ECEN 248 - Lab Report

Lab Number: 4

Lab Title:Simple Arithmetic Logic Unit

Section Number: 510

Student's Name: Paola Avila

Student's UIN: 731007033

Date:3/2/2024

I. Objectives:

In this laboratory session, students gained proficiency in replicating the prior experiment through the utilization of Verilog to program the individual integrated circuits and their associated inputs. The objective of this laboratory session is to acquaint students with Verilog and instruct them on utilizing it for simulating digital circuits.

II. Design:

The laboratory session began by creating a directory named ecen248 in the home directory using the terminal. I began by executing the command "source /opt/coe/Xilinx/Vivado/2015.2/settings64.sh" and then proceeded to initiate Vivado by using the command "vivado". I initiated a new project in Vivado called 'lab5' and selected the 'Zybo Z7-10' board. Following that, a new file named 'two_one_mux.v' was created, and the code provided in the lab manual was duplicated within it. I have added this file to the collection of design assets. Subsequently, I proceeded to initiate a fresh terminal session and then navigated to the specific directory by executing the command "cd $HOME/ecen248/lab5". By executing the command "cp /mnt/lab_files/ECEN248/two_one_mux_tb.v .", I was able to effectively replicate the file titled "two_one_mux_tb.v" within the present directory. During my review of Vivado, I included the test bench file in the simulation sources and assigned it as the top file while keeping unchanged. Afterward, I proceeded with the simulation launch. If the waveform were devoid of any errors, it would be presented. The aforementioned procedure was duplicated for the 4-bit multiplexer, full adder, adder-subtractor, and 4-bit arithmetic logic unit, while adhering to a professional tone and grammatical accuracy in English language, without changing the value of . "While undertaking the coding of the remaining files, I took the initiative to acquaint myself with the code's functionality and acquire the skills required to employ it autonomously for the purpose of achieving the objectives at hand." Throughout the entire process, I made it a priority to capture images of the waveforms and code.

III . Results :

Task1 : Gate-level Schematic of a 1-bit wide, 2:1 MUX

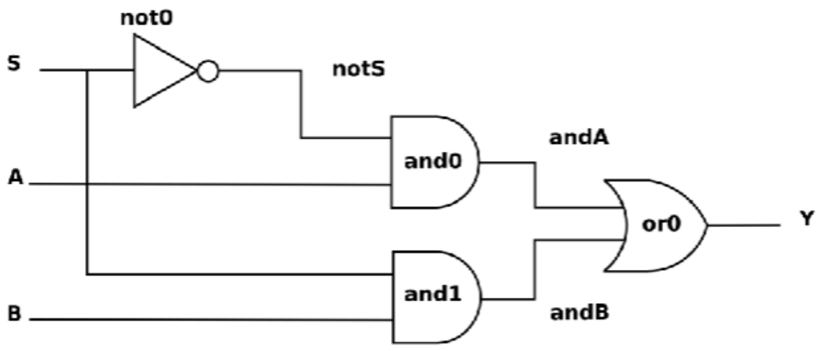
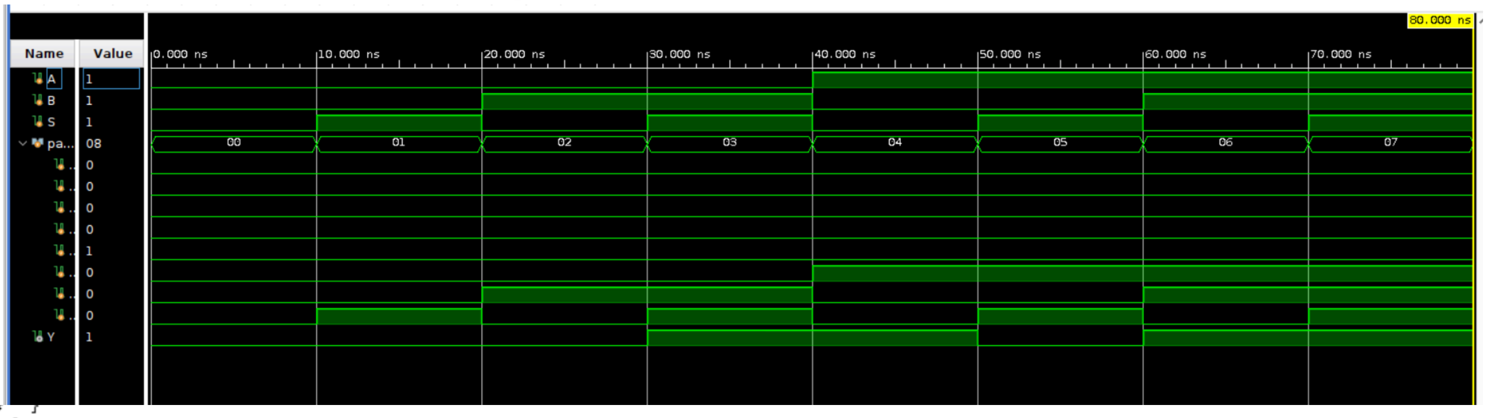
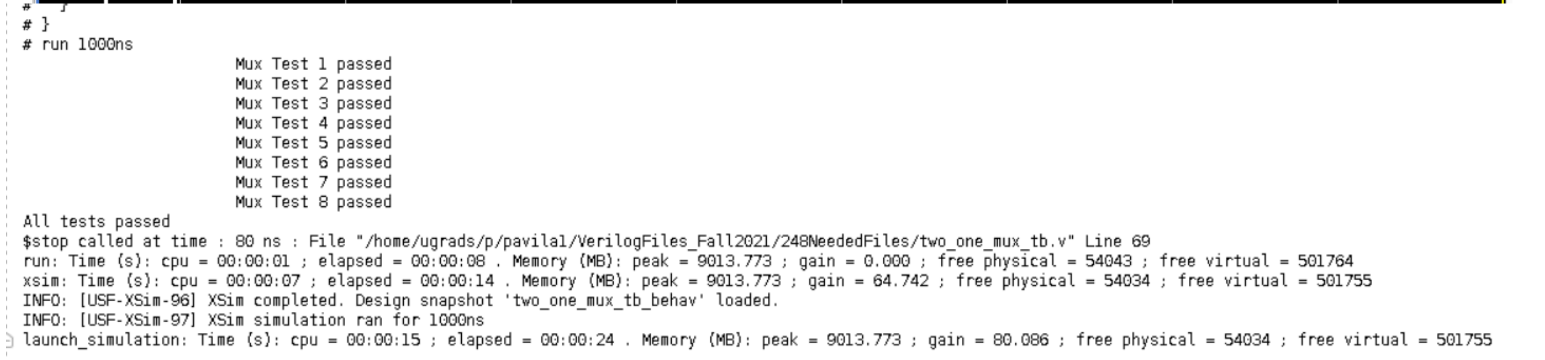
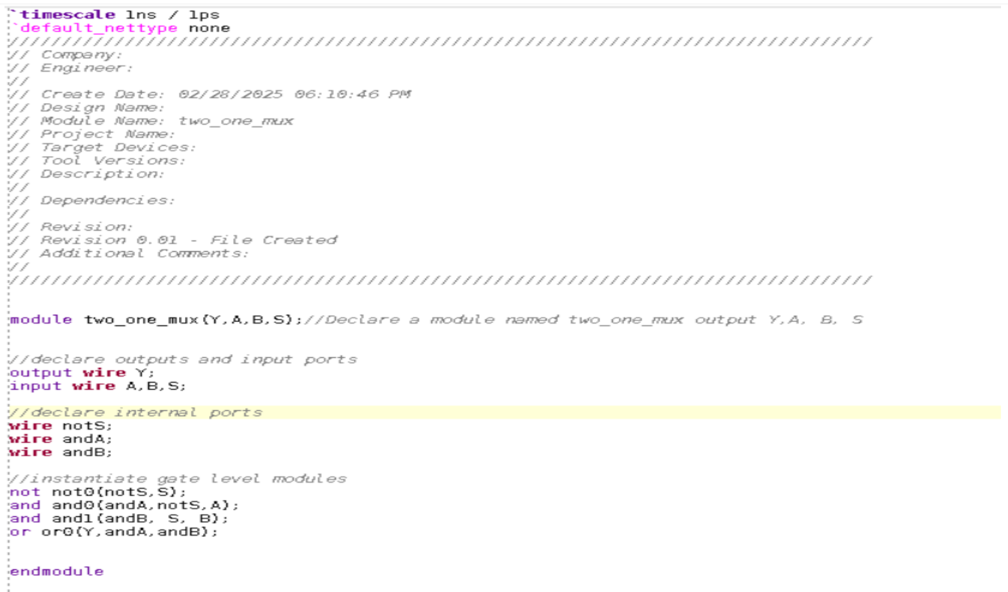


