Design of Two Interleaved Error Detection and Corrections using Hsiao Code and CRC

Authors:
Sagar Wani, Sr Technical Architect, Accenture
Ganesh Nemade, Technical Architect

Abstract

With the continuous downscaling of CMOS technologies, the issues of soft errors and reliability are set to become increasingly challenging. In order to detect and correct these errors and enhance reliability of the circuits, this paper describes a design of two interleaved Double-Adjacent-Error-Corrections (DAECs) for Error Detection and Correction (EDAC), using the Hsiao Code and Cyclic Redundancy Code (CRC). Hsiao codes are modified version of Hamming codes which are widely used in modern systems. A cyclic redundancy check (CRC) is a non-secure hash function designed to detect accidental changes to digital data in computer networks. It is characterized by specification of a generator polynomial, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend. The remainder becomes the result. CRCs are particularly easy to implement in hardware and are therefore commonly used. We also propose two storage formats and algorithm designs that can manage and store the 48-bit codeword in 8-bit and 16-bit memory devices, a typical satellite scenario where board space is scarce. We use Verilog, C++, and ModelSim to create and test our designs.

Introduction

As the geometry of the semiconductors gets smaller in the fast-growing CMOS industry, the reliability of memory devices is difficult to maintain as they are at threat due to external influences such as an electrical surge and ionizing radiation. With the growing reliability concerns, detection and correction of soft errors becomes critical.

The existence of soft errors in System on Chips (SoC), especially in embedded memories, is already well known for a long time as a result of interaction with charged particles or due to radiation. A soft error is a type of error where a signal or datum is wrong. Errors may be caused by a defect, usually understood either to be a mistake in design or construction, or a broken component. A soft error is also a signal or datum which is wrong, but is not assumed to imply such a mistake or breakage. One cause of soft errors is single event upsets from cosmic rays.

Single Event Errors (SEU) do not destroy the circuits involved, but they can cause errors. In space-based microprocessors, one of the most vulnerable portions is often the 1st and 2nd-level cache memories, because these must be very small and have very high speed, which means that they do not hold much charge. Often these caches are disabled if terrestrial designs are being configured to survive SEUs.

There are several prominent single error correction and double error detection (SEC-DED) EDACs proposed, such as the Hamming, Bose– Chaudhuri–Hocquenghem (BCH), and Hsiao Codes. The Hsiao Code EDAC has the best performance for speed, hardware requirement, and three- and four-bit-error detection. Its higher three- and four-bit-error detection rates reduce the probability of an erroneous EDAC correction.

The most-common error-correction codes use Hamming or Hsiao codes that provide single-bit error correction and double-bit error detection (SEC-DED). Other error-correction codes have been proposed for protecting memory – double-bit error correcting and triple-bit error detecting (DEC-TED) codes, single-nibble error correcting and double-nibble error detecting (SNC-DND) codes, Reed–Solomon error correction codes, etc. However, in practice, multi-bit correction is usually implemented by interleaving multiple SEC-DED codes.

Early research attempted to minimize the area and delay overheads of ECC circuits. Hamming first demonstrated that SEC-DED codes were possible with one particular check matrix. Hsiao showed that an alternative matrix with odd weight columns provides SEC-DED capability with less hardware area and shorter delay than traditional Hamming SEC-DED codes. More recent research also attempts to minimize power in addition to minimizing area and delay.

Our approach is to design two fast EDACs that are capable of correcting and detecting these errors. The first EDAC uses the Hsiao Code and requires an encoder matrix. The second EDAC uses a CRC polynomial that requires an encoder matrix. Then interleaving of these two Double-Adjacent-Error-Correction (DAEC) EDAC designs are carried out. The paper also proposes a storage format and algorithm designs to manage and store the 48-bit codeword in 8-bit and 16-bit memory devices.

Literature Survey

In this section, a brief introduction of the work that has been done with respect to this field is discussed.

The interleaving of Hsiao Code and CRC-based EDAC approaches are explained in this paper [1]. It shows promising results with minimum hardware requirements. It also proposes an algorithm to how to manage the 48-bit codeword when board space is scarce. The storage format and algorithm allow us to support this class of EDACs with minimum hardware support.

