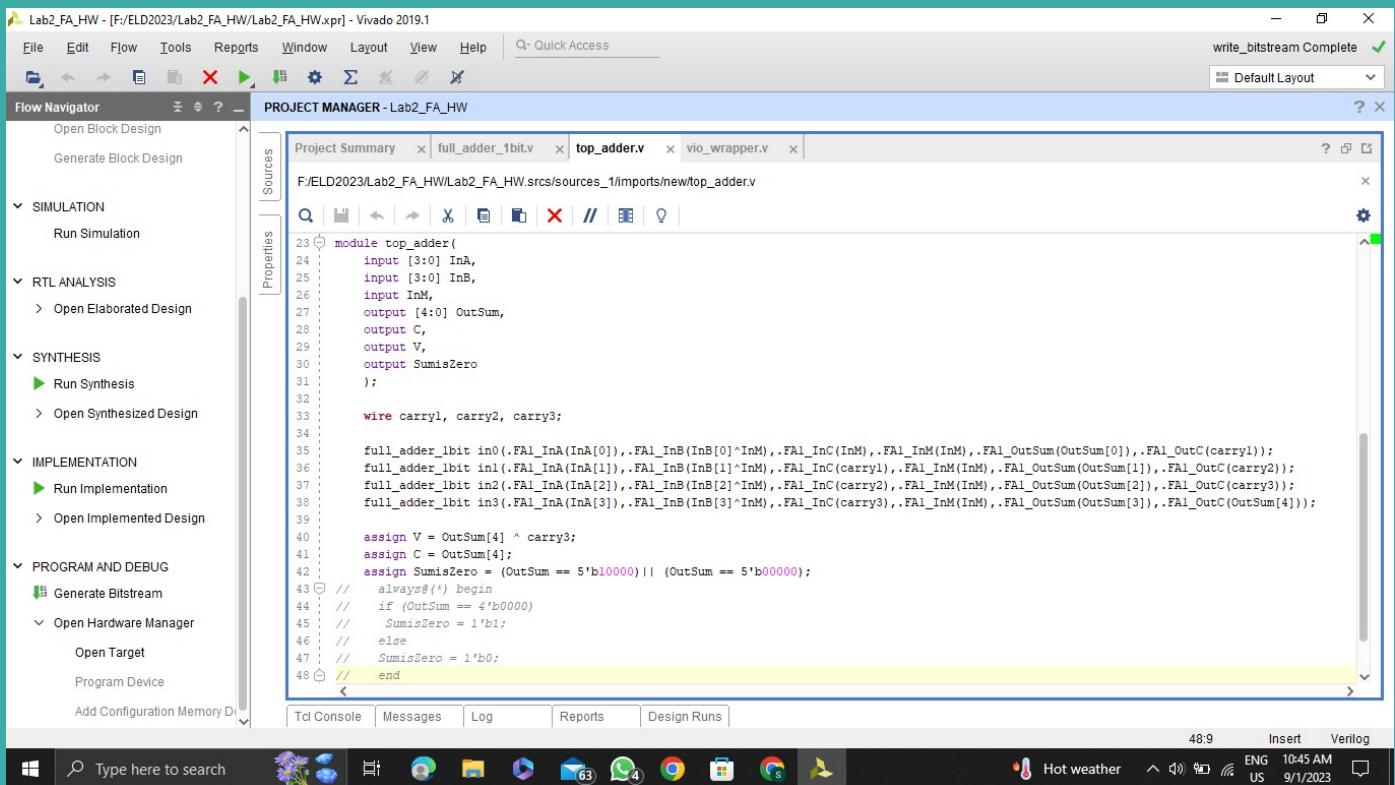
SHIVAM SHUKLA  
2022478

# Source Code



The screenshot shows the Vivado 2019.1 Project Manager for the Lab2\_FA\_HW project. The left sidebar contains a tree view of design tasks: Flow Navigator, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG. The main area displays the source code for the `full_adder_1bit.v` module. The code defines a module with four inputs (`FA1_InA`, `FA1_InB`, `FA1_InC`, `FA1_InM`) and two outputs (`FA1_OutSum`, `FA1_OutC`). It uses a conditional assignment to calculate the sum based on the inputs.

