An Improved Systematic Error Correcting Code for Low Power and Low Complexity in Chip

R. Uthira Devi[1] E.Yamini[2] N.Ragavi[3]
[1,2,3]Sri Eshwar College Of Engineering

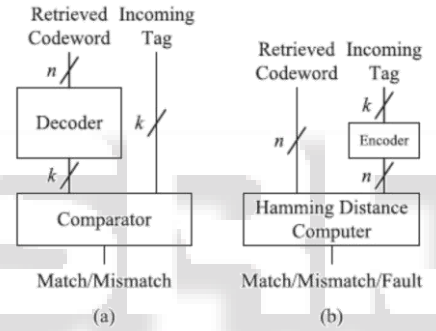
Abstract— An error-correcting code is an algorithm for expressing a sequence of numbers such that any errors which are introduced can be detected and corrected based on the remaining numbers.The Convolution code helps to improve the error performance or reduce the power consumption.The stored data is protected with error-correcting codes (ECC) for reliability reason.This method is proposed to improve the latency for information encoded with ECC. Therefore the chip area and power is reduced by the Convolution code.The proposed code examines whether the incoming data matches the stored data if a certain number of erroneous bits are detected and corrected for more than two bit.

Key words: Low Power and Low Complexity in Chip, Systematic Error Correcting Code

I. 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 fromserving as accelerators and the overall performance of the wholesystem 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 codewords, 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. In this brief, we renovate the SA-based direct compare architecture to reduce the latency and hardware complexity by resolving the aforementioned drawbacks. Therefore, the latency and the hardware complexity are decreased considerably even compared with the SA based architecture. 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, concluding remarks are made in Section V.



II. 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.

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 [7].

The implementation results in [8] support the claim. To resolve the drawbacks of the decode-and-compare,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 *t*max and *r*max 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 $0 < d \leq t\text{max}$, *X* will match *Y* provided at most *t*max errors in *Y* are corrected.
(3) If $t\text{max} < d \leq r\text{max}$, *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 $r\text{max} < d$, *X* does not match *Y*.

Assuming that the incoming address has no errors, we can regard thetwo tags as matched if *d* is in either the first or the second ranges.In this way, while maintaining the error-correcting capability, the architecture can remove the decoder from its critical path at the cost of an encoder being newly introduced. Note that the encoder is, in general, much simpler than the decoder, and thus the encoding cost is significantly less than the decoding cost.

Since the above method needs to compute the Hamming distance,[6] 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 *r*max + 1 if it exceeds *r*max.

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.

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

A. Datapath Design for Systematic Codes

In telecommunication, a convolutional code is a type of error-correcting code that generates parity symbols via the sliding application of a boolean polynomial function to a data stream. The sliding application represents the 'convolution' of the encoder over the data, which gives rise to the term 'convolutional coding.'

The sliding nature of the convolutional codes facilities trellis decoding using an essentially fixed sized trellis. Trellis decoding, in turn, allows maximum likelihood soft decision decoding of convolutional code to be done with reasonable complexity.

The ability to perform economical maximum likelihood soft decision decoding is one of their major benefits of convolutional codes. This is in contrast to classic block codes, which are hard decision decoded.

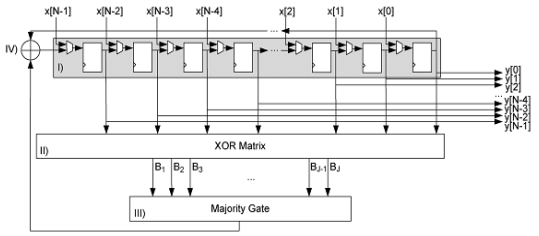
Convolutional codes are often characterized by the base code rate and the depth (or memory) of the encoder [n,k,K]. The base code rate is typically given as n/k, where n is the input data rate and k is the output symbol rate. The depth is often called the "constraint length" 'K', where the output is a function of the previous K-1 inputs. The depth may also be given as the number of memory elements 'v' in the polynomial or the maximum possible number of states of the encoder (typically 2^v).

Convolutional codes are often described as continuous. However, it may also be said that convolutional codes have arbitrary block length, rather than that they are continuous, since most real world convolution encoding is performed on blocks of data. Block processing using convolutional codes typically employs termination.

The arbitrary length of convolutional codes can also be contrasted to classic block codes, which generally have fixed block lengths that are determined by algebraic properties.

The code rate of a convolutional code is commonly modified via symbol puncturing. For example, a convolutional code of base rate n/k=1/2 may be punctured to a higher rate of, for example, 7/8 simply by not transmitting a portion of code symbols. The performance of a punctured convolutional code generally scales well with the amount of parity transmitted.

The ability to perform economical soft decision decoding on convolutional codes, as well as the block length and code rate flexibility of convolutional codes, makes them very popular for digital communications.



