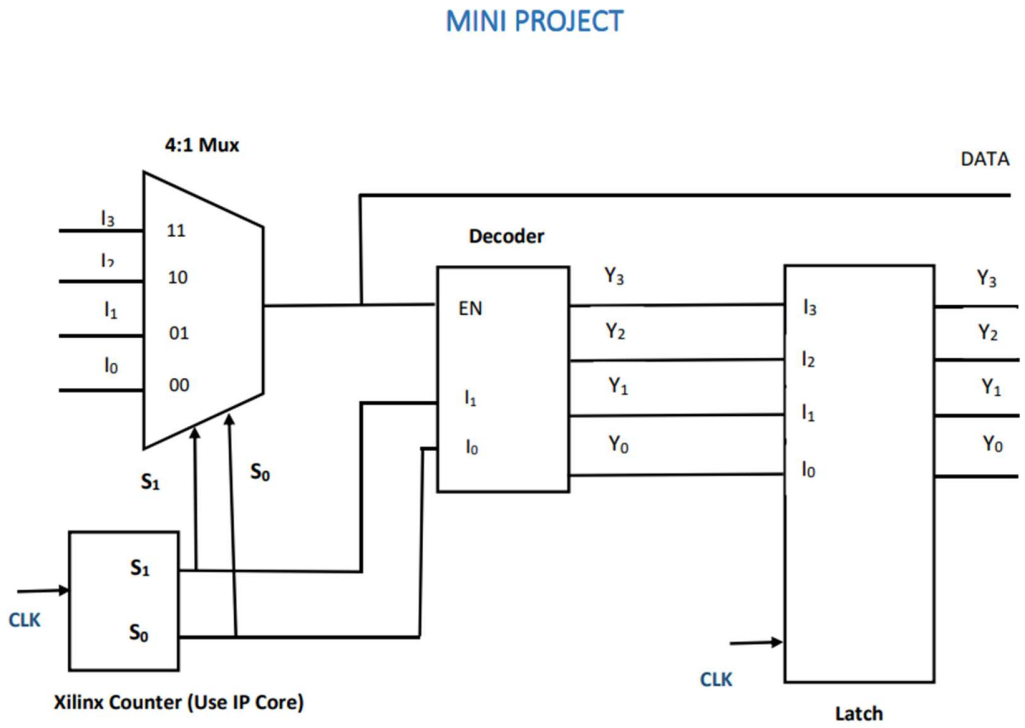
Objective-



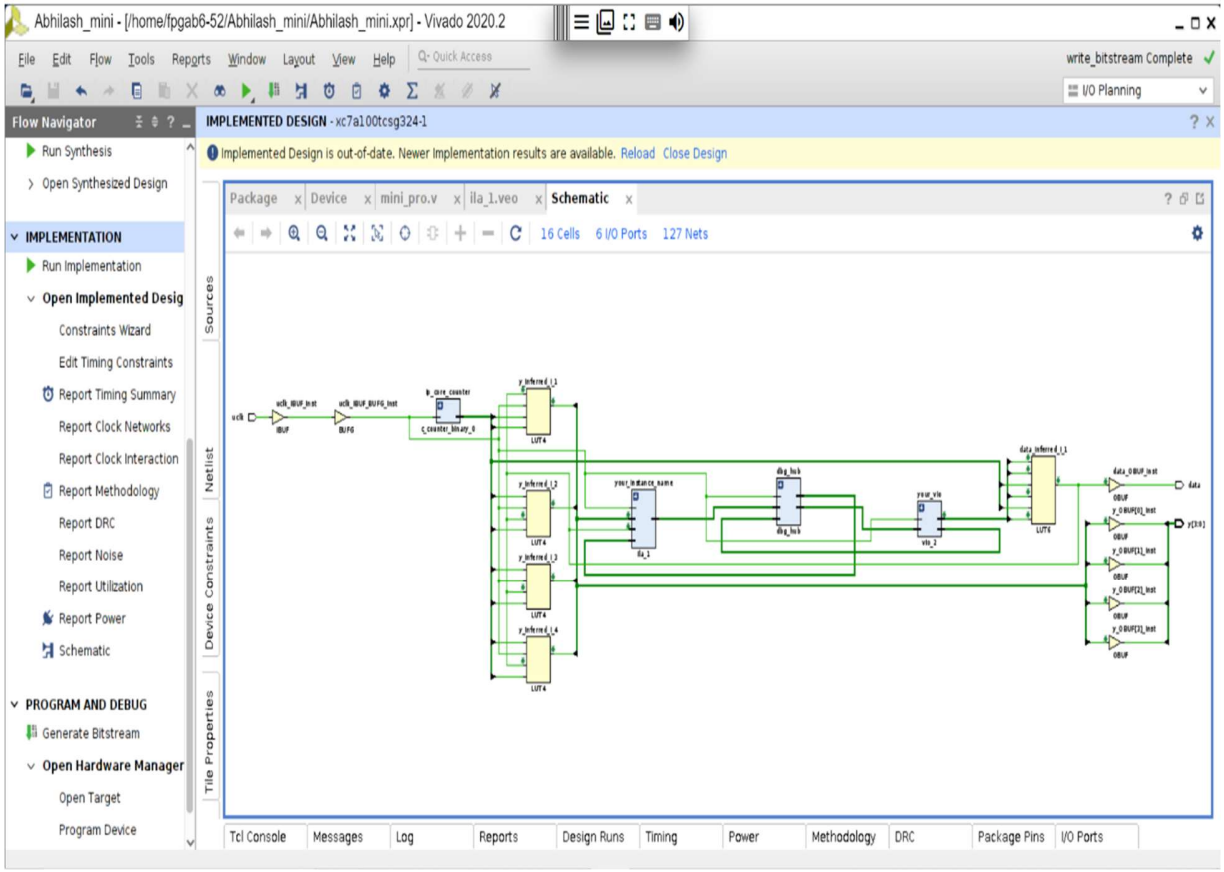
1. **Code the above design in verilog HDL and implement the same on the FPGA Kit allotted to you.**
2. **Use Virtual input and Output IP Core for giving input I3, I2, I1, I0 CLK from the Kit.**
3. **Y3 Y2 Y1 Y0 need to be displayed on Chipscope IP Core.**
4. **4-1 Mux use dataflow**
5. **Decoder use behaviour modeling**
6. **Latch use behaviour modeling**
7. **Counter- Use Xilinx IP Core.**
8. **Connect everything as miniproject.v and implement on the Kit**

