Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 7, No. 4, Dec 2019, pp. 734~741
ISSN: 2089-3272, DOI: 10.11591/ijeei.v7i4.821

A Survey on the Best Choice for Modulus of Residue Code

Ahmad Towhidy[1], Reza Omidi[2], Karim Mohammadi[3]
[1,3] Electrical Engineering School, Iran University of Science and Technology (IUST), Tehran, Iran.
[2] Electrical Engineering Department, Faculty of Engineering, University of Zanjan, Iran

Article Info

Article history:

Received Nov 12, 2018
Revised Apr 30, 2019
Accepted Nov 8, 2019

Keyword:

Fault Detection
Residue Code
Modulus Choice
Self-Check-Adder
Fault Coverage

ABSTRACT

Nowadays, the development of technology and the growing need for dense and complex chips have led chip industries to increase their attention on the circuit testability. Also, using the electronic chips in certain industries, such as the space industry, makes the design of fault-tolerant circuits a challenging issue. Coding is one of the most suitable methods for error detection and correction. The residue code, as one of the best choices for error detection aims, is wildly used in large arithmetic circuits such as multiplier and also finds a wide range of applications in processors and digital filters. The modulus value in this technique directly effects on the area overhead parameter. A large area overhead is one of the most important disadvantages especially for testing the small circuits. The purpose of this paper is to study and investigate the best choice for residue code check base that is used for simple and small circuits such as a simple ripple carry adder. The performances are evaluated by applying stuck-at-faults and transition-faults by simulators. The efficiency is defined based on fault coverage and normalized area overhead. The results show that the modulus 3 with 95% efficiency provided the best result. Residue code with this modulus for checking a ripple carry adder, in comparison with a duplex circuit, 30% improves the efficiency.

Corresponding Author:

Reza Omidi,
Electrical Engineering Department, Faculty of Engineering,
University of Zanjan,
Zanjan 4537138791, Iran
Email: rezaomidi@znu.ac.ir

1. INTRODUCTION

Nowadays the soft errors are the main threat for reliability in the special propose designs such as military and space applications. In these fields, the fault-tolerant design is set to become a vital factor for electronic systems. Since the 1950s, failures of digital circuits have been reported from space tests[1]. For many years, it was considered to be a monitoring error but since the mid-1970s the cause of this phenomena has been reported by researchers. In the 1980s, the fault masking methods have become a central issue in the design of electronic systems[2]. In the past, this issue was an insignificant problem but nowadays the procedure of technology developments such as System-on-Chip architectures, transistor down-scaling, and decrease of the supply voltage, increase the system sensitivity against the soft errors[3, 4]. As an initial hardening technique, the packing and shielding are used for the reduction of the environment effects but these methods are ineffective in some cases like in aerospace because high energetic particles can penetrate through the shield packages and causes the unwanted single event effects (SEE) [5]. As mentioned, by technology developments, the error of digital circuits became more important related to the past [6]. Although, the new methods are used for energy reduction especially in microprocessors not only do not seem to significantly impact SEE rate but also exaggerate the problem [7]. There are several solutions for fault detection in logic and arithmetic circuits. In [8] a healing mechanism has been investigated for retaining the true output in presence of stuck at fault in

interconnect levels of combinational circuit. Meanwhile, in [9] reported a prototype processor with a residue checker that can detect up to 98.0% and 98.8% of the unmasked transient and permanent errors respectively.

Experimental analysis on microprocessor units in Ref. [10] shows that the arithmetic and logic unit (ALU) is the most sensitive unit in the microprocessor against the SEE effects. An ALU consist of the arithmetic and logic operations so that the arithmetic operations are the most complex operations in ALU, so it is necessary to pay more attention to the hardening techniques used for this issue. Coding methods are the most investigated methods to detect a fault in arithmetic units.

