Supplement to

Logic and Computer Design Fundamentals
3rd Edition[1]

Error Detection and Correction

Selected topics not covered in the third edition of *Logic and Computer Design Fundamentals* are provided here for optional coverage and for self-study if desired. This material fits well with the desired coverage in some programs but not may not fit within others due to time constraints or local preferences. This supplement is referenced in Chapter 9 as a part of the coverage of memories. Error detection and correction codes serve as a basis for error detection and correction in semiconductor memories, in particular, DRAM, in systems where greater reliability is required.

The small size of the transistors or capacitors, combined with cosmic ray effects, causes occasional errors in stored information in large, dense RAM chips, particularly those that are dynamic. These errors can be detected and corrected by employing error-detecting and -correcting codes in RAMs. The most common error detection scheme is the parity bit. (See Section 2-7.) A parity bit is generated and stored along with the data word in memory. The parity of the word is checked after reading the word from memory. The word is accepted if the parity of the bits read out is correct. If the parity of the bits read is incorrect, an error is detected, but it cannot be corrected.

An error-correcting code uses multiple parity check bits that are stored with the data word in memory. Each check bit is a parity bit for a group of bits in the data word. When the word is read back from memory, the parity of each group, including the check bit, is evaluated. If the parity is correct for all groups, it signifies that no detectable error has occurred. If one or more of the newly generated parity values is incorrect, a unique pattern called a syndrome results that may be able to identify which bit is in error. A single error occurs when a bit changes in value from 1 to 0 or from 0 to 1 while stored or if it erroneously changes during a write or read operation. If the specific bit in error is identified, then the error can be corrected by complementing the erroneous bit.

Hamming Codes

The most common types of error-correcting codes used in RAM are based on the codes devised by R. W. Hamming. In the Hamming code, k parity bits are added to an n-bit data word, forming a new word of $n + k$ bits. The bit positions are numbered in sequence from 1 to $n + k$. Those positions numbered with powers of two are reserved for the parity bits. The remaining bits are the data bits. The code can be used with words of any length.

Before giving the general characteristics of the Hamming code, we will illustrate its operation with a data word of eight bits. Consider, for example, the 8-bit data word 11000100. We include four parity bits with this word and arrange the 12 bits as follows:

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

The 4 parity bits P_1 through P_8 are in positions 1, 2, 4, and 8, respectively. The 8 bits of the data word are in the remaining positions. Each parity bit is calculated as follows:

$$P_1 = \text{XOR of bits } (3, 5, 7, 9, 11) = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$$

$$P_2 = \text{XOR of bits } (3, 6, 7, 10, 11) = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$$

$$P_4 = \text{XOR of bits } (5, 6, 7, 12) = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$P_8 = \text{XOR of bits } (9, 10, 11, 12) = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

Recall that the exclusive-OR operation performs the odd function. It is equal to 1 for an odd number of 1's among the variables and to 0 for an even number of 1's. Thus, each parity bit is set so that the total number of 1's in the checked positions, including the parity bit, is always even.

The 8-bit data word is written into the memory together with the 4 parity bits as a 12-bit composite word. Substituting the 4 parity bits in their proper positions, we obtain the 12-bit composite word written into memory:

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	1	1	1	0	0	1	0	1	0	0

When the 12 bits are read from memory, they are checked again for errors. The parity of the word is checked over the same groups of bits, including their parity bits. The four check bits are evaluated as follows:

$$C_1 = \text{XOR of bits } (1, 3, 5, 7, 9, 11)$$

$$C_2 = \text{XOR of bits } (2, 3, 6, 7, 10, 11)$$

$$C_4 = \text{XOR of bits } (4, 5, 6, 7, 12)$$

$$C_8 = \text{XOR of bits } (8, 9, 10, 11, 12)$$

A 0 check bit designates an even parity over the checked bits, and a 1 designates an odd parity. Since the bits were written with even parity, the result, $C = C_8C_4C_2C_1 = 0000$, indicates that no error has occurred. However, if $C \neq 0$, the 4-bit binary number formed by the check bits gives the position of the erroneous bit if only a single bit is in error. For example, consider the following three cases:

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	
	0	0	1	1	1	0	0	1	0	1	0	0	No error
	1	0	1	1	1	0	0	1	0	1	0	0	Error in bit 1
	0	0	1	1	0	0	0	1	0	1	0	0	Error in bit 5

In the first case, there is no error in the 12-bit word. In the second case, there is an error in bit position number 1 because it changed from 0 to 1. The third case shows an error in bit position 5 with a change from 1 to 0. Evaluating the XOR of the corresponding bits, we determine the four check bits to be as follows:

	C_8	C_4	C_2	C_1
No error	0	0	0	0
Error in bit 1	0	0	0	1
Error in bit 5	0	1	0	1

Thus, for no error, we have $C = 0000$; with an error in bit 1, we obtain $C = 0001$; and with an error in bit 5, we get $C = 0101$. Hence, when C is not equal to 0, the decimal value of C gives the position of the bit in error. The error can then be corrected by complementing the corresponding bit. Note that an error can occur in the data or in one of the parity bits.