Theory

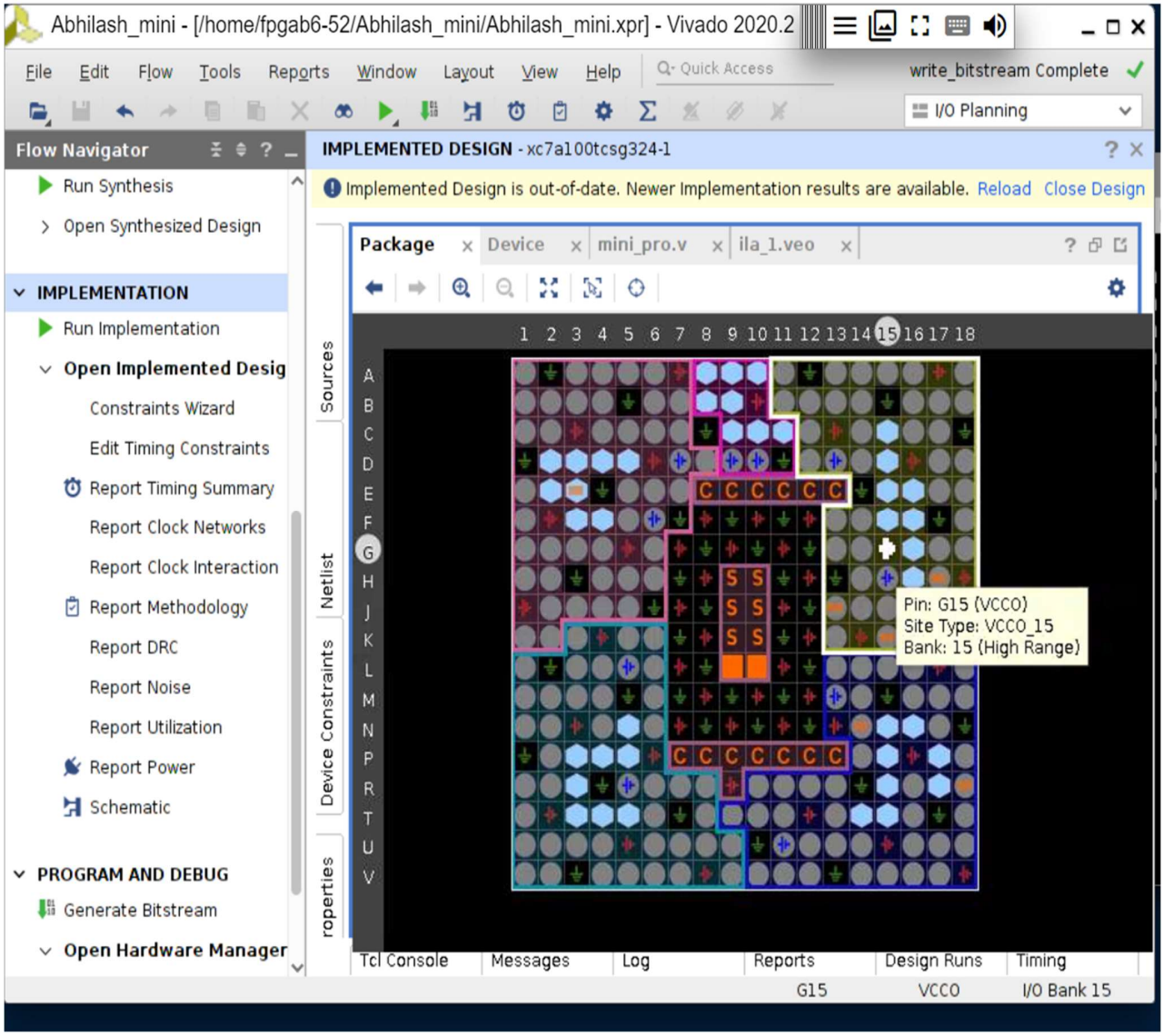
Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves

Implemented Design

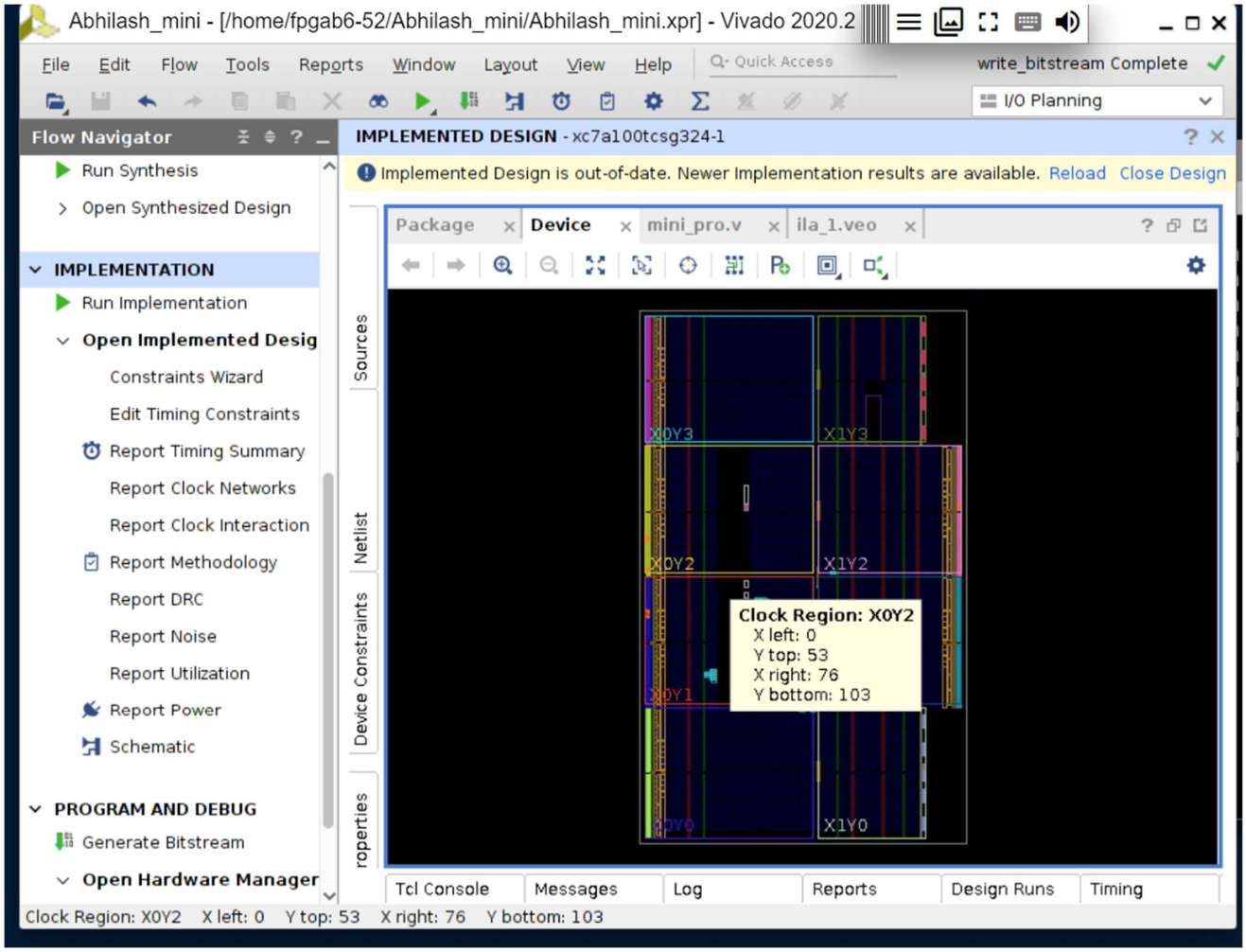
Schematic



Package


Abhilash_mini - [/home/fpgab6-52/Abhilash_mini/Abhilash_mini.xpr] - Vivado 2020.2
File Edit Flow Tools Reports Window Layout View Help
Quick Access
write_bitstream Complete
I/O Planning
Flow Navigator
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Desig
Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
IMPLEMENTED DESIGN - xc7a100tcsg324-1
Implemented Design is out-of-date. Newer Implementation results are available. Reload Close Design
Package
Device
mini_pro.v
ila_1.veo
Sources
Netlist
Device Constraints
roperties
Pin: G15 (VCCO)
Site Type: VCCO_15
Bank: 15 (High Range)
Tcl Console
Messages
Log
Reports
Design Runs
Timing
G15
VCCO
I/O Bank 15

Device


Abhilash_mini - [/home/fpgab6-52/Abhilash_mini/Abhilash_mini.xpr] - Vivado 2020.2
File Edit Flow Tools Reports Window Layout View Help
Quick Access
write_bitstream Complete
I/O Planning
Flow Navigator
IMPLEMENTED DESIGN - xc7a100tcsg324-1
Implemented Design is out-of-date. Newer Implementation results are available. Reload Close Design
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Desig
Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Package
Device
mini_pro.v
ila_1.veo
Sources
Netlist
Device Constraints
roperties
X0Y3
X1Y3
X0Y2
X1Y2
X0Y1
X0Y0
X1Y0
Clock Region: X0Y2
X left: 0
Y top: 53
X right: 76
Y bottom: 103
Tcl Console
Messages
Log
Reports
Design Runs
Timing
Clock Region: X0Y2 X left: 0 Y top: 53 X right: 76 Y bottom: 103