Convolutional codes are often described as continuous. However, it may also be said that convolutional codes have arbitrary block length, rather than that they are continuous, since most real world convolution encoding is performed on blocks of data. Block processing using convolutional codes typically employs termination.

Convolutional codes are often characterized by the base code rate and the depth (or memory) of the encoder [n,k,K]. The base code rate is typically given as n/k, where n is the input data rate and k is the output symbol rate. The depth is often called the "constraint length" 'K', where the output is a function of the previous K-1 inputs. The depth may also be given as the number of memory elements 'v' in the polynomial or the maximum possible number of states of the encoder (typically 2^v).

The arbitrary length of convolutional codes can also be contrasted to classic block codes, which generally have fixed block lengths that are determined by algebraic properties.

The code rate of a convolutional code is commonly modified via symbol puncturing. For example, a convolutional code of base rate n/k=1/2 may be punctured to a higher rate of, for example, 7/8 simply by not transmitting a portion of code symbols. The performance of a punctured convolutional code generally scales well with the amount of parity transmitted.

The ability to perform economical soft decision decoding on convolutional codes, as well as the block length and code rate flexibility of convolutional codes, makes them very popular for digital communications.

To convolutionally encode data, start with *k* memory registers, each holding 1 input bit. Unless otherwise specified, all memory registers start with a value of 0. The encoder has *n*modulo-2 adders (a modulo 2 adder can be implemented with a single Boolean XOR gate, where the logic is: 0+0 = 0, 0+1 = 1, 1+0 = 1, 1+1 = 0), and *n* generator polynomials — one for each adder (see figure below). An input bit m_1 is fed into the leftmost register. Using the generator polynomials and the existing values in the remaining registers, the encoder outputs *n* symbols. These symbols may be transmitted or punctured depending on the desired code rate. Now bit shift all register values to the right (m_1 moves to m_0,m_0 moves to m_{-1}) and wait for the next input bit. If there are no remaining input bits, the encoder continues shifting until all registers have returned to the zero state (flush bit termination).

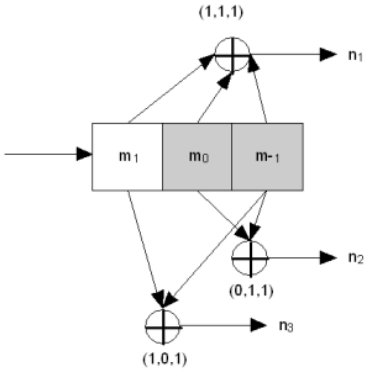
The figure below is a rate 1/3 (*m*/*n*) encoder with constraint length (*k*) of 3. Generator polynomials are G_1 = (1,1,1), G_2 = (0,1,1), and G_3 = (1,0,1). Therefore, output bits are calculated (modulo 2) as follows:

$n_1 = m_1 + m_0 + m_{-1}$

$n_2 = m_0 + m_{-1}$

$n_3 = m_1 + m_{-1}$.

Convolutional coding is a special case of error-control coding. Unlike a block coder, a convolutional coder is not a memoryless device. Even though a convolutional coder accepts a fixed number of message symbols and produces a fixed number of code symbols, its computations depend not only on the current set of input symbols but on some of the previous input symbols.



Convolutional codes are commonly described using two parameters: the code rate and the constraint length. The code rate, k/n, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. The constraint length parameter, K denotes the "length" of the convolutional encoder, i.e. how many k-bit stages are available to feed the combinatorial logic that produces the output symbols. Closely related to K is the parameter m, which indicates how many encoder cycles an input bit is retained and used for encoding after it first appears at the input to the convolutional encoder. The m parameter can be thought of as the memory length of the encoder.

Convolutional codes are widely used as channel codes in practical communication systems for error correction. The encoded bits depend on the current k input bits and a few past input bits. The main decoding strategy for convolutional codes is based on the widely used Viterbi algorithm. As a result of the wide acceptance of convolutional codes,there have been several approaches to modify and extend this basic coding scheme.Trellis coded 0modulation (TCM) and turbo codes are two such examples. In TCM,redundancy is added by combining coding and modulation into a single operation. This is achieved without any reduction in data rate or expansion in bandwidth as required by only error correcting coding schemes.

B. Complexity and Latency

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

III. EVALUATION

For a set of four codes including the (31, 25) code quoted from [6],from three architectures: 1) the conventional decode-and-compare;2) the SA-based direct compare; and 3) the proposed ones 4)convolution code.We measured the metrics at the gate-level first and then implemented the circuits in a 0.13-μm CMOS technology to provide more realistic results by deliberating some practical factors, e.g., gate sizing and wiring delays. In [6], 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 are all measured in this way. Besides, critical-path delays are obtained by performing post layout simulations and equivalent gate counts are measured by counting a two-input NAND as one.As 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 codewords.

IV. 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. 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.

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. 2012.1652 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 22, NO. 7, JULY 2014

[2] 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.

[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] 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

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

[7] 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.

