Simulation Exercise 5.5
4-Bit Adder/Subtractor with Seven Segment Display
ELEE 2640

Benyamain Yacoob

March 2, 2024

Date Performed:	March 5, 2024
Partners:	Ara Oladipo
	Andre Price
Instructor:	Professor Paulik

UNIVERSITY OF DETROIT MERCY

Contents

1 Objectives

First Objective

Code in SystemVerilog for combinational logic design and testing.

Second Objective

Simulate and implement combinational circuits using Xilinx Vivado integrated development environment (IDE).

Third Objective

Complete adder/subtractor work with a seven segment result display.

2 Problem Statement

In this simulation exercise, you will follow the work you did in Simulation Exercise #5 by adding a Seven Segment display subsystem similar to that developed in Simulation Exercise #3, but modified to display the full range of hexadecimal digits. You will be using the SystemVerilog hardware description language (HDL) and the Xilinx ISE(introduced in Simulation Exercise #4) to implement your designs.

3 Materials

Xilinx integrated synthesis environment (ISE)

4 Requirements

Schematic

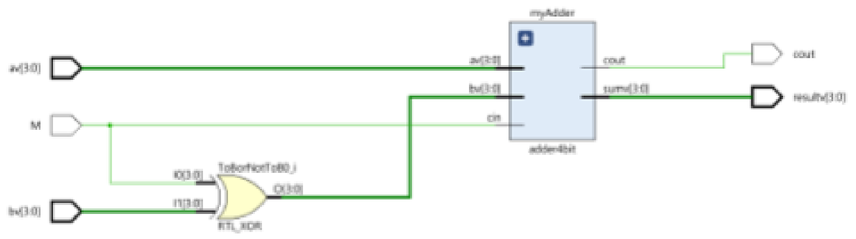


Figure 1: RTL Schematic for 4-Bit Adder/Subtractor

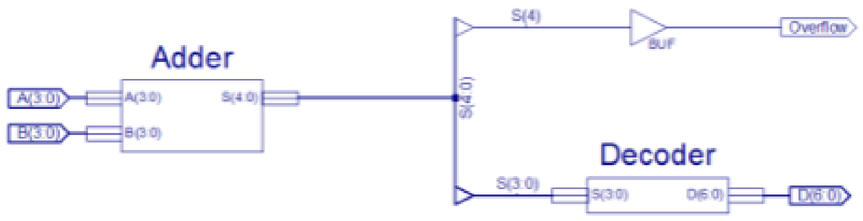


Figure 2: Top-Level Schematic for 4-Bit Adder with Decoder

Seven Segment Display Format

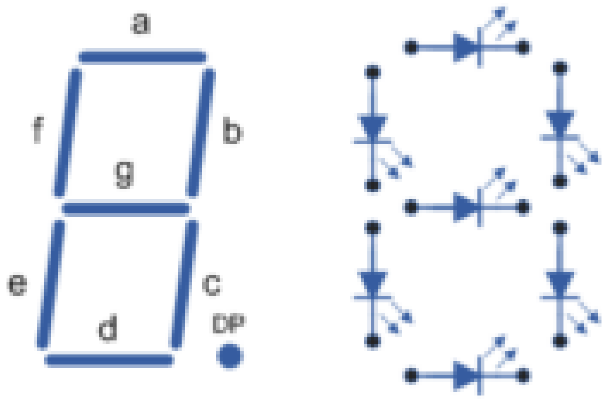


Figure 3: Seven Segment Display Segments and Diodes

HEX Data Display on Seven Segment Device Format

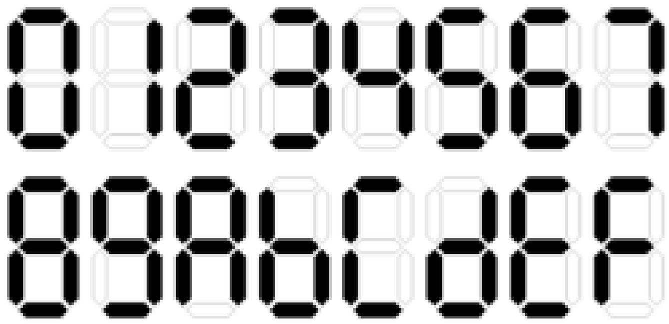


Figure 4: HEX Data Display on Seven Segment Device

IO Devices

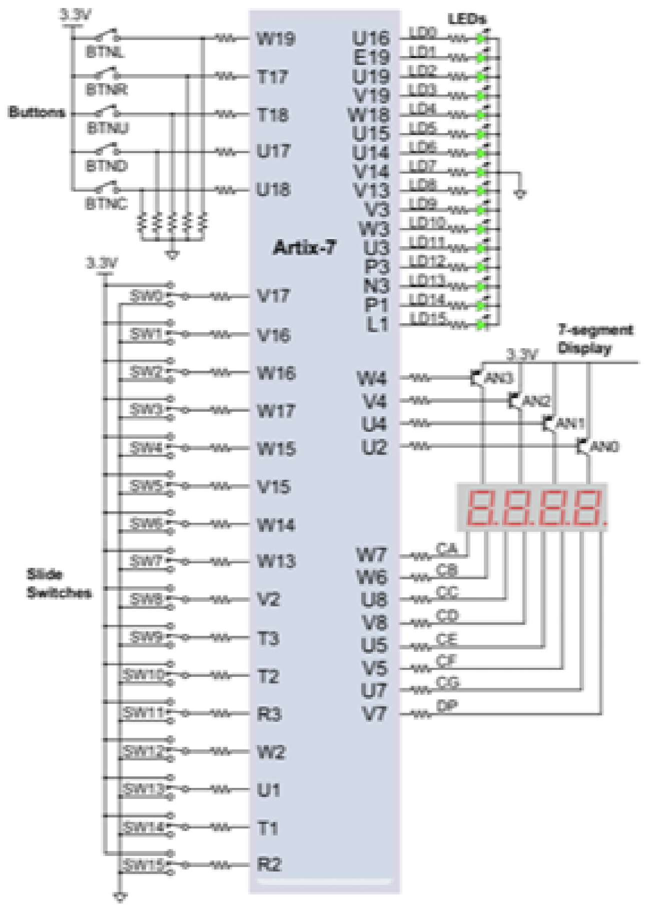
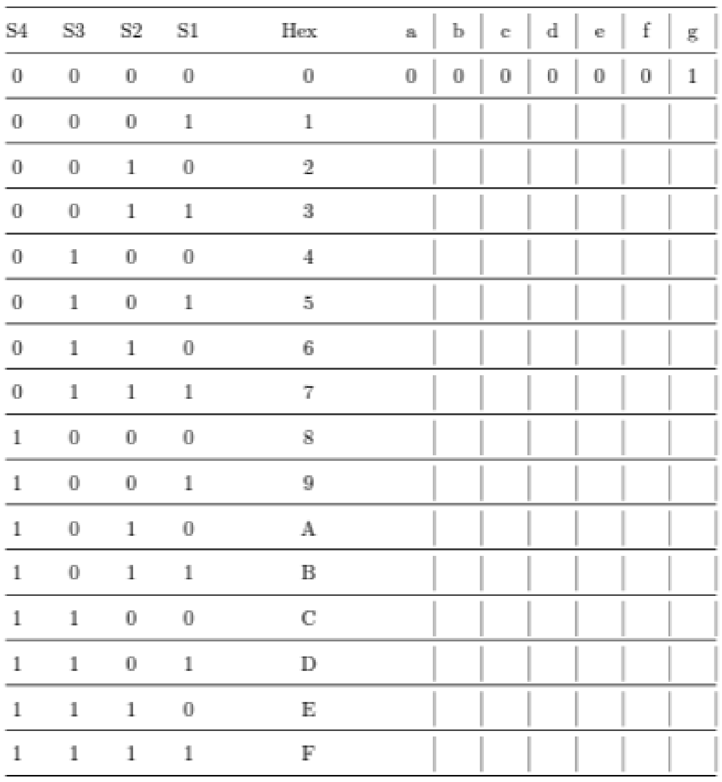