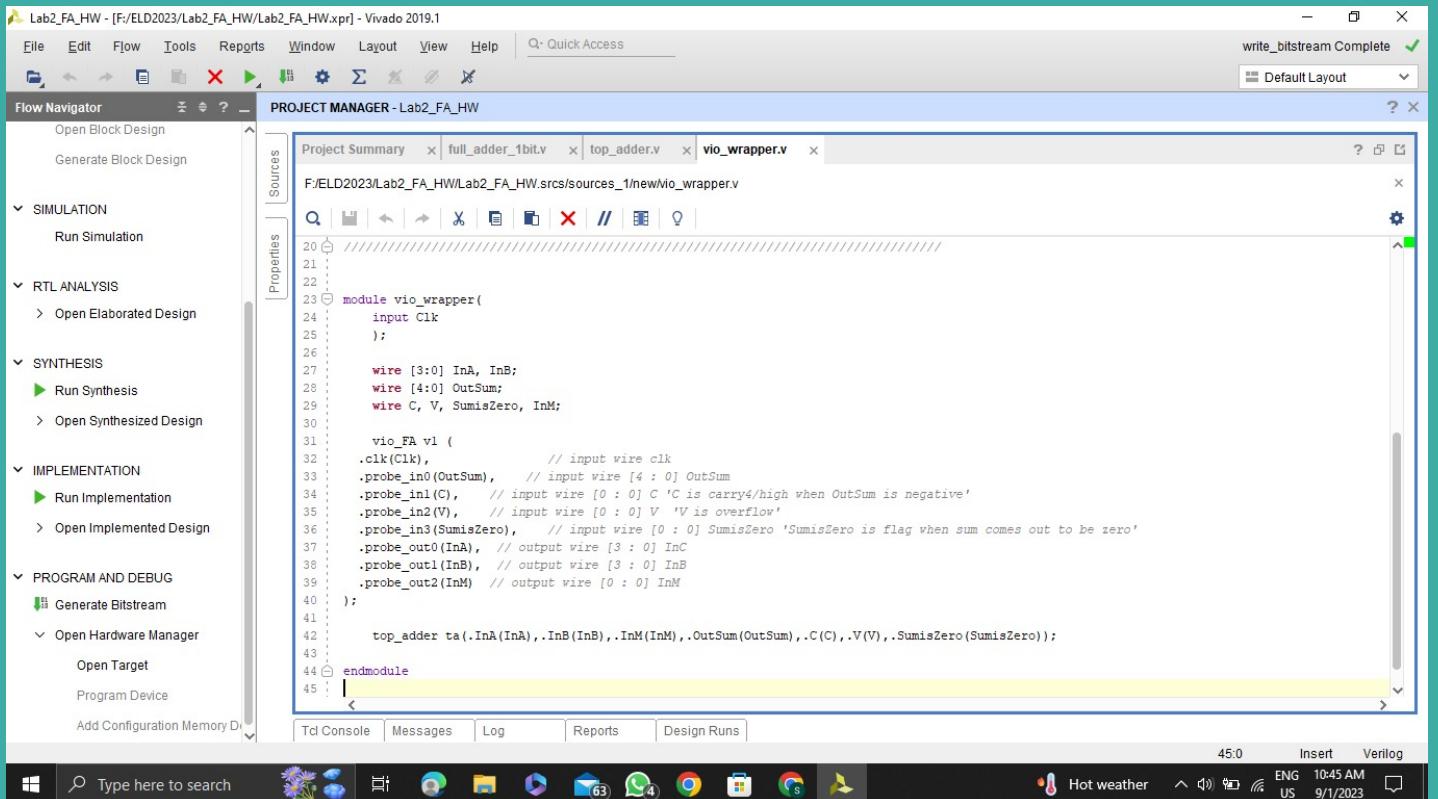
```
// Target Devices:  
// Tool Versions:  
// Description:  
// Dependencies:  
// Revision:  
// Revision 0.01 - File Created  
// Additional Comments:  
  
module full_adder_1bit(  
    input FA1_InA,  
    input FA1_InB,  
    input FA1_InC,  
    input FA1_InM,  
    output FA1_OutSum,  
    output FA1_OutC  
);  
  
    assign FA1_OutSum = FA1_InA^(FA1_InB)^FA1_InC;  
    assign FA1_OutC = ((FA1_InA^(FA1_InB))&FA1_InC) | (FA1_InA&(FA1_InB));  
endmodule
```



