INDRAPRASTHA INSTITUTE of INFORMATION TECHNOLOGY DELHI

Department
of
Electronics & Communication Engineering

Embedded Logic Design(ECE270)

Dr. Sumit J Darak

Lab_2: Design and Implement 3-bit Full Adder and 3x3 Binary Multiplier and test both on Basys3 Board using VIO IP

Mohammad Shariq
2020220

04-10-2021

OBJECTIVE:

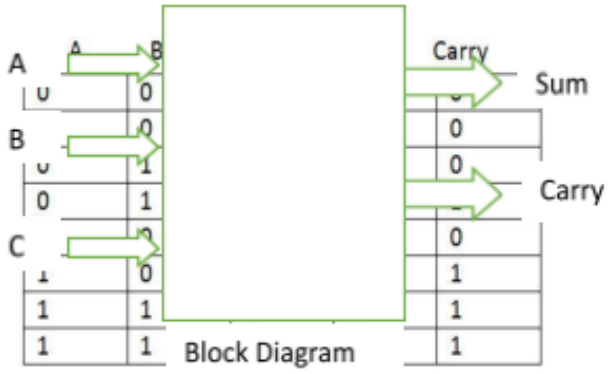
- **Design and implement a 3-bit full adder and verify its functionality.**
- **Implement a 3x3 binary multiplier using the full adder and verify its functionality.**
- **Test the functionality of both on the Basys3 Board using VIO IP (remote access).**

Theory:

- ***Full Adder***

1. 3-bit Full Adder Implementation:

It is a combinational circuit used for three bits addition. The block diagram and truth table are shown below:



Truth Table

The expressions for Sum and Carry Output are as follows:
Sum= A^B^C
Carry=(A^B).C + AB

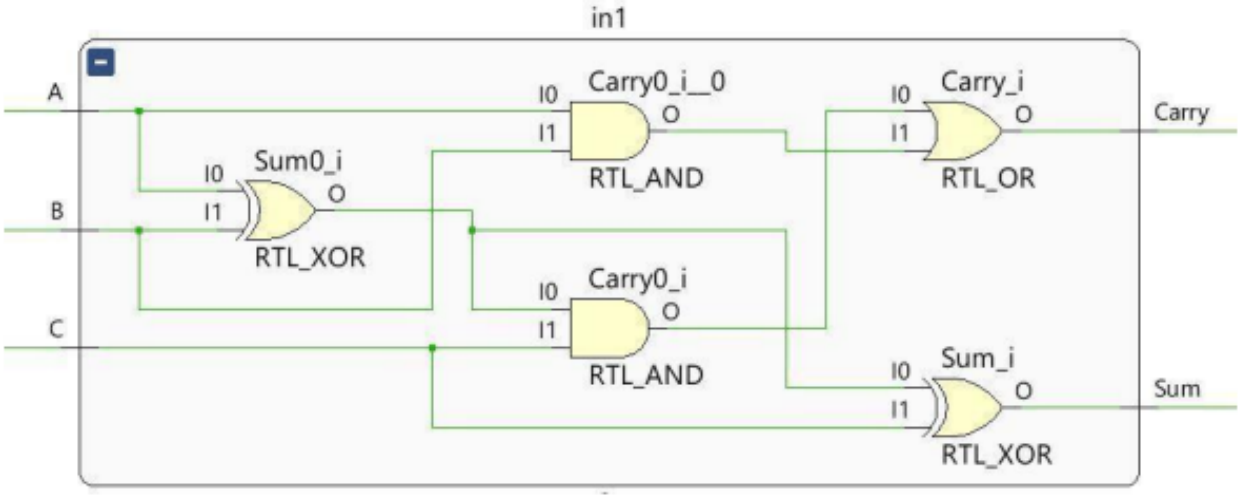


Fig: 1 Gate Level Circuit Diagram of Full Adder

To implement the 3-bit Full Adder, three full adders are connected in cascade as shown in the circuit diagram in Fig:2

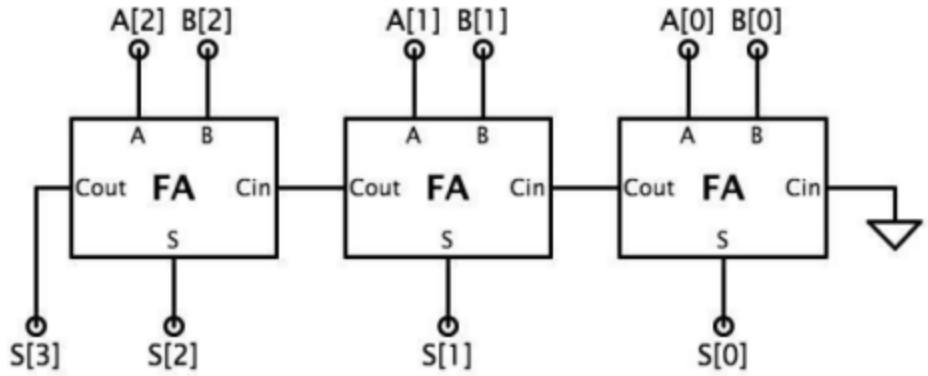


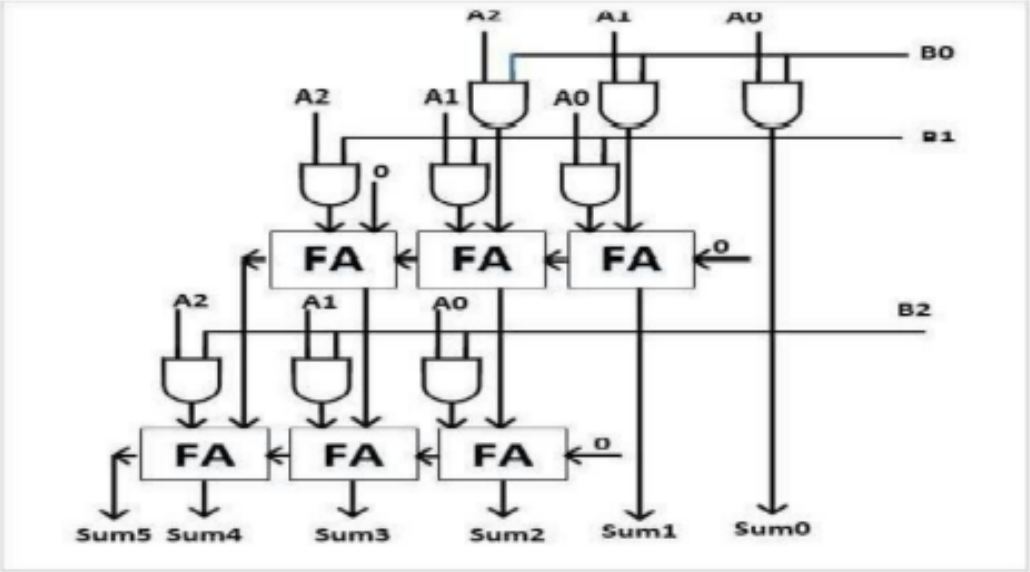
Fig 2: Circuit diagram for 3-bit adder using Full Adders

- ***3x3 Binary Multiplier :***

Multiplication is a very common operation in digital systems. For that reason, custom-designed multipliers are often used. The goal of this lab is to make you familiar with the design and implementation of such a multiplier. Multiplication of a 3-bit number B(2:0) by a 3-bit number A(2:0) can be illustrated as follows.

