Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

3C7 DIGITAL SYSTEMS DESIGN LABORARTORY

Lab Session D Report

Department of Electronic and Electrical Engineering

Professor: Dr. Shreejith Shanker

1. Abstract:

This lab report documents the process of implementing designs on the BASYS 3 development board using Xilinx Vivado. The report comprises of two parts: the first part targeting the implementation of a Pulse Width Modulator design and the second part targets programming the LED's on Basys 3 board to output a pattern typical for a module indicator.

2. Introduction:

This lab session's objectives were to become acquainted with the Xilinx Vivado environment's implementation flow and understand the targeting of designs on the BASYS 3 development board. This report outlines the steps taken to achieve this goal and at the end the appendix includes the observations and results from lab C. This lab session's one of the aims was also to carefully analyse different reports that will be generated and to draw conclusions from them.

Submitted By:
Shreshtha Kamboj
23364317
Submission Date: 26/02/2024

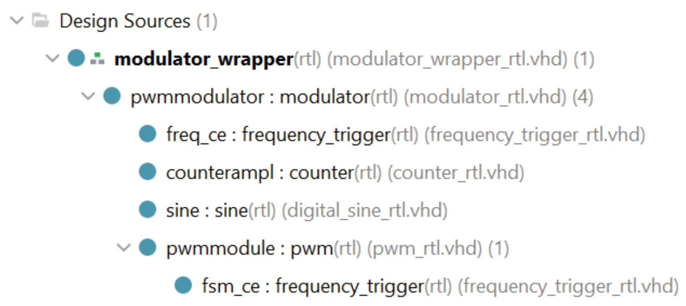
3. Implementation:

The implementation of Pulse Width Modulator design on Vivado for part 1 and implementation of a LED pattern onto the Basys 3 board.

4. Sources:

4.1 For part 1 the project created as modulator has the following components:

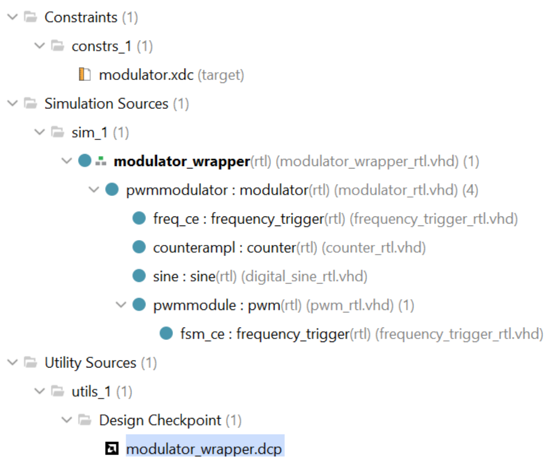
The Design sources consists of modulator_wrapper module which inherits the properties of other instantiations of modules shown below



```
Design Sources (1)
  modulator_wrapper(rtl) (modulator_wrapper_rtl.vhd) (1)
    pwmmodulator : modulator(rtl) (modulator_rtl.vhd) (4)
      freq_ce : frequency_trigger(rtl) (frequency_trigger_rtl.vhd)
      counterampl : counter(rtl) (counter_rtl.vhd)
      sine : sine(rtl) (digital_sine_rtl.vhd)
      pwmmodule : pwm(rtl) (pwm_rtl.vhd) (1)
        fsm_ce : frequency_trigger(rtl) (frequency_trigger_rtl.vhd)
```

Fig 4.1 Design Sources for Part-1

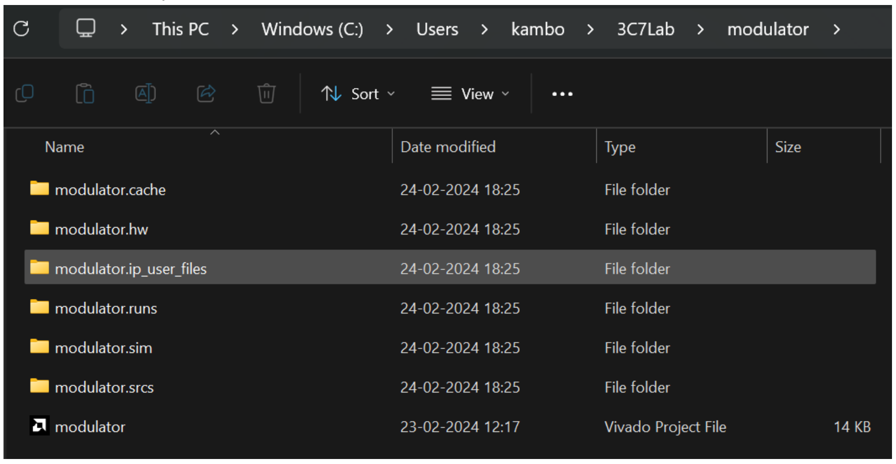
The constraints, Simulation sources, and utility sources are as shown below



```
Constraints (1)
  constrs_1 (1)
    modulator.xdc (target)
Simulation Sources (1)
  sim_1 (1)
    modulator_wrapper(rtl) (modulator_wrapper_rtl.vhd) (1)
      pwmmodulator : modulator(rtl) (modulator_rtl.vhd) (4)
        freq_ce : frequency_trigger(rtl) (frequency_trigger_rtl.vhd)
        counterampl : counter(rtl) (counter_rtl.vhd)
        sine : sine(rtl) (digital_sine_rtl.vhd)
        pwmmodule : pwm(rtl) (pwm_rtl.vhd) (1)
          fsm_ce : frequency_trigger(rtl) (frequency_trigger_rtl.vhd)
Utility Sources (1)
  utils_1 (1)
    Design Checkpoint (1)
      modulator_wrapper.dcp
```

Fig 4.2 Constraints, Simulation Sources, Utility Sources for Part-1

File directory is as follows

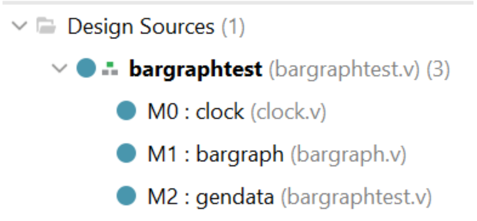

This PC > Windows (C:) > Users > kambo > 3C7Lab > modulator >
Sort
View

Name	Date modified	Type	Size
modulator.cache	24-02-2024 18:25	File folder	
modulator.hw	24-02-2024 18:25	File folder	
modulator.ip_user_files	24-02-2024 18:25	File folder	
modulator.runs	24-02-2024 18:25	File folder	
modulator.sim	24-02-2024 18:25	File folder	
modulator.srcs	24-02-2024 18:25	File folder	
modulator	23-02-2024 12:17	Vivado Project File	14 KB

Fig 4.3 File Directory for Part-1

4.2 For part 2 the project created as project_labDpart2 has the following components:

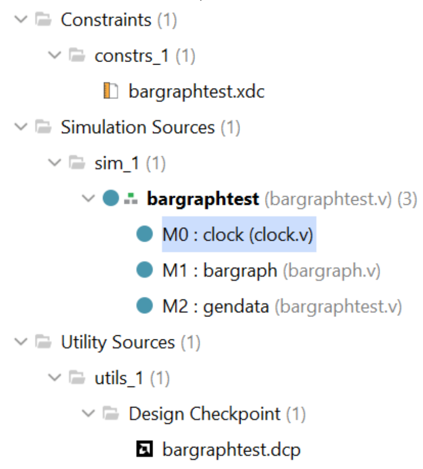
The Design sources consists of bargraphtest module which inherits the properties of other instantiations of modules shown below



```
Design Sources (1)
  bargraphtest (bargraphtest.v) (3)
    M0 : clock (clock.v)
    M1 : bargraph (bargraph.v)
    M2 : gendata (bargraphtest.v)
```

Fig 4.4 Design Sources for Part-2

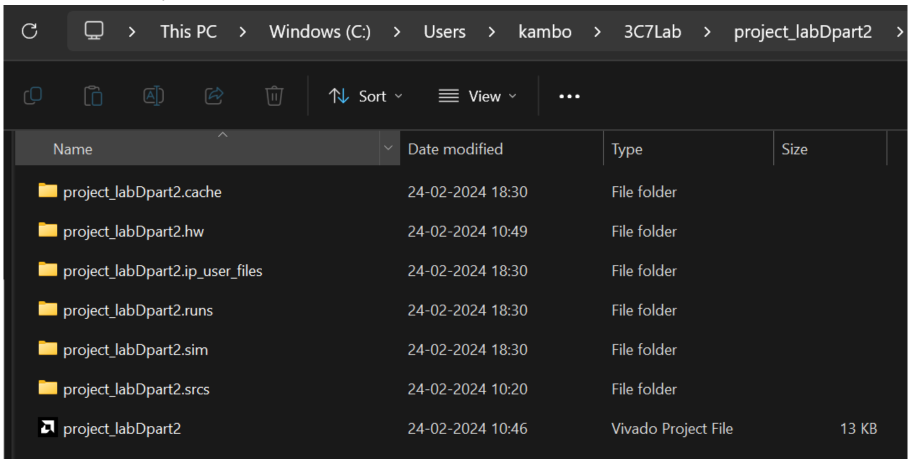
The constraints, Simulation sources, and utility sources are as shown below



```
Constraints (1)
  constrs_1 (1)
    bargraphtest.xdc
Simulation Sources (1)
  sim_1 (1)
    bargraphtest (bargraphtest.v) (3)
      M0 : clock (clock.v)
      M1 : bargraph (bargraph.v)
      M2 : gendata (bargraphtest.v)
Utility Sources (1)
  utils_1 (1)
    Design Checkpoint (1)
      bargraphtest.dcp
```

Fig 4.5 Constraints, Simulation Sources, Utility Sources for Part-2

File directory is as follows



This PC > Windows (C:) > Users > kambo > 3C7Lab > project_labDpart2 >

Name	Date modified	Type	Size
project_labDpart2.cache	24-02-2024 18:30	File folder	
project_labDpart2.hw	24-02-2024 10:49	File folder	
project_labDpart2.ip_user_files	24-02-2024 18:30	File folder	
project_labDpart2.runs	24-02-2024 18:30	File folder	
project_labDpart2.sim	24-02-2024 18:30	File folder	
project_labDpart2.srcs	24-02-2024 10:20	File folder	
project_labDpart2	24-02-2024 10:46	Vivado Project File	13 KB