The screenshot shows the Vivado 2019.1 Project Manager for the Lab2\_FA\_HW project. The left sidebar contains a tree view of design tasks: Flow Navigator, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG. The main area displays the source code for the `top_adder.v` module. The code defines a module with four inputs (`InA`, `InB`, `InM`, `OutSum`) and four outputs (`C`, `V`, `SumisZero`, `carry1`, `carry2`, `carry3`). It includes a wire declaration for `carry1`, `carry2`, and `carry3`. The body of the module contains multiple calls to the `full_adder_1bit` module with different input combinations. It also includes logic to calculate the sum and carry values, and a conditional assignment for the `SumisZero` output.

```
module top_adder(  
    input [3:0] InA,  
    input [3:0] InB,  
    input InM,  
    output [4:0] OutSum,  
    output C,  
    output V,  
    output SumisZero  
);  
  
    wire carry1, carry2, carry3;  
  
    full_adder_1bit in0(.FA1_InA(InA[0]),.FA1_InB((InB[0]^InM)),.FA1_InC(InM),.FA1_InM(InM),.FA1_OutSum(OutSum[0]),.FA1_OutC(carry1));  
    full_adder_1bit in1(.FA1_InA((InA[1])),.FA1_InB((InB[1]^InM)),.FA1_InC(carry1),.FA1_InM(InM),.FA1_OutSum(OutSum[1]),.FA1_OutC(carry2));  
    full_adder_1bit in2(.FA1_InA((InA[2])),.FA1_InB((InB[2]^InM)),.FA1_InC(carry2),.FA1_InM(InM),.FA1_OutSum(OutSum[2]),.FA1_OutC(carry3));  
    full_adder_1bit in3(.FA1_InA((InA[3])),.FA1_InB((InB[3]^InM)),.FA1_InC(carry3),.FA1_InM(InM),.FA1_OutSum(OutSum[3]),.FA1_OutC(OutSum[4]));  
  
    assign V = OutSum[4] ^ carry3;  
    assign C = OutSum[4];  
    assign SumisZero = (OutSum == 5'b10000) || (OutSum == 5'b00000);  
    always@(*) begin  
        if (OutSum == 4'b0000)  
            SumisZero = 1'b1;  
        else  
            SumisZero = 1'b0;  
    end
```

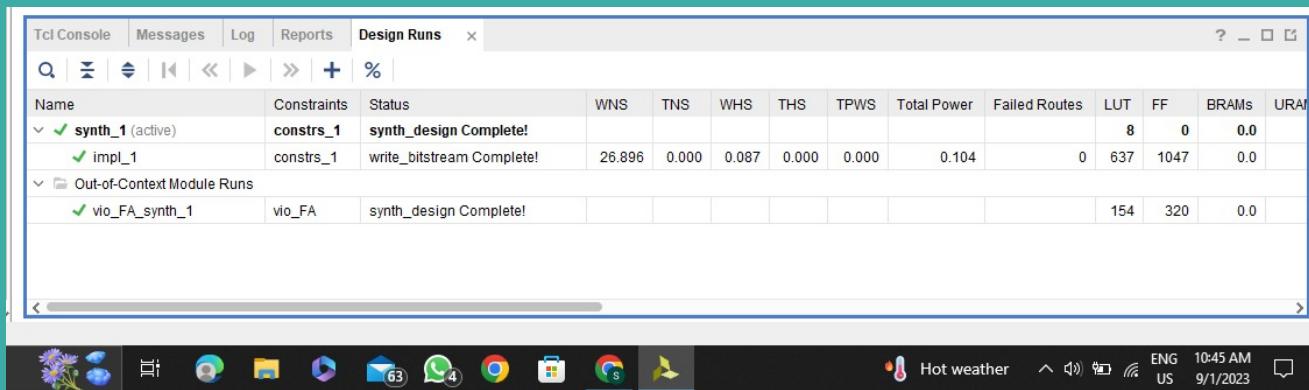
## VIO WRAPPER FILE:



The screenshot shows the Vivado 2019.1 interface with the project "Lab2\_FA\_HW" open. The "PROJECT MANAGER - Lab2\_FA\_HW" tab is selected. In the center pane, the "vio\_wrapper.v" file is displayed in a code editor. The code defines a module vio\_wrapper with various input and output wires and probe statements for simulation.

```
20 //////////////////////////////////////////////////////////////////
21 //
22 module vio_wrapper(
23     input Clk
24 );
25 //
26 wire [3:0] InA, InB;
27 wire [4:0] OutSum;
28 wire C, V, SumisZero, InM;
29 //
30 vio_FA v1 (
31     .clk(Clk),           // input wire clk
32     .probe_in0(OutSum),  // input wire [4 : 0] OutSum
33     .probe_in1(C),       // input wire [0 : 0] C 'C is carry4/high when OutSum is negative'
34     .probe_in2(V),       // input wire [0 : 0] V 'V is overflow'
35     .probe_in3(SumisZero), // input wire [0 : 0] SumisZero 'SumisZero is flag when sum comes out to be zero'
36     .probe_out0(InA),   // output wire [3 : 0] InC
37     .probe_out1(InB),   // output wire [3 : 0] InB
38     .probe_out2(InM)    // output wire [0 : 0] InM
39 );
40 //
41 top_adder ta(.InA(InA),.InB(InB),.InM(InM),.OutSum(OutSum),.C(C),.V(V),.SumisZero(SumisZero));
42 //
43 endmodule
44 //
```

