CPE 324-03: Advanced Digital Logic Design Laboratory

Lab 2

Design Capture and FPGA-based Rapid Prototyping using the DE2-115 Platform

Submitted by: Hannah Kelley

Date of Experiment: January 13, 2026

Report Deadline: January 20, 2026

Demonstration Deadline: January 20, 2026

1 Introduction

The purpose of this laboratory project is to gain experience using the Intel FPGA Quartus Prime Computer-Aided Design (CAD) software to implement a simple digital design: an 8-bit adder/subtractor. A hierarchical schematic version of the circuit is first created using traditional schematic capture techniques. The same design is then reimplemented in Verilog using structural and behavioral description styles. Each version of the design is compiled and downloaded to the Cyclone IV E FPGA on the DE2-115 development board. The functionality of each implementation is verified using the board's slide switches as inputs and LEDs as outputs. This lab project demonstrates the process of capturing, compiling, and programming digital designs using the Quartus CAD tool suite.

2 Experiment Description

2.1 Theory

2.1.1 Hierarchical Circuit Design

Hierarchical circuit design is a method of developing digital systems by dividing a complex circuit into smaller, simpler functional blocks called modules. Each module is designed and tested individually, then interconnected to form higher-level subsystems, which together create the complete system. This approach improves design organization, readability, and reliability by allowing each module to be verified independently before integration. In this laboratory, the 8-bit adder/subtractor is constructed hierarchically by combining basic logic components and full-adder modules into larger functional blocks. The complete circuit is formed by interconnecting these lower-level modules to produce the final arithmetic unit. Hierarchical design also simplifies debugging and future modification, as changes can be made to individual modules without affecting the entire design.

2.1.2 Structural Design

Structural circuit design in Verilog describes a digital circuit by explicitly defining how individual components are interconnected. This style closely resembles a schematic representation, where basic modules such as logic gates and full adders are instantiated and wired together to form a complete system. Each module represents a physical hardware block, and signals are used to connect these blocks. Structural design emphasizes the actual architecture of the circuit and provides a clear representation of how the hardware is constructed.

2.1.3 Behavioral Circuit Design

Behavioral Verilog describes a digital circuit based on its functional behavior rather than its physical structure. Instead of defining individual components and connections, this style uses high-level statements and procedural blocks to specify how outputs respond to changes in inputs. Arithmetic operations, conditional statements, and control logic are used to model the desired behavior of the circuit. Behavioral design allows for faster development and easier modification while still synthesizing into equivalent hardware functionality.

2.2 Part 1 Procedure - Hierarchical Schematic Capture Design

For this portion of the experiment, an 8-bit adder/subtractor was implemented on the FPGA using a bottom-up design approach, where basic logic elements were first created and then combined to form higher-level functional blocks.

The first elements constructed were a 3-input XOR gate and a 2-1 multiplexer. Both components were needed to make a 1-bit full adder. The schematics for the 3-input XOR gate and the 2-1 multiplexer can be seen in Figures 1 and 2. Quartus Prime has a built-in 3-input XOR gate, but after testing the gate did not work as intended. As a result, the new 3-input XOR gate was required. Once the XOR gate and multiplexer were created, they were used to create a full adder. The full-adder schematic can be found in Figure 3. Next, four full-adders were combined in series to create a 4-bit adder/subtractor. This 4-bit adder/subtractor can be found in Figure 4. This design made use of buses to simplify the inputs and outputs of the entire 4-bit module. The final step in creating this design was combining two of the 4-bit modules into a single 8-bit adder/subtractor module. Again, this design made use of buses to simplify the inputs and outputs. The 8-bit module can be found in Figure 5.

