International Journal of Research
Available at https://edupediapublications.org/journals
Special Issue on Conference Papers
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 06
March 2018

A Fast Method for Error Correction Codes of Single Bit with Rapid Decoding For A Subset Of Critical Bits.

[1]**Boorla Santhosh,** [2]**Dr. T.Srinivas**

[1](*Research scholar(25617082), Dept of ECE, Shri Jagadishprasad Jhabarmal Tibrewala Uinversity, Jhunjhunu,Rajasthan .&*
Assistant Professor, Dept. Electronics & Communication Engineering
Mother Theressa College Of Engg & Tech. Peddapalli, , India
[2]*professor, Dept. Electronics & Communication Engineering . Peddapalli, , India*
E-Mail: santhosh.b443@gmail.com[1] *, tspdpl@gmail.com*[2]

ABSTRACT:

Single error correction (SEC) codes are widely usedto protect data stored in memories and registers. In some applications, such as networking, a few control bits are added tothe data to facilitate their processing. For example, flags to mark the start or the end of a packet are widely used. Therefore, it isimportant to have SEC codes that protect both the data and the associated control bits. It is attractive for these codes to provide fast decoding of the control bits, as these are used to determine the processing of the data and are commonly on the critical timing path. In this brief, a method to extend SEC codes to support a fewadditional control bits is presented. The derived codes support fastdecoding of the additional control bits and are therefore suitablefor networking applications.

I.INTRODUCTION

Networking applications require high-speed processing of data and thus rely on complex integrated circuits [1]. In routers and switches, packets typically enter the device through one port, are processed, and are then sent to one or more output ports. During this processing, data are stored and moved through the device [2]. Reliability is a key requirement for networking equipment such as core routers [3]. Therefore, the stored data must be protected to detect and correct errors. This is commonly done using error-correcting codes (ECCs) [4]. For memories and registers, single error correction (SEC) codes that can correct 1-bit errors are commonly used [5], [6].

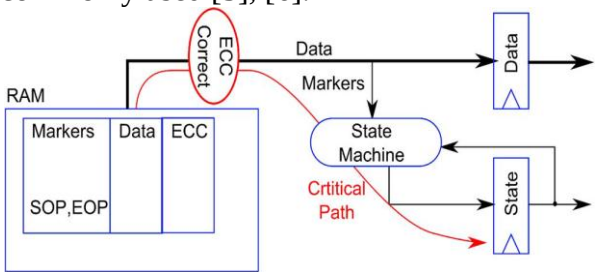

Fig. 1. Typical packet data storage in a networking application.

One problem that occurs when protecting the data in networking applications is that, to facilitate its processing, a few control bits are added to each data block. For example, flags to mark the start of a packet (SOP), the end of a packet (EOP), or an error (ERR) are commonly used [7]. These flags are used to determine the processing of the data, and the associated control logic is commonly on the critical timing path. To access the control bits, if they are protected with an ECC, they must first be decoded. This decoding adds delay and may limit the overall Fig. 1. Typical packet data storage in a networking application. frequency. One option is to protect the data and the control bits as different data blocks using separate ECCs. For example, let us assume 128-bit data blocks with 3 control bits. Then, a SEC code can protect a data block using 8 parity check bits, and another SEC code can protect the 3 control bits using 3 parity check bits. This

option provides independent decoding of data and control bits which reduces the delay but

requires additional parity check bits. Another option is to use a single ECC to protect both the data and control bits. Protecting 128 + 3 bits requires only 8 parity check bits, thus saving 3 bits compared to the use of separate ECCs. However, in this case, the decoding of the control bits is more complex and incurs more delay. In this brief, a method to extend a SEC code to also protect a few additional control bits is proposed. In the resulting codes, the control bits can be decoded using a subset of the parity check bits. This reduces the decoding delay and makes them suitable for networking applications. To evaluate the method, several codes have been constructed and implemented. They are then compared with existing solutions in terms of decoding delay and area.