## BITSTREAM GENERATION :

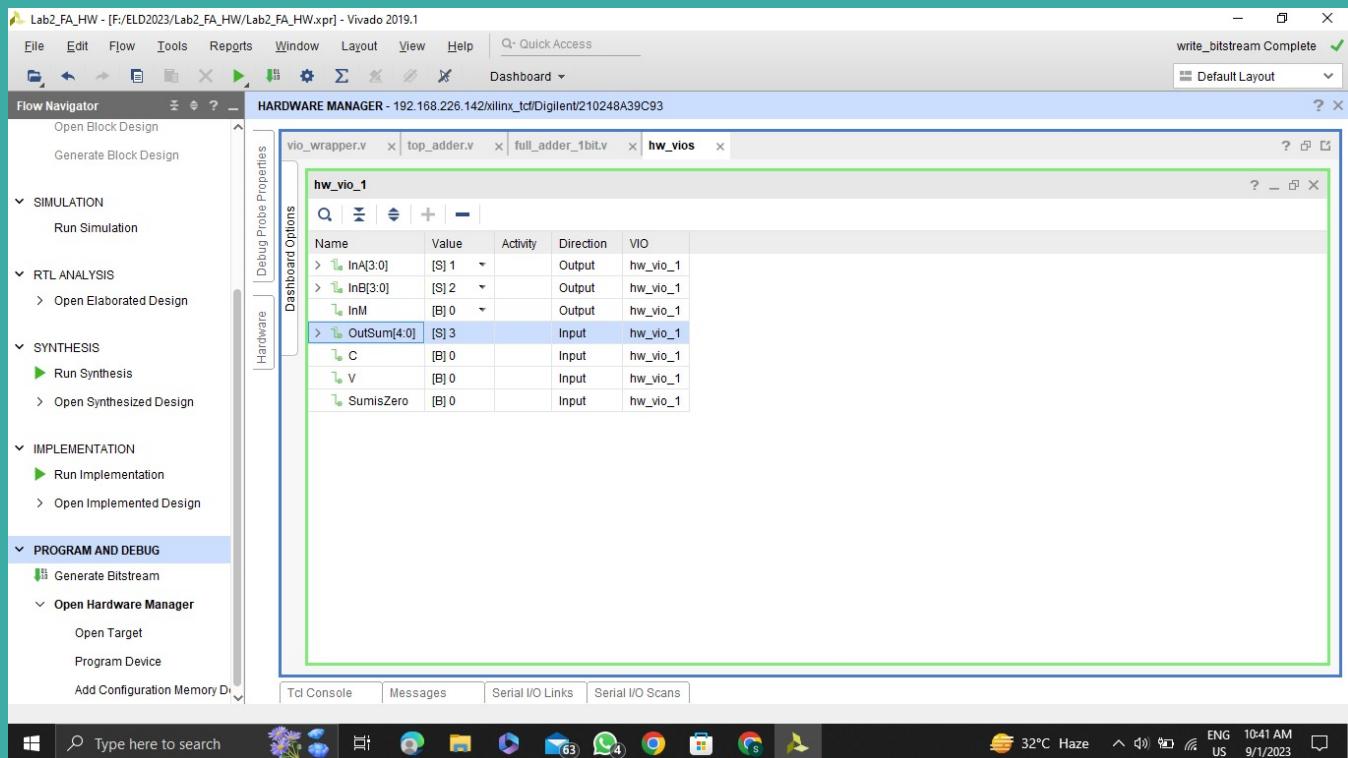


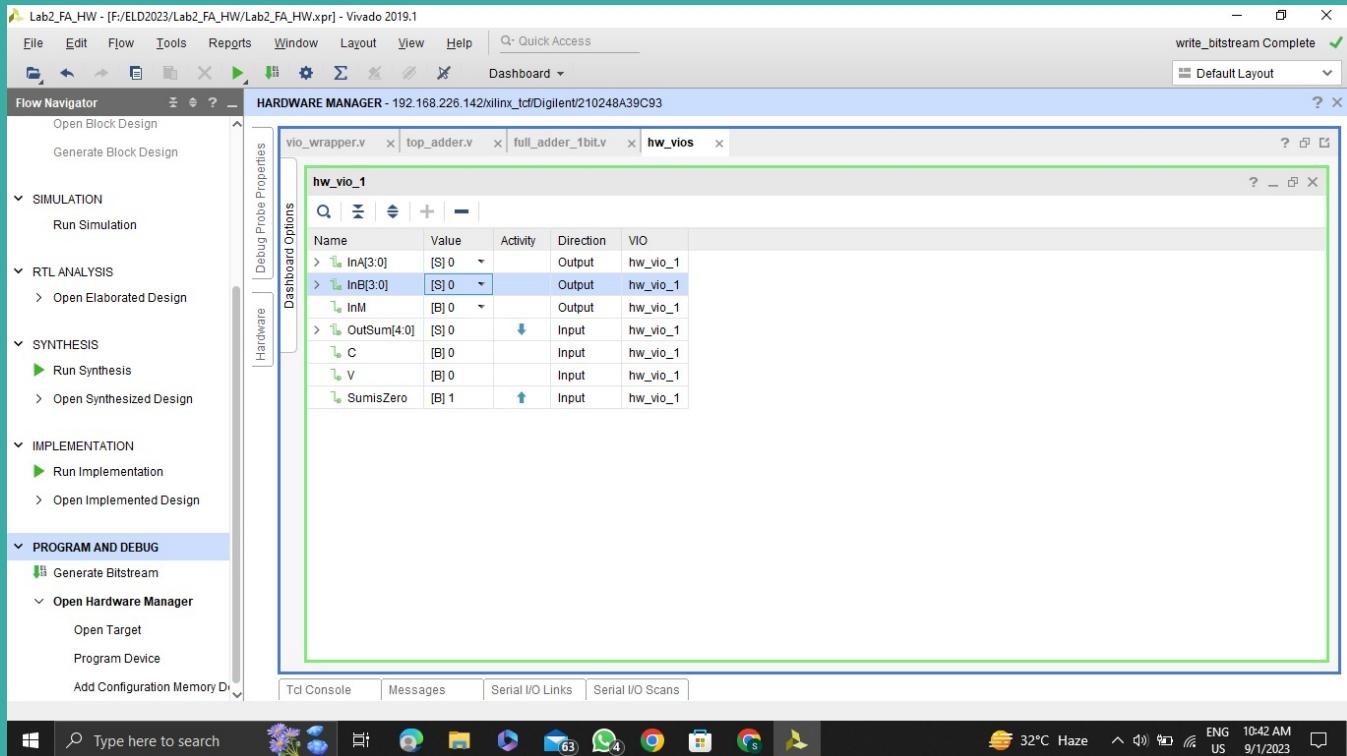
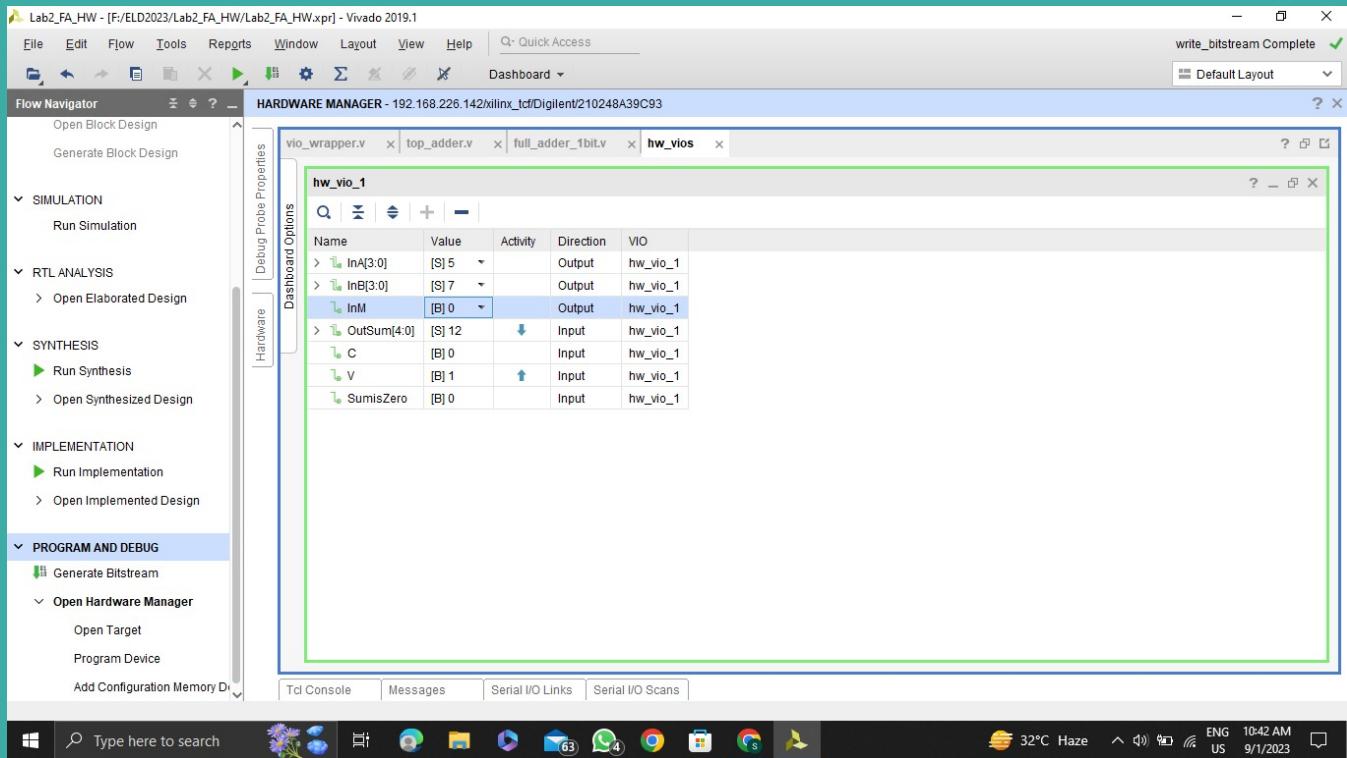
The screenshot shows the Vivado 2019.1 "Design Runs" window. It displays a table of runs, with the first run "synth\_1" being active and marked as "synth\_design Complete!". The second run "impl\_1" is also marked as "write\_bitstream Complete!". The table includes columns for Name, Constraints, Status, WNS, TNS, WHS, THS, TPWS, Total Power, Failed Routes, LUT, FF, BRAMs, and URAM.

