High-Speed CRC Computations Using Improved State-Space Transformations

Christopher Kennedy and Arash Reyhani-Masoleh
Department of Electrical and Computer Engineering
The University of Western Ontario, London, ON N6A 5B9 Canada
Email: christopher.kennedy@ieee.org and areyhani@eng.uwo.ca

***Abstract*—Previously, a state-space similarity transform was proposed to reduce the feedback loop complexity of a parallel Cyclic Redundancy Check architecture and enable retiming. This paper investigates the open research question concerning the impact of varying the vector used to construct the transformation matrix. We perform exhaustive searches of the vector space for frequently referenced generator polynomials when the input size is equal to the degree of the generator polynomial. The set of vectors which yield minimal hardware state-space representations is obtained. Then, application-specific integrated circuit (ASIC) experiments are performed. The ASIC implementation results for the minimized state spaces demonstrate improvement in both area and timing as compared to the original ones. Finally, it is concluded that the vectors obtained for a fixed generator polynomial are also good choices for other input sizes.**

***Index Terms*—Cyclic Redundancy Check (CRC), state-space similarity transformation, optimization, application-specific integrated circuit (ASIC).**

I. Introduction

THE Cyclic Redundancy Check (CRC) is an error detection code that was first introduced in 1961 by Peterson and Brown in their landmark paper [1]. CRC is now used in various digital transmission and storage systems as the primary means of error detection. Examples of digital communication standards that employ the CRC include Asynchronous Transfer Mode (ATM), Ethernet (IEEE 802.3), WiFi (IEEE 802.11), and WiMAX (IEEE 802.16).

In [1], it is shown that a linear feedback shift register (LFSR) can be used to perform the CRC computation serially, and the general form, drawn as a delay diagram, is shown in Figure 1 [2]. It is constructed for the selected generator polynomial $G(z) = 1 + \sum_{i=0}^{n-1} g_i z^i + z^n$. In terms of computation time, the CRC LFSR architecture requires K cycles, where K is the length of the message in bits. Finally, it has a critical path delay of $2 \cdot T_X$, where T_X denotes the delay of a two-input XOR gate[1].

After the serial CRC LFSR architecture was proposed in [1], research into obtaining parallel CRC hardware architectures and software algorithms has received a considerable amount of attention in the literature. These approaches process the message in blocks of r, $r > 1$, bits at a time, see for example [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], and [16] for some of the hardware developments. Most of the hardware approaches suffer from two problems: they are not easily retimed and as the input size is increased they experience diminishing returns. That is, there are delays in the feedback loop [2], and an $\frac{r}{2}$ speed-up limit compared to the serial architecture is usually observed for a parallel architecture that processes r message bits in parallel [6].

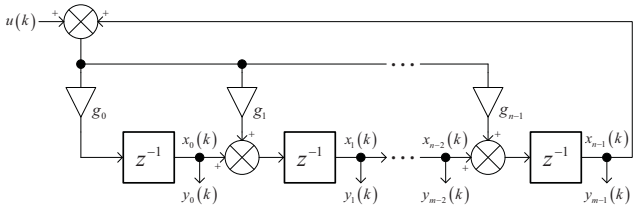

Figure 1: Delay diagram of the CRC LFSR architecture [2].

In [2], it is shown that the serial and parallel CRC computation architectures can be modeled with a state-space representation. Afterward, a state-space similarity transformation is proposed to reduce the feedback loop complexity of a parallel architecture to that of its serial counterpart. This allows for a potential speed-up of r-times over the serial architecture when the logic in the input and output coupling matrices is retimed.

The state-space transformation matrix $\mathbf{T}_{n\times n}$ is constructed by selecting a vector $\mathbf{b}^*_{n\times 1}$ (denoted by $\mathbf{b}_1$ in [2]). For the example in [2], $\mathbf{b}^*_{32\times 1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}^T$ is selected for simplicity. But generally, the vector $\mathbf{b}^*_{n\times 1}$ must be chosen such that $\mathbf{T}_{n\times n}$ is non-singular [2]. Note that all $\mathbf{b}^*_{n\times 1}$s that construct non-singular $\mathbf{T}_{n\times n}$ will yield a transformed system with a state coupling matrix in companion form.

