Hamming Error Control Architecture

Bhoomi A[1], Arushi M Sandilya[2], Asst Professor Niveditha V K[3]
[1,2]*Atria Institute of Technology*
[3]*Asst Professor Atria Institute of Technology*

***Abstract*— In order to provide dependable digital data transfer, this study discusses the design and construction of an error-control hardware circuit employing the Hamming (7,4) code. The system detects and fixes single-bit faults during transmission by using XOR-based logic for parity generation and syndrome computation. In order to encode, transmit, identify, and fix data problems in real time, a prototype hardware circuit was created. The successful implementation emphasizes the significance of forward error correction in digital communication systems and confirms the efficacy of Hamming coding schemes.**

***Index Terms*— Hamming code, digital communication, error correction, syndrome decoding, XOR logic.**

INTRODUCTION

In contemporary digital communication networks, dependable data transfer is crucial. Errors may arise from interference, noise, or distortions while data travels across a noisy channel. This problem is addressed via error detection and correction methods. Because of their effectiveness and ease of use, hamming codes are among the most used.
The Hamming (7,4) code adds three parity bits to four data bits to create a seven-bit codeword. It is appropriate for low-complexity hardware applications since it can identify and fix any single-bit fault. This project uses XOR-based combinational logic to implement the entire encoding and decoding process. Real-time error management is demonstrated by displaying the final corrected output using straightforward indicators.

II. THEORY

Four data bits (D1-D4) and three parity bits (P1-P3) make up a Hamming (7,4) code. Data bits are stored in locations 3, 5, 6, and 7 of the 7-bit word, whereas parity bits are stored in positions 1, 2 and 4. Parity bits are calculated using even parity.

Bit Position:	1	2	3	4	5	6	7
Bit Type:	P1	P2	D1	P3	D2	D3	D4

Parity Bit Equations
$P1 = D1 \oplus D2 \oplus D4$
$P2 = D1 \oplus D3 \oplus D4$
$P3 = D2 \oplus D3 \oplus D4$
Syndrome Calculation
At the receiver, parity checks are recomputed:
$S1 = R1 \oplus R3 \oplus R5 \oplus R7$
$S2 = R2 \oplus R3 \oplus R6 \oplus R7$
$S3 = R4 \oplus R5 \oplus R6 \oplus R7$

The 3-bit syndrome identifies the position of the erroneous bit.

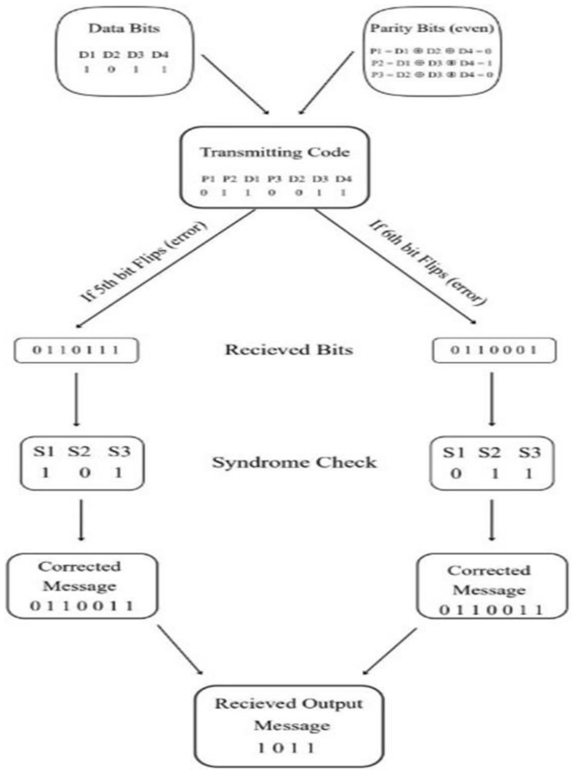
Parity bit	Check bits	Equation
P1	1, 3, 5, 7	$P1 = D1 \oplus D2 \oplus D4$
P2	2, 3, 6, 7	$P2 = D1 \oplus D3 \oplus D4$
P3	4, 5, 6, 7	$P3 = D2 \oplus D3 \oplus D4$

Parity and Structure Bit Calculation:
The seven bits are set up so that parity bits occupy locations that are powers of two (1, 2, 4).
The data bits are stored in locations 3, 5, 6, and 7.
Every parity bit keeps an eye on a particular group of bits, including itself, and is configured so that the set's total number of ones is even (this is even parity).
Parity bit 1 (position 1): covers bits 1, 3, 5, 7
$S1 = R1 \oplus R3 \oplus R5 \oplus R7$
Parity bit 2 (position 2): covers bits 2, 3, 6, 7
$S2 = R2 \oplus R3 \oplus R6 \oplus R7$
Parity bit 4 (position 4): covers bits 4, 5, 6, 7
$S3 = R4 \oplus R5 \oplus R6 \oplus R7$

III. METHODOLOGY

The procedural steps followed
- Assign data locations D1-D4 four input data bits.
- Set aside parity bits for places 1, 2, and 4.
- Use XOR logic to create parity bits.

- Create the seven-bit Hamming codeword.
- For testing, manually introduce single-bit mistakes.
- To identify incorrect bit locations, create a syndrome.
- Flip the bit's value to fix it.
- Retrieve the initial 4-bit information.



IV. OBSERVATION AND CALCULATIONS

An example calculation

- Data bits: D1 D2 D3 D4 = 1 0 1 1
- Parity bits:
  - P1 = 0
  - P2 = 1
  - P3 = 0
- Transmitted codeword: 0 1 1 0 0 1 1

Case 1 - Error in bit 5:

- Syndrome = 101
- Corrected codeword = 0 1 1 0 0 1 1
- Recovered data = 1 0 1 1

Case 2 - Error in bit 6:

- Syndrome = 011
- Corrected codeword = 0 1 1 0 0 1 1
- Recovered data = 1 0 1 1

V. RESULTS

- Single-bit mistakes were successfully identified and fixed by the system.
- The original data bits were correctly recovered in both test scenarios.
- Hardware indicators were used to indicate real-time adjustment.

VI. CONCLUSION

The Hamming (7,4) code's hardware implementation uses straightforward XOR-based circuitry to efficiently identify and fix single-bit faults. The technique's dependability and effectiveness for real-world communication systems are confirmed by the real-time outcomes. This approach might be expanded in the future to address multiple-bit faults or included into sophisticated communication modules.

REFERENCES

[1] R. W. Hamming, "Error detecting and error correcting codes," Bell System Technical Journal, vol. 29, no. 2, pp. 147–160, 1950.

[2] S. Haykin, Communication Systems, 4th ed. Wiley India, 2006.

[3] B. Sklar, Digital Communications: Fundamentals and Applications, 2nd ed. Pearson Education, 2001.

[4] J. G. Proakis and M. Salehi, Digital Communications, 5th ed. McGraw-Hill, 2008.

[5] W. Stallings, Data and Computer Communications, 10th ed. Pearson Education, 2014.

[6] M. M. Mano and M. D. Ciletti, Digital Design, 5th ed. Pearson Education, 2013.

[7] TutorialsPoint, "Hamming Code – Error Detection and Correction," Online article, accessed 2025.

[8] GeeksforGeeks, "Hamming Code in Computer Networks," Online article, accessed 2025.

[9] NPTEL, Digital Communication – Error Control Coding, Video lecture series, accessed 2025.