INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 5, Issue 3 AUG 2016

An Architecture for Matching The Data Protected With an Error-Correcting Code To Reduce Complexity and Latency

N.SIREESHA[1]
sireeshachowdharynallabothula@gmail.com[1]

S.MAHABOOB BASHA[2]
basha.nascent@gmail.com[2]

[1]PG Scholar,Dept of ECE, Rajeev Gandhi memorial college of engineering and technology, Nandyal, Kurnool, Andhra pradesh.

[2]Assistant Professor, Dept of ECE, Rajeev Gandhi memorial college of engineering and technology, Nandyal, Kurnool, Andhra pradesh.

ABSTRACT: Data comparison is widely used in computing system to perform many operation. Where incoming information is needs to be compared with a piece of stored data to locate the matching entry. If both incoming bits and stored bits are matching means there is no error if mismatched means some type of error will occurred like random error or burst error. To detect and correct the error here error correcting codes are used. 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. The proposed architecture examines whether the incoming data matches the stored data if a certain number of burst errors are corrected. The basic function of the BWA is to count the number of 1's among its input bits. It consists of multiple stages of HAs 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.

Key words: error correcting codes, hamming distance, data comparison, parity matrix.

I. INTRODUCTION

Now a days data comparison is widely used in many application. It also perform many operations. This comparison circuit consist of low hardware complexity. System performance is increased besides the data comparison usually resides in the critical path of the components that are devised to increase the system performance. Whose outputs determine the flow of the succeeding operations in apipeline .Therefore the circuit must be designed to have as a low latency as possible overall performance of the whole system would be severely deteriorated. To protect the data error correcting code is used to improve the reliability ECC decoding and correction before the comparison operation can be performed. To eliminate the complex decoding from the critical path. This method does not represent the Whether the retrieved data is extractly matched with the incoming data. To calculate the hamming distance which represents the difference between two numbers. To separate the data part and parity part systematic form is used.

Data comparison circuit isa 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. Error correction codes (ECC) are the one, most commonly used to protect standard memories and circuits [6], while more sophisticated codes are used in critical applications such as space [6]. ECC are widely used to enhance the reliability and data integrity of memory structures in modern microprocessors. For example, caches on modern microprocessors are protected by ECC [3]. If a memory structure is protected with ECC, a piece of data is encoded first and the entire

codeword including the ECC check bits are written into the memory array. When the input data is loaded into the system, it has to be encoded and compared with the data stored in the memory and corrected if errors are detected to obtain the original data. Data comparison circuit is usually in the critical path ofa pipeline stage because the result of the comparison determines the flow of the succeeding operations. When the memory array is protected by ECC, it exacerbates the criticality because of the added latency due to ECC logic. In the cache tag matching example, the cache tag directory must be accessed first. After the tag information is retrieved, it must go through ECC decoding and correction before the comparison operation can be performed. At the mean time, the corresponding data array is waiting for the comparison result to decide which way in the set to load the data from. The most recent solution for the matching problem is the direct compare method[5], 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 [5] as a basic building block for calculating the Hamming distance. However, it does not consider an important fact that a practical ECC codeword is usually represented in a systematic form in which the data and parity bits are completely separated from each other.

II. Literature Survey:

This section describes the conventional decode-and-compare architecture based on the direct compare method. For the sake of concreteness, only the tag matching performed in a cache memory is discussed in this brief, but it is said that the proposed architecture can be applied to similar applications without loss of generality.

A. Direct Compare Method

The key idea behind direct compare scheme is to utilize the information carried by the incoming data (referred to as input) to circumvent the necessity of decoding and correction of the stored codeword which may or may not have errors. For input protected with ECC, in most scenarios, the corrupted codeword is the only copy of that contains the original information. Without redundancy provided by ECC there is no other way to retrieve it. However, for data comparison, the absolute values of the stored information are not that important, but rather the relative value to the incoming data is important for deriving a comparison result. Considering the number of input bits (N) to be 31, i.e., N=31a circuit for direct compare method is proposed with full adders and saturate adders.

