Project Report: $[15, 11, 3]_2$ Hamming Encoder Implemented on ZedBoard

Team Members: 2203101, 2203107, 2203329
Course: CS425 (Coding Theory and Cryptography)
Instructors: Dr. Rahul CS, Dr. Arpita Korwar
github repo

5th April, 2025

1. Objective

The primary goal of this project was to implement a [15,11,3] Hamming encoder using VHDL on a Xilinx ZedBoard. The encoder was designed to generate 4 parity bits from an 11-bit message, allowing single-bit error detection and correction. We achieved this using a component-based approach for basic logic gates and executed the design using Vivado HLS and Xilinx Vivado.

2. Introduction

In digital communication systems, error detection and correction codes play a crucial role in ensuring data integrity. Hamming codes, introduced by Richard Hamming in 1950, are a family of linear error-correcting codes that can detect and correct single-bit errors. A [15,11,3] Hamming code maps 11 bits of data to a 15-bit codeword by adding 4 parity bits. The minimum Hamming distance of 3 allows for the detection of up to 2 errors and correction of a single error. In this project, we implemented a Hamming encoder in VHDL using custom-designed components for fundamental logic gates. The final design was simulated and deployed on the ZedBoard to verify real-time encoding of binary messages.

3. Theory

3.1 Hamming Code [15,11,3]

The [15,11,3] Hamming code:

- Encodes 11-bit messages.
- Adds 4 parity bits.
- Generates a 15-bit codeword.

- Can detect up to 2 errors and correct 1 error.

Parity bits are computed using XOR combinations of message bits to satisfy even parity. The parity equations are:

$$\begin{aligned}
P_0 &= M_0 \oplus M_1 \oplus M_2 \oplus M_6 \oplus M_7 \oplus M_8 \oplus M_{10} \\
P_1 &= M_0 \oplus M_3 \oplus M_4 \oplus M_6 \oplus M_7 \oplus M_9 \oplus M_{10} \\
P_2 &= M_1 \oplus M_3 \oplus M_5 \oplus M_6 \oplus M_8 \oplus M_9 \oplus M_{10} \\
P_3 &= M_2 \oplus M_4 \oplus M_5 \oplus M_7 \oplus M_8 \oplus M_9 \oplus M_{10}
\end{aligned}$$

Each parity bit checks a specific set of message bits using XOR logic.

3.2 Parity Generation Matrix

The following matrix shows which message bits contribute to each parity bit. A 1 indicates that the corresponding message bit is included in the XOR computation of that parity bit.

Parity	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
P0	1	1	1	0	0	0	1	1	1	0	1
P1	1	0	0	1	1	0	1	1	0	1	1
P2	0	1	0	1	0	1	1	0	1	1	1
P3	0	0	1	0	1	1	0	1	1	1	1

Table 1: Parity Generation Matrix for [15,11,3] Hamming Encoder

3.3 The Parity Generator Matrix (PGM)

Considering the message words and their codewords to be represented as binary row vectors, we emphasise the linear nature of the coding scheme by asserting that any message vector $\mathbf{m}$ needs to be left-multiplied with a fixed *Parity Generator Matrix (PGM)* to produce the corresponding codeword $\mathbf{c}$.

Mathematically, this can be expressed as:

$$\mathbf{c} = \mathbf{m} \cdot \text{PGM}$$

The PGM depends solely on the choice of assignments of parity bit subsets to the message bit positions. In this implementation, the message vector $\mathbf{m}$ is of length 11 and the resulting codeword is of length 15. Thus, the PGM is an 11×15 matrix. The first 11 columns form the identity matrix I_{11}, and the last 4 columns specify, in Boolean form, the involvement of each message bit in the calculation of the 4 parity bits P_0 to P_3.

$$\text{PGM} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Each row represents a message bit m_i, and the '1's in the last four columns indicate which parity bits P_0 to P_3 are affected by m_i.

This matrix formalism helps unify the logic of encoding into a single, concise linear algebra operation, and reflects the syndrome decoding method typically used in Hamming code error correction as well.

