BCD to Gray Code Converter and Gray Code to BCD Converter

Submitted by :

- Name:GAUTHAM P NAIR
  Roll No: 20
  KTU ID: MDL23EC052
- Name: SHRAVAN KRISHNA S
  Roll No: 52
  KTU ID: MDL23EC111

INDEX

\

1. Introduction

1.1 Objective

To design and implement a BCD (Binary-Coded Decimal) to Gray Code converter and the reverse Gray Code to BCD converter using Verilog, and to deploy it on a Spartan-7 FPGA (Boolean board).

1.2 Scope

The scope of this project covers encoding and decoding in digital systems. Gray code is commonly used in error detection to reduce the possibility of erroneous transitions, making it crucial in various communication systems.

1.3 Relevance

Understanding Gray code and its conversions is important in fields like digital communication, computer engineering, and embedded systems, where data accuracy is vital during transitions.

1.4 Literature Review

Gray code was first introduced by Frank Gray and is used in various fields like rotary encoders and digital communication to minimise errors during state transitions. Various FPGA implementations for code converters have been researched to optimise processing speeds and accuracy in real-time systems.

2. Design

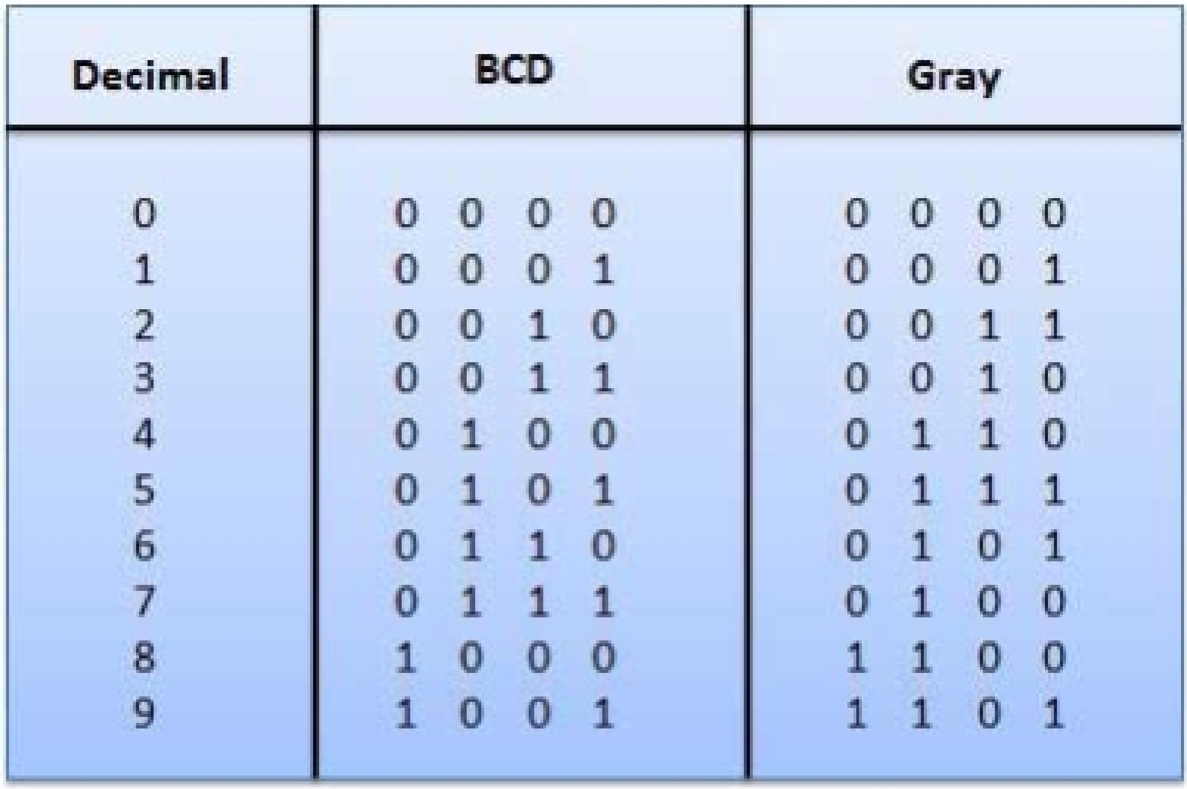
2.1 Design Description

The system consists of two primary modules:

1. **BCD to Gray Code Converter:** Takes a 4-bit BCD input (binary representation of decimal numbers 0 to 9) and converts it to the corresponding 4-bit Gray Code.
2. **Gray Code to BCD Converter:** Accepts a 4-bit Gray Code input and converts it back to BCD.

