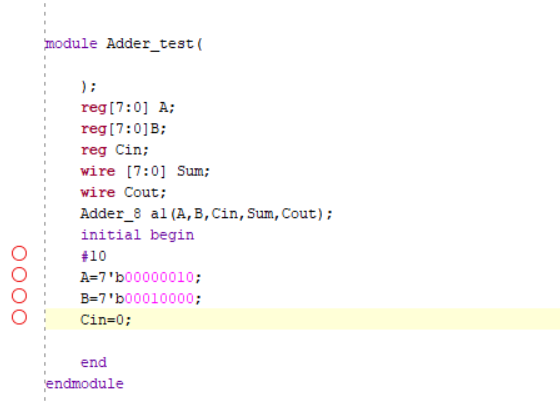
**Lab 11: Using VIO (Virtual Input/Output) IP for debugging**

**Instructor will explain about VIO IP. Proceed further only after the explanation.**

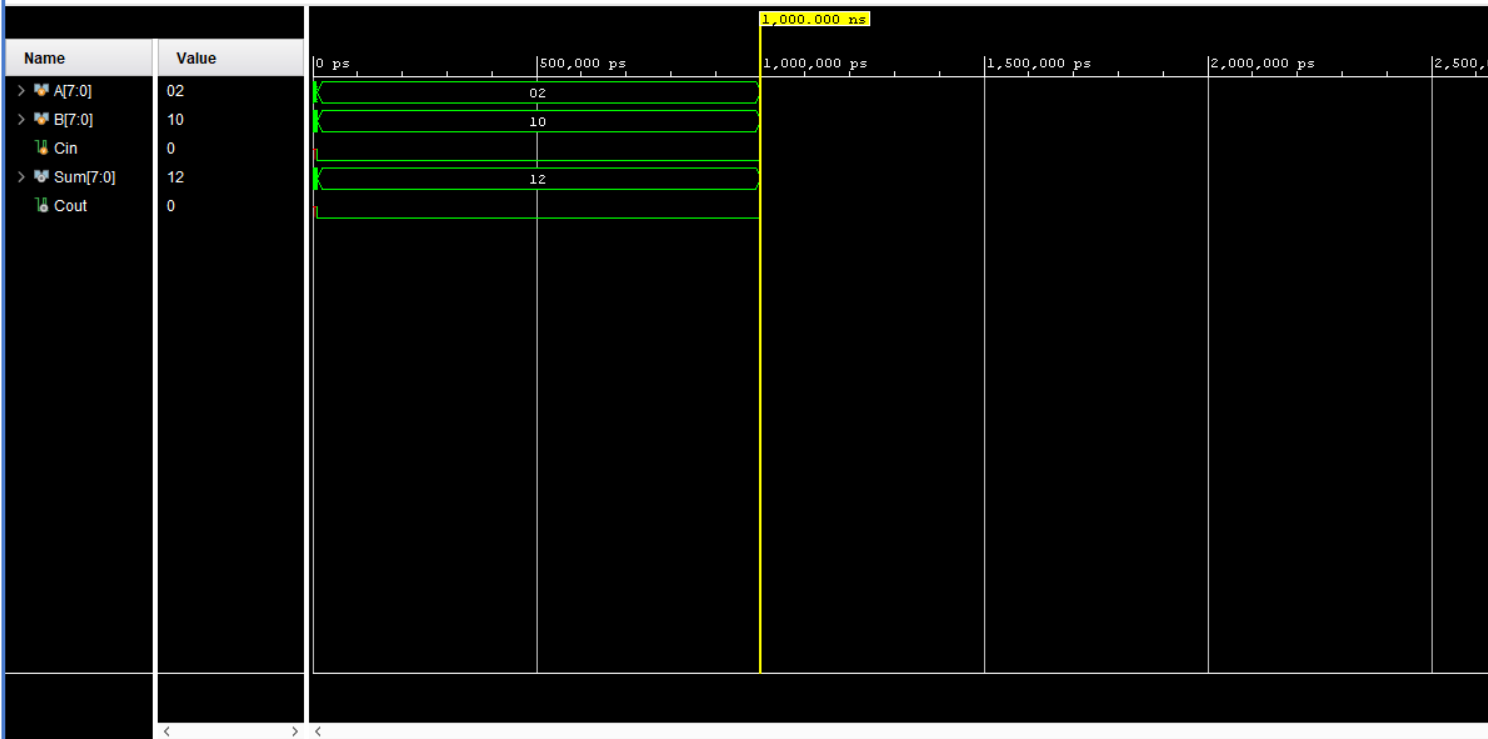
**Exercise 11.1: Implement an 8-bit Adder with Carry-in. Follow the steps shown below.**