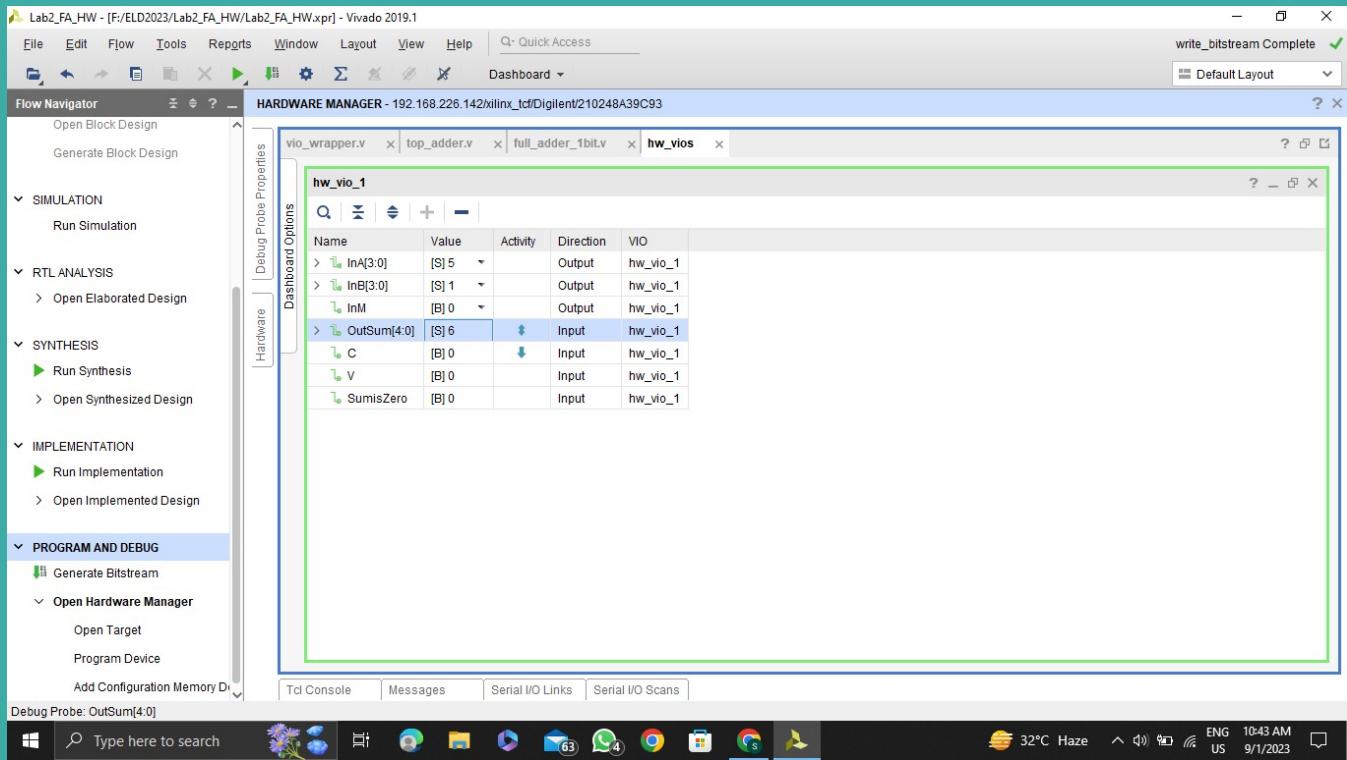
Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM
✓ synth_1 (active)	constrs_1	synth_design Complete!								8	0	0.0	
✓ impl_1	constrs_1	write_bitstream Complete!	26.896	0.000	0.087	0.000	0.000	0.104	0	637	1047	0.0	
✓ vio_FA_synth_1	vio_FA	synth_design Complete!								154	320	0.0	

