**Towards Developing a limited Magnitude Error Correction Methodology**

Multilevel cell (MLC) memories have been advocated for increasing density at low cost in next generation memories. Several bits in a cell can reduce the distance between levels. So the sorted margin makes the margin more vulnerable to defective and that’s leading to an error in store data. To protect the memories from such errors and ensure that the stored data is not corrupted, Error correction codes (ECC) are commonly used. The main scheme is mainly based by combining ECCs that are commonly used to protect traditional memories. The IP (interleaved parity) bits identifying the remaining erroneous bits in the memory cell. The scheme is also competitive in terms of number of parity check bits and memory redundancy. [Multilevel cell (MLC) memory depend mostly on the mapping of levels to bits and the magnitude of the errors.Magnitude-1 errors affect a single bit and thus they can be corrected by an SEC code. Magnitude-2 errors only affect two bits. Magnitude-1 errors always corrupt the lowest bit (the second lowest bits) of the memory cell, only a single parity bit (two parity bits) that covers all lowest bits. The magnitude and sign of the error are known properly then it can be corrected with relatively simple code. Basically One-Bit Parity (OBP) scheme that can detect any magnitude-1 error and the Two-Bit Parity (TBP) scheme that can detect any magnitude-2 error. So the efficient scheme that has magnitude errors correction capability for MLC memories.SEC-DAEC code in the two lowest bits of cell with the IP bits to correct up magnitude-3 errors.](#_Toc63085786)

In this semester, i have completed the SEC-DAEC parity counting and use them for find out the errors. In future i have plan to solve the error for 64-bit datawords. And also will try to improve them with proper error correction code. So that we can optimize the MLC memory properly and make a better and more efficient method for MLC.

Summary Presented By

Mahmudur Rahman Limon (1607049)