ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

*International Advanced Research Journal in Science, Engineering and Technology*
*Vol. 2, Issue 7, July 2015*

# Very Low Complexity Low Latency Architecture for Data Encoding Hard Synthetic Error Correcting Code

**S.Anjaneyulu[1], K. Radhika Reddy[2]**

M. Tech, JITS, Karimnagar[1]

JITS, Karimnagar[2]

**Abstract:** In current scenario, there are situations in a computing system where incoming information needs to be compared with a piece of stored data to locate the matching entry, e.g., cache tag array lookup and translation look-aside buffer matching. Nowadays the reliability issues of memory are Event Upsets (EUs), which are able to invert the stored logical value in memory cells. This issue is more serious when the affected memory cells are part of the configuration memory used for programming the circuit functionality. The consequences may be alterations of the circuit functionality causing errors which may only be corrected by reprogramming the device**.** A new architecture for matching the data protected with an error-correcting code (ECC) is proposed in brief to reduce latency and complexity. The proposed architecture is based on the fact that the codeword of an ECC is usually represented in a systematic form consisting of the raw data and the parity information generated by encoding, and the proposed architecture parallelizes the comparison of the data and that of the parity information. To further reduce the latency and complexity, in addition, a new butterfly-formed weight accumulator (BWA) is proposed for the efficient computation of the Hamming distance. Grounded on the BWA, the proposed architecture examines whether the incoming data matches the stored data, and if not it aims to locate the erroneous bit and they are corrected. The empirical evaluation proves that the proposed methodology discovers the best service for reliability issues of memory.

**Keywords:** Butterfly-Formed Weight Accumulator, Translation Look-Aside Buffer, ECC, EDC, Decimal Matrix Code

## I.INTRODUCTION

Data comparison circuit is a logic that has many applications in a computing system. For example, to check whether a piece of information is in a cache, the address of the information in the memory is compared to all cache tags in the same set that might contain that address. Another place that uses a data comparison circuit is in the translation look-aside buffer (TLB) unit. TLB is used to speed up virtual to physical address translation. Error correcting codes (ECCs) are widely used in modern microprocessors to enhance the reliability and data integrity of their memory structures. Several error detecting codes (EDCs) and error correcting codes (ECCs) have been proposed so far to improve cache reliability. They range from the simple parity check code to the more complex Single Error Correcting/Double Error Detecting (SEC/DED) ECC (used to protect the L2 and L3 caches in the Itanium microprocessor.

## II. DATA COMPARISION METHODS

### 2.1 Decode-And-Compare Architecture

This describes the conventional decode-and-compare architecture. Let us consider a cache memory where a k-bit tag is stored in the form of an n-bit codeword after being encoded by a (n, k) code. In the decode-and-compare architecture, the n-bit retrieved codeword should first be decoded to extract the original k-bit tag. The extracted k-bit tag is then compared with the k-bit tag field of an incoming address to determine whether the tags are matched or not. As the retrieved codeword should go through the decoder before being compared with the incoming tag, the critical path is too long to be employed in a practical cache system designed for high-speed access

### 2.2 Direct Compare Method

Direct compare method is one of the most recent solutions for the matching problem. The direct compare method encodes the incoming data and then compares it with the retrieved data that has been encoded as well. Therefore, the method eliminates the complex decoding from the critical path.

### 2.3 Sa-Based Approach

SA-based approach is the one where a special counter is constructed with an additional` building block called saturating adder (SA).The SA-based direct compare architecture reduces the latency and hardware complexity by resolving the aforementioned drawbacks.

## III. ADVANCED DATA COMPARISION METHODS

### 3.1 DMC Encoding

Because of high-speed caches and main memories, which are prone to soft errors, error correcting codes are used in the design and, more recently, in the design of on chip memories. For the encoding Decimal matrix code (DMC) is proposed to assure reliability in the presence of MCUs with reduced performance overheads, and a 4-bit word is encoded based on the proposed technique.

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

*International Advanced Research Journal in Science, Engineering and Technology*
*Vol. 2, Issue 7, July 2015*

- First, during the encoding process, information bits i are fed to the DMC encoder, and then the horizontal redundant bits H and vertical redundant bits V are obtained from the DMC encoder. When the encoding process is completed, the obtained DMC codeword is stored in the memory.
- Second, the horizontal redundant bits H are produced by performing xor operation of selected symbols per row.
- Third, the vertical redundant bits V are obtained by xor operation among the bits per column.