4. Tools and Hardware Used

- **Software:**
  - Vivado HLS for high-level synthesis
- **Hardware:**
  - ZedBoard
- **Language:**
  - VHDL

5. Design Overview

Custom Logic Gate Components

We created a reusable VHDL package named `CS425`, which includes the following components:

- `NOT_1` - 1-input NOT gate
- `AND_2` - 2-input AND gate
- `OR_2` - 2-input OR gate
- `XOR_2` - 2-input XOR gate
- `XOR_7` - 7-input XOR gate

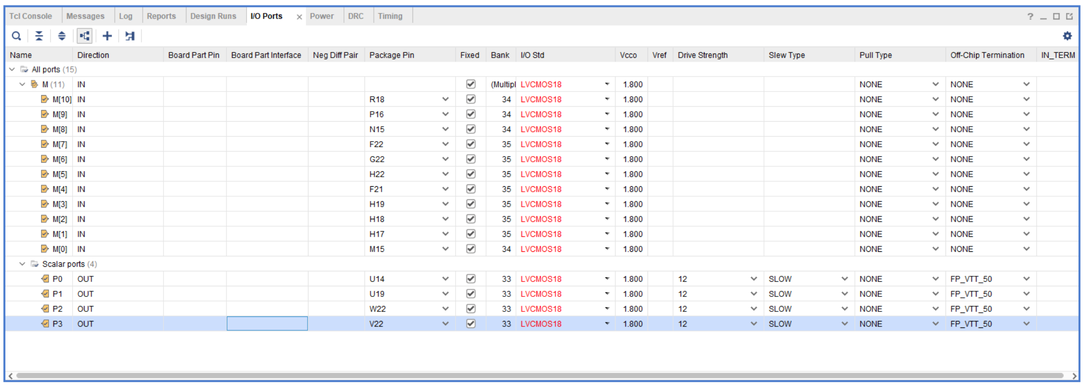
Hamming Encoder

The entity HAMMING_ENCODER takes an 11-bit input message and produces 4 parity bits using instances of the XOR_7 component. These outputs can then be used to generate the 15-bit codeword.

6. Implementation on ZedBoard

6.1 Input and Output

- Message (11 bits) is taken as input on ZedBoard user DIP Switches and Push Buttons
- Parity Bits (4 bits) are shown as output on the ZedBoard user LEDs



Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
All ports (15)															
M (11)	IN					✓	(Multipl	LVCMOS18	1.800				NONE	NONE	
M[10]	IN				R18	✓	34	LVCMOS18	1.800				NONE	NONE	
M[9]	IN				P16	✓	34	LVCMOS18	1.800				NONE	NONE	
M[8]	IN				N15	✓	34	LVCMOS18	1.800				NONE	NONE	
M[7]	IN				F22	✓	35	LVCMOS18	1.800				NONE	NONE	
M[6]	IN				G22	✓	35	LVCMOS18	1.800				NONE	NONE	
M[5]	IN				H22	✓	35	LVCMOS18	1.800				NONE	NONE	
M[4]	IN				F21	✓	35	LVCMOS18	1.800				NONE	NONE	
M[3]	IN				H19	✓	35	LVCMOS18	1.800				NONE	NONE	
M[2]	IN				H18	✓	35	LVCMOS18	1.800				NONE	NONE	
M[1]	IN				H17	✓	35	LVCMOS18	1.800				NONE	NONE	
M[0]	IN				M15	✓	34	LVCMOS18	1.800				NONE	NONE	
Scalar ports (4)															
P0	OUT				U14	✓	33	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50	
P1	OUT				U19	✓	33	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50	
P2	OUT				W22	✓	33	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50	
P3	OUT				V22	✓	33	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50	

