INTERNATIONAL JOURNAL OF COMPUTATIONAL MATHEMATICAL IDEAS (IJCMI)

Implementation of Systematic Error-Correcting Codes for Matching of Data Encoded

M.Madevi1 M.Tech ECE, M.Pradeep2M.Tech Assistant Professor
INDIRA INSTITUTE OF TECHNOLOGY & SCIENCES, Darimadugu (post), Markapuram-523316, PrakasamDist, Andhra Pradesh

Published In Vol: 15 - Issue1 June: 2017 Page No: 71590 to 71595

Abstract: A new design for matching the data protected with associate degree error-correcting code (ECC) is bestowed during this project to degrade latency and complexity. supported the very fact that the code word of associate degree error correction code is sometimes represented in a systematic form consisting of the raw information and also the parity data generated by encryption, the proposed design parallelizes the comparison of data and which 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 if a certain number of erroneous bits are corrected. For a (40, 33) code, the proposed architecture reduces the latency and the hardware complexity by, compared with the most recent implementation. This ECC is done by using Verilog HDL.

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

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. 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 code word 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 of a 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 meantime, 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 decreased considerably compared with the SA based architecture. In the decode-and-compare architecture, the n-bit retrieved code word 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 code word should go through the decoder before being compared with the incoming tag. Conventional decode-and-compare architecture and encode and compare architecture. To resolve the drawbacks of the decode-and-compare architecture, therefore, the decoding of a retrieved code word 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 code word X and compared with an n-bit retrieved code word 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 code word is of a systematic form in which the data and parity parts are completely separated. As the data part of a systematic code word 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. 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 code word 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 code word is usually represented in a systematic form in which the data and parity bits are completely separated from each other .In addition, SA contributes to the increase of the entire circuit complexity as it always forces its output not to be greater than the number of detectable errors by more. In brief, we renovate the SA-based direct compare architecture to reduce the latency and hardware complexity by resolving the 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

II. PROPOSED METHODOLOGY

A.Introduction

The proposed design parallelizes the comparison of the data and which of the parity information. Error correction code permits data that's being read or transmitted to be checked for errors and, once necessary, corrected on the fly. It differs from parity-checking in those errors don't seem to be solely detected however additionally corrected. We will scale back the realm of the proposed System and additionally we will additionally increase the correcting code word length. We will perform the error correction for numerous code word lengths. During this project we tend to do constant operation for (16,8) and (40,33) codes additionally. A translation look a side buffer (TLB) could be a cache that memory management hardware uses to boost virtual address translation speed. The bulk of desktop, laptop, and server processors includes one or additional TLBs within the memory management hardware, and it's nearly continuously gift in any hardware that utilizes paged or divided virtual storage. The TLB is usually enforced as content-addressable memory (CAM). The CAM search key's the virtual address and therefore the search result's a physical address. If the requested address is gift within the TLB, the CAM search yields a match quickly and therefore the retrieved physical address will be wont to access memory. This is often known as a TLB hit. If the requested address isn't within the TLB, it's a miss, and therefore the translation income by trying up the page table in a very method known as a page walk.

The page walk needs plenty of your time in comparison to the processor speed, because it involves reading the contents of multiple memory locations and mistreatment them to reason the physical address. Once the physical address is decided by the page walk, the virtual address to physical address mapping is

entered into the TLB. Here's however it works for information storage:

- Once a unit of information is keep in RAM or peripheral storage, a code that describes the bit sequence within the word is calculated and stored in conjunction with the unit of information. For every 64-bit word, an additional seven bits square measure required to store this code.
- Once the unit of information is requested for reading, a code for the stored and about-to-be-read word is once more calculated mistreatment the initial algorithmic program. The new generated code is compared with the code generated once the word was stored.
- If the codes match, the information is freed from errors and is distributed.
- If the codes do not match, the missing or incorrect bits square measure determined through the code comparison and therefore the bit or bits square measure equipped or corrected.
- No try is created to correct the information that's still in storage. Eventually, it'll be overlaid by new information and, presumptuous the errors were transient, the inaccurate bits can "go away."

Any error that recurs at constant place in storage once the system has been turned off and on once more indicate a permanent computer error and a message is distributed to a log or to a supervisor indicating the placement with the repeated errors. At the 64-bit word level, parity-checking and error correction code need constant range of additional bits. In general, error correction code will increase the reliability of any computing or telecommunications system (or a part of a system) while not adding abundant value. Reed-Solomon codes square measure normally implemented; they are ready to notice and restore "erased" bits additionally as incorrect bits.