			A2	A1	A0	
		X	B2	B1	B0	
			A2B0	A1B0	A0B0	
		A2B1	A1B1	A0B1		
+	A2B2	A1B2	A0B2			
Sum5	Sum4	Sum3	Sum2	Sum1	Sum0	

Implementation using Full Adder: The circuit diagram for the implementation of 3X3 Multiplier is given below:



Observations:

- ***For 3-bit Full Adder***

programme for functionality of full adder

```
21
22
23  module full_adder(
24      input A,
25      input B,
26      input C,
27      output Sum,
28      output Carry
29      );
30      assign Sum=A^B^C;
31      assign Carry=((A^B)&C)|(A&B);
32  endmodule
33
```

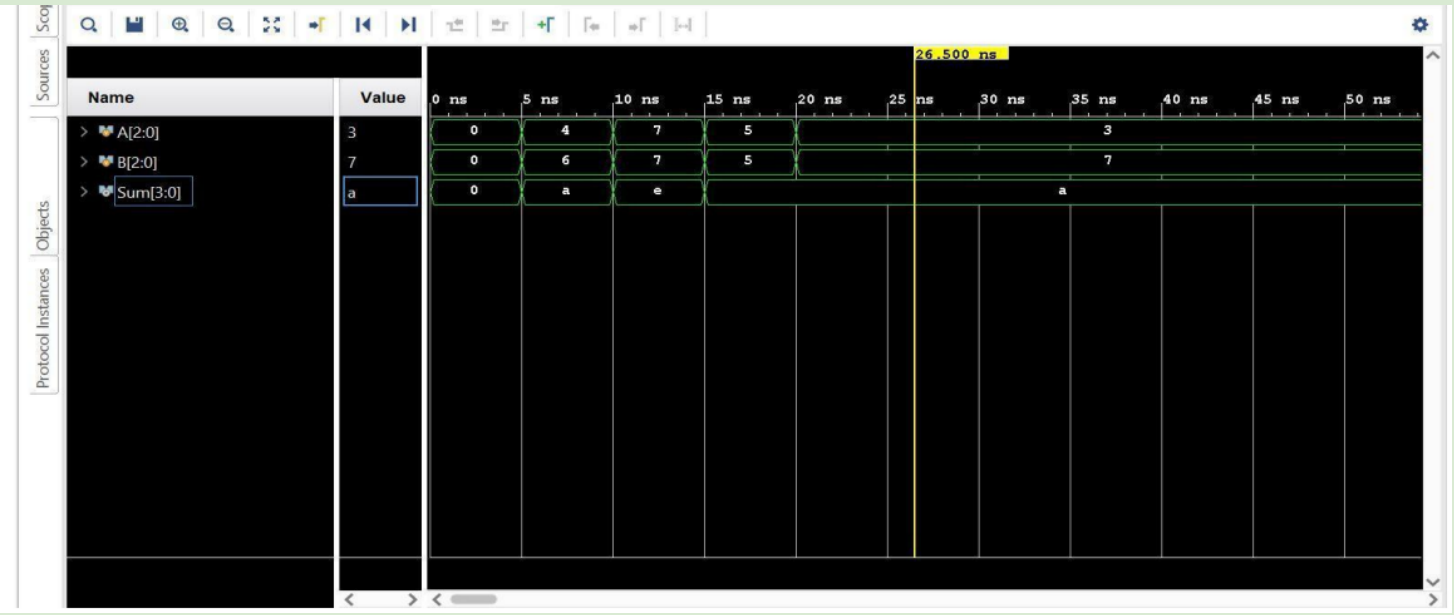
programme for functionality of 3-bit full adder

```
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module top_adder(
24      input [2:0] A,
25      input [2:0] B,
26      output [3:0]Sum
27      );
28      wire c1,c2;
29      full_adder in1(.A(A[0]),.B(B[0]),.C(1'b0),.Sum(Sum[0]),.Carry(c1));
30      full_adder in2(.A(A[1]),.B(B[1]),.C(c1),.Sum(Sum[1]),.Carry(c2));
31      full_adder in3(.A(A[2]),.B(B[2]),.C(c2),.Sum(Sum[2]),.Carry(Sum[3]));
32  endmodule
33
```

Testbench for 3-bit full adder

```
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module tb_fa(
24
25      );
26      reg [2:0] A,B;
27      wire [3:0] Sum;
28
29      top_adder tb1(.A(A),.B(B),.Sum(Sum));
30
31      initial
32      begin
33          A=3'b000;
34          B=3'b000;
35      end
36
37      initial
38      begin
39          #5 A=3'b100;B=3'b110;
40          #5 A=3'b111;B=3'b111;
41          #5 A=3'b101;B=3'b101;
42          #5 A=3'b011;B=3'b111;
43      end
44  endmodule
45
```

Results for testbench



Constraints for the implementation and VIO module programme

Project Summary | vio_wrapper.v | full_adder.v | top_adder.v | const.xdc

D:/IIITD/ELD_LABs/Lab_2_Multiplier/FullAdder_TopAdder_Multiplier/FullAdder_TopAdder_Multiplier.srcs/constrs_1/new/const.xdc

```
### This file is a general .xdc for the Basys3 rev B board
### To use it in a project:
### - uncomment the lines corresponding to used pins
### - rename the used ports (in each line, after get_ports) according to the top level signal names

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
    set_property IOSTANDARD LVCMOS33 [get_ports clk]
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
#set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
#   set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
#set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
#   set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
##set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
#   #set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
##set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
```

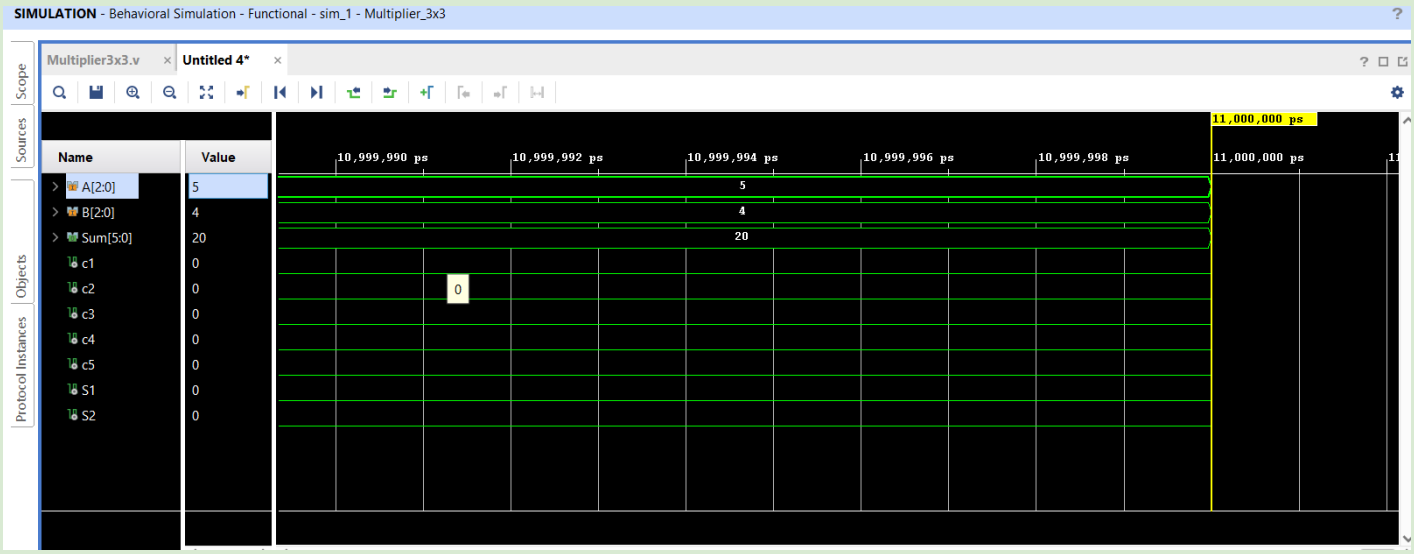
```
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module vio_wrapper(
    input clk
    );
    wire [2:0] A,B;
    wire [3:0] Sum;
        vio_0 vin1 (
    .clk(clk),
    .probe_in0(Sum),
    .probe_out0(A),
    .probe_out1(B)
    );

 top_adder in4(.A(A),.B(B),.Sum(Sum));
endmodule