The research study on two of the most popular linear ECC codes, Hamming and ECC is shown this paper [2]. It explores the efficiency of their application in embedded systems. It shows the results that Hamming codes are simpler to implement and display better efficiency for smaller data lengths while Hsiao code are more complex but have better performance especially for larger data sizes.

In this paper [3], the efficient implementation of error correction code (ECC) processing circuits based on single error correction and double error detection SEC-DED code with check bit pre-computation is proposed for memories. When compared with a conventional implementation utilizing the odd-weight-column code, the implementation based on the proposed SEC-DED code with check bit pre-computation achieves reductions in the number of gates, latency, and power consumption of the ECC processing circuits.

In this paper [4], designs of novel joint crosstalk avoidance and triple-error-correction/quadruple-error-detection codes are proposed, and their performance is evaluated in different NOC fabrics. It is demonstrated that the proposed codes outperform other existing coding schemes in making NOC fabrics reliable and energy efficient, with lower latency.

The author in [5], demonstrated a new way of constructing a class of SEC-DED codes that uses the same number of check bits as the Hamming SEC-DED code but is superior in cost, performance, and reliability. This also minimizes the hardware, thus tending to lower failure rate of decoders.

The author in [6], explains the optimal generating algorithm for a matrix of equal weights columns and equal weights of rows. Then modifies some steps and using the modified algorithm for error correction, it is shown that it is efficient, fast and it s optimal in average cases.

In this paper [7], The Hsiao code is applied to cache memory using FPGA programmable Xilinx circuits. With this implementation the size of the syndrome generator is reduced and the cost of error correcting scheme is also reduced compared to the traditional Hamming code-based solution. This solution using a SEC-DED Hsiao code, increases reliability through fault tolerance, leading to low cost and low memory chip dimension, as this method solves the problem of faults by testing and correcting errors inside the chip.

This paper [8] presents Ultrafast codes, designed for very fast encoding and decoding operations. Ultrafast codes offer high-speed encoder and decoder circuits, and interesting error coverages. These codes can also be useful to protect high-speed memories or caches. Firstly, they have summarized Ultrafast SEC, SEC-DED and SEC-DAEC codes, presented in previous works. Then, new Ultrafast SEC-xAEC-DED codes have been introduced, describing the design methodology and the implementation details. The results confirm that Ultrafast codes achieve very low propagation delays, whereas adding very reasonable increments in the silicon area and the power consumption.

In this paper [9], a new SEC-DED-DAEC code is demonstrated to achieve high reliability protection against neutron induced soft errors in on-chip memory systems. It is showed that the proposed code has higher reliability than other SEC-DED-DAEC codes because it addresses the mis-correction problem. It is proved that the proposed code is suitable for a protection scheme against MCU in on-chip memory system.

The paper [10] proposes the investigation of four SEC-DED codes with different word lengths. The effect of the word length on error control capability is shown. Two different hardware implementations: XOR tree based and LUT based, for these four Hsiao codes are explored in this paper. The results show that the LUT structure outperforms the standard XOR structure in error correction speed in most cases.

Hsiao Code

Hsiao codes, also known as single-error-correcting and double-error-detecting (SEC-DED) codes, are a type error-correcting code used in memory systems. They are designed to detect and correct errors in data that are caused by single bit errors and detect but not correct double-bit errors. Hsiao codes are basically based on Hamming code, which is a linear error-correcting code that can detect and correct single-bit errors. However, unlike the Hamming code, Hsiao codes can also detect double-bit errors by using an additional parity bit.

In Hsiao codes, each data bit is represented by a binary number, and an additional parity is added to the code to detect errors. The number of parity bits added depends on the number of data bits in the code. For example, if there are n data bits, then log2(n+1) parity bits are added to the code.

When data is transmitted or stored using Hsiao codes, the receiver or memory system can detect and correct single-bit errors, but not double-bit errors. If a double-bit error is detected, the system will report an error and discard the data.

Overall, Hsiao codes provide a balance between error detection and correction, making them a popular choice for computer memory systems where errors can occur due to various reasons such as electromagnetic interference, cosmic rays, or manufacturing defects.

