REPORT

Design of Hamming Type Code using FPGA in Verilog

By: R Madhupreetha

Abstract

- This project demonstrates the design and implementation of a Hamming Code encoder and decoder using Verilog, targeted for FPGA deployment.
- Hamming Codes are used for error detection and correction in digital communication systems, making them essential for reliable data transmission over noisy channels.
- The project showcases the functionality of both encoding and decoding systems, ensuring single-bit error detection and correction.
- The implementation is tested on an FPGA using Xilinx tools, making it suitable for real-world applications in wireless communication.

Introduction

Hamming Code

- Hamming code is an error-correcting code used to ensure data accuracy during transmission or storage.
- Hamming code detects and corrects the errors that can occur when the data is moved or stored from the sender to the receiver.
- This simple and effective method helps improve the reliability of communication systems and digital storage.
- It adds extra bits to the original data, allowing the system to detect and correct single-bit errors.

Algorithm of Hamming Code

Hamming Code is simply the use of extra parity bits to allow the identification of an error.

Step 1: Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).

Step 2: All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).

Step 3: All the other bit positions are marked as data bits.

Step 4: Each data bit is included in a unique set of parity bits, as determined its bit position in binary form:

- Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).
- Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).
- Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).
- Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).
- In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

Step 5: Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Objective

The objective of this project is to design and implement a Hamming Code encoder and decoder on an FPGA using Verilog. The system will encode 4-bit data into a 7-bit Hamming Code and decode it back while detecting and correcting any single-bit errors introduced during transmission.

Scope

- Designing a Hamming Code encoder in Verilog.
- Designing a Hamming Code decoder in Verilog.
- Integrating the encoder and decoder.
- Simulating the design using testbenches.
- Implementing the design on an FPGA.

Verilog Implementation

Encoder Design

The encoder takes a 4-bit input and generates a 7-bit Hamming Code. The process involves calculating parity bits and placing them in the correct positions.

```
module hamming_encoder(
    input [3:0] data_in,
    output [6:0] codeword_out
);
    // Parity bit calculations
    wire p1 = data_in[0] ^ data_in[1] ^ data_in[3];
    wire p2 = data_in[0] ^ data_in[2] ^ data_in[3];
    wire p3 = data_in[1] ^ data_in[2] ^ data_in[3];
```

```
    // Construct the codeword
    assign codeword_out = {p3, p2, data_in[3], p1, data_in[2:0]};
endmodule
```

Decoder Design

The decoder checks for errors by calculating the syndrome, which indicates the position of any erroneous bit. If an error is detected, it corrects the bit and outputs the original 4-bit data.

```
module hamming_decoder(
    input [6:0] codeword_in,
    output reg [3:0] data_out,
    output reg error
);
    wire p1_received = codeword_in[3];
    wire p2_received = codeword_in[5];
    wire p3_received = codeword_in[6];
    wire p1_calc = codeword_in[0] ^ codeword_in[1] ^ codeword_in[3];
    wire p2_calc = codeword_in[0] ^ codeword_in[2] ^ codeword_in[3];
    wire p3_calc = codeword_in[1] ^ codeword_in[2] ^ codeword_in[3];

    // Syndrome calculation
    wire [2:0] syndrome = {p3_received ^ p3_calc, p2_received ^ p2_calc, p1_received
^ p1_calc};

    always @(*) begin
        error = (syndrome != 3'b000);  // Set error flag if syndrome is non-zero
        data_out = codeword_in[6:3];   // Default data_out from the codeword

        // Correct the bit if error detected
        case(syndrome)
```

```
            3'b001: data_out[0] = ~data_out[0];  // Error in bit 0
            3'b010: data_out[1] = ~data_out[1];  // Error in bit 1
            3'b011: data_out[3] = ~data_out[3];  // Error in bit 3
            3'b100: data_out[2] = ~data_out[2];  // Error in bit 2
            default: ; // No error
        endcase
    end
endmodule
```