Figure 1: I/O Ports Assignment

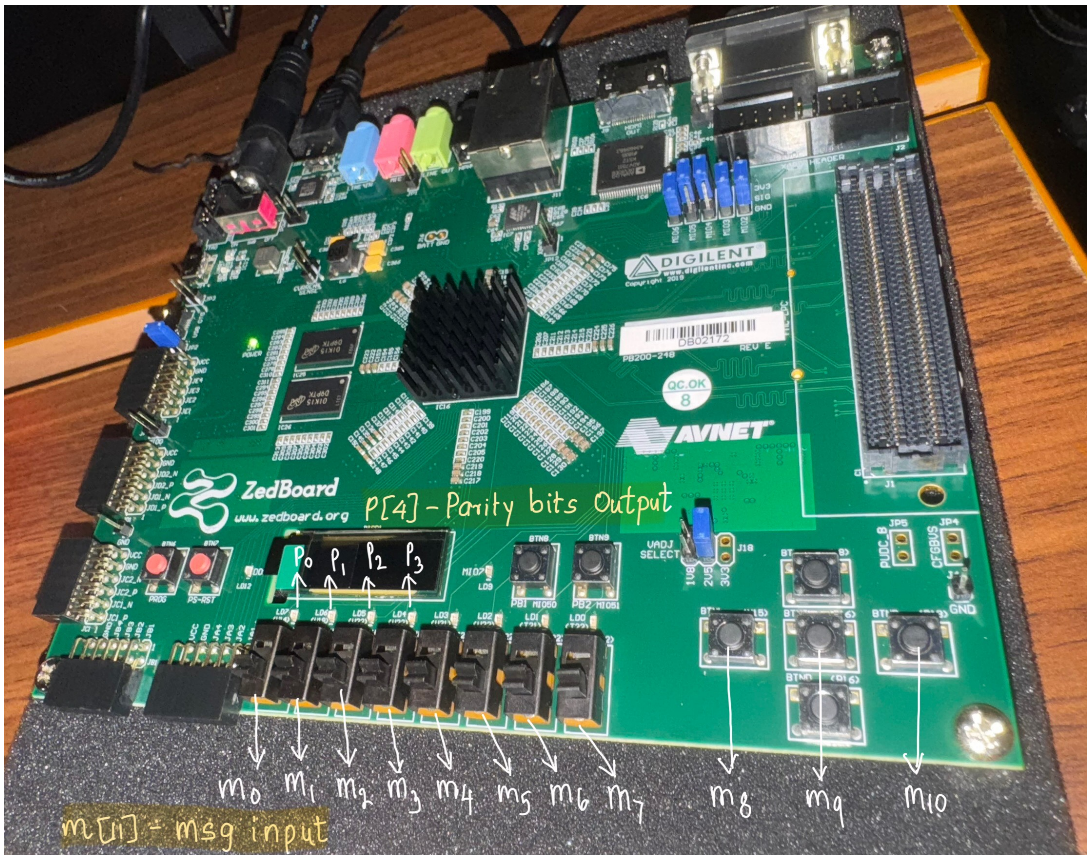
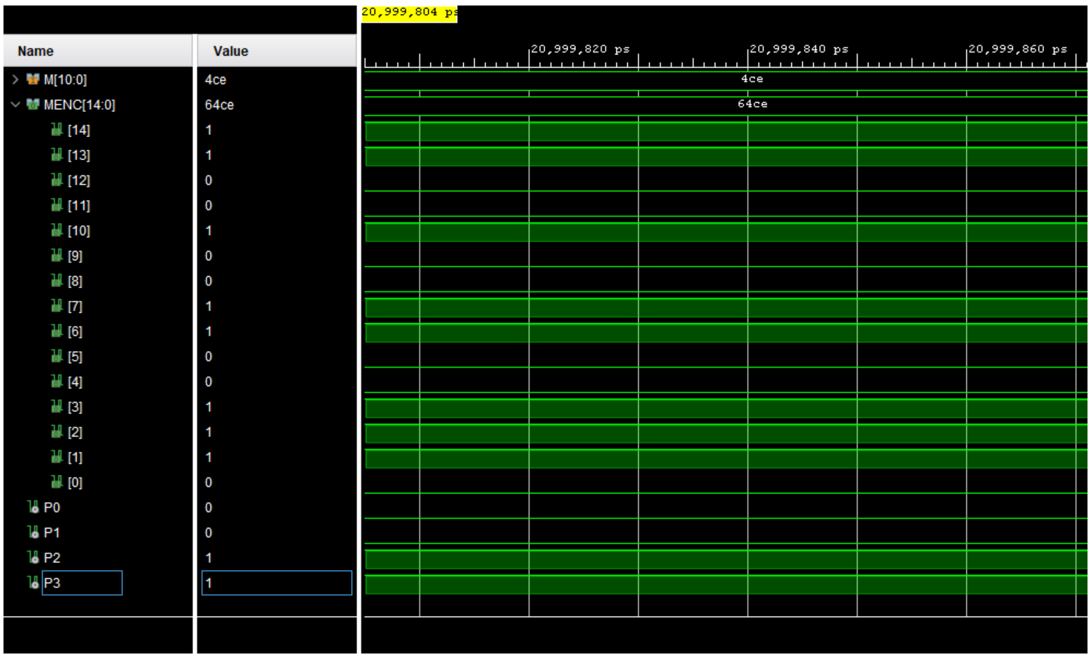