There are a lot of codes for error detection such as Parity code, Hamming code, Cyclic Redundancy Check (CRC) code, AN code, Berger code, and Residue code. In the past decade, various techniques based on residue code have been used and these codes are one of the well-known schemes for fault-tolerant arithmetic. A key problem with much of the literature regarding RNS based checker is that these methods impose an unacceptable area penalty. It is obvious that the modulus choice directly affects on overhead and delay. As we know the majority of prior researches in the residue numbers are carried with modulo 3. A residue code with A = 3 requires only two extra bits and is the least expensive of the low-cost residue codes able to detect all single errors. Area overhead of residue generators makes it unsuitable for testing small circuits, in [11] a new implementation method based on Current Mode Multi-Valued Logic CMMVL has been proposed that can be used for different values of check base.

In this paper, we aim to investigate the best choice for modulus value. This paper provides an extensive analysis to study the modulus effect in the area and fault coverage factor. Although mod 3 residue checker is a straightforward way for checking arithmetic circuits but residue checkers with bigger moduli have their attractive properties such as their ability in the detection of multiple faults and their better fault coverage. So the selection of a suitable check base for residue checker circuit is very important but we can not find a comprehensive comparison between different check base values. So in this article, we compare the efficiency of residue code with different check bases by two parameters the first parameter is area overhead of circuit and the second one is fault coverage. For validation, a ripple-carry adder is used as Circuit Under Test (CUT) then fault is injected to the circuit to achieve fault coverage for different test circuits and finally efficiency is derived by division of fault coverage by area overhead.

The remainder of this paper is organized as follows. Section 2 briefly summarizes the error detection codes. Section 3 is mainly focusing on the residue codes operation and its challenges in practical works. In section 4, the setup structure and results are provided. This section provides a figure of merit to compare efficiency and discuss the results. Conclusions are drawn in Section 5.

2. ERROR DETECTION CODES

The encoding algorithms are based on computing the check bits from input data and then compare them with output data check bits by some check bits that are smaller than input data [12]. Different coding methods are introduced and studied in various literature. It is better to mention that just some of these coding approaches support Arithmetic operations, as the main concern of this paper. Some of the most important methods are listed in the following.

One type of the most popular codes are Error-Correcting Codes (ECCs) that are developed from Hamming codes [13]. In the 1960s IBM used Error correction coding (ECC) in computers. In the 1980's ECC codes prospered for masking the occurrence of SER faults in SRAMs [2]. Mostly Error Correction Codes (ECC) are based on Hamming and Hsiao codes that can correct single-bit and detect double-bit-errors and for increasing reliability Double-bit Error Correction Triple-bit Error Detection codes (DEC-TED) are introduced [14]. Cyclic Redundancy Check (CRC) codes are used for Error detection. The simplicity of CRC codes has already caused it to be a friendly code for error detection applications. CRC codes are used in data transmission. When input data polynomial is divided by CRC polynomial, that is a fixed polynomial, the reminder coefficients will be CRC bits. CRC code adds parity bits to data and sends them to channel. In the receiver, received data is divided by the CRC polynomial again. If computed CRC code is not equal to the received CRC bits, an error is occurred in the channel during the transmission of data [15]. Another code for error detection is AN code. AN code is a non-separate and non-systematic code so data and codeword are processed together and are unlike to residue code, data cannot be extracted from code word directly. AN-encoder is a multiplier that multiplies it's input data with a constant A that is suitable for error detection (the values of A are discussed in [16]).

$$Xc = A * x \quad , \quad 1 < A. \tag{1}$$

In AN coding a data is true if and only if code word be multiples of A and other situations are faulty. So, for checking data modulus with A is computed if it is zero, code word is valid. (Xc mod A = 0.) Finally, to extract the data from code word an integer division is needed. x = xc/A [16]. Berger code is the least redundant systematic code that can detect multi and single bit unidirectional errors [17]. Berger codes use the number of 0's (or the number of 1's) of data as check symbols (Xc, Yc, Sc) [18]. For a given Arithmetic

operation S = X op Y, Sc can be calculated by a function of X, Y, Xc, and Yc , Sc = F(X, Y, Xc, Yc) So if the calculated Sc is equal to the check bits that are generated by S, code word is fault free [19]. Parity prediction code calculates the output's parity from input operand's parity and parity of internal carry. One drawback of parity code is that if an error causes several bits of output operand to be changed it is possible that the output operand's parity stays unchanged, so in this situation parity code is unable to detect the error. In [20] a circuit design has been proposed that resolves this problem. Another drawback of parity prediction is that the area overhead of parity prediction is proportional to the square of its operand's [20]. Residue code is a separable code that is suitable for digital circuits because it covers arithmetic and logic operations [21]. The separability of residue code enables designers to use residue code just for one part of the circuit also extraction of data from codeword can be done without any decoder. Residue code is suitable for operations such as add, subtract, etc. also residue code can be extended for logic operation although it is not as straightforward as using residue code for arithmetic operations [22].