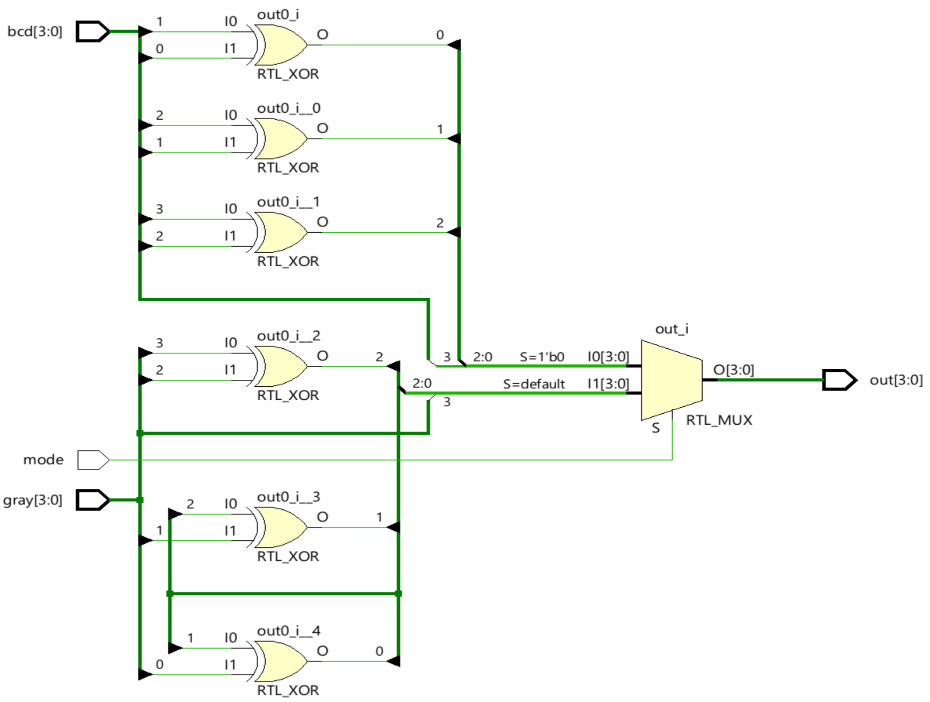
A mode selection signal is used to choose between the two operations:

- Mode = 0: BCD to Gray Code conversion.
- Mode = 1: Gray Code to BCD conversion.



Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

2.2 Circuit Diagram



2.3 Hardware and Software Requirements

- **Hardware:**
  - FPGA Development Board: Xilinx XC7S50-CSGA324-1 Spartan-7 (Boolean Board)
  - Built-in switches for input selection and LEDs for output display
  - USB cable for power and programming
- **Software:**
  - Verilog HDL for code development
  - Icarus Verilog for simulation
  - GTKWave for waveform analysis
  - Vivado for synthesis and programming