B. Hamming Distance

In information theory, the performing distance between 2 strings of equal length is that the variety of positions at that the corresponding symbols area unit totally different. In our own way, it measures the minimum variety of substitutions needed to vary one string into the other, or the minimum variety of errors that might have transformed one string into the other. For binary strings a and b the Hamming distance is adequate the number of one’s in an exceedingly a XOR b. The mathematical space of length-n binary strings, with the Hamming distance, is thought because the Hamming cube; it's equivalent as a mathematical space to

the set of distances between vertices in an exceedingly hypercube graph. One also can read a binary string of length n as a vector in Rn by treating every symbol within the string as a real coordinate; with this embedding, the strings are formed the vertices of associate n-dimensional hypercube, and also the Hamming distance of the strings is admire the Manhattan distance between the vertices. The Hamming Distance may be a number used to denote the distinguish between 2 binary strings. It's a little portion of a broader set of formulas utilized in info analysis. Specifically, Hamming's formulas

Enable computers to detect and correcting error on their own. The Hamming Code earned Richard Hamming the Eduard Rheim Award of feat in Technology in 1996, 2 years before his death. Hamming's additions to info technology are utilized in such innovations as modems and compact discs.

Step 1: Ensure the 2 strings area unit of equal length. The hamming distance will solely be calculated between 2 strings of equal length. String 1: "1001 0010 1101" String 2: "10100010 0010".

Step 2: Compare the primary 2 bits in every string. If they' residential, record a "0" for that bit. If they're totally different, record a "1" for that bit. During this case, the first bit of both strings is "1," thus record a "0" for the primary bit.

Step 3: Compare every bit in succession and record either"1" or "0" as acceptable. String 1: "1001 0010 1101" String 2: "1010 0010 0010" Record: "0011 0000 1111".

Step 4: Add all one's and zeros within the record along to get the Hamming distance. Hamming distance =0+0+1+1+0+0+0+0+1+1+1+1 = 6

VI .REFERENCES

[1] 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.

[2] B.Y Kong, Jihyuck Jo, Hyewon Jeong, Mina Hwang, Soyoung Cha, Bongjin Kim, and In-Cheol Park, "Low-Complexity Low-Latency Architecture for Matching of Data Encoded With Hard Systematic Error-Correcting Codes," IEEE Trans. Very Large Scale Integr.(VLSI) Syst., vol. 22, no. 7, pp. 1648 - 1652, July. 2014.

[3] H. Ando, Y. Yoshida, A. Inoue, I. Sugiyama, T. Asakawa, T. Muta, K.Morita, T. Motokurumada, S. Okada, H. Yamashita, Y. Satsukawa, A.

Konmoto, R. Yamashita, and H. Sugiyama, “A 1.3 GHz fifth generation SPARC64 microprocessor,” in ISSCC. Dig. Tech. Papers, 2003, pp. 246–247.

[4] AMD Inc., Sunnyvale, CA, “Family 10h AMD Opteron™ Processor Product Data Sheet,” PID:40036Rev:3.04,2010.Available:http://support.amd.com/us/Processor_TechDocs/40036.pdf [Online]

[5] 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.

[6] 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.

[7] M. Nicolaidis, “Design for soft error mitigation,” IEEE Trans. Device Mater. Reliab., vol. 5, no. 3, pp. 405–418, Sep. 2005.

[8] C. L. Chen and M. Y. Hsiao, “Error-correcting codes for semiconductor memory applications: A state-of-the-art review,” IBM J. Res. Develop., vol. 28, no. 2, pp. 124–134, 1984.

[9] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, “Fault tolerant solid state mass memory for space applications,” IEEE Trans. Aerospace Electron. Syst., vol. 41, no. 4, pp. 1353–1372, Oct. 2005. [10]M. Y. Hsiao, D. C. Bossen, and R. T. Chien, “Orthogonal latin square codes,” IBM J. Res. Develop., vol. 14, no. 4, pp. 390–394, Jul. 1970.

[11] 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.

[12] M. Y. Hsiao “A class of optimal minimum odd-weight column SECDED codes,” IBM J. Res. Develop., vol. 14, pp. 395–301, Jul. 1970.

[13] R. W. Hamming, “Error detecting and error correcting codes,” Bell Syst. Tech. J., vol. 29, pp. 147–160, Apr. 1950.

[14] V. Gherman, S. Evain, N. Seymour, and Y. Bonhomie, “Generalized parity-check matrices for SEC-DED codes with

V. CONCLUSION

The architecture presented in this project is very efficient in delay reducing and also the complexity as well; it was 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.

Author 1:

M.Madhavi She is a pursuing M.Tech (ECE) From Indira Institute of Technology and Sciences, Darimadugu (post), Markapuram-523316, PrakasamDist, Andhra Pradesh

Author 2:

M.Pradeep M.Tech, He is a Working as Assistant professor in ECE Department of Indira Institute of Technology and Sciences, Darimadugu (post), Markapuram-523316, PrakasamDist, Andhra Pradesh