```

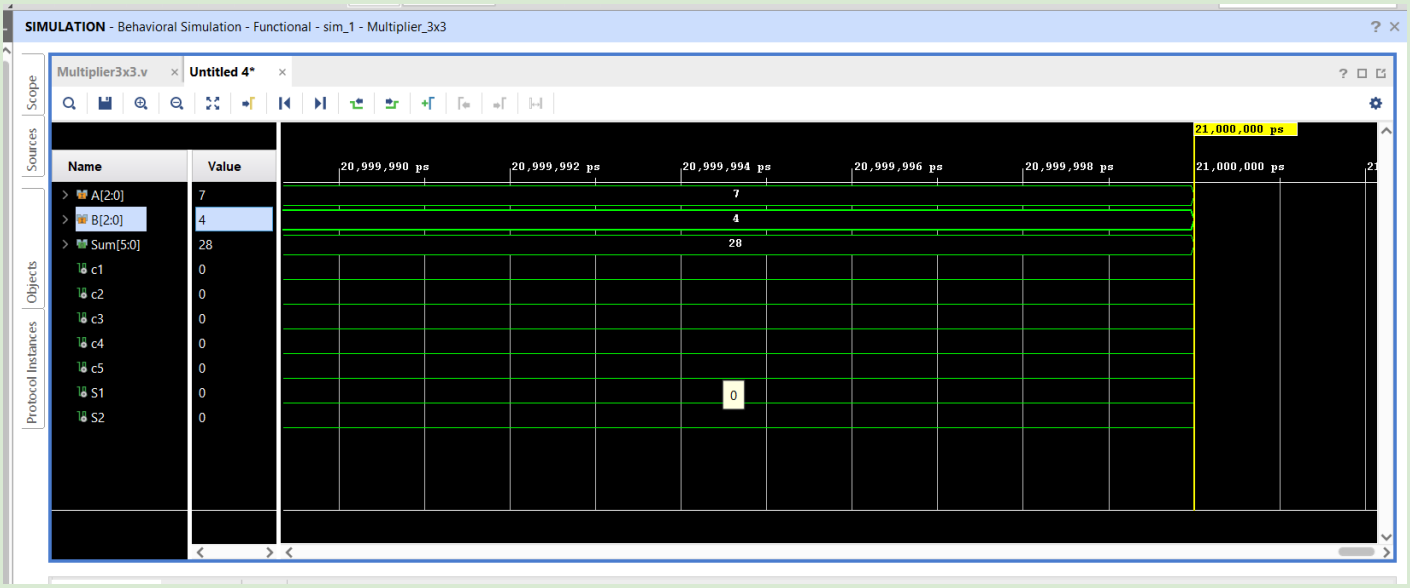
The Screenshots for how to generate BitStream and connect to the Basys3 board using VIO IP for 3 bit full adder are already given in the LAB handout.

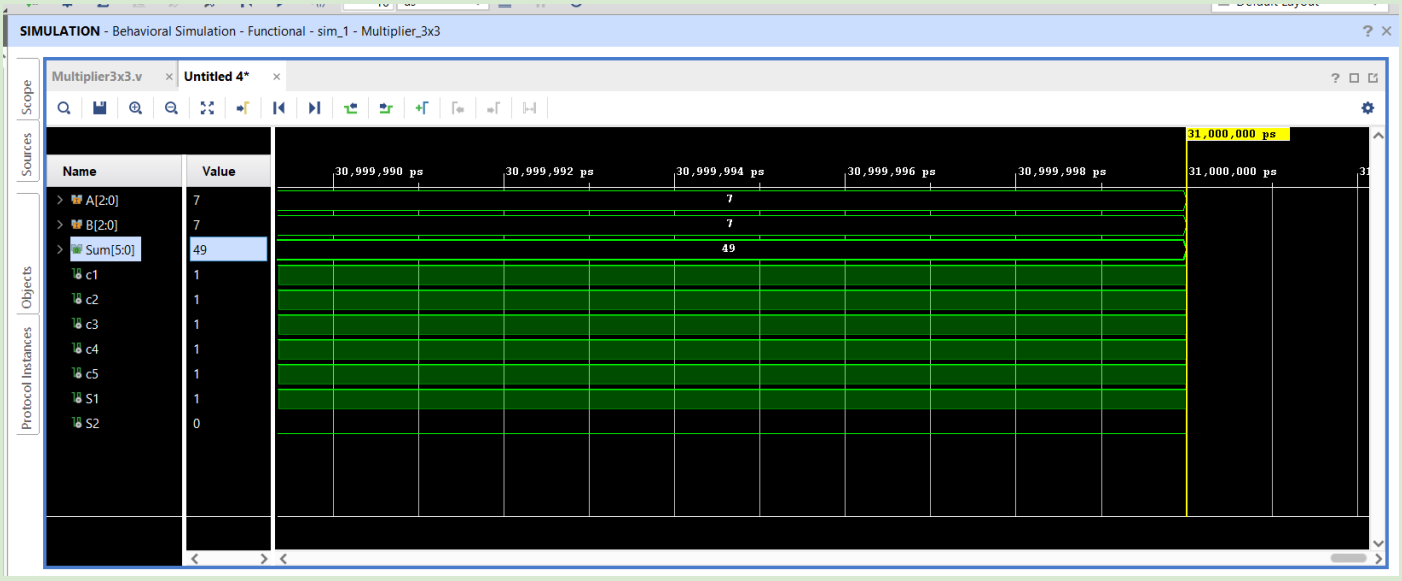
- ***For 3x3 Multiplier***
  functionality Program for 3x3 multiplier:

```
//////////////////////////////////////////////////////////////////////////////////


module Multiplier_3x3 (
    input [2:0] A,
    input [2:0] B,
    output [5:0] Sum
    );
    wire c1,c2,c3,c4,c5;   //input carry
    wire S1,S2;   // sum outputs of an adder as inputs for another adder

    assign Sum[0]= A[0] & B[0];

    full_adder in1(.A(A[0] & B[1]),.B(B[0] & A[1]),.C(1'b0),.Sum(Sum[1]),.Carry(c1));

    full_adder in2(.A(A[2] & B[0]),.B(B[1] & A[1]),.C(c1),.Sum(S1),.Carry(c2));

    full_adder in3(.A(A[2] & B[1]),.B(1'b0),.C(c2),.Sum(S2),.Carry(c3));

    full_adder in4(.A(A[0] & B[2]),.B(S1),.C(1'b0),.Sum(Sum[2]),.Carry(c4));

    full_adder in5(.A(A[1] & B[2]),.B(S2),.C(c4),.Sum(Sum[3]),.Carry(c5));

    full_adder in6(.A(A[2] & B[2]),.B(c3),.C(c5),.Sum(Sum[4]),.Carry(Sum[5]));
endmodule