3. Results

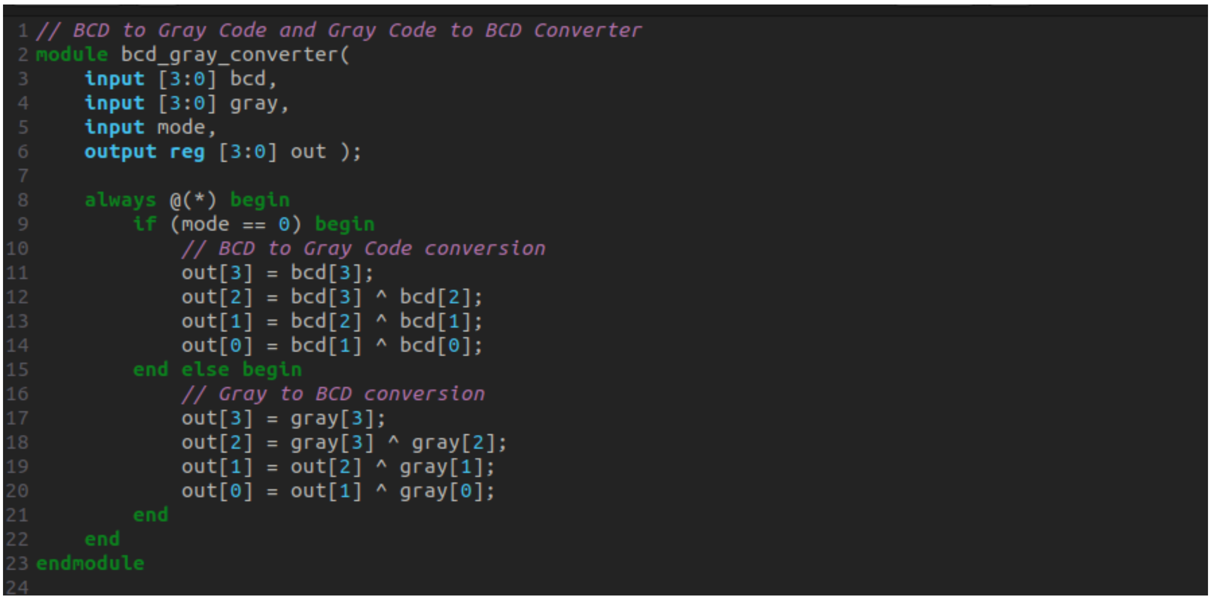
3.1 Experiments and Simulations

- Simulated the Verilog code using Icarus Verilog to verify correct functionality.
- Observed waveform outputs on GTKWave to ensure BCD-to-Gray and Gray-to-BCD conversions were accurate.

3.2 FPGA Implementation

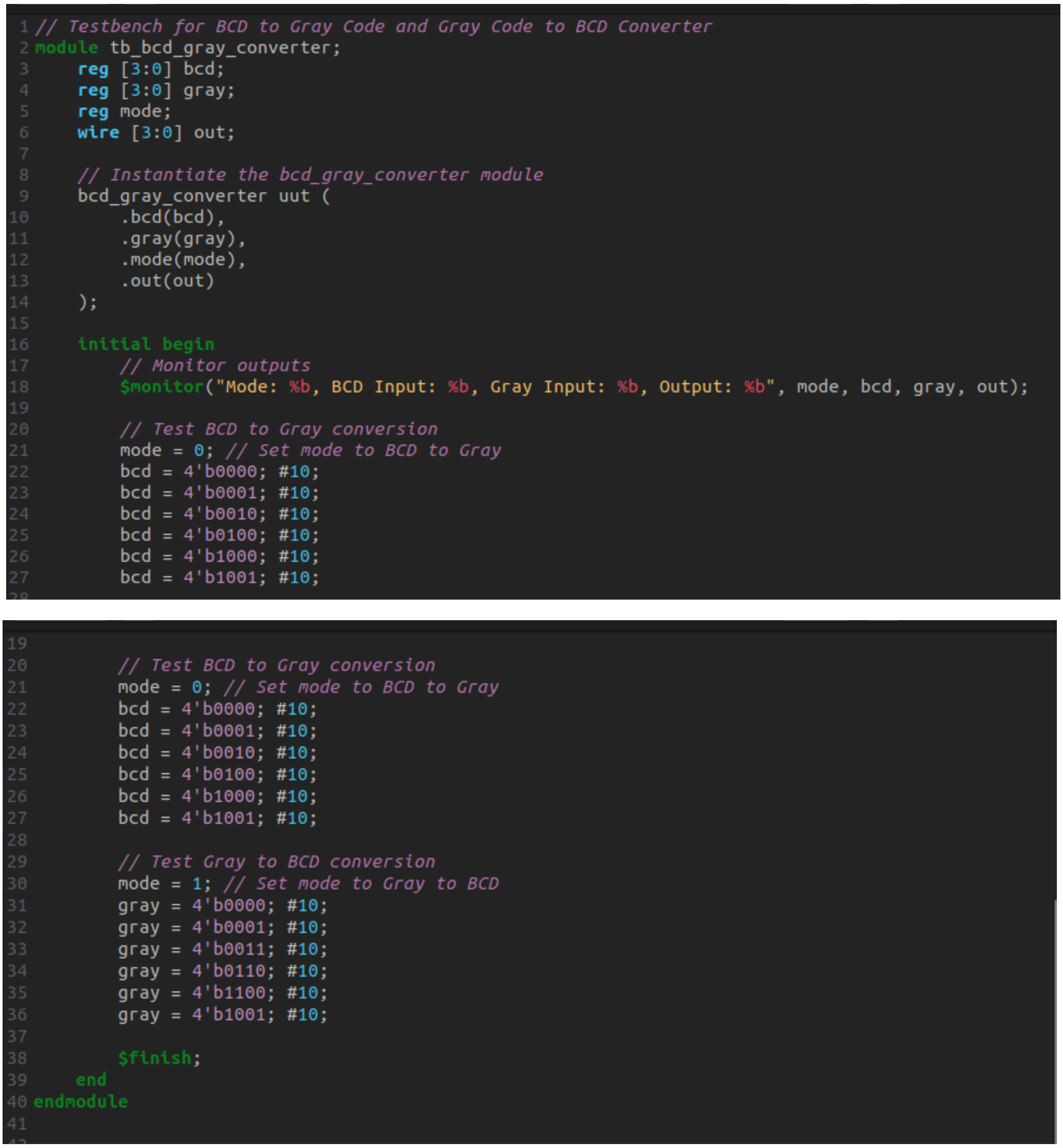
- Implemented the design on the Boolean FPGA board.
- Tested the system by setting different 4-bit inputs on switches for BCD and observing the Gray code on the LEDs, and vice versa.