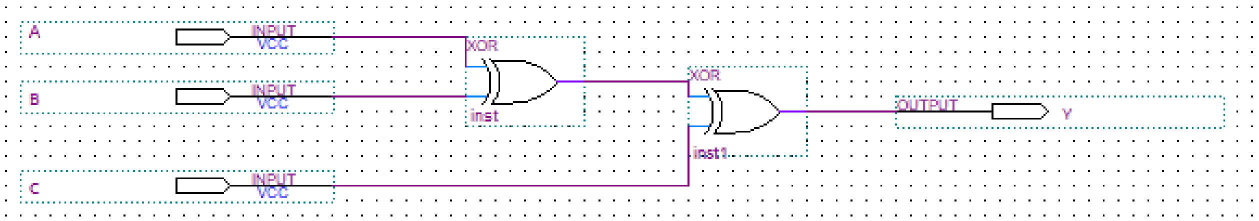


Figure 1 – 3-Input XOR Gate Schematic

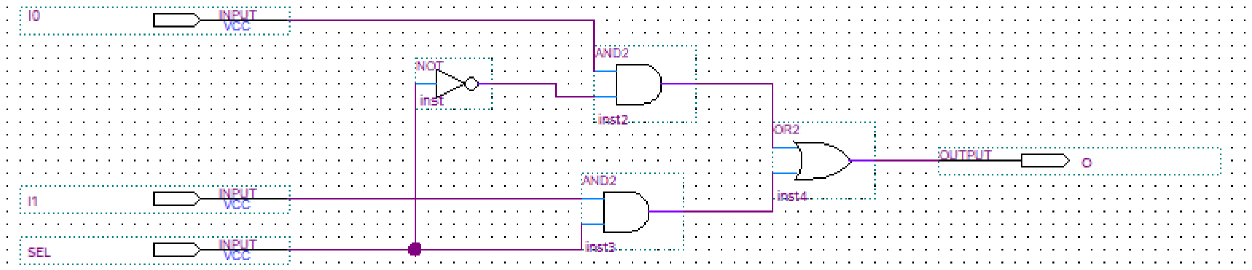


Figure 2 – 2-1 Multiplexer Schematic

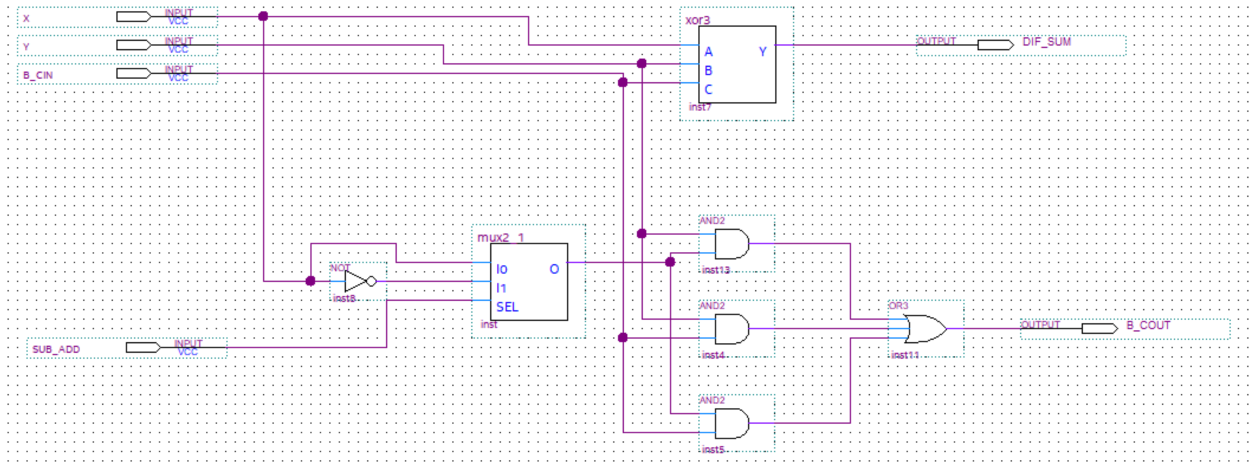


Figure 3 – Full-Adder Schematic

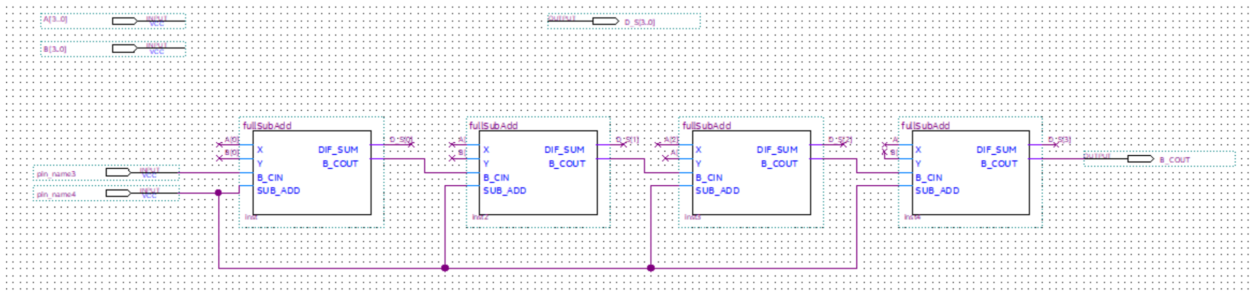


Figure 4 – 4-Bit Adder/Subtractor

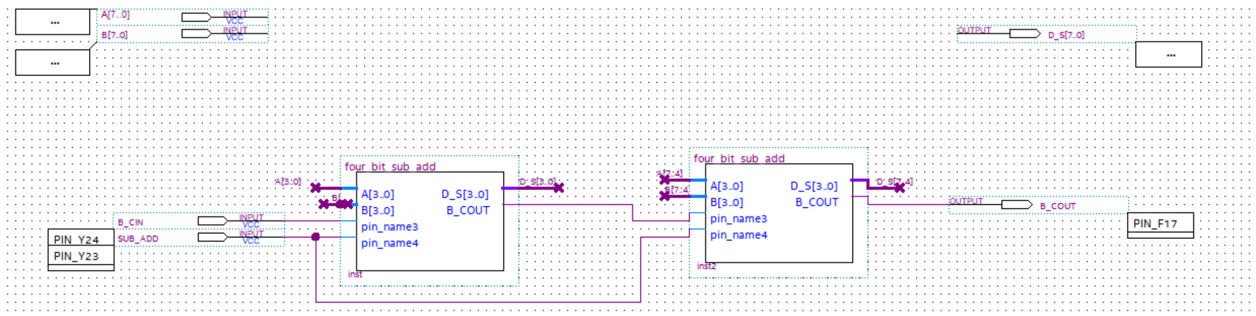