```

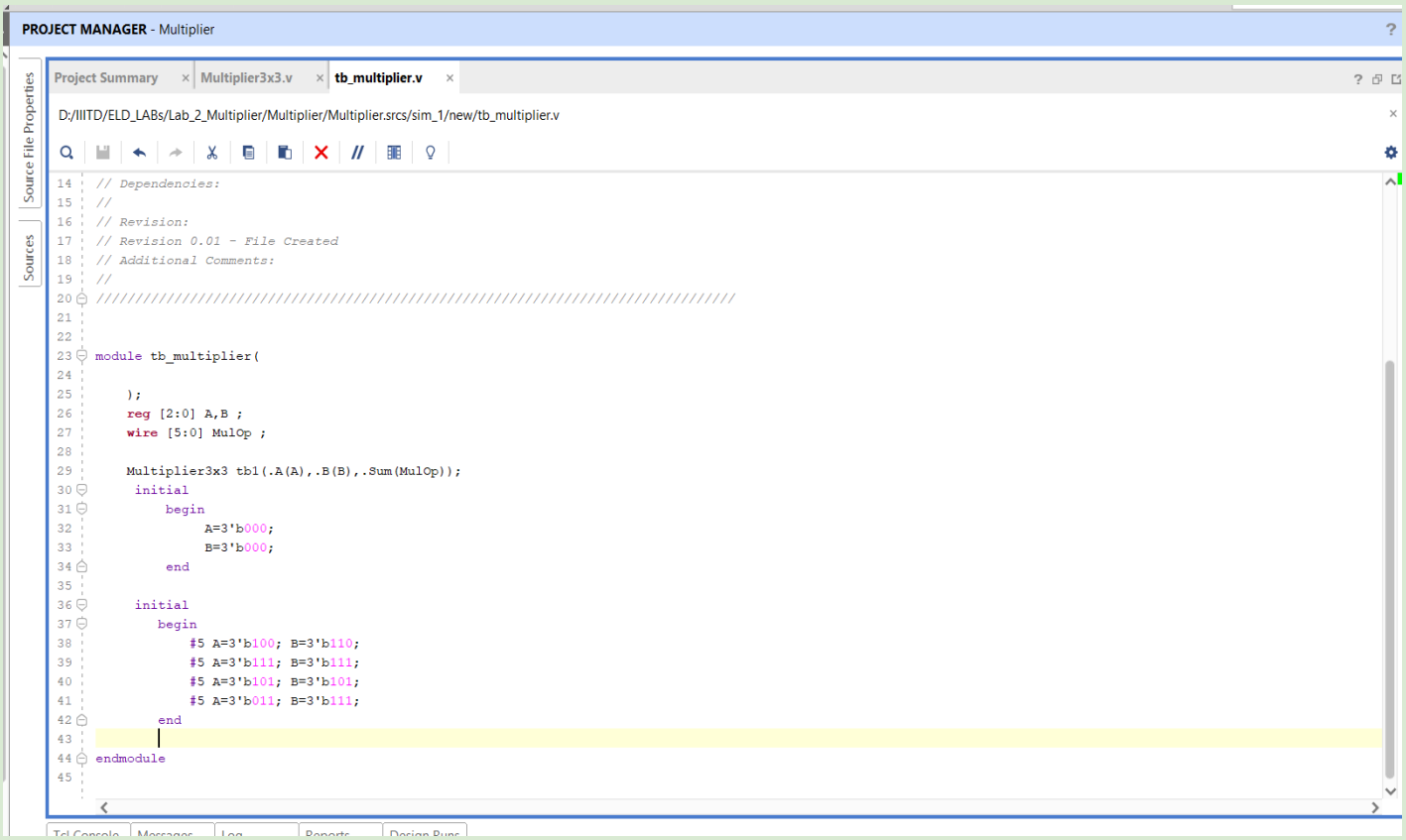
Results for manual Inputs :