Figure 1: Gate-level Schematic of a 1-bit wide, 2:1 MUX





```
# }
# run 1000ns
                    Mux Test 1 passed
                    Mux Test 2 passed
                    Mux Test 3 passed
                    Mux Test 4 passed
                    Mux Test 5 passed
                    Mux Test 6 passed
                    Mux Test 7 passed
                    Mux Test 8 passed
All tests passed
$stop called at time : 80 ns : File "/home/ugrads/p/pavilal/VerilogFiles_Fall2021/248NeededFiles/two_one_mux_tb.v" Line 69
run: Time (s): cpu = 00:00:01 ; elapsed = 00:00:08 . Memory (MB): peak = 9013.773 ; gain = 0.000 ; free physical = 54043 ; free virtual = 501764
xsim: Time (s): cpu = 00:00:07 ; elapsed = 00:00:14 . Memory (MB): peak = 9013.773 ; gain = 64.742 ; free physical = 54034 ; free virtual = 501755
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_one_mux_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:15 ; elapsed = 00:00:24 . Memory (MB): peak = 9013.773 ; gain = 80.086 ; free physical = 54034 ; free virtual = 501755
```



```verilog
`timescale 1ns / 1ps
`default_nettype none
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/28/2025 06:10:46 PM
// Design Name:
// Module Name: two_one_mux
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module two_one_mux(Y,A,B,S);//Declare a module named two_one_mux output Y,A, B, S

//declare outputs and input ports
output wire Y;
input wire A,B,S;

//declare internal ports
wire notS;
wire andA;
wire andB;

//instantiate gate level modules
not not0(notS,S);
and and0(andA,notS,A);
and and1(andB, S, B);
or or0(Y,andA,andB);


endmodule
```

Task 2 : Four_bit_mux

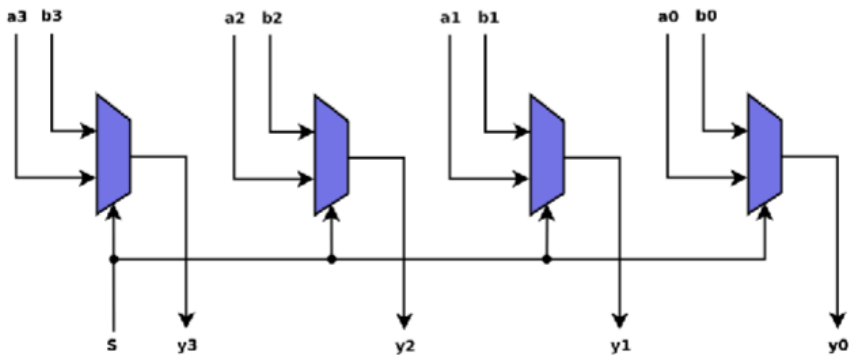
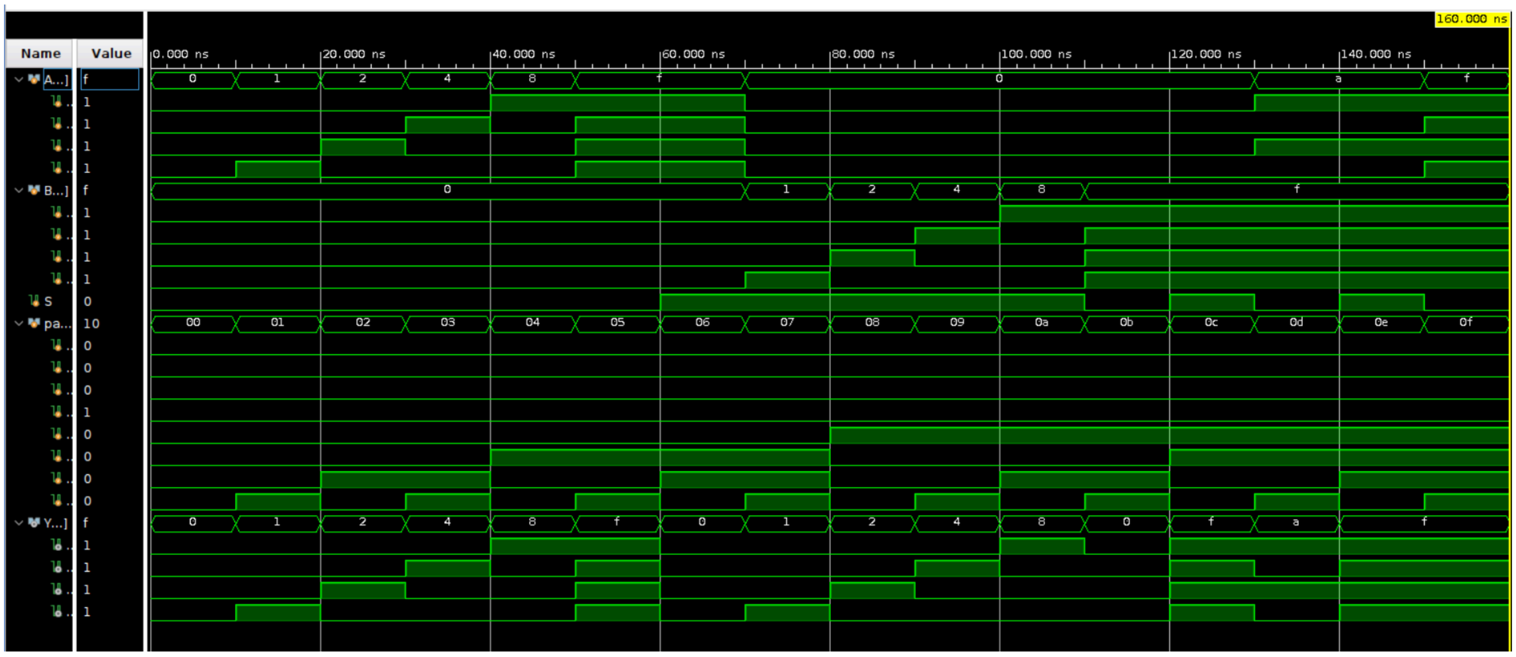
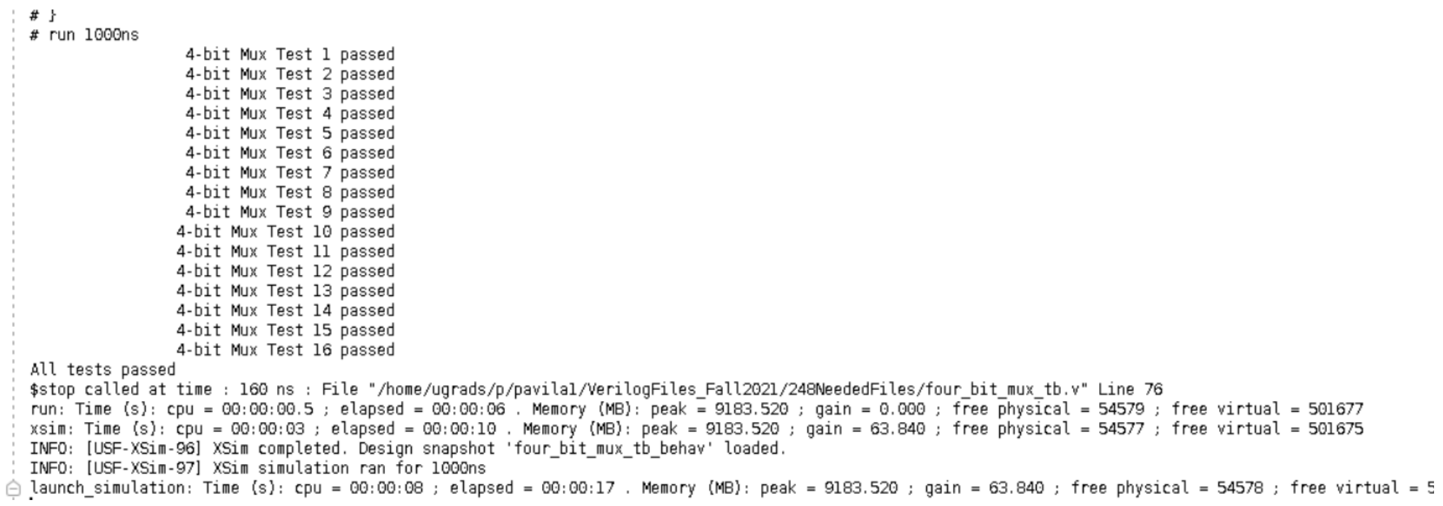
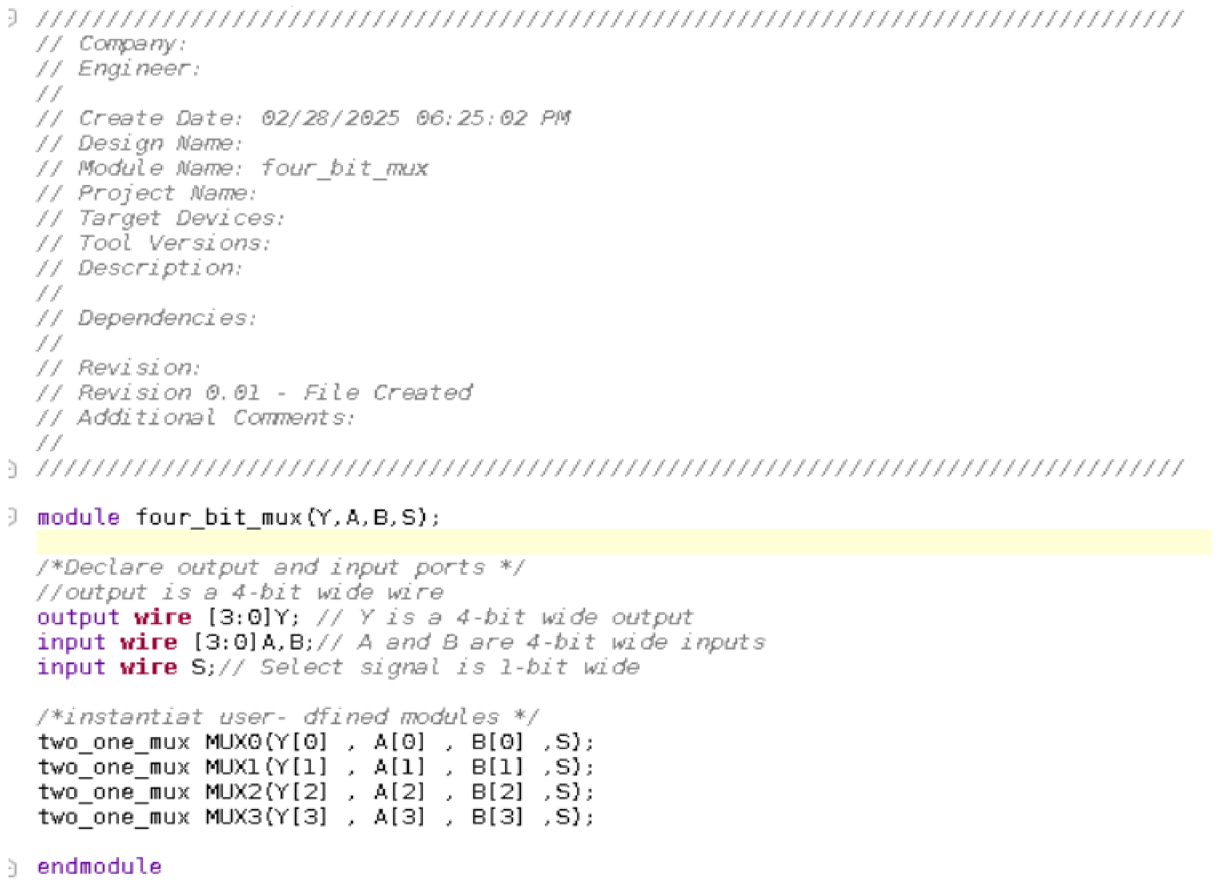