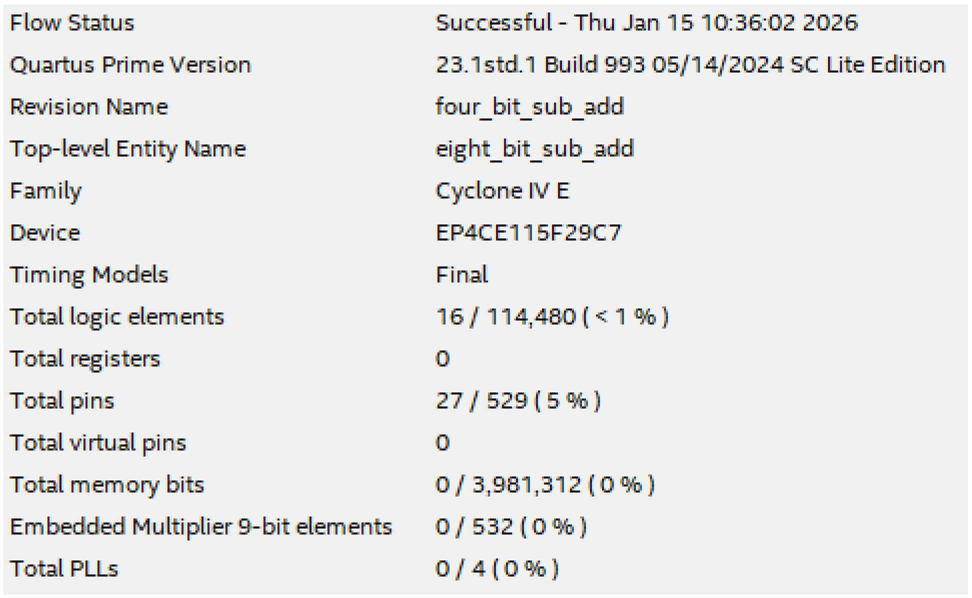


Figure 5 – 8-Bit Adder/Subtractor

After the final module was created, it was compiled and implemented on the lab FPGA using the pin-map given in the lab document. During compilation, the design was observed to use 16 logic elements. A full read-out of compilation statistics can be found in Figure 6.