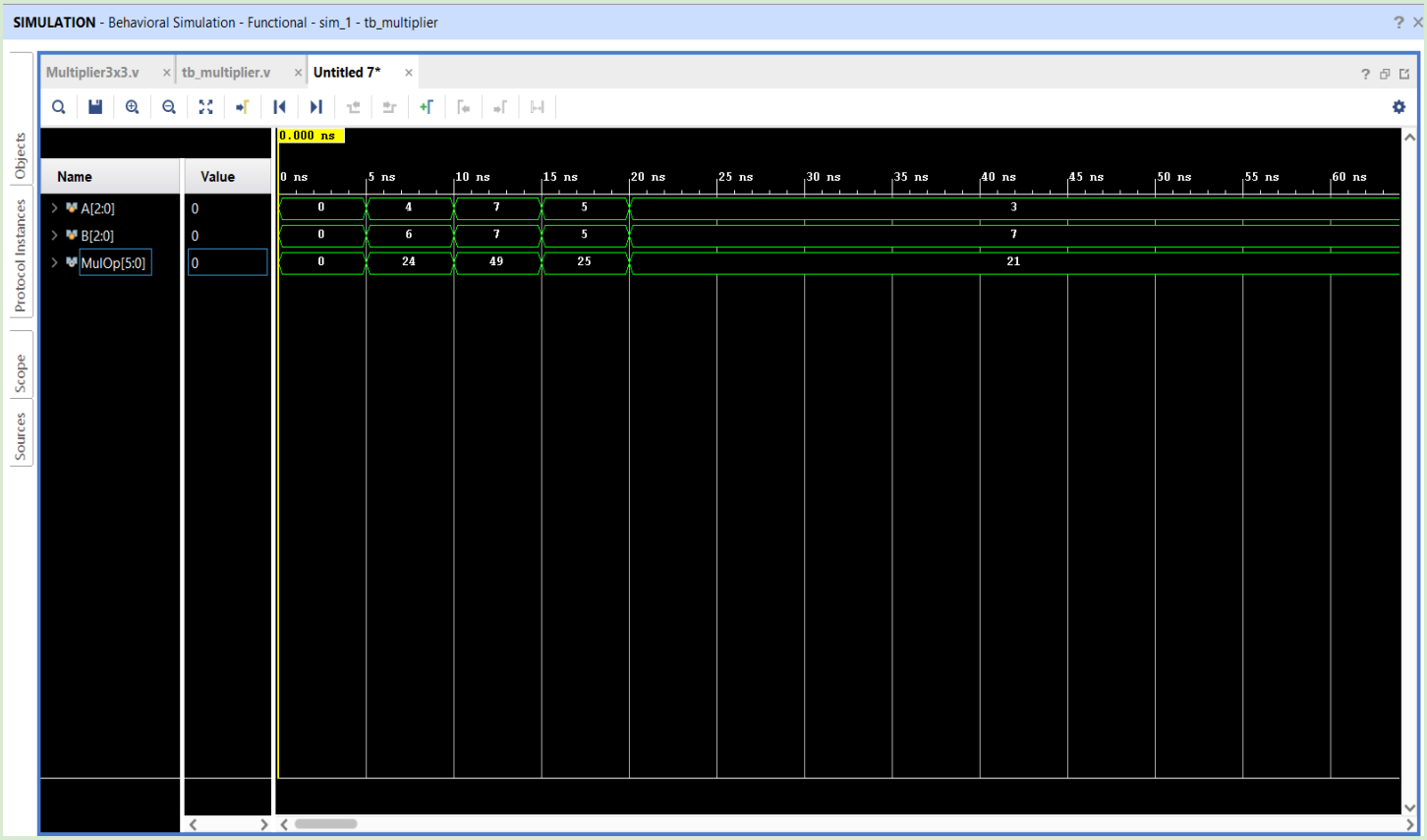




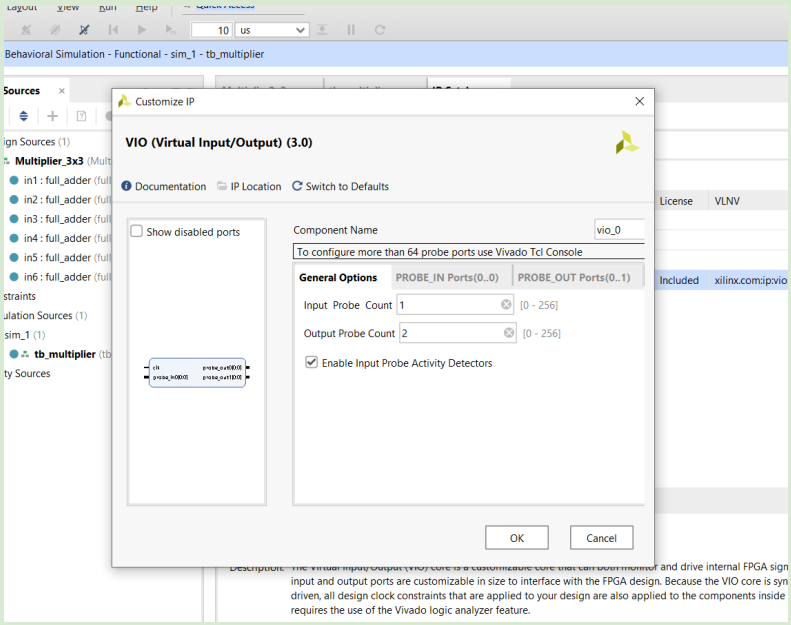
Code for test bench:

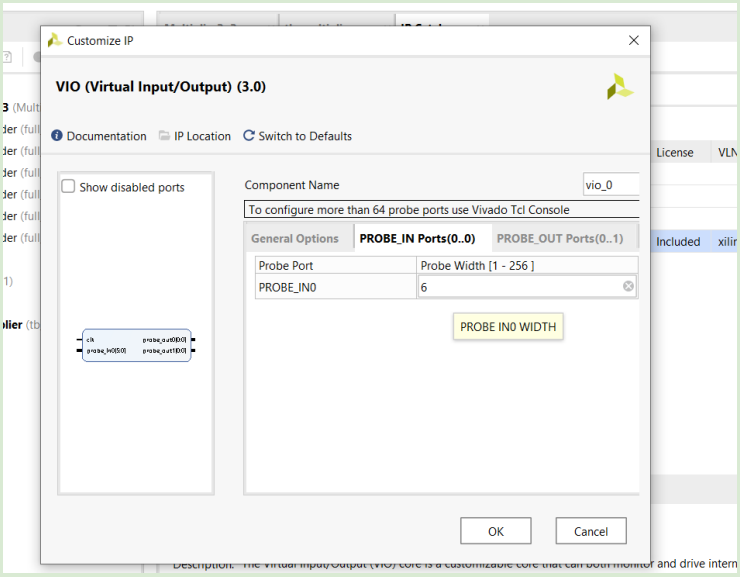


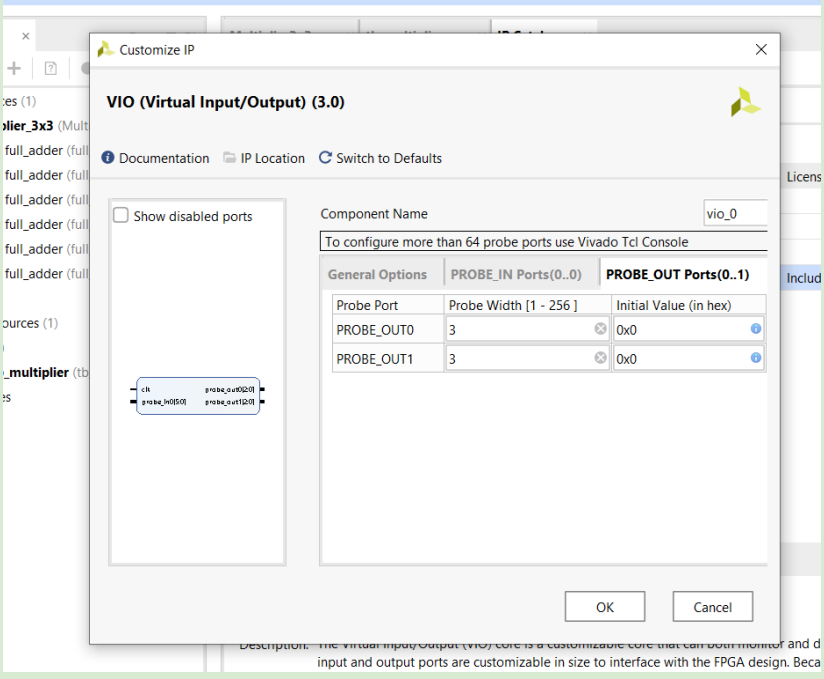
Results for Automated Testbench :

