This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

Multiple-bit Parity-based Concurrent Fault Detection Architecture for Parallel CRC Computation

Dipanwita Gangopadhyay and Arash Reyhani-Masoleh

***Abstract*—As a result of huge advancements in VLSI technology, more and more complex circuits are being implemented making not only the whole digital system more prone to faults, but also the fault detector itself susceptible to faults resulting in the requirement of concurrent fault detection architecture of the encoders and decoders. In this paper, we present a multiple-bit parity-based fault detection architecture for parallel CRC computation. After analyzing the parallel implementation of CRC, we present a formulation to generate a multiple-bit parity prediction structure to incorporate the fault detection architecture. Using the formulations of digit level CRC architecture, the checksum is divided into few blocks and predicted multiple-bit parity of the blocks are compared with the actual parity bits. Finally, with the help of software simulation and ASIC implementation, we show that the proposed scheme is highly efficient in terms of fault detection capability whereas it involves small area and time overhead. As an example, we have shown that the worst case area overhead is 25.7% for CRC-32 with four parity bits, and corresponding time overhead is 15.6%.**

***Index Terms*—CRC, parity prediction, fault detection, multiple-bit**

I. Introduction

THE recent advancements in wireless technology have brought forth very high speed transmission which faces a major challenge from noisy channels. Factors such as signal attenuation, multiple access interference, inter symbol interference and Doppler shift degrade the quality of data to a great extent. As a result, the received signal has a chance of getting corrupted. Consequently, for the alleviation of this problem, error control coding is not only desirable, but has become a must to achieve an acceptable Bit Error Rate (BER). For the purpose of checking the integrity of the data being stored or sent over noisy channels, the most widely adopted among all the error control codes is Cyclic Redundancy Check Code (CRC). CRCs, first introduced by Peterson and Brown in 1961 [1], are used for the detection of any corruption of the digital signal during its production, transmission, processing as well as storage stage. Recent wireless technologies namely, Asynchronous Transfer Mode (ATM) [2], IEEE communication standards, such as wired Ethernet (IEEE 802.3) [3], Wi-Fi (IEEE 802.11) [4] and WiMAX (IEEE 802.16) [5], employ CRC for error detection purpose.

A very well written understanding of the CRC computation and collection of most of the published hardware and software implementations of CRC can be found in [6], [7] and [8]. Various parallel hardware structures have been described in [9], [10], [11], [12], while [13] talks about cascading. Several literature have proposed highly efficient architecture based on performance enhancement [14], [17] as well as resource utilization [18]. The list also includes Two-Step [19], Cascade [13], Look-Ahead [12], State-Space Transformed [20], and Retimed Architectures [14], [25]. The authors of [11] have proposed a three-step LFSR architecture which makes it easy to address the two major issues of large fanout limitation and iteration bound limitation in parallel LFSR architecture which is an integral part of CRC computation. In [15], the authors have proposed an improved solution for the fanout problem. The authors of [15] have shown that the proposed architecture achieves significant improvement in terms of processing speed and hardware efficiency. In [16], the authors have extended the above idea and presented a mathematical proof showing that a transformation exists in state space that can reduce the complexity of the parallel LFSR feedback loop along with a new idea for high speed parallel implementation of linear feedback shift registers based upon parallel IIR filter. In [32], the authors have proposed a novel parallel implementation of long BCH encoders to achieve significant speedup by eliminating the effect of large fanout. The authors have mentioned that similar technique can also be used for CRC.

Rapid advancement of semiconductor technologies results in rapidly increasing soft error rates [26], [27], [28] due to shrink in area, time and power. Temporary faults caused by cosmic radiation, known as single event upset (SEU) [29], are the main sources of errors. A lot of attention has been paid for the reliability improvement of all the elementary components reflected by their much lower failure rates [30]. But the large number and dense population of these elements on the chip degrade the reliability to a large extent. While error detection codes and error correction codes have long been used for checking errors, the reliability of the error correction or detection encoder itself has come up as a major issue. High speed data transmission requires high speed error detection and parallel processing at the cost of increased hardware, which increases the probability of error occurrence of internal faults. As a result, fault detction of encoder and decoder have gained a lot of importance.

In [31], the authors have addressed the problem of designing parallel fault-secure encoders by generating not only error correcting check bits, but also independently and in parallel, error detecting check bits. The next step involves the comparison of error detection check bits with another set of error detecting check bits generated from error correction check bits. This design is significantly cost effective compared to duplication with comparison technique. There are several other literature which deal with the similar topic. In [34], the authors have

The authors are with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON, Canada N6A 5B9, (e-mail: dgangop@uwo.ca; areyhani@uwo.ca).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

discussed the design of a fault secure encoder in between a fault secure RAM and a faulty channel. Error detection and error correction using Residue Number System have been discussed in [35]. In [36], we find the description of fault detection of Reed Solomon encoder, while [37] covers both Reed Solomon encoders and decoders. In both [36] and [37], the area overhead between the self checking implementation and without the self checking capability is mentioned as 50%.

In [23], the authors have presented a new approach for the automated synthesis of multilevel circuits with concurrent error detection using a parity-check code. They have developed a new procedure namely structure-constrained logic optimization and proved that this design along with a checker forms a self-checking circuit. In [24], the authors have addressed the issue of developing reliability driven logic synthesis algorithms by proposing three schemes. The first scheme is duplication and comparison, the second being utilizing Berger code and the third scheme involves partitioning the outputs of the circuit and concurrent error detection based on parity code. In this third scheme, the authors have partitioned the circuit into groups and generated the parity signals in such a way that only one output signal from each group influences a parity signal. They have considered non-overlapping multilevel circuits for the synthesis of outputs in each partition and the parity signals. Our proposed method of the multiple bit parity based concurrent fault detection scheme for parallel CRC computation inherently exploits the non-overlapping feature of the parity generation circuits.

In this paper, we have presented a design concept of a multiple-bit parity-based concurrent fault detection for parallel CRC architecture. In the proposed scheme, the generated CRC output is divided into a few blocks based on the number of parity bits and has been made to go through a multiple-bit parity comparison unit. Then, the actual parity bits are compared with the predicted multiple-bit parity of the blocks to incorporate the fault detection architecture. We note that to the best of our knowledge, this paper is the first study specific to parallel CRC computation to generate concurrent fault detection using multiple-bit parity. The contribution of this paper is summarized as follows:

- First, we have derived the formulation for the multiple-bit parity-based fault detection of parallel CRC architecture for the case $l \geq m$ when l is the number of input message bits processed in each iteration and m is the degree of the CRC generator polynomial which defines the underlying CRC code. Next, after extending the matrix based formulation of parallel CRC structure for $l < m$, we have proposed the concurrent fault detection formulation for this case too.
- Having derived all the required formulations, we have presented the hardware architecture based on the proposed formulation for both the above cases.
- We have discussed a thorough theoretical error and fault analysis using detailed error and fault models along with the investigation of the influence of the generator polynomial on the error propagation. We have also formulated the error detection capability of the proposed scheme covering all the possible $2^m - 1$ errors. Furthermore, we have done several fault detection simulation using C codes for various CRC polynomials and different values of the number of parity bits. The simulation results presented in the tabular form confirm the significantly high fault detection capability of the proposed scheme.
- Finally, we have presented a theoretical complexity analysis showing the low required overhead of the proposed scheme. We have done ASIC implementation using Verilog code to confirm these small values of the theoretical complexity overheads.

The remainder of this paper is organized as follows. First in Section II, preliminary ideas regarding parallel CRC have been provided which are essential for the understanding of the proposed scheme. Next in Section III, we propose multiple-bit parity prediction formulations for parallel CRC for all the three possible cases i.e. $l > m$, $l = m$ and $l < m$. In this section we also provide the architecture of our proposed scheme for all these cases. In Section IV, we present our analysis and simulation results followed by complexity analysis in Section V. Finally, we conclude the paper in Section VI.

II. Preliminaries

In this section the basic concept of CRC computation is described followed by a brief description of parallel CRC formulations. Then, matrix multiplication based parallel CRC architecture proposed in [17] is discussed. We have specially emphasized on this architecture since our proposed concurrent fault detection architecture described in the next section, is based on this architecture. It is noted that the proposed scheme can be applicable on the other parallel CRC structures as well. In case of other parallel CRC architecture, from the syndrome equation, using a similar approach as shown in the proof of Lemma 1, one can derive the necessary equation required to design the fault detection unit. For the parallel structure which uses matrix formulation, our approach can be directly used. Parallel CRC computations proposed in [9], [11], [12] and [20] are formulations where our proposed scheme can be applied. However, for non-matrix formulation structures, we need to follow a similar derivation along with certain modifications. As a result, our multiple bit parity prediction based approach will be applicable for most of the parallel CRC architecture by using a similar derivation on the digit level syndrome formulation.

A. CRC Basics

The generator polynomial of a systematic linear $(m+k, k)$ error detecting code over $GF(2)$ is represented by

$$G(x) = 1 + \sum_{i=1}^{\tau-1} g_i x^i + x^\tau + x^m, \;\; g_i \in \{0, 1\}. \qquad (1)$$

In CRC computation first, a message is processed to generate its CRC checksum. Then the CRC is appended to the message to form the codeword, which is transmitted to the receiver over the channel. The receiver again calculates the CRC of the codeword and compares it with the received CRC. If an error is found, the transmission protocol either discards the corrupted data and/or sends a retransmission request. The k - bit input message $U(x)$ can similarly be represented as $U(x) =$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

$\sum_{i=0}^{k-1} u_i x^i,\ \ u_i \in \{0,1\}$. The $m-$bit CRC checksum, known as the syndrome, is the remainder of the division of $U(x)\,x^m$ by $G(x)$ in $GF(2)$ i.e., $S(x) = (x^m\,U(x))\ mod\ G(x)$.The checksum $S(x)$ is concatenated with the input message $U(x)$ to generate the encoded message $M(x) = x^m\,U(x) + S(x)$, which is transmitted over the transmission channel. In a CRC decoder, $M(x)$ is divided by $G(x)$ to find the remainder. A non-zero value of the remainder implies the presence of an error.

The above polynomial division of CRC is achieved by feeding the message bits $U(x)$ into a LFSR structure consisting of registers each of which is accompanied by common clock and clear signals. The coefficient of the highest order term is the first bit to enter the LFSR, while the coefficient of the lowest order term is the last bit [8]. At the end, the calculated value of the remainder is shown by the content of the registers. It is noted that syndrome calculation of a $k-$bit message requires k clock cycles in a LFSR-based CRC computation. The number of clock cycles can be reduced using parallel circuit. This is explained in the next section.

B. Parallel CRC

The parallel computation of CRC can be performed by treating the message block-wise iteratively where l message bits are processed at each iteration. If $k\ mod\ l \neq 0$, then $(l-k\ mod\ l)$ zeroes are prepended to $U(x)$ to make the message length a multiple of l. The $k-$bit message is split into $q = \lceil \frac{k}{l} \rceil$ message blocks each being $l-$bit long. Thus as shown in Fig. 1, $U(x)$ is divided into q parts, $U(x) = \sum_{n=0}^{q-1} U_n(x)$, where $U_n(x) = B^{(n)}(x)\,x^{l(q-1-n)}$ and $B^{(n)}(x) = \sum_{j=0}^{l-1} u_{l(q-1-n)+j}\,x^j$.

