ECE 571

Intro to system Verilog

Final Term Project

Design and verification of ECC

SECDED (single bit error correction and double bit error detection)

Project 10

1. KOLLA BALAJI
2. PREM DEEP MAHANTI
3. PAVAN KUMAR BRUNDHAVANAM
4. RAHUL ADDEPALLY

Git hub:

GIT_LINK

Error Correction Code (ECC) memory systems are one of the most often utilized RAS techniques in the memory subsystem. By generating ECC SECDED (Single bit Error Correction and Double-bit Error Detection) codes for the actual data and storing them in additional DRAM storage, the DDR controller can correct single bit errors and detect double-bit errors on the received data from the DRAMs.

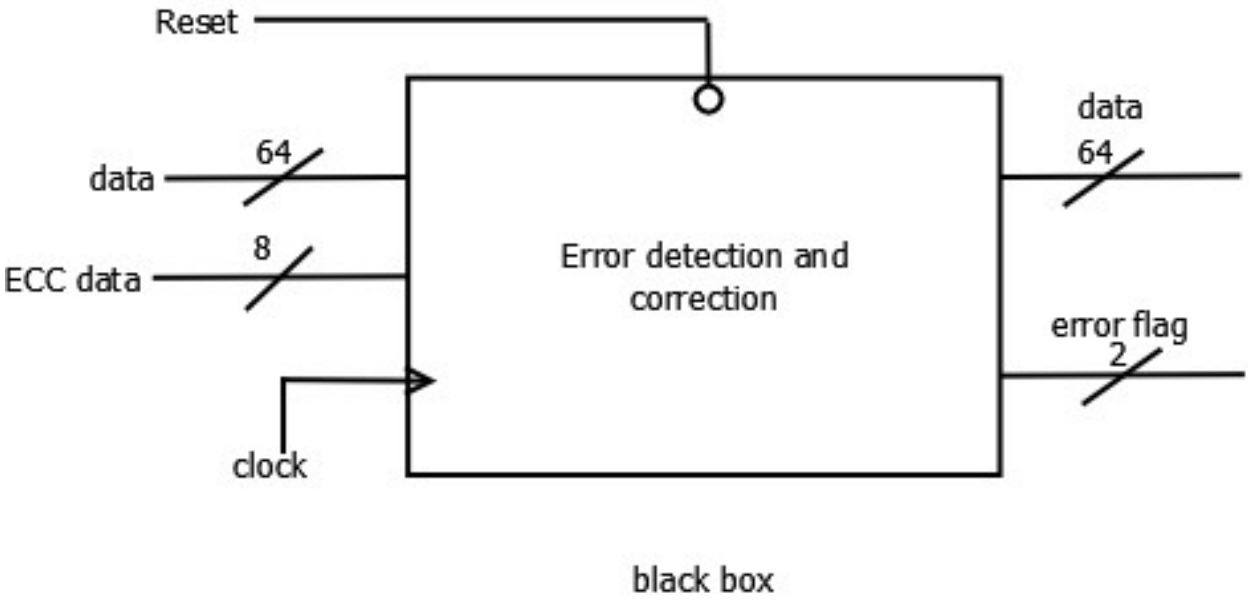
The ECC generation and check sequence is as follows:

- The ECC codes are generated by the controller based on the actual WR (WRITE) data. The memory stores both the WR data and the ECC code.
- During a RD (READ) operation, the controller reads both the data and respective ECC code from the memory. The controller regenerates the ECC code from the received data and compares it against the received ECC code.
- If there is a match, then no errors have occurred. If there are mismatches, the ECC SECDED mechanism allows the controller to correct any single-bit error and detect double-bit errors. Such an ECC scheme provides an end-to-end protection against single-bit errors that can occur anywhere in the memory subsystem between the controller and the memory.

Design decision:

In this project we designed single bit error correction and double bit error detection.

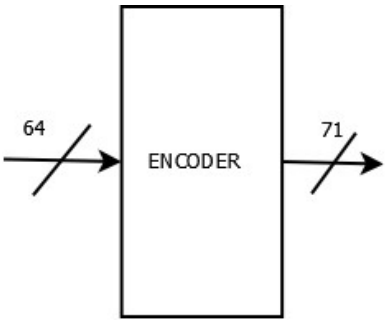
Black box is as shown below:



black box

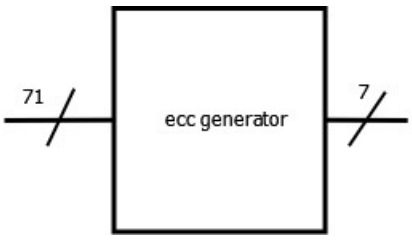
Blocks in black box:

Encoder block:



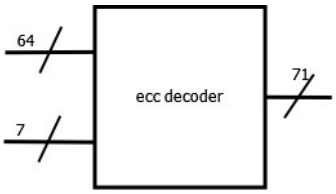
In this block 64-bit input data is converted in to 72-bit code word data, which means data is placed in the non 2 power locations.