Figure 2: 4-bit wide, 2:1 MUX





```
# }
# run 1000ns
                 4-bit Mux Test 1 passed
                 4-bit Mux Test 2 passed
                 4-bit Mux Test 3 passed
                 4-bit Mux Test 4 passed
                 4-bit Mux Test 5 passed
                 4-bit Mux Test 6 passed
                 4-bit Mux Test 7 passed
                 4-bit Mux Test 8 passed
                 4-bit Mux Test 9 passed
                4-bit Mux Test 10 passed
                4-bit Mux Test 11 passed
                4-bit Mux Test 12 passed
                4-bit Mux Test 13 passed
                4-bit Mux Test 14 passed
                4-bit Mux Test 15 passed
                4-bit Mux Test 16 passed
All tests passed
$stop called at time : 160 ns : File "/home/ugrads/p/pavila1/VerilogFiles_Fall2021/248NeededFiles/four_bit_mux_tb.v" Line 76
run: Time (s): cpu = 00:00:00.5 ; elapsed = 00:00:06 . Memory (MB): peak = 9183.520 ; gain = 0.000 ; free physical = 54579 ; free virtual = 501677
xsim: Time (s): cpu = 00:00:03 ; elapsed = 00:00:10 . Memory (MB): peak = 9183.520 ; gain = 63.840 ; free physical = 54577 ; free virtual = 501675
INFO: [USF-XSim-96] XSim completed. Design snapshot 'four_bit_mux_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:08 ; elapsed = 00:00:17 . Memory (MB): peak = 9183.520 ; gain = 63.840 ; free physical = 54578 ; free virtual = 5
```



```verilog
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/28/2025 06:25:02 PM
// Design Name:
// Module Name: four_bit_mux
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module four_bit_mux(Y,A,B,S);

/*Declare output and input ports */
//output is a 4-bit wide wire
output wire [3:0]Y; // Y is a 4-bit wide output
input wire [3:0]A,B;// A and B are 4-bit wide inputs
input wire S;// Select signal is 1-bit wide

/*instantiat user- dfined modules */
two_one_mux MUX0(Y[0] , A[0] , B[0] ,S);
two_one_mux MUX1(Y[1] , A[1] , B[1] ,S);
two_one_mux MUX2(Y[2] , A[2] , B[2] ,S);
two_one_mux MUX3(Y[3] , A[3] , B[3] ,S);

endmodule
```