It should be noted that both divide-symbol and arrange-matrix are implemented in logical instead of in physical. Therefore, the proposed DMC does not require changing the physical structure of the memory.

The proposed DMC scheme, for a 4-bit word is as shown in Figure.

| | | |
|---|---|---|
| **i1** | **i0** | **H0** |
| **i3** | **i2** | **H1** |
| **V1** | **V0** | |

**4-bit DMC logical organization**

In the above figure the cells from i0 to i3 are information bits. The 4-bit word has been divided into two symbols of 2-bit.k1 = 2 and k2 = 2 have been chosen simultaneously.H0 and H1 are horizontal check bits;V0 and V1 are vertical check bits. The horizontal bits H can be obtained as follows:

$$H0 = i0 \oplus i1$$

$$H1 = i2 \oplus i3$$

For the vertical bits V, we have

$$V0 = i0 \oplus i2$$

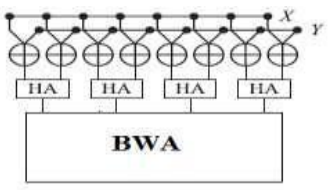
$$V1 = i1 \oplus i3$$

The obtained parity bit is appended with the information bits so as to obtain the encoded bit.

### 3.2 XOR Bank

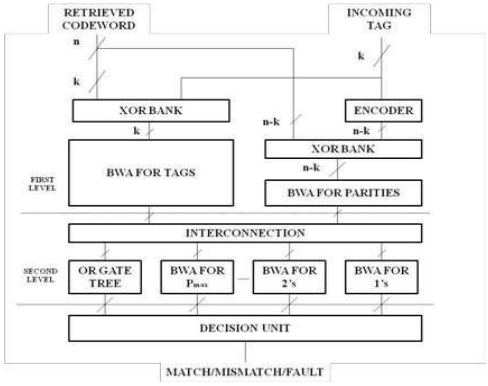
Xor bank represents the array of bit-wise comparators (exclusive OR gates). It performs XOR operations for every pair of bits in X and Y so as to generate a vector representing the bitwise difference of the two codewords. The output from the XOR bank is then fed into BWA consisting of half adders (HAs). The numbers of 1's are accumulated by passing the value through the BWA.

XOR bank structure for (8, 4) code



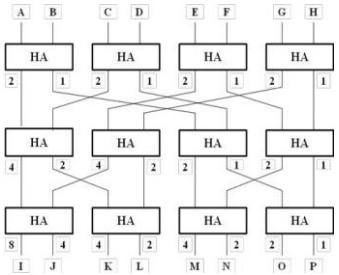

### 3.3 Butterfly Formed Weight Accumulator

The proposed architecture grounded on the data path design is given below. It contains multiple butterfly-formed weight accumulators (BWAs) proposed to improve the latency and complexity of the Hamming distance computation. The basic function of the BWA is to count the number of 1's among its input bits.




**Proposed data path design**

The proposed architecture consists of multiple stages of HAs as shown in figure where each output bit of a HA is associated with a weight. The HAs in a stage are connected in a butterfly form so as to accumulate the carry bits and the sum bits of the upper stage separately. In other words, both inputs of a HA in a stage, except the first stage, are either carry bits or sum bits computed in the upper stage. This connection method leads to a property that if an output bit of a HA is set, the number of 1's among the bits in the paths reaching the HA is equal to the weight of the output bit.




**General structure of BWA**

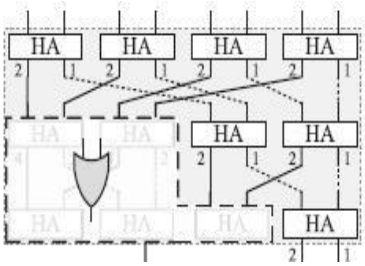
In above figure for example, if the carry bit of the gray-colored HA is set, the number of 1's among the associated input bits, i.e., A, B, C, and D, is 2. At the last stage of above figure the number of 1's among the input bits, d, can be calculated as

$$d = 8I + 4(J + K + M) + 2(L + N + O) + P$$

Since what we need is not the precise Hamming distance but the range it belongs to, it is possible to simplify the circuit. When rmax = 1, for example, two or more than two 1's among the input bits can be regarded as the same case that falls in the fourth range. In that case, we can

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
*International Advanced Research Journal in Science, Engineering and Technology*
*Vol. 2, Issue 7, July 2015*

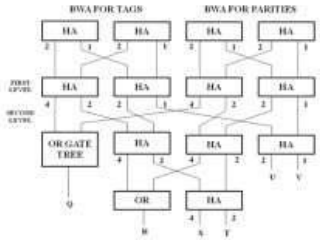
replace several HAs with a simple OR-gate tree as shown below. This is an advantage over the SA that resorts to the compulsory saturation.