Figure 16. General Purpose I/O devices on the Basys3

Figure 5: General Purpose IO Devices on the Basys3

Truth Table

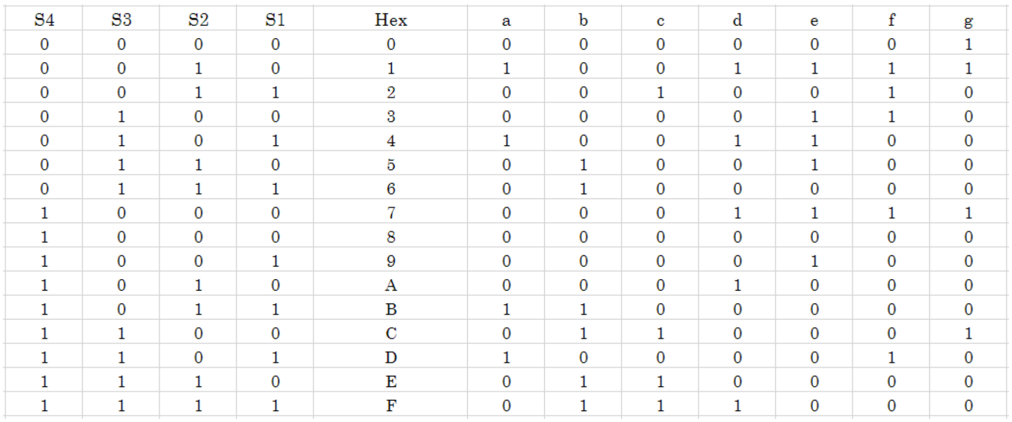


S4	S3	S2	S1	Hex	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1							
0	0	1	0	2							
0	0	1	1	3							
0	1	0	0	4							
0	1	0	1	5							
0	1	1	0	6							
0	1	1	1	7							
1	0	0	0	8							
1	0	0	1	9							
1	0	1	0	A							
1	0	1	1	B							
1	1	0	0	C							
1	1	0	1	D							
1	1	1	0	E							
1	1	1	1	F							

Figure 6: Truth Table for Seven Segment Display Decoder

5 Seven Segment Display Decoder

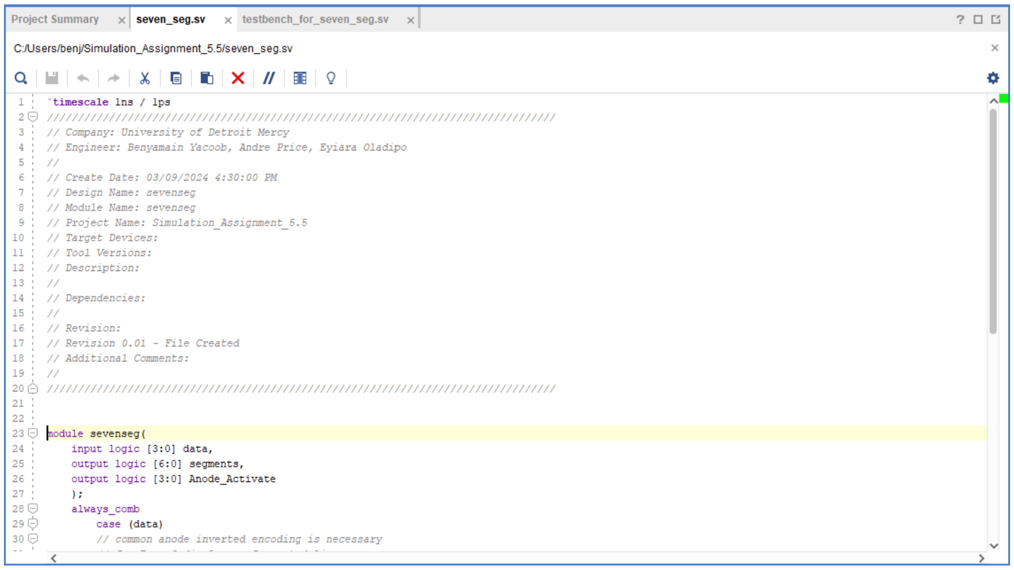
5.1 Seven Segment Display Decoder Truth Table



S4	S3	S2	S1	Hex	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	1	1	0	0	1	1	1	1
0	0	1	1	2	0	0	1	0	0	1	0
0	1	0	0	3	0	0	0	0	1	1	0
0	1	0	1	4	1	0	0	1	1	0	0
0	1	1	0	5	0	1	0	0	1	0	0
0	1	1	1	6	0	1	0	0	0	0	0
1	0	0	0	7	0	0	0	1	1	1	1
1	0	0	0	8	0	0	0	0	0	0	0
1	0	0	1	9	0	0	0	0	1	0	0
1	0	1	0	A	0	0	0	1	0	0	0
1	0	1	1	B	1	1	0	0	0	0	0
1	1	0	0	C	0	1	1	0	0	0	1
1	1	0	1	D	1	0	0	0	0	1	0
1	1	1	0	E	0	1	1	0	0	0	0
1	1	1	1	F	0	1	1	1	0	0	0

Figure 7: Truth Table for Seven Segment Display Decoder

5.2 Seven Segment Display Decoder Implementation Code



C:/Users/benj/Simulation_Assignment_5.5/seven_seg.sv

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company: University of Detroit Mercy
// Engineer: Benyamain Yacoob, Andre Price, Eyiara Oladipo
//
// Create Date: 03/09/2024 4:30:00 PM
// Design Name: sevenseg
// Module Name: sevenseg
// Project Name: Simulation_Assignment_5.5
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////