3.3 Screenshots



```
// BCD to Gray Code and Gray Code to BCD Converter
module bcd_gray_converter(
    input [3:0] bcd,
    input [3:0] gray,
    input mode,
    output reg [3:0] out );

    always @(*) begin
        if (mode == 0) begin
            // BCD to Gray Code conversion
            out[3] = bcd[3];
            out[2] = bcd[3] ^ bcd[2];
            out[1] = bcd[2] ^ bcd[1];
            out[0] = bcd[1] ^ bcd[0];
        end else begin
            // Gray to BCD conversion
            out[3] = gray[3];
            out[2] = gray[3] ^ gray[2];
            out[1] = out[2] ^ gray[1];
            out[0] = out[1] ^ gray[0];
        end
    end
endmodule
```

Icarus verilog model

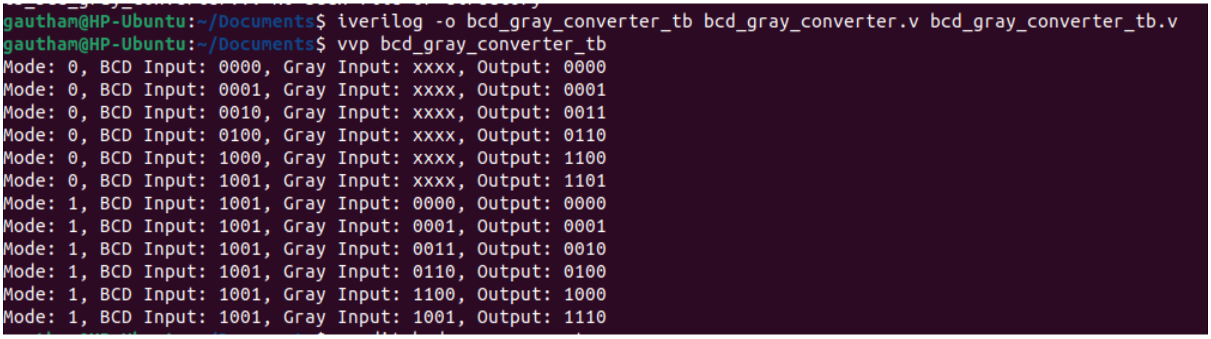


```verilog
// Testbench for BCD to Gray Code and Gray Code to BCD Converter
module tb_bcd_gray_converter;
    reg [3:0] bcd;
    reg [3:0] gray;
    reg mode;
    wire [3:0] out;

    // Instantiate the bcd_gray_converter module
    bcd_gray_converter uut (
        .bcd(bcd),
        .gray(gray),
        .mode(mode),
        .out(out)
    );

    initial begin
        // Monitor outputs
        $monitor("Mode: %b, BCD Input: %b, Gray Input: %b, Output: %b", mode, bcd, gray, out);

        // Test BCD to Gray conversion
        mode = 0; // Set mode to BCD to Gray
        bcd = 4'b0000; #10;
        bcd = 4'b0001; #10;
        bcd = 4'b0010; #10;
        bcd = 4'b0100; #10;
        bcd = 4'b1000; #10;
        bcd = 4'b1001; #10;
```

```verilog

        // Test BCD to Gray conversion
        mode = 0; // Set mode to BCD to Gray
        bcd = 4'b0000; #10;
        bcd = 4'b0001; #10;
        bcd = 4'b0010; #10;
        bcd = 4'b0100; #10;
        bcd = 4'b1000; #10;
        bcd = 4'b1001; #10;
```