Task 3: Full adder

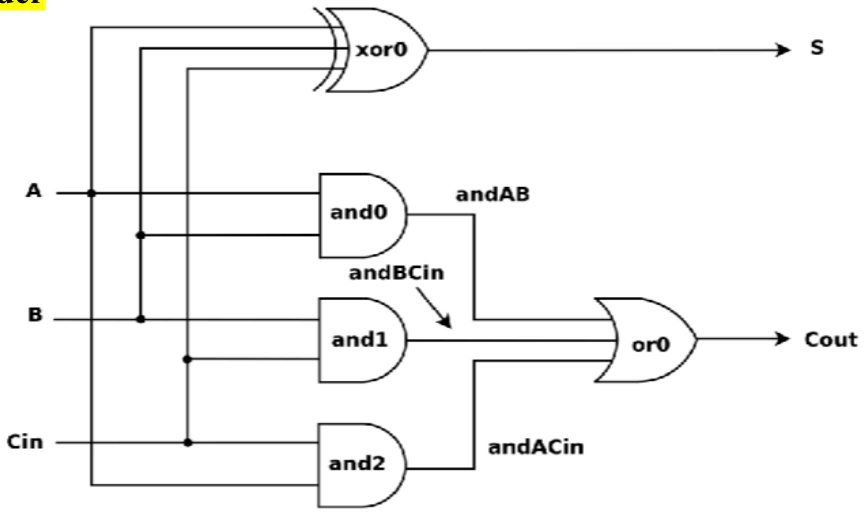
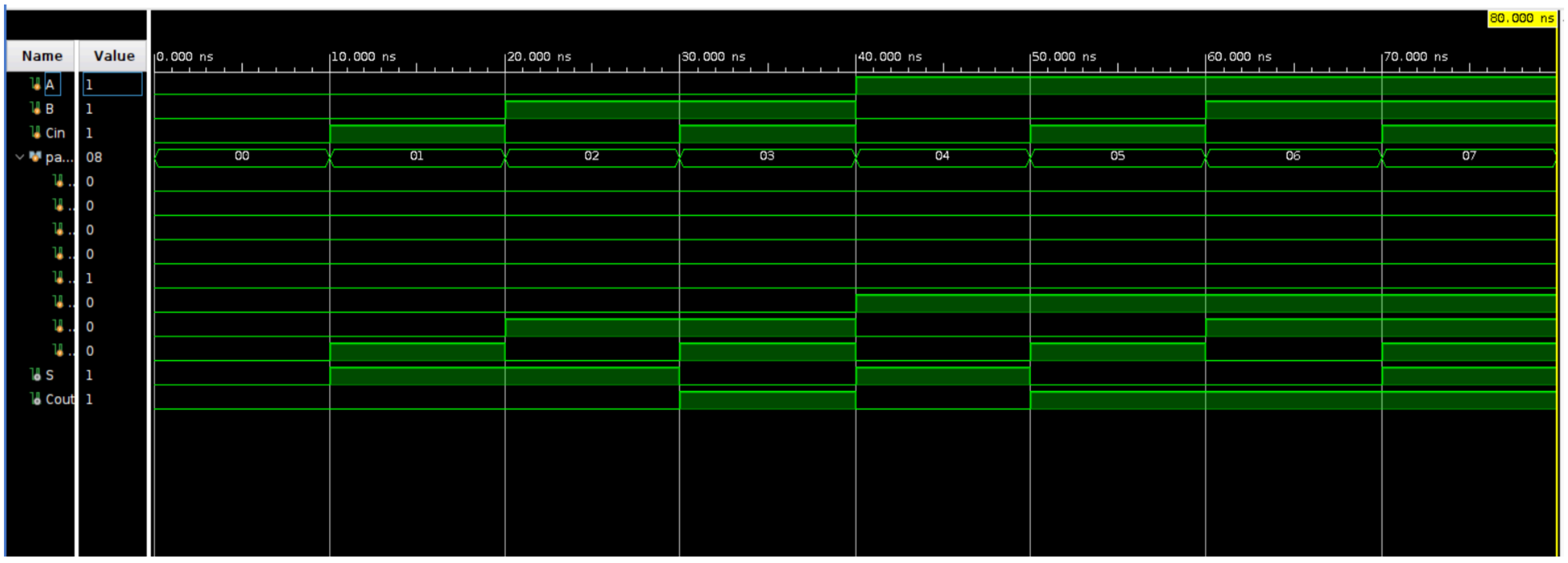
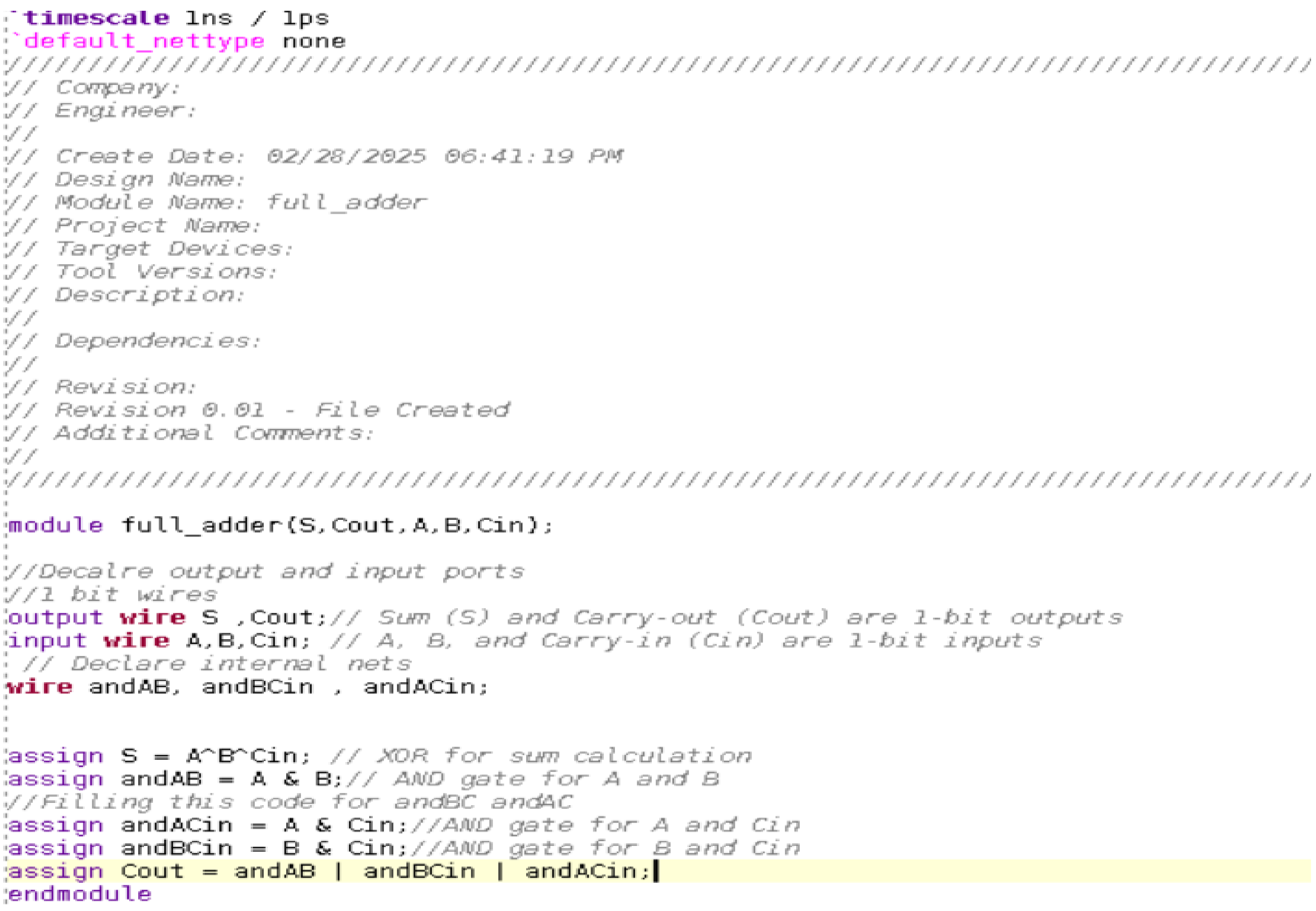