In table1 that is derived from Ref. [14], a summary of explained error coding techniques are presented. As shown in the table1 ECC and CRC codes are suitable for data error detection applications such as memories and are not a good choice for error detection in logic and Arithmetic circuits. Although AN code is a suitable code for error detection in arithmetic operations, it is not a concurrent error detection code and it can't be used as a separable code. Berger code and residue codes are suitable for error detection in arithmetic operations but Berger codes don't cover some arithmetic operations such as multiply or division. This paper shows the history of Residue codes and several simulations that are performed to find out the efficient check base for residue code. Simulations for Fault coverage are performed by Tetra-Max program.

Table 1. Summary of error coding methods (derived from ref. [14])

Codes	Separable	Domain	HW Cost	Concurrent Error Detection	Supported Operators
ECC	Yes	Data	Low- medium	Yes	--
CRC	Yes	Data	Very low	Yes	--
AN code	No	Logic & Data	Low	No	(+), (-), (/)
Berger code	Yes	Logic & Data	Low	Yes	(+), (-), (logic op)
Parity Prediction	Yes	Logic & Data	Low	No	(+), (-), (/), (*)
Residue code	Yes	Logic & Data	Low	Yes	(+), (-), (/), (*), (SQRT)

3. RESIDUE CODE OVERVIEW AND APPLICATION

The residue code as one of the most attractive techniques for arithmetic operation is categorized in self-checking designs that can concurrently detect errors in design. Residue generators are The most expensive part of the residue check circuit. Residue generators are used in many different applications for example residue generators are the major unit of Residue-Number-Systems (RNS). RNS is used in many application areas involving digital filters and cryptography [23, 24]. In 1958 Peterson performs a study about self-checking adders. He designed a circuit to handle overflow bit in self-check adders [25]. Basically, the residue codes are suitable for Arithmetic circuits but in [22] a method has been suggested for extension of residue code for logic operations (such as XOR, OR, etc.), although it is not a straightforward method in comparison with using residue code for arithmetic operations, because the nature of residue code is arithmetic [22]. ALU consists of arithmetic and logic operations, in [21] an architecture for fault-tolerant ALU has been suggested that it uses residue code for arithmetic operations and it uses duplex circuits for the logic units because, for logic circuits, duplication is more efficient than residue code [21]. As mentioned residue code is unsuitable for all applications because of its hardware complexity. Low [26], proposed a new Architecture for residue generators that their modulus is not in well-known types such as or . In [26], it's mentioned that residue generation for an arbitrary modulus is a complex arithmetic operation and its complexity is related to the word length ratio of data to modulus [26]. according to the equation and (that r , i are not negative integers) residue codes by check bases in form of are generated parallel by using a tree of full adders [27, 28], so it causes residue generator to work in higher speed with no extra hardware overhead [28]. For the implementation of residue generators by this type of modulus, the n-bit data input (w) is partitioned to a-bit parts from the least significant bits, then the residue is calculated by modulo addition of a-bit parts. So it needs 14 FAs for a 16-bit mod 3 residue generator [29]. In [29] it has been mentioned that the complexity of residue check system is increased by increasing the check base-A [29]. Most of the residue checkers by odd check bases can detect single faults in faulty data because the difference of faulty data and fault free data is equal by and is not divisible by odd numbers [29]. In [30] two circuit diagrams are suggested for implementation of mod 3 residue generators. The first implementation is based on tree of full adders by end-around-carry technique and the second

implementation is based on tree of 4-bit residue generators. The result of comparison for these circuits has shown that although the area of the second circuit is 1.7 times more than full-adder based implementation and its delay is lower.