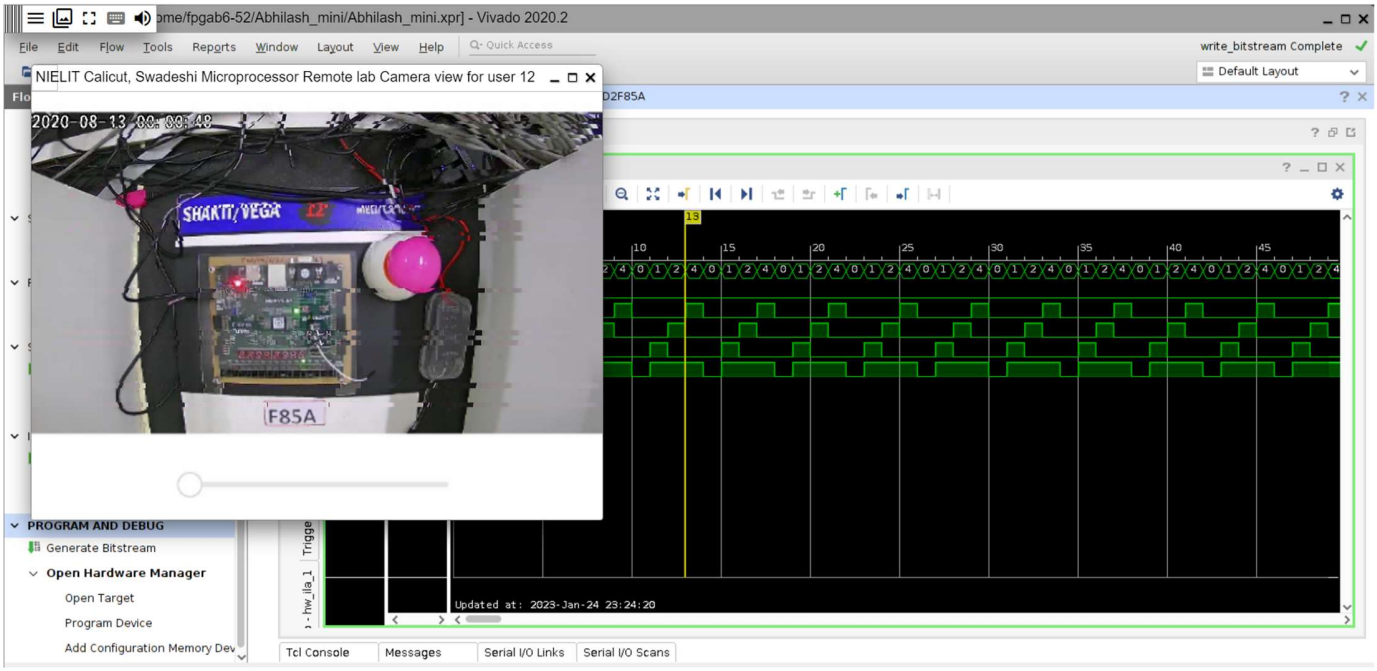
Source Code-Verilog

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////
// Company:
// Engineer:
//
// Create Date: 23.01.2023 16:56:14
// Design Name:
// Module Name: mini_pro
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//////////////////


module mini_pro(
input uclk,
output data,
output reg [3:0]y

    );

    wire [3:0]I;
    wire [1:0]s;
    reg [3:0]decout;
    //counter ip instantiation

c_counter_binary_0  ip_core_counter(
  .CLK(uclk),  // input wire CLK
  .Q(s)      // output wire [1 : 0] Q
);

//mux vio instantiation
vio_2 your_vio (
```

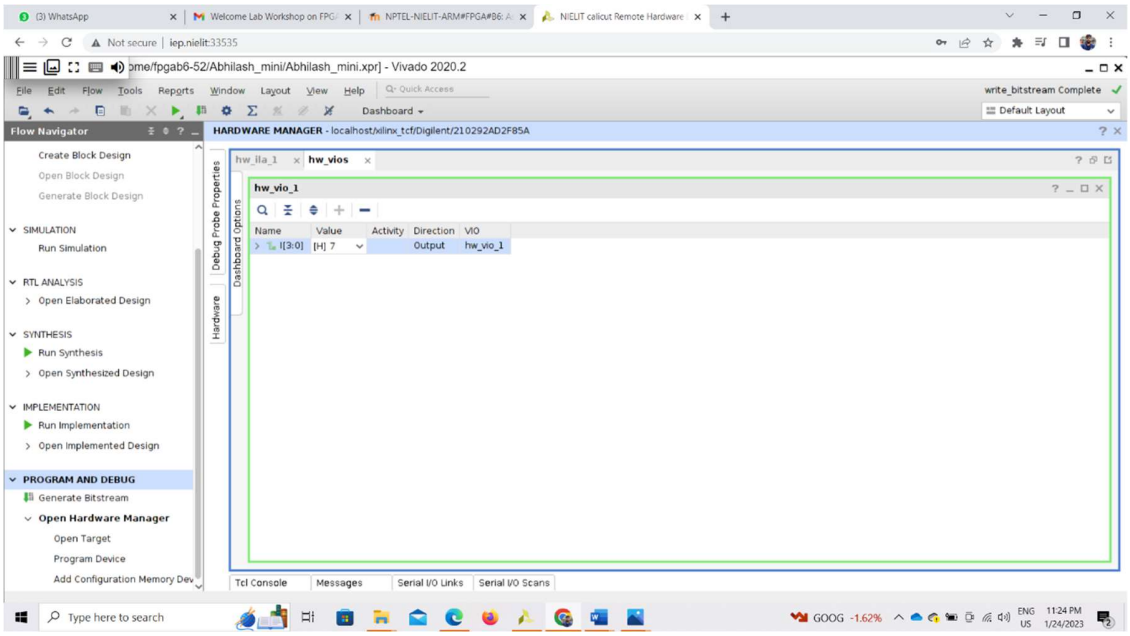
```
  .clk(uclk),                 // input wire clk
  .probe_out0(I)  // output wire [3 : 0] probe_out0
);

assign data = s[1] ? (s[0] ? I[3] : I[2]) : (s[0] ? I[1] :
I[0]);
//decoder behaviour modeling,
   always @(data,s)
     begin
       // using condition if statement
       // implement the 2:4 truth table
       if(data==1)
         begin
           if(s==2'b00) decout=4'b0001;
           else if(s==2'b01) decout=4'b0010;
           else if(s==2'b10) decout=4'b0100;
           else if(s==2'b11) decout=4'b1000;
           else decout=4'bxxxx;
         end
       else
        decout=4'b0000;
     end
 //latch final output
       always @(decout,uclk)
     begin
       // using condition if statement
       // implement the 2:4 truth table
       if(uclk==1) y<=decout;
       else y<=decout;
     end
//ila chipscpe
ila_1 your_instance_name (
     .clk(uclk), // input wire clk
     .probe0(y), // input wire [3:0]  probe0
     .probe1(data) // input wire [0:0]  probe1
);