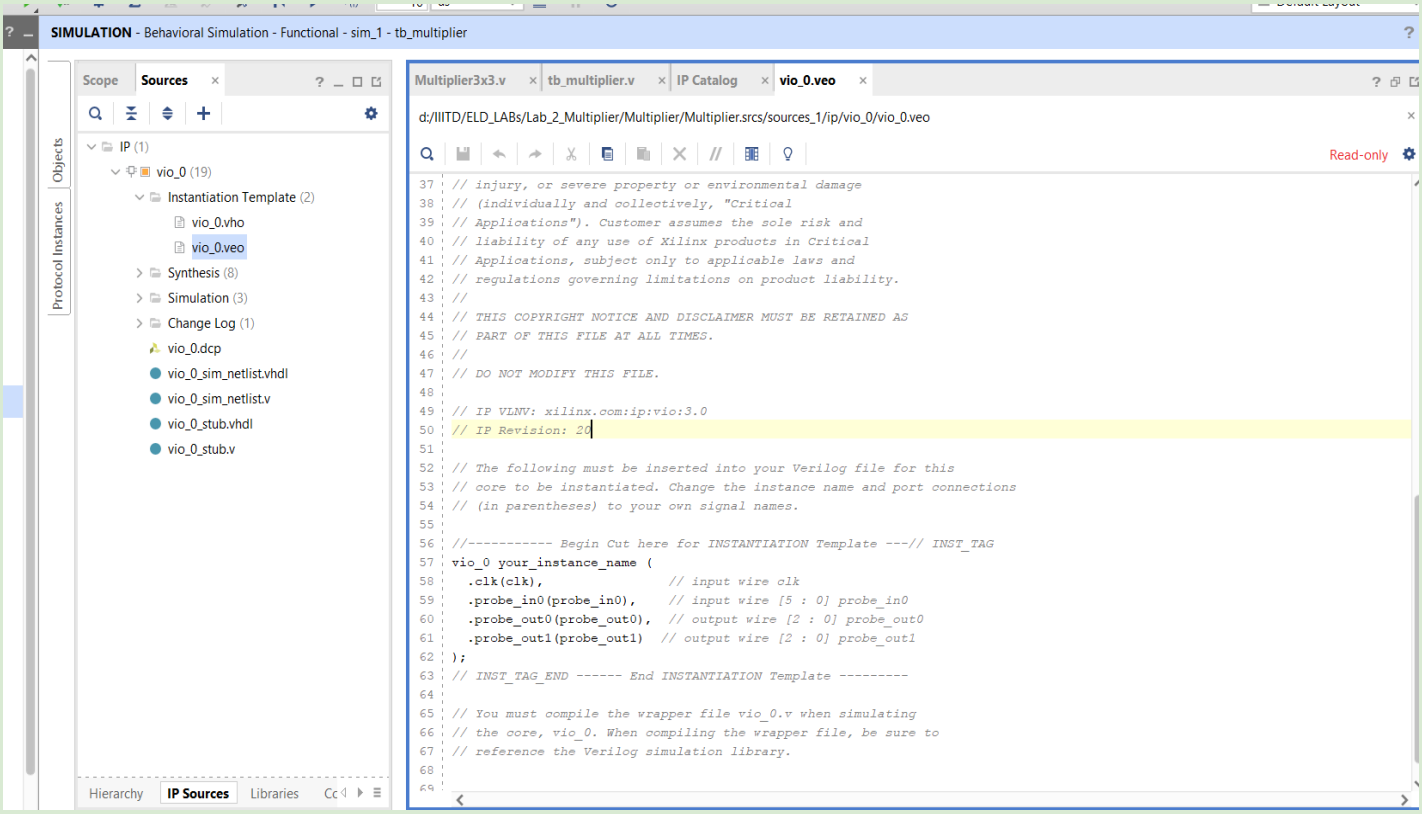


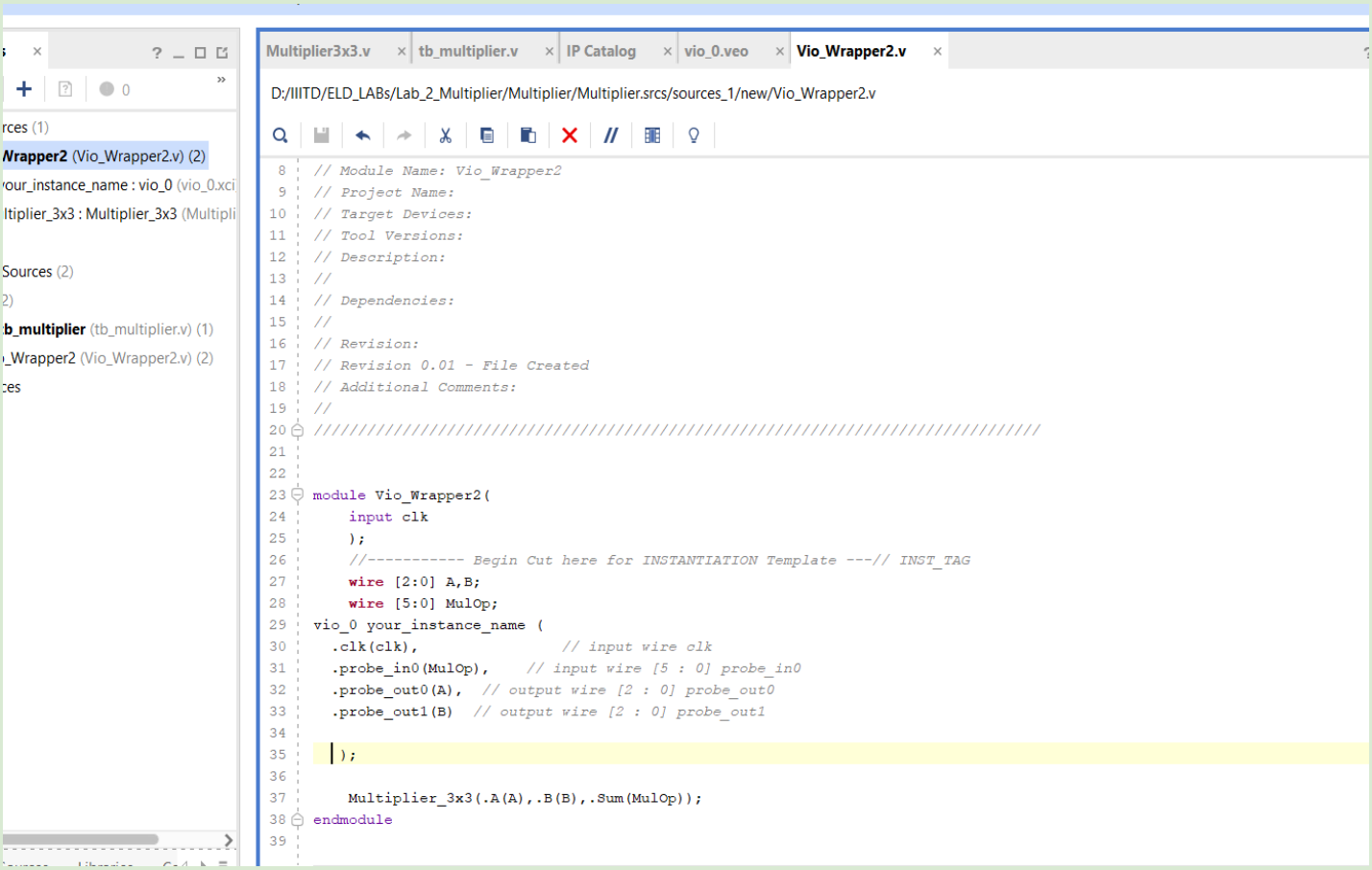
VIO Settings:



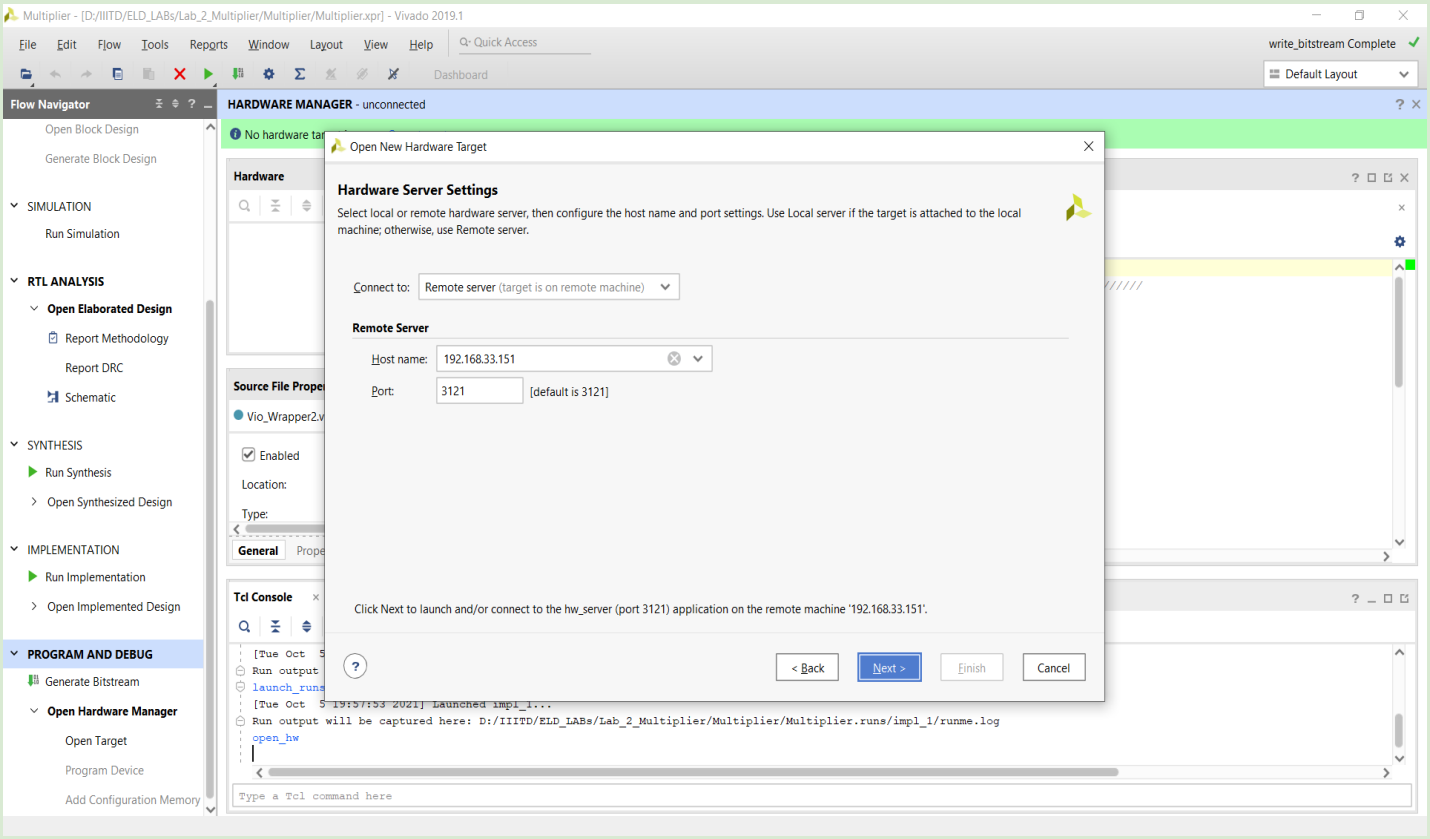


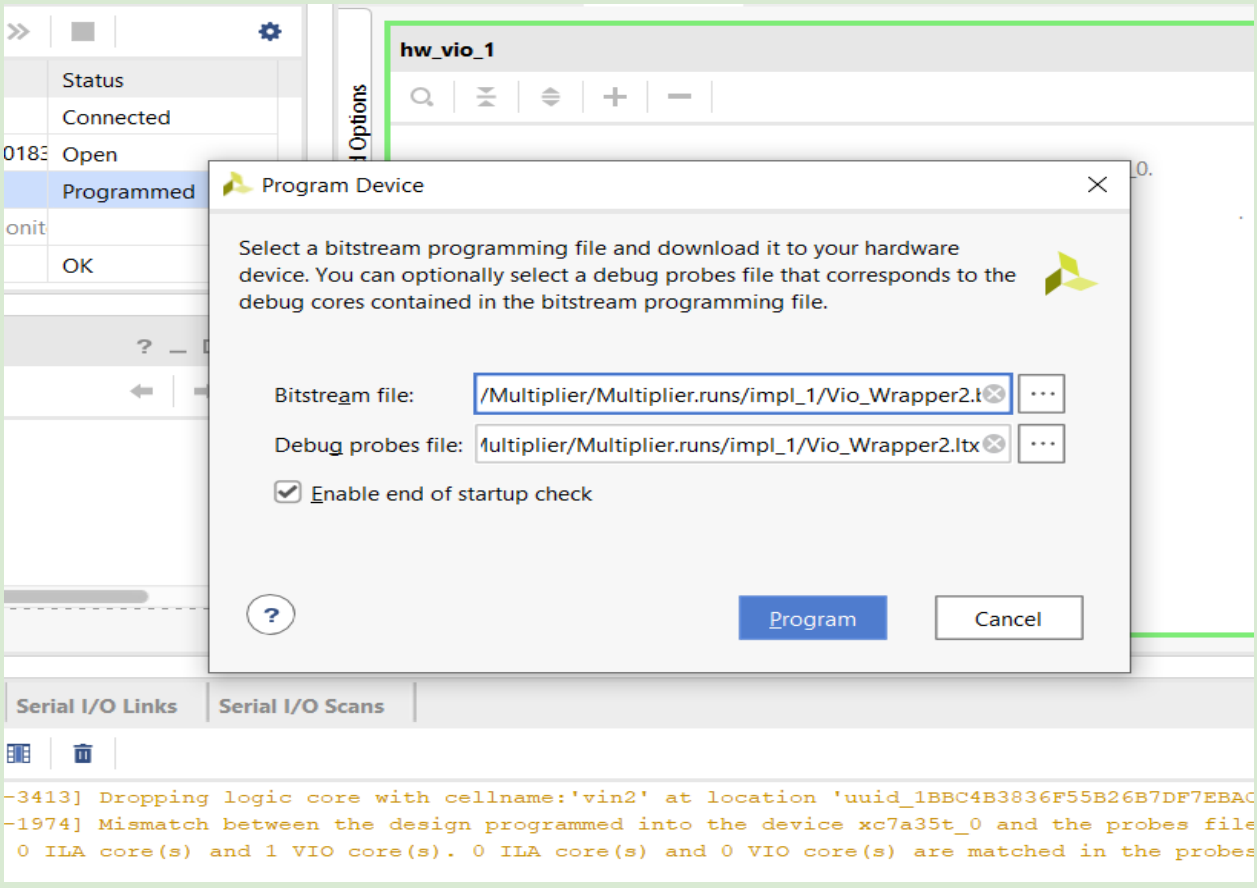



SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_multiplier
Scope
Sources
IP (1)
vio_0 (19)
Instantiation Template (2)
vio_0.vho
vio_0.veo
Synthesis (8)
Simulation (3)
Change Log (1)
vio_0.dcp
vio_0_sim_netlist.vhdl
vio_0_sim_netlist.v
vio_0_stub.vhdl
vio_0_stub.v
Hierarchy
IP Sources
Libraries
Multiplier3x3.v
tb_multiplier.v
IP Catalog
vio_0.veo
d:/IIITD/ELD_LABs/Lab_2_Multiplier/Multiplier/Multiplier.srcs/sources_1/ip/vio_0/vio_0.veo
Read-only
// injury, or severe property or environmental damage
// (individually and collectively, "Critical
// Applications"). Customer assumes the sole risk and
// liability of any use of Xilinx products in Critical
// Applications, subject only to applicable laws and
// regulations governing limitations on product liability.
//
// THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS
// PART OF THIS FILE AT ALL TIMES.
//
// DO NOT MODIFY THIS FILE.
// IP VLNV: xilinx.com:ip:vio:3.0
// IP Revision: 26
// The following must be inserted into your Verilog file for this
// core to be instantiated. Change the instance name and port connections
// (in parentheses) to your own signal names.
//----------- Begin Cut here for INSTANTIATION Template ---// INST_TAG
vio_0 your_instance_name (
.clk(clk), // input wire clk
.probe_in0(probe_in0), // input wire [5 : 0] probe_in0
.probe_out0(probe_out0), // output wire [2 : 0] probe_out0
.probe_out1(probe_out1) // output wire [2 : 0] probe_out1
);
// INST_TAG_END ------ End INSTANTIATION Template ---------
// You must compile the wrapper file vio_0.v when simulating
// the core, vio_0. When compiling the wrapper file, be sure to
// reference the Verilog simulation library.