module sevenseg(
    input logic [3:0] data,
    output logic [6:0] segments,
    output logic [3:0] Anode_Activate
    );
    always_comb
        case (data)
        // common anode inverted encoding is necessary
```

Figure 8: Seven Segment Display Decoder Implementation Code

5.3 Seven Segment Display Elaborated Design Schematic

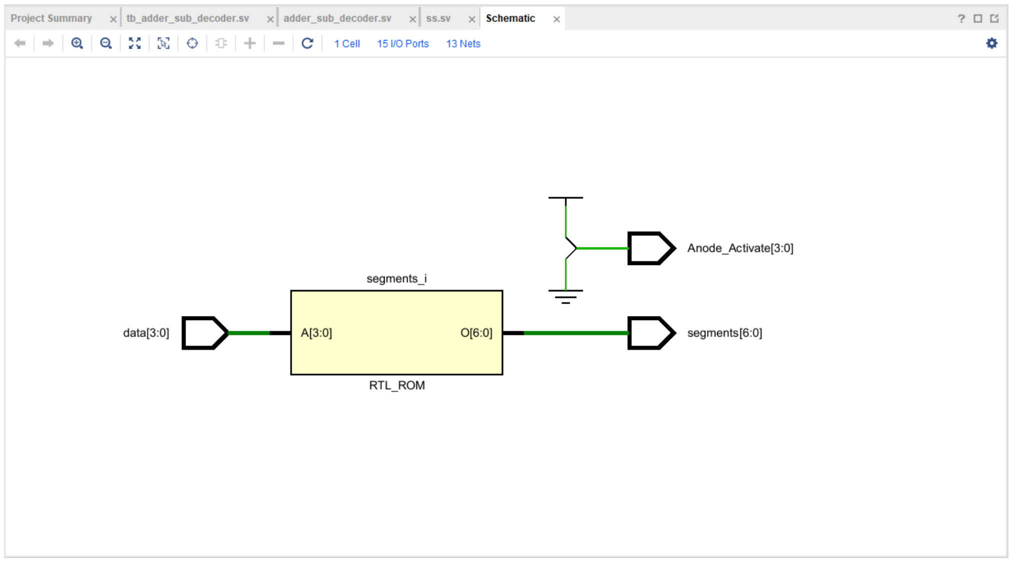


Figure 9: Seven Segment Display Elaborated Design Schematic

5.4 Seven Segment Display Decoder Elaborated Design Schematic

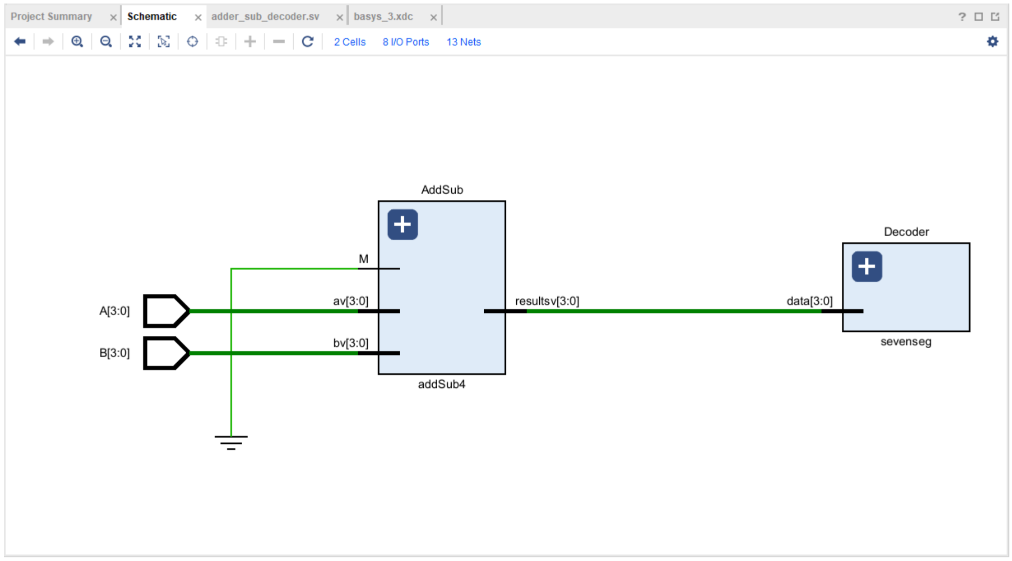


Figure 10: Seven Segment Display Decoder Elaborated Design Schematic

5.5 Seven Segment Synthesized Design

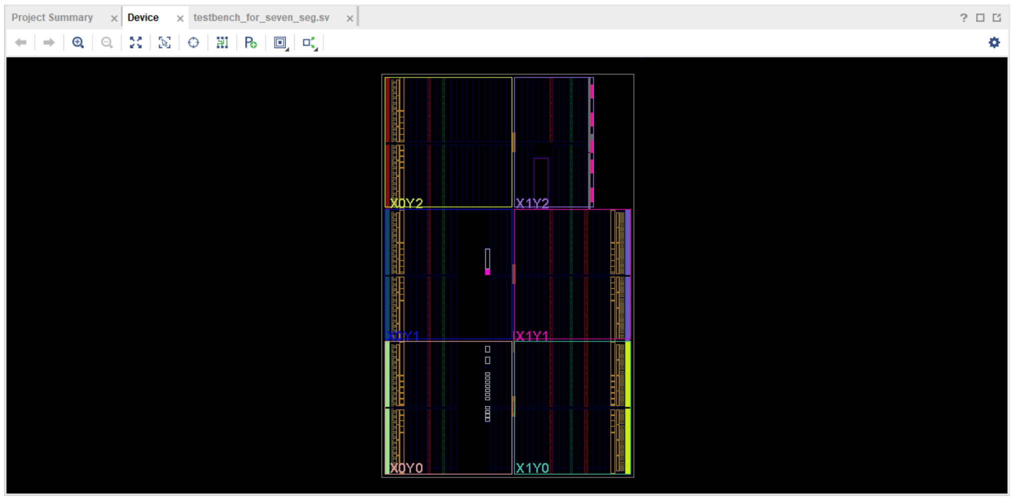


Figure 11: Seven Segment Display Decoder Synthesized Design

5.6 Seven Segments Synthesized Schematic

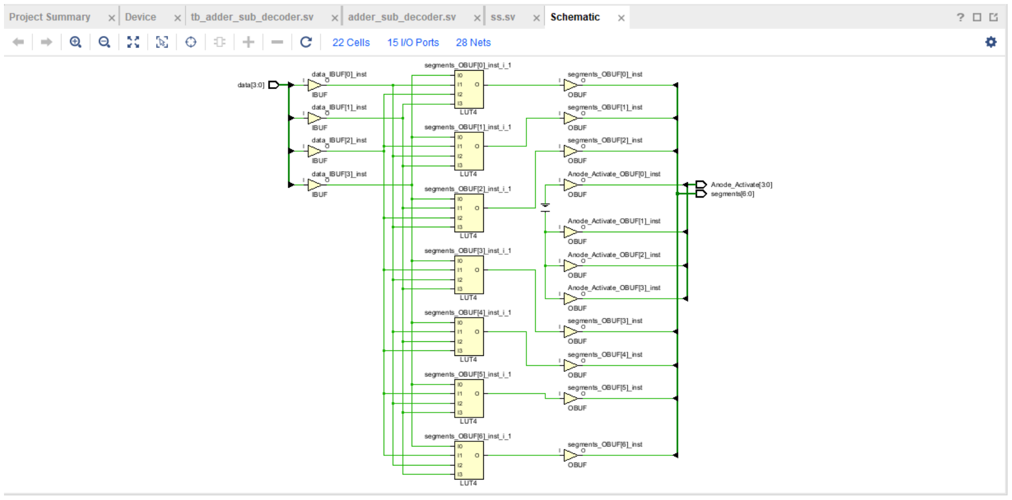
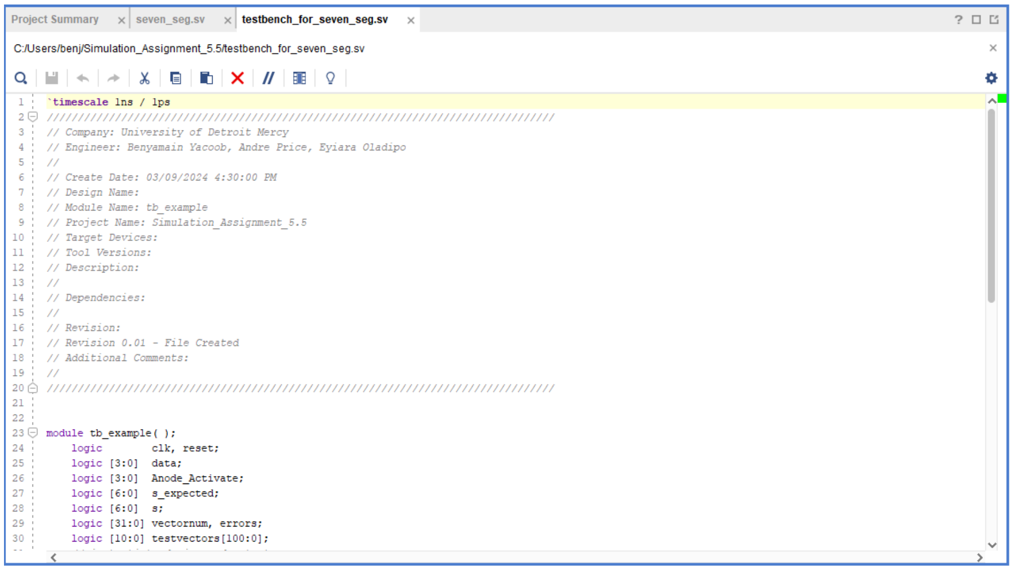


Figure 12: Seven Segment Display Decoder Synthesized Schematic

5.7 Seven Segment Display Test Bench

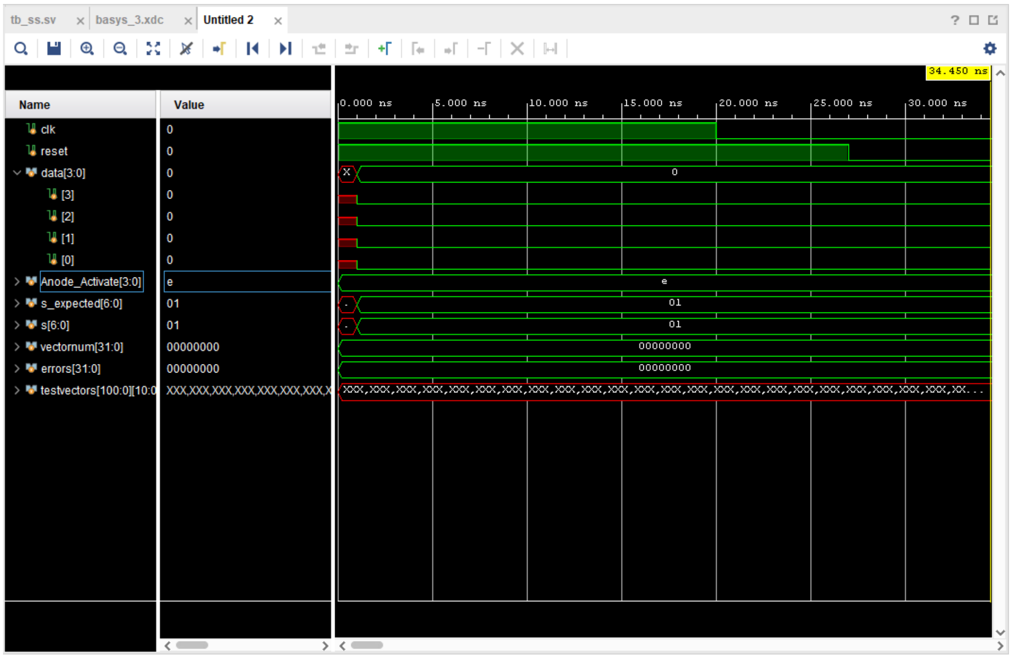
5.7.1 Seven Segment Display Decoder Test Bench Implementation Code


Project Summary
seven_seg.sv
testbench_for_seven_seg.sv
C:/Users/benj/Simulation_Assignment_5.5/testbench_for_seven_seg.sv

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: University of Detroit Mercy
// Engineer: Benyamain Yacoob, Andre Price, Eyiara Oladipo
//
// Create Date: 03/09/2024 4:30:00 PM
// Design Name:
// Module Name: tb_example
// Project Name: Simulation_Assignment_5.5
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module tb_example( );
    logic        clk, reset;
    logic [3:0]  data;
    logic [3:0]  Anode_Activate;
    logic [6:0]  s_expected;
    logic [6:0]  s;
    logic [31:0] vectornum, errors;
    logic [10:0] testvectors[100:0];
```