In [31] an architecture has been used for fault detection in fast adders and it has been named as "long residue checking" (LRC), this architecture can be used as a separable checking circuit. The most important advantage of LRC in spite of other residue checkers is its ability in checking Residue codes by large checking moduli because by Increasing checking modulus of residue code, error coverage of residue code will increase [31]. So the best modulus selection for Long Residue Checker is the maximum modulus width because maximum modulus causes maximum error coverage, also power consumption and latency is lower [31]. In other methods residue checking for adder was based on equation but in LRC method this equation is changed to another equation that is based on subtraction, ( ) so the main idea of LRC is the cancelation of sum and carry by their own. Implementation of the cancelation check circuit is as simple as conventional circuits for large modulus and doesn't need carry propagation [31].

Langdon and Tang [32], compared residue code and parity prediction concurrent error detection methods for group look-ahead adder. Their comparison showed that area overhead of residue mode 3 check adder prediction is not high in comparison with parity if and if and only if the check bits of residue code be provided by input operands [32]. Gaddess [33] proposed a sequential hardware shared architecture that checks the adder by itself. in this circuit result of addition is done in one sequence and result checking is done in several sequences after that [33]. In [34] several methods for concurrent error detection are investigated and their ability in detection of multiple faults is compared. Also, it is mentioned that logic area overhead of Berger-code and Bose-Lin code is very high [34]. One advantage of Duplex circuits in comparison with Berger and Bose-Lin codes is its ability in the detection of multiple faults but the area overhead of parity prediction method is lower than duplication [34]. Area overhead of residue code by check base b = 3 or 7 is acceptable for circuits such as multipliers and adders but its area overhead for logic circuits is very high [34]. So the drawback of residue code is it's area overhead, especially for small circuits.

4. THE MODULUS EVALUATION AND RESULTS

For simulation of Residue code, a VHDL code for ripple carry adder as a functional unit and also a VHDL code for residue codes are implemented. Also, it is assumed that residue check bits of input operands are presented by input Data. So, we just implement one Residue generator for data output of Adder. We use a ripple carry adder (simplest adder) to evaluate the capability of residue codes for small circuits compared with DMR circuit.

4.1. Area overhead of checking codes

DMR technique is not suitable for complex circuits because duplication of a big circuit needs unacceptable Area overhead. But for a small circuit, it can be used as a useful technique. So, for complex circuits, another technique such as coding is used and, in this article, we used residue code. A block diagram of residue checking circuit has been shown in figure 1. As shown in figure1 For checking functional unit three units (Residue checker, residue calculator, and comparator) are needed. Also for DMR circuit two functional units and one comparator are needed. Area overhead of the Residue calculator unit relates to the number of bits that are needed for representation of residue code.

In figure2 Area overhead for units of residue check circuit for different values of modulus are reported. Note that for better comparison area overhead of Residue checker units are normalized by area overhead of functional unit. As shown in figure 2 Residue generator area overhead for moduli of the forms 2^n is approximately zero, because, for this form of modulus, the residue can be obtained by separation of n bits of LSB and doesn't need any hardware for computation of residue. Also residue calculator by this type of modulus, and adder as a functional unit is an n-bit adder so we name this type of modulus as Partial Duplex.

Residue code by modulus of the forms $2^n - 1$ related to other modulus (expected 2^n) has lower area overhead. For residue codes by even moduli by a form of $(2^n \pm 1).2^k$ for example 6, 10, 12, and 14, k-bits of residue that are the least significant bits are calculated by methods similar to Partial Duplex and other bits of residue are calculated by a modulo $2^n \pm 1$ residue generator. So area overhead is near to circuits that use modulo $2^n \pm 1$ residue generator, but in this type of modulus, the residue is a (k+n)-bit word so the area overhead of residue calculator and comparator for this modulus is more thanmoduli. Also as shown in figure2(a) $2^n \pm 1$ for small circuits such as 8 bit Adder and for some moduli such as 9, 10, 11, and 13, residue code is not a good choice in comparison with DMR circuits, because we use a simple circuit (Ripple Adder) for functional unit and for simple circuits area overhead of residue code is more than DMR but for complex circuits residue code is better than DMR (for example 64 bit adder). (modulo 11 and 13 residue generators are designed by methods that are suggested by low [26].)

