RESEARCH ARTICLE OPEN ACCESS

An Efficient Low Complexity Low Latency Architecture for Matching of Data Encoded With Error Correcting Code Using a Cache Memory

Aswathi D[1], Keerthi Sahadevan[2]
[1]*M Tech Student*
[2]*Assistant Professor, Ece Department*

ABSTRACT
An efficient architecture is introduced for the matching of data encoded with error correcting code using a cache memory is presented in brief. Using cache memory it reduces latency and complexity to an fine level. And this architecture further reduces the dynamic power without affecting the time. For the comparison of data, hamming distance along used to check whether the data match the data kept in main memory. Instead of butterfly formed weight accumulator(previous work) here no other mechanism is presented for calculating hamming distance.
Index Terms: Butterfly formed weight accumulator(BWA), cache memory, error correcting code, Hamming distance

I. INTRODUCTION

In digital communication and information theory, error detection and correction has great practical importance in maintaining information integrity across noisy channels. Error coding is a method of detecting and correcting these errors to ensure that the information is transferred intact from its source to its destination. There are various error correcting techniques to detect and correct the error. One of the popular technique based on forward error correction is hamming code.

Data comparison is widely used in computing system such as , 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. By the use of shared cache memory it improves the parallel encoding scheme by adding a shared memory during correction feedback space. This will help to improve the power without affecting the timing. cache memory is mainly used to store the data after performing the matching between received data along with tag data.

The previous studies consider the characteristics of systematic codes in designing the architecture and proposed a low-complexity processing element that computes the Hamming distance. Therefore, the latency and the hardware complexity are decreased considerably even compared with the SA based architecture. But in this thesis work doesn't use the entire concept of previous work. By the use of shared cache memory improves the parallel encoding scheme during correction feedback space. This will also help to improve the power without affecting the timing.

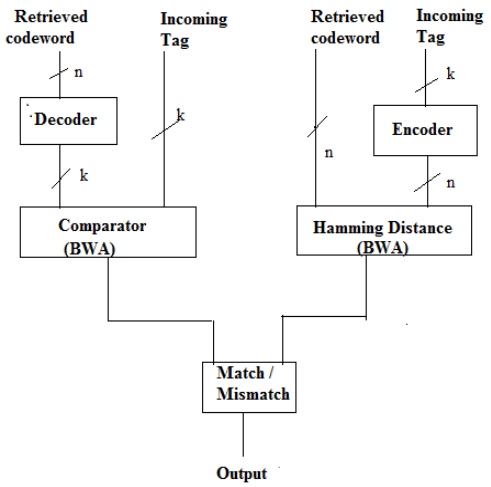

Fig 1: Architecture for BWA for finding hamming distance

The rest of this brief is organized as follows. Section II reviews previous works. The proposed architecture is explained in section III, and evaluated in section IV. Finally, conclusion are made in section V.

II. PREVIOUS WORKS

This section describes the architecture for butterfly formed weight accumulator for finding hamming distance. Let us consider a memory in which n-bit retrieved code word is first decoded and produces an output of k-bit tag. This k-bit tag is compared with the k-bit incoming tag. Likely , k-bit incoming tag is encoded and produces an output of n-bit tag which will be compared with the k-bit retrieved code word.

Here the comparison is performed by determining the hamming distance. The hamming distance is usually represented in butterfly- formed weight accumulator as shown in fig.1. The main function of BWA is to count number of 1's among the input bits. To perform BWA ,it uses multiple stages of half adders .Each HA is associated with some weight. Output of each HA is arranged in butterfly form so that the sum and carry bits are completely separated. Lastly check whether these two output where match / mismatch.

A. Error Correction Code

Error detection and correction (EDAC) techniques are used to ensure that data is correct and has not been corrupted, either by hardware failures or by noise occurring during transmission or a data read operation from Memory. There are many different error correction codes in existence. The reason for the different codes being used in different applications has to do with the historical development of the data storage, the types of data errors occurring, and the overhead associated with each of the error detection techniques. The basic concept of error detection and correction method is as follow :

1. Networks must be able to transfer data from one system to another without data can be corrupted during transmission.
2. For reliable communication, errors must be detected and corrected.

Any error-correcting code can be used for error detection and correction. Error-correcting code (ECC) or forward error correction (FEC) code is a system of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a back-channel is not required in forward error correction, and it is therefore suitable for simplex communication such as broadcasting.