II. CONCURRENT ERROR DETECTION SCHEMES

TYPES OF ERROR DETECTI ON SCHEMES
Schemes for Error Detection find wide range of applications, since only after the detection of error, can any preventive measure be initiated. The principle of error detecting scheme is very simple, an encoded codeword needs to preserve some characteristic of that particular scheme, and a violation is an indication of the occurrence of an error. Some of the error detection techniques are discussed below Parity Codes. These are the simplest form of error detecting codes, with a hamming distance of two (d=2), and a single check bit (irrespective of the size of input data). They are of two basic types: Odd and Even. For an even -parity code the check bit is defined so that the total number of 1s in the code word is always even; for an odd code, this total is odd. So, whenever a fault affects a single bit, the total count gets altered and hence the fault gets easily detected. A major drawback of these codes is that their multiple fault detection capabilities are very limited. Checksum Codes:In these codes the summation of all the information bytes is

appended to the information as bit checksum. Any error in the transmission will be indicated as a resulting error in the checksum. This leads to detection of the error. When b=1, these codes are reduced to parity check codes. The codes are systematic in nature and require simple hardware units.
2.1.3 m-out-of-n Codes: In this scheme the codeword is of a standard weight m and standard length n bits. Whenever an error occurs during transmission, the weight of the code word changes and the error gets detected. If the error is a 0 to 1 transition an increase in weight is detected, similarly 1 to 0 leads to a reduction in weight of the code, leading to easy detection of error. This scheme can be used for detection of unidirectional errors, which are the most common form of error in digital systems.
Berger Codes: Berger codes are systematic unidirectional error detecting codes. They can be considered as an extension of the parity codes. Parity codes have one check bit, which can be considered as the number of information bits of value 1 considered in modulo2. On the other hand Berger codes have enough check bits to represent the count of the information bits having value 0. The number of check bits (r) required for k -bit information is given by r = [log 2 (k − 1)] Of all the unidirectional error detecting codes that exist suggests, m - out of - n codes to be the most optimal.
These codes however, are not of much application because of its non separable nature. Amongst the separable codes in use, the Berger codes have been proven to be most optimal, requiring the smallest number of check bits.The Berger Codes, however, are not optimal when only t unidirectional errors need to be detected instead of all unidirectional errors. For this reason a number of different modified Berger codes exist: Hao Dong introduced a code that accepts slightly reduced error detection capabilities, but does so using fewer check bits and smaller checker sizes. In

this code the number of check bits is independent of the number of information bits. Bose and Lin have introduced their own variation on Berger codes and Bose has further introduced a code that improves on the burst error detection capabilities of his previous code, where erroneous bit are expected to appear in groups.

III. PROPOSED METHOD TO DESIGN THE CODES

As discussed in the introduction, the goal is to design SEC codes that can protect a data block plus a few control bits
such that the control bits can be decoded with low delay. As mentioned before, the data blocks to be protected have a size that is commonly a power of two, e.g., 64 or 128 bits. To protect a 64-bit data block with a SEC code, 7 parity check bits are needed, while 8 are enough to protect 128 bits. In the first case, there are 27 = 128 possible syndromes, and therefore, the SEC code can be extended to cover a few additional control bits. The same is true for 128 bits and, in general, for a SEC code that protects a data block that is a power of two.

This means that the control bits can also be protected with no additional parity check bits. This is more efficient than using two separate SEC codes (one for the data bits and the other for the control bits) as this requires additional parity check bits. The main problem in using an extended SEC code is that the decoding of the control bits is more complex. To illustrate this issue, let us consider a 128-bit data block and 3 control bits. The initial SEC code for the 128-bit data block has the parity check matrix shown in Fig. 2. This code has a parity check matrix with minimum total weight and balanced row weights to minimize encoding and decoding delay [4].