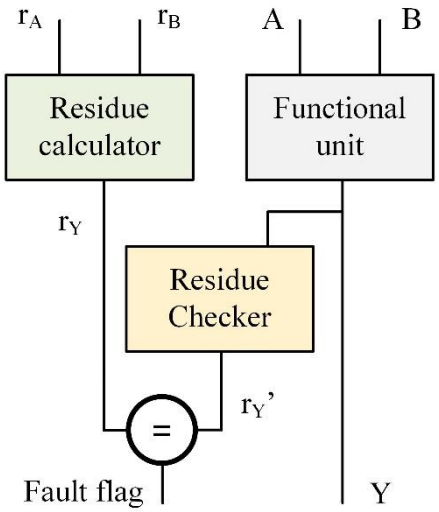

Figure 1. block diagram of residue checking circuit

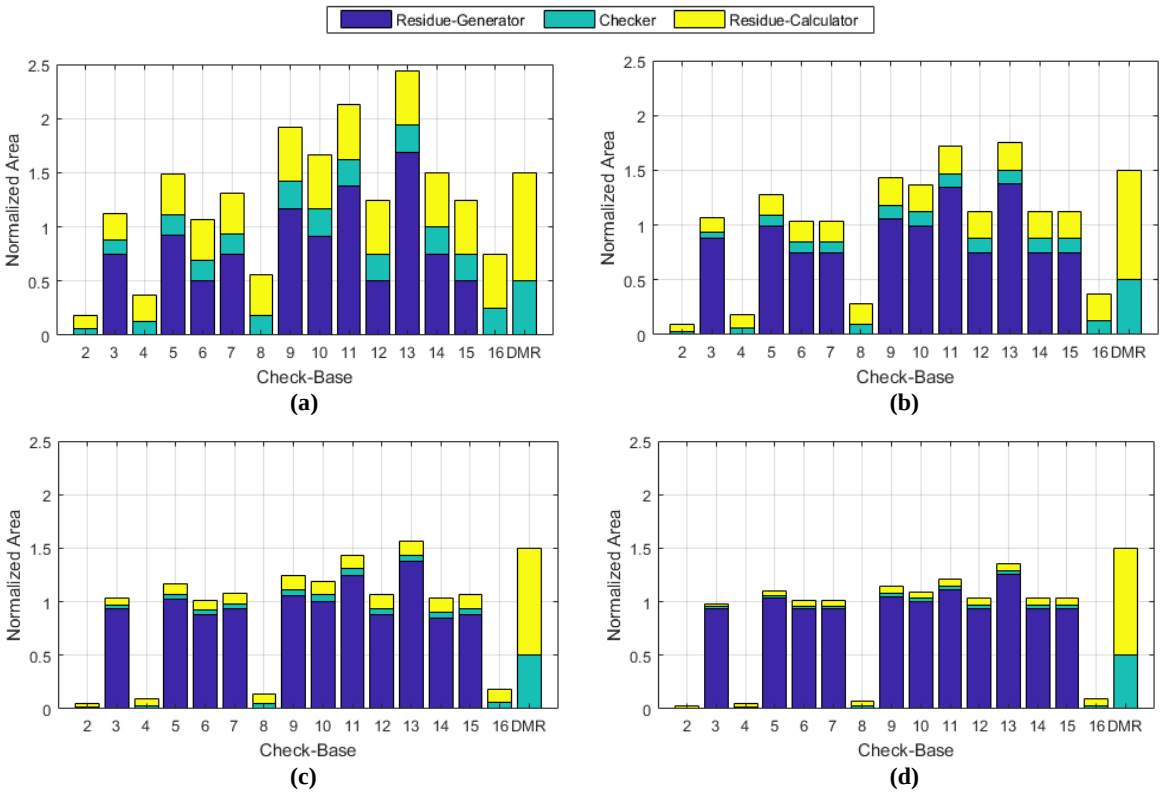

Figure 2. residue checker area overhead for (a) 8-bit, (b) 16-bit, (c) 32-bit, and (d) 64-bit Adder.

