Combinational Logic Design: Part 2: 8-bit Binary to Gray Code Converter

Author Adil R.

May 2025

1 Introduction

Gray code is a binary numeral system where two successive values differ in only one bit. This property makes it valuable in various digital applications including rotary encoders, Karnaugh maps, and error correction systems. This document describes the design and implementation of an 8-bit binary to Gray code converter using Verilog HDL.

2 Design Description

2.1 Specifications

The 8-bit binary to Gray code converter has the following characteristics:

- Input: 8-bit binary (B[7:0])
- Output: 8-bit Gray code (G[7:0])

2.2 Conversion Algorithm

The conversion from binary to Gray code follows these rules:

- The most significant bit (MSB) remains the same: G[7] = B[7]
- Each subsequent bit is the XOR of adjacent binary bits:

$$G[i] = B[i+1] \oplus B[i] \quad \text{for} \quad i = 6 \text{ down to } 0$$

3 Verilog Implementation

The implementation uses XOR operations for efficient conversion.

```
`timescale 1ns / 1ps

module binary_to_gray(
    input  [7:0] bin,   // Binary input
    output [7:0] gray   // Gray code output
);

    assign gray[7] = bin[7];  // MSB stays same
    assign gray[6:0] = bin[7:1] ^ bin[6:0];  // XOR adjacent bits

endmodule
```

Listing 1: 8-bit Binary to Gray Code Converter

4 Test Bench Design

The test bench thoroughly verifies the design through:

- Edge case testing
- Known pattern verification
- Random input testing
- Automatic result checking

```
`timescale 1ns / 1ps

module tb_binary_to_gray;
    reg [7:0] bin;     // Binary input
    wire [7:0] gray;  // Gray output

    // Instantiate DUT
    binary_to_gray dut (
        .bin(bin),
        .gray(gray)
    );

    // Test stimulus & verification
    initial begin
        $display("Starting testbench...");

        // Test all zeros
        bin = 8'b00000000; #10;
        check_gray(8'b00000000);

        // Test single bit transitions
        for (int i = 0; i < 8; i++) begin
            bin = (1 << i); #10;
            check_gray(bin ^ (bin >> 1));
        end

        // Test all ones
        bin = 8'b11111111; #10;
        check_gray(8'b10000000);

        // Test random patterns
        repeat(100) begin
            bin = $random; #10;
            $display("Binary: %b     Gray: %b", bin, gray);
            check_gray(bin ^ (bin >> 1));
        end

        $display("All tests passed successfully!");
        $finish;
    end

    // Automatic verification task
    task check_gray(input [7:0] expected_gray);
        if (gray !== expected_gray) begin
            $error("Error: Binary=%b     Expected Gray=%b, Got Gray=%b",
                   bin, expected_gray, gray);
            $finish;
        end
    endtask

endmodule
```

Listing 2: Test Bench for Binary to Gray Converter

5 Conclusion

The implemented 8-bit binary to Gray code converter:

- Correctly implements the conversion algorithm
- Passes all test cases including edge conditions
- Can be easily scaled to different bit widths

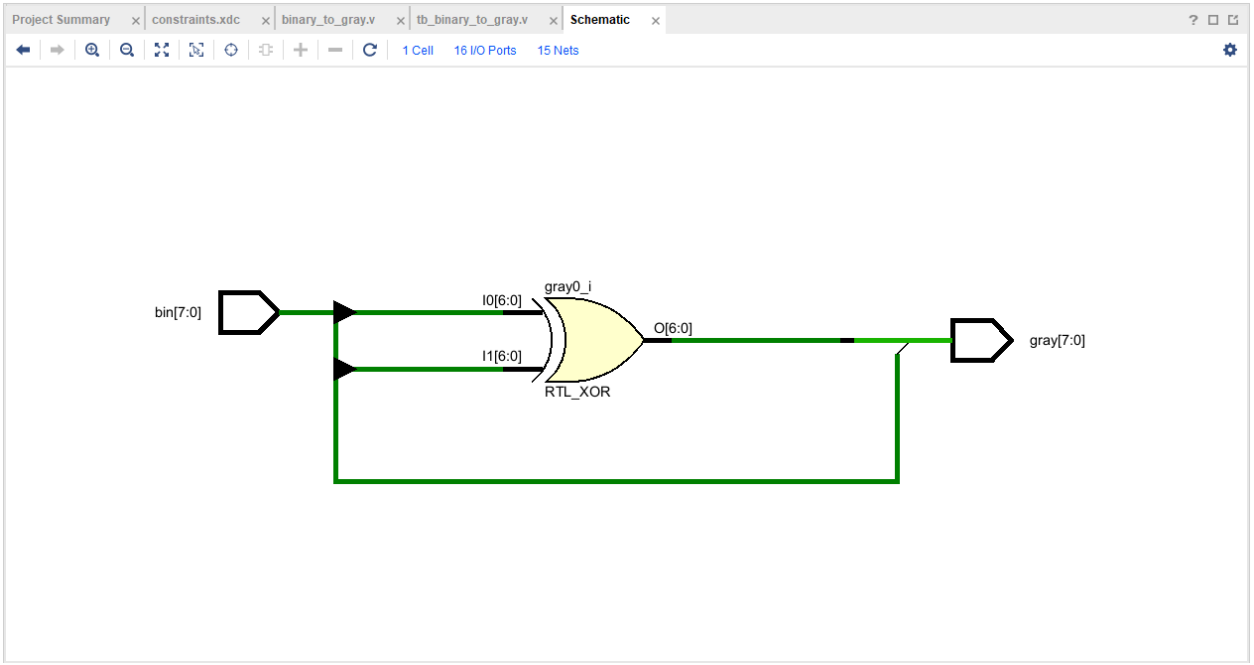

Figure 1: schematic

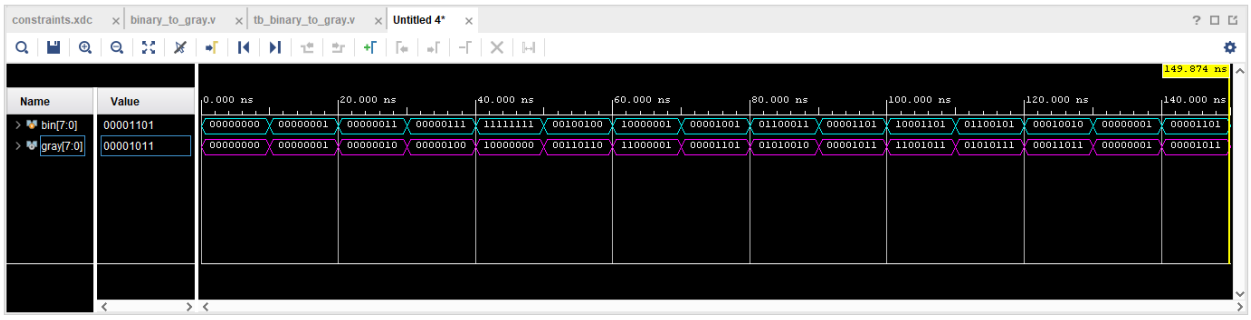

Figure 2: waveform