Figure 3: Full-Adder Gate-level Schematic



```
;  ,
: }
: run 1000ns
                Full Adder Test 1 passed
                Full Adder Test 2 passed
                Full Adder Test 3 passed
                Full Adder Test 4 passed
                Full Adder Test 5 passed
                Full Adder Test 6 passed
                Full Adder Test 7 passed
                Full Adder Test 8 passed
ll tests passed
stop called at time : 80 ns : File "/home/ugrads/p/pavilal/VerilogFiles_Fall2021/248NeededFiles/full_adder_tb.v" Line 59
sim: Time (s): cpu = 00:00:03 ; elapsed = 00:00:10 . Memory (MB): peak = 9307.840 ; gain = 81.648 ; free physical = 61166 ; free virtual = 508449
NFO: [USF-XSim-96] XSim completed. Design snapshot 'full_adder_tb_behav' loaded.
NFO: [USF-XSim-97] XSim simulation ran for 1000ns
aunch_simulation: Time (s): cpu = 00:00:10 ; elapsed = 00:00:19 . Memory (MB): peak = 9307.840 ; gain = 81.648 ; free physical = 61166 ; free virtual = 508449
```



```verilog
`timescale 1ns / 1ps
`default_nettype none
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/28/2025 06:41:19 PM
// Design Name:
// Module Name: full_adder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module full_adder(S,Cout,A,B,Cin);

//Decalre output and input ports
//1 bit wires
output wire S ,Cout;// Sum (S) and Carry-out (Cout) are 1-bit outputs
input wire A,B,Cin; // A, B, and Carry-in (Cin) are 1-bit inputs
 // Declare internal nets
wire andAB, andBCin , andACin;


assign S = A^B^Cin; // XOR for sum calculation
assign andAB = A & B;// AND gate for A and B
//Filling this code for andBC andAC
assign andACin = A & Cin;//AND gate for A and Cin
assign andBCin = B & Cin;//AND gate for B and Cin
assign Cout = andAB | andBCin | andACin;
endmodule
```

Task 4: Addition/SubtractionUnit

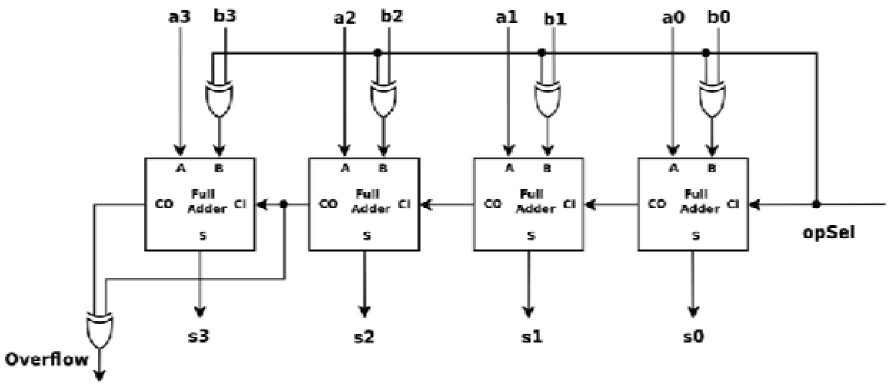
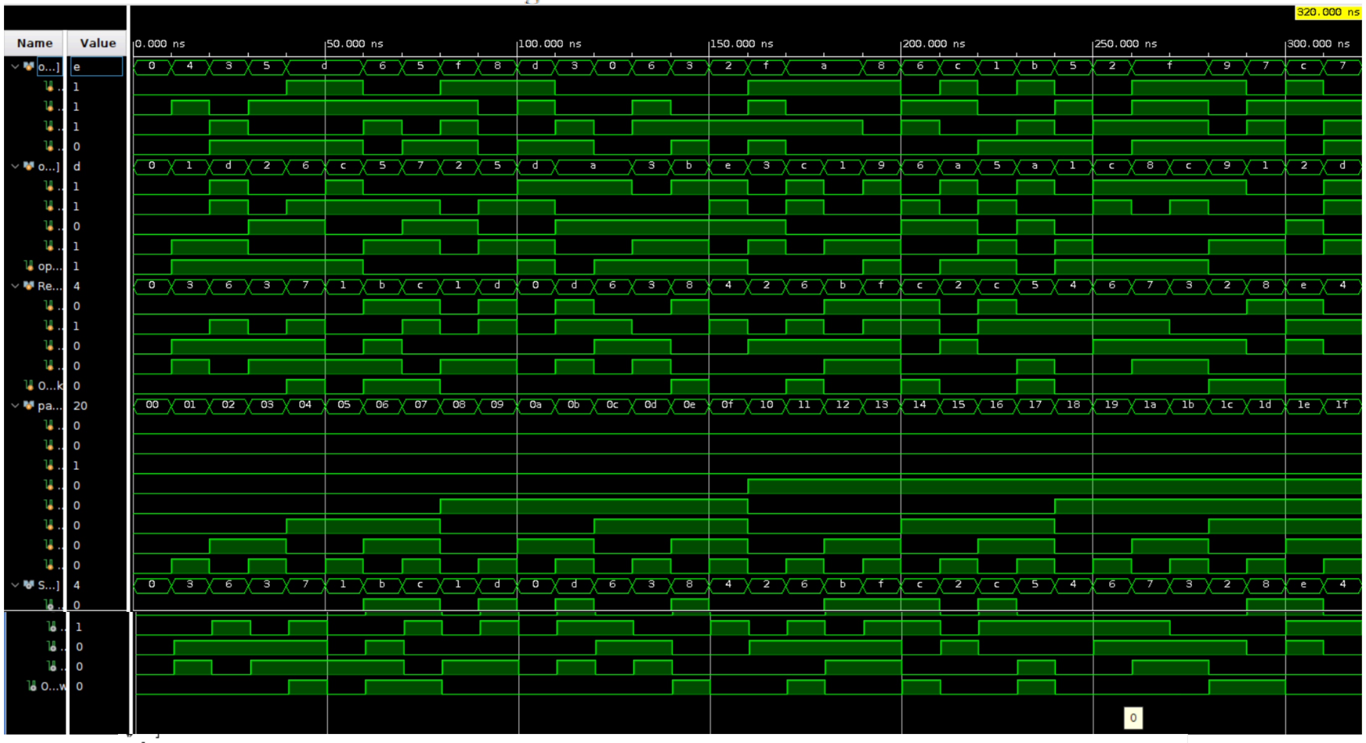
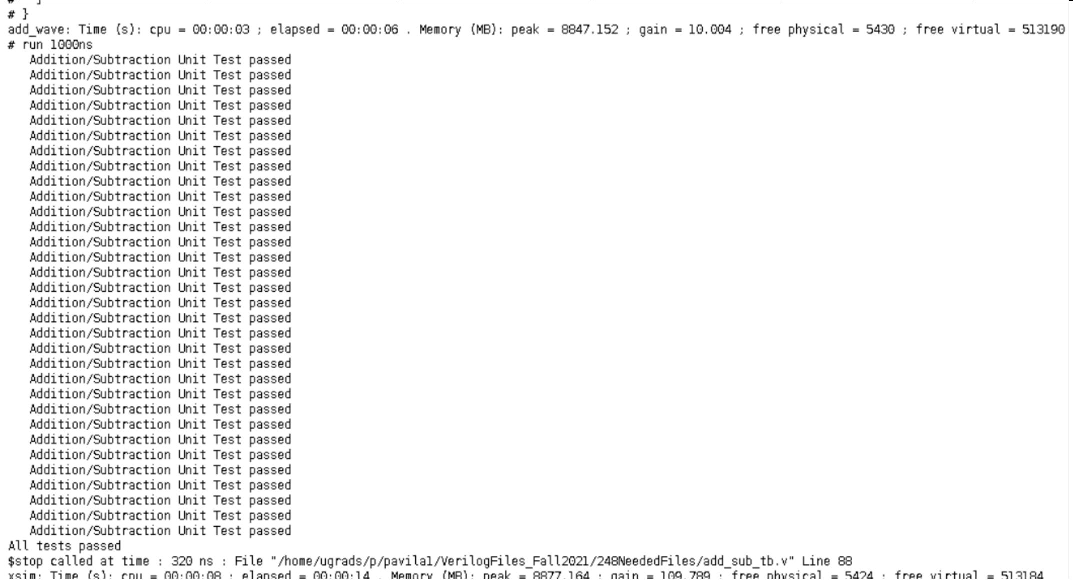
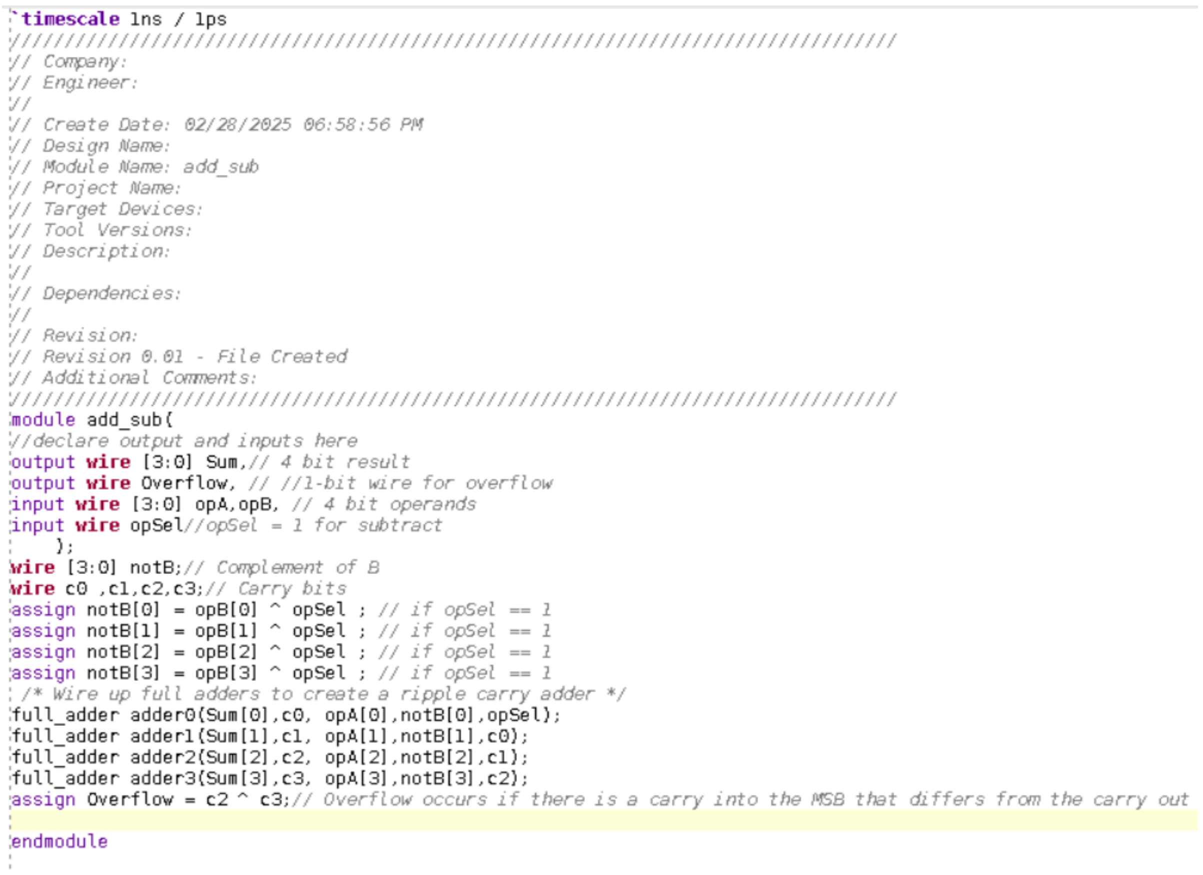