Fig 4.6 File Directory for Part-2

5. Schematics Generated:

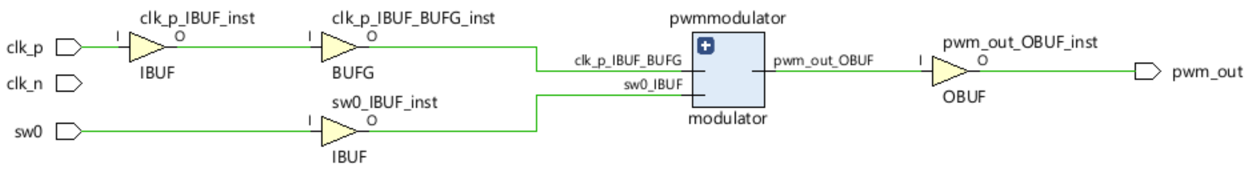


Fig 5.1 Elaborated Design for Part-1

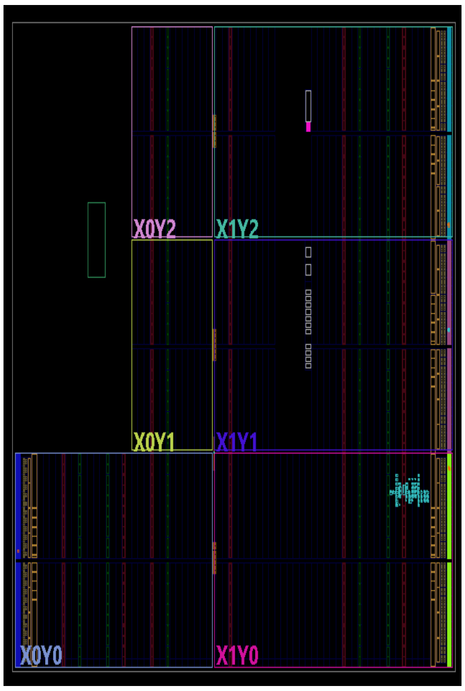


Fig 5.2 Implemented Device for Part-1

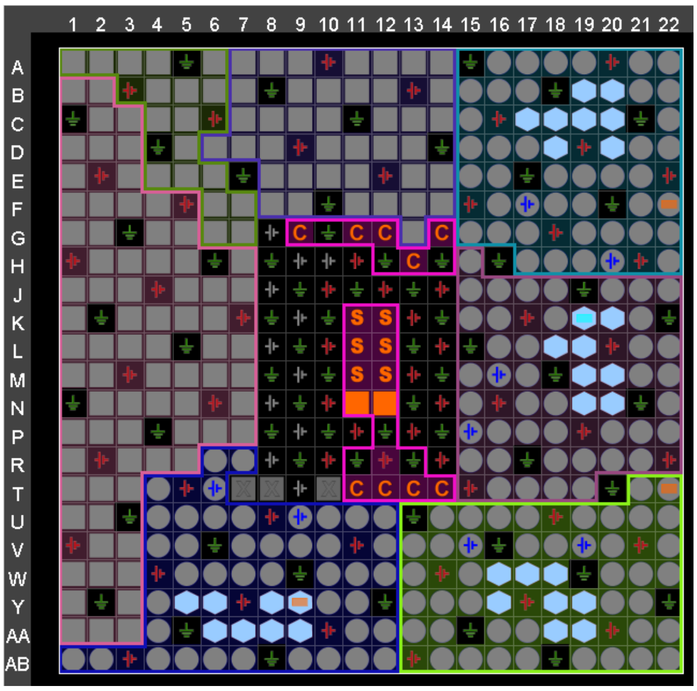


Fig 5.3 Implemented Package for Part-1

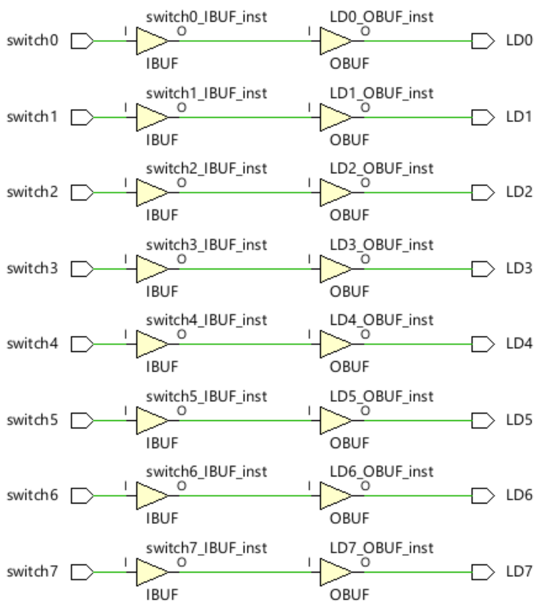


Fig 5.4 Elaborated Design for Part-2

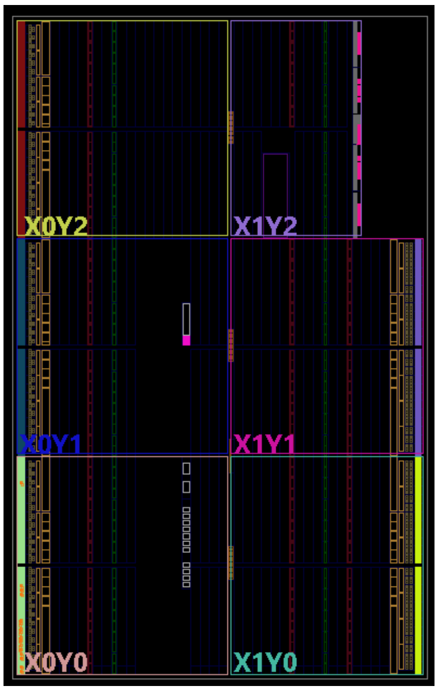


Fig 5.5 Implemented Device for Part-2

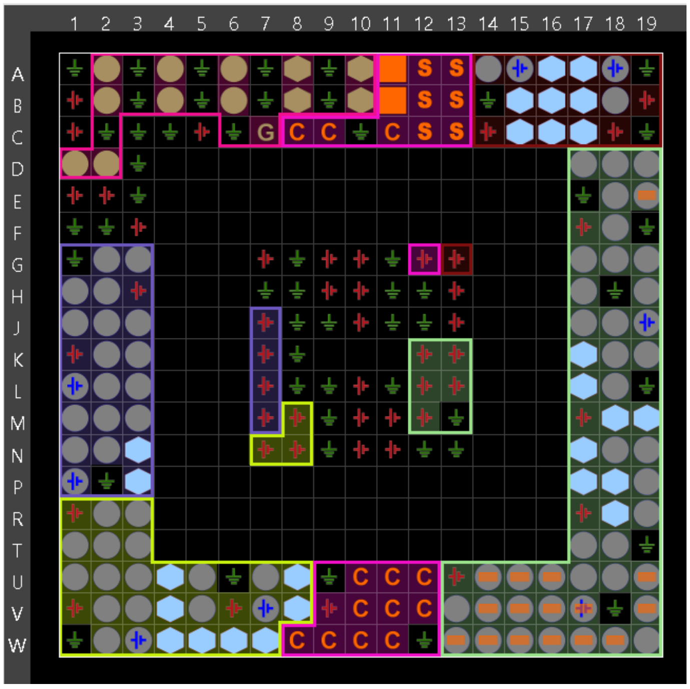
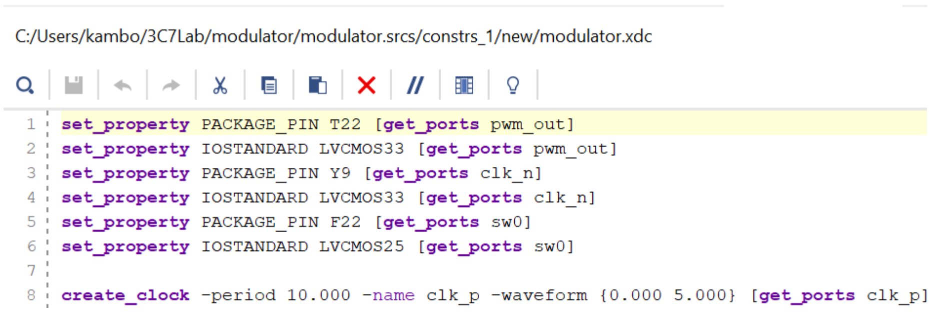


Fig 5.6 Implemented Package for Part-2

6. Code Snippets:

6.1 For Part-1:



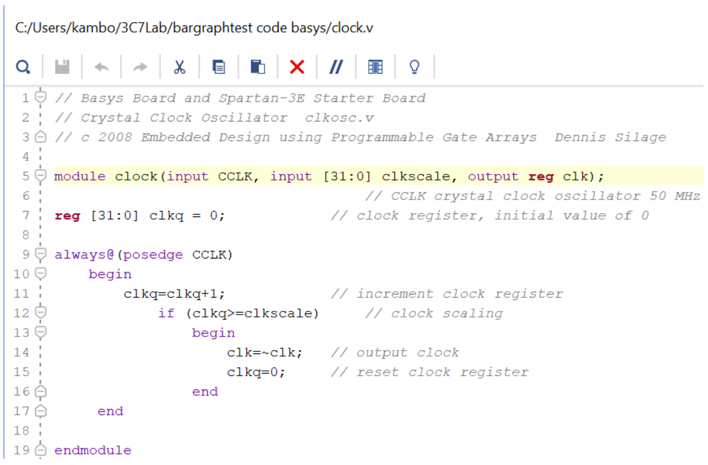
C:/Users/kambo/3C7Lab/modulator/modulator.srcs/constrs_1/new/modulator.xdc

```
set_property PACKAGE_PIN T22 [get_ports pwm_out]
set_property IOSTANDARD LVCMOS33 [get_ports pwm_out]
set_property PACKAGE_PIN Y9 [get_ports clk_n]
set_property IOSTANDARD LVCMOS33 [get_ports clk_n]
set_property PACKAGE_PIN F22 [get_ports sw0]
set_property IOSTANDARD LVCMOS25 [get_ports sw0]

create_clock -period 10.000 -name clk_p -waveform {0.000 5.000} [get_ports clk_p]
```

Fig 6.1 Modulator.xdc File with Physical Constraints

These lines of code provide correct functionality, timing, and compatibility with the destination FPGA device by defining pin assignments, I/O standards, and clock limitations for different ports and signals in the FPGA design.