1. Follow same process as earlier labs for creating a project and implement a Verilog module (**Adder\_8.v**) for an 8-bit adder with 8-bit inputs **A**, **B**, 1-bit Carry-in **Cin,** 8-bit **Sum** and1-bit Carry out **Cout.** (You can use any modelling style).
2. Test your adder by writing proper test bench (**Test\_adder\_8.v**) for testing your 8-bit adder.

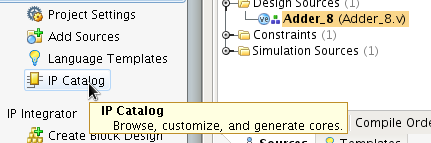
**Question: Paste the image test bench code (Test\_adder\_8.v**)**.**

Answer: 

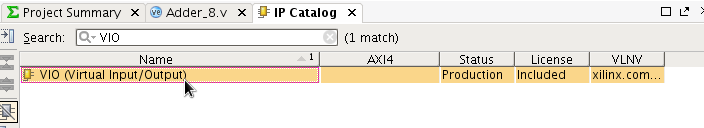
**Question: Paste the image showing the simulated waveforms.**

Answer: 

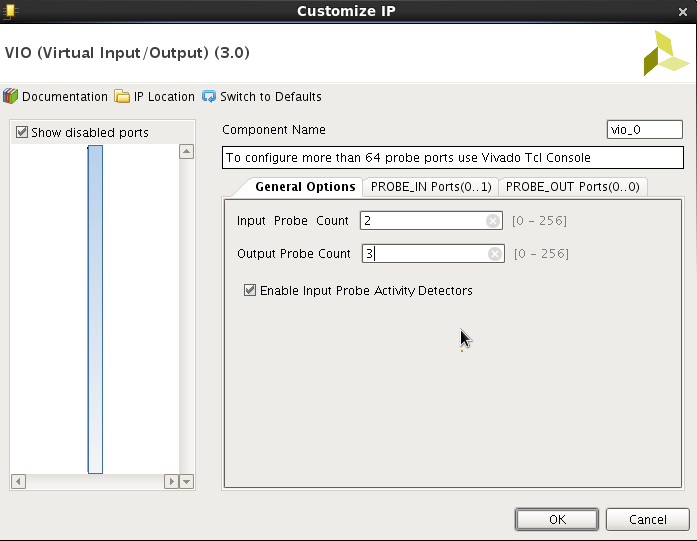
1. To debug the hardware implementation using VIO follow the steps below. Click on “IP catalog” which appears on the “Flow Navigator” window.

