IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

SYSTEMATIC ERROR-CORRECTING CODES IMPLEMENTATION FOR MATCHING OF DATA ENCODED

M.Naga Kalyani*, K.Priyanka

* PG Student [VLSID] Department of ECE, Shri Vishnu Engineering College for Women, Bhimavaram, India.

Assistant Professor Department of ECE, Shri Vishnu Engineering College for Women, Bhimavaram, India.

ABSTRACT

The Architecture used for matching the data protected with an error-correcting code (ecc) presented in my project to reduce the latency and complexity. The proposed architecture parallelizes the comparison of the data and that of the parity information. A new butterfly-formed weight accumulator (BWA) is proposed for the efficient computation of the Hamming distance. The basic function of the BWA is to count the number of 1's among its input bits. It consists of multiple stages of half adders. The proposed architecture checks whether the incoming data matches with the stored data.

KEYWORDS: Error-Correcting Code (ECC), Butterfly-Formed Weight Accumulator (BWA), Hamming Distance.

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.

The most recent solution for the matching problem is the direct compare method , which 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. In performing the comparison, the method does not examine whether the retrieved data is exactly the same as the incoming data. Instead, it checks if the retrieved data resides in the error correctable range of the codeword corresponding to the incoming data. As the checking necessitates an additional circuit to compute the Hamming distance, i.e., the number of different bits between the two code words, the saturate adder (SA) was presented. as a basic building block for calculating the Hamming distance. However, did not consider an important fact that may improve the effectiveness further, a practical ECC codeword is usually represented in a systematic form in which the data and parity parts are completely separated from each other. In addition, as the SA always forces its output not to be greater than the number of detectable errors by more than one, it contributes to the increase of the entire circuit complexity.

To resolve the drawbacks of the decode-and-compare architecture, therefore, the decoding of a retrieved codeword is replaced with the encoding of an incoming tag in the encode-and-compare architecture. A k-bit incoming tag is first encoded to the corresponding n-bit codeword X and compared with an n-bit retrieved codeword Y. The comparison is to examine how many bits the two code words differ, not to check if the two code words are exactly equal to each other. For this, we compute the Hamming distance d between the two code words and classify the cases according to the range of d. In the SA-based architecture, the comparison of two code words is invoked after the incoming tag is encoded. Therefore, the critical path consists of a series of the encoding and the n-bit comparison. However, it did not

consider the fact that, in practice, the ECC codeword is of a systematic form in which the data and parity parts are completely separated. As the data part of a systematic codeword is exactly the same as the incoming tag field, it is immediately available for comparison while the parity part becomes available only after the encoding is completed.

PREVIOUS DATA COMPARISION METHODS

Decode-And-Compare Architecture

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. Because the retrieved codeword should go through the decoder before compared with the incoming tag. Because the decoder is one of the most complicated processing element the complexity overhead is high.

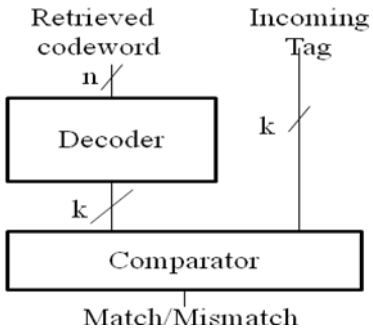

Fig 1: Decode-And-Compare Architecture

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.

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 drawbacks.

The decoding is usually more complex and takes more time than encoding as it encompasses a series of error detection or syndrome calculation, and error correction . The implementation results in support the claim. To resolve the drawbacks of the decode-and-compare architecture, therefore, the decoding of a retrieved codeword is replaced with the encoding of an incoming tag in the encode-and-compare architecture More precisely, a k-bit incoming tag is first encoded to the corresponding n bit codeword X and compared with an n-bit retrieved codeword Y. The comparison is to examine how many bits the two code words differ, not to check if the two code words are exactly equal to each other. For this, we compute the Hamming distance d between the two code words and classify the cases according to the range of d. Let t_{max} and r_{max} denote the numbers of maximally correctable and detectable errors, respectively. The cases are as follows.