Ecc-generator block



This block calculates the R values by using the 72-bit codeword.

Ecc-decoder:



In this block decoder place all the ecc bits in the 2 power locations.

Decoder block

Based on the R values and ecc_data, syndrome is calculated. parity(R0) is calculated from the old and new data.

Single bit error correction and double bit error detection are performed based on the R0 values and syndrome.

If (syndrome == 0 && R0_old == R0_new)

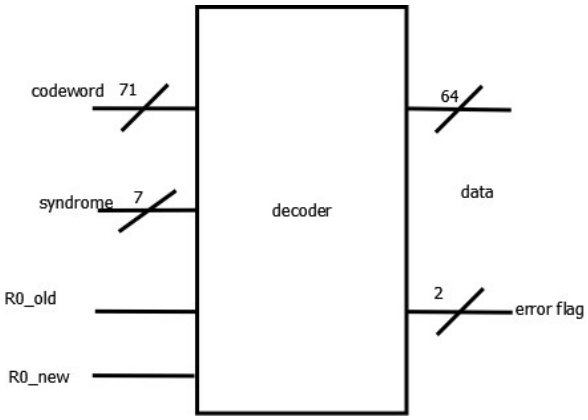
No error in the data then error flag is 00.

If (syndrome! = 0 && R0_old != R0_new)

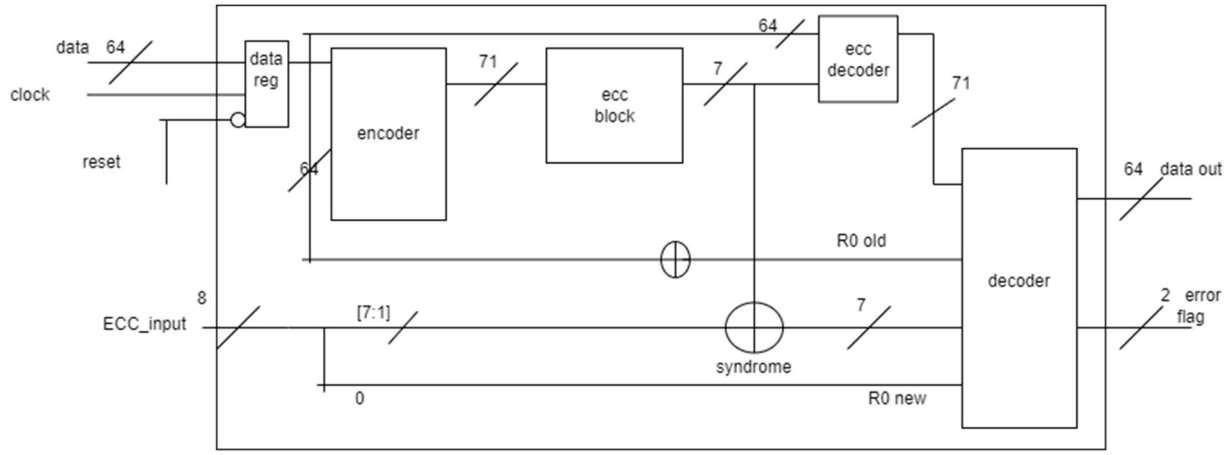
Single bit error is detected and corrected and then error flag is 01.

If (syndrome! = 0 && R0_old == R0_new)

Double bit error is detected and then error flag is 10.



Block diagram:



Verification Test Plan:

Assertions:

Test case 1:

Assertion to verify the input data valid after reset is asserted.

Test case 2:

Assertion to verify the no error in data.

Test case 3:

Assertion to verify the single bit error in data.

Test case 4:

Assertion to verify the double bit error in data.

Constrained randomization:

Weighted distribution is used to give the error probability to the injection module for the single bit error case, double bit error case and no error case.

Coverage report:

```
Coverage Report Summary Data by file

================================================================================
=== File: ECC.sv
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        11        10         1    90.90%
    Conditions                       3         2         1    66.66%
    Expressions                    205       205         0   100.00%
    Statements                      26        26         0   100.00%
    Toggles                        462       432        30    93.50%

================================================================================
=== File: Golden_Ecc.sv
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        17        16         1    94.11%
    Conditions                       5         4         1    80.00%
    Expressions                      2         2         0   100.00%
    Statements                      50        50         0   100.00%
    Toggles                        746       572       174    76.67%

================================================================================
=== File: Testbench.sv
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        22        19         3    86.36%
    Conditions                      11         9         2    81.81%
    Expressions                      2         2         0   100.00%
    Statements                      78        70         8    89.74%
    Toggles                        986       718       268    72.81%


TOTAL ASSERTION COVERAGE: 100.00%  ASSERTIONS: 7

Total Coverage By File (code coverage only, filtered view): 88.44%
```