III. PROPOSED ARCHITECTURE

This section presents a new architecture to further reduce latency and complexity by effectively computing the hamming distance by using the characteristics of same systematic codes in which both data as well as parity are completely separated from each other. And the representation of systematic code word are illustrated below in fig.2. This architecture introduces an cache memory to store the corrected data for future response.

The block codes in which message bits are transmitted in unaltered form are called systematic code. For error detection and error correction the use of systematic codes simplifies implementation of the decoder. For the code possess a systematic structure, code word is divided into two parts

Fig 2: structure of systematic code

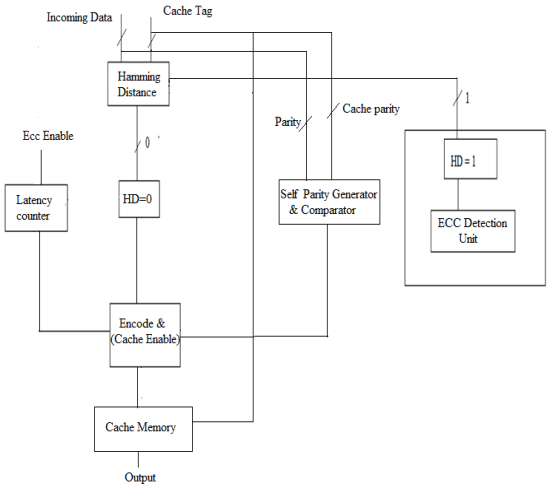

Fig 3: Proposed Architecture for efficient computation

The new proposed architecture gives better power optimization by feedback register, which will not allow the final register to change after each comparison. Block diagram for the proposed architecture is presented in fig:3. This architecture helps to get efficient result by holding the final register with previous value until the next correct result.

The proposed architecture describes the hamming distance between incoming data along with cache tag. Cache tag contains the data to be matched with incoming data. If both of them are matched by checking the hamming distance between these two data and there is no bit changes, that means HD=0. If the hamming distance is equal to zero. The data moves to encode and cache enable and hold for sometimes until the next data arrives simultaneously at the same time parity from data and self generating parity from cache tag are needed to be match if it matches. It will move to encode and cache enable. Once ECC enable makes high and enable cache. Both the data as well as parity will be encoded and reached to cache memory and gives the output. Hence comparison of both data as well as parity bit is compared. While comparing the cache data and parity which produces the corresponding output. Another condition that, the hamming distance is equal to 1(HD = 1) which implies that the bit change in one position. In this case we can either correct the error or send a acknowledgment to the transmitter to inform that the received data contains erroneous bits.

This Thesis uses enable generator to enable this signal to retransmit the data again. And mainly concentrate on detection only. This will be obtain from ECC Detection unit. Using cache memory reduces the dynamic power without affecting timing. Hence improve parallel encoding.

A.Cache Memory used for matching data

Cache memory is a type of memory used to hold frequently used data. It is relatively small, but very fast. One of the effective method to improve speed is the use of cache memory. Web browser typically uses a cache to make webpages load faster by storing a copy of the webpages files locally such as local computer called as web cache.

The interaction between CPU chip and main memory in the form of RAM is slower. Installing more memory is not possible. So that chip designer design with small form of memory that are directly located on to the chip itself called CPU cache. It is smaller and can be accessed much faster than main memory. It stores frequently used pieces of information. so that they can be retrieved more quickly. This information is a duplicate of information stored elsewhere and can be readily available.

The data present in the cache is same as the data want to match. if it matches .we can store and obtain the result. If it is not same ,data needs to be fetched first from main memory .And the comparison is performed so faster by using cache memory. Here we are performing two operations whether it equals to zero or not. If the content of both matches the result will store in the cache memory. So whenever the data is needed we can fetch it from the cache memory without going into main memory .so we can increase the speed without affecting the timing also. whenever the data is not matched we go for retransmission. By retransmission it will retransmit the data again.

B. Cache Memory Vs Main Memory

Cache and RAM are temporary memories. But may vary based on speed, size and cost. It is usually present on CPU chip itself. RAM is placed on motherboard and is connected to CPU via memory bus. Speed of Cache is closer to CPU and much faster than RAM. RAM has to travel via memory bus while CPU cache is right there. Size of Cache is less than that of primary memory. Size of primary memory or RAM in today's computers is a few GB's , while size of cache is few MB's. Also cost of Cache is more expensive than RAM.