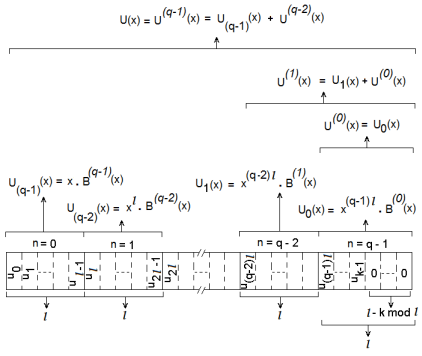

Fig. 1: Message splitted in q blocks each being of length l.

Since $U_n(x)$ is the l-bit message block or binary polynomial being processed at the n-th iteration, $B^{(n)}(x)$ can have maximum of degree $(l-1)$ and can also be represented as

$$B^{(n)}(x) = \sum_{j=0}^{l-1} b_j^{(n)} x^j,\ \ b_j^{(n)} \in \{0,1\}. \tag{2}$$

The portion of $U(x)$ that contains all the blocks for $j=0$ to $j=n$ is denoted by $U^{(n)}(x)$ and can be written as a recursive equation as follows

$$U^{(n)}(x) = \sum_{j=0}^{n} U_j(x) \ = \sum_{j=\{(q-1)-n\}l}^{k-1} u_j x^j$$

$$= x^l U^{(n-1)}(x) + B^{(n)}(x), \tag{3}$$

where $U^{(0)}(x) = U_0(x)$ and $U^{(q-1)}(x) = U(x)$.

In Fig. 1, it is shown that $U_0(x) = \sum_{j=(q-1)l}^{k-1} u_j x^j$ which has $(l - k\ mod\ l)$ zeroes prepended to it. In the first iteration cycle of the parallel CRC structure, $U_0(x)$ is processed to generate the output $S^{(0)}(x)$. Then in the next iteration, $U_1(x) = \sum_{j=(q-2)l}^{(q-1)l-1} u_j x^j$ is processed to generate the output $S^{(1)}(x)$. So after two cycles the processed portion of the message is $U^{(1)}(x) = U^{(0)}(x) + U_1(x)$. In this way, after $(q-1)$ cycles, the processed message is $U^{(q-2)}(x) = U^{(q-3)}(x) + U_{(q-2)}(x)$. Then in the last iteration cycle, the last block i.e. $U_{(q-1)}(x)$ is processed to generate $S^{(q-1)}(x)$. So after the last iteration cycle the processed message is $U^{(q-1)}(x) = U^{(q-2)}(x) + U_{(q-1)}(x)$, which is actually the whole message as shown in Fig. 1.

Let $S^{(n)}(x)$ be the syndrome of $U^{(n)}(x)$, which can be represented as

$$S^{(n)}(x) \ = (x^m\,U^{(n)}(x))\ mod\ G(x). \tag{4}$$

By substituting (3) into (4), one can obtain

$$S^{(n)}(x) = \{x^l\,S^{(n-1)}(x) + x^m\,B^{(n)}(x)\}\ mod\ G(x). \tag{5}$$

We assume $l > m$ for the rest of the discussion in this section. The other two conditions, namely $l < m$ and $l = m$, will be considered in Section IV. Now we get from [17], that for $l > m$ (5) reduces to

$$S^{(n)}(x) = \{x^m\,T^{(n)}(x)\}\ mod\ G(x), \tag{6}$$

where

$$T^{(n)}(x) = x^{l-m}\,S^{(n-1)}(x) + B^{(n)}(x). \tag{7}$$

$T^{(n)}(x)$ can be represented as

$$T^{(n)}(x) = \sum_{j=0}^{l-1} t_j^{(n)}\,x^j,\ \ t_j^{(n)} \in \{0,1\}, \tag{8}$$

and $S^{(n)}(x)$ can be represented as $S^{(n)}(x) = \sum_{j=0}^{m-1} s_j^{(n)}\,x^j,\ \ s_j^{(n)} \in \{0,1\}$. From (6), (7) and (8), we can write

$$t_j^{(n)} = \begin{cases} b_j^{(n)} & j \in [0, l-m-1] \\ b_j^{(n)} + s_{j-(l-m)}^{(n-1)} & j \in [l-m, l-1]. \end{cases} \tag{9}$$

A matrix multiplication scheme has been proposed in [17], which represents (8) in matrix multiplication notations as

$$T^{(n)}(x) = \mathbf{x}\ \mathbf{t}^{(n)}, \tag{10}$$

where $T^{(n)}(x)$ is a scalar, $\mathbf{x} = \begin{bmatrix} x^0, & ..., & x^{l-1} \end{bmatrix}$ is the row vector and $\mathbf{t}^{(n)} = \begin{bmatrix} t_0^{(n)}, & ..., & t_{l-1}^{(n)} \end{bmatrix}^T$ is the column vector.

Similarly, using (7) and (10), we can get the matrix formulation as

$S^{(n)}(x) = (x^m\ \mathbf{x}\,\mathbf{t}^{(n)})\ mod\ G(x) = \mathbf{x}\,\mathbf{s}^{(n)}$, where

$$\mathbf{s}^{(n)} = \mathbf{G}\ \mathbf{t}^{(n)}. \tag{11}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

$\mathbf{s^{(n)}} = \begin{bmatrix} s_0^{(n)} ... s_{m-1}^{(n)} \end{bmatrix}^T$ is the column vector and $\mathbf{G}$ is a $m \times l$ matrix whose each element can be represented as $g_{i,j}$ when $g_{i,j} \in \{0, 1\}$ for $0 \leq i \leq m-1$ and $0 \leq j \leq l-1$.

The parallel implementation of a basic structure of CRC is shown in Fig. 2.

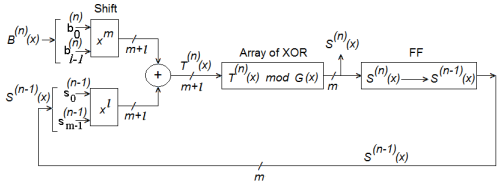

Fig. 2: Parallel Implementation of CRC. The block diagram is derived from [9], [11].

In this figure, at the n-th iteration, l- bit $B^{(n)}(x)$ and m-bit $S^{(n-1)}(x)$ are shifted and XORed to generate $(m+l)$-bit $T^{(n)}(x)$ according to (7) and then following (6), $T^{(n)}(x)$ is made to pass through an array of XOR gates to generate $T^{(n)}(x)\ mod\ G(x)$ which is the next state value of the syndrome $S^{(n)}(x)$. This $S^{(n)}(x)$ is then passed through flip-flops and the current state value of the syndrome $S^{(n-1)}(x)$ is sent back as feedback. In the first iteration cycle, $B^{(0)}(x)$ and $S^{(-1)}(x)$ are processed to generate $S^{(0)}(x)$ when $S^{(-1)}(x) = 0$, while, in the second iteration cycle, this $S^{(0)}(x)$ is fed back and processed with $B^{(1)}(x)$ to generate $S^{(1)}(x)$. Both $B^{(0)}(x)$ and $B^{(1)}(x)$ can be obtained from $U^{(0)}(x)$ and $U^{(1)}(x)$ as shown in Fig. 1. In a similar way, the whole message is processed in total q cycles. In the last cycle, $B^{(q-1)}(x)$ and $S^{(q-2)}(x)$ are processed to generate $S^{(q-1)}(x)$ which is the final CRC value i.e. $S^{(q-1)}(x) = S(x)$. The relation between $U^{(q-1)}(x)$ and $B^{(q-1)}(x)$ is shown in Fig. 1.

The parallel implementation based on the above matrix multiplication of CRC for the case of $l > m$ proposed in [17] is shown in Fig. 3.

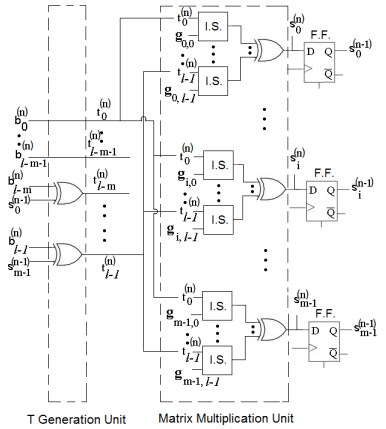

Fig. 3: Parallel Implementation of CRC based on Matrix Multiplication for $l > m$ [17]. The circuit is derived from [9], [11].

In Fig. 3, the whole design has been split in two different blocks, specifically the T Generation Unit and the Matrix Multiplication Unit. The shift and the adder with two $(m+l)$ inputs stage shown in Fig. 2 are combined into a single stage in Fig. 3 and is named the T Generation Unit, while the reduction stage consisting of an array of XOR in Fig. 2 is named the Matrix Multiplication Unit in Fig. 3, since the reduction is achieved using matrix multiplication as given in (11). In the T Generation Unit, checksum values from a previous iteration i.e. $S^{(n-1)}(x)$ and the present block of the message i.e. $B^{(n)}(x)$ are taken as inputs and XORed to get the output which is $T^{(n)}(x)$ according to (7). In the Matrix Multiplication Unit, this $T^{(n)}(x)$ is multiplied with $\mathbf{G}$ for the computation of the present CRC terms $S^{(n)}(x)$ based on (11). Each I.S. box in this unit works as input selection (I.S.) depending on its input from the $\mathbf{G}$ matrix. We consider $g_{i,j}$ as the i-th row and j-th column element of the $\mathbf{G}$ matrix. The functionality of I.S. box is similar to the AND gate. If the value of $g_{i,j}$ is 0, then the corresponding I.S. box becomes open-circuit and if the value of $g_{i,j}$ is 1, then the corresponding I.S. box becomes short-circuit, i.e. if the value of $g_{i,j}$ is 1, then the output of the corresponding I.S. box is t_j and if the value of $g_{i,j}$ is 0, then the output is 0.

In the next section, we present the formulation of multiple-bit parity-based concurrent fault detection of parallel CRC architecture. Towards that end, we derive a formulation for the calculation of the predicted parity of the syndrome bits , for each of the three conditions i.e. $l > m$, $l = m$ and $l < m$, when l is the number of input message bits in each iteration and m is the number of CRC bits. We start with the case $l > m$ since the matrix based CRC formulation of this case has already been presented in [17]. For the other two cases, we have to extend the idea given in [17] to generate the required CRC formulation. In this section, we also present the architecture of the proposed concurrent fault detection of parallel CRC structure based on multiple-bit parity prediction for all the three cases. We present the multiple-bit parity formulation and architecture following the order, i.e. first the structure for $l > m$ is discussed followed by the design for $l = m$ and finally the structure corresponding to $l < m$ is constructed.

Before going to the next section, we present TableI listing all the symbols used throughtout this paper along with their definitions.

III. Concurrent Fault Detection of Parallel CRC Structure

In this section, we consider the parallel CRC architecture proposed in [17] to apply a multiple-bit parity approach on this architecture to achieve concurrent fault detection. This multiple-bit parity checking circuit will compare predicted parity and actual parity of the syndrome bits at the end of each iteration and raise an alarm if they do not match. The block diagram of the proposed scheme has been depicted in Fig. 4.