# VIO OUTPUT

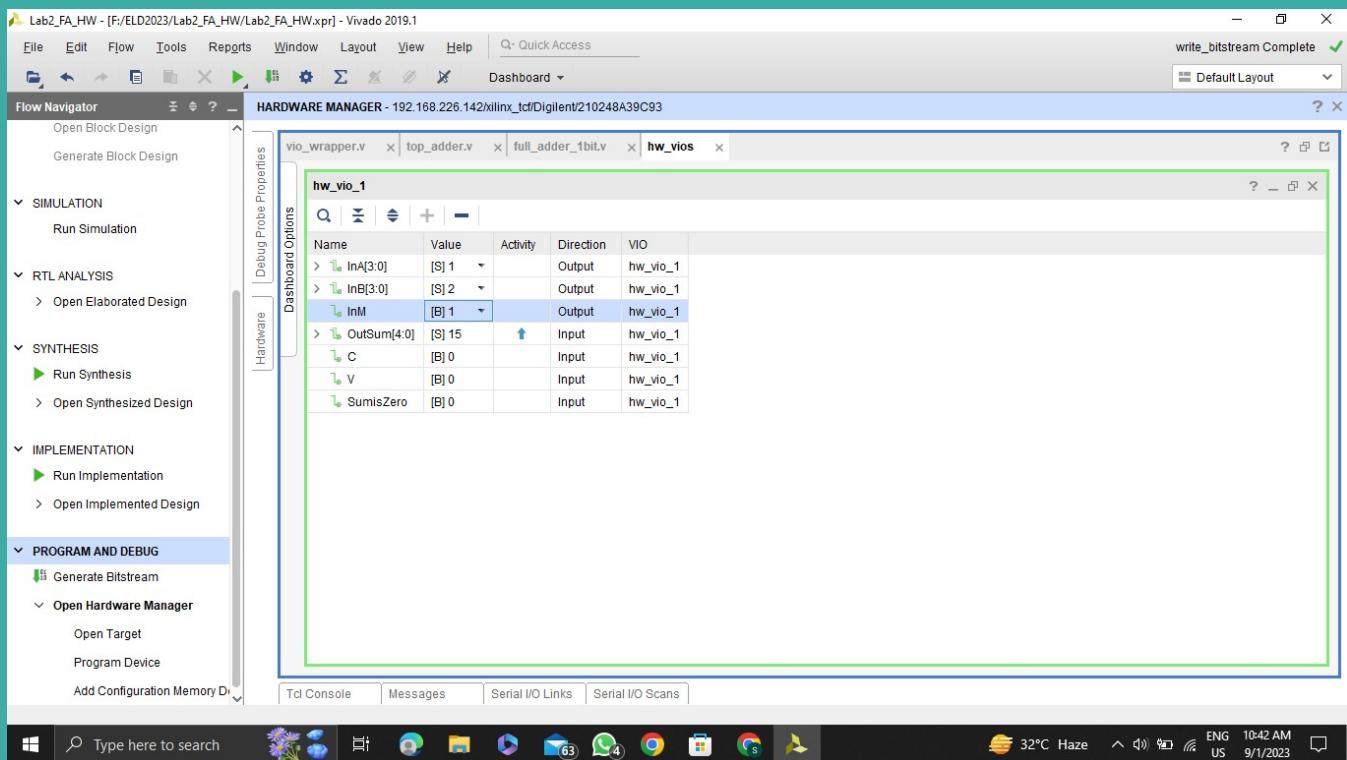
# M = 0 , Adder :