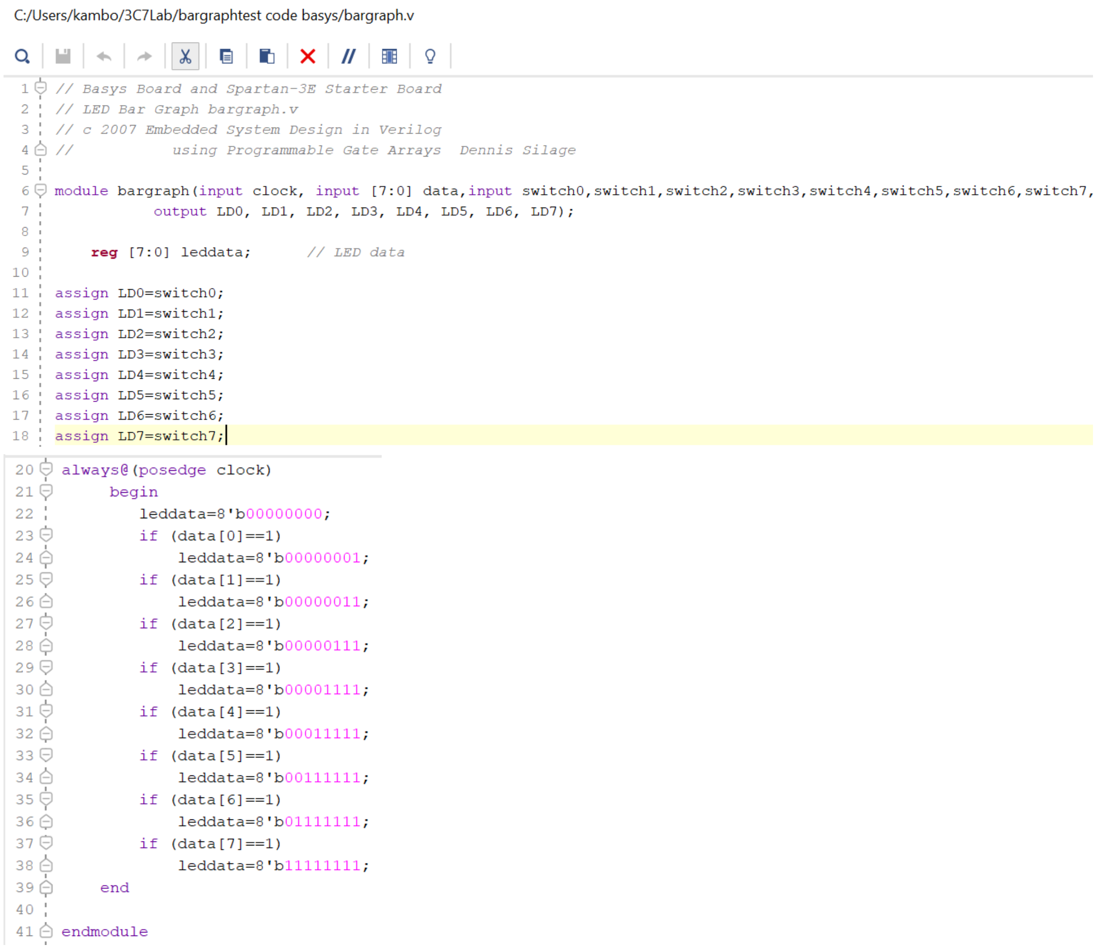
6.2 For Part-2:



C:/Users/kambo/3C7Lab/bargraphtest code basys/clock.v

```
// Basys Board and Spartan-3E Starter Board
// Crystal Clock Oscillator  clkosc.v
// c 2008 Embedded Design using Programmable Gate Arrays  Dennis Silage

module clock(input CCLK, input [31:0] clkscale, output reg clk);
                                    // CCLK crystal clock oscillator 50 MHz
reg [31:0] clkq = 0;            // clock register, initial value of 0

always@(posedge CCLK)
    begin
        clkq=clkq+1;            // increment clock register
            if (clkq>=clkscale)     // clock scaling
                begin
                    clk=~clk;   // output clock
                    clkq=0;     // reset clock register
                end
    end

endmodule
```

Fig 6.2 Clock Module for Generating Clock Signals

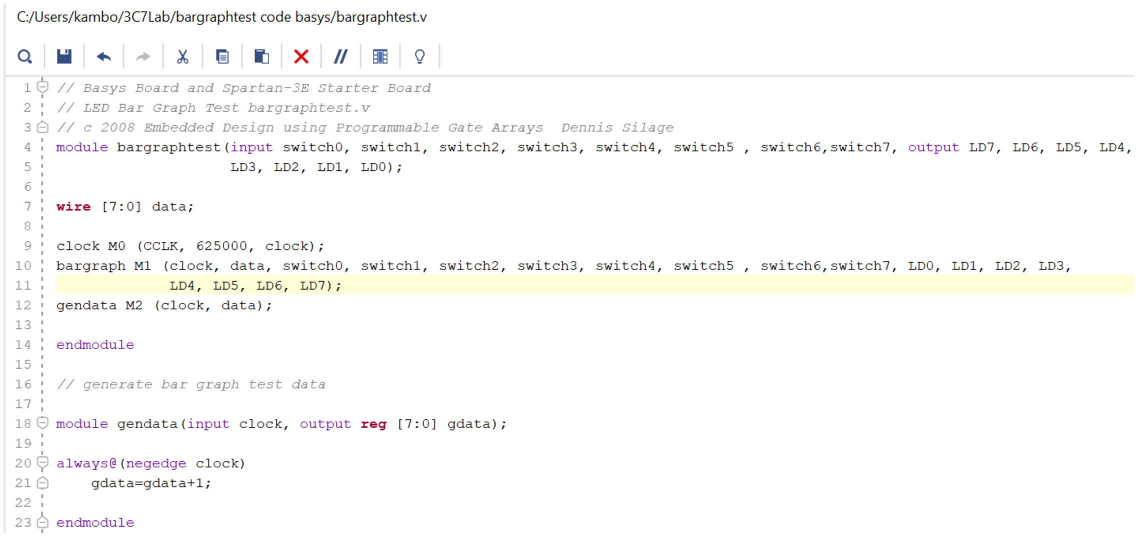


C:/Users/kambo/3C7Lab/bargraphtest code basys/bargraph.v

```verilog
// Basys Board and Spartan-3E Starter Board
// LED Bar Graph bargraph.v
// c 2007 Embedded System Design in Verilog
//           using Programmable Gate Arrays  Dennis Silage

module bargraph(input clock, input [7:0] data,input switch0,switch1,switch2,switch3,switch4,switch5,switch6,switch7,
           output LD0, LD1, LD2, LD3, LD4, LD5, LD6, LD7);

    reg [7:0] leddata;      // LED data

assign LD0=switch0;
assign LD1=switch1;
assign LD2=switch2;
assign LD3=switch3;
assign LD4=switch4;
assign LD5=switch5;
assign LD6=switch6;
assign LD7=switch7;
```

```verilog
always@(posedge clock)
     begin
        leddata=8'b00000000;
        if (data[0]==1)
            leddata=8'b00000001;
        if (data[1]==1)
            leddata=8'b00000011;
        if (data[2]==1)
            leddata=8'b00000111;
        if (data[3]==1)
            leddata=8'b00001111;
        if (data[4]==1)
            leddata=8'b00011111;
        if (data[5]==1)
            leddata=8'b00111111;
        if (data[6]==1)
            leddata=8'b01111111;
        if (data[7]==1)
            leddata=8'b11111111;
    end

endmodule
```

Fig 6.3 Module for assigning LED’s to switches on Basys 3 board

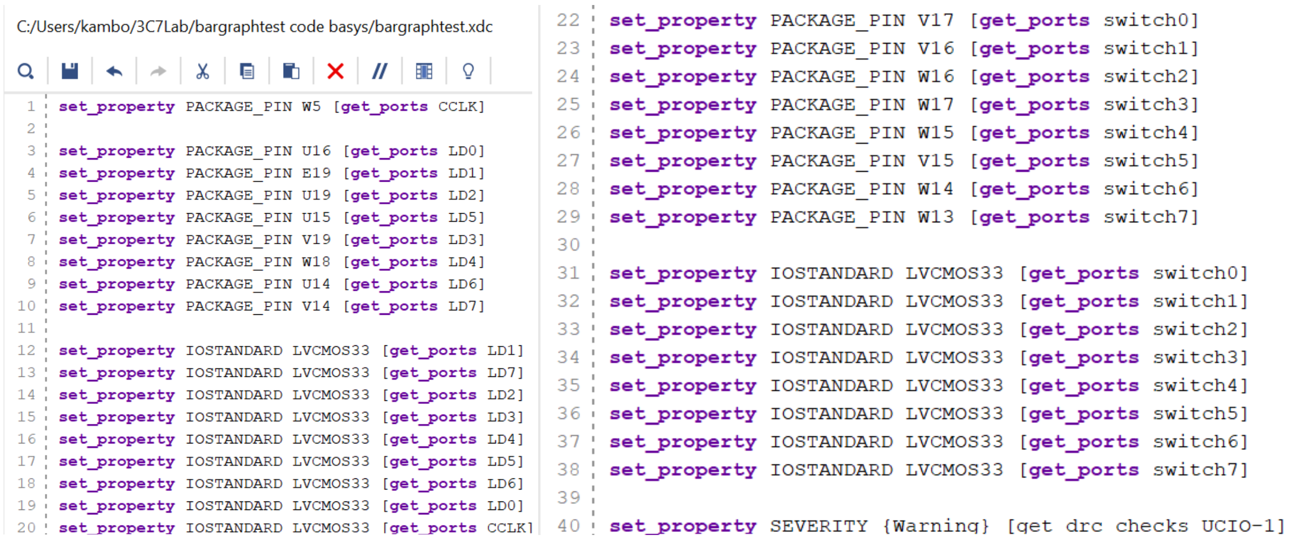


C:/Users/kambo/3C7Lab/bargraphtest code basys/bargraphtest.v

```verilog
// Basys Board and Spartan-3E Starter Board
// LED Bar Graph Test bargraphtest.v
// c 2008 Embedded Design using Programmable Gate Arrays  Dennis Silage
module bargraphtest(input switch0, switch1, switch2, switch3, switch4, switch5 , switch6,switch7, output LD7, LD6, LD5, LD4,
                    LD3, LD2, LD1, LD0);

wire [7:0] data;

clock M0 (CCLK, 625000, clock);
bargraph M1 (clock, data, switch0, switch1, switch2, switch3, switch4, switch5 , switch6,switch7, LD0, LD1, LD2, LD3,
             LD4, LD5, LD6, LD7);
gendata M2 (clock, data);

endmodule

// generate bar graph test data

module gendata(input clock, output reg [7:0] gdata);

always@(negedge clock)
    gdata=gdata+1;

endmodule
```

Fig 6.4 Module which has instantiation of modules

The following image shows the constraints for Basys 3 board implementation. The implementation will be explained in the demonstration section.