endmodule
```

RESULT-

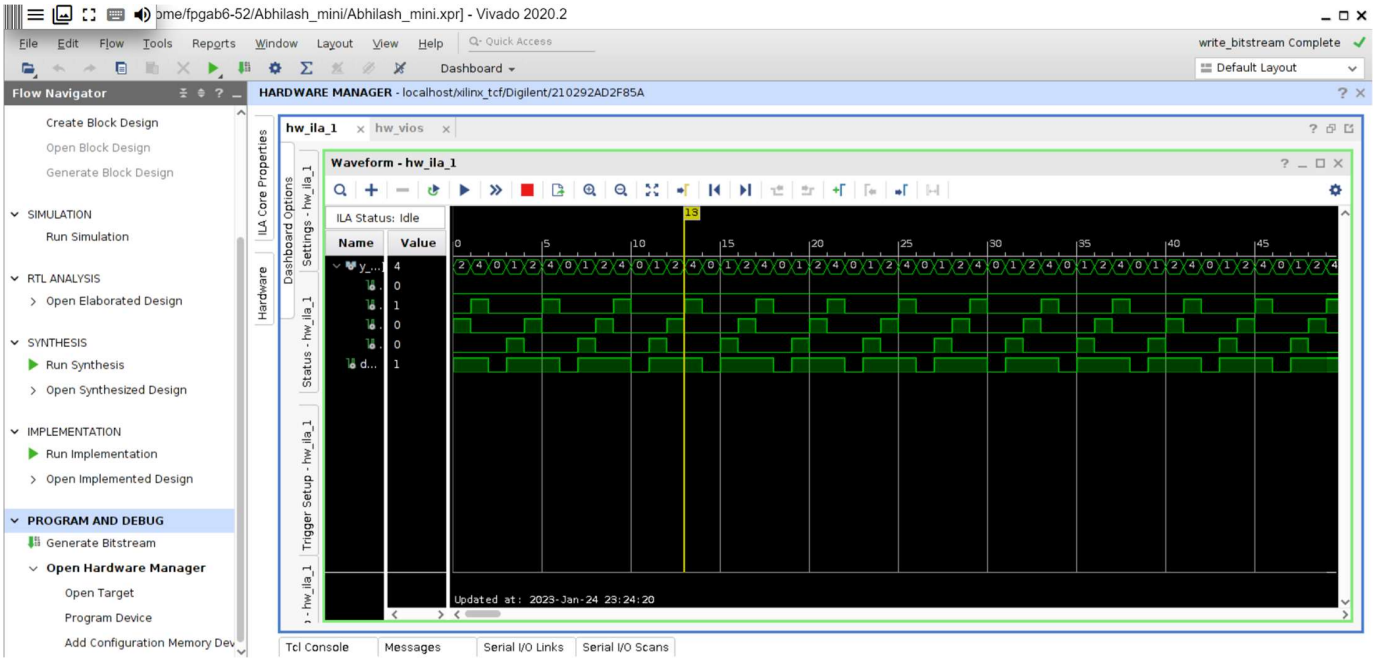
Screenshots at different value of vio



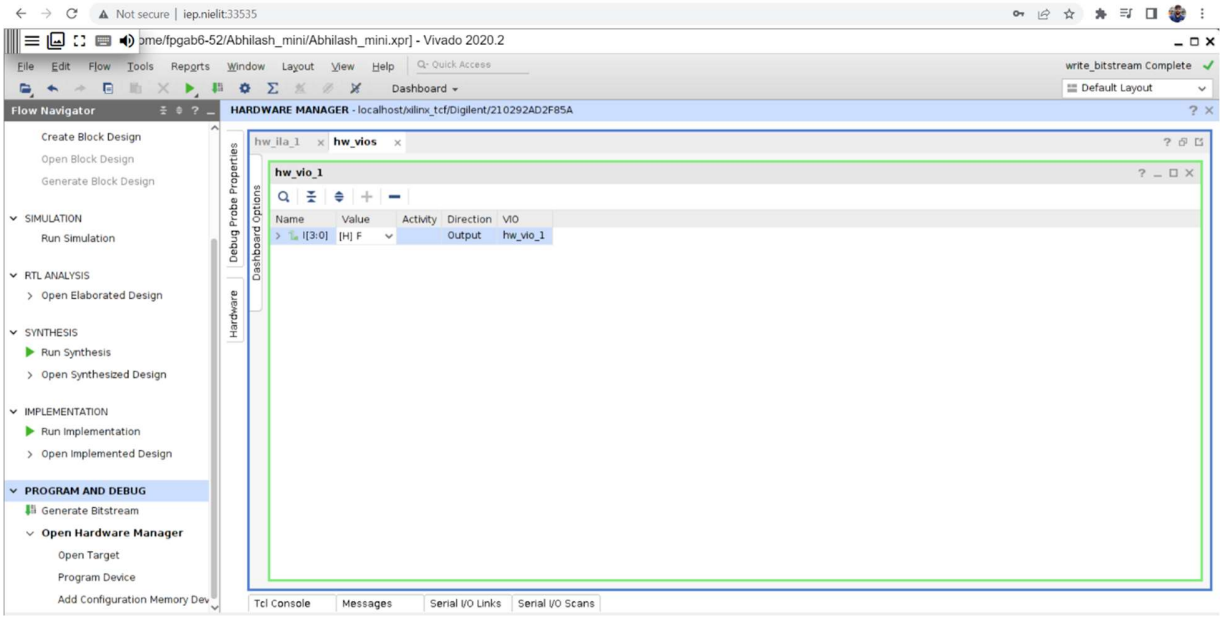
Input-(0111)



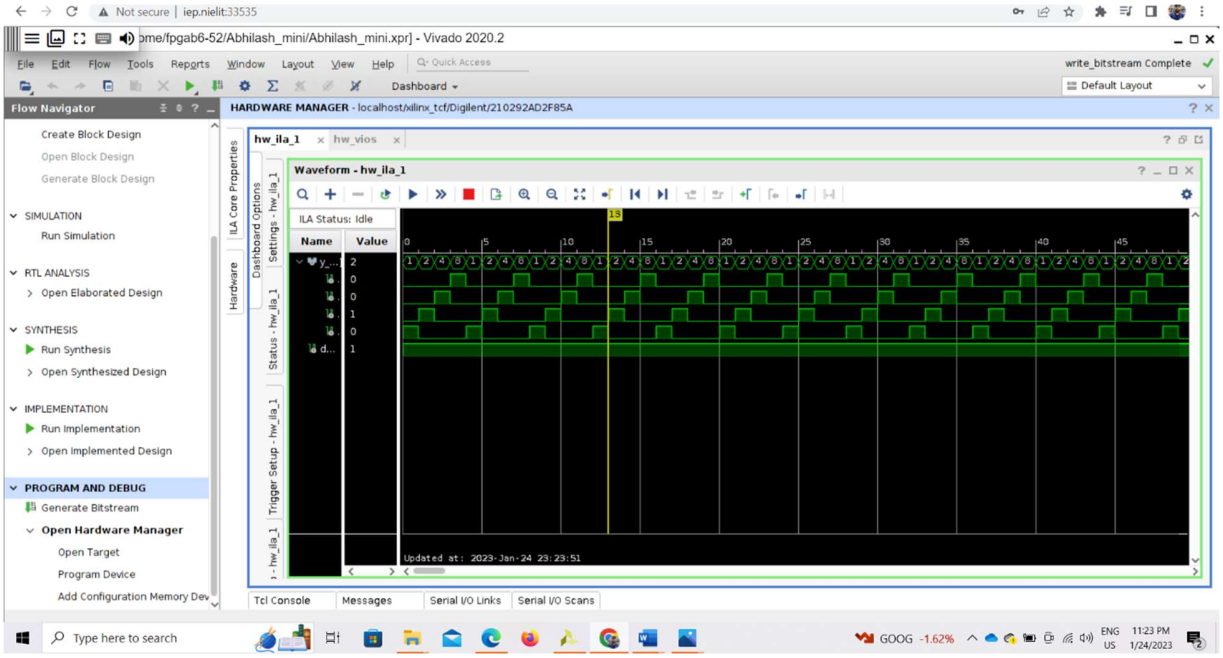
Output



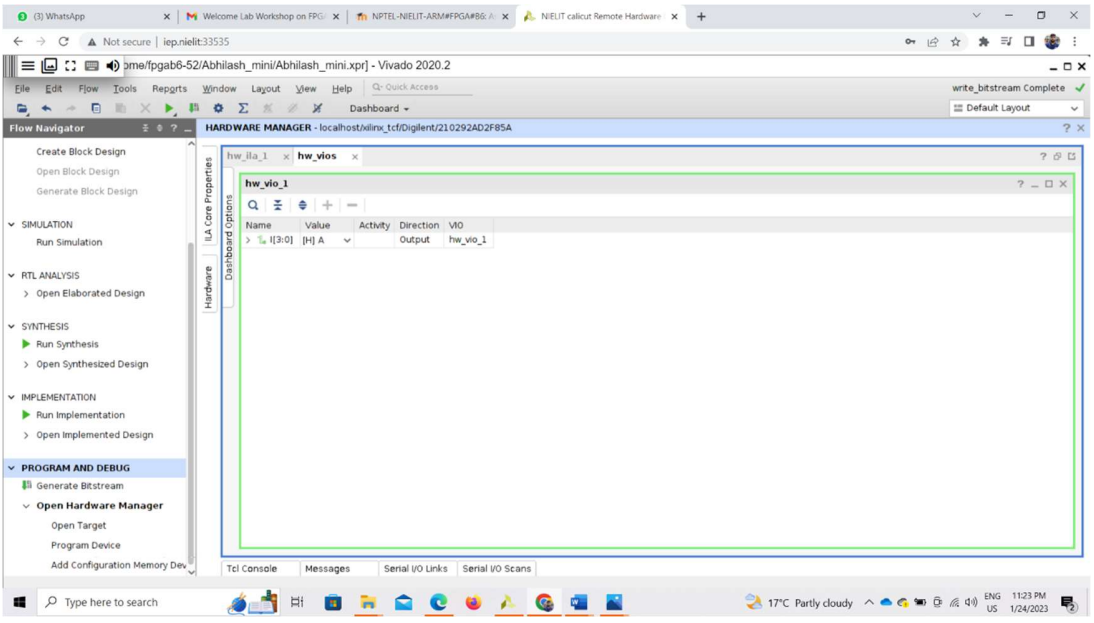