In Fig. 4, the output of the parity calculation unit is the actual multiple-bit parity of CRC syndrome bits after n-th iteration , whereas the output of the parity prediction unit is the predicted multiple-bit parity. The actual parity is obtained by XORing the outputs of the Matrix Multiplication Unit, whereas the predicted parity is evaluated as a different function of inputs. Functionality of T Generation Unit and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

TABLE I: List of the Symbols used throughout the paper.

Symbols	Definitions
l	Number of input message bits in each iteration
m	Degree of CRC generator polynomial
$G(x)$	Generator polynomial of CRC over $GF(2)$
$U(x)$	k - bit input message
$S(x)$	$m-$bit CRC checksum, known as the syndrome
$M(x)$	The encoded message to be transmitted
q	Number of message blocks each being $l-$bit long
$U_n(x)$	Message fragment processed at $n-$th iteration
$U^{(n)}(x)$	Message segment containing all the blocks processed till $n-$th iteration
$B^{(n)}(x)$	Present block of the message when $U_n(x) = B^{(n)}(x)\, x^{l(q-1-n)}$
$T^{(n)}(x)$	$x^{l-m}\, S^{(n-1)}(x) + B^{(n)}(x)$
$\mathbf{G}$	$m \times l$ matrix whose each element can be represented as $g_{i,j}$
$s_i^{(n)}$	i-th bit of the m-bit syndrome after the n-th iteration
$\overline{p}(s_c^{(n)})$	Predicted parity of the $c-$th block of the syndrome for $n-$th iteration
$p(g_{j,c})$	Parity of the r elements of $j-$th column of $\mathbf{G}$
$p(s_0^{(n)})$	Actual parity of the $c-$th block of the syndrome for $n-$th iteration
w	Number of parity bits i.e. number of blocks in which $m-$bit CRC output is splitted
r	Number of bits of each block
$e(x)$	Error polynomial
E_c	Theoretical error coverage

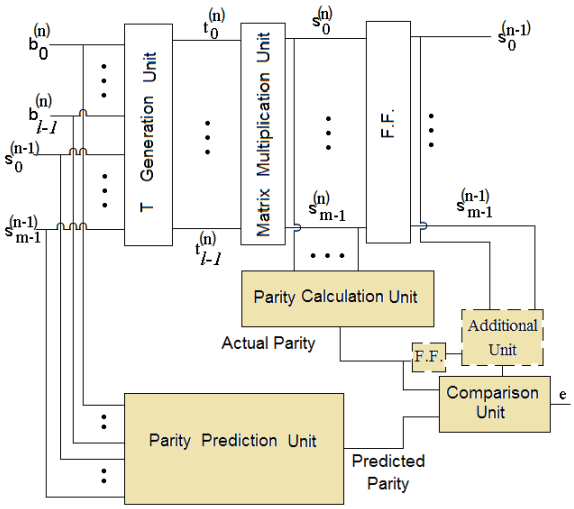

Fig. 4: Simplified Block Diagram of the Proposed Multiple-bit Parity-based Error Detection Structure of Parallel CRC Architecture. The coloured portion shows the overhead.

Matrix Multiplication Unit have already been explained in Section II regarding Fig. 3. The coloured portion of Fig. 4 shows the overhead. The additional design unit, shown in Fig. 4, computes the parity of the output of the flip-flops and compares it with the actual parity.

Now we propose the formulation for the multiple-bit parity prediction when $l \geq m$ and then we show the corresponding hardware structure.

A. Formulation of Parity Prediction for $l \geq m$

In this subsection, first we present formulations for a multiple-bit parity prediction architecture for CRC when the number of message bits in each iteration, i.e., l is greater than or equal to the number of CRC bits, i.e., m.

From (9) and (11), we get the value of the i-th bit of the m-bit syndrome after the n-th iteration for $l > m$ as follows,

$$s_i^{(n)} = \sum_{j=0}^{l-m-1} g_{i,j}\, b_j^{(n)} + \sum_{j=l-m}^{l-1} g_{i,j}\, (b_j^{(n)} + s_{j-(l-m)}^{(n-1)}). \quad (12)$$

Again, for $l = m$, we derive from (5), $S^{(n)}(x) = \{x^m\, T^{(n)}(x)\}\ mod\ G(x)$, where

$$T^{(n)}(x) = S^{(n-1)}(x) + B^{(n)}(x). \quad (13)$$

Next from (6), (7), (8) and (13), for $j \in [0, l-1]$,

$$t_j^{(n)} = b_j^{(n)} + s_j^{(n-1)}(x). \quad (14)$$

Now from (2), (11), (12) and (14), we derive the value of the syndrome after the n-th iteration,

$$s_i^{(n)} = \sum_{j=0}^{l-1} g_{i,j}\, (b_j^{(n)} + s_j^{(n-1)}). \quad (15)$$

The above formulation can be extended for the multiple-bit parity structure as explained in the next lemma. We consider that the parallel CRC structure, having l-bit input and m-bit output, is splitted into w blocks while each block is characterized by r-bit output. Without loss of generality, we can consider that $m = w \times r$. Using this information, we propose a lemma for w-bit parity.

Lemma 1. *Let $\overline{p}(s_c^{(n)}) \in \{0,1\}$ be the predicted parity of the c-th block of the syndrome for n-th iteration* and $p(g_{j,c}) \in \{0,1\}$ be the parity of the r elements of the j-th column of $\mathbf{G}$ starting from $c.r$-th element, then *for $l \geq m$,*

$$\overline{p}(s_c^{(n)}) = \sum_{j=0}^{l-1} b_j^{(n)}\, p(g_{j,c}) + \sum_{j=l-m}^{l-1} s_{j-(l-m)}^{(n-1)}\, p(g_{j,c}). \quad (16)$$

where $p(g_{j,c})$ is the parity of the r elements of the j-th column of $\mathbf{G}$ starting from the cr-th element and is described as $p(g_{j,c}) = \sum_{i=cr}^{r(c+1)-1} g_{i,j}$ and $0 \leq c \leq w-1$.

Proof: Using (12), the predicted parity of the c-th block of the syndrome after the n-th iteration can be expressed as,

$$\overline{p}(s_c^{(n)}) = \sum_{i=cr}^{r(c+1)-1} s_i^{(n)}, \quad (17)$$

From (12) and (17) for $l > m$,

$$\overline{p}(s_c^{(n)}) = \sum_{i=cr}^{rc+r-1} \left[\sum_{j=0}^{l-m-1} g_{i,j}\, b_j^{(n)} + \right.$$

$$\left. + \sum_{j=l-m}^{l-1} g_{i,j}\, (b_j^{(n)} + s_{j-(l-m)}^{(n-1)}) \right]. \quad (18)$$

Again for $l = m$, the predicted parity for the c-th block

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

$$\overline{p}(s_c^{(n)}) = \sum_{i=cr}^{r(c+1)-1} \left[\sum_{j=0}^{l-1} g_{i,j} \, (b_j^{(n)} + s_j^{(n-1)}) \right]. \qquad (19)$$

Now combining (18) and (19), we can further derive (16) for $l \geq m$. ∎

This formulation will be used to generate the proposed fault detection of parallel CRC architecture using multiple-bit parity for $l \geq m$. A parity prediction circuit will be designed based on (16) and the actual multiple-bit parity of the syndrome will be compared with these predicted parity bits generating an error signal in the case of a mismatch.

Now we will provide an illustrative example for the case $l = m$.

Example 1. We consider CRC generator polynomial as $G(x) = 1 + x^3 + x^4 + x^5 + x^8$ and degree of parallelism $l = 8$. Then the equation of the first column of $\mathbf{G}$ is $r_0(x) = 1 + x^3 + x^4 + x^5$. Now we get,

$r_1(x) = x + x^4 + x^5 + x^6$, $r_2(x) = x^2 + x^5 + x^6 + x^7$, $r_3(x) = 1+x^4+x^5+x^6+x^7$, $r_4(x) = 1+x+x^3+x^4+x^6+x^7$, $r_5(x) = 1 + x + x^2 + x^3 + x^7$, $r_6(x) = 1 + x + x^2 + x^5$, $r_7(x) = x + x^2 + x^3 + x^6$. Combining all these,

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

We assume that $B^{(n)}(x) = 1+x^6+x^7$ and $S^{(n-1)}(x) = 1 + x^2 + x^6$. So, $b_0^{(n)} = 1$, $b_1^{(n)} = 0$, $b_2^{(n)} = 0$, $b_3^{(n)} = 0$, $b_4^{(n)} = 0$, $b_5^{(n)} = 0$, $b_6^{(n)} = 1$, $b_7^{(n)} = 1$ and $s_0^{(n-1)} = 1$, $s_1^{(n-1)} = 0$, $s_2^{(n-1)} = 1$, $s_3^{(n-1)} = 0$, $s_4^{(n-1)} = 0$, $s_5^{(n-1)} = 0$, $s_6^{(n-1)} = 1$, $s_7^{(n-1)} = 0$. Now using (15), we get, $s_0^{(n)} = g_{0,2} + g_{0,7} = 0$, $s_1^{(n)} = g_{1,2} + g_{1,7} = 1$, $s_2^{(n)} = g_{2,2} + g_{2,7} = 0$, $s_3^{(n)} = g_{3,2} + g_{3,7} = 1$, $s_4^{(n)} = g_{4,2} + g_{4,7} = 0$, $s_5^{(n)} = g_{5,2} + g_{5,7} = 1$, $s_6^{(n)} = g_{6,2} + g_{6,7} = 0$, $s_7^{(n)} = g_{7,2} + g_{7,7} = 1$. So, output of the parallel CRC circuit for n-th iteration is $S^{(n)}(x) = x + x^3 + x^5 + x^7$. Now considering $w = 4$ and $r = 2$, actual multiple-bit parity of these syndrome bits can be represented as $p(s_0^{(n)}) = s_0^{(n)} + s_1^{(n)} = 1$, $p(s_1^{(n)}) = s_2^{(n)} + s_3^{(n)} = 1$, $p(s_2^{(n)}) = s_4^{(n)} + s_5^{(n)} = 1$, $p(s_3^{(n)}) = s_6^{(n)} + s_7^{(n)} = 1$.

Again considering $w = 4$, we get the values of the predicted parity from (16) as, $\overline{p}(s_0^{(n)}) = p(g_{2,0}) + p(g_{7,0}) = (g_{0,2} + g_{1,2}) + (g_{0,7} + g_{1,7}) = s_0^{(n)} + s_1^{(n)} = p(s_0^{(n)}) = 1$. Similarly, $\overline{p}(s_1^{(n)}) = p(g_{2,2}) + p(g_{7,2}) = (g_{2,2} + g_{3,2}) + (g_{2,7} + g_{3,7}) = s_2^{(n)} + s_3^{(n)} = p(s_0^{(n)}) = 1$, $\overline{p}(s_2^{(n)}) = p(g_{2,4}) + p(g_{7,4}) = (g_{4,2} + g_{5,2}) + (g_{4,7} + g_{5,7}) = s_4^{(n)} + s_5^{(n)} = p(s_2^{(n)}) = 1$ and $\overline{p}(s_3^{(n)}) = p(g_{2,6}) + p(g_{7,6}) = (g_{6,2} + g_{7,2}) + (g_{6,7} + g_{7,7}) = s_6^{(n)} + s_7^{(n)} = p(s_3^{(n)}) = 1$. The values of the predicted parity exactly matches with the values of the actual parity which clearly shows the correctness of the proposed Lemma 1.