C:/Users/kambo/3C7Lab/bargraphtest code basys/bargraphtest.xdc

```
set_property PACKAGE_PIN W5 [get_ports CCLK]

set_property PACKAGE_PIN U16 [get_ports LD0]
set_property PACKAGE_PIN E19 [get_ports LD1]
set_property PACKAGE_PIN U19 [get_ports LD2]
set_property PACKAGE_PIN U15 [get_ports LD5]
set_property PACKAGE_PIN V19 [get_ports LD3]
set_property PACKAGE_PIN W18 [get_ports LD4]
set_property PACKAGE_PIN U14 [get_ports LD6]
set_property PACKAGE_PIN V14 [get_ports LD7]

set_property IOSTANDARD LVCMOS33 [get_ports LD1]
set_property IOSTANDARD LVCMOS33 [get_ports LD7]
set_property IOSTANDARD LVCMOS33 [get_ports LD2]
set_property IOSTANDARD LVCMOS33 [get_ports LD3]
set_property IOSTANDARD LVCMOS33 [get_ports LD4]
set_property IOSTANDARD LVCMOS33 [get_ports LD5]
set_property IOSTANDARD LVCMOS33 [get_ports LD6]
set_property IOSTANDARD LVCMOS33 [get_ports LD0]
set_property IOSTANDARD LVCMOS33 [get_ports CCLK]
set_property PACKAGE_PIN V17 [get_ports switch0]
set_property PACKAGE_PIN V16 [get_ports switch1]
set_property PACKAGE_PIN W16 [get_ports switch2]
set_property PACKAGE_PIN W17 [get_ports switch3]
set_property PACKAGE_PIN W15 [get_ports switch4]
set_property PACKAGE_PIN V15 [get_ports switch5]
set_property PACKAGE_PIN W14 [get_ports switch6]
set_property PACKAGE_PIN W13 [get_ports switch7]

set_property IOSTANDARD LVCMOS33 [get_ports switch0]
set_property IOSTANDARD LVCMOS33 [get_ports switch1]
set_property IOSTANDARD LVCMOS33 [get_ports switch2]
set_property IOSTANDARD LVCMOS33 [get_ports switch3]
set_property IOSTANDARD LVCMOS33 [get_ports switch4]
set_property IOSTANDARD LVCMOS33 [get_ports switch5]
set_property IOSTANDARD LVCMOS33 [get_ports switch6]
set_property IOSTANDARD LVCMOS33 [get_ports switch7]

set_property SEVERITY {Warning} [get_drc_checks UCIO-1]
```

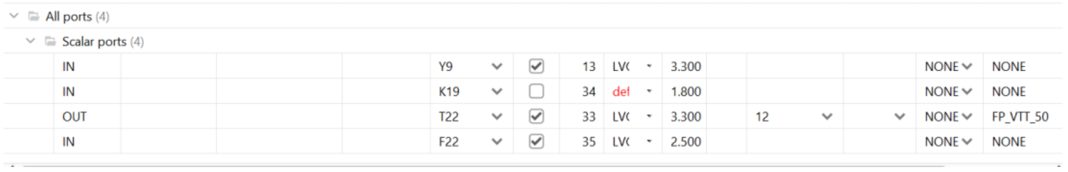
Fig 6.5 Constraints File for Basys 3 Board

7. Demonstration

7.1 For Part-1:

Configuration Device: ZedBoard (xc7z020clg484-1)
Target Language: Verilog

Initially the I/O Ports are not placed to specific pin location. First, we assign a pin location to each of the I/O Ports.



All ports (4)
Scalar ports (4)

IN	Y9	✓	13	LV(	3.300			NONE	NONE
IN	K19	☐	34	def	1.800			NONE	NONE
OUT	T22	✓	33	LV(	3.300	12		NONE	FP_VTT_50
IN	F22	✓	35	LV(	2.500			NONE	NONE

Fig 7.1 I/O Ports tab with assigned pin locations and I/O standards

When we save the above file then modulator.xdc file is automatically created. This file tells us about the Pin Assignments, Timing Constraints, Clock Constraints, I/O Standard & Voltage Constraints, Physical Constraints, User-defined Constraints.

Timing Constraints can be defined using two methods. The first approach is using Constraints Wizard and the second approach using the Constraints Editor. In this lab report we have used the former approach, i.e., using Constraints Wizard. In this we follow certain steps and encounter dialog boxes at each point described below:-

- Primary Clocks - Clock Sources with a missing clock definition. We specify 100MHz frequency and wizard automatically completes other parameters.
- Generated Clocks - Derived from primary clocks in FPGA fabric. In our design, the wizard determined that there are no unconstrained generated clocks.
- Forwarded Clocks - Generated clock on a primary output port of the FPGA. Commonly used for source synchronous buses when capture clock travels with data. The wizard has also determined that there are no unconstrained forwarded clocks in our design.
- External Feedback Delays - In the timing reports, clock delay compensation is calculated using either MMCM or PLL feedback delay external to the FPGA. The wizard did not find any unconstrained MMCM external feedback delay in our design.

- Input Delays - We do not need a delay period for this input port so we uncheck ***sw0*** input port.
- Output Delays - displays all the outputs that are unconstrained in the design. We do not need a delay period for this output port so we uncheck the pwm_out output port.
- Combinational Delays - Wizard looks for unconstrained combinational paths. A path that moves across the FPGA without getting caught by any sequential elements is called a combinational path. Our design does not contain any combinational paths.
- Physically Exclusive Clock Groups - Clocks that do not exist in the design at the same time. There are no unconstrained physically exclusive clock groups in our design.
- Logically Exclusive Clock Groups with Interaction - clocks that are running simultaneously, excluding areas of shared clock trees. There are no unconstrained logically exclusive clock groups with interaction in our design.
- Asynchronous Clock Domain Crossings - suggests limitations for the safe crossover of clock domains. Our design does not contain any unconstrained clock domain crossings.

After completing the above steps, the implementation was run and various reports were analysed.

7.2 For Part-2:

Configuration Device: BASYS 3 (xc7a35tcpg236-1)
Target Language: Verilog

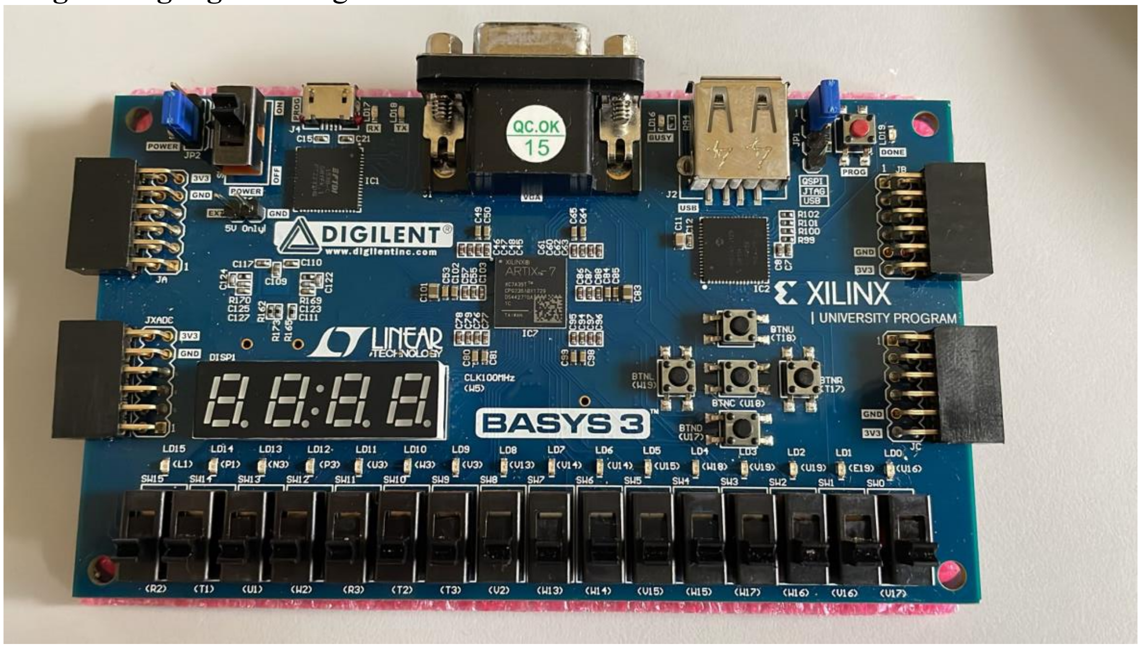


Fig 7.2 Basys 3 Board

SWITCHES - INPUT		LED - INPUT	
V17	SWITCH 0	U16	LED 0
V16	SWITCH 1	E19	LED 1
W16	SWITCH 2	U19	LED 2
W17	SWITCH 3	U15	LED 3
W15	SWITCH 4	V19	LED 4
V15	SWITCH 5	W18	LED 5
W14	SWITCH 6	U14	LED 6
W13	SWITCH 7	V14	LED 7

The input is taken using the switches assigned. State ‘0’ means OFF and state ‘1’ means ON. The output is generated using LEDs assigned. State ‘0’ means OFF and state ‘1’ means ON.