4.2. Fault Coverage

Fault coverage is an important meter for specification of digital circuits. The fault coverage factor is defined as the percentage of detected faults out of all faults [35, 36].

$$Fault\_coverage = \frac{\text{Number of detected faults}}{Total\ number\ of\ faults} \quad (2)$$

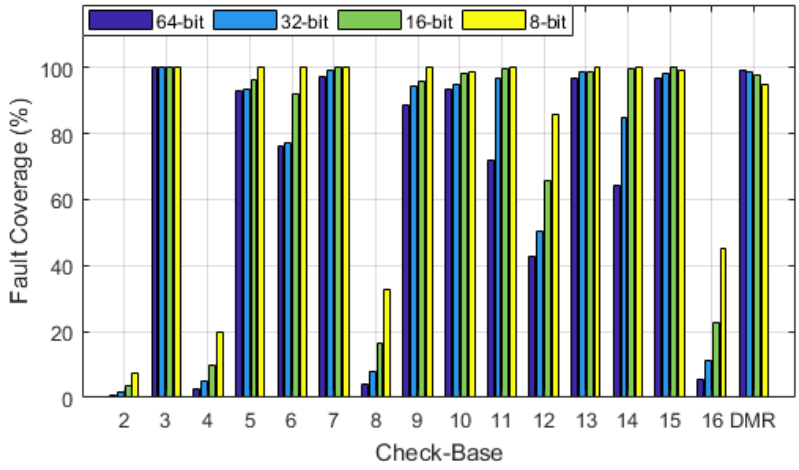

Figure 3. Fault coverage of adder circuit for different values of check base and data width

For validation of the test circuit, we check the output of residue code checker (Fault Flag) by Tetramax program to obtain fault coverage for under test unit. The total number of stuck at faults that are reported by Tetramax, for 64, 32, 16, and 8-bit adders, respectively, are 1280, 640, 320, and 160. The values of fault coverage for different values of modulus are shown in Figure 3. As shown in this figure, for "Partial duplex" residue codes, fault coverage is related to the number of residue check bits. In other words, for moduli of the forms 2^n the amount of fault coverage is depended on the ratio of duplex part to the total of the functional unit. Also, partial duplex circuits have the least values of fault coverage in comparison with other moduli.

4.3. Efficiency validation

Area overhead of residue code is related to modulus, so for comparison of residue codes, an efficiency function is needed. In this article to achieve the best modulus for residue code, we defined a function for efficiency as bellow.

$$Efficiency = \frac{Fault \ \ coverage}{Normalized \ \ area \ \ overhead} \tag{3}$$

According to equation 3, the efficiency of residue codes is calculated and is shown in the figure 4. As shown in figure the efficiency of Partial Duplex circuits is limited and the best form of Dualism is full duplex or DMR circuit. In Partial Duplex circuits, low fault coverage causes low efficiency and for the full Duplex circuit, high area overhead is the problem of efficiency. Also, this problem is worse in large circuits.

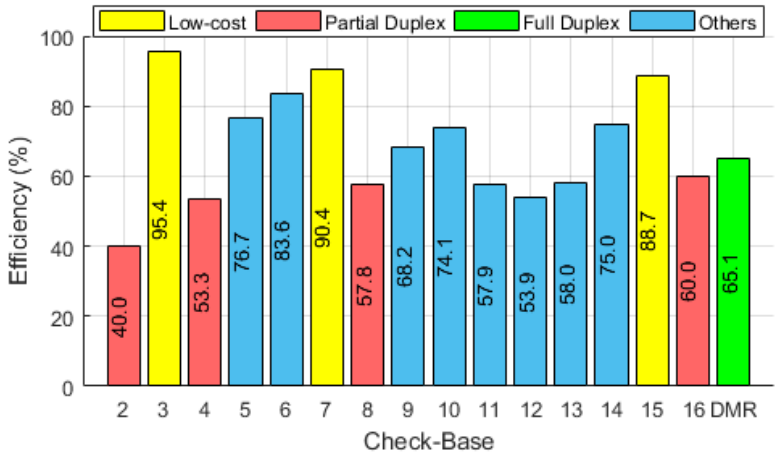

Figure 4. Efficiency of residue code for different values of check base and data width