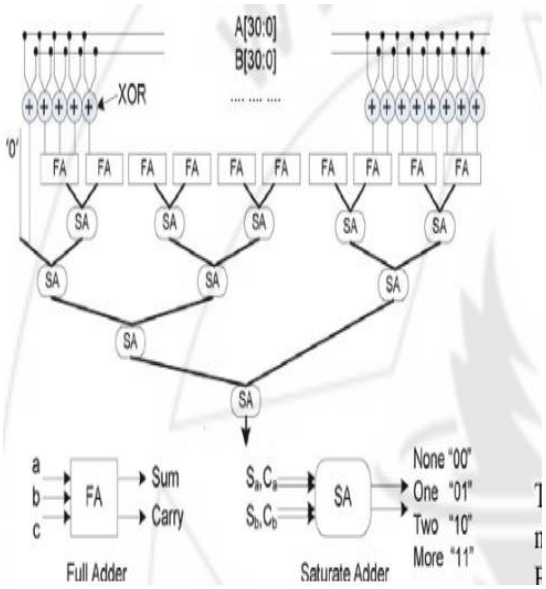

Figure 1: Direct compare (DC) circuit for N=31

B. Decode-and-Compare Architecture

Let us consider a cache memory where an n-bit codeword is stored after being encoded bya (n, k) code. In the decodeand-compare architecture depicted in Fig. 2, 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. Since the decoder is one of the most complicated processing elements, in addition, the complexity overhead is not negligible. Grounded on the fact of implementing the decoding architecture in hardware, it results in increase of hardware complexity, since the decoding technique includes large no of gates when implemented.

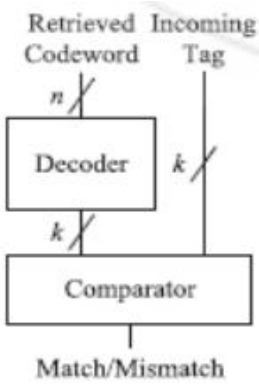

Figure 2: Decode-and-Compare architecture

III.Proposed Architecture

This section presents a new architecture that can reduce the latency and complexity of the data comparison by using the characteristics of systematic codes.

A. Block Diagram

The Fig.3 describes the flow of the proposed architecture. The incoming data is encoded by appending the parity bits.

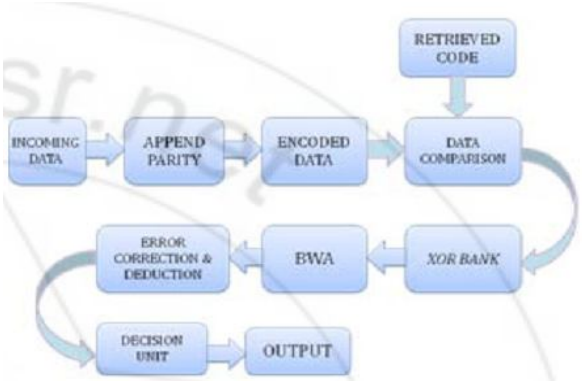

Figure 3:.Block Diagram

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 ones which are fed into error correction and error deduction unit. Thus the output is obtained from the decision unit.

B. Data Path Design

In the SA-based architecture [5], the comparison of two codeword is performed after the incoming tag is encoded. Therefore, the critical path consists of a series of the encoding and the n-bit comparison. However, the fact that, in practice, the ECC codeword is of a systematic form in which the data and parity parts are completely separated is not taken into the account. As the data part ofa 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. Grounded on this fact, the comparison of the k-bit tags can be started before the remaining(n–k)-bit

comparison of the parity bits. In the proposed architecture, therefore, the encoding process to generate the parity bits from the incoming tag is performed in parallel with the tag comparison, resulting in the reduction of the overall latency.

C. Construction of Low Delay Single-Error Correction Codes

The proposed method to construct SEC and SEC-DED codes tries to minimize the number of ones in each row and in each column of the parity check H matrix. Reducing the number of ones in the rows lowers the delay when computing the parity bits in the encoder. To minimize the number of ones the value w = 2 can be used to obtain SEC codes. It is also interesting to analyze the case w = 3 as inthat case the code is SEC-DED. Since for the parity bits the columns have only a one, the condition is not met as other columns havea one in that bit. Therefore, this modification cannot correct errors in the parity bits. This is not an issue for registers as the correction of parity bits is not normally needed. The method to construct the code starts by finding the smallest value of n − k for which the following is true:

$$\binom{n-k}{w} \geq k. \quad (1)$$

For w = 2, this value can be found analytically by solving (1) that is a quadratic equation in n. As the value of n has to be larger than k, only one of the two possible solutions of the equation is valid in our case. The value of n − k obtained is

