ENHANCING FAST DECODING OF SINGLE ERROR CORRECTION CODES FOR ESSECTIAL BIT GROUPS

[1]Syed Karimoon,[2]Mr. C. Bhargav
[1]M.Tech Student,[2]Assistant Professor
Department of Electronics and Communication Engineering
St. Johns College Of Engineering & Technology,Yerrakota, Yemmiganur,Kurnool

ABSTRACT

Single error correction (SEC) codes are widely used to protect data stored in memories and registers. In some applications, such as networking, a few control bits are added to the datato facilitate their processing. For example, flags to mark the start or the end of a packet are widely used. Therefore, it is important 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 few additional control bits is presented.

The derived codes support fast decoding of the additional control bits and are therefore suitable for networking applications. Environmental interference and physical defects in the communication medium can cause random bit errors during data transmission. Error coding is a method of detecting and correcting these errors to ensure information is transferred intact from its source to its destination. Error coding is used for fault tolerant computing in computer memory, magnetic and optical data storage media, satellite and deep space communications, network communications, cellular telephone networks, and almost any other form of digital data communication. Error coding uses mathematical formulas to encode data bits at the source into longer bit words for transmission.

The "code word" can then be decoded at the destination to retrieve the information. Different error coding schemes are chosen depending on the types of errors expected, the communication medium's expected error rate, and whether or not data retransmission is possible. Faster processors and better communications technology make more complex coding schemes, with better error detecting and correcting capabilities, possible for smaller embedded systems, allowing for more robust communications.

I. INTRODUCTION

Complex integrated circuits are necessary for networking applications because they need to handle data quickly [1]. Packets usually enter routers and switches by a single port, undergo processing, and then be routed to one or more output ports. Data are saved and sent across the device during this operation [2].

For networking devices like core routers, reliability is a crucial necessity [3]. Therefore, in order to identify and fix mistakes, the stored data needs to be safeguarded. Error-correcting codes (ECCs) are frequently used for this [4]. Single error correction (SEC) algorithms that can fix 1-bit mistakes are frequently utilised for registers and memory [5, 6].

One issue with data protection in networking applications is that each data block has a few control bits added to it to make processing easier. Flags are often used, for instance, to indicate an error (ERR), the start of a packet (SOP), or the end of a packet (EOP) [7]. The accompanying control logic is frequently on the crucial time route, and these flags are utilised to decide how the data is processed. If the control bits are secured by an ECC, they must be decoded before they may be accessed. The overall frequency may be limited by this decoding, which also adds latency. One choice is to use distinct ECCs to safeguard the data and control bits as distinct

data blocks. Let's use 128-bit data blocks with three control bits as an example. Next, an SEC code can use eight parity check bits to secure a data block, and another SEC code can use three parity check bits to protect the three control bits. Although this method minimises the latency by allowing data and control bits to be decoded independently, it necessitates additional parity check bits. Using a single ECC to safeguard the control and data bits is an additional choice. Compared to using independent ECCs, protecting 128 + 3 bits only requires 8 parity check bits, saving 3 bits. But in this instance, the control bit decoding is more difficult and takes longer.

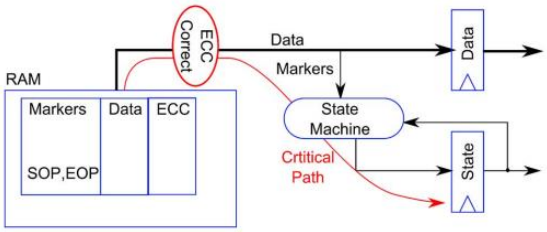

Fig. 1. Typical packet data storage in a networking application

This brief suggests a way to add a few more control bits to an SEC code such that they are likewise protected. A portion of the parity check bits in the resultant codes can be used to decode the control bits. As a result, they are appropriate for networking applications and have a lower decoding time. A number of codes have been developed and put into use in order to assess the approach. They are then contrasted with current solutions in terms of area and decoding latency.