Figure 13: Test Bench Implementation Code

5.7.2 Seven Segment Display Decoder Test Bench Behavioral Simulation



Name	Value
clk	0
reset	0
data[3:0]	0
[3]	0
[2]	0
[1]	0
[0]	0
Anode_Activate[3:0]	e
s_expected[6:0]	01
s[6:0]	01
vectornum[31:0]	00000000
errors[31:0]	00000000
testvectors[100:0][10:0]	XXX,XXX,XXX,XXX,XXX,XXX,XXX,X

Figure 14: Test Bench Behavioral Simulation

5.7.3 Seven Segment Display Decoder Test Bench Schematic

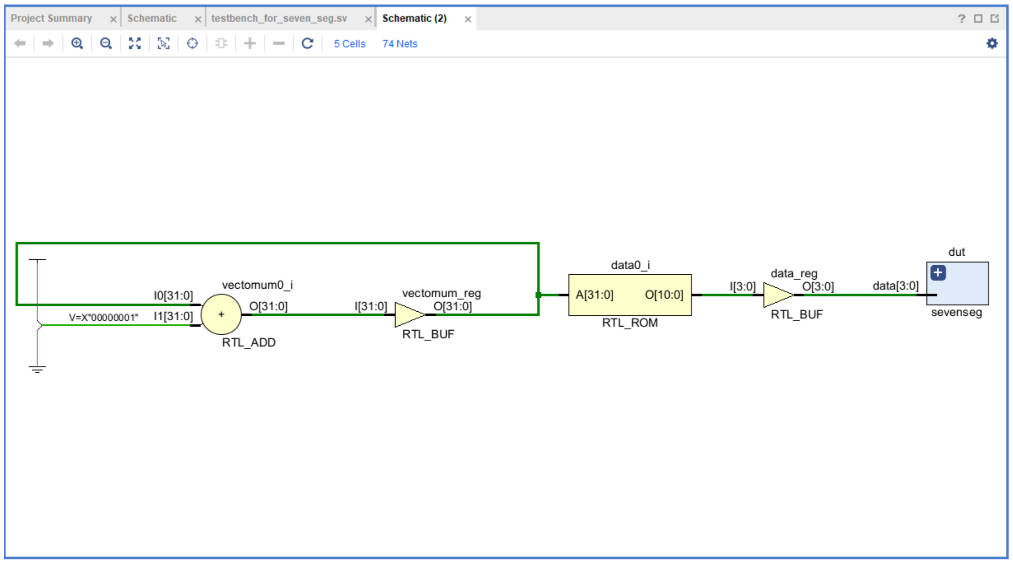
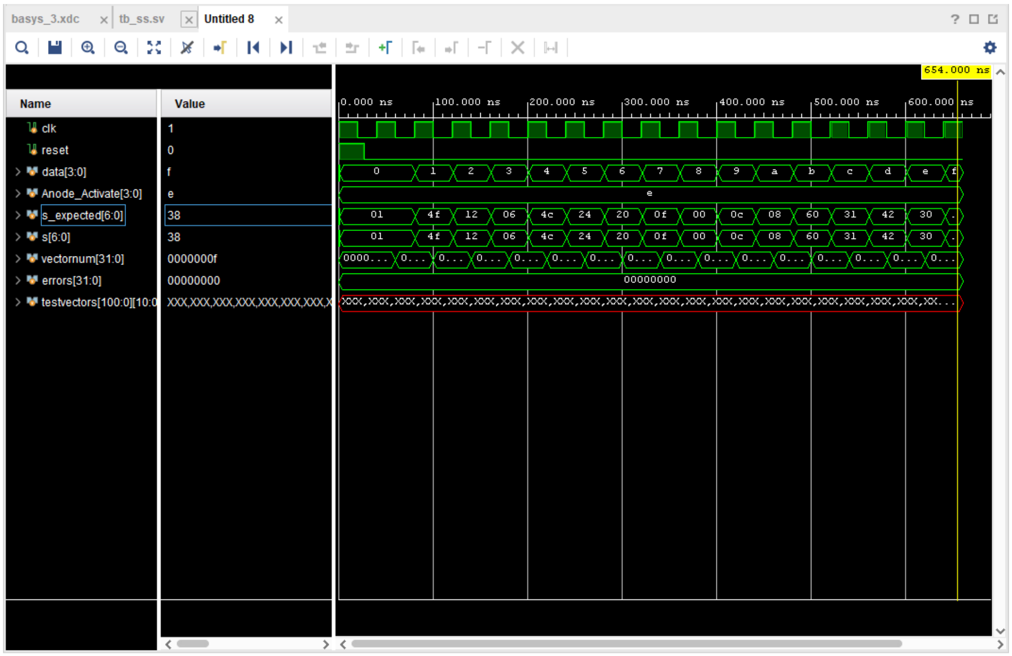