Figure 4: Addition/Subtraction Unit





```
# }
add_wave: Time (s): cpu = 00:00:03 ; elapsed = 00:00:06 . Memory (MB): peak = 8847.152 ; gain = 10.004 ; free physical = 5430 ; free virtual = 513190
# run 1000ns
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
   Addition/Subtraction Unit Test passed
All tests passed
$stop called at time : 320 ns : File "/home/ugrads/p/pavila1/VerilogFiles_Fall2021/248NeededFiles/add_sub_tb.v" Line 88
xsim: Time (s): cpu = 00:00:08 ; elapsed = 00:00:14 . Memory (MB): peak = 8877.164 ; gain = 109.789 ; free physical = 5424 ; free virtual = 513184
```



```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date: 02/28/2025 06:58:56 PM
// Design Name: 
// Module Name: add_sub
// Project Name: 
// Target Devices: 
// Tool Versions: 
// Description: 
// 
// Dependencies: 
// 
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//////////////////////////////////////////////////////////////////////////////////
module add_sub(
//declare output and inputs here
output wire [3:0] Sum,// 4 bit result
output wire Overflow, // //1-bit wire for overflow
input wire [3:0] opA,opB, // 4 bit operands
input wire opSel//opSel = 1 for subtract
    );
wire [3:0] notB;// Complement of B
wire c0 ,c1,c2,c3;// Carry bits
assign notB[0] = opB[0] ^ opSel ; // if opSel == 1
assign notB[1] = opB[1] ^ opSel ; // if opSel == 1
assign notB[2] = opB[2] ^ opSel ; // if opSel == 1
assign notB[3] = opB[3] ^ opSel ; // if opSel == 1
 /* Wire up full adders to create a ripple carry adder */
full_adder adder0(Sum[0],c0, opA[0],notB[0],opSel);
full_adder adder1(Sum[1],c1, opA[1],notB[1],c0);
full_adder adder2(Sum[2],c2, opA[2],notB[2],c1);
full_adder adder3(Sum[3],c3, opA[3],notB[3],c2);
assign Overflow = c2 ^ c3;// Overflow occurs if there is a carry into the MSB that differs from the carry out

endmodule
```

Task 5 : Four_bit_alu :