In SA based the output should not to be greater than the number of errors detected by more than one. The cases are classified according to the range of d. Let t_{max} and r_{max} denote the numbers of maximum correctable and detectable errors. Some of the conditions are as follows:

CONDITION	DECISION
d=0	X matches with Y
0 <d ≤ Tmax	X will match Y provided at most Tmax errors in Y are corrected.
Tmax <d ≤ Rmax	Y has detectable but Un correctable errors
Rmax <d	X does not match with Y

Table 1: Decision Based on Hamming Distance

Assuming that the incoming address has no errors, we can regard the two tags as matched if d is in either the first or the second ranges. In this way, while maintaining the error-correcting capability, the architecture can remove the decoder from its critical path at the cost of an encoder being newly introduced. Note that the encoder is, in general, much simpler than the decoder, and thus the encoding cost is significantly less than the decoding cost. since the above method needs to compute the Hamming distance, The circuit shown in Fig. 2 first 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 code words. The following half adders (HAs) are used to count the number of 1's in two adjacent bits in the vector. The numbers of 1's are accumulated by passing through the following SA tree.

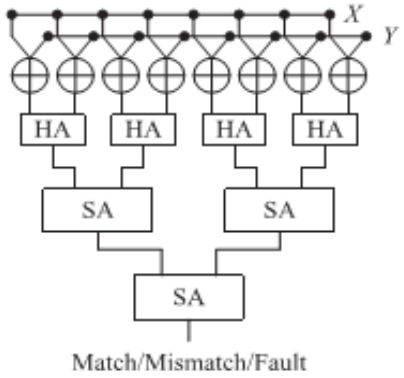

Fig 2: SA-based architecture supporting the direct compare method

PROPOSED METHODOLOGY

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.

Then the encoded data is compared with the data in the memory which can be retrieved. The XOR bank and Butterfly formed weighted accumulator is used to find the number of bit changes and to calculate the number of one's which are fed into error correction and error deduction unit. Thus the output is obtained from the decision unit.