Next, we assume there is a stuck-at-fault in the syndrome generation circuit resulting in an erroneous value of the CRC output. Let the actual erroneous output be $S^{(n)}(x) = x+x^3+x^6+x^7$. Now, the 4−bit parity of the actual syndrome bits are $p(s_0^{(n)}) = 1$, $p(s_1^{(n)}) = 1$, $p(s_2^{(n)}) = 0$ and $p(s_3^{(n)}) = 0$. A comparison of the actual parity and the predicted parity shows a mismatch in the value of $p(s_2^{(n)})$ and $\overline{p}(s_2^{(n)})$, i.e., a mismatch between the predicted parity and actual parity of the block denoted by $c = 2$. Similarly another mismatch is seen between $p(s_3^{(n)})$ and $\overline{p}(s_3^{(n)})$, i.e., between the predicted parity and actual parity of the block denoted by $c = 3$. These two errors will result in the flag to raise an alarm depicting a successful concurrent fault detection. It should be noted that, since there are 2 erroneous bits in the output, single bit parity prediction based fault detection will not be able to detect this error. But our multiple-bit parity scheme can easily detect this kind of error. A more detailed investigation towards this direction is presented in Section IV.

B. Multiple-bit Parity Prediction Architecture for $l \geq m$

In this section, we present the architecture of the proposed concurrent fault detection of parallel CRC structure based on multiple-bit parity prediction. We have already formulated multiple-bit parity prediction expressions for $l \geq m$ when l is the number of input message bits in each iteration and m is the number of CRC bits.

Fig. 5(a) shows the block diagram of the proposed multi bit parity prediction scheme and Fig. 5(b) shows the parity prediction circuit developed in (16). In Fig. 5(b) each I.S. has one input as $p(g_{j,c})$, when $p(g_{j,c})$ is the parity of the r elements of j-th column of $\mathbf{G}$ starting from $r.c$-th element for $0 \leq j \leq l-1$. In Fig. 5(b), the outputs of the I.S. boxes are shown to be connected to the Binary tree of XOR (BTX) block. The number of inputs to the BTX block is equal to the number of ones in $p(g_{j,c})$ for $0 \leq j \leq l-1$, which is fixed for a given generator polynomial or $G(x)$.

In Fig. 5(a), the checksum portion of the design is shown to have been divided into w blocks while c denotes the number of the block. Each block consists of r syndrome bits. After n-th iteration the single bit predicted parity of all those w blocks range from $\overline{p}(s_0^{(n)})$ to $\overline{p}(s_{w-1}^{(n)})$ forming a multiple-bit parity prediction having width w. This multiple-bit parity is compared to the actual parity in the Comparison Unit and any discrepency will raise an alarm. The Comparison Unit includes w 2-input XOR gates and a two rail parity checker circuit consisting of a w-input AND gate and another w-input NOR gate. Since this final error signal is a single point of failure, the inclusion of a two rail parity check structure and two final error signals ensure detection of any stuck at faults in the comparison unit. This multiple-bit parity prediction architecture has a definite advantage. From the mismatch, we can approximately figure out the location of the faulty portion in the circuit depending upon the location of the erroneous actual parity bit. The mapping between a fault in the parallel CRC circuit and an error in the actual parity bits will be discussed in the next section.

Next we present the multiple-bit parity prediction architecture for $l = m$. Since this diagram corresponds to the case $l = m$, the internal structure of T Generation Unit and Matrix Multiplication Unit differs from those shown in Fig. 3. Hence those details are shown in Fig. 5(c). The details of the Parity Prediction Unit basically the internal structure of the design of the calculation of each predicted parity bits based on (16) corresponding to each block is shown in Fig. 5(d). Each I.S. in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

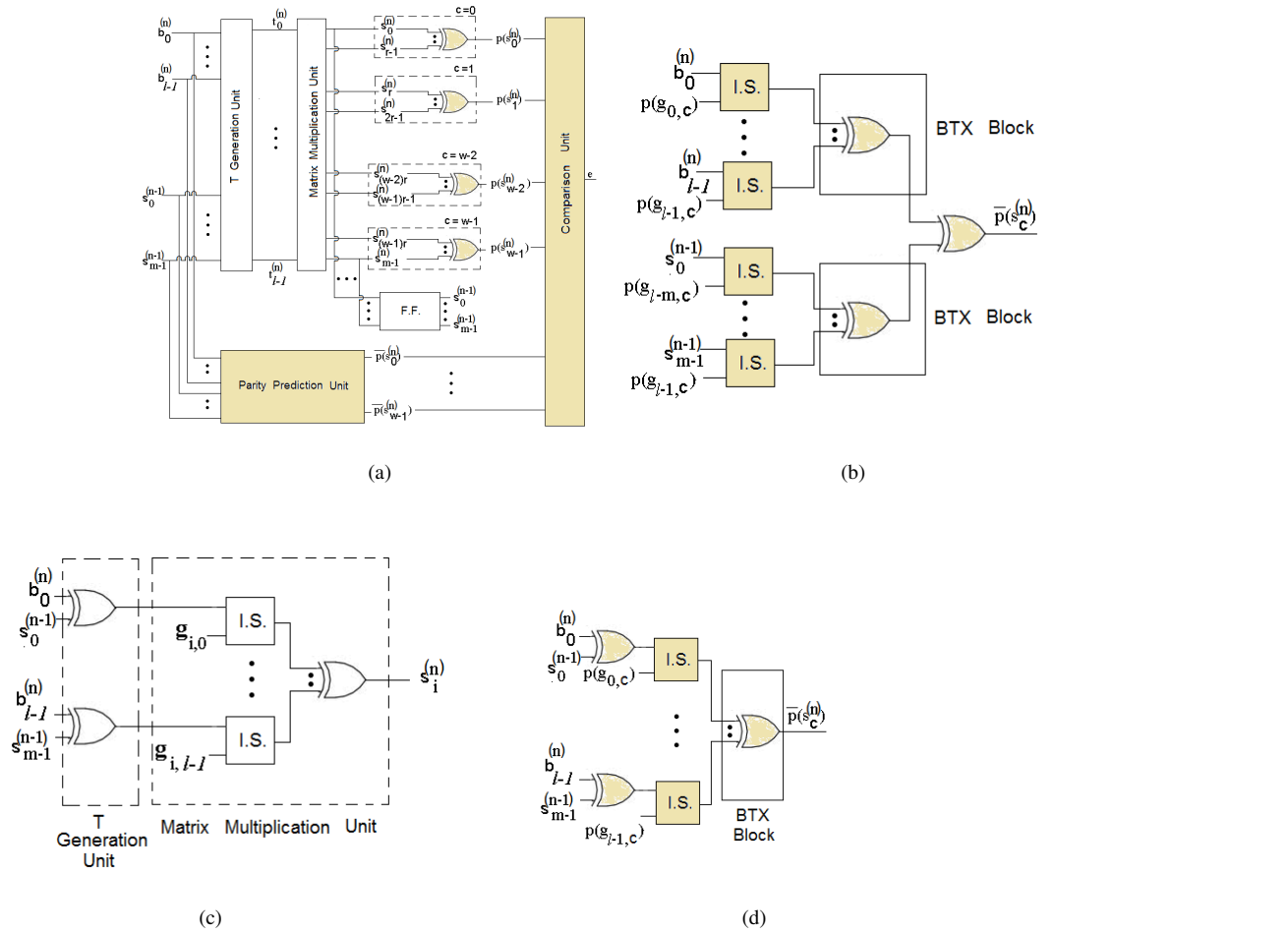

Fig. 5: (a) Block Diagram of the Proposed Multi bit Scheme, (b) Parity Prediction Circuit of each block for $l > m$, (c) Details of T generation Unit and Matrix Multiplication Unit for $l = m$, (d) Parity Prediction Circuit of each block for $l = m$.

Fig. 5(d) has one input as $p(g_{j,c})$, when $p(g_{j,c})$ is the parity of the r elements of j-th column of $\mathbf{G}$ starting from $r.c$-th element for $0 \leq j \leq l-1$.

The syndrome bits for this case also are divided into w blocks while c denotes the number of the blocks similar to the case shown in Fig. 5(a). The single bit predicted parity of all those w blocks after n-th iteration are shown to range from $\overline{p}(s_0^{(n)})$ to $\overline{p}(s_{w-1}^{(n)})$ thus forming a multiple-bit parity prediction having width w. Similar to the concept shown in Fig. 4, the w -bit predicted parity is compared to the actual w -bit parity and a high value of error (e) indicates the presence of an error. Similar to the previous case, in this case too we have the advantage of approximately catching the faulty portion of the circuit depending on the location of the erroneous actual parity bit.

In the next subsection, the parallel CRC architecture proposed in [17] for the case $l \geq m$ has been extended to generate the parallel CRC structure for $l < m$. Then we apply a multiple-bit parity approach on this architecture to achieve concurrent fault detection. Similar to the case depicted in the previous section, this multiple-bit parity checking circuit will also compare predicted parity and actual parity of the syndrome bits at the end of each iteration and raise an alarm if they do not match as shown in Fig. 4.

C. Formulation of Parity Prediction for $l < m$

Next we consider the last remaining case when the number of message bits in each iteration (l) is less than the number of CRC bits (m). First we derive the matrix based parallel CRC formulation based on (12). This is an extension of the idea proposed in [17]. Then we propose a formulation and then, by the support of this formulation, we propose a lemma to design a multiple-bit parity prediction architecture of CRC for $l < m$. Towards that end, we define a $m \times (m-l)$ matrix $\mathbf{H}$, which will be required further on in this subsection. Each element of matrix $\mathbf{H}$ can be represented as $h_{i,j}$ when $0 \leq i \leq m-1$ and $0 \leq j \leq m-l-1$ and for each i- th column only $h_{i+l,\,i} = 1$ and the rest of the elements are zero.