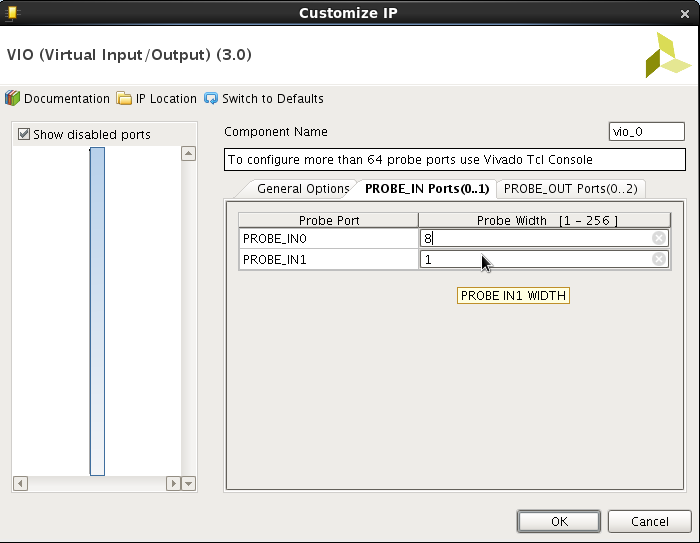


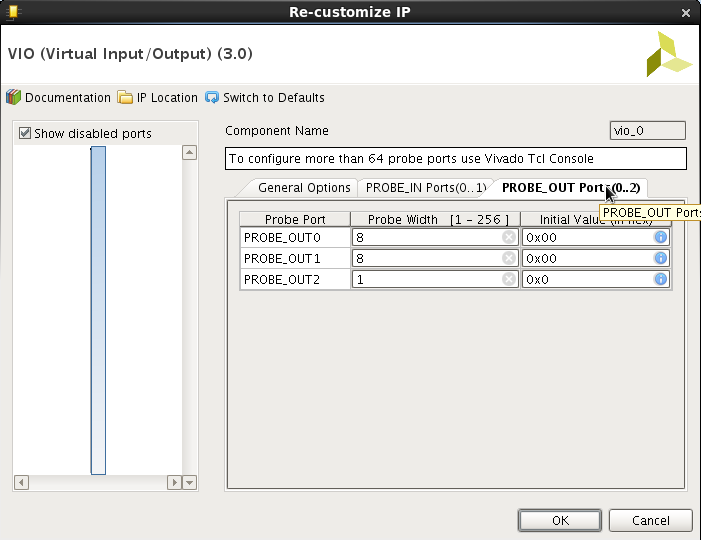
1. Now search for VIO in “IP Catalog” window and double click on the VIO (Virtual Input/Output) to configure it.



1. The inputs of your module (Adder\_8.v) should be connected as output probes for VIO because it will generate the inputs according to the values that you specify. Also the outputs of your module should be connected as input probes. Since Adder\_8 has two outputs (Sum, Cout) and three inputs (A, B, Cin). The input probes count should be selected as 2 and output probes count as 3. You will also be able to set the size for each of input and output probes by selecting PROBE\_IN Ports (for input probes) and PROBLE\_OUT Ports (For output probes).



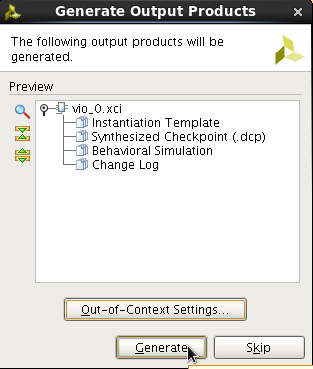




Here PROBE\_IN 0 is mapped to Sum and PROBE\_IN 1 is mapped to Cout. Similarly PROBLE\_OUT 0, PROBE\_OUT 1 and PORBE\_OUT 2 are mapped to A, B and Cin respectively.

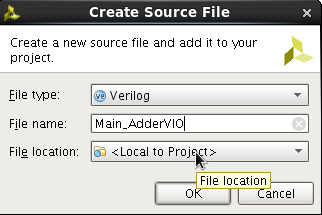
Click OK once configuration of VIO IP is done.

1. The following window will appear. Click on “Generate”. Then click “Ok”





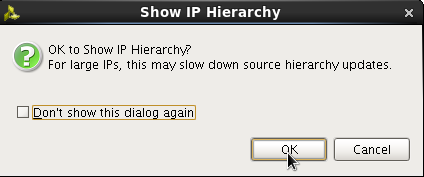
1. After this stage the VIO IP will be added to the project. To integrate this module with your adder module. Create another module which will instantiate both your module (Adder\_8.v) as well as the VIO IP. This is done by right clicking on the project 🡪”Add Sources” 🡪 “Add Design Sources” 🡪 “Create File”



Let the file name be **Main\_AdderVIO.v.** Assume this module has one input Clk (which is needed for VIO IP) and one 8-bit output Sum1 (to check the adder output on board). There will be no other inputs to this module since the values of A, B and Cin will be given through VIO IP.