Cyclic Redundancy Code

Cyclic Redundancy Code (CRC) is an error-detection technique used in digital communication networks and storage systems. It is a type of checksum that is used to ensure the integrity of data transmitted over a channel or stored in memory.

CRC works by dividing the data into blocks of a fixed size and adding a checksum to each block. The checksum is computed by performing a polynomial division of the data with a fixed generator polynomial. The remainder of the division is the checksum, which is appended to the data block. When the data is received, the same polynomial division is performed on the received data, and the resulting remainder is compared to the checksum that was sent. If they match, the data is considered to be error-free.

CRC codes are widely used in communication systems such as Ethernet, Wi-Fi, and Bluetooth, as well as in storage systems such as hard disk drives and optical discs. They are simple and efficient to implement in hardware and software, and can detect most errors that occur during transmission or storage. However, they are not perfect and can still fail to detect some types of errors, such as those that occur in bursts.

Methodology

Firstly, we design EDAC that uses the Hsiao code’s encoder matrix. Then we design a second EDAC that uses a CRC encoder matrix. Finally, we interleave both the EDACs to improve the error detection and correction capabilities of our EDACs. We also design a storage format and algorithm design to manage and store the 48-bit codeword in 8-bit and 16-bit memory devices.

Design of Hsiao code’s encoder matrix

A C++ program is used to create Hsiao Code’s matrix generator, to generate the H-matrices compliant with the three Hsiao Code’s rules. Another C++ program is created, a code generator, to take the generated matrix as the input and produce the Verilog code for the EDAC’s encoder. The three Hsiao Code’s rules are:

1. Every column should have an odd number of 1’s; i.e., all column vectors are of odd-weights. In the following, we use n to indicate the number of rows in the H-matrix, and r to indicate the number of 1’s in each column, which must be an odd number. • use the combination formula C (n, r)=((n!)/ (r! (n-r)!)) to generate all possible odd-weighted

columns with n=8 and r=3. n=8 because the matrix has eight rows. r=3 because 3 is the smallest odd number next to 1, and 1 is already used in the check-bit columns. We therefore get a total of 56 combinations.

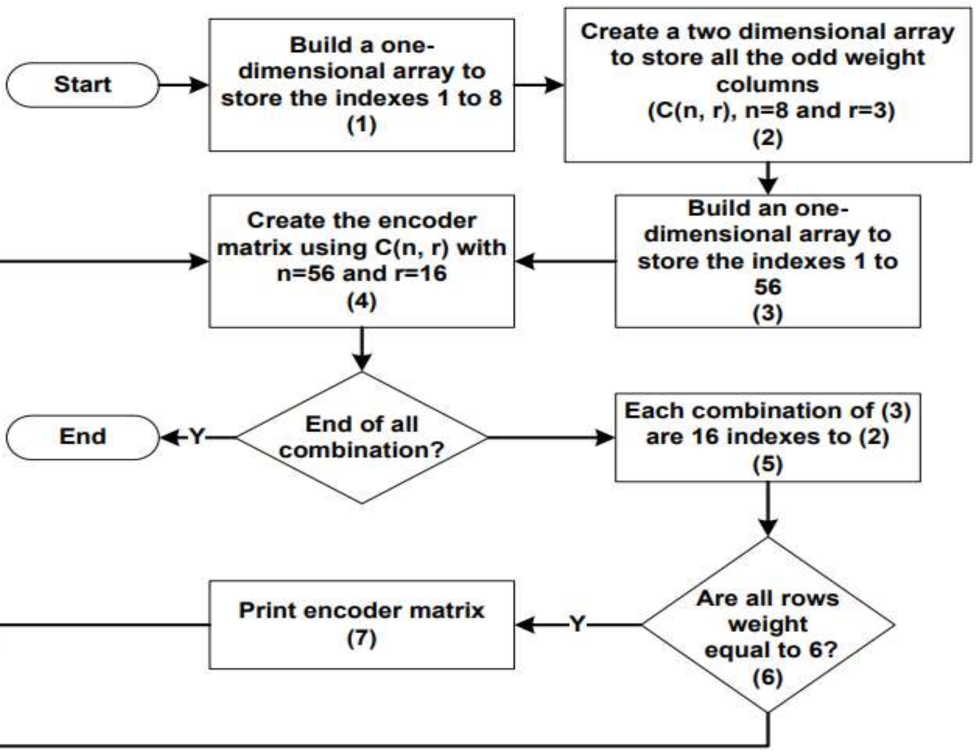
2. The total number of 1's in the H-matrix should be at a minimum.