**Revised structure with OR-gate tree**

Each XOR stage generates the bitwise difference vector for either data bits or parity bits, and the following processing elements count the number of 1's in the vector, i.e., the Hamming distance. Each BWA at the first level is in the revised form shown in figure above, and generates an output from the OR-gate tree and several weight bits from the HA trees. In the interconnection, such outputs are fed into their associated processing elements at the second level.

The output of the OR-gate tree is connected to the subsequent OR-gate tree at the second level, and the remaining weight bits are connected to the second level BWAs according to their weights.More precisely, the bits of weight w are connected to the BWA responsible for w-weight inputs. Each BWA at the second level is associated with a weight of a power of two that is less than or equal to Pmax, where Pmax is the largest power. As the weight bits associated with the fourth range are all ORed in the revised BWAs, there is no need to deal with the powers of two that are larger than Pmax. A simple (8, 4) single-error correction double-error detection code is considered and the corresponding first and second level circuits are shown below.




**First and second level circuits for (8,4) code**

### 3.4 Decision Unit

Taking the outputs of the preceding circuits (BWA), the decision unit finally determines the incoming tag matches the retrieved codeword by considering the four ranges of the Hamming distance. The decision unit is infact a combinational logic of which functionality is specified by a truth table that takes the outputs of the preceding circuits as inputs. For the (8, 4) code that the corresponding first and second level circuits are given above, the truth table for the decision unit is described in Table I. Since U and V cannot be set simultaneously,

### 3.5 Error Deduction and Correction

Decimal error deduction technique is proposed and it has several advantages over the simple binary error deduction technique. The Limits of Simple Binary Error Detection can be given as follows

- It requires low redundant bits; its error detection capability is limited. The main reason for this is that its error detection mechanism is based on binary.
- The number of even bit errors cannot be detected.
- Can detect only a finite number of errors finite number of errors

However, when the decimal algorithm is used to detect errors, these errors can be detected so that the decoding error can be avoided. The reason is that the operation mechanism of decimal algorithm is different from that of binary. First of all, the horizontal redundant bits H1 H0 are obtained from the original information bits. When MCUs occur in symbols, i.e., the bits in symbols are upset to "1" from "0" or vice versa.

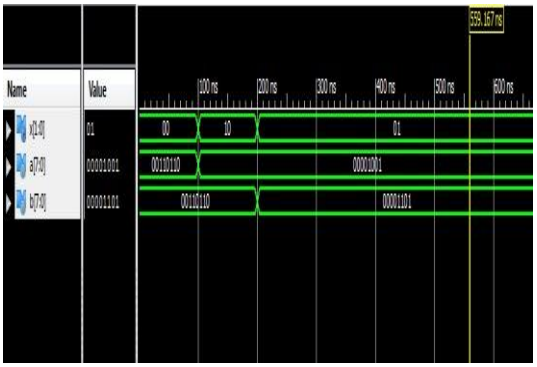
The proposed DMC can easily correct upsets of the following types

- Type 1 is a single error
- Type 2 is an inconsecutive error in two consecutive symbols
- Type 3 is a consecutive error in two consecutive symbols
- Type 4 is an inconsecutive error in two inconsecutive symbols
- Type 5 is a consecutive error in four consecutive symbols z

## IV. Simulation Results

ISim provides a complete, full-featured HDL simulator integrated within ISE. HDL simulation now can be an even more fundamental step within your design flow with the tight integration of the ISim within your design environment. The Xilinx® ISE Simulator (ISim) is a Hardware Description Language (HDL) simulator that enables you to perform functional (behavioural) and timing simulations for VHDL, Verilog and mixed language designs.

### 4.1 Proposed Architecture




### 4.2 Synthesis Result

To investigate the advantages of using our technique in terms of area we implemented and synthesized for a Xilinx

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
*International Advanced Research Journal in Science, Engineering and Technology*
*Vol. 2, Issue 7, July 2015*

XC3S500E different versions of a32-bit, 32-entry, dual read ports, single write port register file.
This device utilization includes the following.

- Logic Utilization
- Logic Distribution
- Total Gate count for the Design

The device utilization summery is shown below, which gives the details of number of devices used from the available devices and also represented in 1%. Hence as the result of the synthesis process, the device utilization in the used device and package is shown for Proposed Architecture.