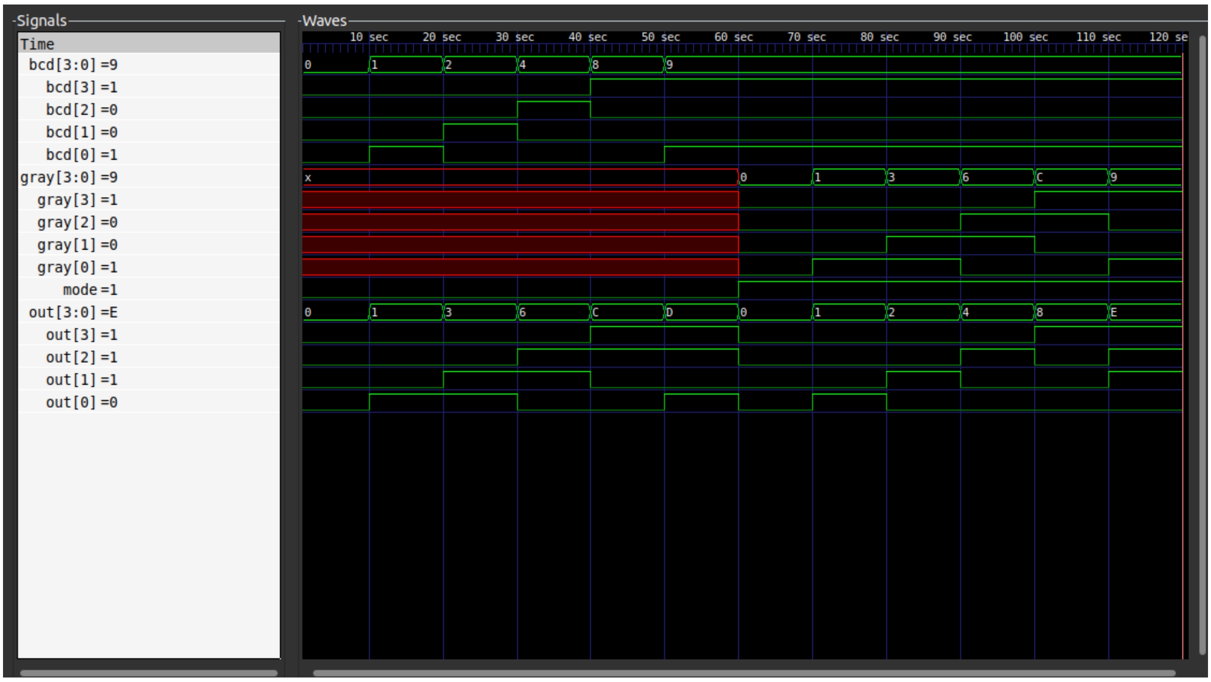
```verilog

        // Test Gray to BCD conversion
        mode = 1; // Set mode to Gray to BCD
        gray = 4'b0000; #10;
        gray = 4'b0001; #10;
        gray = 4'b0011; #10;
        gray = 4'b0110; #10;
        gray = 4'b1100; #10;
        gray = 4'b1001; #10;

        $finish;
    end
endmodule