Figure 15: Test Bench Schematic

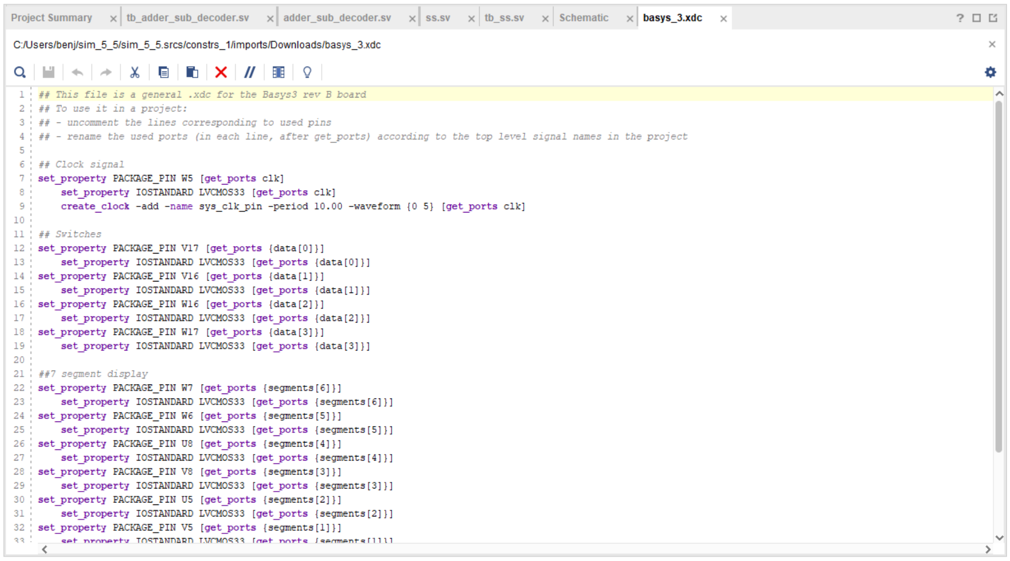
5.7.4 Seven Segment Display Decoder Post Synthesis Timing Simulation



```
Name                      Value
clk                       1
reset                     0
data[3:0]                 f
Anode_Activate[3:0]       e
s_expected[6:0]           38
s[6:0]                    38
vectornum[31:0]           0000000f
errors[31:0]              00000000
testvectors[100:0][10:0   XXX,XXX,XXX,XXX,XXX,XXX,XXX,X
```

Figure 16: Test Bench Post Synthesis Timing Simulation

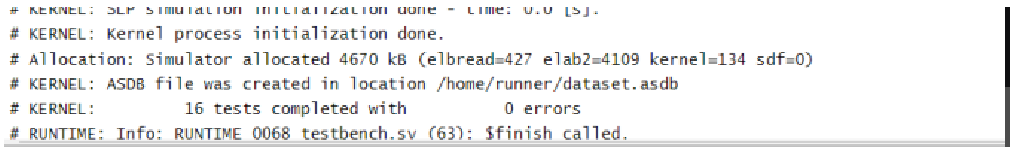
5.7.5 Seven Segment Display Decoder Constraints File



```
## 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 in the project

## 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 {data[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {data[0]}]
set_property PACKAGE_PIN V16 [get_ports {data[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {data[1]}]
set_property PACKAGE_PIN W16 [get_ports {data[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {data[2]}]
set_property PACKAGE_PIN W17 [get_ports {data[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {data[3]}]

##7 segment display
set_property PACKAGE_PIN W7 [get_ports {segments[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {segments[6]}]
set_property PACKAGE_PIN W6 [get_ports {segments[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {segments[5]}]
set_property PACKAGE_PIN U8 [get_ports {segments[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {segments[4]}]
set_property PACKAGE_PIN V8 [get_ports {segments[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {segments[3]}]
set_property PACKAGE_PIN U5 [get_ports {segments[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {segments[2]}]
set_property PACKAGE_PIN V5 [get_ports {segments[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {segments[1]}]
```

Figure 17: Basys3 Constraints File

5.7.6 Seven Segment Display Decoder Console Output



```
# KERNEL: SLP simulation initialization done - time: 0.0 [s].
# KERNEL: Kernel process initialization done.
# Allocation: Simulator allocated 4670 kB (elbread=427 elab2=4109 kernel=134 sdf=0)
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# KERNEL:         16 tests completed with          0 errors
# RUNTIME: Info: RUNTIME_0068 testbench.sv (63): $finish called.
```

Figure 18: Seven Segment Display Decoder Console Output

5.8 Results and Analysis Discussion

Provide a detailed explanation of the RTL schematic hardware generated and associate each element with the corresponding code. Essentially, you need to explain the code-to-hardware relationship?

The schematic diagram shows two inputs, *A* and *B*, correlating with the adder-subtractor inputs in the code. The *M* input determines whether the adder-subtractor acts as a subtractor or as an adder. The adder-subtractor's output is fed into the seven segment decoder (*sevenseg* module), which generates an output mirroring the results from the earlier truth table.

What is the delay from an input change to a stable digit input and segment output value when performing a post synthesis timing simulation?

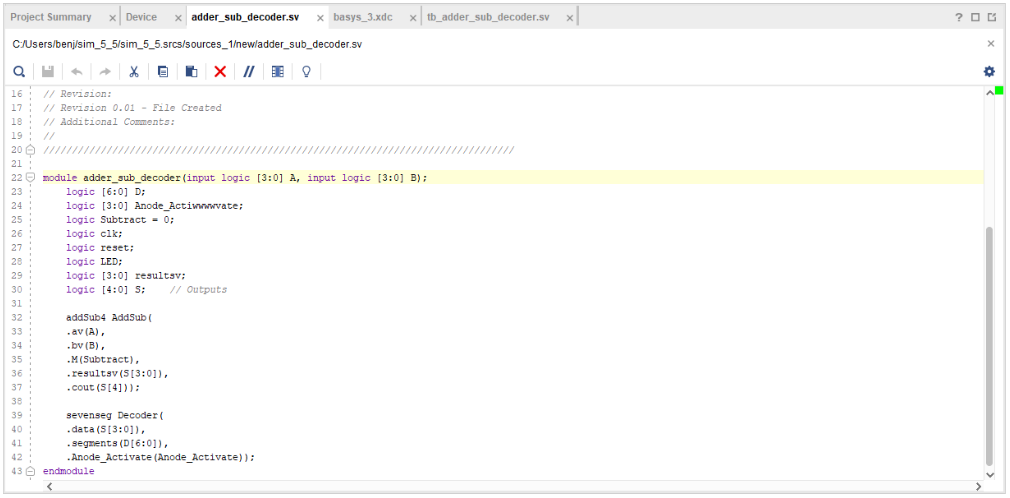
The delay from an input change to a stable *digit* input and *segment* output value when performing a post synthesis timing simulation is about 20 nanoseconds (ns). We find this value by measuring the endpoint of when an active high of a stable digital input is simulated, after which we find the time gap from the endpoint of stable input to the start of segment output value. Therefore, the difference in delay can be modeled from the equation below, but the final result should be 20 ns. We define active as when the simulation showcases the state of either being active low signal or high signal. To account for the differences in readings, we should mention that we do not expect the delay that we recorded to be exact with what is measured in the rubric.

$$Delay = Segment_{OUT} - Digit_{IN} \tag{1}$$

This section paved the way to showing the group specifically how the rest of the simulation assignment was going to go. That conclusion being filled with galling issues and warnings. The group spent a lot of time trying to debug Vivado, some methods being renaming some files, checking environment variables to ensure that everything was accurate, or even restarting the entire software. It seemed as if no matter what we did, we kept getting the "***spawn failed, no error***" message. At times, our schematic files didn't even open, which made taking screenshots or verifying our work much difficult than previously. With so many issues regarding bugs, this section, and this simulation assignment in general, proved itself to be much more insufferable than any of the other previous one, no matter their difficulty.