Flow Status	Successful - Thu Jan 15 10:36:02 2026
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	four_bit_sub_add
Top-level Entity Name	eight_bit_sub_add
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	16 / 114,480 (< 1 %)
Total registers	0
Total pins	27 / 529 (5 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

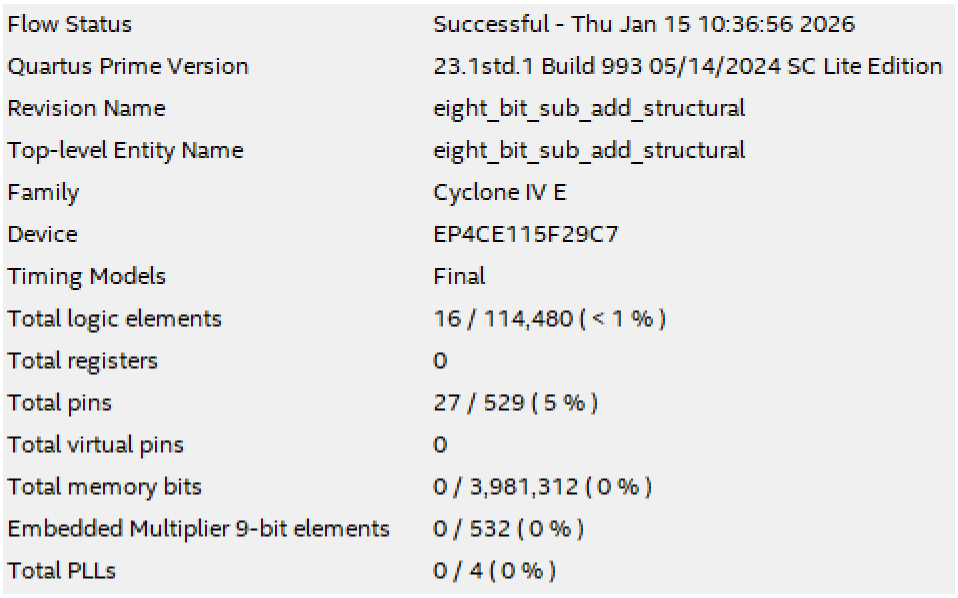
Figure 6 – Hierarchical Design Compilation Statistics

After implementation, the design was tested using the test vectors discussed in detail in the results section of this report. The design was then demonstrated to the TA on January 15th, 2026 at 9:57 A.M.

2.3 Part 2 Procedure - Structural Verilog HDL Design

The structural design approach implements the 8-bit adder/subtractor in Verilog by describing the circuit as an interconnection of smaller functional modules. The structural Verilog code for the 8-bit adder/subtractor was provided as part of the laboratory assignment and implemented directly in Quartus Prime. This design uses the structural design method, in which the circuit is described as an explicit interconnection of hardware components rather than by functional behavior. The top-level module, eight_bit_sub_add, connects two four-bit subtractor/adder modules in series, allowing the carry or borrow from the lower four bits to propagate to the upper four bits. Each four-bit subtractor/adder module is constructed from individual full subtractor/adder blocks, which are themselves implemented using basic logic gates and a 2-to-1 multiplexer. The SUB_ADD control signal determines whether the circuit performs addition or subtraction by controlling the selection logic within the full subtractor/adder. This approach closely mirrors a gate-level schematic and provides a clear representation of how the arithmetic operation is physically realized in hardware.

The given code was compiled implemented on the lab FPGA and tested. During compilation, the design was observed to use 16 logic elements. A full read-out of compilation statistics can be found in Figure 7. After implementation on the lab FPGA, the design was tested. The test vectors used and results are discussed in more detail in the results section of this report. After testing, the design was then demonstrated to the TA on January 15th, 2026 at 10:17 A.M.