$$n-k \geq \left\lceil \frac{1+\sqrt{8k+1}}{2} \right\rceil \quad (2)$$

that showsa growth of the number of parity bits with square root of k that is larger than logarithmic growth of Hamming codes. This means that as k increases, the overhead of the proposed codes in terms of the number of additional parity bits compared to Hamming will also increase. Similarly, for w = 3, the solution to (1) is given by

$$n-k \geq \left(\frac{\sqrt{243k^2-1}}{3^{3/2}}+3k\right)^{1/3} + \frac{1}{3\left(\frac{243k^2-1}{3^{3/2}}+3k\right)^{1/3}}+1 \quad (3)$$

that, as k is larger than one, it can be approximated by

$$n-k \geq (6k)^{1/3}+1 \quad (4)$$

The growth of the number of parity check bits with k is smaller than for w = 2, but is still larger than the logarithmic growth of traditional SEC-DED codes. In the second step to constructing the codes,a different combination of w of the n−k added bits is used for each of the first k columns of the H matrix. Equation (1) guarantees that there are sufficient different combinations. The remaining n − k columns form an identity matrix of size n − k. The reduction in the number of ones enables a lower encoding and decoding delay. In a general case, a Hamming code will have rows witha number of ones that is roughly k/2. This compares with the proposed SEC codes(w = 2) for which the number of ones in a row is by design at most n − k − 1. Similarly, to locate an error a traditional SEC code requires an n−k input AND gate compared with a simple two input AND gate in the proposed code. In practical implementations, this results in a significant reduction of the encoding and decoding delays. One distinct feature of the proposed codes is that they correct errors on the data bits only. This is similar to other codes such as Orthogonal Latin square (OLS) codes [10]. However, in OLS codes, each pair of data bits participates inat most one shared parity check bit to ensure that majority logic decoding can be used. This is different from the proposed

scheme in which the goal is to ensure that no data bit participates in all the parity check bits, in which another data bit participates. This is then used to simplify the location and correction of an error, as described before. Another difference is that OLS codes are commonly used when multiple error correction capabilities are needed although SEC can also be implemented. The main issues with SEC OLS codes are that they are only implemented for a few block sizes and requirea large number of parity check bits. Finally, it is worth mentioning that the parity check matrices of the proposed codes are similar to that of low density parity check (LDPC) codes commonly used in communication systems [11]. Nevertheless, since LDPC codes usually have large block size, and must provide multiple error correction, the encoding and decoding procedures are very different from our proposed codes and require complex logic circuitry [11].

D. Architecture for Hamming Distance Computation

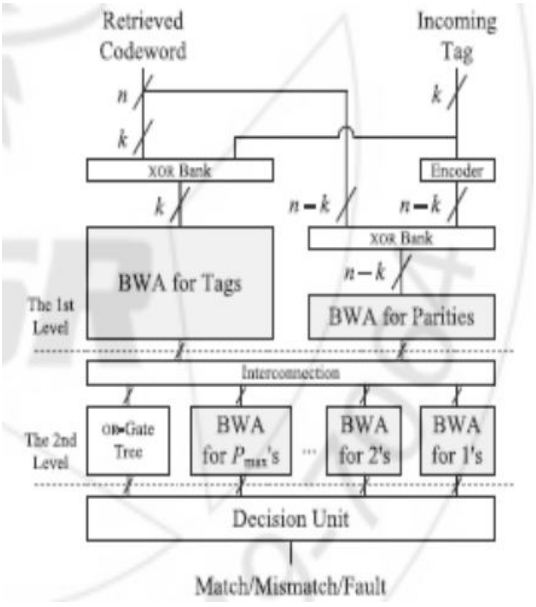

Figure 4:.Architecture of Hamming Distance Computation

The proposed architecture grounded on the data path design is shown in Fig.4. It contains multiple butterfly-formed weight accumulators (BWAs) proposed for the Hamming distance computation. The basic function of the BWA is to count the number of 1's among its input bits. It consists of multiple stages of HAs as shown in Fig.5(a), where each output bit ofa HA is associated with a weight. The HAs ina 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 ofa HA is set, the number of 1's among the bits in paths reaching the HA is equal to the weight of the output bit.