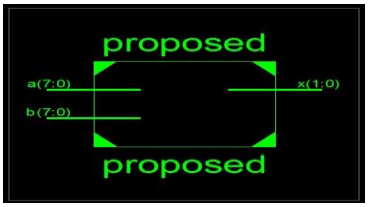
| proposed Project Status | | | |
|---|---|---|---|
| **Project File:** | ffa1.xise | **Parser Errors:** | No Errors |
| **Module Name:** | proposed | **Implementation State:** | Placed and Routed |
| **Target Device:** | xc3s500e-5fg320 | **• Errors:** | No Errors |
| **Product Version:** | ISE 13.2 | **• Warnings:** | 6 Warnings (6 new) |
| **Design Goal:** | Balanced | **• Routing Results:** | All Signals Completely Routed |
| **Design Strategy:** | Xilinx Default (unlocked) | **• Timing Constraints:** | |
| **Environment:** | System Settings | **• Final Timing Score:** | 0 (Timing Report) |

| Device Utilization Summary [-] | | | | |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of 4 input LUTs | 18 | 9,312 | 1% | |
| Number of occupied Slices | 11 | 4,656 | 1% | |
| Number of Slices containing only related logic | 11 | 11 | 100% | |
| Number of Slices containing unrelated logic | 0 | 11 | 0% | |
| Total Number of 4 input LUTs | 18 | 9,312 | 1% | |
| Number of bonded IOBs | 18 | 232 | 7% | |
| Average Fanout of Non-Clock Nets | 1.85 | | | |

Figure Device utilization summary of Proposed Architecture

### 4.3 RTL Schematic

The RTL (Register Transfer Logic) can be viewed as blue box after synthesize of design is made. It shows the inputs and outputs of the system. By double-clicking on the diagram we can see gates, flip-flops and MUX. The corresponding schematics of the proposed architecture after synthesis are shown below. The top level of Proposed Architecture has the inputs A and B which are of 8 bit each, x as output which as 2bit.




Top-level of proposed Architecture

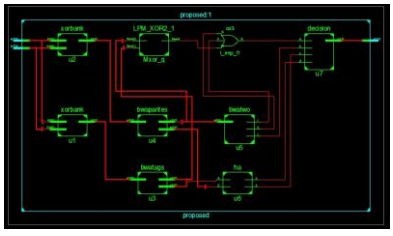


Figure 4: Internal block of proposed Architecture

The internal block of ripple carry adder shows inputs and outputs as well as the intermediate gates and their interconnection between them. This internal block contains logic gates, flip flops, etc.

## V. CONCLUSION AND FUTURE ENHANCEMENT

To reduce the latency and hardware complexity, a new architecture has been presented for matching the data protected with an ECC. The proposed architecture examines whether the incoming data matches the stored data if a certain number of erroneous bits are corrected. To reduce the latency, the comparison of the data is parallelized with the encoding process that generates the parity information. The parallel operations are enabled based on the fact that the systematic codeword has separate fields for the data and parity. In addition, an efficient Processing architecture has been presented to further minimize the latency and complexity. As the proposed architecture is effective in reducing the latency as well as the complexity considerably, it can be regarded as a promising solution for the comparison of ECC-protected data. Though this brief focuses only on the tag match of a cache memory, the proposed method is applicable to diverse applications that need such comparison.

## REFERENCE

1. Jing Guo, Liyi Xiao, Zhigang Mao and Qiang Zhao, **"**Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code", IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 22, No. 1, January 2014.
2. D. Radaelli, H. Puchner, S. Wong, and S. Daniel, "Investigation of multi-bit upsets in a 150 nm technology SRAM device," IEEE Trans. Nucl. Sci., vol. 52, no. 6, pp. 2433–2437, Dec. 2005.
3. E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on induced soft error in SRAMs from an 250 nm to a 22 nm design rule," IEEE Trans. Electron Devices, vol. 57, no. 7, pp. 1527– 1538, Jul. 2010.
4. S. Liu, P. Reviriego, and J. A. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 1, pp. 148–156, Jan. 2012.
5. P. Reviriego, M. Flanagan, and J. A. Maestro, "A (64,45) triple error correction code for memory applications," IEEE Trans. Device Mater. Rel., vol. 12, no. 1, pp. 101–106, Mar. 2012.
6. S. Baeg, S. Wen, and R. Wong, "Interleaving distance selection with a soft error failure model," IEEE Trans. Nucl. Sci., vol. 56, no. 4, pp. 2111–2118, Aug. 2009.


DOI 10.17148/IARJSET.2015.2715