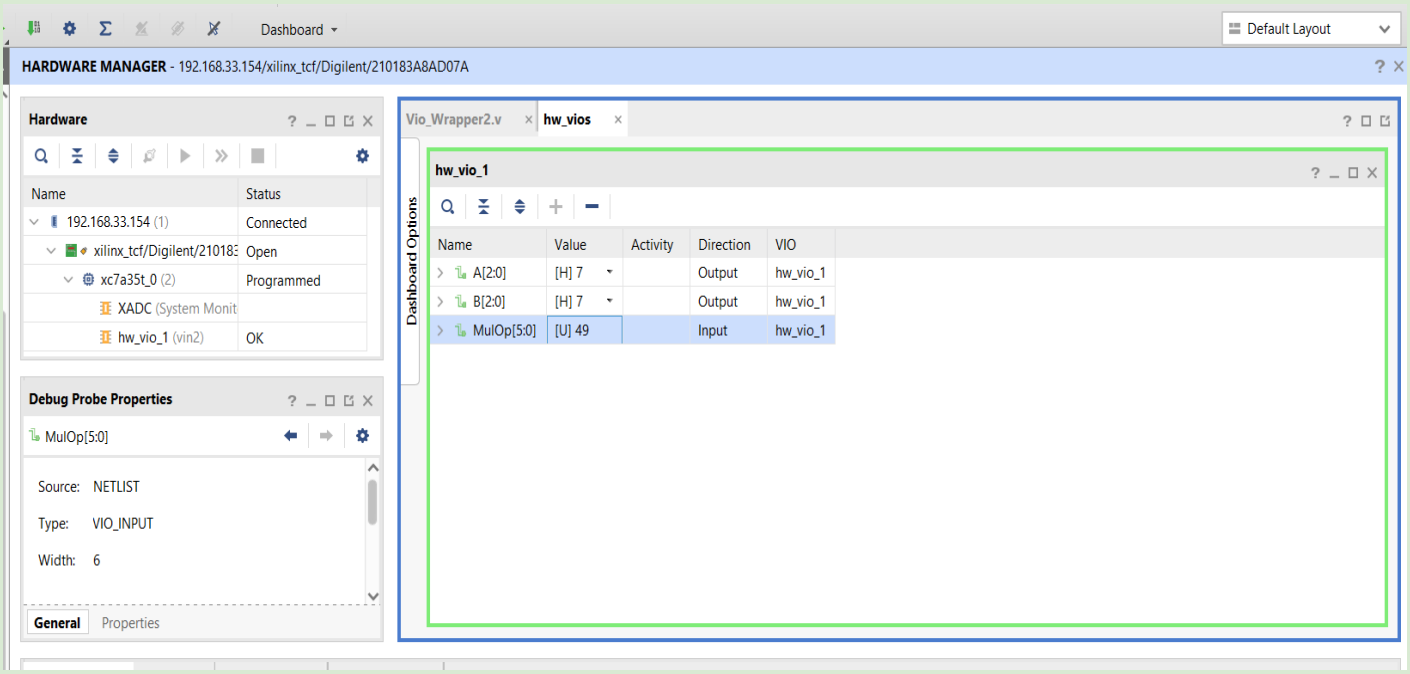

Multiplier3x3.v
tb_multiplier.v
IP Catalog
vio_0.veo
Vio_Wrapper2.v
D:/IIITD/ELD_LABs/Lab_2_Multiplier/Multiplier/Multiplier.srcs/sources_1/new/Vio_Wrapper2.v
// Module Name: Vio_Wrapper2
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
module Vio_Wrapper2(
input clk
);
//----------- Begin Cut here for INSTANTIATION Template ---// INST_TAG
wire [2:0] A,B;
wire [5:0] MulOp;
vio_0 your_instance_name (
.clk(clk), // input wire clk
.probe_in0(MulOp), // input wire [5 : 0] probe_in0
.probe_out0(A), // output wire [2 : 0] probe_out0
.probe_out1(B) // output wire [2 : 0] probe_out1
);
Multiplier_3x3(.A(A),.B(B),.Sum(MulOp));
endmodule

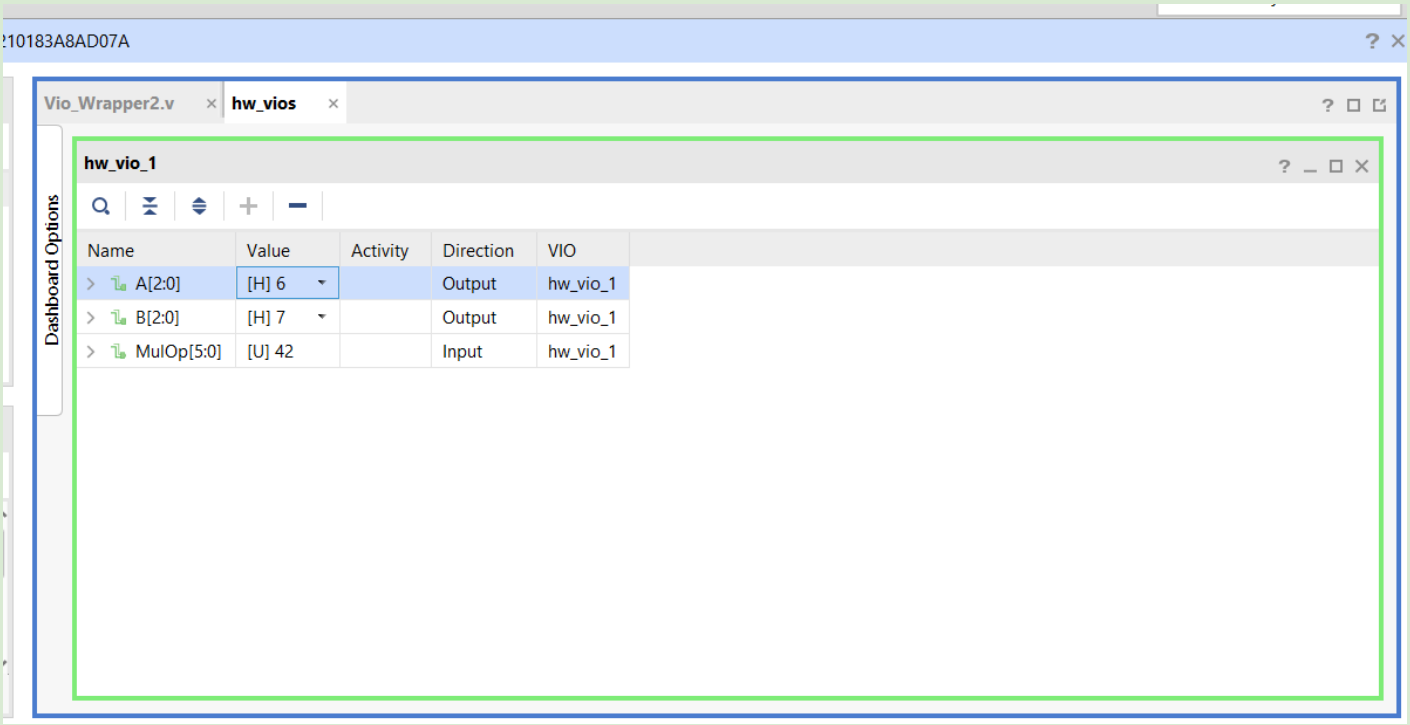
BitStream Generated Successfully:





Testing on BASYS3 Board using VIO IP





Conclusion:

Successfully tested functionality of 3-bit Full adder and 3x3 binary Multiplier on BASYS3 board using VIO IP, (Remote Access).