For $l < m$, we derive from (5), $S^{(n)}(x) = T_t^{(n)}(x) \; mod \; G(x)$, where

$$T_t^{(n)}(x) = T_1^{(n)}(x) + T_2^{(n)}(x), \tag{20}$$

$$T_1^{(n)}(x) = x^l \sum_{j=0}^{m-l-1} s_j^{(n-1)} \, x^j, \tag{21}$$

and

$$T_2^{(n)}(x) = x^m \sum_{j=0}^{l-1} (s_{j+m-l}^{(n-1)} + b_j^{(n)}) \, x^j. \tag{22}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

Using (11), we can derive from (20), (21) and (22),

$$\mathbf{s^{(n)}} = \mathbf{H}\,\mathbf{t_1^{(n)}} + \mathbf{G}\,\mathbf{t_2^{(n)}}, \tag{23}$$

when we can express $T_1^{(n)}(x) = \mathbf{x}\,\mathbf{t_1^{(n)}}$, and $T_2^{(n)}(x) = \mathbf{x}\,\mathbf{t_2^{(n)}}$. Here $\mathbf{t_1^{(n)}} = \left[t_{1\,0}^{(n)},\ ...,\ t_{1\,(m-l)-1}^{(n)}\right]^T$ and $\mathbf{t_2^{(n)}} = \left[t_{2\,0}^{(n)},\ ...,\ t_{2\,(m-l)-1}^{(n)}\right]^T$ are the column vector. Now from (23),

$$s_i^{(n)} = \sum_{j=0}^{m-l-1} h_{i,j}\, s_j^{(n-1)} + \sum_{j=0}^{l-1} g_{i,j}\,(b_j^{(n)} + s_{j+m-l}^{(n-1)}). \tag{24}$$

Similar to the previous cases discussed in the last two subsections, in the present case too, we extend the above-mentioned formulation to the multiple-bit parity structure. The parallel CRC structure involving l-bit input and m-bit output is considered to be splited into w blocks each having r-bit output. We can safely consider without loss of generality that $m = w\ \times r$. A lemma is proposed using this information for $l < m$.

Lemma 2. *Let $\overline{p}(s_c^{(n)}) \in \{0,1\}$ be the predicted parity of the c-th block of the syndrome for the n-th iteration, $p(g_{j,c}) \in \{0,1\}$* be the parity of the r elements of the j-th column of $\mathbf{G}$ starting from the $c.r$-th element and $p(h_{j,c}) \in \{0,1\}$ be the parity of the r elements of the j-th column of $\mathbf{H}$ starting from the $c.r$ -th element, then

$$\overline{p}(s_c^{(n)}) = \sum_{j=0}^{m-l-1} s_j^{(n-1)}\, p(h_{j,c}) + \sum_{j=0}^{l-1} (b_j^{(n)} + s_{j+m-l}^{(n-1)}\, p(g_{j,c})). \tag{25}$$

Proof: Similar to the proof of Lemma 1, using (17), we get $\overline{p}(s_c^{(n)}) = \sum_{i=cr}^{r(c+1)-1} s_i^{(n)}$, Now using (24),

$$\overline{p}(s_c^{(n)}) = \sum_{i=cr}^{r(c+1)-1} \left[\sum_{j=0}^{m-l-1} h_{i,j}\, s_j^{(n-1)} + \sum_{j=0}^{l-1} g_{i,j}\,(b_j^{(n)} + s_{j+m-l}^{(n-1)}). \right] \tag{26}$$

From (26) we can further derive (25). ■

We will use the above formulation to generate the proposed concurrent fault detection of CRC architecture using multiple-bit parity for $l < m$. The actual parity bits of the CRC syndrome are XORed with the predicted multiple-bit parity-based on (25). If these two sets of parity bits do not match with each other, then a flag indicating the occurance of an error is generated.

D. Multiple-bit Parity Prediction Architecture for $l < m$

Next we present the multiple-bit parity prediction architecture for $l < m$. In Fig. 6, the block diagram of the matrix formulation based parallel CRC architecture for $l < m$ have been shown in details. This circuit is designed from (24). Since this diagram corresponds to the case $l < m$, the internal structure of CRC Generation Unit greatly differs from T generation unit and Matrix Multiplication Unit as shown in Fig. 3 which corresponds to the case $l > m$.

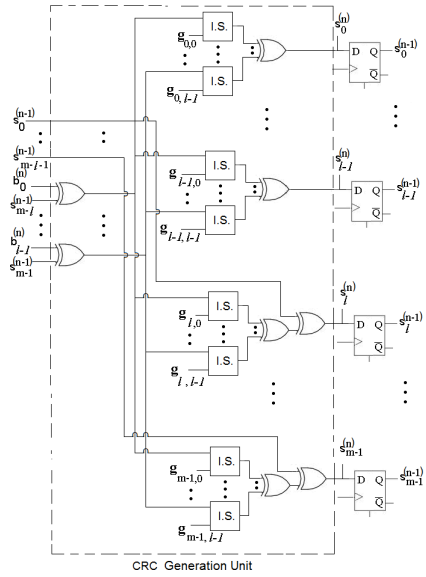

Fig. 6: Parallel CRC Structure based on Matrix Multiplication for $l < m$.

The block diagram of the proposed multiple-bit parity scheme for $l < m$ is similar to the diagram shown in Fig. 5(a). Next for $l < m$ the details of the Parity Prediction Unit basically the internal structure of the design of the calculation of each predicted parity bit corresponding to each block is shown in Fig. 7. This circuit is generated using (25). In Fig. 7 each I.S. has one input as $p(g_{j,c})$, when $p(g_{j,c})$ is the parity of the r elements of j-th column of $\mathbf{G}$ starting from $c.r$-th element for $0 \le j \le l-1$.

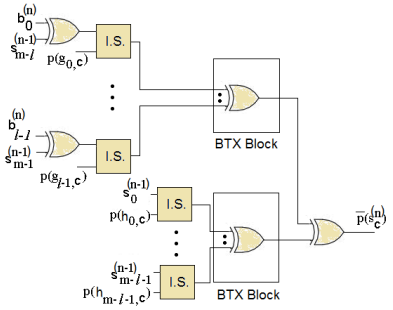

Fig. 7: Parity Prediction Circuit of each block of the Proposed Multi bit Scheme for $l < m$.

The actual parity bits are compared with the predicted parity bits shown to be generated in Fig. 7 and in presence of an error, e goes high. Both the predicted and actual parity are w-bit wide forming a multiple-bit parity prediction having width w. In this case also we have the advantage that from the mismatch, we can approximately figure out which portion of the circuit has got a fault depending on the location of the erroneous actual parity bit.

Now we are ready with all the required formulations to analyze the architecture of multiple-bit parity-based concurrent fault detection of parallel CRC architecture.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

IV. Analysis and Simulation Results

A physical defect due to imperfection, flaw, shorts or opens in a circuit is defined as a fault and an deviation from correct result is called error. In other words, faults occured in physical universe is manifested by errors in informational universe which further leads to failures [30]. In this section, we first discuss the theoretical error coverage analysis along with a comparison of detectable errors of our proposed technique with other error detection techniques. Then we propose a lemma for the computation of error coverage of our proposed scheme. We also present simulation results supporting the theoretical value obtained from the lemma. Finally we consider fault models and fault analysis of parallel CRC structures based on generator polynomials. At the end of this section, we present the results of the simulation in which first we inject faults into the parallel CRC structure and calculate the number of cases where the injected faults result in occured errors. Then we calculate the number of cases where our proposed scheme has been successful in identifying the errors. All these results have been presented in tabular format.

A. Error Model

In this subsection, we present a comparison of the error coverage among the three methods, namely Duplication with Compariosn, Single bit Parity Check and Proposed Multi bit Parity Check. Duplication with Comparison method can detect any error and single bit parity check can detect only odd number of bit flips. While our proposed method can detect any odd number of bit flips, it can also detect some of the even number of bit flips as well.

In this subsection, theoretical error coverage has also been explained. In the following subsections, fault types and fault models have been discussed. For $m-$bit CRC checksum, a fault has the effect of flipping any number of bits from those m bits. An addition of error polynomial with the expected checksum represents this fault. This error polynomial can be represented as $e(x) = \sum_{i=0}^{m-1} e_i x^i$ when $e_i \in GF(2)$. And then $Se(x) = S(x) + e(x)$ presents the erroneous output. A no error situation takes place when $e(x) = 0$ i.e. $e_i = 0$ for $0 \leq i \leq m-1$. An error in $i-$th bit position is manifested by $e_i = 1$ in the error polynomial because it flips the $i-$th bit making the output erroneous. Since there are total m bits in the output, there are total $2^m - 1$ possible errors.

Total number of possible 2-bit error in m-bit CRC syndrome is $\binom{m}{2}$. In a single bit parity check scheme of CRC, 2-bit error goes undetected. But our proposed multiple-bit scheme can detect some of the 2-bit errors. This scheme cannot detect a 2-bit error if both of the erroneous bits fall in the same block. So the number of possible undetectable 2-bit errors for multiple-bit scheme is equal to the product of the number of blocks, i.e., w and number of possible 2-bit errors in a block, i.e., $\binom{r}{2}$ where r is the number of bits in each block. Hence the number of possible detectable 2-bit errors in the proposed multiple-bit parity scheme = total number of possible 2-bit errors - number of undetectable 2-bit errors $=[\binom{m}{2} - w\binom{r}{2}] = [\frac{m(m-r)}{2}]$.

Similarly, we can calculate the number of 4-bit error detectable by multiple-bit parity scheme. Total number of possible 4-bit error in a m-bit CRC syndrome is $\binom{m}{4}$. For a multiple-bit parity scheme, there can be various ways in which erroneous bits can be distributed. The multiple-bit parity scheme cannot detect a 4-bit error in two cases. Firstly, when all of the four erroneous bits are in the same block which can happen in $w\binom{r}{4}$ ways. Secondly, when two of them are present in one block and the other two in another block which can happen in $\left[\frac{w}{2}\binom{r}{2}(w-1)\binom{r}{2}\right]$ways, while the reason for the division by 2 is the fact that each pair gets selected twice. So, the number of possible detectable 4-bit errors in the proposed multiple-bit parity scheme = total number of possible 4-bit errors - number of undetectable 4-bit errors = $\binom{m}{4} - \left[w\binom{r}{4} + \frac{w}{2}(w-1)\binom{r}{2}\binom{r}{2}\right] = \frac{m}{24}\left[(m-1)(m-2)(m-3) - (r-1)\{r^2 w - r(w-4) + 6\}\right]$.

A comparison of the fault coverage in terms of number of detectable errors of the above three schemes is shown in Table II.

B. Error Analysis

In this subsection we propose a lemma for the calculation of error coverage in terms of number of detectable errors using our proposed multiple-bit scheme.

Lemma 3. *Let m be the degree of CRC generator polynomial and w be the number of parity bits, then error coverage E_c can be expressed as*

$$E_c = \frac{2^{m-w}(2^w - 1) - 1}{2^m - 1}. \tag{27}$$

Proof: For m - bit CRC computation, total number of possible errors can be $N_{Total} = 2^m - 1$. An error detection scheme using single bit parity can detect error in presence of any odd number of erroneous bits, but it cannot detect error when number of erroneous bits is even. However, our proposed multiple-bit parity scheme is capable of detecting all odd number of errors and some of the even number of errors. For m - bit CRC computation, among all the possible errors, half of them will have odd number of erroneous bits. So possible odd number of errors is $N_{odd} = 2^{m-1}$. The rest of the errors have even number of erroneous bits. So possible even number of errors is $N_{even} = 2^{m-1} - 1$. Here this -1 corresponds to the case of zero number of errors which means actually no error. The proposed multiple-bit scheme consists of w blocks while each block is characterized by r bits. Similar to the previous reasoining here also we can write that, in each block total number of errors, possible odd number of errors and possible even number of errors can be given by respectively $n_{total} = 2^r - 1$, $n_{odd} = 2^{r-1}$ and $n_{even} = 2^{r-1} - 1$. Now for our proposed scheme, an error will go undetected in case if all the blocks contain either no error or even number of errors, i.e., only if there is no odd number of error in any of the blocks our scheme will fail to detect the error. So for our proposed scheme, number of undetectable error, $N_{Undetectable} = 2^{m-w}$. So error coverage, which is the ratio of detectable errors and total number of errors, can be expressed as

$$E_c = \frac{N_{Total} - N_{Undetectable}}{N_{Total}}$$

$$= \frac{(2^m - 1) - 2^{m-w}}{2^{m-1}} = \frac{2^{m-w}(2^w - 1) - 1}{2^m - 1}.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

TABLE II: Comparison of Theoretical Fault Coverage in terms of Detectable Errors.