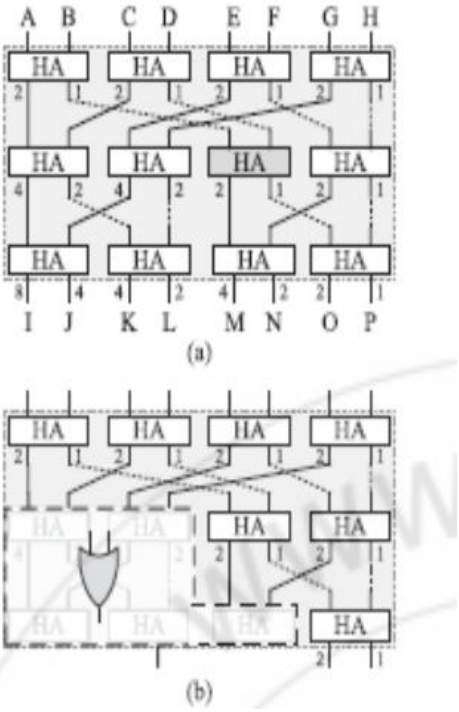

Figure 5 (a):.General structure and (b) New revised structure

The basic function of the BWA is to count the number of 1's among its input

bits. It consists of multiple stages of HAs as shown in Fig. 5(a), 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. In Fig. 5(a), 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 Fig. 5(a), 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. (2)

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 in Fig. 5(b). This is an advantage over the SA that resorts to the compulsory saturation expressed in (1). Note that in Fig. 6, there is no overlap between any pair of two carry-bit lines or any pair of two sum-bit lines. As the overlaps exist only between carry-bit lines and sum-bit lines, it is not hard to resolve overlaps in the contemporary technology that provides multiple routing layers no matter how many bits a BWA takes. We now explain the overall architecture in more detail. Each XOR stage in Fig. 4 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 Fig. 5(b), and generates an output from theOR-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 theOR-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 ware 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 of two that is not greater than rmax+1. 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 thanPmax.

E. General Expressions for the Complexity

The complexity as well as the latency of combinational circuits heavily depends on the algorithm employed. It is unfortunately hard to derive an analytical and fully deterministic equation that shows the relationship between the number of gates and the latency for the proposed architecture The complexity of the proposed architecture, C, can be expressed as

C=CXOR+CENC+CBWA(k)+CBWA(n-k)+C2nd+CDU≤n+CENC+2CBWA(n)+CDU ……(6)

where CXOR, CENC, C2nd, CDU, and CBWA(n) are complexities of XOR banks, an encoder, the second level circuits, the decision unit, anda BWA for n inputs, respectively. Using the recurrence relation, CBWA(n) can be calculated as

$$C_{\mathrm{BWA}}(n) = C_{\mathrm{BWA}}(\lceil n/2 \rceil) + C_{\mathrm{BWA}}(\lceil n/2 \rceil) + 2\lfloor n/2 \rfloor$$

F. General Expressions for the Latency

The latency of the proposed architecture, L, can be expressed as

$L \leq \max [LXOR + LBWA (k), LENC + LXOR + LBWA (n-k)] + L2nd + LDU$..(8)

where LXOR, LENC, L2nd, LDU, and LBWA (n) are the latencies of an XOR bank, an encoder, the second level circuits, the decision unit, and a BWA for n inputs, respectively. Note that the latencies of the OR-gate tree and BWAs for $x \leq n$ inputs at the second level are all bounded by log2n. As one of BWAs at the first level finishes earlier than the other, some components at the second level may start earlier. Similarly, some BWAs or the OR-gate tree at the second level may provide their output earlier to the decision unit so that the unit can begin its operation without waiting for all of its inputs. In such cases, L2nd and LDU can be partially hidden by the critical path of the preceding circuits, and L becomes shorter than the given expression.