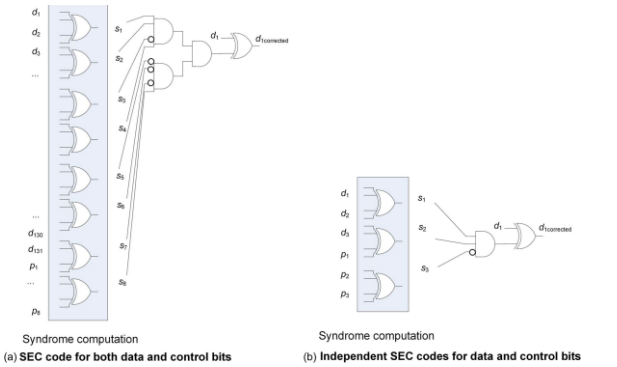

Fig. 2 Decoding of a control bit for single and independent SEC codes for data and control. (a) SEC code for both data and control bits. (b) Independent SEC codes for data and control bits.

Three additional data columns can be easily added to obtain a code that protects the additional control bits. For example, the matrix in Fig. 3 can be used, in which three additional columns (marked as control bits) have been added to the left. The problem is that now, to decode the 3 control bits, we need to compute the 8 parity check bits and compare the results against the columns of the control bits. This is significantly more complex than the decoding of an independent SEC code for the three control bits. The decoding of a bit in each case is shown in Fig. 2, and the difference in complexity is apparent.

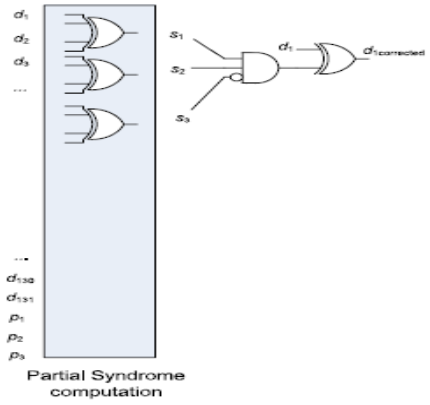

Fig. 3. Bit decoding of a control bit in the proposed SEC code.

As discussed earlier, our goal is to simplify the decoding of the control bits while using a single SEC code for both data and control bits. To do so, the first step is to note that, in some

cases, SEC decoding can be simplified to check only some of the syndrome bits. One example is the decoding of constant-weight SEC codes proposed in [11]. In this case, only the syndrome bits that have a 1 in the column of the parity check matrix need to be checked. This simplifies the decoding for all bits but, in most cases, requires additional parity check bits. In our case, the main focus is to simplify the decoding of the control bits as those are commonly on the critical path. To do so, the parity check bits can be divided in two groups: a first group that is shared by both data and control bits and a second that is used only for the data bits. Then, the decoding of the control bits only requires the recomputation of the first group of parity check bits. This scheme is better illustrated with an example. Let us consider a 128-bit data block and 3 control bits protected with 8 parity check bits. Those 8 bits are divided in a group of 3 shared between data and control bits and a second group of 5 that is used only for the data bits.To protect the control bits, the first three parity check bits can be assigned different values for each control bit, and the remaining parity check bits are not used to protect the control bits. The rest of the values are used to protect the data bits, and for each value, different values of the remaining five parity check bits can be used. In this example, the first group has 3 bits that can take 8 values, and three of them are used for the columns that correspond to the control bits. This leaves 5 values that can be used to protect the data bits. The second group of parity check bits has 5 bits that can be used to code 32 values for each of the 5 values on the first group. Therefore, a maximum of $5 \times 32 = 160$ data bits can be protected. In fact, the number is lower as the zero value on the first group cannot be combinedwith a zero or a single one

on the second group as the corresponding column would have weight of zero or one. In any case, 128 data bits can be easily protected.
An example of the parity check matrix of a SEC code derived using this method is shown in Fig. 2. The three first columns correspond to the added control bits. The two groups of parity check bits are also separated, and the first three rows are shared for data and control bits, while the last five only protect the data bits. It can be observed that the control bits can be decoded by simply recomputing the first three parity check bits. In addition, the zero value on these three bits is also used for some data bits. This means that those bits are not needed to recompute the first three parity check bits. The decoding of one of the control bits is illustrated in Fig. 3. It can be observed that the circuitry is significantly simpler than that of a traditional SEC code (see left part of Fig. 4). This will be confirmed by the experimental results presented in the next section. The method can also be used to protect more than three control bits. In a general case, let us consider that we need to protect d data bits and c control bits using p parity check bits. Then, p is divided in two groups *pcd* and *pd*. The first group is shared between control and data bits, and the second is used only for the data bits. The proposed codes do have an impact on the decoding delay for the data bits. For the decoders, the added delay on data bits is significant for most word sizes. However, as discussed in the introduction, the major design goal is to reduce the decoding delay of the control bits as these typically determine the critical timing path.