Input-(1111)



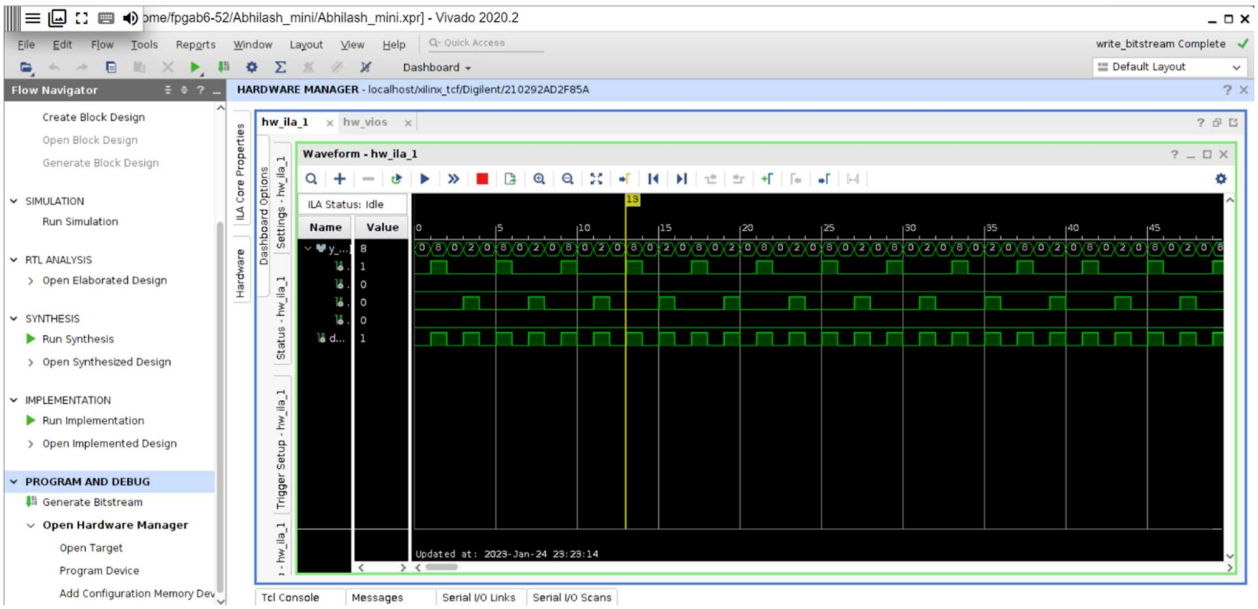
Output



Input-(1010)



Output



Conclusion

The ILA and VIO are free customizable IPs from Xilinx. The ILA IP helps you easily probe internal signals inside your FPGA and bring them out into a simulation-like environment to monitor them and verify their behavior.

Unlike ILA, the VIO IP allows you to virtually drive internal signals inside your FPGA to stimulate or control your design, like driving the RESET signal.