A simple comparison shows that implementation of residue checkers by moduli in form of $2^n - 1$ (Low Cost Residue code [28]) is the more efficient among other residue codes and DMR circuit, it is because of lower area overhead and higher fault coverage of these circuits. Modulo 11, 12, and 13 residue codes for

simple circuits are not a good choice compared to DMR although for complex circuits (functional unit) Residue code will be better than DMR.

5. CONCLUSION

In this paper, we present a comparison between residue code with different moduli and DMR fault detection methods. Residue codes are suitable for increasing the reliability of arithmetic circuits. The purpose of this paper is to study the best modulus for residue code to obtain the best Fault coverage with less Area overhead, with an aim to the validation of Fault coverage by Tetramax program. In this paper, we performed the simulations of Residue codes for different moduli for a simple ripple carry adder. According to results, area overhead of residue code for unusual check bases such as 11, 12, and 13 is not acceptable in comparison with duplex circuits. Also, the area overhead of residue generators with check bases in the form of 2^n is slight but their fault coverage is not suitable. Mod 3 residue code is the best selection for check base although other check bases in the form of $2^n \pm 1$ are efficient. Some check bases are composed of two types $(2^n \pm 1).2^k$ of check base. Although their efficiency is near to low-cost codes, but their check word length is some bigger and it caused unacceptable increases in the residue calculator unit in complex circuits.

REFERENCES

[1] F. Wang, and V. D. Agrawal, "Single Event Upset: An Embedded Tutorial." pp. 429-434.
[2] W. Heidergott, "SEU tolerant device, circuit and processor design." pp. 5-10.
[3] B. Narasimham, B. L. Bhuva, R. D. Schrimpf, L. W. Massengill, M. J. Gadlage, O. A. Amusan, W. T. Holman, A. F. Witulski, W. H. Robinson, J. D. Black, J. M. Benedetto, and P. H. Eaton, "Characterization of Digital Single Event Transient Pulse-Widths in 130-nm and 90-nm CMOS Technologies," IEEE Transactions on Nuclear Science, vol. 54, no. 6, pp. 2506-2511, 2007.
[4] M. Santarini, "Cosmic radiation comes to ASIC and SOC design-As 1C-process geometries shrink, single-event effects, such as soft errors and latch-ups, will soon become primary concerns for designers of ASICs and," Edn, vol. 50, no. 10, pp. 46-60, 2005.
[5] R. Omidi, and H. Zarrabi, "New Protection Technique Against Unidirectional MEUs for FIR Filters," Circuits, Systems, and Signal Processing, vol. 37, no. 1, pp. 367-382, January 01, 2018.
[6] N. N. Mahatme, N. J. Gaspard, T. Assis, S. Jagannathan, I. Chatterjee, T. D. Loveless, B. L. Bhuva, L. W. Massengill, S. J. Wen, and R. Wong, "Impact of technology scaling on the combinational logic soft error rate." pp. 5F.2.1-5F.2.6.
[7] A. Dixit, and A. Wood, "The impact of new technology on soft error rates." pp. 5B.4.1-5B.4.7.
[8] S. Meyyappan, and V. Alamelumangai, "Black Box Model based Self Healing Solution for Stuck at Faults in Digital Circuits," International Journal of Electrical and Computer Engineering (IJECE), vol. 7, no. 5, pp. 2451-2458, 2017.
[9] K. A. Campbell, P. Vissa, D. Z. Pan, and D. Chen, "High-level synthesis of error detecting cores through low-cost modulo-3 shadow datapaths." pp. 1-6.
[10] P. Duba, and R. K. Lyer, "Transient fault behavior in a microprocessor-A case study." pp. 272-276.
[11] A. Towhidy, R. Omidi, and K. Momammadi, "An Efficient Current Mode MVL Residue Code Checker for Fault Tolerant Arithmetic," Journal of Circuits, Systems, and Computers, 2019.
[12] E. Ossi, D. Limbrick, W. Robinson, and B. Bhuva, "Soft-error mitigation at the architecture-level using berger codes and instruction repetition."
[13] I.-R. Chen, and I. L. Yen, "Analysis of probabilistic error checking procedures on storage systems," The Computer Journal, vol. 38, no. 5, pp. 348-354, 1995.
[14] J. S. Carretero Casado, "Low-cost and efficient fault detection and diagnosis schemes for modern cores," 2015.
[15] D. V. Sarwate, "Computation of cyclic redundancy checks via table look-up," Communications of the ACM, vol. 31, no. 8, pp. 1008-1014, 1988.
[16] U. Schiffel, "Hardware error detection using AN-codes," 2010.
[17] D. Pierce, and P. K. Lala, "Modular implementation of efficient self-checking checkers for the Berger code," Journal of Electronic Testing, vol. 9, no. 3, pp. 279-294, 1996.
[18] J. Lo, S. Thanawastien, and T. R. N. Rao, "Berger check prediction for array multipliers and array dividers," IEEE Transactions on Computers, vol. 42, no. 7, pp. 892-896, 1993.
[19] J. Lo, S. Thanawastien, T. R. N. Rao, and M. Nicolaidis, "An SFS Berger check prediction ALU and its application to self-checking processor designs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 11, no. 4, pp. 525-540, 1992.
[20] M. Nicolaidis, R. O. Duarte, S. Manich, and J. Figueras, "Fault-secure parity prediction arithmetic operators," IEEE Design & Test of Computers, vol. 14, no. 2, pp. 60-71, 1997.
[21] V. S. Veeravalli, "Fault tolerance for arithmetic and logic unit." pp. 329-334.
[22] I. L. Sayers, and D. J. Kinniment, "Low-cost residue codes and their application to self-checking VLSI systems," IEE Proceedings E - Computers and Digital Techniques, vol. 132, no. 4, pp. 197-202, 1985.
[23] H. K. Garg, and H. Xiao, "New Residue Arithmetic Based Barrett Algorithms: Modular Integer Computations," IEEE Access, vol. 4, pp. 4882-4890, 2016.
[24] P. A. Mohan, Residue number systems: algorithms and architectures: Springer Science & Business Media, 2012.