	Duplication with Compariosn	Single bit Parity Check	Proposed Multi bit Parity Check
Single bit Error	100%	100%	100%
Odd number of bits in Error	100%	100%	100%
Double Bit Errors	$\frac{m(m-1)}{2}$	0	$\frac{m(m-r)}{2}$
Quadruple Bit Errors	$\frac{m(m-1)(m-2)(m-3)}{24}$	0	$\frac{m}{24}\left[(m-1)(m-2)(m-3)-(r-1)\{r^2w-r(w-4)+6\}\right]$

TABLE III: Theoretical Complexity Analysis in terms of area overhead and critical path delay.

	Double Modular Redundancy	Proposed Multi bit Parity Check
Number of XOR gates	$2ml$	$ml+w(l+m)$
Number of FFs	m	m
Delay	$T_X+\lceil log_2 l\rceil T_X+T_X+\lceil log_2 m\rceil T_X$	$T_X+\lceil log_2 l\rceil T_X+T_X+\lceil log_2 r\rceil T_X+\lceil log_2 w\rceil T_X$

■

We have simulated our proposed scheme for several frequently used CRC generator polynomials. The simulation has been performed for $m = 8$, $m = 16$ and $m = 32$. The polynomials used for these simulations are mentioned here. For $m = 8$, we have used $G(x) = x^8+x^2+x+1$. For $m = 16$, we have used $G(x) = x^{16}+x^{15}+x^2+1$. Finally for $m = 32$, we have used $G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+ +x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$. For each of these values of m, number of parity bits, i.e., w is changed and error coverage value is recorded for each particular value of w for corresponding value of m.

A thorough investigation of the simulation results validates the accuracy of Lemma 3. For this simulation, we have introduced errors in the CRC checksum output bits i.e. we made some of the bits erroneous to find out whether the proposed method can catch the error. For CRC-8, the error can occur in 8 output bits in total $\left(2^8-1\right)$ ways and we have injected all $\left(2^8-1\right)$ number of possible errors. But for CRC-16, and for CRC-32, we have introduced random errors while making sure to include all the smaller number of errors, i.e., the cases when the number of errors are one, two, three, four and five.

C. Fault Model and Analysis of CRC based on Generator Polynomial

In this subsection, we analyse the possible fault locations and fault propagation through the CRC structure shown in Fig. 3. Now we consider various types of faults and their corresponding locations. We can safely assume without loss of generality that the inputs i.e. $B^{(n)}(x)$, $S^{(n-1)}(x)$ and $\mathbf{G}$ matrix are free from errors. Error due to stuck-at faults can take place in the calculation of $T^{(n)}(x)$ and $S^{(n)}(x)$. Now first we consider the case when there is an error in $T^{(n)}(x)$. This error may not propagate if the corresponding bit position of $\mathbf{G}$ matrix for a particular bit of $S^{(n)}(x)$ is 0. But this same error can flip another output bit. In the Matrix Multiplication Unit, I.S. boxes are followed by XOR gates and any single bit error in the input of a XOR gate is sure to propagate through the BTX block to reach the output. Hence any error present in the I.S. box output will propagate to the CRC checksum output and will generate an erroneous result. We have implemented the fault model in C language to confirm the above fact. In this simulation, we have introduced several stuck-at-faults in various locations and found out the erroneous bits to investigate the nature of error propagation depending on the value of $\mathbf{G}$ matrix. We have a detailed discussion about this fault model in this subsection. Now Fig. 8 shows all possible stuck-at-fault locations in parallel CRC.

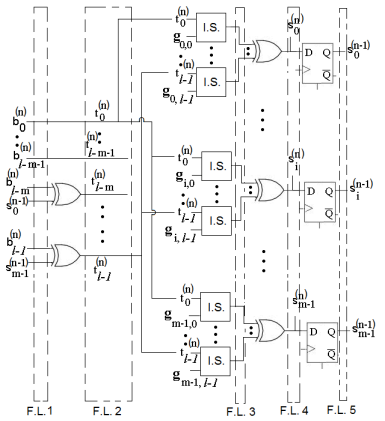

Fig. 8: Fault Locations of Parallel CRC Structure.

In this figure, we have shown that there are mainly five positions for stuck-at-faults. Those four positions have been named F.L. 1, F.L. 2, F.L. 3, F.L. 4 and F.L. 5 in the diagram. Now from Fig. 8, it is evident that any fault in F.L. 4 and F.L. 5 affects only the corresponding syndrome bits, but any single stuck-at-fault in F.L. 1 and F.L. 2 may affect multiple syndrome bits though the manifestation of the fault depends upon the value of the $\mathbf{G}$ matrix. Now a single fault in F.L. 4 implies that the particular syndrome bit is stuck at either 1 or 0. So a single fault in F.L.4 makes only one bit of the syndrome erroneous. Similarly, a single stuck-at-fault in F.L.3 usually affects a single syndrome bit and can be detected by the output flag. It can also correspond to multiple syndrome bit errors. This can be ensured by arguing that resource sharing may not have a significant impact in this case because there is very little scope of resource sharing for two syndrome bits in F.L. 3 zone due to the fact that the matrix $\mathbf{G}$ is in most cases a sparse matrix. The possible locations of the stuck-at fault in the output of the flip-flops have been considered in F.L. 5. An additional design unit, shown in dotted line in Fig. 4, has been added to detect the presence of single stuck-at fault in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

the output of the flip-flops. This additional design unit first calculates the w-bit parity of the output of the flip-flops using the same formulations provided for the Parity Calculation Unit. Then these parities are compared with the actual parities of the previous cycle which are calculated in the Parity Calculation Unit and passed through flip-flops. Any mismatch between the w-bit parity of the Additional Unit and that of the previous cycle Parity Calculation Unit can be flagged using an error signal. Therefore, this error signal detects the presence of any single stuck-at fault in the output of the flip-flops.

Now we discuss the cases of stuck-at-faults in F.L.1 and F.L. 2. Any single stuck-at-fault in F.L. 1 will result in erroneous value of the $T^{(n)}(x)$ in only one bit position i.e., if there is a stuck-at-fault either in $b_i^{(n)}$ and/or $s_i^{(n-1)}$, then only $t_i^{(n)}$ will be erroneous. Also if there are faults in F.L. 1 simultaneously in the locations which are not the inputs of the same XOR gate, then multiple bits in $T^{(n)}(x)$ will be erroneous. Now any single bit error in $T^{(n)}(x)$ or a single stuck-at-fault in F.L. 2 may influence all the syndrome bits depending on the value of the $\mathbf{G}$ matrix. If there is a stuck-at-fault in $t_i^{(n)}$, then $s_j^{(n)}$ will be erroneous only if $g_{j,i} = 1$. This clearly indicates that for each different CRC generator polynomial, fault propagation will be different. First we consider CRC-8-CCITT polynomial $\left(G(x) = x^8 + x^2 + x + 1\right)$ which has been widely used in Asynchronous Transfer Mode Header Error Control/Check (ATM HEC), and then we will consider CRC-8 $\left(G(x) = x^8 + x^4 + x^3 + x^2 + 1\right)$ polynomial which has AES3 (Audio Engineering Society) as its application area. Then we consider some other frequently used polynomials specifically CRC-16 $\left(G(x) = x^{16} + x^{15} + x^2 + 1\right)$ which is used for USB applications. Finally we consider a widely used CRC-32 polynomial used for MPEG, GZIP and PNG.

1) CRC-8-CCITT Polynomial: The generator matrix of CRC-8-CCITT polynomial $\left(G(x) = x^8 + x^2 + x + 1\right)$ is given below.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Now we show in Table IV the erroneous syndrome bits resulted from a particular stuck-at-fault of $T^{(n)}(x)$ in F.L. 2.

TABLE IV: Fault Analysis of CRC-8-CCITT.

Stuck-at-Fault Location	Erroneous Syndrome Bits
$t_0^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_2^{(n)}$
$t_1^{(n)}$	$s_1^{(n)}, s_2^{(n)}, s_3^{(n)}$
$t_2^{(n)}$	$s_2^{(n)}, s_3^{(n)}, s_4^{(n)}$
$t_3^{(n)}$	$s_3^{(n)}, s_4^{(n)}, s_5^{(n)}$
$t_4^{(n)}$	$s_4^{(n)}, s_5^{(n)}, s_6^{(n)}$
$t_5^{(n)}$	$s_5^{(n)}, s_6^{(n)}, s_7^{(n)}$
$t_6^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_2^{(n)}, s_6^{(n)}, s_7^{(n)}$
$t_7^{(n)}$	$s_1^{(n)}, s_3^{(n)}, s_7^{(n)}$
$t_8^{(n)}$	$s_1^{(n)}, s_2^{(n)}, s_4^{(n)}$
$t_9^{(n)}$	$s_1^{(n)}, s_3^{(n)}, s_5^{(n)}$

From Table IV, we see that all the single stuck-at-faults in $T^{(n)}(x)$ affect odd number of bits in syndrome, then all of these single stuck-at-faults of $T^{(n)}(x)$ result in odd number of bit errors which can be detected by a single-bit parity prediction scheme as well as by our proposed multiple-bit parity scheme. But not in all cases we will get to see a situation where affected bit numbers are always odd. In the next subsection, we will discuss a case involving a single stuck-at-fault of F.L. 2 affecting even number of syndrome bits thus making the scenario undetectable by single bit parity prediction.

2) CRC-8 Polynomial: The generator matrix of CRC-8 polynomial $\left(G(x) = x^8 + x^4 + x^3 + x^2 + 1\right)$ is given below.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Now we show in Table V, the erroneous syndrome bits resulted from a particular stuck-at-fault of $T^{(n)}(x)$ in F.L. 2.

TABLE V: Fault Analysis of CRC-8.