Flow Status	Successful - Thu Jan 15 10:36:56 2026
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	eight_bit_sub_add_structural
Top-level Entity Name	eight_bit_sub_add_structural
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	16 / 114,480 (< 1 %)
Total registers	0
Total pins	27 / 529 (5 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

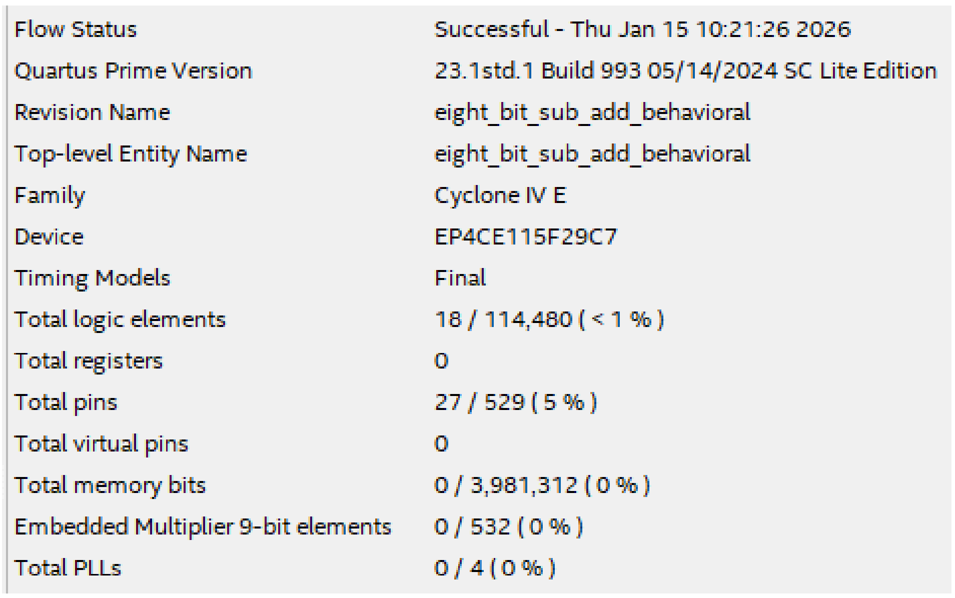
Figure 7 –Structural Design Compilation Statistics

2.4 Part 3 - Behavioral Verilog HDL Design

The behavioral Verilog design implements the same 8-bit two's complement adder/subtractor previously realized using hierarchical schematic capture and structural design methods. Again, the code was given in the lab manuals. In contrast to the structural approach, the behavioral model describes the circuit in terms of its arithmetic functionality rather than explicit hardware interconnections. Addition and subtraction operations are defined using high-level Verilog constructs within a single top-level module.

The design uses an always block to evaluate the inputs and control signals, with the SUB_ADD signal selecting between addition and subtraction. A 9-bit internal register is used to capture the full result of the operation, allowing the most significant bit to represent the carry or borrow output. The lower eight bits are assigned to the result output, while the ninth bit is used to generate the B_COUT signal.

The given code was compiled implemented on the lab FPGA and tested. During compilation, the design was observed to use 18 logic elements. A full read-out of compilation statistics can be found in Figure 8. After implementation on the lab FPGA, the design was tested. The test vectors used and results are discussed in further detail in the results section of this report. After testing, the design was then demonstrated to the TA on January 15th, 2026 at 10:29 A.M.



Flow Status	Successful - Thu Jan 15 10:21:26 2026
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	eight_bit_sub_add_behavioral
Top-level Entity Name	eight_bit_sub_add_behavioral
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	18 / 114,480 (< 1 %)
Total registers	0
Total pins	27 / 529 (5 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 8 – Behavioral Design Compilation Statistics

3 Results

All three designs—hierarchical, structural, and behavioral—were evaluated using the same set of test vectors and produced identical results. This outcome is expected, as each design represents a different implementation approach for the same underlying functionality within Quartus Prime. The test vectors and results are shown in Table 1.

Test Case	A	B	B_CIN	Result	B_COUT
Add unsigned inputs with no carry	0000_0101	0000_1010	0	0000_1111	0
Subtract unsigned inputs with no borrow	0000_1110	0000_0110	0	0000_1000	0
Add unsigned inputs with carry	0000_0101	0000_1010	1	0001_0000	0
Subtract unsigned inputs with borrow	0000_1110	0000_0110	1	0000_0111	0
Add unsigned inputs with internal carry propagating to B_COUT	1111_1111	0000_0001	0	0000_0000	1
Subtract unsigned inputs with internal borrow propagating to B_COUT	0000_1110	0000_1111	0	1111_1111	1
Add inputs that cause an overflow into B_COUT	1111_1111	0000_0000	1	0000_0000	1