1. Now to get the port information of VIO IP, click on the “expand” icon of the instance of VIO (Vio\_0.XCI) in the sources pane. Then click OK

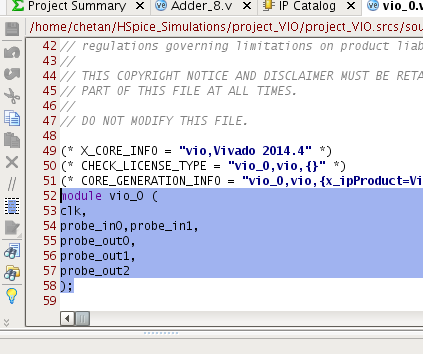




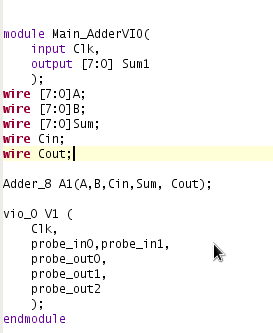
1. Now double click on the Vio\_0.v file. This will have the module definition of the VIO which you need for instantiating VIO in main module (**Main\_AdderVIO.v** )



1. The port information of VIO IP will be as below



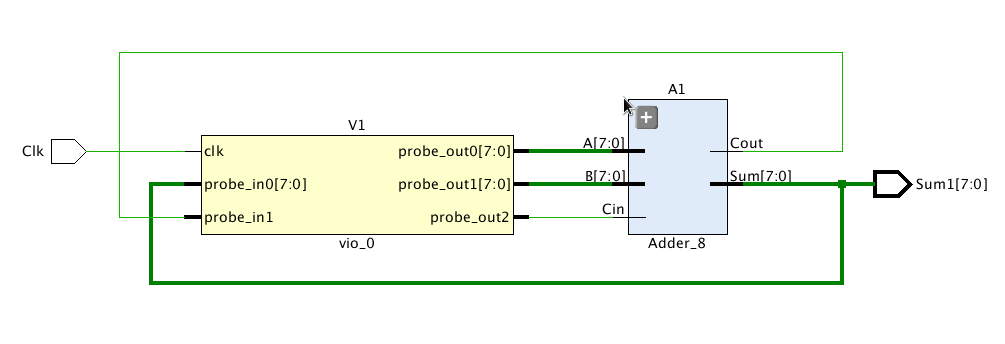
1. Using this information, instantiate VIO IP and Adder\_8 in Main\_AdderVIO.v. Incomplete instantiation is shown below. The inputs and outputs of the Adder\_8 are intermediate and hence defined as wire. Also the Sum is mapped to Sum1 to monitor the output on the Kit.



