ECC Implementation to Reduce Accumulation of Memory and Interface Channel Errors

Dinesh Maheshwari
CTO, Memory Products Division
Cypress Semiconductor
October 24th, 2013
ECC is required to drive SRAM SER to 0.

Raw SER for SRAM is reducing with process node but is not ~0.

ECC is needed to reduce it further.
ASIC/ASSP/NPU performs SEC-DED ECC on data using extra bits

Memory is oblivious of ECC bits – except for wider word

The most common technique is SEC-DED Hamming code, eg. 64b + 8b
SRAM raw SER still ~200 FIT/Mb

Interface speeds are increasing → interface BER is increasing

Memory and interface BER accumulate but are non-correlated
   - Even MBU are not correctable
   - Odd MBU cause silent data corruption (SDC)

There is no clear indication on memory failure in field
Separate interface errors from memory errors

Both ASIC/ASSP/NPU and memory perform ECC checks and correction on memory transactions

Requires ASIC/ASSP/NPU and memory to use the same ECC algorithms and ECC syndrome bit placement
Next-gen Memory ECC – Write

ASIC/ASSP/NPU generates ECC bits and writes data and ECC bits to memory

Memory does ECC check

- If no error, writes to array
- If correctable error, corrects, writes to array and logs
- If uncorrectable error, flags to ASIC and logs
Memory reads from array and does ECC

- If no error, forwards to ASIC
- If correctable error, corrects, forwards to ASIC and logs
- If uncorrectable error, flags to ASIC as error and logs

ASIC receives data and performs ECC check and correction for interface errors
Memory ECC

### Traditional approach

**Pros**
- No access latency impact
- ASIC can choose ECC scheme

**Cons**
- Memory & I/F errors can accumulate – more SDC
- No intimation of memory failure in field

### Next-gen approach

**Pros**
- Memory & I/F errors corrected independently – less SDC
- Intimation of failing memory

**Cons**
- Requires ASIC and memory to use same ECC scheme
- Some latency impact