Subtract two signed positive numbers that generate a negative result	0000_0000	0000_0001	0	1111_1111	1

Table 3 – Behavioral Design Test Vectors and Results

The identical results obtained across the hierarchical, structural, and behavioral implementations confirm the functional equivalence of the three design methods. Although each approach differs in how the circuit is described and implemented within Quartus Prime, all three represent the same 8-bit adder/subtractor logic and therefore produce the same outputs for a given set of inputs. Using a common set of test vectors ensures a consistent basis for comparison and verifies that the arithmetic operations, including carry and borrow propagation, are correctly implemented in each design. These results demonstrate that different design methodologies can be used interchangeably to describe the same digital system while maintaining correct functionality.

In addition to the test vectors, several other ideas were considered after the designs were implemented:

1. **For your three versions of this design, what kind of compilation errors or warnings did you encounter? How did you correct the errors and how did you determine which warnings were important and which ones could be ignored?**
   During implementation of the hierarchical, structural, and behavioral designs, common compilation issues included disconnected wires, mismatched signal names, and warnings indicating that no clock was defined. Wiring and naming errors were corrected by verifying signal connections and ensuring consistency across modules. Clock-related warnings were ignored as the designs did not use a clock. Quartus Prime compilation messages and schematic viewers were used to identify and resolve functional errors while safely ignoring non-impactful warnings.
2. **How does hierarchical schematic capture and hierarchical structural Verilog HDL design techniques help one manage the complexity of the design process?**
   Hierarchical schematic capture and structural Verilog techniques help manage design complexity by breaking large systems into smaller, manageable modules. Designing and verifying each block independently makes errors easier to isolate and correct. This modular approach improves readability, simplifies debugging, and allows components to be reused or modified without affecting the entire system. Hierarchical design is particularly useful for larger circuits, where flat designs are difficult to understand and maintain.
3. **Do you think the stimulus you used is good enough to verify the full functionality of your design? Why or why not? Would this stimulus be good**

enough to be used to automatically test the design for manufacturing defects? Why or why not?
The stimulus used in this experiment is sufficient to verify the functional correctness of the 8-bit adder/subtractor, covering addition and subtraction, carry and borrow inputs, and propagation through all bit positions. However, it is not adequate for automated manufacturing tests, which require exhaustive or statistically significant patterns to detect physical defects such as stuck-at faults or timing issues, both of which could be ignored for this experiment.

4. **Were there any differences between the amount of FPGA resources that were used for each design. If so how did they differ? If not do you believe that the designs are implemented in the same manner by the Quartus Prime synthesis tool?**
   Minor differences in FPGA resource utilization were observed between the hierarchical, structural, and behavioral designs due to the way each description style is interpreted by the synthesis tool. Behavioral designs often allow the synthesis tool more freedom to optimize logic, potentially resulting in slightly different resource usage compared to structural designs, which explicitly define hardware interconnections. Since resource usage is similar across all versions, this suggests that Quartus Prime synthesized the designs into functionally equivalent hardware implementations despite the differing description methods.
5. **Describe the design trade-offs associated with the structural and behavioral Verilog versions in terms of ease of design entry and the ability to control the low-level attributes of the design (such as what type of subtractor/adder is created).**
   The structural Verilog design allows precise control over low-level circuit implementation, specifying exactly how arithmetic operations are built from logic components. This is useful for controlling hardware architecture or timing but is more time-consuming and prone to wiring errors. Behavioral Verilog, in contrast, is faster and easier to implement, focusing on functionality rather than structure, though it offers less control over the resulting hardware. The choice between the two depends on design goals, balancing development ease against the need for low-level control.

4 Conclusion

In this laboratory, an 8-bit adder/subtractor was implemented using hierarchical schematic capture, structural Verilog, and behavioral Verilog design methods. All three approaches produced identical results for the same set of test vectors, demonstrating their functional equivalence.

The experiment highlighted the trade-offs between design methods: structural Verilog provides precise control over hardware, behavioral Verilog simplifies implementation, and hierarchical schematic capture abstracts design complexity. Overall, the project reinforced the principles of digital design, verification, and FPGA implementation, and demonstrated how different design methodologies can be used to achieve the same functional outcome.