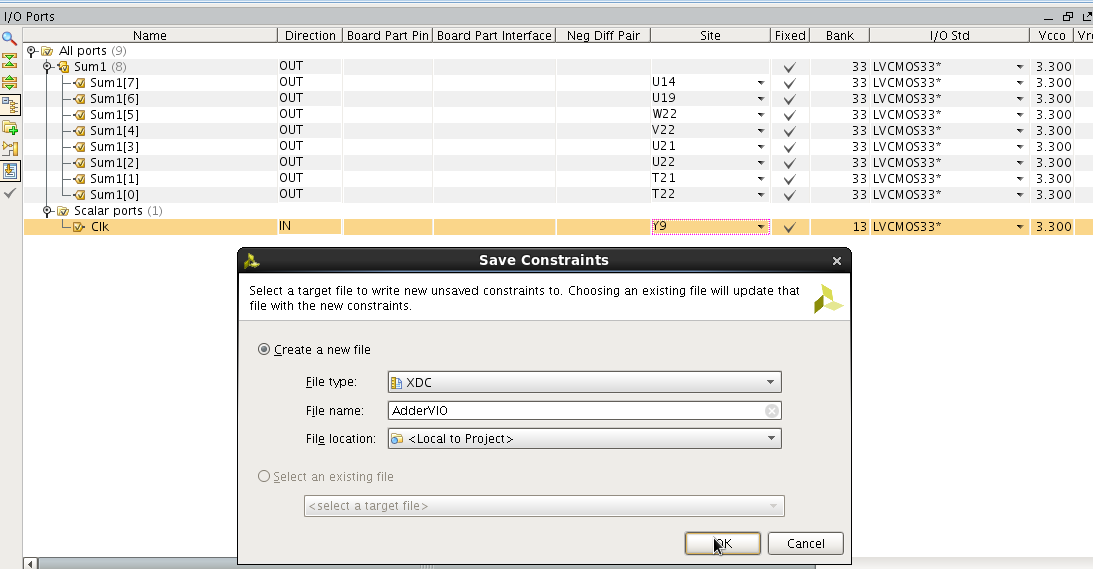


To complete the instantiation, replace clk, probe\_in0, probe\_in1, probe\_out0, probe\_out1 and probe\_out2 to **Clk**, **Sum**, **Cout**, **A**, **B**, **Cin**. Please note that the “**clk”** is already changed to “**Clk”** in above image.

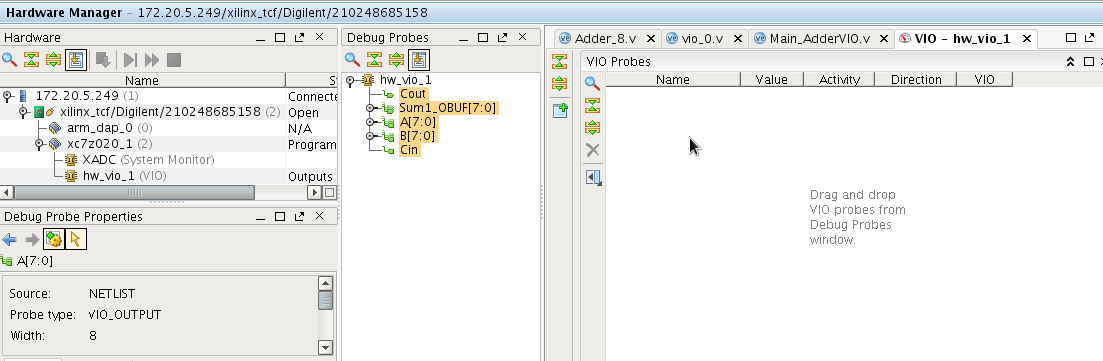
1. If everything is fine then your schematic after “RTL Analysis” Step will be as shown below.



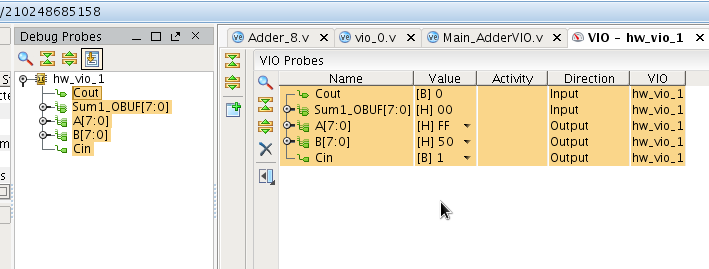
1. After the above step. All the remaining steps, which are briefly summarized below, will be same as earlier Labs.
2. I/O Planning 🡪 Save .xdc constraints file (your main module has only two ports Clk and Sum1)



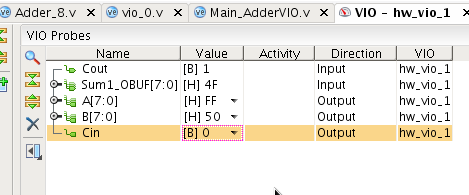
1. After I/O Planning, you can directly click on”Generate Bitstream”. Vivado will ask if earlier steps like synthesis and implementation have to be done. Click OK.
2. After the bitstream is generated, port your design on to FPGA (**Open Hardware Manager**🡪 **New Target**🡪… **Program Device**)
3. Once you program the device. The following window will open.