6 4-Bit Adder/Subtractor with Seven Segment Display Decoder

6.1 4-Bit Adder/Subtractor with Seven Segment Display Decoder Implementation Code



```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module adder_sub_decoder(input logic [3:0] A, input logic [3:0] B);
    logic [6:0] D;
    logic [3:0] Anode_Actiwwwwvate;
    logic Subtract = 0;
    logic clk;
    logic reset;
    logic LED;
    logic [3:0] resultsv;
    logic [4:0] S;    // Outputs

    addSub4 AddSub(
    .av(A),
    .bv(B),
    .M(Subtract),
    .resultsv(S[3:0]),
    .cout(S[4]));

    sevenseg Decoder(
    .data(S[3:0]),
    .segments(D[6:0]),
    .Anode_Activate(Anode_Activate));
endmodule
```

Figure 19: 4-Bit Adder/Subtractor with Seven Segment Display Decoder Implementation Code

6.2 4-Bit Adder/Subtractor with Seven Segment Display Decoder RTL Schematic

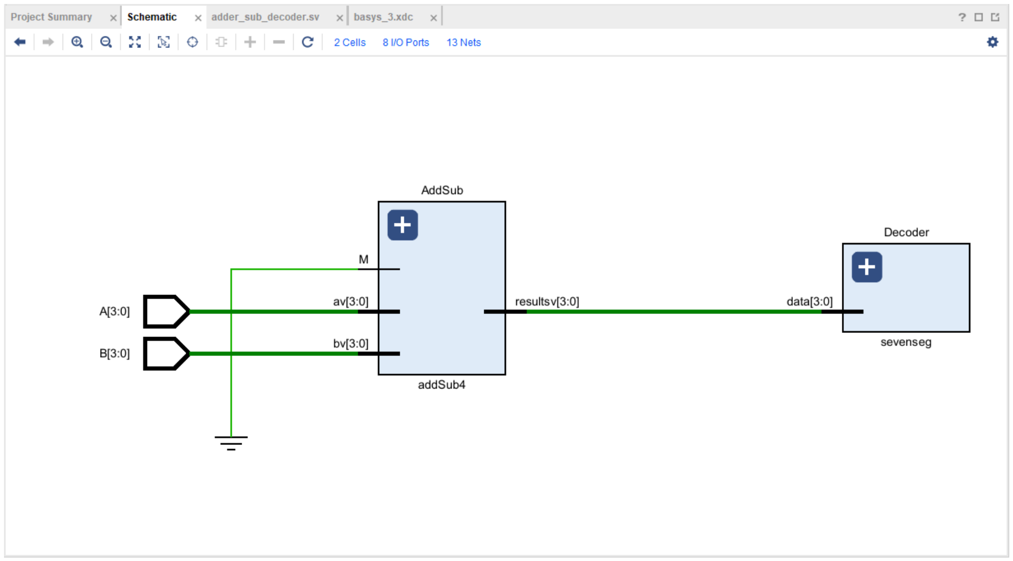


Figure 20: RTL Schematic

6.3 4-Bit Adder/Subtractor with Seven Segment Display Decoder Synthesized Design

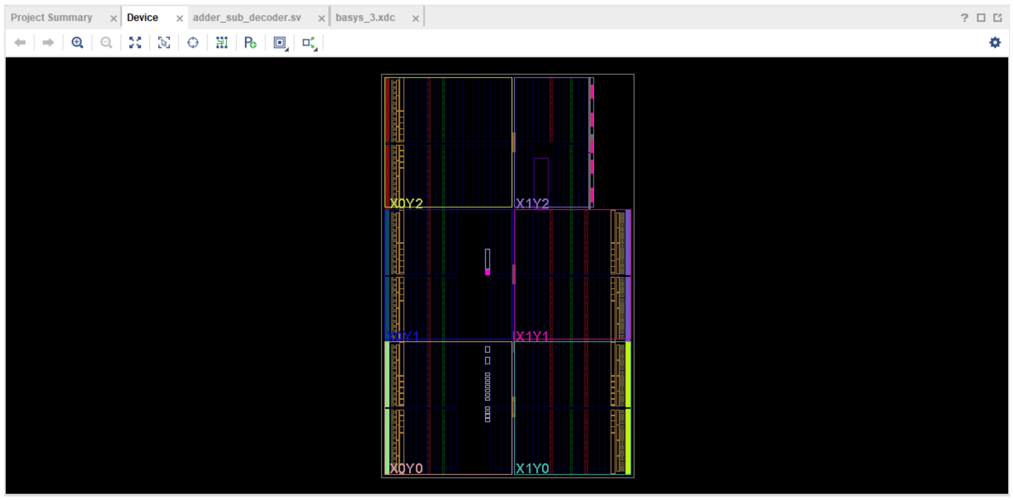
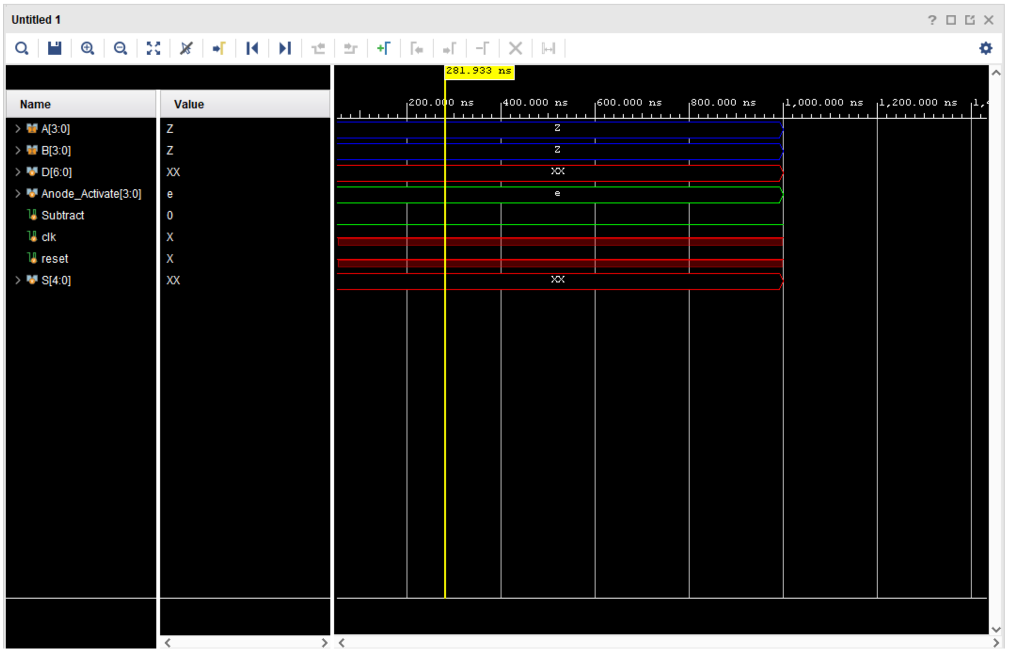