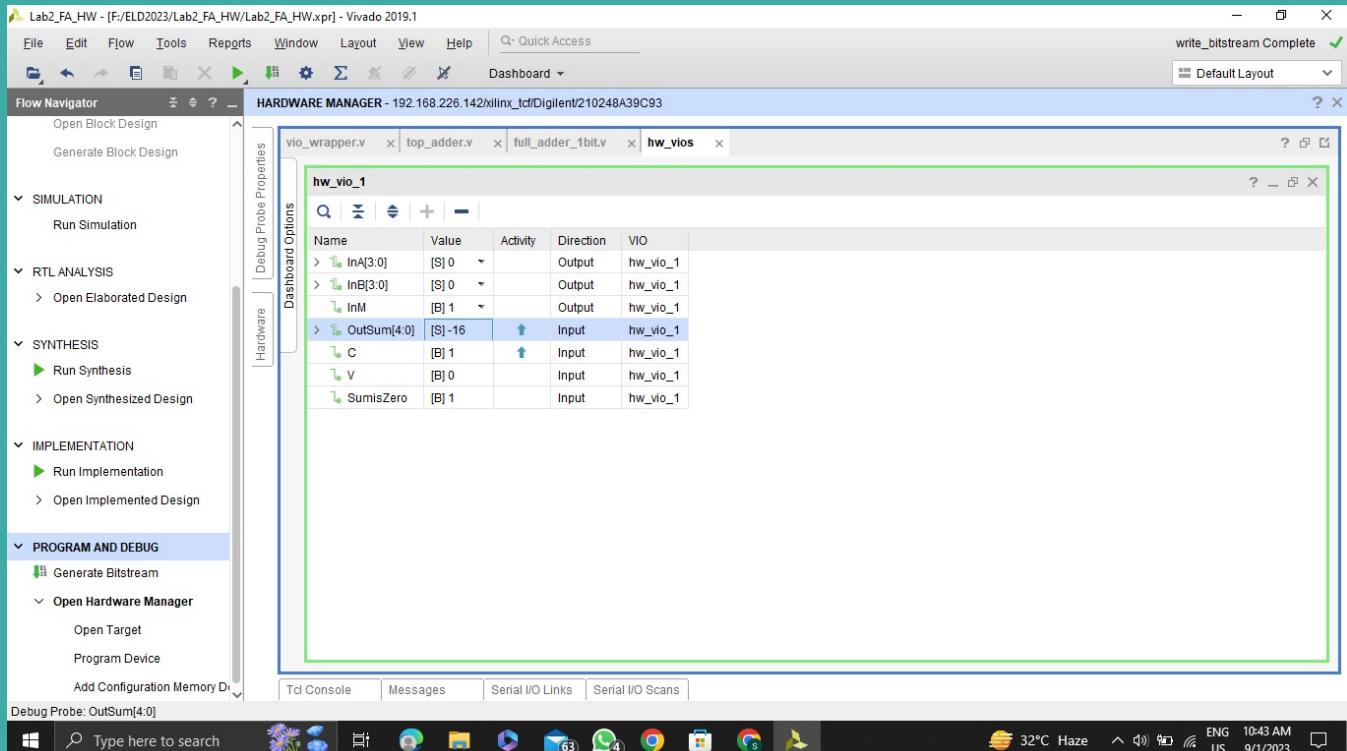
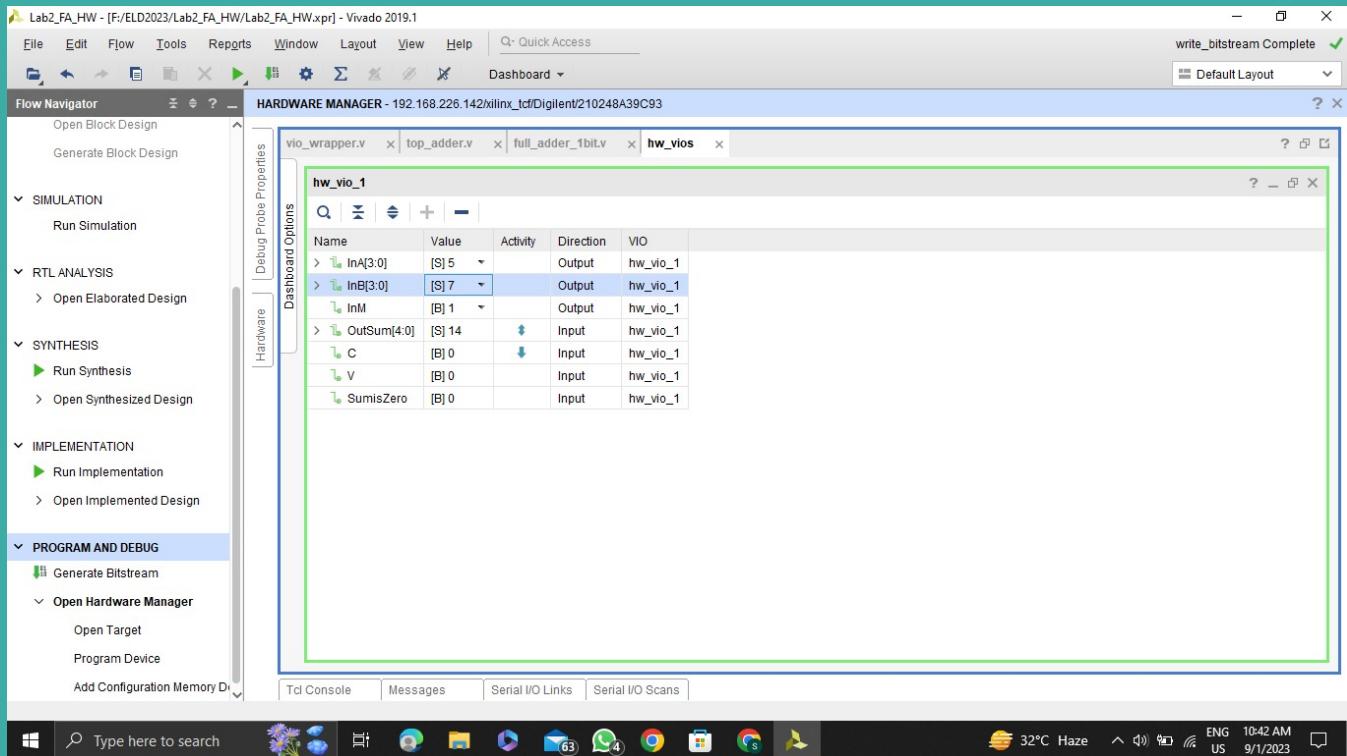






## # M = 1 , Subtractor :





Thank You