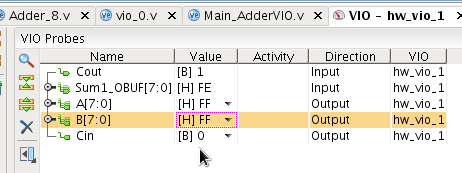


1. Now drag and drop all the signals (probes) that you want to check into VIO probes window.



1. Initially some random values will be there for all inputs. You can give new values to A, B, Cin by clicking on the corresponding signal/probe value. Please give a value for one of the signals and then type “Enter” for it to be reflected. The outputs will change in the VIO probe window as well as on the board. Here the values displayed are in hexadecimal. You can change the display radix by right clicking and selecting the radix to be hexadecimal, decimal or signed decimal.





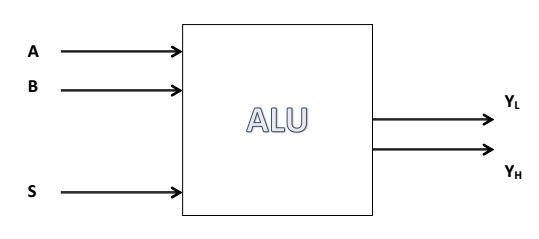
**Exercise 11.2: Implement an ALU and debug your implementation using VIO. Implementation can be done using Verilog (use operators like +, \* etc. directly).**

Implement an ALU with the following specifications.

* ALU has two inputs A, B of 8-bits each.
* S is a control Signal of 3-bits
* Y is a 16-bit output. YL represents least significant 8-bits of Y and YH represents most significant 8-bits of Y.
* The functionality of the ALU is given in Table below. Here OR, XOR AND represents bitwise operations

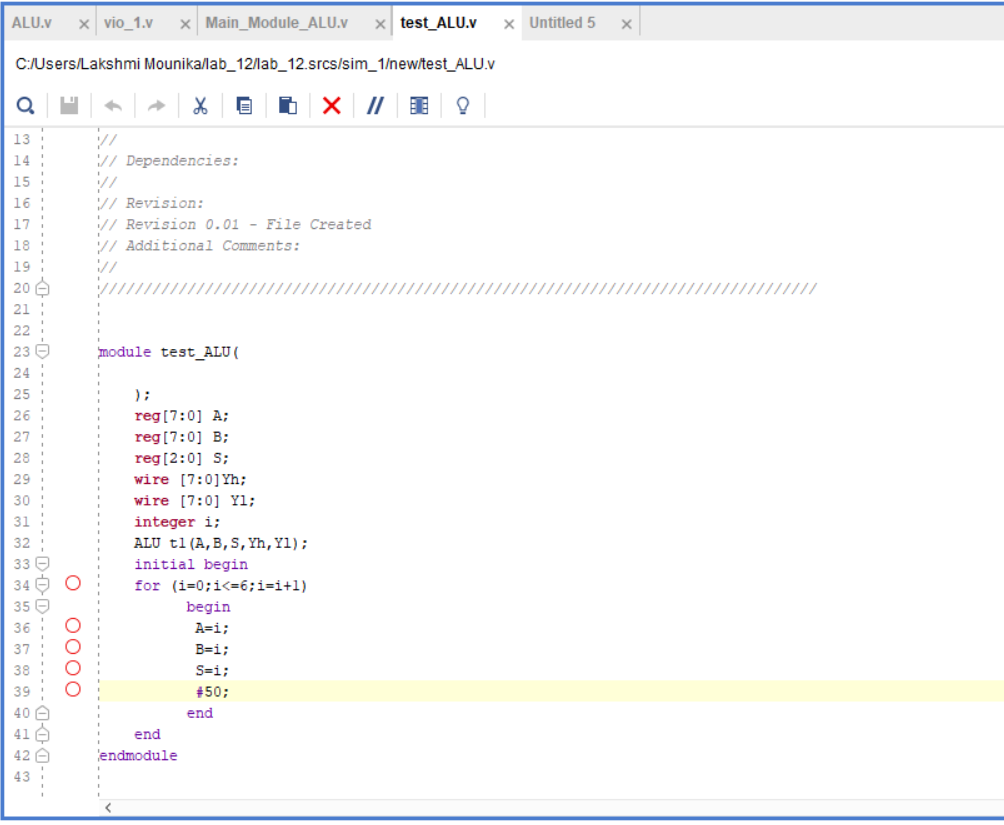
**ALU Functionality table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S** | **S2** | **S1** | **S0** | **ALU Function** | **OUTPUT** |
| 0 | 0 | 0 | 0 | ADD | Y = A+B |
| 1 | 0 | 0 | 1 | SUB | Y = A-B |
| 2 | 0 | 1 | 0 | MULT | Y = A\*B |
| 3 | 0 | 1 | 1 | INCR | YL= A+1 |
| 4 | 1 | 0 | 0 | SHIFT | YL= A<<B |
| 5 | 1 | 0 | 1 | AND | YL=A *AND* B |
| 6 | 1 | 1 | 0 | OR | YL=A *OR* B |
| 7 | 1 | 1 | 1 | XOR | YL= A *XOR* B |