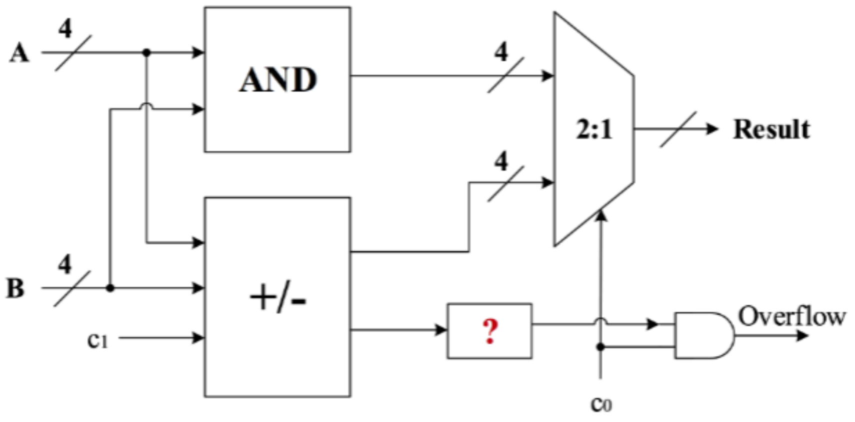
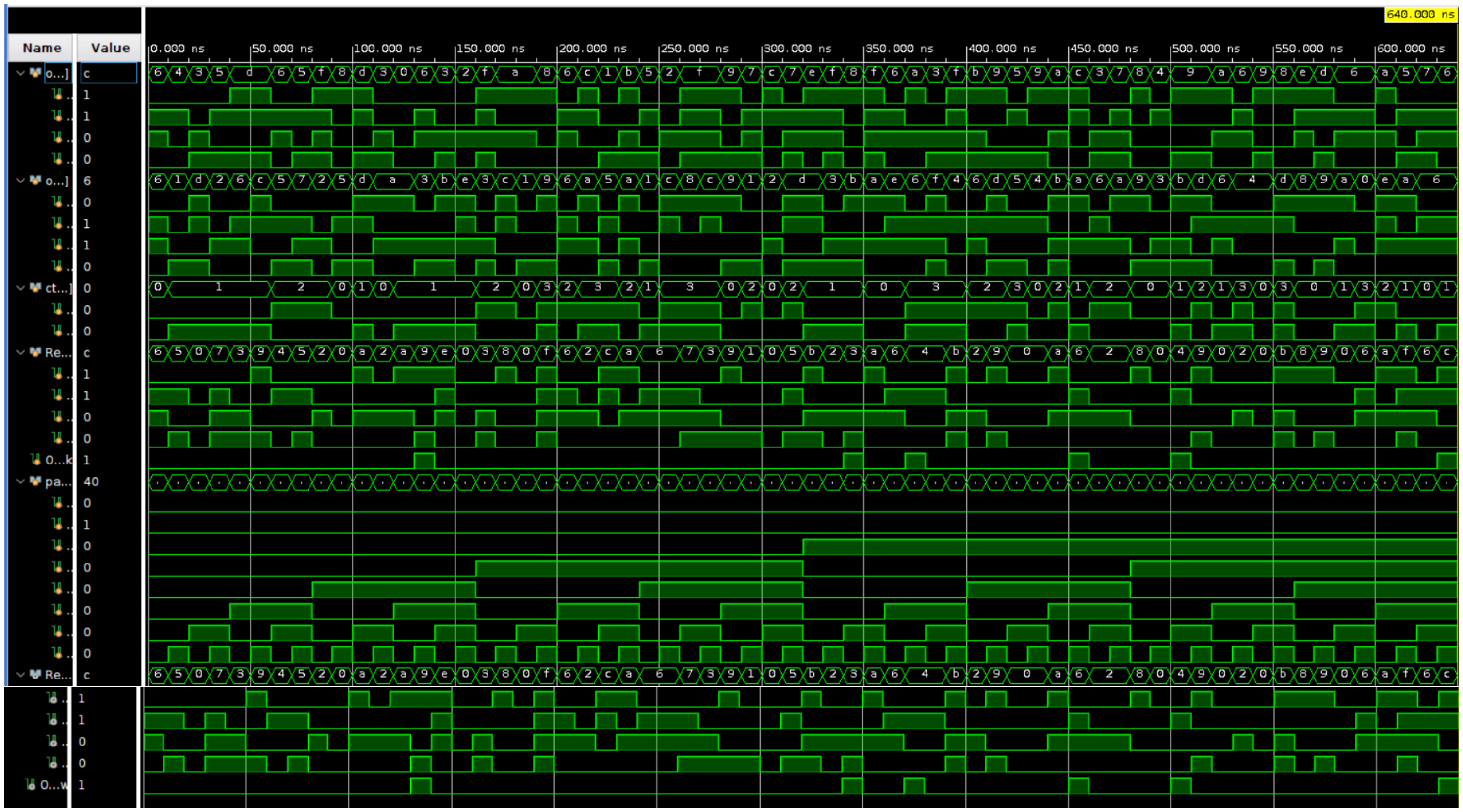
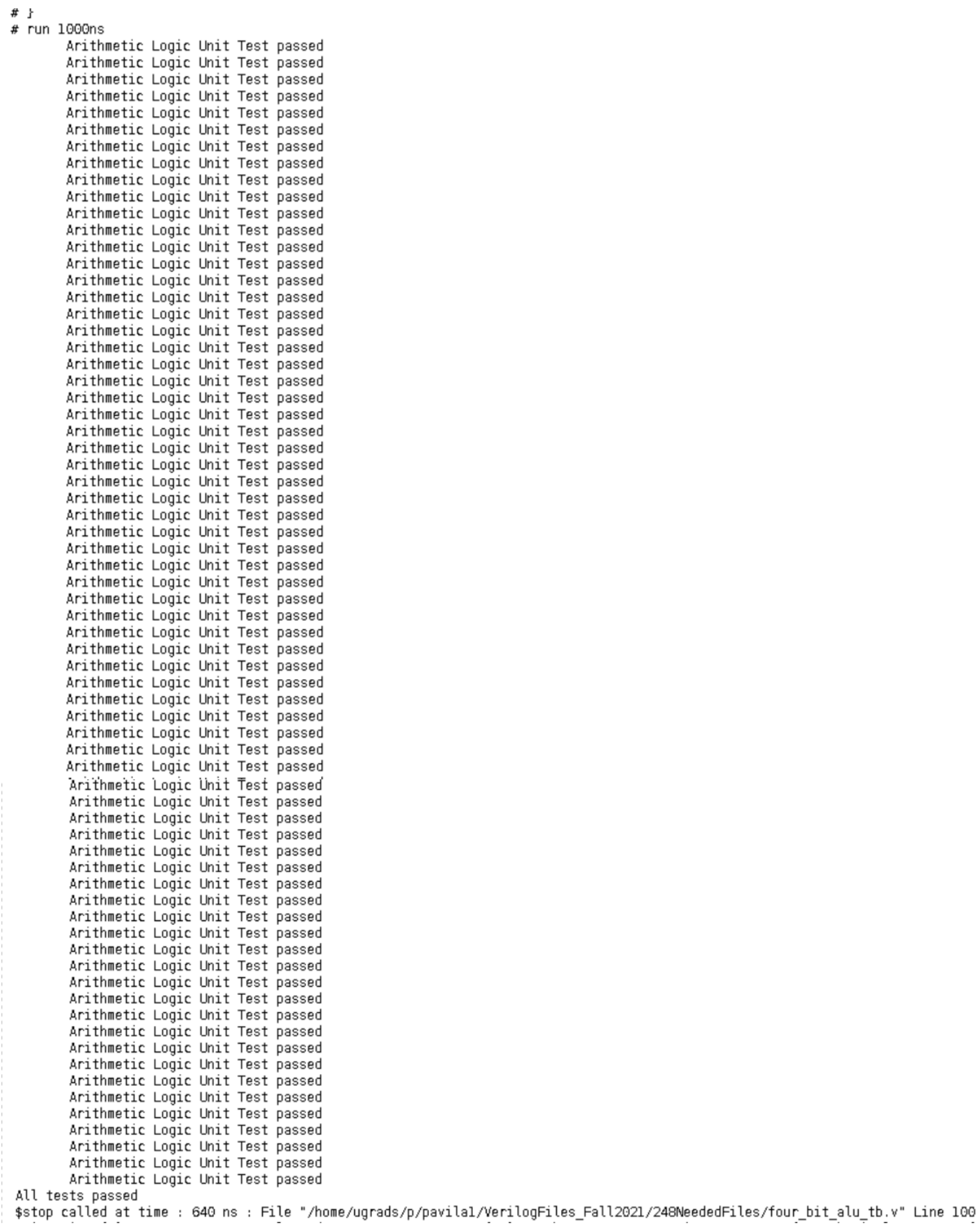
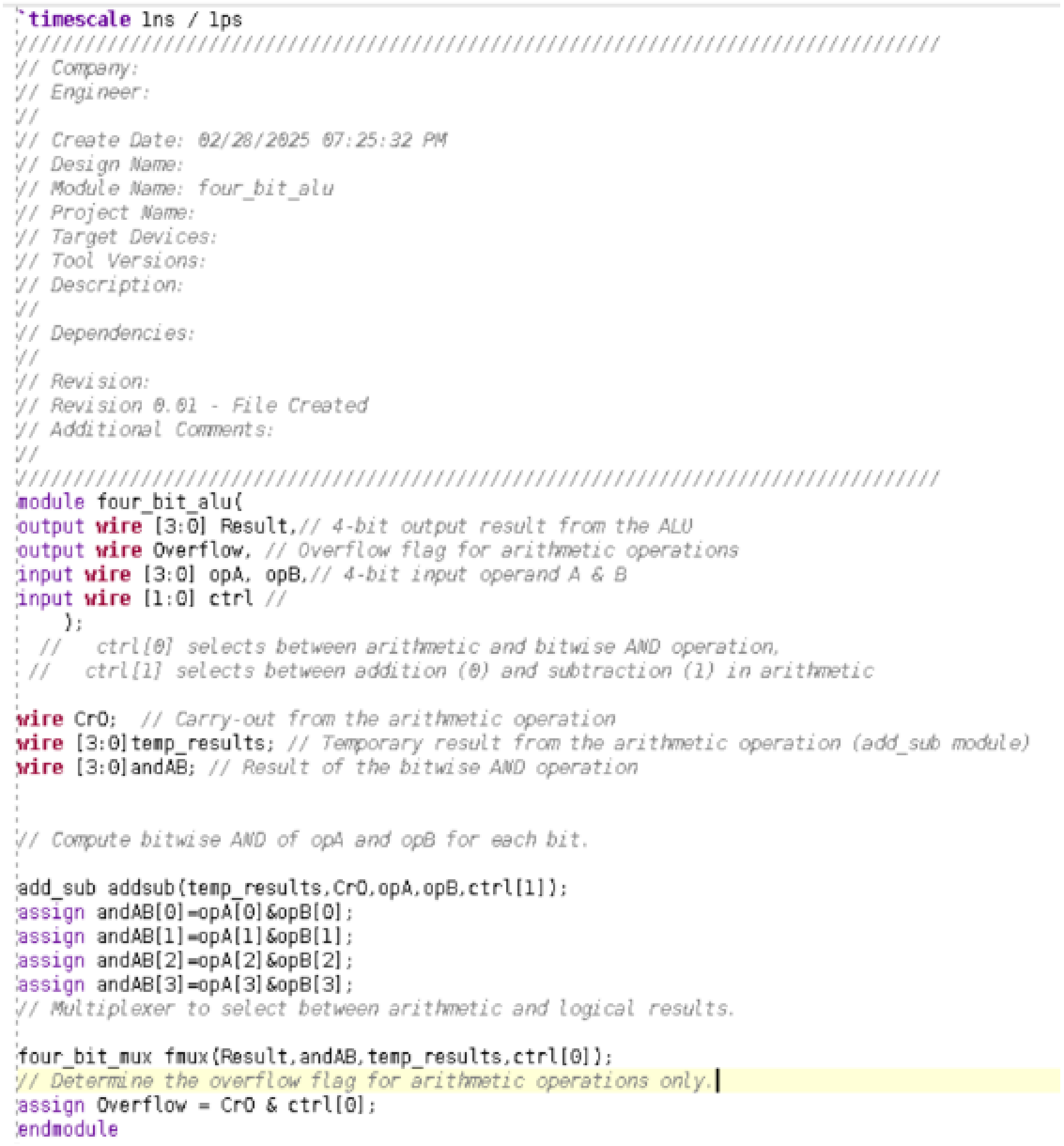


