Vol 06, Article 05586; June 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

Low-Complexity Low-Latency Architecture for Matching of Data Encoded with Hard Systematic Error-Correction Codes

SANTOSHKUMAR
PG student, Dept of ECE, VTU Regional Center, Kalaburgi, INDIA
santoshkmr910@gmail.com

ABSTRACT

Data comparison is widely used in computing system to perform so many operations. 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 occur 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. A new architecture for matching the data protected with an ***Error-Correcting Code*** *(ECC) is presented in this brief to reduce latency and complexity.*

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 look aside 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 [1]–[5], 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, [6] 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 [6], 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 in [6] as a basic building block for calculating the Hamming distance. However, [6] 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 [7]. 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.

Here, 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 SA based architecture.

2. PREVIOUS WORKS

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

Vol 06, Article 05586; June 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

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=31 a circuit for direct compare method is proposed with full adders and saturate adders.

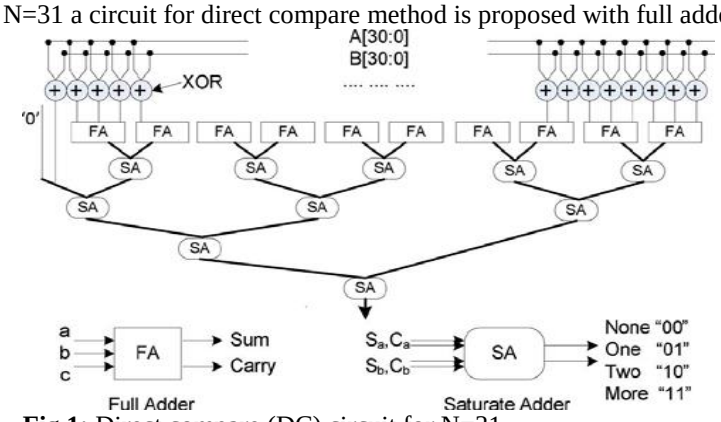

Fig 1: Direct compare (DC) circuit for N=31

B. Decode-and-Compare Architecture

Let us consider a cache memory where an *n*-bit codeword is stored after being encoded by a (*n*, *k)* code. In the decode-and-compare architecture depicted in Fig. 2 (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. 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.

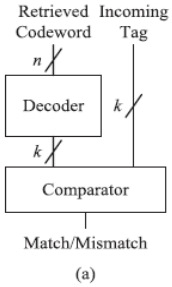

Fig 2: Decode-and-Compare architecture

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.

A. Block Diagram

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

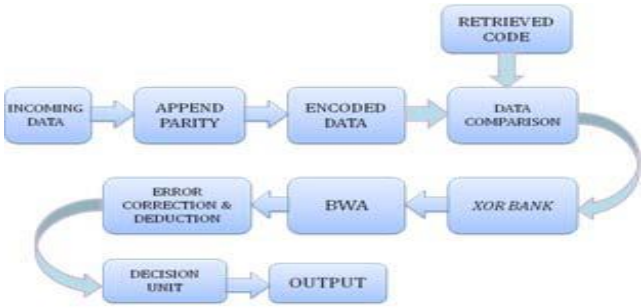

Fig 3:.Block Diagram

B. Datapath Design for Systematic Codes

In the SA-based architecture [6], 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 as shown in Fig. 4(a).

Vol 06, Article 05586; June 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

However, [6] 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. 4. 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).

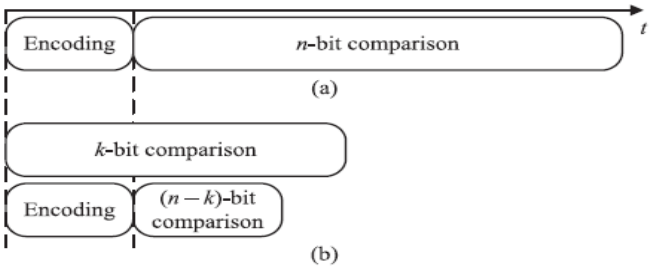

Fig.4. Timing diagram of the tag match in (a) direct compare method (b) proposed architecture.