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

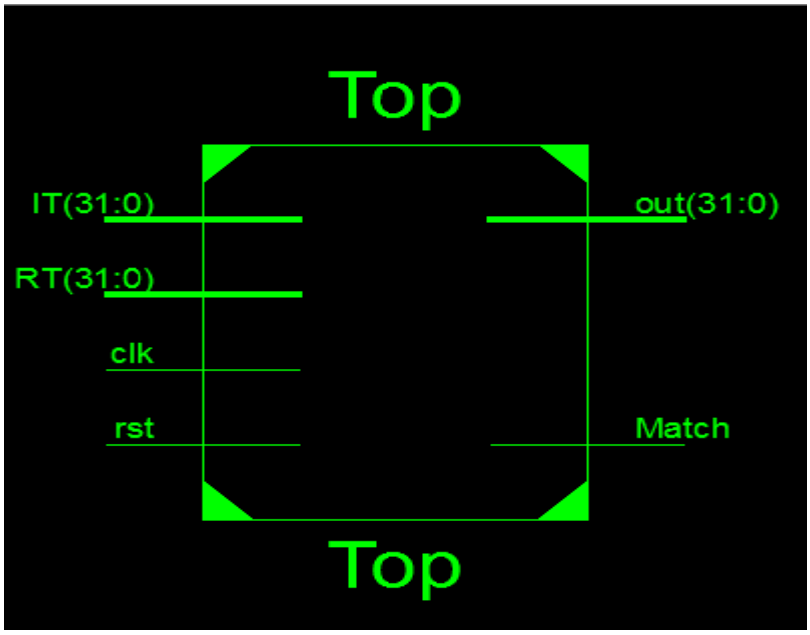

Fig 3: RTL Schematic of First Level Module

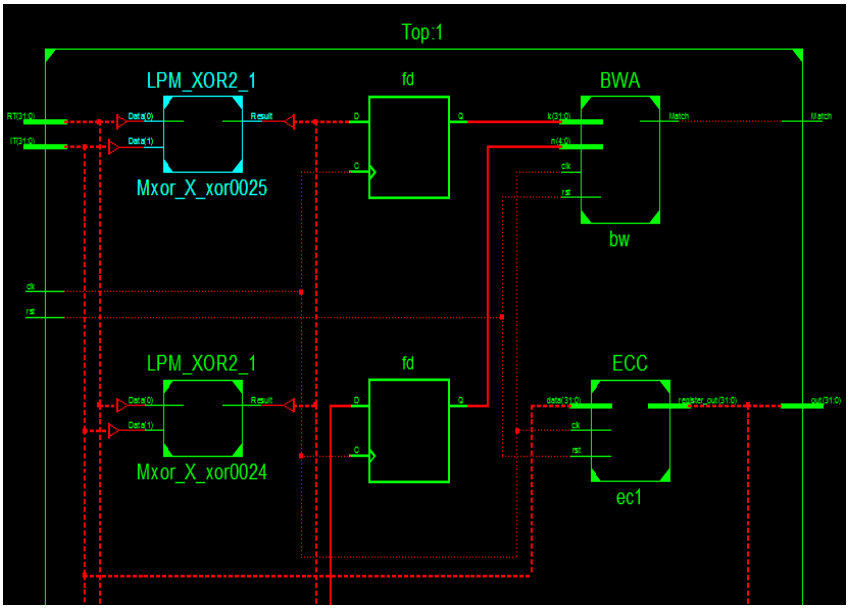

Fig 4: RTL Schematic of Second Level Module

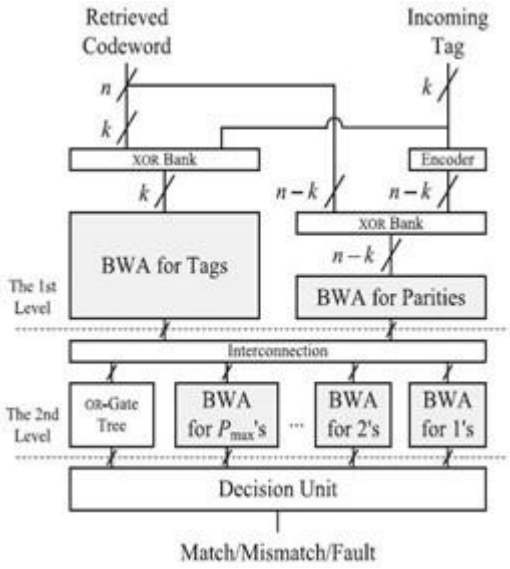

Figure 5: Architecture of Hamming Distance Computation

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.