[25] W. W. Peterson, “On Checking an Adder,” IBM Journal of Research and Development, vol. 2, no. 2, pp. 166-168, 1958.
[26] J. Y. S. Low, and C. Chang, “A New Approach to the Design of Efficient Residue Generators for Arbitrary Moduli,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 60, no. 9, pp. 2366-2374, 2013.
[27] A. Avižienis, A set of algorithms for a diagnosable arithmetic unit: Jet Propulsion Laboratory, California Institute of Technology, 1964.
[28] S. J. Piestrak, "Self-testing checkers for arithmetic codes with any check base A." pp. 162-167.
[29] S. J. Piestrak, and P. Patronik, "Design of Fault-Secure Transposed FIR Filters Protected Using Residue Codes." pp. 575-582.
[30] S. J. Piestrak, F. Pedron, and O. Sentieys, "VLSI implementation and complexity comparison of residue generators modulo 3." pp. 1-4.
[31] M. B. Sullivan, and J. E. E. Swartzlander, "Long Residue Checking for Adders." pp. 177-180.
[32] G. G. Langdon, and C. K. Tang, “Concurrent Error Detection for Group Look-ahead Binary Adders,” IBM Journal of Research and Development, vol. 14, no. 5, pp. 563-573, 1970.
[33] T. G. Gaddess, “An Error-Detecting Binary Adder: A Hardware-Shared Implementation,” IEEE Transactions on Computers, vol. C-19, no. 1, pp. 34-38, 1970.
[34] S. Mitra, and E. J. McCluskey, "Which concurrent error detection scheme to choose ?." pp. 985-994.
[35] Synopsys, "TetraMAX ATPG user guide," 2013.
[36] L.-T. Wang, C.-W. Wu, and X. Wen, VLSI test principles and architectures: design for testability: Elsevier, 2006.