```

Icarus verilog testbench

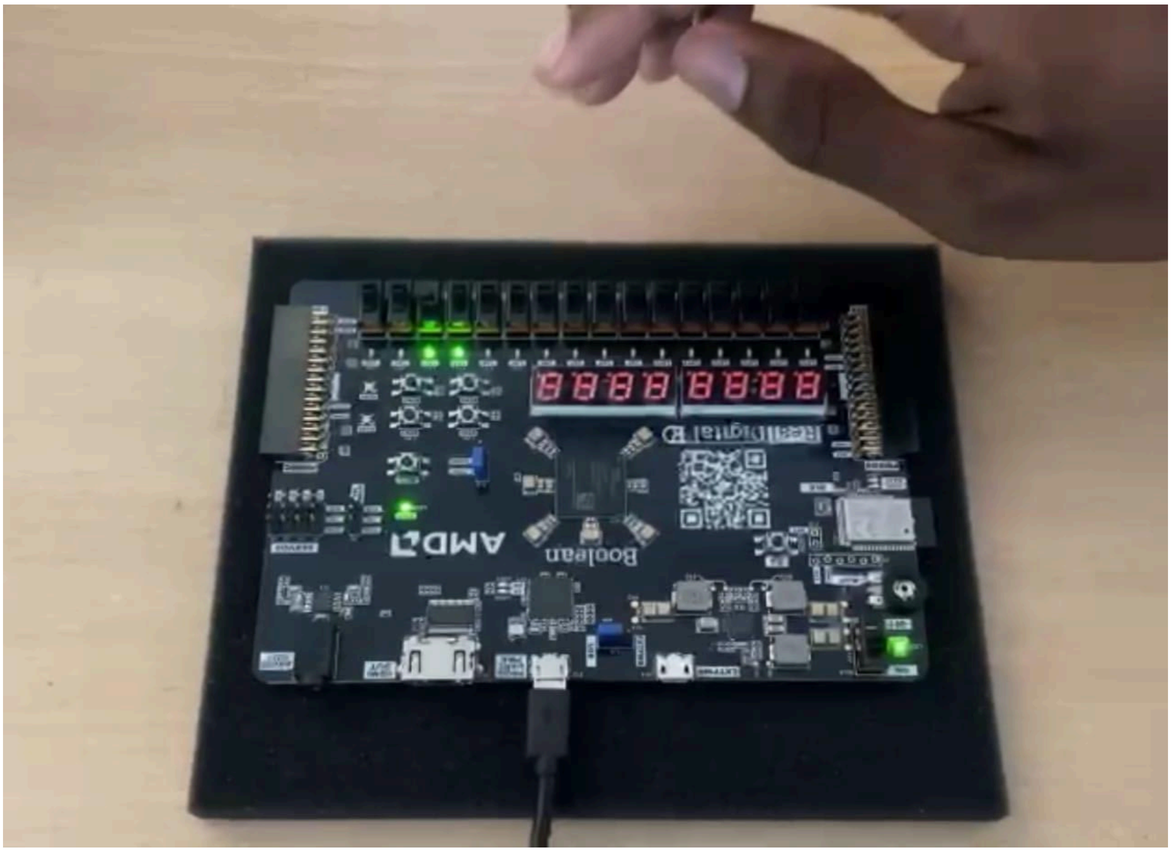


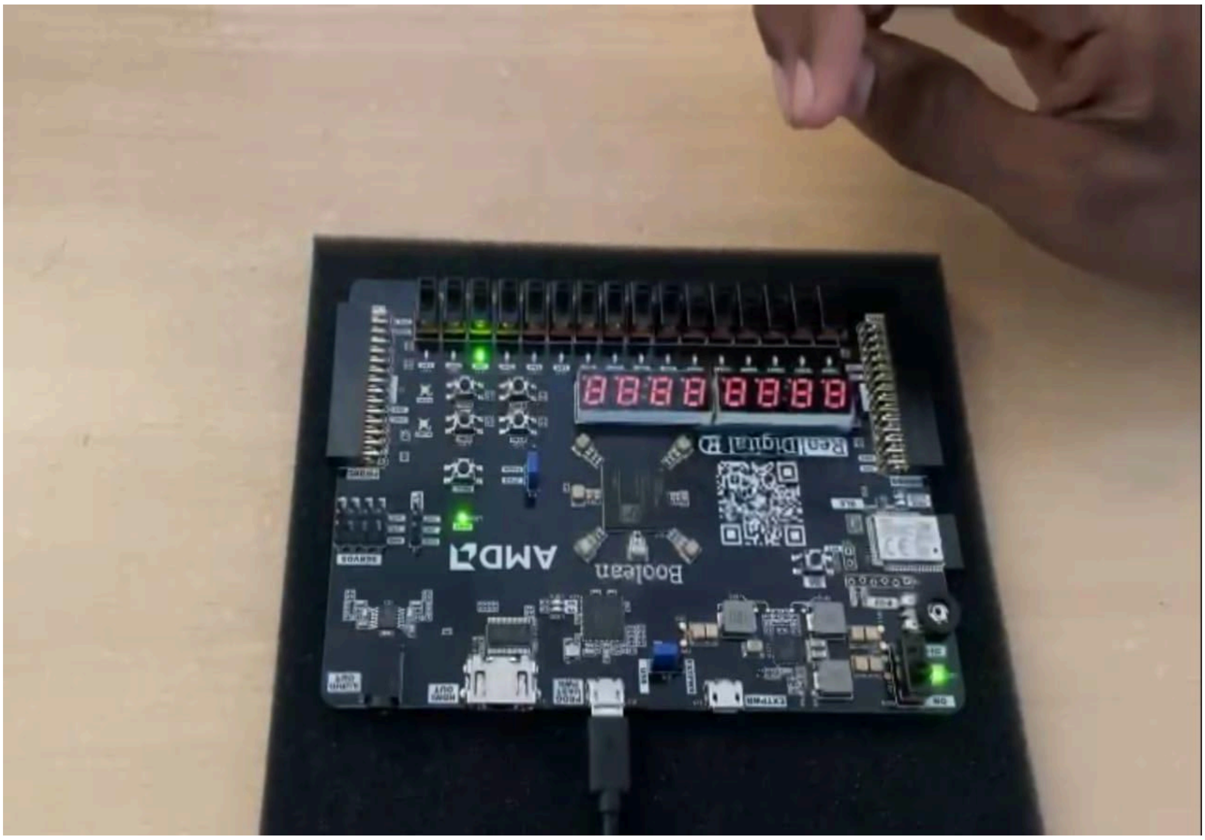
```
gautham@HP-Ubuntu:~/Documents$ iverilog -o bcd_gray_converter_tb bcd_gray_converter.v bcd_gray_converter_tb.v
gautham@HP-Ubuntu:~/Documents$ vvp bcd_gray_converter_tb
Mode: 0, BCD Input: 0000, Gray Input: xxxx, Output: 0000
Mode: 0, BCD Input: 0001, Gray Input: xxxx, Output: 0001
Mode: 0, BCD Input: 0010, Gray Input: xxxx, Output: 0011
Mode: 0, BCD Input: 0100, Gray Input: xxxx, Output: 0110
Mode: 0, BCD Input: 1000, Gray Input: xxxx, Output: 1100
Mode: 0, BCD Input: 1001, Gray Input: xxxx, Output: 1101
Mode: 1, BCD Input: 1001, Gray Input: 0000, Output: 0000
Mode: 1, BCD Input: 1001, Gray Input: 0001, Output: 0001
Mode: 1, BCD Input: 1001, Gray Input: 0011, Output: 0010
Mode: 1, BCD Input: 1001, Gray Input: 0110, Output: 0100
Mode: 1, BCD Input: 1001, Gray Input: 1100, Output: 1000
Mode: 1, BCD Input: 1001, Gray Input: 1001, Output: 1110
```

Icarus verilog output



GTK wave simulation output





FPGA Implementation

4. Conclusion and Future Scope

4.1 Conclusion