Top Module

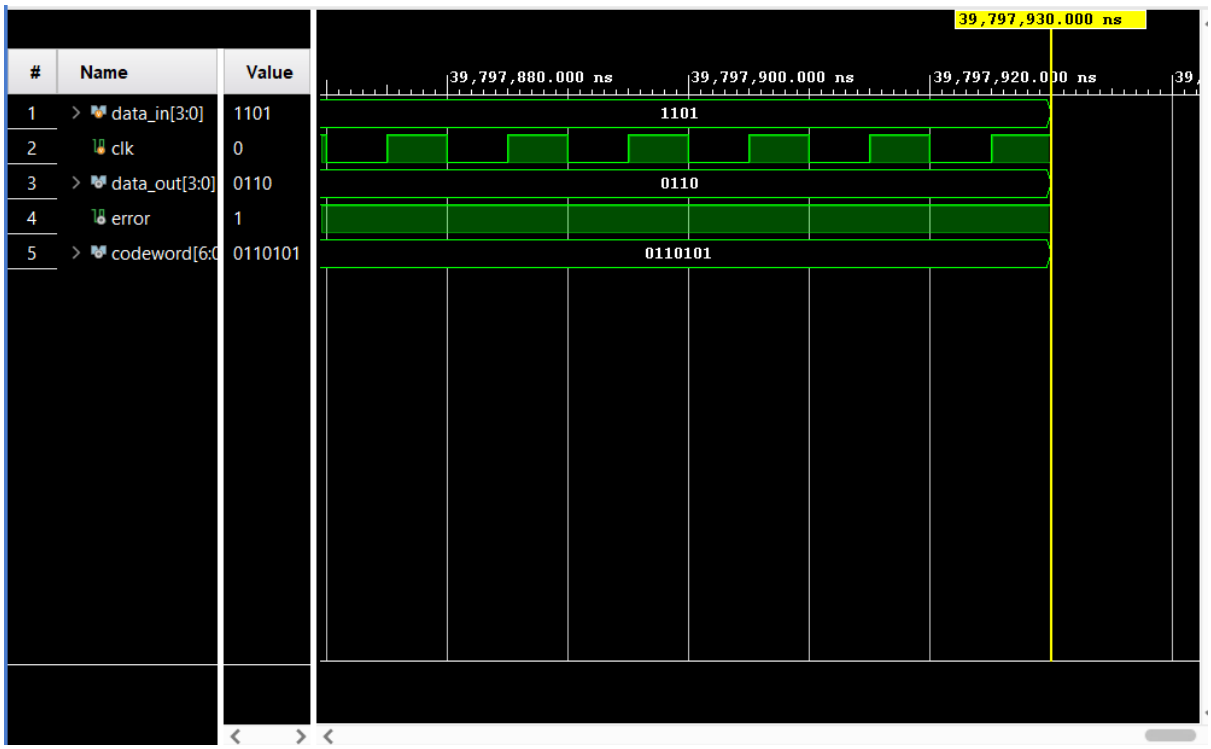
```
module hamming_top(
    input [3:0] data_in,
    input clk,
    output [3:0] data_out,
    output error
);
    wire [6:0] codeword;

    // Instantiate the encoder
    hamming_encoder encoder (
        .data_in(data_in),
        .codeword_out(codeword)
    );

    // Instantiate the decoder
    hamming_decoder decoder (
        .codeword_in(codeword),
        .data_out(data_out),
        .error(error)
    );
endmodule
```

Results

Simulation results show that the Hamming encoder correctly generates the 7-bit Hamming Code for a given 4-bit input, and the decoder successfully detects and corrects any single-bit errors, outputting the correct 4-bit data.



Conclusion

This project successfully demonstrates the design and implementation of a Hamming Code encoder and decoder using Verilog. The design was simulated and implemented on an FPGA, showing the practicality of Hamming Codes in real-world digital communication systems.

Appendix

Project Files

- **hamming_encoder.v:** Verilog file for the Hamming Code encoder.
- **hamming_decoder.v:** Verilog file for the Hamming Code decoder.
- **hamming_top.v:** Verilog file for integrating the encoder and decoder
- **hamming_top_tb.v:** Testbench for the encoder and decoder.
- **README.md:** Project documentation.