Error Detection Using Two Dimensional Parity Sum Checking Matrix

Chirag Jamadagni,
Department of Computer Science and Engineering,
National Institute of Technology, Karnataka
chirag.jamadagni@gmail.com

Ananya Rajkumar,
Department of ComputerScience and Engineering,
National Institute of Technology, Karnataka
ananya.rajkumar95@gmail.com

BR Chandavarkar,
Department of Computer Science and Engineering,
National Institute of Technology, Karnataka
brcnitk@gmail.com

Abstract -- Error detection and error correction have become increasingly important terminologies in context of the data link layer in communication. Through the years, several efforts have been made to reduce the number of errors while transmitting data in the form of bits. Addition of a parity bit to the message or information to be sent is employed to detect a single bit error or an odd number of bit errors. The two dimensional parity checking matrix is an extension of this concept to detect two bit errors. In this paper, we propose a modified matrix known as two dimensional parity sum checking matrix, which is more efficient in calculating three and four bit errors.

I. INTRODUCTION

Transmission of a message through a channel, in the form of bits in the data link layer is prone to errors due to modification in the value of one or many bits in the message transmitted. As a result, the message on the receiver's side is corrupted, and efficiency in data transfer is reduced. Error detection and error correction methods are introduced at the sender and/or receiver's sides to validate the correctness of the message sent or received respectively. A common method employed is the use of redundant bits.

In the case of block coding, the information to be transmitted is broken down into blocks or datawords. These datawords are padded with redundant bits to form codewords. A parity bit is an example of a redundant bit added based on the number of 1s in a given dataword. A parity bit is computed and added at the sender's side and retrieved and validated at the receiver's side. A single parity can detect any error in an odd number of bits. However, it is insensitive to an even number of errors. A single parity bit attached to a dataword is incapable of correcting errors.

A parity scheme can be modified to increase its efficiency in error detection and also incorporate error correction (for a limited number of errors). Increasing redundancy facilitates greater capability of error detection.[1] The two dimensional parity sum checking matrix applies this concept to the basic two dimensional parity checking matrix by introducing more than one row and column parity bits to increase efficiency in error detection. Section 2 of this paper explains the proposed parity sum checking matrix, generalized to a dataword of any length. In sections 3, the parity sum checking matrix is explained using datawords of byte length (8 bits) and the efficiency of error detection is calculated. Section 4 illustrates the hardware implementation and the circuit required for the matrix example in Section 3. Section 5 analyses the efficiency of the introduced redundancy in the circuit. The paper concludes with Section 6.

II. TWO DIMENSIONAL PARITY SUM CHECKING MATRIX

In a traditional two dimensional parity checking matrix, the datawords are arranged in rows and columns with a parity bit at the end of each row and column, calculated for that row or column. This parity bit can be an odd parity or even parity depending on the scheme implemented. This matrix table is sent to the receiver, where the syndrome for each row and column is found out and the location of the error bit(s) is (are) determined using the corresponding row and column with errors in syndrome. An error of four bits cannot be detected by this method and in general, an error of 2^n bits, where n>1 goes undetected.

A two dimensional parity sum checking matrix is a slightly modified version of the conventional parity checking matrix. The datawords are arranged in rows and columns such that the number of bits in each row or column is of the form $n = 2^k$, where $k \epsilon N$. For rows or columns with n dataword bits, the codeword is obtained by adding a parity sum of $\log_2 n$ (k) bits at the end. The parity sum is determined by counting the number of 1s in the respective row or column and representing it as a binary sum. Thus, every matrix computed is of size [m * $\log_2 m$][[n * $\log_2 n$] where m is the number of dataword bits in every row and n is the number of dataword bits in every column.

The values m and n maybe equal or unequal for rows and columns but must be the same for all rows or all columns. The number of redundant bits required will vary depending on the number of rows and columns. For uniformity in the matrix and an uncomplicated hardware circuit, m and n are kept equal. The number of bits in each row or column not being of the form 2^k can also be accommodated. The number of redundant bits in such a

scenario changes to ceiling ($\log_2 n$). The drawback of this alternative is that the parity sum bits are not optimally used (all possible combinations of the parity bits are not exhausted) and is hence not recommended.