The Hamming code can be used for data words of any length. In general, for k check bits and n data bits, the total number of bits, $n + k$, that can be in a coded word is at most $2^k - 1$. In other words, the relationship $n + k \leq 2^k - 1$ must hold. This relationship gives $n \leq 2^k - 1 - k$ as the number of bits for the data word. For example, when $k = 3$, the total number of bits in the coded word is $n + k \leq 2^3 - 1 = 7$, giving $n \leq 7 - 3 = 4$. For $k = 4$, we have $n + k \leq 15$, giving $n \leq 11$. Thus, the data word may be less than 11 bits, but must have at least five bits; otherwise, only three check bits will be needed. The relationships for $k = 3$ and $k = 4$ justify the use of four check bits for the eight data bits in the previous example.

The grouping of bits for parity generation and checking can be determined from a list of the binary numbers from 0 through $2^k - 1$. The least significant bit is a 1 in the binary numbers 1, 3, 5, 7, and so on. The second significant bit is a 1 in the binary numbers 2, 3, 6, 7, and so on. Comparing these numbers with the bit positions used in generating and checking parity bits in the Hamming code, we note the relationship between the bit groupings in the code and the position of the 1-bits in the binary count sequence. Each group of bits starts with a number that is a power

of 2—for example, 1, 2, 4, 8, 16, and so forth. These numbers are also the position numbers for the parity bits.

The basic Hamming code can detect and correct an error in only a single bit. Some multiple-bit errors are detected, but they may be corrected erroneously, as if they were single-bit errors. By adding another parity bit to the coded word, the Hamming code can be used to correct a single error and detect double errors. If we include this additional parity bit, the previous 12-bit coded word becomes $001110010100P_{13}$, where P_{13} is evaluated from the exclusive-OR of the other 12 bits. This produces the 13-bit word 0011100101001 (even parity). When this word is read from memory, the check bits and also the parity bit P are evaluated over the entire 13 bits. If $P = 0$, the parity is correct (even parity), but if $P = 1$, the parity over the 13 bits is incorrect (odd parity). The following four cases can occur:

If $C = 0$ and $P = 0$	No error occurred.
If $C \neq 0$ and $P = 1$	A single error occurred that can be corrected.
If $C \neq 0$ and $P = 0$	A double error occurred that is detected but cannot be corrected.
If $C = 0$ and $P = 1$	An error occurred in the P_{13} bit.

Note that this scheme will detect more than two erroneous bits in many cases, but is not guaranteed to detect all such errors.

A modified Hamming code to generate and check parity bits for a single-error-correction, double-error-detection scheme is most often used in real systems. The modified code uses a different parity check bit scheme that balances the number of inputs to the logic for each check bit and thus the number of inputs to each circuit that does the checking. The balancing minimizes the delay through the error correction and detection circuits. These circuits can be used in a RAM subsystem to add check bits during write operations and to correct single errors and detect double errors during read operations. (See Problem A.6-5-6.)

References

1. HAMMING, R. W. "Error Detecting and Error Correcting Codes." *Bell System Tech. Jour.*, 29 (1950): 147–160.

Problems

The plus (+) indicates a more advanced problem.

1. A 12-bit Hamming code word containing 8 bits of data and 4 parity bits is read from memory. What was the original 8-bit data word that was written into memory if the 12-bit word read out is
(a) 010011111000 **(b)** 011101010010 **(c)** 010000000101

2. How many parity check bits must be included with the data word to achieve single error correction and double error detection when the data word contains (a) 16 bits; (b) 32 bits; (c) 64 bits?

3. It is necessary to formulate the Hamming code for four data bits D_3, D_5, D_6. and D_7, together with three parity bits P_1, P_2, and P_4.
(a) Evaluate the 7-bit composite code word for the data word 0110.
(b) Evaluate the three check bits C_1, C_2, and C_4, assuming no error.
(c) Assume an error in bit D_5 during storage into memory. Show how the error in the bit is detected and corrected.
(d) Add a parity bit P to include double error detection in the code. Assume that errors occurred in bits P_2 and D_5. Show how this double error is detected.

4. *A modified single-error-correcting, double-error-detecting Hamming code for four bits of data D_3, D_5, D_6, and D_7 has the following parity bit equations:

$$P_1 = D_3 \oplus D_5 \oplus D_6$$

$$P_2 = D_3 \oplus D_5 \oplus D_7$$

$$P_4 = D_3 \oplus D_6 \oplus D_7$$

$$P_8 = D_5 \oplus D_6 \oplus D_7$$

(a) Find the binary values of the four check bits for a single error in each of the eight bit positions of the code.
(b) Assuming that either a single or double error has occurred, indicate the type of error for each of the following words read from memory:
(1) 00110011 (2) 01100100 (3) 01000101
(c) Give the correct data bit values for the single-error cases in (b).

5. Given the 8-bit data word 10110100, generate the 13-bit composite word for the Hamming code that corrects single errors and detects double errors.

6. Given the 11-bit data word 00100101010, generate the corresponding 15-bit Hamming code word.