IMPLEMENTATION OF SYSTEMATIC ERROR CORRECTING CODES FOR MATCHING OF DATA ENCODED

Amala A[1]*, Jasmine M[2]

[1]* M. Tech Scholar, Department of ECE,
Sri Venkatesa Perumal College of Engineering & Technology, Puttur, India

[2] Assoc. Professor, Department of ECE,
Sri Venkatesa Perumal College of Engineering & Technology, Puttur, India

Abstract: A Computing system is one,where an input data compared with a stored data to locate the matching entry.For example translation look aside buffer and Cache tag array lookup matching. In this paper, we propose new architecture in order to reduce complexity and latency for matching the data protected with an error-correcting code(ECC). It is based on the codeword of an ECC generated by encoding is usually represented in a systematic form and it consists the raw data and the parity information. The proposed architecture parallelizes the comparison of the data and that of the parity information.To reduce the latency and complexity, we propose a new butterfly-formed weight accumulator(BWA) for The efficient computation of the Hamming distance.The proposed architecture checks whether the incoming data matches with the stored data.

Keywords: Data comparison, error-correcting codes (ECCs), Hamming distance, systematic codes, tag matching.

1. INTRODUCTION

Data comparison is widely used in computing systems to perform many operations such as the tag matching in a cache memory and the virtual-to-physical address translation in a translation lookaside buffer (TLB). Because of such prevalence, it is important to implement the comparison circuit with low hardware complexity. Besides, the data comparison usually resides in the critical path of the components that are devised to increase the system performance, e.g., caches and TLBs, whose outputs determine the flow of the succeeding operations in a pipeline. The circuit, therefore, must be designed to have as low latency as possible, or the components will be disqualified from serving as accelerators and the overall performance of the whole system would be severely deteriorated. As recent computers employ error-correcting codes (ECCs) to protect data and improve reliability, complicated decoding procedure, which must precede the data comparison, elongates the critical path and exacerbates the complexity overhead. Thus, it becomes much harder to meet the above design constraints. Despite the need for sophisticated designs as described, the works that cope with the problem are not widely known in the literature since it has been usually treated within industries for their products. Recently, however, triggered the attraction of more and more attentions from the academic field.

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 codewords, 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.

In this brief, we renovate the SA-based direct compare architecture to reduce the latency and hardware complexity by resolving the aforementioned drawbacks. More specifically, we consider the characteristics of systematic codes in designing the proposed architecture and propose a low-complexity processing element that computes the Hamming distance faster. Therefore, the latency and the hardware complexity are decreased considerably even compared with the SAbased architecture.

The rest of this brief is organized as follows. Section 2 reviews previous works. The proposed architecture is explained in Section 3, and Results in Section 4. Finally, concluding remarks are made in Section 5.

2.PREVIOUS WORKS

This section describes the conventional decode-and-compare architecture and the encode-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 the proposed architecture can be applied to similar applications without loss of generality.

A.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 depicted in Fig. 1(a), 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.

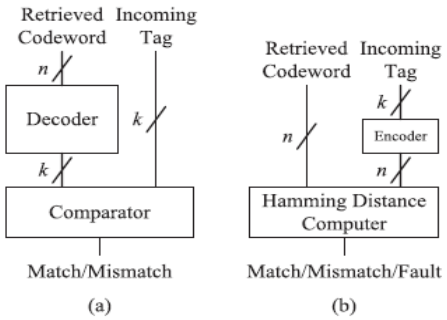

Fig. 1. (a) Decode-and-compare architecture and (b) encode-and-compare architecture.

B.Encode-and-Compare Architecture

Note that 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 as shown in Fig. 1(b).

The comparison is to examine how many bits the two codewords differ, not to check if the two codewords are exactly equal to each other. For this, we compute the Hamming distance d between the two codewords and classify the cases according to the range of d. Let tmax and rmax denote the numbers of maximally correctable and detectable errors, respectively. The cases are summarized as follows.
1) If d = 0, X matches Y exactly.
2) If 0 < d ≤ tmax, X will match Y provided at most tmax errors in Y are corrected.
3) If tmax < d ≤ rmax, Y has detectable but uncorrectable errors.
In this case, the cache may issue a system fault so as to make the central processing unit take a proper action.
4) If rmax < d, X does not match Y .
Since the above method needs to compute the Hamming distance, presented a circuit dedicated for the computation. 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 codewords. 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. In the SA tree, the accumulated value z is saturated to rmax + 1 if it exceeds rmax. More precisely, given inputs x and y, z can be expressed as follows:

$$z = \begin{cases} x + y, & \text{if } x + y \le r_{\max} \\ r_{\max} + 1, & \text{otherwise.} \end{cases}$$

The final accumulated value indicates the range of d. As the compulsory saturation necessitates additional logic circuitry, the complexity of a SA is higher than the conventional adder.

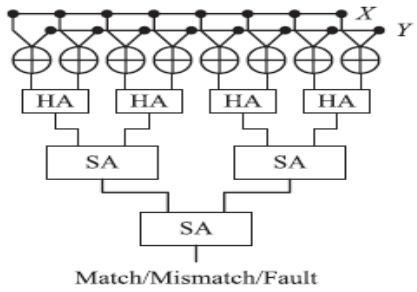

Fig. 2. SA-based architecture supporting the direct compare method

3.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. In addition, a new processing element is presented to reduce the latency and complexity further.