III. Byte Length Dataword Parity Sum Matrix

To explain and illustrate the implementation of the two dimensional parity sum checking matrix, an example with datawords of byte length is considered. An example of an arbitrary matrix in this configuration is given in Figure 1. Therefore,

number of bits in each row/column = $8 = 2^3$

number of redundant bits/parity sum bits per row/column = $\log_2 8 = 3$

length of codeword in every row/column = 8+3 = 11

		Datawords								Row Parity sum P1	P2	P3
		1	1	0	1	1	0	0	0	1	0	0
		0	1	0	0	0	1	1	1	1	0	0
		1	0	1	0	1	1	1	0	0	1	1
		1	1	0	0	0	1	0	0	0	1	1
		0	0	1	0	1	0	1	0	0	1	1
		1	0	1	1	1	0	1	1	1	0	0
		1	0	1	1	1	1	0	1	1	1	0
		0	1	1	0	0	0	1	0	0	1	1
Column Parity Sum	P1	0	1	1	0	0	1	1	0			
	P2	1	0	0	1	1	0	0	1			
	P3	1	0	1	1	1	0	0	1			

- Detected bit
- Undetected bit

Figure 1: Byte length parity sum checking matrix

The parity sum checking matrix can detect all errors up to 2 bits and correct those of 1 bit. For errors up to 7 bits with no more than one bad bit in the same row or column, the errors can be detected. In any row or column with an odd number of errors in that row or column, there is always a change in the corresponding sum and therefore, such an error is always noticed. The possibility of an error going unrecognised arises only when there exists a bad bit in the same column and row of the unrecognised erroneous bit. While this is true even for the orthodox two dimensional parity checking matrix, the inclusion of parity sum increases the probability of detecting such an error as compared to the former method.

An error of 8 bits cannot be detected as in a 3 parity sum, the values of zero and eight are the same (3 least significant bits are taken). Thus, if in a single row (or column) with all 1s or 0s an 8 bit error is come across, the error is not spotted by the row (or column) but every column (row) of the matrix indicates an error, making it impossible to spot the error.

A. Three bit triangular error and four bit square-form error detection

As seen in the example shown in Figure 1, a four bit error in the form of a square is detected in the bits marked in green but remains undetected for those highlighted in yellow.

A bad bit occurs when 0 changes to 1 or 1 changes to 0[1]. In a row or column with 2 bad bits, there are 4 possible combinations.

Value(Changed) in row/column		Error detection by corresponding row/column
0(1)	0(1)	Detected
0(1)	1(0)	Undetected
1(0)	0(1)	Undetected
1(0)	1(0)	Detected

Figure 2: Table of detected and undetected combinations of bad bits

When there are an even number of bits in a single row or column and their original bit values are not identical, the value of sum does not vary on encountering their bad bits. Hence, these errors cannot be spotted by that row or column. However, from Figure 1 and Figure 2, we see that when two bad bits in a single row or column with the same original values are changed, with their incorrect values also being the same, an affirmative change in the sum always detects the error. Extending this concept to three bit errors in an upper or lower triangular form, (as shown in Figure 3) there are only two cases in which the bad bit indicated by b2 cannot be detected, i.e. when (b1b2b3) = 010 or (b1b2b3) = 101. It is only in these cases that the row and column parity sums of b2 remain unchanged. However, for a regular two dimensional parity checking matrix, b2 cannot be detected for any combination of 1s and 0s for (b1b2b3) (2^3 combinations).

Percentage undetected for three bit triangular errors

$$\frac{2}{2^3} * 100$$

25

Therefore, for a three bit triangular error of the from b1b2b3, (Figure 3) the error in b2 is likely to be detected 75% of the time using a parity sum while it is never detected in the traditional matrix.