IV. Results

Table 2: Comparison For Latency And Complexity

ECC	Architecture	Gate-Level Counting		Implementation Results	
		Latency[a]	Complexity[b]	CPD[c]	EGC[d]
(16, 11)	Conventional	14 (1.17)[e]	137 (1.10)	2.13 (1.16)	320 (1.20)
	SA-based	14 (1.17)	132 (1.06)	2.12 (1.16)	304 (1.14)
	Proposed	12 (1.00)	125 (1.00)	1.83 (1.00)	266 (1.00)
(24, 18)	Conventional	16 (1.23)	238 (1.24)	2.31 (1.16)	491 (1.19)
	SA-based	18 (1.38)	211 (1.10)	2.46 (1.23)	475 (1.15)
	Proposed	13 (1.00)	192 (1.00)	2.00 (1.00)	412 (1.00)
(31, 25)	Conventional	16 (1.23)	336 (1.29)	2.48 (1.24)	684 (1.22)
	SA-based	18 (1.38)	290 (1.11)	2.57 (1.29)	634 (1.13)
	Proposed	13 (1.00)	261 (1.00)	1.99 (1.00)	561 (1.00)
(40, 33)	Conventional	18 (1.20)	473 (1.38)	2.64 (1.20)	861 (1.21)
	SA-based	22 (1.47)	377 (1.10)	2.96 (1.35)	816 (1.15)
	Proposed	15 (1.00)	342 (1.00)	2.20 (1.00)	709 (1.00)

[a] The number of gates in the critical path.
[b] The count of all gates.
[c] The critical path delay(CPD)in nanoseconds.
[d] The equivalent gate count(EGC).
[e] The numbers in the parenthesis are normalized values.

Table II shows the latencies and hardware complexities resulting from three architectures: 1) the conventional decode-and-compare; 2) The SA-based direct compare; and 3) the proposed ones. In [5], the latency is measured from the time when the incoming address is completely encoded. As the critical path starts from the arrival of the incoming address to a cache memory, the encoding delay must be, however, included in the latency computation. The latency values in Table II are all measured in this way. Besides, critical-path delays in Table II are obtained by performing post layout simulations and equivalent gate counts are measured by countinga two-input NAND as one. As shown in Table II, the proposed architecture is effective in reducing the latency as well as the hardware complexity even with considering the practical factors. Note that the effectiveness of the proposed architecture over the SA-based one in shortening the latency gets larger as the size of a codeword increases. The reason is that, the latencies of the SA-based architecture and the proposed one is dominated by SAs and HAs, respectively. As the bit-width doubles, at least one more stage of SAs or HAs needs to be added. Since the critical path of a HA consists of only one gate while that of a SA has several gates, the proposed architecture achieves lower latency than its SA-based counterpart, especially for long code words.

V. Conclusion

A new architecture has been presented for matching the data protected with an ECC to reduce the complexity and delay. The proposed architecture examines whether the incoming data matches the stored data if a certain number of erroneous bits are corrected. Systematic codeword is used to enabled the based on parallel operation it has separate the data and parity bits. To reduce the latency, the comparison of the data is parallelized with

the encoding process that generates the parity information. an efficient processing architecture has been presented to further minimize the latency and complexity. The experimental results show that the efficiency of the proposed method. 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.

References

[1] Byeong Yong Kong, Jihyuck Jo, Hyewon Jeong, Mina Hwang,Soyoung Cha, Bongjin Kim, and InCheol Park "Low-Complexity Low-Latency Architecture for Matching Of Data EncodedWith Hard Systematic Error-Correcting Codes" IEEE transactions on (vlsi) systems, vol. 22, no. 7, July 2014

[2] J. D. Warnock, Y.-H. Chan, S. M. Carey, H. Wen, P. J. Meaney, G. Gerwig 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.

[3].W. Wu, D. Somasekhar, and S.-L. Lu, "Direct compare of information coded with error-correcting codes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 11, pp. 2147–2151, Nov. 2012.

[4].S. Lin and D. J. Costello, Error Control Coding: Fundamentals and Applications, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[5] Y. Lee, H. Yoo, and I.-C. Park, "6.4Gb/s multithreaded BCH encoder and decoder for multi-channel SSD controllers," in ISSCC Dig. Tech. Papers, 2012, pp. 426–427 .

[6] M. Tremblay and S. Chaudhry, "A thirdgeneration 65nm 16-core 32-thread plus 32-scoutthread CMT SPARC processor," in ISSCC. Dig. Tech. Papers, Feb. 2008, pp. 82–83.

[7] H. Ando, Y. Yoshida, A. Inoue, I. Sugiyama, T. Asakawa, K. Morita, T. Muta, and T. Motokurumada, S. Okada, H. Yamashita, and Y. Satsukawa, "A 1.3 GHz fifth generation SPARC64 microprocessor," in IEEE ISSCC. Dig. Tech. Papers, Feb. 2003, pp. 246–247.