II. DATA PROTECTION IN NETWORKING APPLICATIONS

Data speeds ranging from 10 to 400 Gbit/s are supported by current networking technology, and terabit rates are anticipated soon [8]. Current ASICs generally employ clock rates between 300 MHz and 1 GHz, while FPGAs often use lower clock frequencies (less than 400 MHz). On-chip packet data buses are broad, often ranging from 64 to 2048 bits in width, to accommodate these high data speeds [9], [10].

Fig. 2. Parity check matrix for a minimum-weight SEC code that protects 128 data bits.

Fig. 3. Parity check matrix for a minimum-weight SEC code that protects 128 data and 3 control bits.

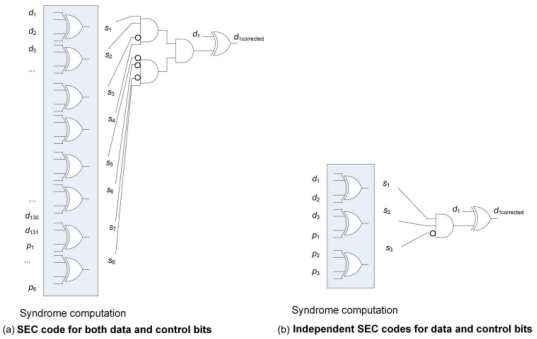

Fig. 4. Decoding of a control bit for single and independent SEC codes for data and control. (a) SEC code for both data and control bits. (b) Independent SEC codes for data and control bits.

In order to adjust processing speeds, packet data must often be kept in RAMs, for example, in FIFOs. Delineating the packet boundaries is essential for storing packet data. In the most basic scenario, a single EOP marker can be used to identify each bus section. The subsequent packet is thus presumed to begin with the next valid section. In reality, designers also indicate the beginning of packets explicitly using a SOP marker. In packet processing, there are also several instances where a packet contains errors and needs to be discarded. An extra control signal (ERR) could be needed to identify such errored packets [7].

As stated in the introduction, storing the data and the markers in a single broad memory, as illustrated in Fig. 1, is appealing from the standpoint of error prevention. Consequently, comparatively fewer ECC bits are needed. When the data is read out, this method has an issue. Usually, a state machine that regulates the reading of the ensuing data receives the markers as input. For instance, the state

machine could have to read out a certain number of bytes of data (such as in a deficit round robin scheduler) or just one packet (up to an EOP). As indicated in red, the ECC correction logic and the state machine logic make up the important time route. In a conventional Hamming SEC code, the number of logic layers needed to decode the syndrome and carry out correction rises in tandem with the width of the data bus. Critical time on the signal channels associated with the adjustment of the markers that feed downstream state machines is a common observation made by circuit designers. The ability of unique ECC codes to quickly decode the limited amount of marker bits makes them very appealing.

In some situations, the system can handle the packet data with a block size granularity. For instance, when the data is only being moved from one place to another, this would be the situation. On the other hand, knowing the packet data size with a byte precision is crucial in other situations. This would be the situation when checks are made for maximum transfer unit length or when bit rate is crucial (for scheduling and policing). It could be necessary to retain extra marker bits called EOPSIZE, which indicate how many of the bytes in the EOP transfer are legitimate, because the basic SOP and EOP markers are insufficient to determine the precise packet size. Keep in mind that any transfers made before the EOP are always deemed to be complete. Therefore, an extra 4 EOPSIZE bits could be needed on a 128-bit data bus, increasing the total number of marker bits to 7 (SOP, EOP, ERR, and EOPSIZE[3:0]).

Fig. 5. Proposed parity check matrix for a SEC code that protects 128 data and 3 control bits.