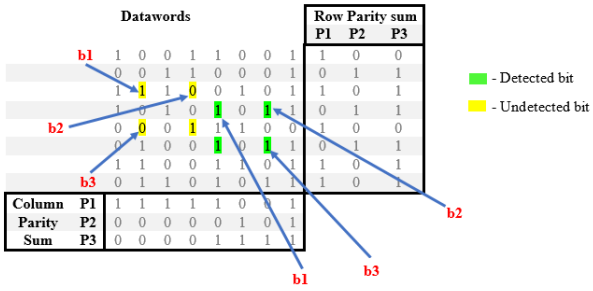

Figure 3: three bit triangular error detection

Similarly, the probability of detecting a four bit square error can also be determined. An error with alternating 0s and 1s when the bad bits are taken in cyclic order, can be of the form 1010 or 0101. If such an error occurs, the parity sum checking matrix cannot detect any erroneous bit. A similar situation where the error is not noticed is when a three bit triangular error occurs in the four bit square error. To distinguish this form from the previous error combination, for a cycle with 010, the fourth bit must also be 0 and for a cycle with 101, the fourth bad bit should be 1. For every such error combination, there are 4 different cycle orientations, making error detection not feasible for 8 such four bit combinations. (2 arrangements * 4 orientations) However, in each of these error squares, only one bad bit is not detected, i.e., the bit in the position analogous to b2 in the triangular error of the cycle. On the contrary, the standard parity checking matrix fails to determine any bad bit in the 16 possible configurations for a four bit error in a square form.

Percentage undetected for four bit square errors

$$= \sum \frac{\text{number of bits undetected} * \text{number of undetected bit configurations}}{\text{number of bits per configuration} * \text{number of configurations}} * 100$$

$$= \frac{4 * 2 + 1 * 8}{4 * 16} * 100$$

$$= \frac{8}{64} * 100$$

$$= 25\%$$

Thus, 75% of four bit errors occurring in the form of a square can be detected, while in the typical parity checking matrix, no such error can be detected.

IV. CIRCUIT DIAGRAM FOR HARDWARE IMPLEMENTATION OF PARITY SUM

Calculation of the parity sum must be performed at the sender's side as well as the receiver's as shown in Figure 4 and Figure 5 respectively. The sender calculates the parity sum P1 P2 P3 with P1 as MSB (Most significant bit) and P3 as LSB (Least significant bit), for each row and column and transmits the matrix to the receiver. The receiver decodes the matrix by passing the first eight bits of every row and column to the summing circuit and its output is compared with the respective parity bit of the row or column. If all the three bits are equal, there is no error detected in the circuit. Otherwise, the corresponding row or column contains at least one error.

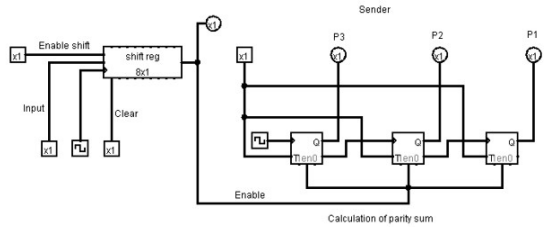

Figure 4: Parity sum sender circuit

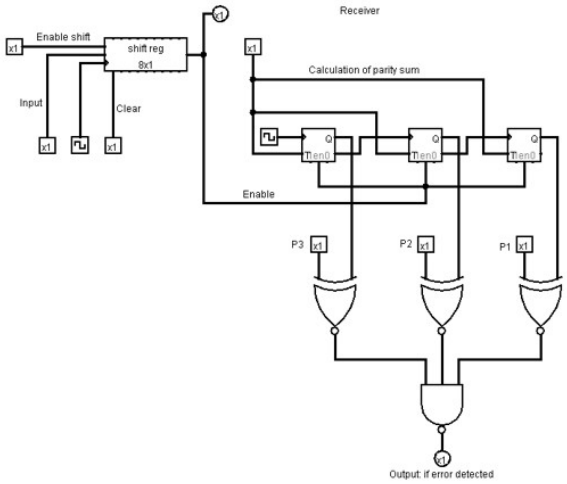

Figure 5: Parity sum receiver circuit

To calculate the sum of 1s in a given dataword, the data bits of the required row or column are fed into an 8x1 shift register as input. Shift is enabled and the output of the shift register is given to the enable of a three bit ripple counter. Thus, the counter increments only when a data bit 1 is the serial output of the shift register, counting the number of ones in the dataword. After every 8 bits, the shift can be disabled, the value of the outputs are noted and a new data word is allowed to come into the shift register.

In the sender's circuit, the computation of sum is taken as the output and is filled in the Parity sum portion of the table. Once the table is complete, it is sent to the receiver. In the receiver's circuit, the sum of ones in the received dataword is calculated and compared bit by bit with the Parity sum values of the table. Equality of all three bits gives the output 0, indicating no error and an output of 1 implies error(s) in the given dataword.