C. Architecture for 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 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 paths reaching the HA is equal to the weight of the output bit.

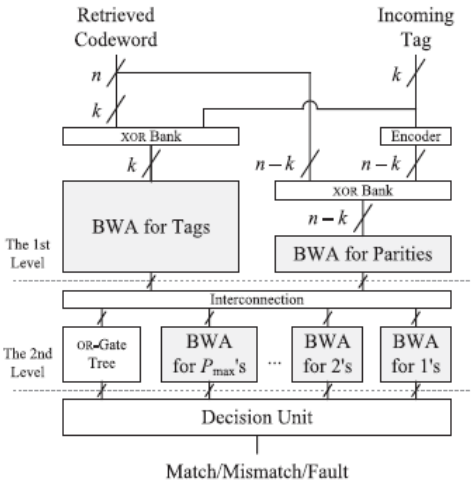

Fig 4:.Architecture of Hamming Distance Computation

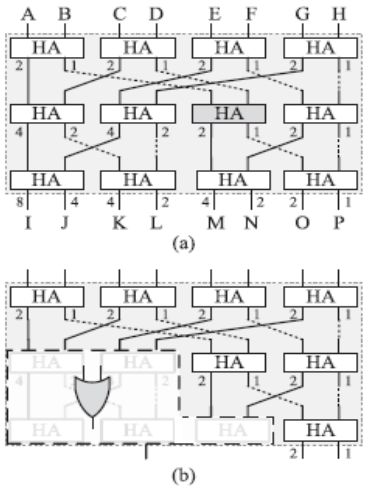

Fig 5 (a):.General structure and (b) New revised structure

In Fig. 5(a), for example, if the carry bit of the 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

$$d = 8I + 4\,(J + K + M) + 2\,(L + N + O) + P \quad (5)$$

Since what we need is not the precise Hamming distance but the range it belongs to, it is possible to simplify the circuit. When *r*max = 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. Note that in Fig. 5 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 generates the bitwise difference vector for either data bits or parity bits.

Vol 06, Article 05586; June 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

D. 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

where *C*XOR, *C*ENC, *C*2nd, *C*DU, and *C*BWA*(n)* are complexities of XOR banks, an encoder, the second level circuits, the decision unit, and a BWA for *n* inputs, respectively. Using the recurrence relation, *C*BWA*(n)* can be calculated as

*C*BWA*(n)* = *C*BWA *(_n/2_)* + *C*BWA *(_n/2_)* + 2 *_n/2*

E. General Expressions for the Latency

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

$$L \le \max [L\text{XOR} + L\text{BWA}(k), L\text{ENC} + L\text{XOR} + L\text{BWA}(n-k)] + L2\text{nd} + L\text{DU}$$
$$\le \max(1+[\log_2 k], L\text{ENC} + 1+[\log_2(n-k)]+[\log_2 n]+ L\text{DU}$$

where *L*XOR, *L*ENC, *L*2nd, *L*DU, and *L*BWA*(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 $x \le n$ inputs at the second level are all bounded by$[\log_2 n]$. 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, *L*2nd and *L*DU can be partially hidden by the critical path of the preceding circuits, and *L* becomes shorter than the given expression.

4. RESULTS

Table I 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.

Table.1 Comparison For Latency And Hardware 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 the gates.
[c]The critical-path delay (CPD) in nanoseconds.
[d]The equivalent gate count (EGC) measured by counting a two-input NAND as one.
[e]The numbers in parentheses are normalized values.

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 counting a 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.

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

Vol 06, Article 05586; June 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

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.

ACKNOWLEDGMENTS

The authors would like to sincerely thank the contribution of my lectures and guild, who provided a great support.

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

[3] 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.

[4] 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.

[5] AMD Inc. (2010). *Family 10h AMD Opteron Processor Product Data Sheet*, Sunnyvale, CA, USA [Online]. Available: http://support.amd.com/us/Processor_TechDocs/40036.pdf

[6] 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.

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

[8] 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.