Adding Error Detecting and Correcting Code features to OpenRAM

Nursultan Kabylkas

CMPE 223 - Spring 2017

Problem

Data stored in memory is prone to errors. Bits in the memory may flip due to various reasons. Here are some examples:

- Electromagnetic interference;
- Lowering the supply voltage of an SRAM to near- or sub-threshold voltage;
- Extending the refresh period of a DRAM.

Semiconductor memories are ubiquitous. Therefore, reliability and data integrity of the RAM in safety critical systems are of the utmost importance. Unreliable memory in those systems not only may lead to simple crashes but also may lead to injuries and even deaths. Toyota's sudden unintended acceleration case is a sad but a good example. It turned out, Toyota faced wrongful death and personal injury lawsuits [1]. Single bit flip caused the loss of pedal control and freezed the car in acceleration mode [2].

To improve data reliability, I propose to add automatic error detection and correction module to existing OpenRAM infrastructure. I propose to implement **S**ingle **E**rror **C**orrection **D**ouble **E**rror **D**etection (SECDED) algorithm. The algorithm is a model of Hamming Code invented by Richard Hamming. SECDED adds multiple parity bits to the data word when it is written to the memory. Using these parity bits, SECDED can detect single and double bit error when reading data from the memory and also correct single bit errors.

Implementation

Figure 1 depicts the block diagram of an existing OpenRAM project. I propose to introduce the following changes to the current design:

- increase the width of the bit cell array due to additional parity bits
- add Read-SECDED circuitry after Sense Amp Array and Write-SECDED circuitry before Write Driver Array (refer to Figure 2).

OpenRAM is a memory compiler and data word width depends on the user input. However, let's consider specific example to understand the operation of the algorithm more clearly. Let's consider the case when data word width is 16 bits.

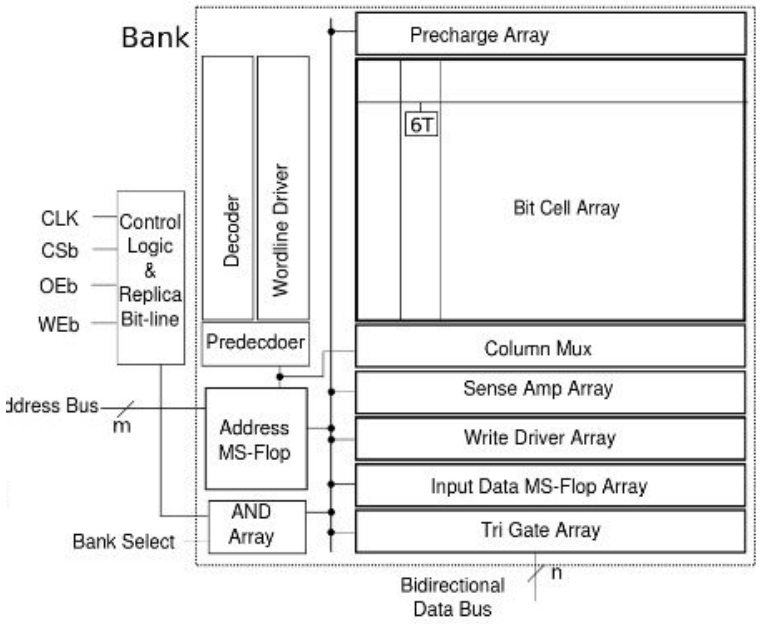

Figure 1 - Block Diagram of OpenRAM

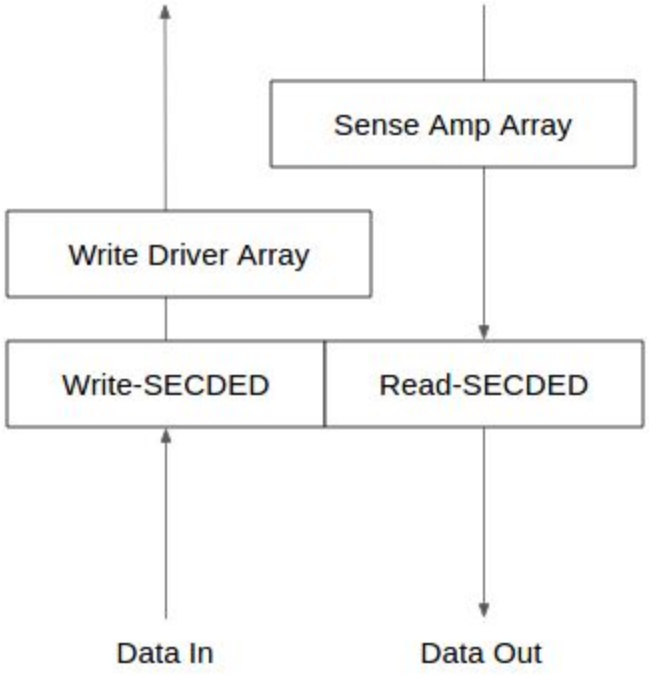

Figure 2 - Proposed Features

21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 1 - Data word after adding parity bits

Table 1 illustrates the data word after the SECDED translation. 16 bit data is translated into 22 bits. Six additional parity bits are added. According to the original Hamming Code algorithm, those bit-positions that represent powers of two dedicated for parity bits. They are shown as red in the table. First five parity bits are created for single detection single correction, and calculated as shown in Table 2. Numbers in the table represent bit positions.