Test Case: Displaying Board Number in Binary on Basys 3 Board
Board Number = 44
Binary Equivalent = 8b’00101100

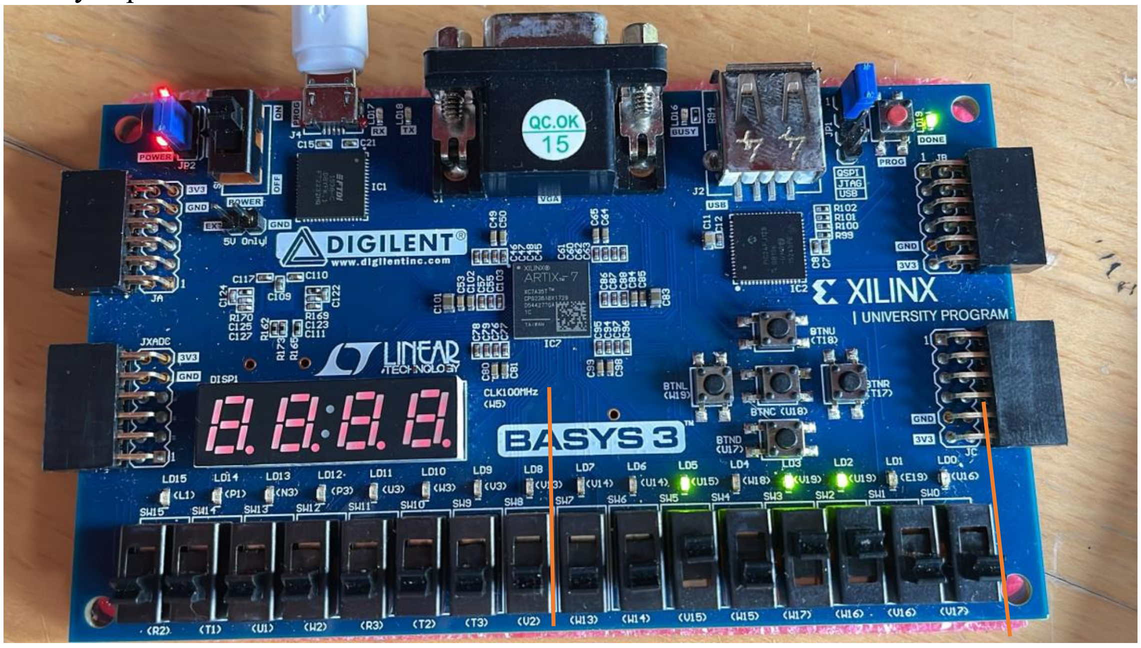
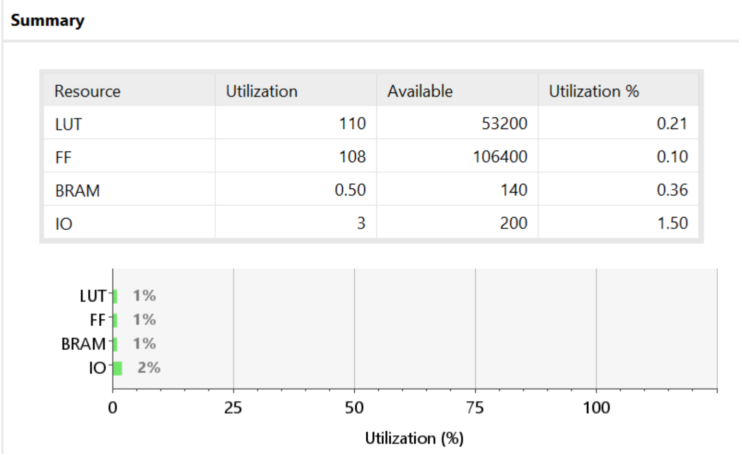


Fig 7.3 Board Number 44 Output Shown on Board using LEDs

8. Observations:

8.1 For Part-1: Report Analysis

Utilization Report, which is produced following the synthesis and implementation phases, provides information on how well our design is utilizing the FPGA's resources. Here we observe that IO resource is utilizing the most percentage of resources, i.e., 1.50% .

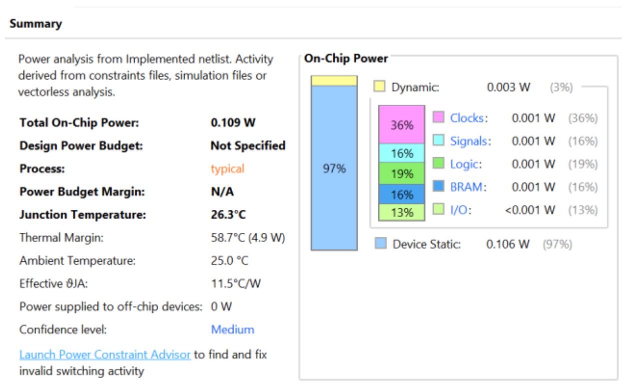


Summary

Resource	Utilization	Available	Utilization %
LUT	110	53200	0.21
FF	108	106400	0.10
BRAM	0.50	140	0.36
IO	3	200	1.50

Fig 8.1 Summary of Utilization Report

Power Consumption Report provides information about the target FPGA device's power consumption profile throughout the execution of our design. Here we observe that the total on-chip power is 0.109W and the Thermal margin is 58.7℃ which tells the designer reliability of their FPGA designs and making thermal management strategies to ensure safe operation of device.



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.109 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**26.3°C**
Thermal Margin:	58.7°C (4.9 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

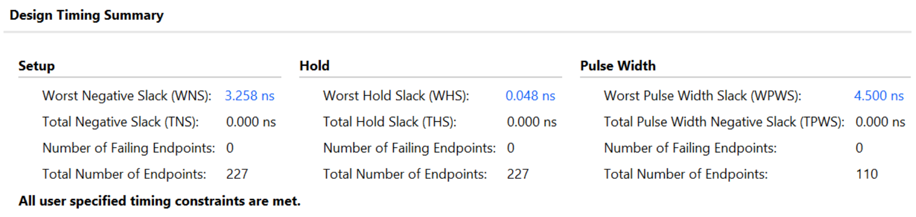
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.003 W	(3%)	
Clocks:	0.001 W	(36%)	36%
Signals:	0.001 W	(16%)	16%
Logic:	0.001 W	(19%)	19%
BRAM:	0.001 W	(16%)	16%
I/O:	<0.001 W	(13%)	13%
Device Static:	0.106 W	(97%)	97%

Fig 8.2 Summary of Power Report

Design Timing Report provides insights into our design's timing behaviour, including clock limitations, timing violations, and key routes. For a designer, this is the most crucial report because it is only via analysis of this report that he can determine whether he is fulfilling his timing goals.



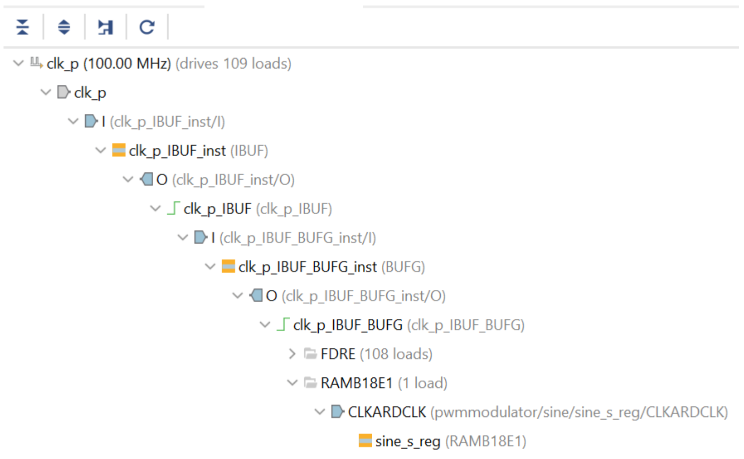
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.258 ns	Worst Hold Slack (WHS):	0.048 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	227	Total Number of Endpoints:	227	Total Number of Endpoints:	110

All user specified timing constraints are met.

Fig 8.3 Summary of Design Timing

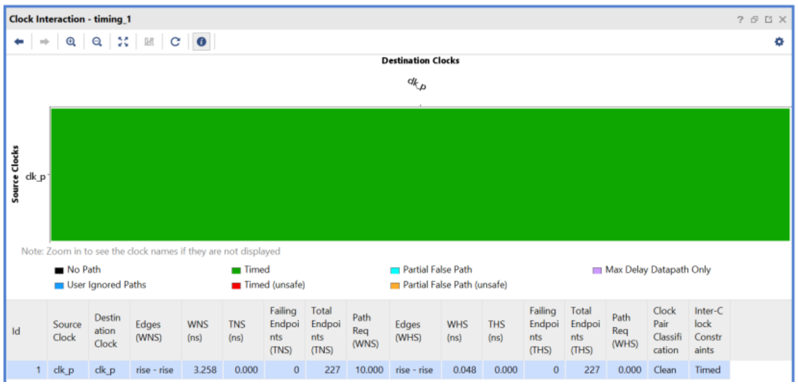
Clock Networks Report provides insightful information about our FPGA design's clocking architecture, assisting us in comprehending the device's clock signal distribution, routing, skew, and synchronization. This data is necessary to ensure dependable design operation, optimize design performance, and handle timing concerns.



```
clk_p (100.00 MHz) (drives 109 loads)
  clk_p
    I (clk_p_IBUF_inst/I)
      clk_p_IBUF_inst (IBUF)
        O (clk_p_IBUF_inst/O)
          clk_p_IBUF (clk_p_IBUF)
            I (clk_p_IBUF_BUFG_inst/I)
              clk_p_IBUF_BUFG_inst (BUFG)
                O (clk_p_IBUF_BUFG_inst/O)
                  clk_p_IBUF_BUFG (clk_p_IBUF_BUFG)
                    FDRE (108 loads)
                    RAMB18E1 (1 load)
                      CLKARDCLK (pwmmodulator/sine/sine_s_reg/CLKARDCLK)
                        sine_s_reg (RAMB18E1)
```

Fig 8.4 Summary of Clock Networks

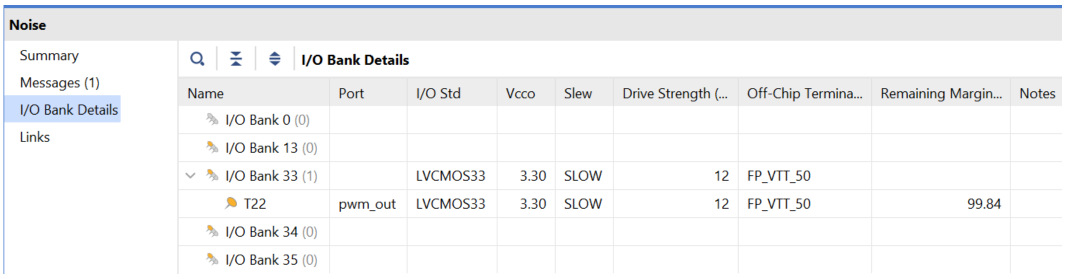
Clock interaction Report is a useful tool for comprehending the complexity of clock distribution and synchronization within our FPGA design, spotting possible timing problems associated with clock interactions, and putting strategies into place to guarantee correct synchronization and timing closure across various clock domains.