Figure 7: Simple ALU Design





```
# }
# run 1000ns
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
       Arithmetic Logic Unit Test passed
All tests passed
$stop called at time : 640 ns : File "/home/ugrads/p/pavila1/VerilogFiles_Fall2021/248NeededFiles/four_bit_alu_tb.v" Line 100
```



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/28/2025 07:25:32 PM
// Design Name:
// Module Name: four_bit_alu
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module four_bit_alu(
output wire [3:0] Result,// 4-bit output result from the ALU
output wire Overflow, // Overflow flag for arithmetic operations
input wire [3:0] opA, opB,// 4-bit input operand A & B
input wire [1:0] ctrl //
    );
  //   ctrl[0] selects between arithmetic and bitwise AND operation,
 //   ctrl[1] selects between addition (0) and subtraction (1) in arithmetic

wire CrO;  // Carry-out from the arithmetic operation
wire [3:0]temp_results; // Temporary result from the arithmetic operation (add_sub module)
wire [3:0]andAB; // Result of the bitwise AND operation


// Compute bitwise AND of opA and opB for each bit.

add_sub addsub(temp_results,CrO,opA,opB,ctrl[1]);
assign andAB[0]=opA[0]&opB[0];
assign andAB[1]=opA[1]&opB[1];
assign andAB[2]=opA[2]&opB[2];
assign andAB[3]=opA[3]&opB[3];
// Multiplexer to select between arithmetic and logical results.

four_bit_mux fmux(Result,andAB,temp_results,ctrl[0]);
// Determine the overflow flag for arithmetic operations only.
assign Overflow = CrO & ctrl[0];
endmodule
```

Conclusion :

In this lab, I utilized Verilog to simulate a 4-bit Arithmetic Logic Unit (ALU) capable of performing binary addition and subtraction while also detecting overflow conditions. Through this process, I gained a deeper understanding of Verilog programming and improved my proficiency in calling functions within the Vivado design environment.

Post-lab Deliverables:

Q1,2) My results show the code and the screenshots of the waveform

Q3) Through the lab I can examine that the test bench is 1-bit, 2:1 multiplexer (MUX) is outputting the waveform that was discussed in Verilog . The code that I provided has comments that showed how each section is implemented but to go more in depth The 1-bit, 2:1 multiplexer (MUX) using structural Verilog. It declares a module named two_one_mux with output Y and inputs A, B, and S. Internal wires notS, andA, and andB are declared. Gates are instantiated to implement the MUX operation: a NOT gate negates S, two AND gates select inputs A and B based on the value of S, and an OR gate combines the selected inputs to produce the output Y.

Q.4) The code does not test all possible combinations of A and B for each bit position, nor does it test all possible values of S. In order to test all possible input combinations for a 4-bit, 2:1 MUX,you would need to create test cases that cover all possible combinations of A, B, and S.

Q5)Breadboarding offers hands-on experience and immediate feedback through physical assembly but is limited in scalability and prone to errors. Circuit simulation allows for detailed analysis, complex designs, and easy iteration but requires proficiency in simulation software and may not capture real-world effects accurately. HDLs provide hierarchical design, reusability, and simulation readiness, making them suitable for complex designs, while schematics offer intuitive understanding but may lack scalability and require additional effort for visualization. The choice between breadboarding and simulation depends on the project's complexity and the need for hands-on experimentation versus detailed analysis. Similarly, the preference for HDLs or schematics hinges on factors such as design goals, resources, and the desired level of abstraction, with HDLs favored for scalable, simulation-ready designs and schematics for intuitive understanding in simpler contexts.

Q6) Structural and dataflow abstraction represent distinct approaches in digital design and computer science. Structural abstraction delves into the physical connections and interactions among components within a system, emphasizing detailed descriptions of components and their interconnections, often through hierarchical block diagrams or schematic representations. It proves invaluable when designing hardware components or digital circuits where precise physical connections are paramount, as seen in FPGA or ASIC design. Conversely, dataflow abstraction focuses on how data flows through a system and the operations performed on that data, typically represented using dataflow diagrams or flowcharts to illustrate data dependencies and operations. This approach shines in algorithmic design and software systems where the emphasis lies on data processing rather than physical connections, and it's particularly adept at modeling parallel and concurrent systems. While structural abstraction suits detailed hardware design, dataflow abstraction excels in algorithmic and high-level system design, with both often employed together across different stages of the design process to achieve comprehensive system development.

Student Feedback

Overall this lab was a very good introduction towards Verilog and I really did learn about how the program works and how to transition the circuit towards the code. I had no issues with this lab. It was very enjoyable and insightful to learn about this new subject.