A.Block Diagram

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

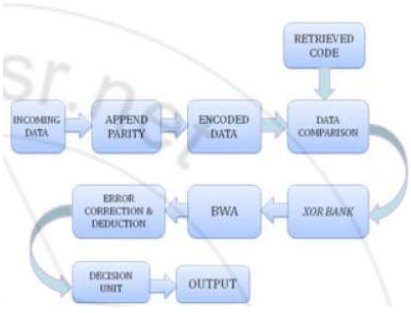

Fig3:Block Diagram

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.Datapath Design

In the SA-based architecture, the comparison of two codewords is invoked after the incoming tag is encoded. Therefore, the critical path consists of a series of the encoding and the n-bit comparison as shown in Fig. 4(a). However, 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 shown in Fig. 5. 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. 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, reducing the overall latency as shown in Fig. 4(b).

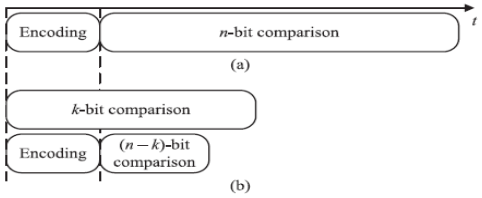

Fig.4. Timing diagram of the tag match in (a) direct compare method (b) proposed architecture.

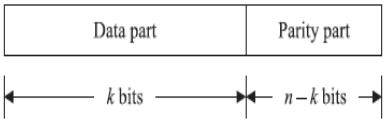

Fig. 5. Systematic representation of an ECC codeword.

C.Architecture for Computing the Hamming Distance

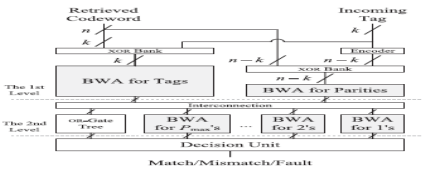

Fig. 6. Proposed architecture optimized for systematic codewords

The proposed architecture grounded on the datapath design is shown in Fig. 6. 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. It consists of multiple stages of HAs as shown in Fig. 7(a), where each output bit of a HA is associated with a weight.

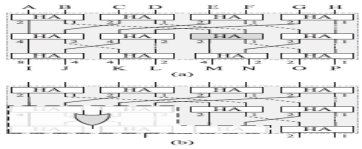

Fig. 7. Proposed BWA. (a) General structure and (b) new structure revised for the matching of ECC-protected data. Note that sum-bit lines are dotted for visibility.

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. 7(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. 6(a), the number of 1's among the input bits, d, can be calculated as

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

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. 7(b). This is an advantage over the SA that resorts to the compulsory saturation.

Note that in Fig. 7, 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. 6 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. 7(b), 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 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 than Pmax.

D.General Expressions for the Complexity

The complexity as well as the latency of combinational circuits heavily depends on the algorithm employed. In addition, as the complexity and the latency are usually conflicting with each other, 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 and also for the conventional SA-based architecture. To circumvent the difficulty in analytical derivation, we present instead an expression that can be used to estimate the complexity and the latency by employing some variables for the nondeterministic parts. The complexity of the proposed
architecture, C, can be expressed as

$$\begin{aligned} C &= C_{XOR} + C_{ENC} + C_{BWA}(k) + C_{BWA}(n-k) + C_{2nd} + C_{DU} \\ &\leq n + C_{ENC} + 2C_{BWA}(n) + C_{DU} \end{aligned}$$

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

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

where the seed value, CBWA(1), is 0. Note that when a + b = c, CBWA(a) + CBWA(b) ≤ CBWA(c) holds for all positive integers a, b, and c. Because of the inequality and the fact that an OR-gate tree for n inputs is always simpler than a BWA for n inputs, both CBWA(k) + CBWA(n –k) and C2nd are bounded by CBWA(n).

F.General Expressions for the Latency

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

$$\begin{aligned} L &\le \max\left[L_{\mathrm{XOR}} + L_{\mathrm{BWA}}(k), L_{\mathrm{ENC}} + L_{\mathrm{XOR}} + L_{\mathrm{BWA}}(n-k)\right] \\ &\quad + L_{2\mathrm{nd}} + L_{\mathrm{DU}} \\ &\le \max\left(1 + \lceil \log_2 k \rceil, L_{\mathrm{ENC}} + 1 + \lceil \log_2 (n-k) \rceil\right) \\ &\quad + \lceil \log_2 n \rceil + L_{\mathrm{DU}} \end{aligned}$$

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 ≤ n inputs at the second level are all bounded by [log2 n . 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.

4.RESULTS

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 as follows. The latencies of the SA-based architecture and the proposed one are 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.