III. PROPOSED METHOD TO DESIGN THE CODES

The objective is to create SEC codes that can safeguard a data block together with a few control bits while allowing for low-latency decoding of the control bits, as was covered in the introduction. As previously stated, the size of the data blocks that need to be safeguarded is often a power of two, such as 64 or 128 bits. Seven parity check bits are required to secure a 64-bit data block with an SEC code, but eight are sufficient to secure 128 bits. Since there are 27 = 128 potential syndromes in the first scenario, a few more control bits can be covered by the SEC code. This also applies to 128 bits and, more generally, to an SEC code that safeguards a power-of-two data block. This implies that no extra parity check bits are needed to safeguard the control bits. This is more effective than employing two distinct SEC codes, one for the control bits and one for the data bits, which necessitates extra parity check bits. The primary issue with using an expanded SEC code is the complexity of the control bit decoding. Let's look at a 128-bit data block with three control bits to demonstrate this problem. The parity check matrix for the 128-bit data block's first SEC code is displayed in Fig. 2. To reduce encoding and decoding latency, this algorithm has a parity check matrix with balanced row weights and a minimal total weight [4]. To get a code that secures the extra control bits, it is simple to add three more data columns. For instance, the matrix shown in Figure 3, which has three extra columns (designated as control bits) added to the left, can be utilised.

The issue is that in order to decode the three control bits, we must first compute the eight parity check bits and then compare the output to the control bits' columns. Compared to decoding an individual SEC code for the three control bits, this is far more complicated. Figure 4 illustrates the decoding of a bit in each scenario, demonstrating the variation in complexity.

As previously said, our objective is to use a single SEC code for both data and control bits while making the decoding of the control bits simpler. The first step in doing so is to acknowledge that SEC decoding can occasionally be streamlined to merely examine a subset of the syndrome bits. The decoding of constant-weight SEC codes, as suggested in

[11], is one instance. Only the syndrome in this instance

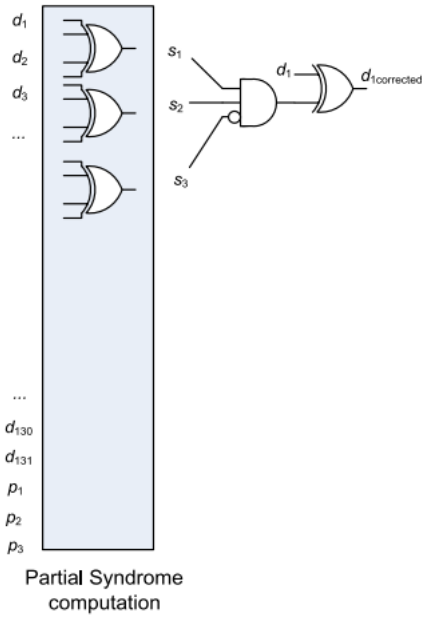

Fig. 6. Bit decoding of a control bit in the proposed SEC code.

TABLE I
MINIMUM NUMBER OF Pcd BITS FOR 128 AND 256 DATA BITS

Control bits	128 Data Bits	256 Data Bits
3	3	3
4	4	4
5	4	4
6	4	4
7	4	4
8	5	5

It is necessary to check the bits that have a 1 in the parity check matrix's column. This makes decoding all bits easier, but it usually necessitates extra parity check bits. Since the control bits are frequently on the crucial route, the primary goal in our situation is to make decoding them easier. This may be accomplished by splitting the parity check bits into two groups: one that is utilised exclusively for the data bits and the other that is shared by the control and data bits. The first set of parity check bits simply has to be recalculated in order to decode the control bits. An example is a better way to explain this method. Let's look at a 128-bit data block with three control bits that are shielded by eight parity check bits. These eight bits are separated into two groups: one of five is used exclusively for the data bits, while the other group of three is shared by the control and data bits. The first three parity check bits can be given distinct values for each control bit in order to secure the control bits; the remaining parity check bits are not utilised in this way. Different values of the remaining five parity check bits can be used for each value, and the remaining values are utilised to secure the data bits. Three of the eight possible values in the first set of bits in this example are utilised for the columns that match the control bits. There are five values left that can be utilised to safeguard the data bits. Each of the five values on the first set of parity check bits can be coded with one of the five bits in the second group. Thus, the maximum number of data bits that may be safeguarded is $5 \times 32 = 160$. Since the appropriate column would have a weight of zero or one, the zero value on the first group cannot be joined with another zero or a single one on the second group, so the number is really lower. In any event, protecting 128 data bits is simple.