V. ANALYSIS AND RESULT

A quantitative analysis of the proposed error detection method is performed by calculating the overhead and code rate for a generalized two dimensional parity sum checking matrix. Their graphs vs dataword length are plotted using matlab.

For a matrix of the order [m * $\log_2$m][[n * $\log_2$n], the overhead and code rate are calculated as follows:

Overhead[1]

$$= \frac{\text{Number of parity bits}}{\text{number of information bits}}$$

$$= \frac{\frac{\log m}{\log 2} * m + \frac{\log n}{\log 2} * n}{m * n}$$

Code rate[1]

$$= \frac{\text{Number of information bits}}{\text{number of (information bits + parity bits)}}$$

$$= \frac{m * n}{(m * n) + (\frac{\log m}{\log 2} * m + \frac{\log n}{\log 2} * n)}$$

For a simplified [n * log2n][[n * log2n] matrix,

Overhead

$$= \frac{2 * \frac{\log n}{\log 2} * n}{n * n}$$

$$= \frac{2 * \frac{\log n}{\log 2}}{n}$$

Using the overhead formula for a byte length dataword matrix, overhead is computed to be 0.75.

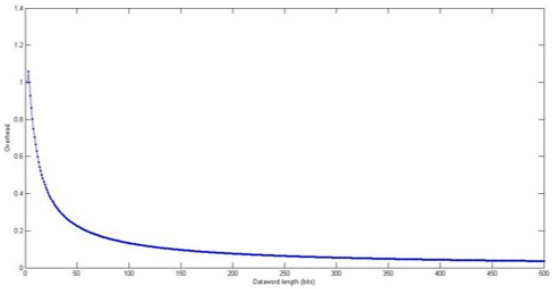

Figure 6: Dataword length (bits) vs Overhead

Code rate

$$\frac{n * n}{(n * n) + \left(\frac{\log n}{\log 2} * n + \frac{\log n}{\log 2} * n\right)}$$

$$\frac{n}{n + \frac{2 * \log n}{\log 2}}$$

Using the code rate formula for a byte length dataword matrix, code rate is computed to be 0.571.

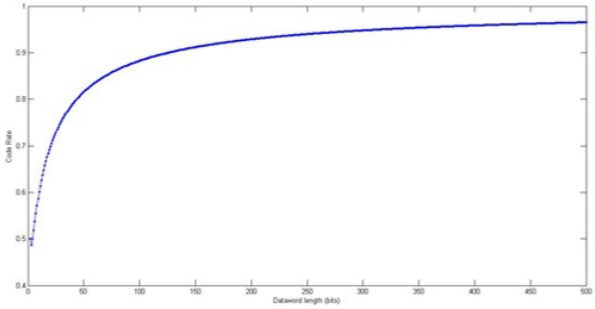

Figure 7: Dataword length (bits) vs Code rate

Plotting a graph dataword length (in bits) vs the overhead of the matrix (Figure 6) and a graph of dataword length (in bits) vs the coderate (Figure 7), we observe a favourable decrease in overhead and increase in code rate with increasing dataword length. However, error detection becomes less efficient with increasing dataword length due to increasing number of bad bits in the matrix. An optimal size of a dataword must be chosen arriving at a compromise between overhead, code rate and efficiency.

VI. CONCLUSION

A two dimensional parity sum checking matrix proposed in this paper has been discussed in detail. A circuit to be implemented at the sender's and receiver's sides has been illustrated. A quantitative analysis, evaluating the parameter of implementing this matrix has also been performed. The overhead increases and code rate reduces by including redundant bits in the parity sum. Through an example of a byte length dataword parity sum checking matrix, we saw that for single, 2 bit and multiple errors detected by a two dimensional parity checking matrix, the modified matrix has the same efficiency in error detection and error correction. However, for 3 and four bit errors, 75% of the errors that were undetected by the original matrix are detected by the proposed model, hence making it more efficient than its traditional counterpart in error detection.

REFERENCES

[1] Anne, N.B.; Utthaman Thirunavukkarasu; Latifi, S., "Three and four-dimensional parity-check codes for correction and detection of multiple errors," *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on* , vol.2, no., pp.837,842 Vol.2, 5-7 April 2004