4.1 simulation results

fig 8: RTL Schematic

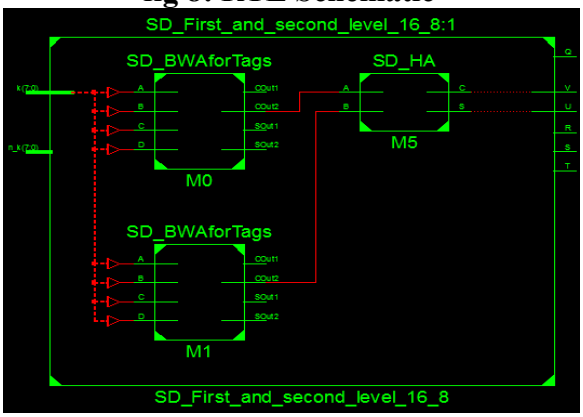

Fig 9: Internal structure for 16_8 RTL schematic

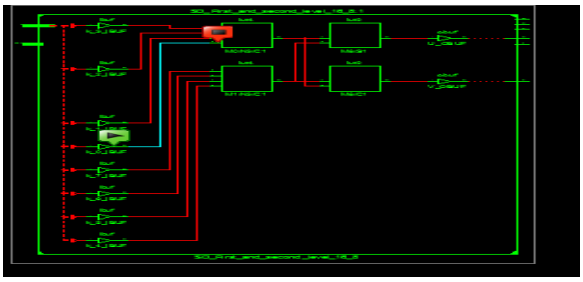

Fig 10: Technology Schematic of 16_8

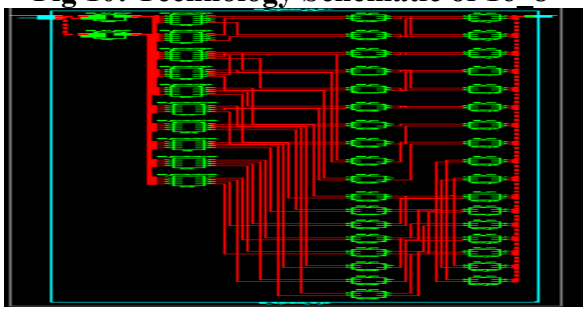

Fig 11: Internal structure for 40_33 RTL schematic

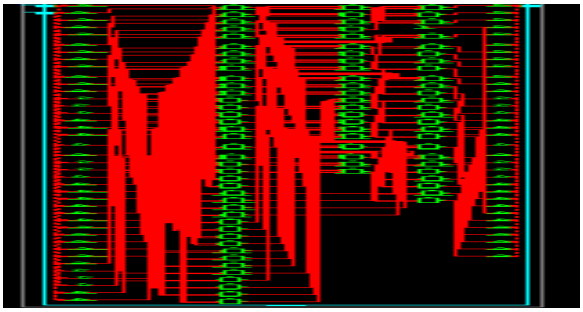

Fig 12: Technology Schematic of 40_33 bits

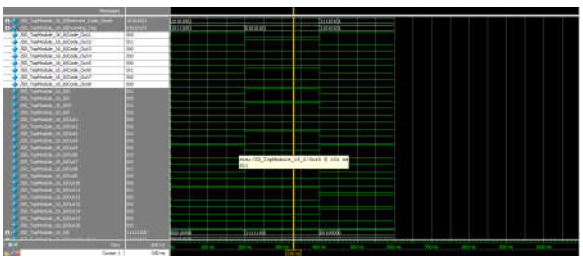

Fig 13: Simulation result of 16_8 bits

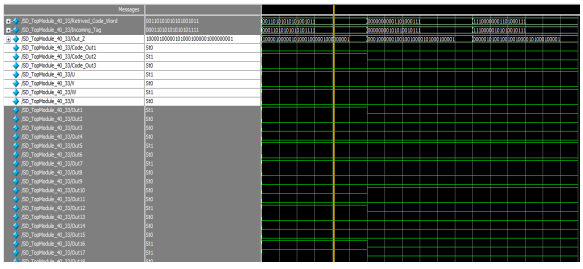

Fig 14 Simulation result of 40_33 bits

5.CONCLUSION

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. Toreduce 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.

REFERENCES

[1] J. Chang, M. Huang, J. Shoemaker, J. Benoit, S.-L. Chen, W. Chen, S. Chiu, R. Ganesan, G. Leong, V. Lukka, S. Rusu, and D. Srivastava, "The 65-nm 16-MB shared on-die L3 cache for the dual-core Intel xeon processor 7100 series," IEEE J. Solid-State Circuits, vol. 42, no. 4, pp. 846–852, Apr. 2007.

[2] 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 themicroprocessor chip for the zEnterprise system," IEEE J. Solid-State Circuits, vol. 47, no. 1, pp. 151–163, Jan. 2012.

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

[4] Y. Lee, H. Yoo, and I.-C. Park, "6.4Gb/s multi-threaded BCH encoder and decoder for multi-channel SSD controllers," in ISSCC Dig. Tech.Papers, 2012, pp. 426–427. S. Lin and D. J. Costello, Error Control Coding: Fundamentals and Applications, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall,2004

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

[6] G. Olocco and J. Tillich, "A family of self-dual codes which behave in many respects like random linear codes of rate 1/2," in Proceedings of the IEEE ISIT , 24-29 June 2001, p. 15.