Fig. 7. Proposed parity check matrix for a SEC code that protects 128 data and 7 control bits.

Figure 5 illustrates the parity check matrix of an SEC code that was obtained using this technique. The additional control bits are represented by the first three columns. The latter five rows exclusively safeguard the data bits, whereas the first three rows share the data and control bits. The two sets of parity check bits are likewise divided. It is evident that the first three parity check bits only need to be recalculated in order to decode the control bits. Additionally, certain data bits also use the zero value on these three bits. This indicates that the first three parity check bits may be recalculated without those bits.

Fig. 6 shows the decoding of one of the control bits. As can be seen from the left portion of Fig. 4, the circuitry is much simpler than that of a conventional SEC code. The experimental findings in the next section will support this.

More than three control bits can be protected using this technique. Let's assume for the

moment that we need to use p parity check bits to secure d data bits and c control bits. P is then separated into two categories, pcd and pd. Control and data bits share the first group, but only the data bits utilise the second. The following formula can be used to determine how many data bits this technique can protect. There are 2P cd − c combinations of the first group that can be utilised to secure the data bits. A total of (2P cd − c) · 2P d can be obtained by using up to 2P d values for each of those. However, pd + 1 should be deducted since the combinations of the second group with weight zero or one cannot be utilised for the zero value. Likewise, the zero value on the second group cannot be utilised for the pcd values with weight one on the first group since the resulting column would have weight one. As a result, pcd must also be deducted, resulting in (2P cd − c) · 2P d − (pd + 1) − pcd. This is how many data bits, on top of the control bits, can be secured. To be able to secure the block of data bits with the same number of parity check bits, the PCD must be increased in tandem with the number of control bits. Table I provides examples for 128 and 256 data bits. The minimal value should be utilised since increasing pcd complicates control bit decoding.

TABLE II
ASIC CIRCUIT AREA (μM2) FOR 3 ADDITIONAL CONTROL BITS

	Minimum weight SEC code	Proposed SEC code
Encoder (64+3 all bits)	250.6	247.9
Decoder (64+3 all bits)	589.7	575.9
Encoder (128+3 all bits)	489.2	494.5
Decoder (128+3 all bits)	1063.2	1078.9
Encoder (256+3 all bits)	930.5	947.2
Decoder (256+3 all bits)	1822.6	1910.1

TABLE III
ASIC CIRCUIT DELAY (NS) FOR 3 ADDITIONAL CONTROL BITS

	Minimum weight SEC code	Proposed SEC code
Encoder (64+3, all bits)	0.48	0.48
Decoder (64+3, control bits)	0.63	0.55
Decoder (64+3, data bits)	0.71	0.76
Encoder (128+3, all bits)	0.66	0.67
Decoder (128+3, control bits)	0.75	0.63
Decoder (128+3, data bits)	0.88	0.98
Encoder (256+3, all bits)	0.85	0.88
Decoder (256+3, control bits)	0.92	0.75
Decoder (256+3, data bits)	1.02	1.29

Fig. 7 illustrates the parity check matrix used to safeguard 128 data and 7 control bits. It is evident that in this instance, the first group requires more bits, which makes the control bit decoding a little more difficult. Instead of the eight bits needed for a conventional SEC code, the control bits can still be deciphered using just four syndrome bits. Lastly, it should be mentioned that in the event of multiple mistakes, the suggested technique raises the risk of miscorrection for control bits. This is because just a portion of the bits are used to decode the control bits.

IV. EVALUATION

The suggested system has been developed for 64, 128 and 256 data bits while taking into account three and seven more control bits in order to evaluate its merits. The codes shown in Figures 5 and 7 are the same as those used for the case of 128 data bits. The minimumweight SEC codes with balanced row weight (shown in Fig. 3 for the case of 128 data bits and 3 control bits) are compared to the encoders and decoders. The smallest decoding latency for a conventional SEC code ought to be offered by these codes.