Clock Interaction - timing_1
Destination Clocks
clk_p
Source Clocks
clk_p
Note: Zoom in to see the clock names if they are not displayed
■ No Path
🟩 Timed
🟦 Partial False Path
🟪 Max Delay Datapath Only
🟦 User Ignored Paths
🟥 Timed (unsafe)
🟧 Partial False Path (unsafe)

Id	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Path Req (WNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	Path Req (WHS)	Clock Pair Classification	Inter-Clock Constraints
1	clk_p	clk_p	rise - rise	3.258	0.000	0	227	10.000	rise - rise	0.048	0.000	0	227	0.000	Clean	Timed

Fig 8.5 Summary of Clock Interaction

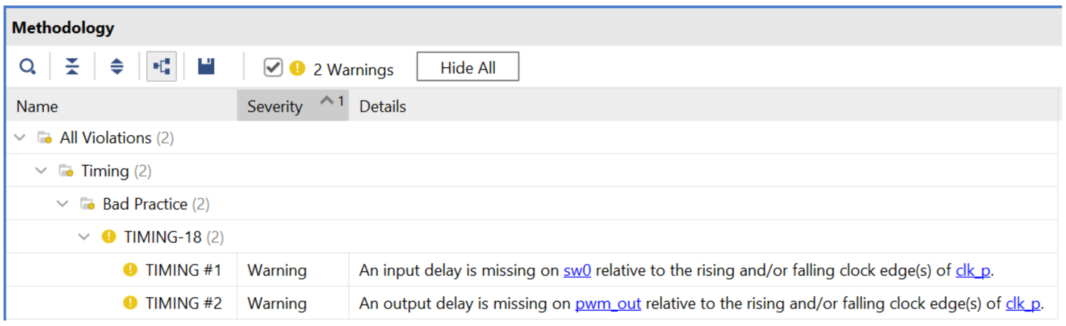
Noise Report is a useful tool for evaluating the noise properties of our FPGA design, spotting any noise-induced errors and signal integrity problems, and putting mitigation plans into practice to improve design performance and reliability.



Name	Port	I/O Std	Vcco	Slew	Drive Strength (...	Off-Chip Termina...	Remaining Margin...	Notes
I/O Bank 0 (0)								
I/O Bank 13 (0)								
I/O Bank 33 (1)		LVCMOS33	3.30	SLOW	12	FP_VTT_50		
T22	pwm_out	LVCMOS33	3.30	SLOW	12	FP_VTT_50	99.84	
I/O Bank 34 (0)								
I/O Bank 35 (0)								

Fig 8.6 Summary of Noise Report

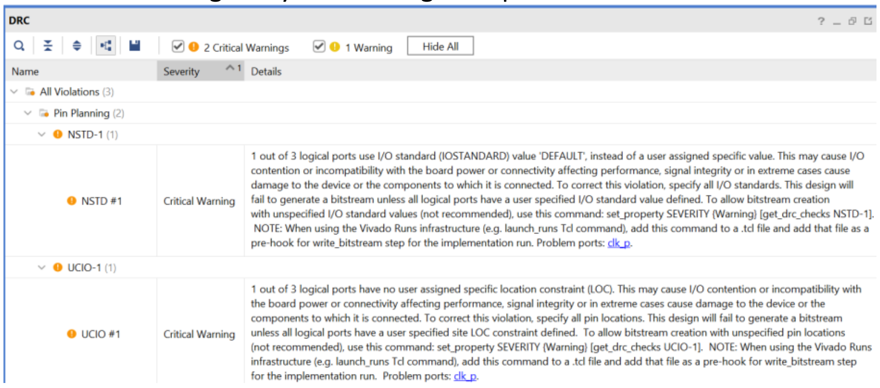
Methodology Report provides insights into the design process, constraints, optimizations, verification techniques, and best practices followed to achieve successful FPGA designs. In this case we are getting warnings because we unchecked these boxes when we were creating Time Constraints.



Name	Severity	Details
All Violations (2)		
Timing (2)		
Bad Practice (2)		
TIMING-18 (2)		
TIMING #1	Warning	An input delay is missing on sw0 relative to the rising and/or falling clock edge(s) of clk_p.
TIMING #2	Warning	An output delay is missing on pwm_out relative to the rising and/or falling clock edge(s) of clk_p.

Fig 8.7 Summary of Methodology Report

DRC Report contributes to the early detection and resolution of possible problems during the design phase, which enhances design quality and lowers the possibility of mistakes during final implementation. Here we see for our FPGA design we are being shown critical designs.


DRC

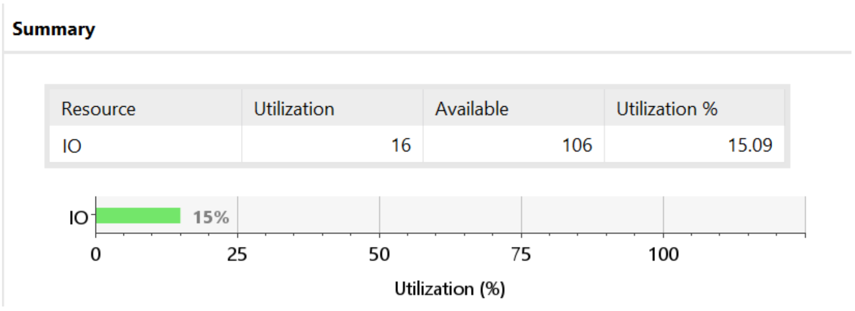
2 Critical Warnings 1 Warning Hide All

Name	Severity	Details
All Violations (3)		
Pin Planning (2)		
NSTD-1 (1)		
NSTD #1	Critical Warning	1 out of 3 logical ports use I/O standard (IOSTANDARD) value 'DEFAULT', instead of a user assigned specific value. This may cause I/O contention or incompatibility with the board power or connectivity affecting performance, signal integrity or in extreme cases cause damage to the device or the components to which it is connected. To correct this violation, specify all I/O standards. This design will fail to generate a bitstream unless all logical ports have a user specified I/O standard value defined. To allow bitstream creation with unspecified I/O standard values (not recommended), use this command: set_property SEVERITY {Warning} [get_drc_checks NSTD-1]. NOTE: When using the Vivado Runs infrastructure (e.g. launch_runs Tcl command), add this command to a .tcl file and add that file as a pre-hook for write_bitstream step for the implementation run. Problem ports: clk_p.
UCIO-1 (1)		
UCIO #1	Critical Warning	1 out of 3 logical ports have no user assigned specific location constraint (LOC). This may cause I/O contention or incompatibility with the board power or connectivity affecting performance, signal integrity or in extreme cases cause damage to the device or the components to which it is connected. To correct this violation, specify all pin locations. This design will fail to generate a bitstream unless all logical ports have a user specified site LOC constraint defined. To allow bitstream creation with unspecified pin locations (not recommended), use this command: set_property SEVERITY {Warning} [get_drc_checks UCIO-1]. NOTE: When using the Vivado Runs infrastructure (e.g. launch_runs Tcl command), add this command to a .tcl file and add that file as a pre-hook for write_bitstream step for the implementation run. Problem ports: clk_p.

Fig 8.8 Summary of DRC Report

8.2 For Part-2: Report Analysis

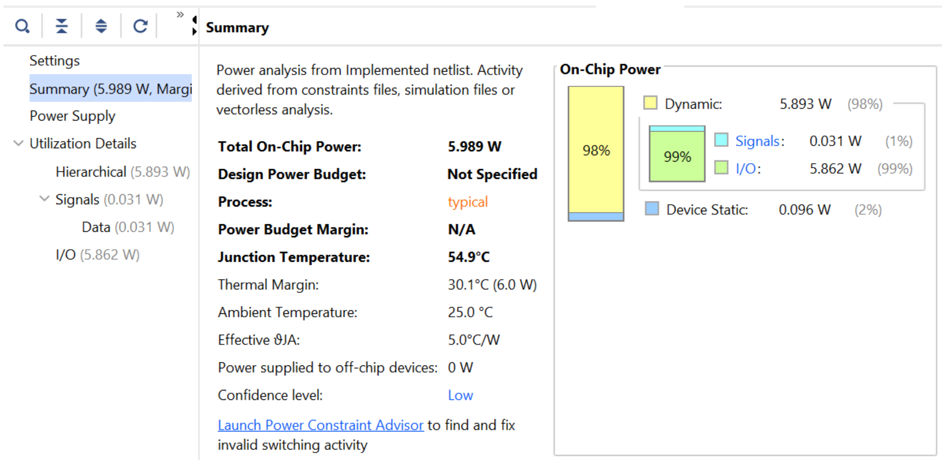
Utilization Report, which is produced following the synthesis and implementation phases, provides information on how well our design is utilizing the FPGA's resources. Here we observe that IO resource is utilizing the most percentage of resources, i.e., 15.09% .


Summary

Resource	Utilization	Available	Utilization %
IO	16	106	15.09

Fig 8.9 Summary of Utilization Report

Power Consumption Report provides information about the target FPGA device's power consumption profile throughout the execution of our design. Here we observe that the total on-chip power is 5.989W and the Thermal margin is 30.1℃ which tells the designer reliability of their FPGA designs and making thermal management strategies to ensure safe operation of device.