V. CONCLUSION AND FUTURE WORK

In this brief, a method to construct SEC codes that can protect a block of data and some additional control bits has been presented. The derived codes are designed to enable fast decoding of the control bits. The derived codes

have the same number of parity check bits as existing SEC codes and therefore do not require additional cost in terms of memory or registers. To evaluate the benefits of the proposed scheme, several codes have been implemented and compared with minimum-weight SEC codes. The proposed codes are useful in applications, where a few control bits are added

to each data block and the control bits have to be decoded with low delay. This is the case on some networking circuits. The scheme can also be useful in other applications where the critical delay affects some specific bits such as in some finite-state machines. Another example is arithmetic circuits where the critical path is commonly on the least significant bits. Therefore, reducing the delay on those bits can increase the overall circuit speed. The use of the proposed scheme for those applications beyond networking is an interesting topic for future work. It may be possible to apply the idea of modifying the matrix of the code to enable fast decoding of a few bits to more advanced ECCs that can correct multiple bit errors. Finally, the scheme can also be extended to support more control bits by using one or two additional parity check bits. This would provide a solution to achieve fast decoding without using two separate codes for data and control bits.

REFERENCES

[1] P. Bosshart *et al.*, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," in *Proc. SIGCOMM*, 2013, pp. 99–110.
[2] J. W. Lockwood *et al.*, "NetFPGA—An open platform for gigabit-rate network switching and routing," in *Proc. IEEE Int. Conf. Microelectron. Syst. Educ.*, Jun. 2007, pp. 160–161.
[3] A. L. Silburt, A. Evans, I. Perryman, S.-J. Wen, and D. Alexandrescu, "Design for soft error resiliency in Internet core routers," *IEEE*

Trans. Nucl. Sci., vol. 56, no. 6, pp. 3551–3555, Dec. 2009.
[4] E. Fujiwara, *Code Design for Dependable Systems: Theory and Practical Application*. Hoboken, NJ, USA:Wiley, 2006.
[5] 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, Mar. 1984.
[6] V. Gherman, S. Evain, N. Seymour, and Y. Bonhomme, "Generalized parity-check matrices for SEC-DED codes with fixed parity," in *Proc. IEEE On-Line Test. Symp.*, 2011, pp. 198–20.
[7] Ten Gigabit Ethernet Medium Access Controller, OpenCores. [Online]. Available: http://opencores.org/project/ethmac
[8] P. Zabinski, B. Gilbert, and E. Daniel, "Coming challenges with terabitper-second data communication," *IEEE Circuits Syst. Mag.*, vol. 13, no. 3, pp. 10–20, 3rd Quart. 2013.
[9] UltraScale Architecture Integrated Block for 100 G Ethernet v.14. LigCOREIP Product Guide. PG165, Xilinx, San Jose, CA, USA. Jan. 22, 2015.
[10] OpenSilicon Interlaken ASIC IP Core. [Online]. Available: www.opensilicon.Com /open-silicon-ips /interlaken-controller-ip/
[11] P. Reviriego, S. Pontarelli, J. A. Maestro, and M. Ottavi, "A method to construct low delay single error correction (SEC) codes for protecting data bits only," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 3, pp. 479–483, Mar. 2013.
[12] J. E. Stine *et al.* "FreePDK: An open-source variation-aware design kit," in *Proc. IEEE Int. Conf. Microelectron. Syst. Educ.*, Jun. 2007, pp. 173–174.