Stuck-at-Fault Location	Erroneous Syndrome Bits
$t_0^{(n)}$	$s_0^{(n)}, s_2^{(n)}, s_3^{(n)}, s_4^{(n)}$
$t_1^{(n)}$	$s_1^{(n)}, s_3^{(n)}, s_4^{(n)}, s_5^{(n)}$
$t_2^{(n)}$	$s_2^{(n)}, s_4^{(n)}, s_5^{(n)}, s_6^{(n)}$
$t_3^{(n)}$	$s_3^{(n)}, s_5^{(n)}, s_6^{(n)}, s_7^{(n)}$
$t_4^{(n)}$	$s_0^{(n)}, s_2^{(n)}, s_3^{(n)}, s_6^{(n)}, s_7^{(n)}$
$t_5^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_2^{(n)}, s_7^{(n)}$
$t_6^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_4^{(n)}$
$t_7^{(n)}$	$s_1^{(n)}, s_2^{(n)}, s_5^{(n)}$
$t_8^{(n)}$	$s_2^{(n)}, s_3^{(n)}, s_6^{(n)}$
$t_9^{(n)}$	$s_3^{(n)}, s_4^{(n)}, s_7^{(n)}$

In Table V, we see that for $t_4^{(n)}$, $t_6^{(n)}$, $t_7^{(n)}$, $t_8^{(n)}$, $t_9^{(n)}$, odd number of syndrome bits are affected. But for $t_0^{(n)}$, $t_1^{(n)}$, $t_2^{(n)}$, $t_3^{(n)}$, $t_5^{(n)}$, even number of syndrome bits are affected resulting these faults undetectable by single bit parity prediction. But since the erroneous bits are not consecutive bits, all of these errors can be detected by our proposed multiple-bit parity prediction scheme as shown in Fig. 5(a). using Lemma 1.

3) CRC-16 Polynomial: Considering the generator matrix of CRC-16 polynomial $\left(G(x) = x^{16} + x^{15} + x^2 + 1\right)$, now we show in Table VI, the erroneous syndrome bits resulted from a particular stuck-at-fault of $T^{(n)}(x)$ in F.L. 2.

We investigate the number of erroneous syndrome bits resulted from various particular stuck-at-fault of $T^{(n)}(x)$ in F.L. 2. Following the value of the $\mathbf{G}$ matrix we find that, all the single stuck-at-faults in $T^{(n)}(x)$ affect odd number of bits in syndrome. Thus, all of these single stuck-at-faults of $T^{(n)}(x)$ result in odd number of bit errors which can be detected by a single-bit parity prediction scheme as well as by our proposed scheme.

4) CRC-32 Polynomial: Considering the generator matrix of CRC-32 polynomial

$$(G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} +$$

$$+x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1),$$

we can investigate the number of erroneous syndrome bits resulted from various particular stuck-at-fault of $T^{(n)}(x)$ in F.L. 2 following the value of the $\mathbf{G}$ matrix. We see that for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

TABLE VI: Fault Analysis of CRC-16.

Stuck-at-Fault Location	Erroneous Syndrome Bits
$t_0^{(n)}$	$s_0^{(n)}, s_2^{(n)}, s_{15}^{(n)}$
$t_1^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_2^{(n)}, s_3^{(n)}, s_{15}^{(n)}$
$t_2^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_3^{(n)}, s_4^{(n)}, s_{15}^{(n)}$
$t_3^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_4^{(n)}, s_5^{(n)}, s_{15}^{(n)}$
$t_4^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_5^{(n)}, s_6^{(n)}, s_{15}^{(n)}$
$t_5^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_6^{(n)}, s_7^{(n)}, s_{15}^{(n)}$
$t_6^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_7^{(n)}, s_8^{(n)}, s_{15}^{(n)}$
$t_7^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_8^{(n)}, s_9^{(n)}, s_{15}^{(n)}$
$t_8^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_9^{(n)}, s_{10}^{(n)}, s_{15}^{(n)}$
$t_9^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_{10}^{(n)}, s_{11}^{(n)}, s_{15}^{(n)}$
$t_{10}^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_{11}^{(n)}, s_{12}^{(n)}, s_{15}^{(n)}$
$t_{11}^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_{12}^{(n)}, s_{13}^{(n)}, s_{15}^{(n)}$
$t_{12}^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_{13}^{(n)}, s_{14}^{(n)}, s_{15}^{(n)}$
$t_{13}^{(n)}$	$s_0^{(n)}, s_1^{(n)}, s_{14}^{(n)}$
$t_{14}^{(n)}$	$s_1^{(n)}, s_2^{(n)}, s_{15}^{(n)}$
$t_{15}^{(n)}$	$s_0^{(n)}, s_3^{(n)}, s_{15}^{(n)}$

$t_0^{(n)}$, $t_1^{(n)}$, $t_2^{(n)}$, $t_3^{(n)}$, $t_4^{(n)}$, $t_5^{(n)}$, $t_{16}^{(n)}$, $t_{17}^{(n)}$, $t_{18}^{(n)}$, $t_{24}^{(n)}$, $t_{25}^{(n)}$, even number of syndrome bits are affected resulting these faults undetectable by single bit parity prediction. But since the erroneous bits are not consecutive bits, all of these errors can be detected by our proposed multiple-bit parity prediction scheme.

The above discussion shows the relation between a stuck-at-fault and resulting location of erroneous bits depending on presence of 1 or 0 in rows and columns of $\mathbf{G}$ matrix. So if we find a bit erroneous, we can simply backtrack using $\mathbf{G}$ matrix to approximately locate a possible region having stuck-at fault. In this way our proposed scheme also helps to find out an approximate position of the faulty portion of the circuit. Further study in this direction may lead to online error correction of CRC.

D. Fault Simulation

Now we present the results of the fault injection simulation. In this simulation, first faults are injected into the parallel CRC circuit. Then the resulting number of error occurance is computed. Finally we calculate the number of errors detected by our proposed scheme and tabulate the results. We have coded this simulation using C language and used the generator polynomials mentioned before. For CRC-8, CRC-16 and CRC-32, we have introduced all possible single stuck-at-faults for investigation of error propagation. For this purpose we have incorporated the fault model consisting of F.L. 1, F.L. 2, F.L. 3, F.L. 4 and F.L. 5 discussed in the previous subsection. A detail investigation reveals that the total number of possible single stuck-at-fault in F.L. 2, F.L. 3, F.L. 4 and F.L. 5 are respectively l, $(m \times l)$, $m \times (l-1)$ and m. Table VII gives a detail result of all the possible single stuck-at fault for F.L. 2, F.L. 3, F.L. 4 and F.L. 5, and also provides the summation of all these results.

The table clearly proves the high error detection capability of the proposed scheme.

V. Complexity Overhead and Implementation Results

In this section, first we discuss the theoretical complexity of the presented error detection method using tables and figures. Next using these area overhead and error detection capability, we come to a trade-off of these two parameters to calculate the particular value of number of parity bits to achieve high error coverage without having high area overhead for the proposed multiple-bit parity-based concurrent fault detection of parallel CRC structure. We also present the implementation result of the proposed scheme.

A complexity analysis between double modular redundancy and the proposed scheme is presented in Table III. While the parallel CRC implementation based on matrix multiplication employs $m.l$ number of XOR gates, the required number of XOR gates for the proposed multiple-bit parity-based concurrent fault detection of CRC structure is $m.l + w.(l+m)$, thus resulting in the overhead of $\frac{w.(l+m)}{m.l}$. If we consider the case when $m = l$, the overhead becomes $\frac{2.w}{m}$ which is 25% for the operating value of w explained later in this section. The critical path delay comparison has also been summarised in Table III, where T_X represents the delay of a two input XOR gate. The correctness of the proposed scheme has been verified by Verilog coding which also proves the validity of the given overhead formulation. In Verilog, we have coded the proposed multiple-bit parity scheme and validated the functionality of the proposed architecture. We have added an additional design unit to detect the fault in the output of the flip-flops. The area overhead of this Additional Unit can be expressed as $w(r-1)+w$ number of XOR gates, w number of flip-flops and the delay can be expressed as $(1 + \lceil log_2 r \rceil)\ T_X$. Since this delay is much less compared to the critical path delay of the Parity Prediction Unit, the inclusion of this Additional Unit does not have significant impact on timing overhead of the overall architecture. We have also implemented this Additional Unit using Verilog code.

We have selected three values of w, i.e. 2, 3 and 4 for $m = 32$, as the corresponding error covrerages yield good results presented in the previous section and the area overheads are depicted in Table VIII. Similarly, we have implemented the proposed scheme for $m = 16$ and the area overhead for $w = 2$ is shown in Table VIII. We have given the area overhead for the proposed architecture in two separate cases, without the Additional Unit and with the Additional Unit. This Additional Unit is very similar to the Parity Calculation Unit in terms of architecture. So if we add the Additional Unit but at the same time remove the Parity Calculation Unit, then we will not have this extra area overhead due to the Additional Unit.

From previous discussions, we see that there is a trade-off between fault coverage in terms of number of detectable errors and area overhead since increase in number of parity bits (w) increases both the fault coverage and area overhead.

For three different values of m namely $m = 64$, $m = 32$ and $m = 16$, we compare area overhead (in terms of extra gates) and fault coverage (in terms of number of detectable errors) for 2-bit errors for several values of number of parity bits (w). We have chosen 2-bit error because among all the even number of errors undetectable by single bit parity scheme, this is the most common. For all these calculations, we have considered three different values for l covering all three cases namely $l > m$ represented by $l = 2m$, $l = m$ and $l < m$ represented by $l = m/2$. This analysis helps us to come into the conclusion that the value of w for which we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

TABLE VII: Error Detection Capability of the proposed scheme for single-bit stuck-at fault for all fault locations.

Generator Polynomial	Number of Parity Bits	Stuck-at-fault in F.L. 2			Stuck-at-fault in F.L. 3			Stuck-at-fault in F.L. 4			Stuck-at-fault in F.L. 5			Total Number of Stuck-at-fault		
		Errors Occured	Errors Detected	Percentage	Errors Occured	Errors Detected	Percentage	Errors Occured	Errors Detected	Percentage	Errors Occured	Errors Detected	Percentage	Errors Occured	Errors Detected	Percentage
CRC-8-CCITT	2	8	8	100	80	80	100	72	72	100	8	8	100	168	168	100
	3	8	8	100	80	80	100	72	72	100	8	8	100	168	168	100
CRC-8	2	8	7	87.5	80	80	100	72	72	100	8	8	100	168	167	99.405
	3	8	8	100	80	80	100	72	72	100	8	8	100	168	168	100
CRC-16	2	16	16	100	160	160	100	144	144	100	16	16	100	336	336	100
	3	16	16	100	160	160	100	144	144	100	16	16	100	336	336	100
	4	16	16	100	160	160	100	144	144	100	16	16	100	336	336	100
CRC-32	2	32	24	75	320	320	100	288	288	100	32	32	100	672	664	98.81
	3	32	29	90.6	320	320	100	288	288	100	32	32	100	672	669	99.553
	4	32	31	96.875	320	320	100	288	288	100	32	32	100	672	671	99.851
	5	32	31	96.875	320	320	100	288	288	100	32	32	100	672	671	99.851

TABLE VIII: Complexity Overhead of the Proposed Scheme from ASIC Implementation Results

Generator Polynomial	Number of Parity Bits	Area					Time		
		Original Architecture (μm^2)	Fault Detection Architecture without Additional Unit (μm^2)	Overhead Percentage	Fault Detection Architecture with Additional Unit (μm^2)	Overhead Percentage	Original Architecture(ns)	Fault Detection Architecture (ns)	Overhead Percentage
CRC-16	2	16811	21249	26.4	22072	31.3	2.93	3.37	15.2
CRC-32	2	72917	82031	12.5	84364	15.7	4.23	4.76	12.7
	3	72917	87135	19.5	88885	21.9	4.23	4.83	14.3
	4	72917	91656	25.7	93407	28.1	4.23	4.89	15.6

get a high value of fault coverage while the area overhead is not very high, i.e. the operating value of w can be safely considered as $w = \frac{m}{8}$ for $l = m$. In short, if we use this operating value of w, we will achieve good fault coverage and at the same time, area overhead of the design will not be too high. Table II shows that fault coverage is independent of l while Table III shows that area overhead is linearly dependent on l which gets confirmed from our calculation. For $l = m$, the operating value of w is $w = \frac{m}{8}$. If $l > m$, the operating value of w is found to be reduced from $w = \frac{m}{8}$ to take care of linearly increasing area overhead. So, for $l > m$, operating value of w is less than $\frac{m}{8}$. Similarly, for $l < m$, operating value of w is found to be safely increased from $w = \frac{m}{8}$.