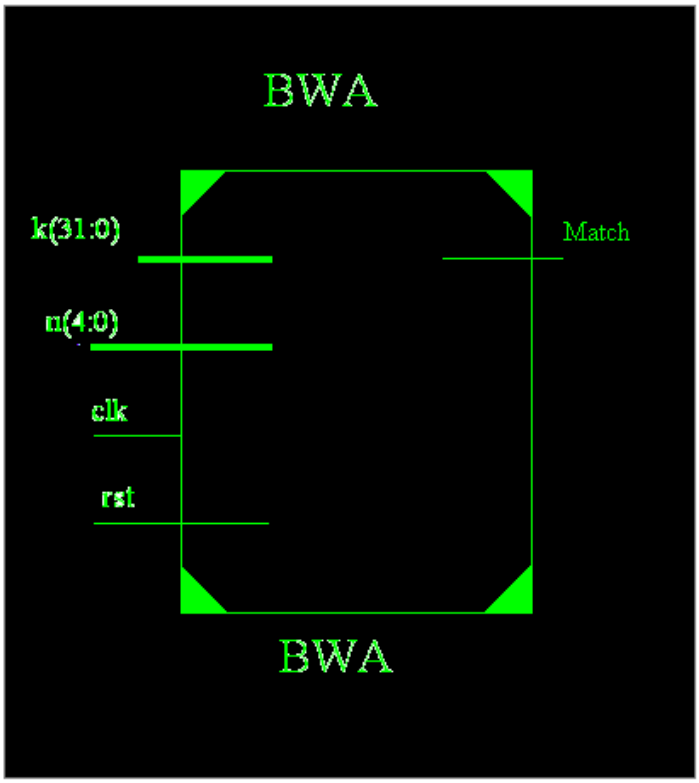

Fig 6: RTL Schematic of BWA

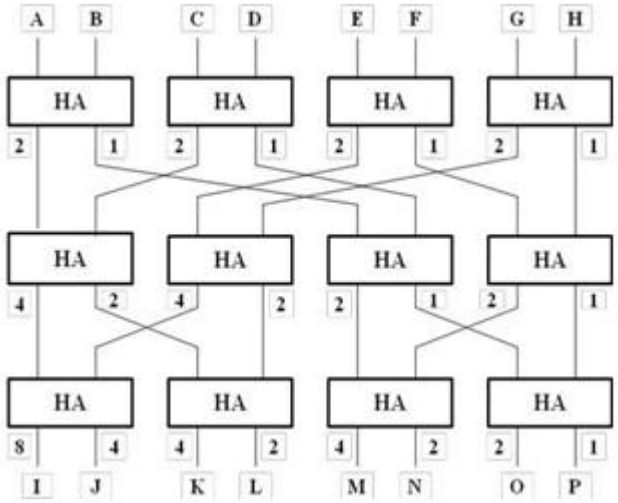

Fig 7: General structure of BWA

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 replace several HAs with a simple OR-gate tree as shown below.

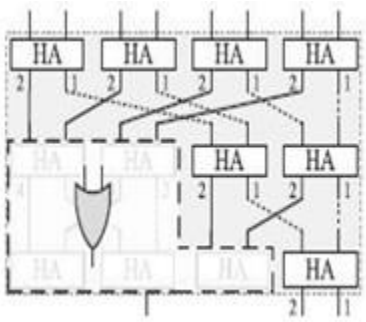

Fig 8: Revised structure with OR-gate tree

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.

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 in fact 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, such cases are implicitly included in do not care terms.

Q	R	S	T	U	V	DECISION
0	0	0	0	0	x	MATCH
0	0	0	0	1	x	FAULT
0	0	0	1	0	0	FAULT
0	0	0	1	0	1	MISMATCH
0	0	0	1	1	x	MISMATCH
1	1	1	x	x	x	MISMATCH

Table 2: Truth table of the Decision unit

SIMULATION RESULTS

In this chapter all the simulation results which are done using Xilinx ISE 12.1 are shown in below.