Cache memory is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data. It stores and retains data only until a computer is powered up.

cache has a significantly shorter access time than the main memory due to the applied faster but more expensive implementation technology. The cache has a limited volume that also results from the properties of the applied technology. If information fetched to the cache memory is used again, the access time to it will be much shorter than in the case if this information were stored in the main memory and the program will execute faster.

C. Calculation of complexity and latency

This Thesis work further reduces the complexity and latency as compared with the previous work. Total number of XOR gates used is reduced. Architecture of proposed one is completely modified and the time taken by each module is also reduces. This is a one of the advantage of modifying the architecture. As compared with previous work ,the proposed one in which the arrival time and the response time may reduced in certain circumstances.
The complexity for the proposed architecture can be expressed as,

$$C = C_{XOR} + C_{AND} + C_{XNOR} \quad (1)$$

Where $C_{XOR,}$ $C_{AND,}$ C_{XNOR} are the complexity of XOR gate, AND gate and XNOR gate.
The latency for the proposed architecture can be expressed as,

$$L = L_{XOR} + L_{AND} + L_{XNOR} \quad (2)$$

Where $L_{XOR,}$ $L_{AND,}$ L_{XNOR} are the Latency of XOR gate, AND gate and XNOR gate

Slack is defined as the difference between required time and arrival time. The expression is given as,

Slack =required time – arrival time (3)

IV. EVALUATION

As the proposed architecture the latency as well as the complexity are completely reduced by using limited number of gates . Thus, increases the time of execution and also increases the speed without affecting the time. Hence there is no path delay found while executing the program in XILINX 14.2 software. Thus implementing the deign in this software successfully. And obtained results as shown below by finding atleast an error.

Table I Performance Comparison

Architecture	Latency	Complexity
Proposed architecture	2	No of blocks / components reduced
Previous architecture	12	125

Fig: 4 Simulation results by detecting error

This is the Xilinx simulation result which shows output as ecc_result0, ecc_result1, ecc_result2, ecc_result3 in which they are obtained by xor the rec_data and tag_data. Also sum_dst is obtained from ecc_result as mentioned in above graph. This graph shows result of detecting error by giving tag_line of "0001" which means that the tag_line contains memory[1] tag_data.it compares the rec_data with tag_data. Also mentioned below the top level module for the proposed architecture. Comparison results is shown in table 1

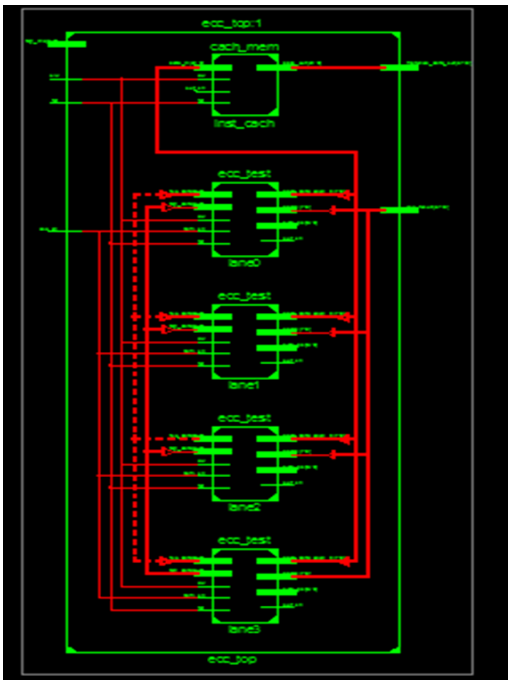

Fig: 5 Top view result of FPGA implementation

V. CONCLUSION

The architecture for matching of data wit error correcting code using cache memory is completely studied and simulated in XILINX ISE 14.2 and implemented in FPGA using virtex 5. Using cache memory effective dynamic power is reduced. And also reduce the latency and complexity. The previous work uses Butterfly formed wait accumulator for effective computation of hamming distance. But it requires many half adders to do the operations. This thesis work doesn't uses that technique. so that less number of adders are required. This is one type of benefit to this project. Hence no path delay exist in this thesis work. With the introduction of shared cache memory in feedback correction stage . Frequently used data can be taken from cache memory itself without doing any operation. We cannot find hamming distance again and again just retrieve data from cache memory.

REFERENCES

[1]. Byeong Yong Kong, Jihyuck Jo, Hyewon Jeong, Mina Hwang, Soyoung Cha, Bongjin Kim, and In-CheolPark," 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 ,July 2014

[2]. W. Wu, D. Soma sekhar, 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.

[3]. 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.

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

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