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**5.989 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**54.9°C**
Thermal Margin:	30.1°C (6.0 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

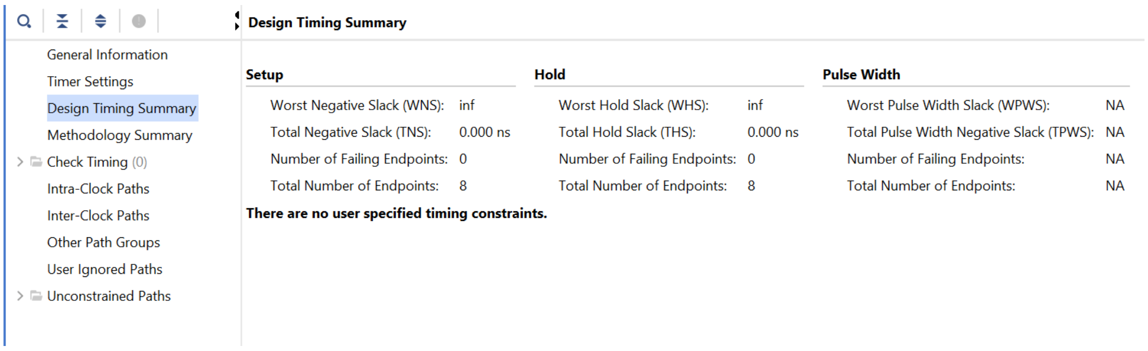
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

98%	Dynamic:	5.893 W	(98%)
99%	Signals:	0.031 W	(1%)
	I/O:	5.862 W	(99%)
	Device Static:	0.096 W	(2%)

Fig 8.10 Summary of Power Report

Design Timing Report provides insights into our design's timing behaviour, including clock limitations, timing violations, and key routes. In this case we did not specify any timing constraints and hence, we see such results.


Design Timing Summary

General Information
Timer Settings
Design Timing Summary
Methodology Summary
Check Timing (0)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	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:	8	Total Number of Endpoints:	8	Total Number of Endpoints:	NA

There are no user specified timing constraints.

Fig 8.11 Summary of Design Timing

9. Appendix:

9.1 LAB C Circuit:

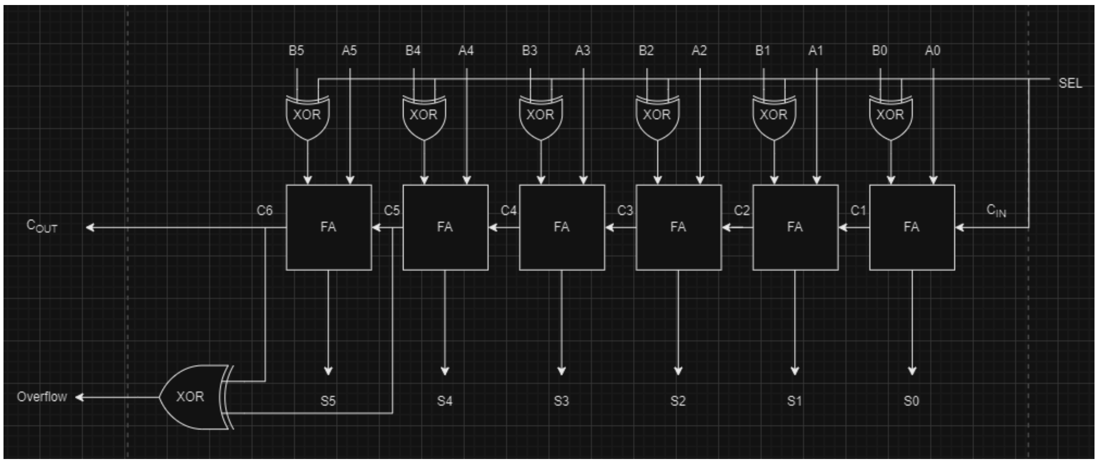


Fig 9.1 6-Bit Full Ripple Adder/Subtractor Diagram

In the above Circuit diagram, we have created a 6-Bit adder/subtractor which basically adds or subtracts two input binary numbers and gives us the result. Additionally, we are also checking the sum of each individual adder and the overflow of the 6-Bit adder/subtractor.

9.2 Elaborated Design:

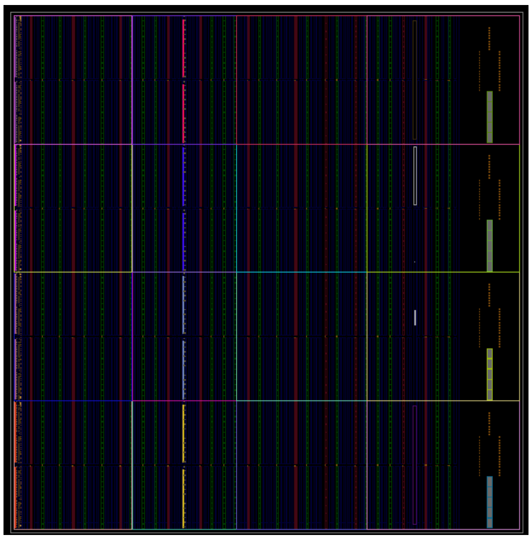


Fig 9.2 Implemented Device for Lab C

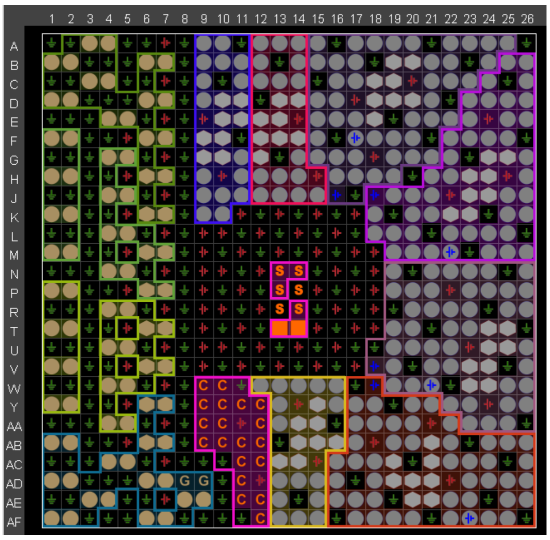
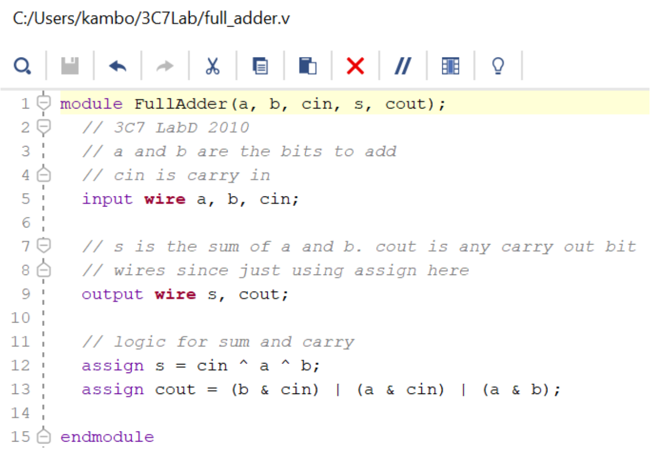


Fig 9.3 Implemented package for Lab C

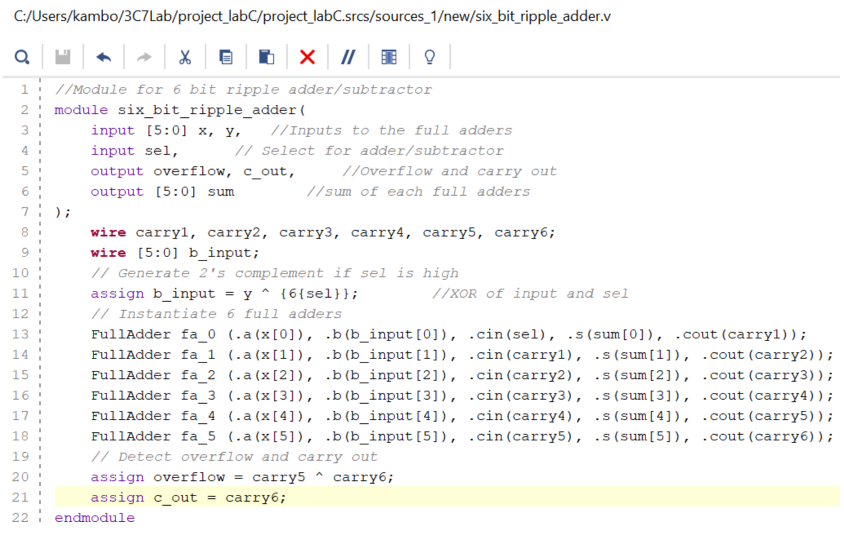
9.3 Code Snippets:



C:/Users/kambo/3C7Lab/full_adder.v

```
module FullAdder(a, b, cin, s, cout);
  // 3C7 LabD 2010
  // a and b are the bits to add
  // cin is carry in
  input wire a, b, cin;

  // s is the sum of a and b. cout is any carry out bit
  // wires since just using assign here
  output wire s, cout;

  // logic for sum and carry
  assign s = cin ^ a ^ b;
  assign cout = (b & cin) | (a & cin) | (a & b);

endmodule
```

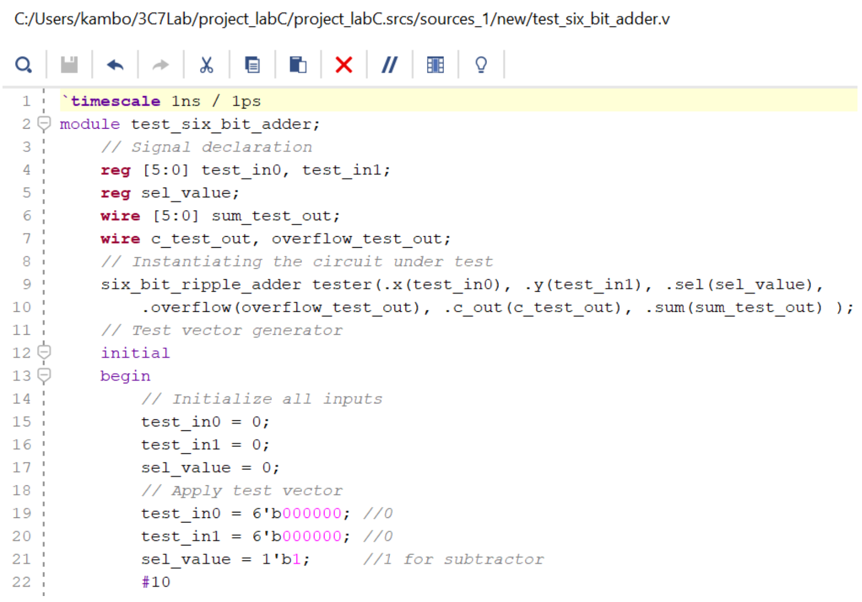
Fig 9.4 Module which has 1-Bit Full Adder/Subtractor