   • We compute the 1's in the H-matrix as (16 * 3) + 8, or 56 1s' since the first 16 columns of the matrix contain three 1's, and the last eight columns are for check-bits.

3. The number of 1's in each row of the H-matrix should be made equal, or as close as possible, to the average number, i.e., the total number of 1's in the H-matrix divided by the number of rows.

   • We divide the 56 1s' by 8 and get 7 1s' per row. This number includes the check-bit in the last eight columns.

   Based on the rules above, we designed our matrix generator in C++. Figure shows the flowchart of our generator that works as follows:



Start
Build a one-dimensional array to store the indexes 1 to 8 (1)
Create a two dimensional array to store all the odd weight columns (C(n, r), n=8 and r=3) (2)
Build an one-dimensional array to store the indexes 1 to 56 (3)
Create the encoder matrix using C(n, r) with n=56 and r=16 (4)
End of all combination?
Y
End
Each combination of (3) are 16 indexes to (2) (5)
Are all rows weight equal to 6? (6)
Y
Print encoder matrix (7)

1. First, build a one-dimensional array to store the row indexes, 1 to 8, since the new matrix has eight rows. Each of the numbers refers to one of the eight rows.
2. A 56x3 array to store all the combinations of the row indexes in the column vectors that each contains a 1. Since each column vector in the H-matrix has three 1's, we choose the combinations of three out of the eight row indexes generated in step 1 (1 to 8) and store each combination in a column vector in this 56x3 array. There is a total of 56, or C (8, 3), such combinations stored in this array.
3. Build one-dimensional array to store the numbers 1 to 56. This is an array of the column indexes of the 56x3 array generated in step 2.

4. The column indexes to build the H-matrix as follows. Since the H-matrix has 16 columns, not including the eight columns used for the check-bit, and each column contains three 1's, we choose a combination of 16 out of the 56 column indexes generated in step 3 (1 to 56). If we have exhausted all the C(56, 16) combinations, then end the program; otherwise continue to step 5.
5. Use the 16 indexes generated in step 4 to index into the 56x3 array, and obtain a 16x3 sub-array. Use each of the 16 columns to build a 16x8 matrix. For example, if the first column is chosen, then we build a column vector in the new matrix containing eight rows, by storing 1's in rows 1, 2, and 3, and 0's in all remaining rows in this column vector.
6. If all the rows in the matrix generated in step 5 contain 6 1s' (not including the check-bit), go to step 7, otherwise go to step 4.
7. Print the encoder matrix, indicating that we successfully generated the matrix; then go to step 4 to check if we have exhausted all the C (56, 16) combinations.

Design of Cyclic Redundancy Code's encoder matrix

Cyclic Redundancy code is another type of error detection technique used to ensure the integrity of data transmitted over a channel or stored in memory. For networking, the interest is the Hamming Distance (HD), which is the least possible number of bit inversions in a message that can create an error undetectable by that message's CRC-based Frame Check Sequence.

We design the next EDAC based on a CRC polynomial to leverage the CRC detection capability. To increase the information rate, we compute the 16-bit message with an 8-bit CRC generator. To reduce the bit-weight in the encoder matrix, we set the CRC's seed value to zero. When a two-input XOR gate has an input equal to a logical zero, its output value is equal to the other input's value. Based on this Boolean state, we eliminate the XOR gates required for the seed value, thus reducing the bit-weight of our CRC matrix. We develop a CRC-based encoder matrix generator in C++ to generate our EDAC encoder matrix with these criteria.

Design of an interleaved EDAC

The EDAC uses two identical single error correction and double error detection (SEC-DED) EDACs configured as the even and odd encoders, syndromes, and decoders. For example, two Hsiao Code encoder matrices generated, or two CRC encoder matrices generated can be used as the two identical encoders. This configuration improves the error correction and detection rates. For example, if a double-bit-error has one error occurring on an even-bit, and the other on an odd-bit of a codeword, this error can be corrected.