0	$2 \oplus 4 \oplus 6 \oplus 8 \oplus 10 \oplus 12 \oplus 14 \oplus 16 \oplus 18 \oplus 20$
1	$2 \oplus 5 \oplus 6 \oplus 9 \oplus 10 \oplus 13 \oplus 14 \oplus 17 \oplus 18$
3	$4 \oplus 5 \oplus 6 \oplus 12 \oplus 13 \oplus 14 \oplus 19 \oplus 20$
7	$8 \oplus 9 \oplus 10 \oplus 11 \oplus 12 \oplus 13 \oplus 14$
15	$16 \oplus 17 \oplus 18 \oplus 18 \oplus 20$

Table 2 - Parity calculation

Hamming code can be designed for any data word width. The steps described above can be generalized as follows:

1. Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, ...). All other bits are dedicated for data.
2. The position of the parity bit determines the sequence of bits that it alternately checks and skips:
    a. Parity 1 (position 0): check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (0, 2, 4, 6, 8, ...).
    b. Parity 2 (position 1): check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (1,2,5,6,9,10).
    c. Continue as much as needed.
3. Set a parity bit to 1 if the total number of ones in the positions it checks is odd (XOR).

As you might noticed, bit position 21 is not a power of two. One additional parity bit detects double errors that are not correctable. This extra parity bit is an overall parity bit and is calculated by XOR-ing all the data bits and parity bits. Figure 3 shows the block diagram of Write-SECDED module.

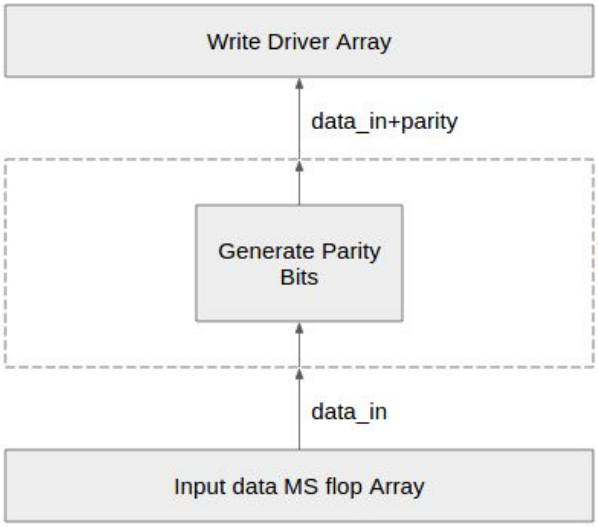

Figure 3 - Block diagram of Write-SECDED module

The error detection/corrections happens when data is read from the memory. Refer to Figure 4 to see the block diagram of Read-SECDED. The module checks for error by XOR-ing previously calculated parity bits with freshly calculated parity bits generated from the data stored in the memory. In literature, the calculated value is referred to as “Syndrome”. The value of a syndrome will indicate the position of the bit flip if a single error has occurred.

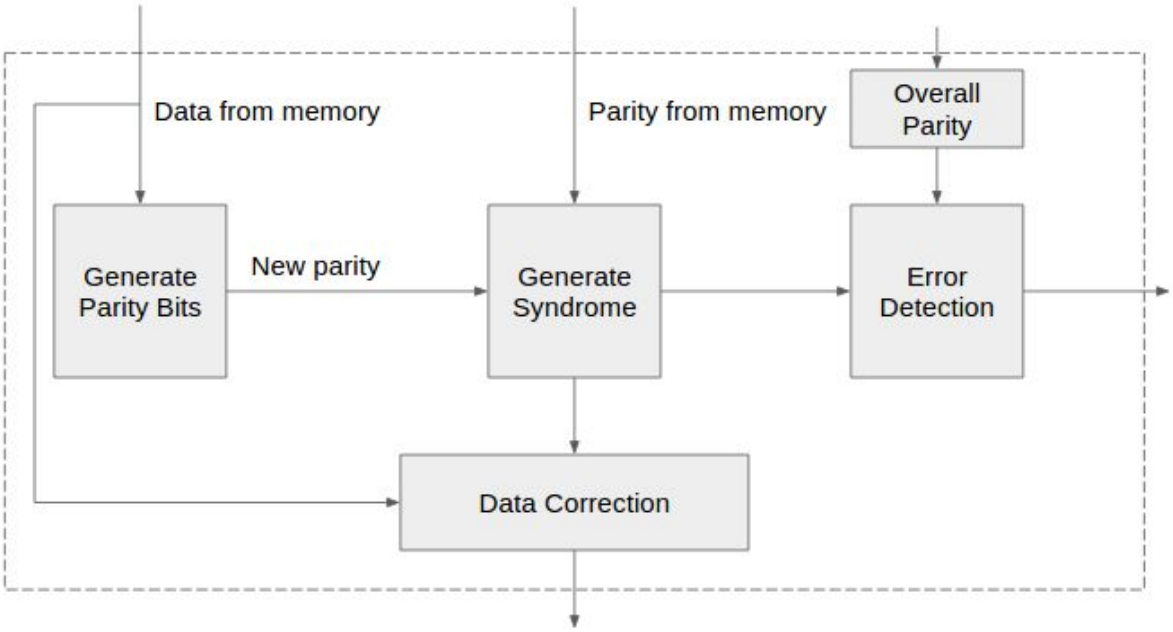

Figure 4 - Block diagram of Read-SECDED module

Plan

Table 3 illustrates rough project schedule for the Spring 2017 quarter. Code and Test rows will be splitted into many specific subtasks within 2 weeks. Right now, they are generalized due to the lack of detailed knowledge about OpenRAM system.

	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Finals week
Finalize the proposal										
Get acquainted with OpenRAM										
Read OpenRAM paper and presentation										
Run OpenRAM										
Get acquainted with the code										
Talk to Professor Matthew and ask for feedback										
Code										
Test										
Presentation										

Table 3 - Project Schedule

References

[1] https://users.ece.cmu.edu/~koopman/pubs/koopman14_toyota_ua_slides.pdf
[2] http://www.safetyresearch.net/Library/BarrSlides_FINAL_SCRUBBED.pdf