Figure 21: Synthesized Design

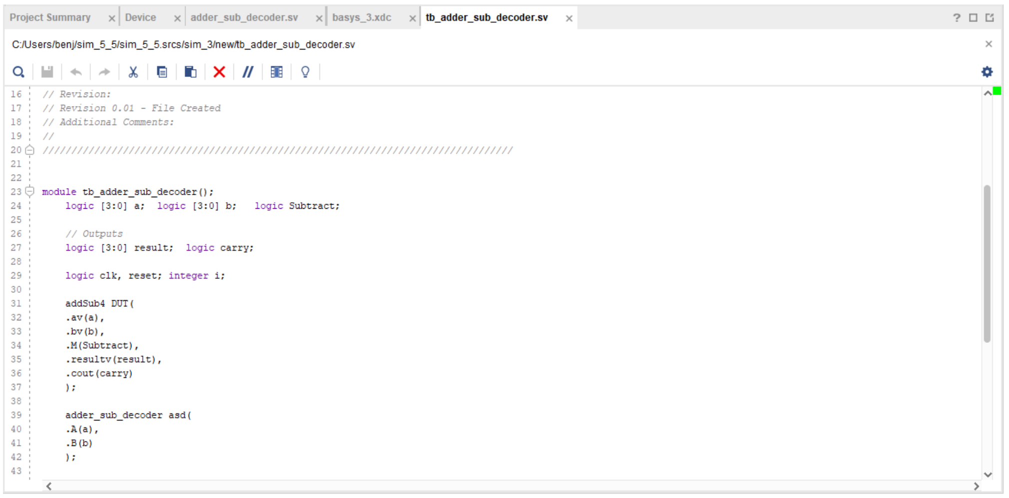


Name	Value
A[3:0]	Z
B[3:0]	Z
D[6:0]	XX
Anode_Activate[3:0]	e
Subtract	0
clk	X
reset	X
S[4:0]	XX

Figure 22: Behavioral Simulation

6.4 4-Bit Adder/Subtractor with Seven Segment Display Decoder Test Bench

6.4.1 4-Bit Adder/Subtractor with Seven Segment Display Decoder Test Bench Code



C:/Users/benj/sim_5_5/sim_5_5.srcs/sim_3/new/tb_adder_sub_decoder.sv

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module tb_adder_sub_decoder();
    logic [3:0] a;  logic [3:0] b;   logic Subtract;

    // Outputs
    logic [3:0] result;  logic carry;

    logic clk, reset; integer i;

    addSub4 DUT(
    .av(a),
    .bv(b),
    .M(Subtract),
    .resultv(result),
    .cout(carry)
    );

    adder_sub_decoder asd(
    .A(a),
    .B(b)
    );

```

Figure 23: Test Bench Code

6.4.2 4-Bit Adder/Subtractor with Seven Segment Display Decoder Test Bench Behavioral Simulation

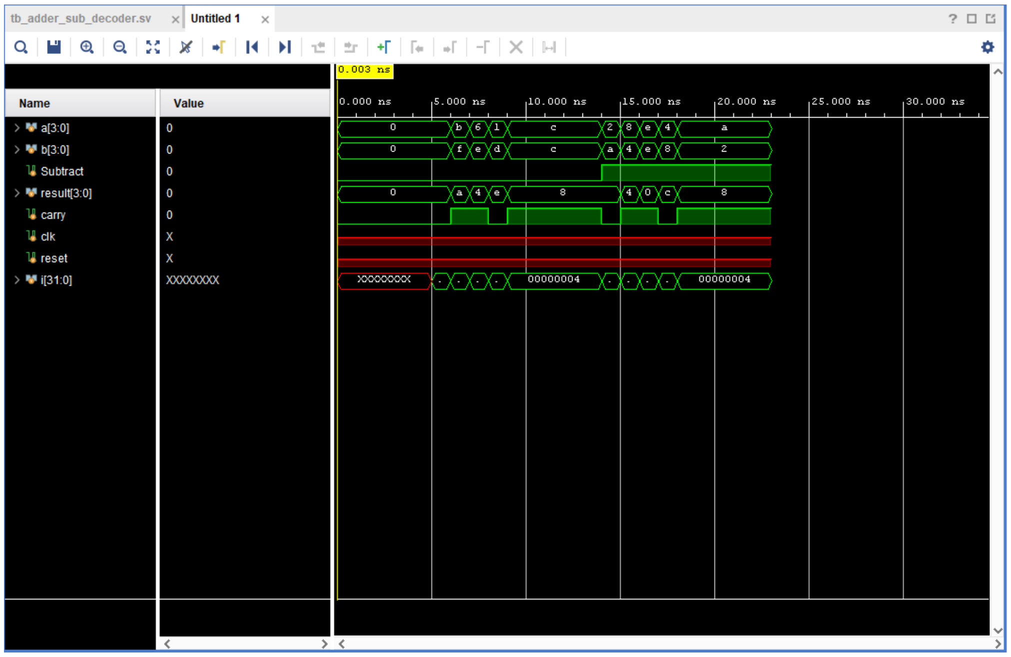
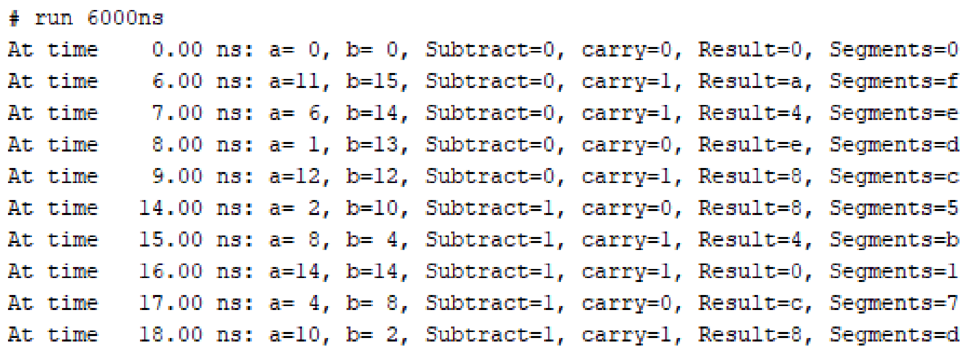


Figure 24: Test Bench Behavioral Simulation

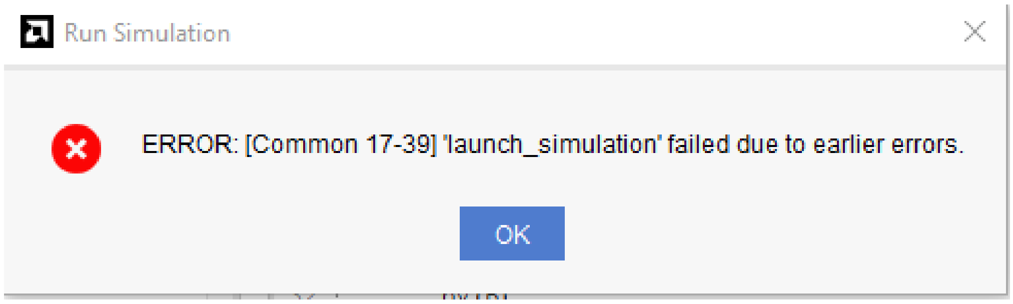
6.4.3 4-Bit Adder/Subtractor with Seven Segment Display Decoder Test Bench Console Output