Codeword storage format and algorithm

A hardware- or software-based EDAC provides methods to ensure data reliability in memory devices. The cost for more reliable data is lower information rate because an EDAC creates additional bits for error detection and correction. These extra bits are known as the check-bit. Depending on the EDAC implementation, the number of bits used for the check-bit varies. In our research, we use a 16-bit check-bit (or two 8-bit check-bits).

When the EDAC corrects an error, the fault-tolerant memory controller writes the corrected codeword (message and check-bit) back to the same memory location. This process is known as memory scrubbing. A 48-bit codeword would need six 8-bit, three 16-bit, or one 32-bit and one 16- bit memory device to store the message (code or data) and the check-bit. In a scenario where a satellite's board space is scarce, these memory devices' configurations are not desirable. Instead, we recommend using an 8-bit or a 16-bit memory device to store the 48-bit codewords.

The algorithm for storing the message and check-bit in an 8-bit or a 16-bit memory device works as follows:

1. The EDAC computes the 16-bit check-bit from the 32-bit message (data or code).

2. The memory controller writes the 32-bit message as follows:

• 8-bit memory device— writes the 32-bit message in four bytes.

• 16-bit memory device—writes the 32-bit message in two 16-bit words

3. After writing the fourth byte, the address is 0000_0003h.

4. The controller takes the last address (0000_0003h), shifts the address value to the right by one bit, and inverts the address. The address after inversion is the write address for the check-bit. In this case, the address is FFFF_FFFEh.

5. The memory controller writes the lower byte of the 16-bit check-bit to address FFFF_FFFEh and the higher byte to FFFF_FFFFh.

Conclusion

In this paper, we have presented a brief review of two error detection and correction (EDAC) methods, Hsiao code and Cyclic Redundancy code. The interleaving of these methods is discussed. The interleaving of these two methods can improve the error detection rates and error correction rates, is shown through this paper. This approach shows promising results with minimum hardware requirements. Also, managing the storage of 48-bit codeword when board space is scarce is discussed in this paper.

References

[1] Mong Tee Sim, Yanyan Zhuang "Design of Two Interleaved Error Detection and Corrections using Hsiao Code and CRC", 2020, IEEE.

[2] G. Tshagharyan, G. Harutyunyan, S. Shoukourian, Y. Zorian "Experimental study on Hamming and Hsiao Codes in the Context of Embedded Applications", 2017, IEEE.

[3] Sanguhn Cha and Hongil Yoon "Efficient Implementation of Single Error Correction and Double Error Detection Code with Check Bit Pre-computation for Memories", Journal of Digital Semiconductor Technology and Science, Vol.12, No.4, 2012.

[4] Amlan Ganguly, Partha Pratim Pande, and Benjamin Belzer, "Crosstalk-Aware Channel Coding Schemes for Energy Efficient and Reliable NOC Interconnects", 2009, IEEE.

[5] M.Y. Hsiao, "A Class of Optimal minimum Odd-Weight-Column SED-DED Codes", 1970

[6] Li Chen, “Hsiao-Code Check Matrices and Recursively Balanced Matrices”, (Chinese. English summary) [J] J. Nanjing Inst. Technol. 16, No.2, 33-39 (1986). [ISSN 0254-4180], 2013.

[7] NOVAC Ovidiu, SZTRIK Janos, VARI-KAKAS Stefan, KIM Che-Soong “Reliability Increasing Method Using a SEC-DED Hsiao Code to Cache Memories, Implemented with FPGA Circuits”, Journal of Computer Science and Control Systems, Vol.4, No.2, 2011.

[8] Luis-J. Saiz Adalid, Joaquin Gracia-Moran, Daniel Gil-Tomas, J Carlos Baraza-Calvo, and Pedro-J. Gil-Vicente, “Ultrafast Codes for Multiple Adjacent Error Correction and Double Error Detection”, 2019, IEEE.

[9] Ho-yoon Jun and Yong-surk Lee, “Single Error Correction, Double Error Detection and Double Adjacent Error Correction with No Mis-correction Code”, 2013, IEICE Electronics Express.

[10] Wei Gao, “A Study on Implementation of ECC for Embedded RAM”, Queen’s University, Kingston, Ontario, Canada, 2003.