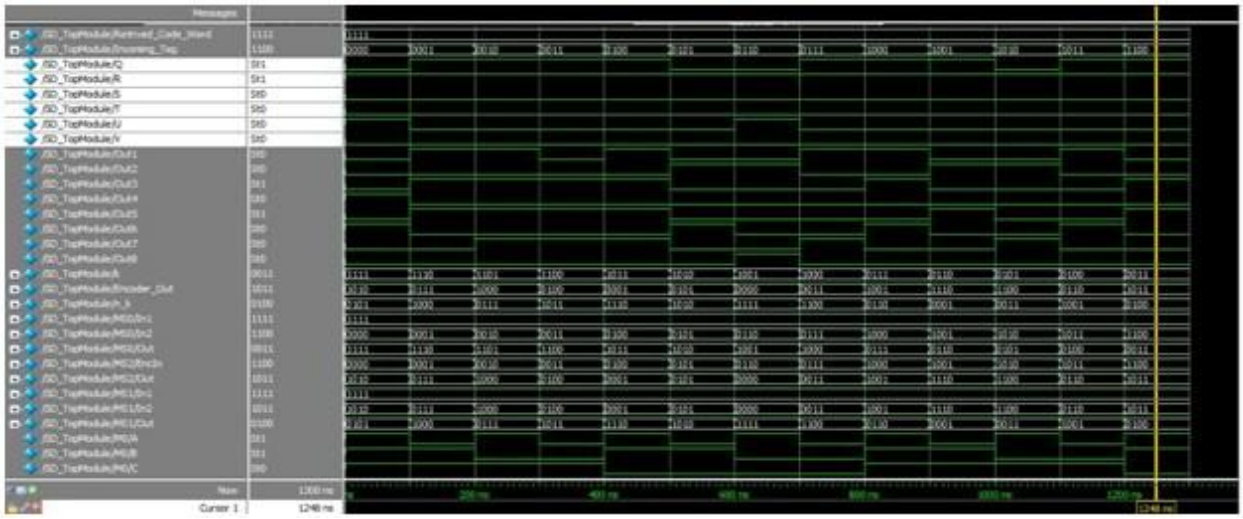

Fig.9. Simulation results of 8,4 bits

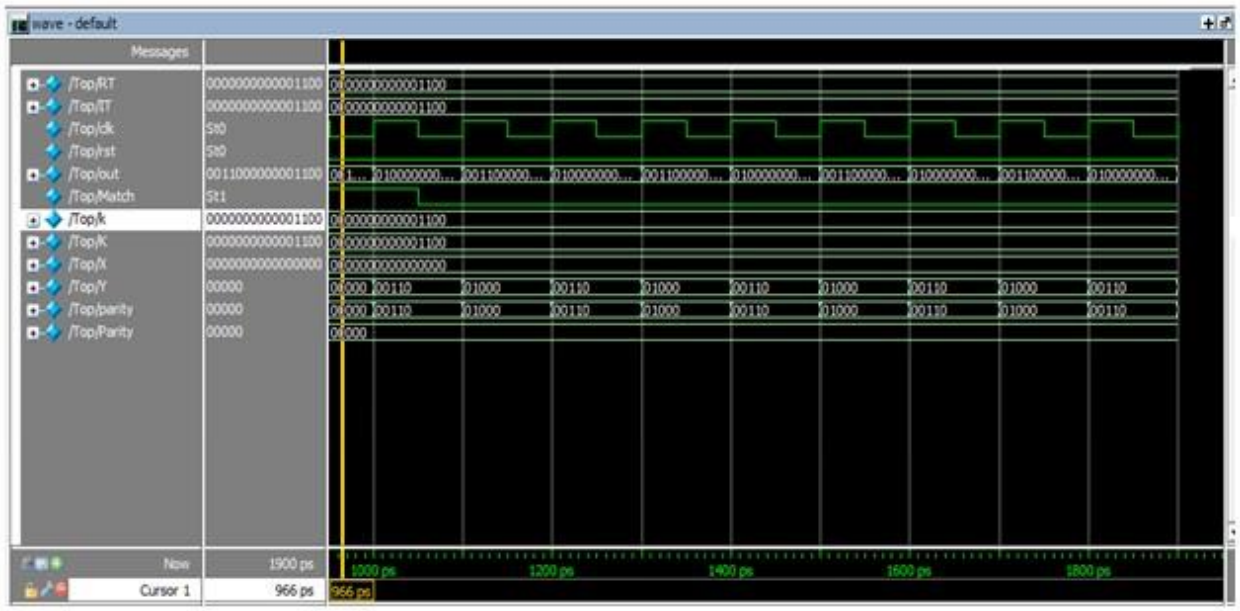

Fig 10: Simulation result of 16_8 bits

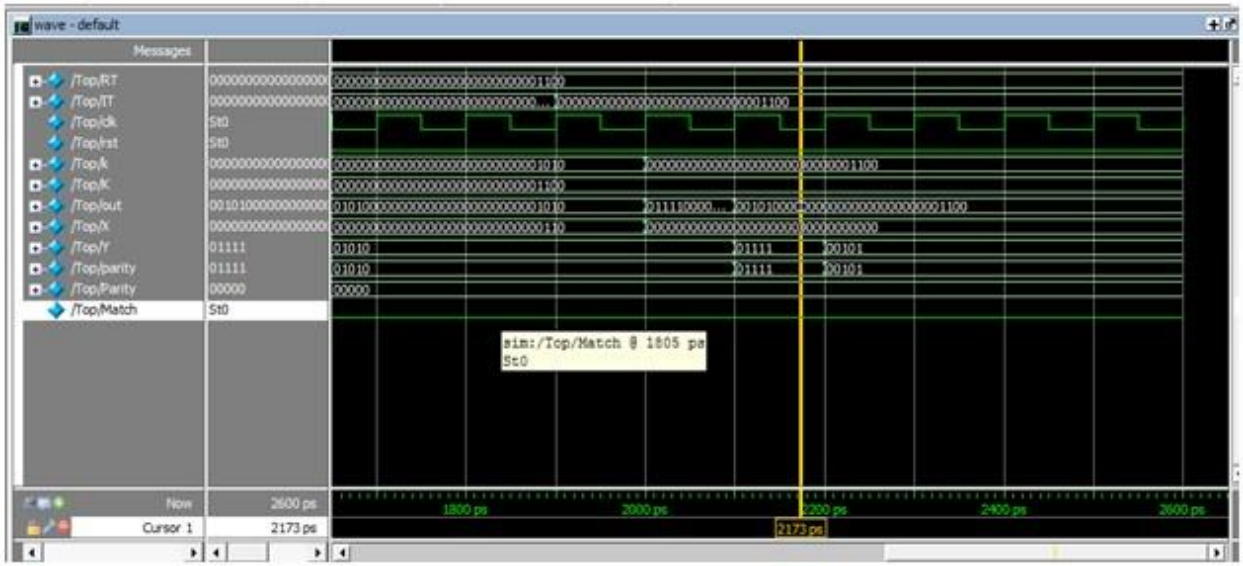

Fig 11: Simulation result of 40_33 bits

SYNTHESIS REPORT

Parameter	Exsisting Systems		Proposed Method
	Decode and Compare Architecture	Encode and Compare Architecture	
Delay	4.063 ns	4.040 ns	7.456 ns
Memory Usage	192528 kb	176656 kb	180880 kb

CONCLUSION

In this paper, 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. Pipelined Vector Precoding architecture in decoder will reduce the error rate and hardware complexity.

REFERENCES

[1] W. Wu, D. Somasekhar, and S.L. Lu, "Direct compare of information coded with error-correcting codes," IEEE Trans. Very Large Scale Integration (VLSI) Systems., vol. 20, no. 11, pp. 2147–2151, Nov. 2012.
[2] Byeong Yong Kong, Jihyuck Jo, Hyewon Jeong, Mina Hwang, Soyoung Cha, Bongjin Kim, and In-Cheol, "Low-Complexity Low-Latency Architecture for Matching of Data Encoded With Hard Systematic Error-Correcting Codes", IEEE Transactions On Very Large Scale Integration (VLSI) Systems., vol.22, no. 7,pp.1648-1652, July 2014.