P[4] - Parity bits Output
P_0 P_1 P_2 P_3
m_0 m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9 m_{10}
m[11] - msg input

Figure 2: ZedBord

6.2 Simulation Results

- Behavioral simulation verified the logic correctness of parity generation.



```
20,999,804 ps
                      20,999,820 ps   20,999,840 ps   20,999,860 ps
Name          Value
> M[10:0]     4ce
v MENC[14:0]  64ce
    [14]      1
    [13]      1
    [12]      0
    [11]      0
    [10]      1
    [9]       0
    [8]       0
    [7]       1
    [6]       1
    [5]       0
    [4]       0
    [3]       1
    [2]       1
    [1]       1
    [0]       0
  P0          0
  P1          0
  P2          1
  P3          1
```

Figure 3: Simulation Results

6.3 Schematic Diagram

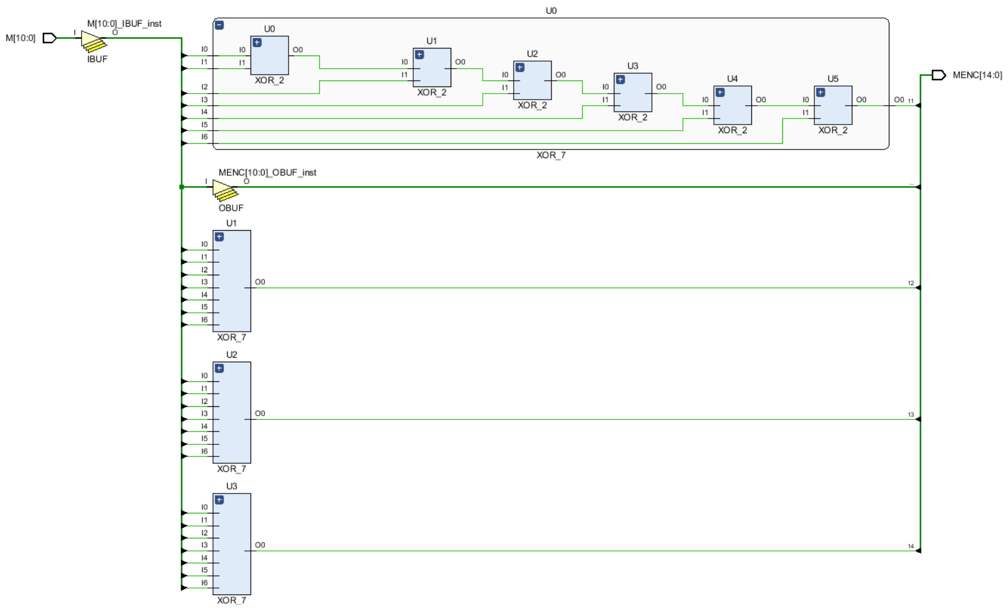


Figure 4: Circuit Diagram

Conclusion

We successfully implemented a [15,11,3] Hamming encoder using modular VHDL components and deployed it on the ZedBoard. The project demonstrated our understanding of both digital design concepts and error-correcting code theory. The use of gate-level design improved modularity and understanding of low-level logic operations.

Future Work

- Develop it for any arbitrary $[n, k, d]_2$ Hamming Code.
- Develop a Decoder that manages error correction.
- Extend the design to more complex codes like BCH or Reed-Solomon.

References

1. Digital Design Class Notes
2. Coding Theory Class Notes and Assignment Descriptions