VI. Conclusions

In this paper, we have proposed a concurrent fault detection design structure for parallel CRC architecture. Two different cases, namely $l \geq m$ and $l < m$ have been considered and the formulation for the multiple-bit parity-based concurrent fault detection scheme has been derived followed by the implementation of the corresponding architecture for each of them.

The error and fault coverage of the proposed scheme have been analyzed in detail using error and fault models. We have also presented a formulation for the error detection capability of the scheme covering all the possible $2^m - 1$ cases where the output can become erroneous. Moreover, a thorough analysis of all the possible single stuck-at fault locations have been discussed. Furthermore, we have investigated the relation between stuck-at fault location and error propagation resulting in an erroneous output and the relation between this propagation with the CRC generator polynomial. Finally, the fault detection capability and the theoretical time and area overheads of the proposed schemes have been reported and confirmed by software simulation and ASIC implementation results. We have coded the proposed scheme in C for software simulation and in Verilog for ASIC implementation purpose. The proposed method is shown to be area efficient and at the same time it has a high fault detection capability. The proposed multiple-bit parity prediction scheme has also been shown to have not only the ability to detect a fault, but also to point out the region of the fault. This potential advantage can further be utilized while extending the proposed scheme to correct the error online in future, i.e., to achieve an error free output even in the presence of a hardware fault.

Acknowledgements

The authors would like to thank the reviewers for their valuable comments. This work has been supported by NSERC Discovery Accelerator Supplementary Grants awarded to Arash Reyhani-Masoleh. The authors also thank CMC for the CAD tools.

References

[1] W. Peterson and D. Brown, "Cyclic codes for error detection," *in Proc. IRE*, vol. 49, no. 1, pp. 228–235, 1961.
[2] ATM Layer Specification, ITU-T Recommendation I.361, 1999.
[3] IEEE Standard for Information Technology: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. ANSI/IEEE Std 802.3-2005.
[4] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. ANSI/IEEE Std 802.11-1999.
[5] IEEE Standard for Local and Metropolitan Area Networks: Air Interface for Fixed and Mobile Broadband WirelessAccess Systems. ANSI/IEEE Std. 802.16-2004.
[6] C. Kennedy, "High Performance Hardware and Software Implementations of the Cyclic Redundancy Check Computation," *Thesis, University of Western Ontario,* 2009.
[7] S. Lin and D.J. Costello, *Error Control Coding*, Englewood Cliffs, NJ: Prentice-Hall, 1983.
[8] T. Ramabadran and S. Gaitonde, "A tutorial on CRC computations," *IEEE Micro*, vol. 8, no. 4, pp. 62–75, Aug. 1988.
[9] T.-B. Pei and C. Zukowski, "High-speed parallel CRC circuits in VLSI," *IEEE Trans. on Communications,* vol. 40, no. 4, pp. 653–657, Apr. 1992.
[10] G. Albertengo and R. Sisto, "Parallel CRC generation," *IEEE Micro,* vol. 10, no. 5, pp. 63–71, Oct. 1990.
[11] G. Campobello, G. Patane, and M. Russo, "Parallel CRC realization," *IEEE Trans. on Computers,* vol. 52, no. 10, pp. 1312–1319, Oct. 2003.
[12] M.-D. Shieh, M.-H. Sheu, C.-H. Chen, and H.-F. Lo, "A systematic approach for parallel CRC computations," *Journal of Information Science and Eng.,* vol. 17, no. 3, pp. 445–461, 2001.
[13] M. Sprachmann, "Automatic generation of parallel CRC circuits," *IEEE Design and Test of Computers,* vol. 18, no. 3, pp. 108–114, 2001.
[14] C. Cheng and K.K. Parhi, "High-speed parallel CRC implementation based on unfolding, pipelining, and retiming," *IEEE Trans. on Circuits and Syst. II, Express Briefs*, vol. 53, no. 10, pp. 1017–1021, Oct. 2006.
[15] C. Cheng, K.K. Parhi, "High Speed VLSI Architecture for General Linear Feedback Shift Register (LFSR) Structures," *IEEE Conference,* 2009.
[16] M. Ayinala and K.K. Parhi, "High-Speed Parallel Architectures for Linear Feedback Shift Registers," *IEEE Trans. on* Signal Processing, vol. 59, no. 9, pp. 4459-4469, Sep. 2011.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2479617, IEEE Transactions on Computers

[17] C. Kennedy and A. Reyhani-Masoleh, "High-speed parallel CRC circuits," *in Proc. 42nd Asilomar Conf. Signals, Systems and Computers,* pp. 1823–1829, Oct. 2008.

[18] M. Braun, J. Friedrich, T. Grün, and J. Lembert, "Parallel CRC computation in FPGAs," *Lecture Notes in Computer Science, Springer-Verlag,* vol. 1142, pp. 156–165, 1996..

[19] R. Glaise, "A Two-Step Computation of Cyclic Redundancy Code CRC-32 for ATM Networks," *IBM Journal of Research and Development,* vol.41, no.6, pp. 705-709, 1997.

[20] J. Derby, "High-Speed CRC Computation Using State-Space Transformations," *IEEE Global Telecommunications Conference (GLOBECOM),* vol. 1, pp. 166-170*,* 2001.

[21] P. Reviriego, S. Pontarelli and J.A. Maestro, "Concurrent Error Detection for Orthogonal Latin Squares Encoders and Syndrome Computation," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems,* vol. 21, no 12, pp. 2334-2338, December 2013.

[22] K. Namba and F. Lombardi, "A novel scheme for concurrent error detection of OLS parallel decoders," *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2013 IEEE International Symposium on. IEEE, 2013.

[23] N.A. Touba and E.J. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.16, no. 7, pp. 783-789, 1997.

[24] K. De, C. Natarajan, D. Nair and P. Banerjee, "RSYN: A system for automated synthesis of reliable multilevel circuits," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol.2, no.2, pp. 186-195, 1994.

[25] M. Walma, "Pipelined Cyclic Redundancy Check (CRC) Calculation," *in Proc. International Conference on Computer Communications and Networks, (ICCCN),* pp. 365-370, Aug. 2007.

[26] E. Normand, "Single event upset at ground level," *IEEE Trans. on Nuclear Science, vol.* 43, no. 6, pp. 2742–2750, Dec. 1996.

[27] D. Larner, "Sun flips bits in chips," *Electronics Times*, no. 878, pp. 72 and 24, 10 Nov. 1997.

[28] J.F. Ziegler, "Terrestrial cosmic rays intensities," *IBM Journal of Research and Development,* vol. 42, pp. 117–139,1998.

[29] C.R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. on Device and Materials Reliability,* vol. 5, no. 3, pp. 305–316, Sep. 2005.

[30] B.W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley Publishing Company, 1989.

[31] S. J. Piestrak, A. Dandache and F. Monteiro, "Designing Fault-Secure Parallel Encoders for Systematic Linear Error Correcting Codes," *IEEE Transactions on Reliability,* vol. 52, no. 4, Dec. 2003.

[32] K. K. Parhi, "Eliminating the fanout bottleneck in parallel long BCH encoders," *IEEE Trans. on Circuits and Systems I, Reg. Papers,* vol. 51, no. 3, pp.512–516, Mar. 2004.

[33] X. Zhang and K.K. Parhi, "High-speed architectures for parallel long BCH encoders," *in Proc. ACM Great Lakes Symp. on VLSI, Boston, MA,* pp. 1–6, Apr. 2004.

[34] H. Jaber, F. Monteiro, S.J. Piestrak, A. Dandache, "Design of parallel fault-secure encoders for systematic cyclic block transmission codes," *Microelectronics Journal,* vol. 40, no. 12, pp. 1686-1697, Dec. 2009.

[35] S. Pontarelli, G.C. Cardarilli, M. Re and A. Salsano, "A Novel Error Detection And Correction Technique for RNS based FIR Filters," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '08),* pp. 436-444, Oct. 2008.

[36] G.C. Cardarilli, S. Pontarelli, M.Re and A. Salsano, "A Self Checking Reed Solomon Encoder: Design and Analysis," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '05),* pp. 111-119, Oct. 2005.

[37] G.C. Cardarilli, S. Pontarelli, M. Re and A. Salsano, "Concurrent Error Detection in Reed–Solomon Encoders and Decoders," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems,* vol. 15, no. 7, pp. 842-846, July 2007.

[38] S. Bayat-Sarmadi and M.A. Hasan, "Concurrent Error Detection of Polynomial Basis Multiplication over Extension Fields using a Multiple-bit Parity Scheme," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '05),* pp. 102-110, Oct. 2005.

[39] D.C. Feldmeier, "Fast Software Implementation of Error Detection Codes," *IEEE/ACM Trans. on Networking*, vol.3, no. 6, pp. 640-651, Dec. 1995.

Dipanwita Gangopadhyay received the BE. degree in electrical and computer engineering from Jadavpur University, Kolkata, India, in 2004, and the M.S. degree in electrical and computer engineering from Indian Institute of Technology (IIT) Madras, Chennai, India, in 2007. She was an Engineer with Qualcomm India Private Ltd, Bangalore, India, from 2007 to 2009. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Western University, London, Ontario.

Arash Reyhani-Masoleh received the BSc degree in electrical and electronic engineering from Iran University of Science and Technology in 1989, the MSc degree in electrical and electronic engineering from the University of Tehran in 1991, both with the first rank, and the PhD degree in electrical and computer engineering from the University of Waterloo in 2001. From 1991 to 1997, he was with the Department of Electrical Engineering, Iran University of Science and Technology. From June 2001 to September 2004, he was with the Center for Applied Cryptographic Research, University of Waterloo, where he was awarded a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship in 2002. In October 2004, he joined the Department of Electrical and Computer Engineering, Western University, London, Canada, where he is currently a tenured associate professor. His current research interests include fault-tolerant computing, algorithms and VLSI architectures for computations in finite fields, cryptography, and error-control coding. He has been a two-time recipient of NSERC Discovery Accelerator Supplement (DAS) award in 2010 and 2015. Currently, he serves as an associate editor for Integration, the VLSI Journal (Elsevier). He is a member of the IEEE and the IEEE Computer Society.