[3] Pedro Reviriego, Salvatore Pontarelli, Juan Antonio Maestro, and Marco Ottavi, "A Method to Construct Low Delay Single Error Correction Codes for Protecting Data Bits Only," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems., vol. 32, no. 3, pp. 479 - 483, March 2013.
[4] G. Li, I. J. Fair, and W. A. Krzymien, "Low-density parity-check codes for space-time wireless transmission," IEEE Trans. Wirel. Commun., vol.5, no. 2, pp. 312–322, Feb. 2006.
[5] J. D. Warnock, Y.-H. Chan, S. M. Carey, H. Wen, P. J. Meaney, G. Gerwig, H. H. Smith, Y. H. Chan, J. Davis, P. Bunce, A. Pelella, D. Rodko, P. Patel, T. Strach, D. Malone, F. Malgioglio, J. Neves, D. L. Rude, and W. V. Huott "Circuit and physical design implementation of the microprocessor chip for the Enterprise system," IEEE J. Solid-State Circuits, vol. 47, no. 1, pp. 151–163, Jan. 2012.
[6] V. Gherman, S. Evain, N. Seymour, and Y. Bonhomme, "Generalized parity-check matrices for SEC-DED codes with fixed parity," in Proc. IEEE On-Line Testing Symp., Jul. 2011, pp. 198–20.

PARAMETER	EXISTING SYSTEM	PROPOSED SYSTEM
Power consumption	100 mW	52 mW
Latency	50 ns	15.13 ns