All of the designs have been built in HDL and mapped to a 45-nm ASIC library using Synopsis DC in order to assess the suggested codes for an ASIC implementation [12]. As the primary design objective for the decoders, the synthesis was set up to focus most of its efforts on minimising latency on the control bits. The tool was set up to minimise delay on all bits for the encoders. The suggested codes and the minimum-weight codes were subjected to the same synthesis limitations in every instance. Both the delay and the circuit area have been assessed.

TABLE IV ASIC CIRCUIT AREA (μM2) FOR 7 ADDITIONAL CONTROL BITS

	Minimum weight SEC code	Proposed SEC code
Encoder (64+7 all bits)	264.7	266.0
Decoder (64+7 all bits)	607.5	581.2
Encoder (128+7 all bits)	501.7	488.7
Decoder (128+7 all bits)	1081.3	1084.5
Encoder (256+7 all bits)	956.0	937.9
Decoder (256+7 all bits)	1892.9	1947.1

TABLE V ASIC CIRCUIT DELAY (NS) FOR 7 ADDITIONAL CONTROL BITS

	Minimum weight SEC code	Proposed SEC code
Encoder (64+7, all bits)	0.54	0.54
Decoder (64+7, control bits)	0.67	0.60
Decoder (64+7, data bits)	0.72	0.80
Encoder (128+7, all bits)	0.67	0.67
Decoder (128+7, control bits)	0.81	0.72
Decoder (128+7, data bits)	0.89	1.02
Encoder (256+7, all bits)	0.86	0.87
Decoder (256+7, control bits)	0.92	0.83
Decoder (256+7, data bits)	0.99	1.34

Tables II and III provide the outcomes for the scenario with three more control bits. The findings for the minimum-weight SEC codes are likewise included in the tables. In this instance, there is a 12%–18% decrease in the control bits' decoding latency. This demonstrates how the suggested plan could shorten the critical path. Sometimes somewhat lower, sometimes slightly higher, the circuit area is comparable to the minimum-weight SEC codes.

The decoding latency for the data bits is affected by the suggested codes. For the majority of word sizes, the additional delay on data bits is substantial for the decoders. However, as the control bits usually dictate the crucial time route, the main design objective is to decrease the decoding delay of the control bits, as was covered in the introduction.

Tables IV and V provide the outcomes for the scenario with seven control bits. The circuit space needed for the encoder and decoder in the suggested codes is comparable to that of the minimum-weight codes. The data bits decode more slowly in terms of delay. However, the control bits' decoding latency can be decreased by around 9% to 11% using the suggested codes. Compared to the situation with three control bits, this decrease is less. This is to be expected when the decoder complexity rises along with the amount of parity bits (pcd) utilised to decode the control bit (from three to four). As a result, as the number of control bits rises, the advantages of the suggested method diminish.

In conclusion, the suggested technique may be applied to decrease the control bits' decoding time, particularly when there aren't many control bits.

V. CONCLUSION AND FUTURE WORK

A technique for creating SEC codes that can safeguard a data block and a few extra control bits has been provided in this short. The derived codes are created to allow the control bits to be decoded quickly. The derived codes don't need more memory or registers since they contain the same amount of parity check bits as the current SEC codes. A number of codes have been put into place and contrasted with minimum-weight SEC codes in order to assess the advantages of the suggested plan.

Applications where a few control bits are introduced to each data block and the control bits must be decoded quickly can benefit from the suggested codes. On certain networking circuits, this is the case. In other situations, such as in some finite-state machines, where the critical delay impacts certain bits, the approach may also be helpful. Arithmetic circuits are another example, where the least important bits often have the critical route. Consequently, the overall circuit speed may be raised by decreasing the delay on those bits. An intriguing area for further research is the applicability of the suggested technique to those applications outside of networking. More sophisticated ECCs that can fix multiple bit faults could be able to use the concept of changing the code matrix to allow for quick decoding of a few bits. Lastly, by adding one or two more parity check bits, the system may be expanded to include more control bits. This would offer a way to do quick decoding without utilising two different codes for the control and data 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.