The project achieved the successful design and implementation of a BCD to Gray Code and Gray to BCD converter on an FPGA. The Verilog code demonstrated reliable performance both in simulation and when deployed on hardware. This converter could be a building block in larger digital systems that require precise encoding and decoding.

4.2 Future Scope

Potential extensions of this project could include:

- Handling Larger Inputs: Expanding the design to convert larger binary values (e.g., 8-bit or 16-bit).

- Error Detection Mechanisms: Adding error detection or correction mechanisms for enhanced robustness.
- Exploring Other Encoding Schemes: Investigating the benefits and trade-offs of other encoding schemes like excess-3 or binary-coded Gray.
- Applications in Embedded Systems: Integrating the converter into real-time embedded systems to evaluate performance.

5. References

- Xilinx, "*Vivado Design Suite User Guide: Logic Simulation*," UG937 (v2020.1), May 2020. [Online]. Available at: https://www.xilinx.com/support/documentation/user_guides/ug937-vivado-simulation.pdf
- M. Morris Mano, *Digital Logic and Computer Design*, 2nd ed. Prentice Hall, 2000.
- Stephen Williams, "*Icarus Verilog: A Verilog Simulation and Synthesis Tool*," 2001. Available at Icarus Verilog.
- EDA Playground, "Online EDA Tools for Verilog and VHDL," [Online]. Available at: https://www.edaplayground.com/.