C:/Users/kambo/3C7Lab/project_labC/project_labC.srcs/sources_1/new/six_bit_ripple_adder.v

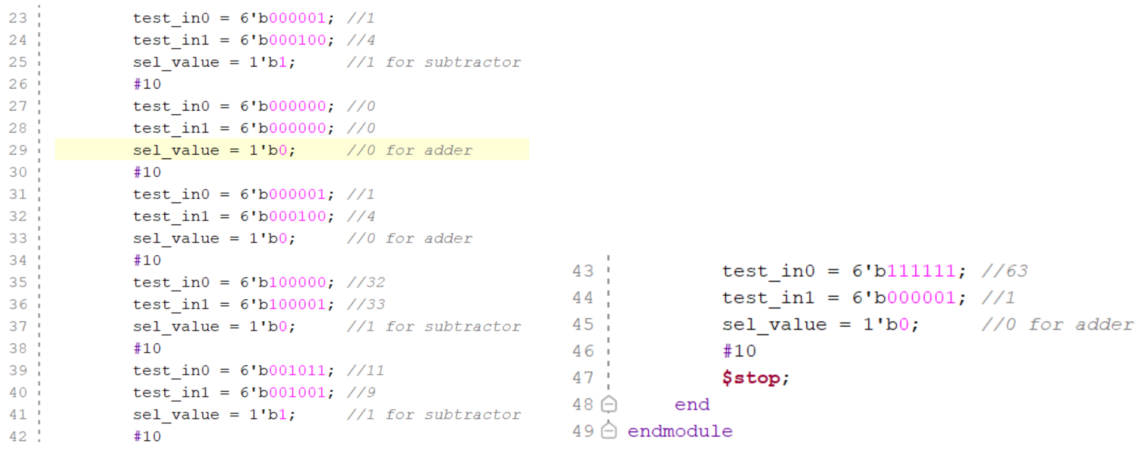
```
//Module for 6 bit ripple adder/subtractor
module six_bit_ripple_adder(
    input [5:0] x, y,   //Inputs to the full adders
    input sel,      // Select for adder/subtractor
    output overflow, c_out,     //Overflow and carry out
    output [5:0] sum        //sum of each full adders
);
    wire carry1, carry2, carry3, carry4, carry5, carry6;
    wire [5:0] b_input;
    // Generate 2's complement if sel is high
    assign b_input = y ^ {6{sel}};        //XOR of input and sel
    // Instantiate 6 full adders
    FullAdder fa_0 (.a(x[0]), .b(b_input[0]), .cin(sel), .s(sum[0]), .cout(carry1));
    FullAdder fa_1 (.a(x[1]), .b(b_input[1]), .cin(carry1), .s(sum[1]), .cout(carry2));
    FullAdder fa_2 (.a(x[2]), .b(b_input[2]), .cin(carry2), .s(sum[2]), .cout(carry3));
    FullAdder fa_3 (.a(x[3]), .b(b_input[3]), .cin(carry3), .s(sum[3]), .cout(carry4));
    FullAdder fa_4 (.a(x[4]), .b(b_input[4]), .cin(carry4), .s(sum[4]), .cout(carry5));
    FullAdder fa_5 (.a(x[5]), .b(b_input[5]), .cin(carry5), .s(sum[5]), .cout(carry6));
    // Detect overflow and carry out
    assign overflow = carry5 ^ carry6;
    assign c_out = carry6;
endmodule
```

Fig 9.5 Module which has 6 instantiations of Full Adder Module



C:/Users/kambo/3C7Lab/project_labC/project_labC.srcs/sources_1/new/test_six_bit_adder.v

```
`timescale 1ns / 1ps
module test_six_bit_adder;
    // Signal declaration
    reg [5:0] test_in0, test_in1;
    reg sel_value;
    wire [5:0] sum_test_out;
    wire c_test_out, overflow_test_out;
    // Instantiating the circuit under test
    six_bit_ripple_adder tester(.x(test_in0), .y(test_in1), .sel(sel_value),
        .overflow(overflow_test_out), .c_out(c_test_out), .sum(sum_test_out) );
    // Test vector generator
    initial
    begin
        // Initialize all inputs
        test_in0 = 0;
        test_in1 = 0;
        sel_value = 0;
        // Apply test vector
        test_in0 = 6'b000000; //0
        test_in1 = 6'b000000; //0
        sel_value = 1'b1;     //1 for subtractor
        #10
```



```
        test_in0 = 6'b000001; //1
        test_in1 = 6'b000100; //4
        sel_value = 1'b1;     //1 for subtractor
        #10
        test_in0 = 6'b000000; //0
        test_in1 = 6'b000000; //0
        sel_value = 1'b0;     //0 for adder
        #10
        test_in0 = 6'b000001; //1
        test_in1 = 6'b000100; //4
        sel_value = 1'b0;     //0 for adder
        #10
        test_in0 = 6'b100000; //32
        test_in1 = 6'b100001; //33
        sel_value = 1'b0;     //1 for subtractor
        #10
        test_in0 = 6'b001011; //11
        test_in1 = 6'b001001; //9
        sel_value = 1'b1;     //1 for subtractor
        #10
        test_in0 = 6'b111111; //63
        test_in1 = 6'b000001; //1
        sel_value = 1'b0;     //0 for adder
        #10
        $stop;
    end
endmodule
```

Fig 9.6 Testbench containing Test Vectors

9.4 Output Waveform:

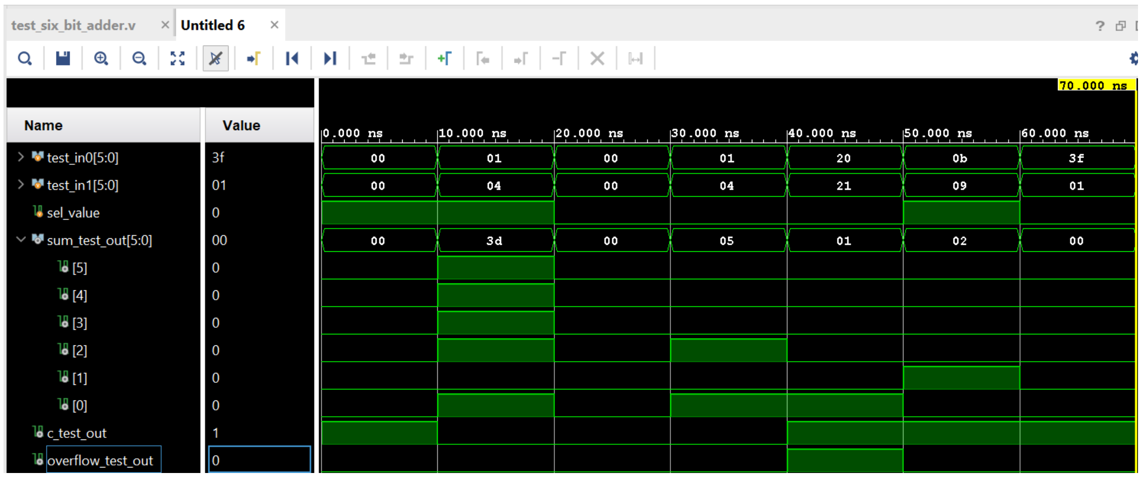


Fig 9.7 Waveform Indicating 7 test cases

9.5 Observation Table:

Test ID	x	y	SEL	C_{OUT} exp	Overflow exp	Sum exp	C_{out} obs	Overflow obs	Sum obs	Pass/Fail
1	6b’00_0000	6b’00_0000	1	1b’0	1b’0	6b’00_0000	1b’1	1b’0	6b’00_0000	FAIL
2	6b’00_0001	6b’00_0100	1	1b’0	1b’0	6b’11_1101	1b’0	1b’0	6b’11_1101	PASS
3	6b’00_0000	6b’00_0000	0	1b’0	1b’0	6b’00_0000	1b’0	1b’0	6b’00_0000	PASS
4	6b’00_0001	6b’00_0100	0	1b’0	1b’0	6b’00_0101	1b’0	1b’0	6b’00_0101	PASS
5	6b’10_0000	6b’10_0001	1	1b’0	1b’0	6b’11_1111	1b’1	1b’1	6b’00_0001	FAIL
6	6b’00_1011	6b’00_1001	1	1b’1	1b’0	6b’00_0010	1b’1	1b’0	6b’00_0010	PASS
7	6b’11_1111	6b’00_0001	0	1b’1	1b’0	6b’11_1110	1b’1	1b’0	6b’00_0000	FAIL

As we can see from the above table that few cases do not pass. For test case 1 we observe that a carryout is not expected but the waveform has it. For test case 5 we observe that neither of carryout, overflow, and sum are same as observed. For test case 7 the expected sum is not what is observed in the waveform.

10. Conclusion:

In Lab C we learnt how to implement a 6-bit Full Ripple Adder/Subtractor and tested it for various cases using the test vectors defined in testbench file. We observed that not all test cases matched observed and expected outputs.

In Lab D Part-1 we learnt how to implement a Pulse Width Modulator on Vivado and how to define time constraints and create a xdc (Xilinx Design Constraint) file. We also analysed different reports such as Utilization Report and understood what that reports tell us about our FPGA design.

In Lab D Part-2 we learnt how to use Basys 3 Board. We familiarized with switches and LEDs on the board. Then we created an output pattern for displaying the binary equivalent of our board number using the LEDs. We also analysed few reports and learnt about resource utilization and power consumption.