```
# run 6000ns
At time    0.00 ns: a= 0, b= 0, Subtract=0, carry=0, Result=0, Segments=0
At time    6.00 ns: a=11, b=15, Subtract=0, carry=1, Result=a, Segments=f
At time    7.00 ns: a= 6, b=14, Subtract=0, carry=1, Result=4, Segments=e
At time    8.00 ns: a= 1, b=13, Subtract=0, carry=0, Result=e, Segments=d
At time    9.00 ns: a=12, b=12, Subtract=0, carry=1, Result=8, Segments=c
At time   14.00 ns: a= 2, b=10, Subtract=1, carry=0, Result=8, Segments=5
At time   15.00 ns: a= 8, b= 4, Subtract=1, carry=1, Result=4, Segments=b
At time   16.00 ns: a=14, b=14, Subtract=1, carry=1, Result=0, Segments=1
At time   17.00 ns: a= 4, b= 8, Subtract=1, carry=0, Result=c, Segments=7
At time   18.00 ns: a=10, b= 2, Subtract=1, carry=1, Result=8, Segments=d
```

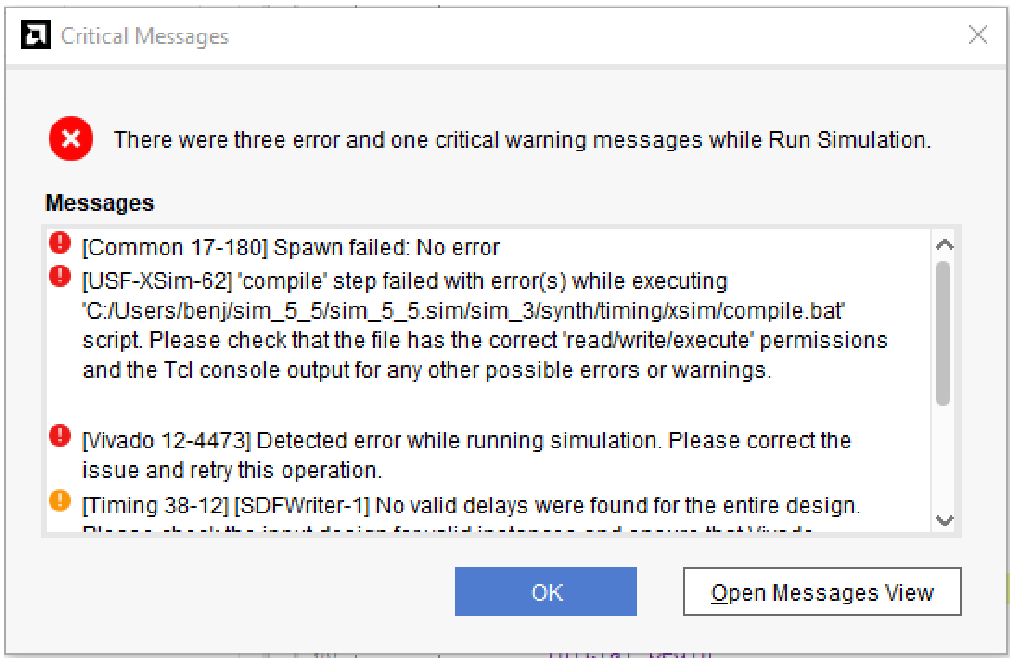
Figure 25: Test Bench Behavioral Simulation Console Output

6.4.4 4-Bit Adder/Subtractor with Seven Segment Display Decoder Test Bench Post Synthesis Timing Simulation


Run Simulation

ERROR: [Common 17-39] 'launch_simulation' failed due to earlier errors.

Figure 26: Test Bench Post Synthesis Timing Simulation Error #1


Critical Messages

There were three error and one critical warning messages while Run Simulation.

Messages

```
[Common 17-180] Spawn failed: No error
[USF-XSim-62] 'compile' step failed with error(s) while executing
'C:/Users/benj/sim_5_5/sim_5_5.sim/sim_3/synth/timing/xsim/compile.bat'
script. Please check that the file has the correct 'read/write/execute' permissions
and the Tcl console output for any other possible errors or warnings.

[Vivado 12-4473] Detected error while running simulation. Please correct the
issue and retry this operation.
[Timing 38-12] [SDFWriter-1] No valid delays were found for the entire design.
```

Figure 27: Test Bench Post Synthesis Timing Simulation Error #2

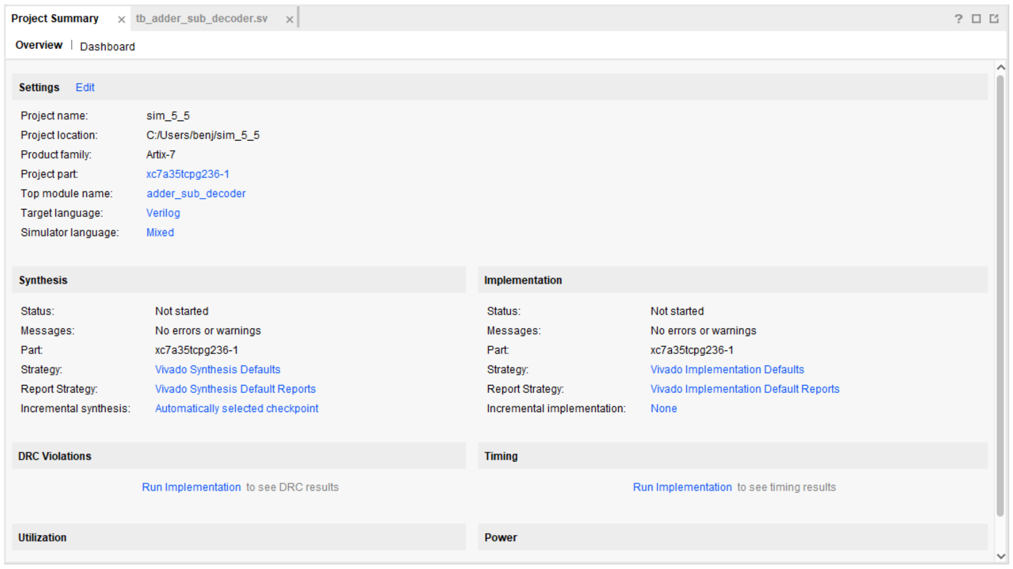
6.5 Results and Analysis Discussion

The 4-bit adder and subtractor section continued the trend of exasperation displayed in the first section, but somehow much worse. This section involved issues such as files not running because of extra spaces and difficulties coding the test benches. There were times when the technical difficulties were so abundant that the group almost gave up. ***We were not able to open synthesized design schematic for seven***

segment decoder adder subtractor post synthesis timing simulation for the test bench seven segment decoder adder subtractor was not completed.

Despite the efforts made by the group, the errors highlighted in the previous paragraph still remain present due to difficulties coming up with solutions for them. The fact that those errors are there with no clear solution in mind highlights the difficulty present in the simulation assignment. The group hopes to ask questions pertaining to the solutions for these problems to the professor.

7 Project Summary



Project Summary | tb_adder_sub_decoder.sv

Overview | Dashboard

Settings Edit

Project name:	sim_5_5
Project location:	C:/Users/benj/sim_5_5
Product family:	Artix-7
Project part:	xc7a35tcpg236-1
Top module name:	adder_sub_decoder
Target language:	Verilog
Simulator language:	Mixed

Synthesis

Status:	Not started
Messages:	No errors or warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports
Incremental synthesis:	Automatically selected checkpoint

Implementation

Status:	Not started
Messages:	No errors or warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None

DRC Violations

Run Implementation to see DRC results

Timing

Run Implementation to see timing results

Utilization

Power

Figure 28: Project Summary

8 Results and Conclusion

This section discusses the results of the Vivado simulation experiments. The discussion provides in-depth explanations of the results and any discrepancies between the results and theoretical expectations. It also explores potential sources of error and discusses the accuracy of the tests, the debugging process, and what was learned from these experiences.

Simulation Exercise #5.5 was aided in large by the help of the original Simulation Exercise #5. This major aspect of the exercise provided a semblance of familiarity with the topic and allowed the group to remain on the same page when it comes to the understanding of the material presented. As per the trend for all previous simulation exercises, this one also had its own share of complexity that forced the group to think.

However, there were some forms of complexity that went beyond the conceptual material needed for the simulation exercise. This was mainly in regards to the large amount of bugs and errors when coding both the seven segment displays and the 4-bit adder and subtractor. There were times when files refused to open because of spaces in the log files or when Vivado would run files different from the ones we wanted. This lowered group morale at large since some issues were based on the software itself breaking. With a lot of turning the software on and off and deleting spaces, we eventually were able to get the screenshot we needed for the report, despite doing so taking a long time to do.

9 Group Contributions

The work for this simulation assignment was divided among three individuals: Benyamain Yacoob, Andre Price Jr, and Eyiara Oladipo. Andre and Benyamain collaborated using Vivado software and writing in SystemVerilog to code the circuits. It should be noted, however, that this was a group effort and all members were present to answer any questions related to the circuits. Benyamain and Eyiara took the lead with the formatting and organizing the Overleaf document. Benyamain proofread the document before submitting.