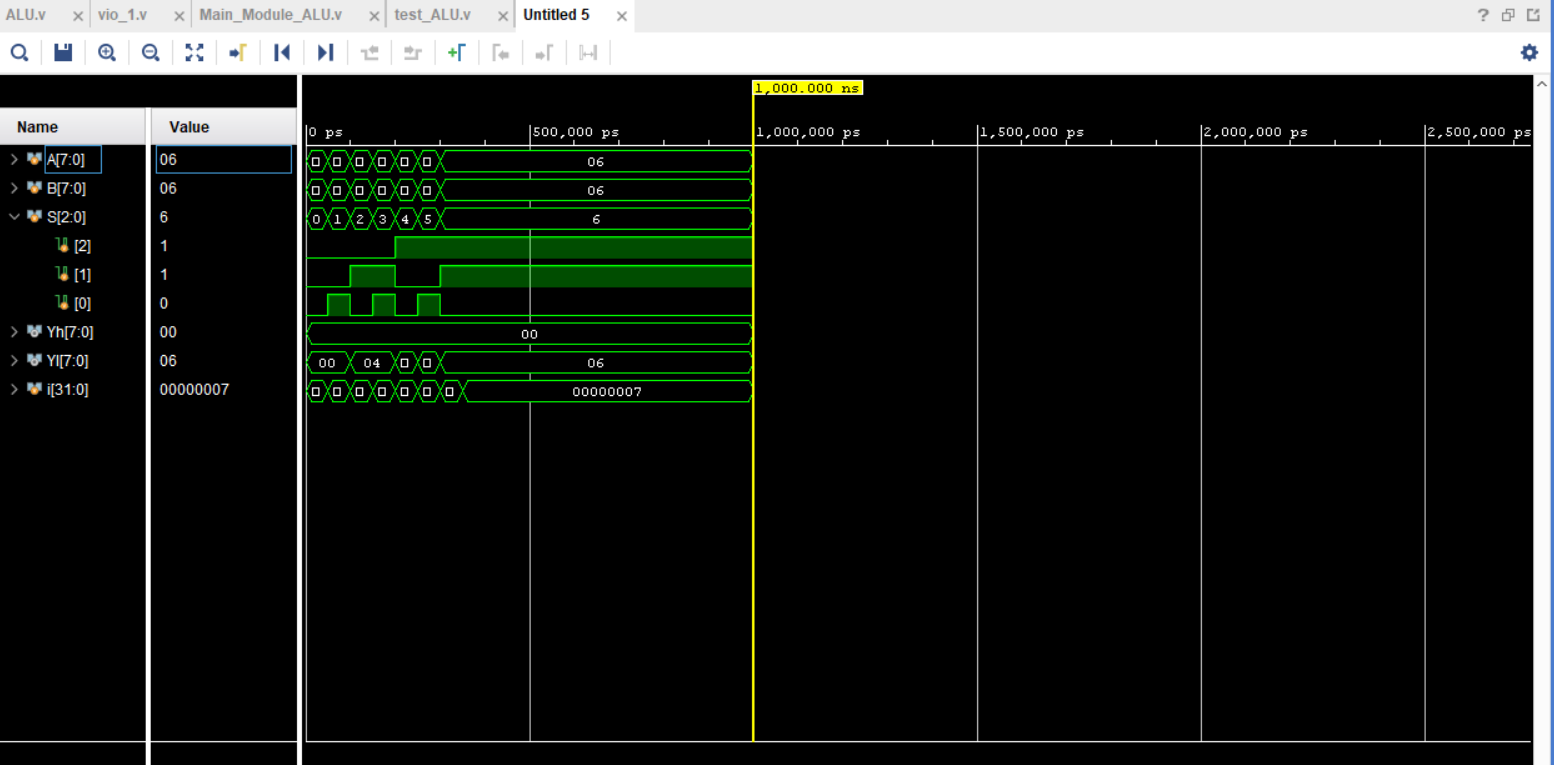


1. Test your adder by writing proper test bench for testing your ALU.

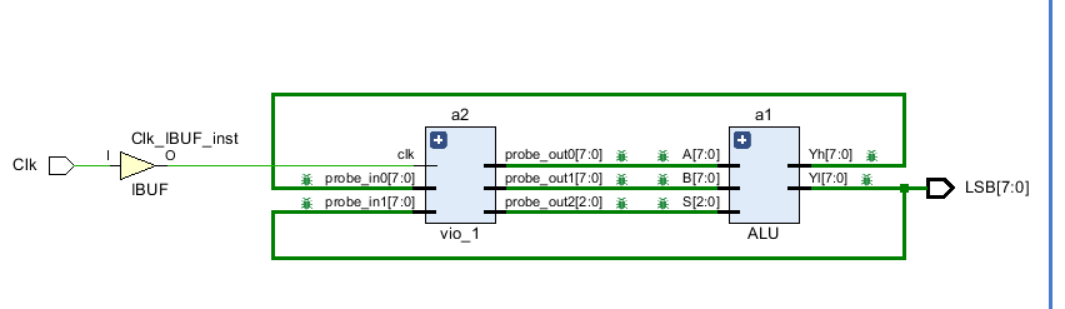
**Question: Paste the image test bench code.**

Answer: 

**Question: Paste the image showing the simulated waveforms.**

Answer: 

1. Elaborate the design (RTL Analysis) and paste the schematic below.

Answer: 

1. Do I/O Planning, Synthesize and implement the design or you can directly click on”Generate Bitstream”. Vivado will ask if earlier steps like synthesis and implementation have to be done. Click OK.
2. After the bitstream is generated, port your design on to FPGA (**Open Hardware Manager**🡪 **New Target**🡪… **Program Device**)
3. Show the **VIO** output window for few test cases and paste its image below.

Answer: Done in Lab

1. **List the concepts you learnt from this lab (Conclusions/Observations)**

Answer: The key concepts in the experiment are how we can generate the output virtually with the help of VIO IP. This method is helpful in the cases where we the output is more than 8 bit bus because it cannot be accommodated by FPGA DIP Switches. VIO communicates with the hostand displays the results on screen. It takes the output from the desired Verilog as input and the output of VIO is input to design. Apart from these two, there should be another module description to merge these two and to view our results on FPGA Board.