In this paper, we investigate the effect of varying the vector $\mathbf{b}^*_{n\times 1}$ on the hardware complexity of the resultant transformed system. We perform exhaustive searches to find the set of $\mathbf{b}^*_{n\times 1}$ vectors that minimize the hardware complexity for frequently referenced generator polynomials when the input size is equal to the generator polynomial degree. Afterward, application-specific integrated circuit (ASIC) implementations are carried out to compare improved and original systems. The ASIC results verify the predicted reduction in area and also demonstrate improvements in timing. Finally, we conclude that the $\mathbf{b}^*_{n\times 1}$ vectors obtained are good choices for other input sizes as well.

The remainder of this paper is organized as follows. In Section II, a brief summary of the CRC preliminaries is given. In Section III, we summarize the state-space transformation approach presented in [2]. In Section IV, the vector $\mathbf{b}^*_{n\times 1}$ search methodology is described. In Section V, the results are presented. Finally, the paper is concluded in Section VI.

[1]In this paper, we assume that all XOR gates have two inputs.

978-1-4244-3355-1/09/$25.00©2009 IEEE

II. CRC Preliminaries

In this section, we provide some of the background and review the related papers. The CRC of a message, denoted by the polynomial $S(z)$, is an n-bit checksum formed from the remainder of the binary polynomial division of an augmented K-bit message polynomial $U(z) = \sum_{i=0}^{K-1} u_i z^i$ and generator polynomial $G(z)$,

$$S(z) = \mathtt{crc}(U(z)) = (x^n \cdot U(z)) \bmod G(z). \quad (1)$$

The generator polynomial dictates the effectiveness of the frame check sequence (FCS) produced by the CRC computation. All generator polynomials are of the form $G(z) = 1 + \sum_{i=1}^{n-1} g_i z^i + z^n$, where n is the degree of the polynomial [1]. Some frequently referenced generator polynomials [14] are listed in Table I.

In [2], some of the existing parallel CRC architectures, e.g., [3], [5], [6], and [8], have been discussed and shown that they are not good candidates for retiming. We agree with this analysis excluding [8]. In [8], a two-step reduction process is proposed based on the parallel formulation of [7]. In this approach, a multiple of the generator polynomial is first used to reduce the message, then this intermediate result is reduced once by the desired generator polynomial to obtain the final CRC. The best achievable critical path delay of the first stage is T_X, and since there is no feedback in the second stage, it can be retimed with delay T_X. Therefore, this architecture is faster than the $2 \cdot T_X$ retimed result of [2]. However, the difficulty lies in finding suitable multiple polynomials for large generator polynomials and input sizes. Moreover, the area complexity is quite large in terms of the number of FFs and XOR gates.

Some new parallel CRC architectures have been proposed, in [9], [10], [11], [12], [13], [14], [15], and [16]. In [9], a look-ahead technique is used to parallelize the computation. In [10], it is shown how the LFSR combinational logic can be cascaded. In [11], a retimed CRC architecture is proposed, however the minimum critical path delay depends on the complexity of feedback loop. Later, a different approach is taken in [12], where the feedback loop is retimed. In [13], a parallel architecture is proposed based on state-space solution techniques. Unfolding, pipelining, and retiming techniques are applied to the serial LFSR architecture in [14]. In [15], a retimed architecture is proposed with T_X delay. Finally, in [16], a parallel architecture is proposed based on binary polynomial manipulations, for the case when the input size is greater than the generator polynomial degree.

The approaches of [9], [10], [13], and [16], all have complexity in their feedback loops. Consequently, they are not suitable candidates for retiming. The retimed architecture proposed in [11], suffers from the same problem of the previous approaches with its critical path residing in the feedback loop. In [12], one must assume that there are buffered packets, to ensure good throughput. In [14], the proposed technique achieves good timing results, however the performance is dictated by the coefficients of the generator polynomial. In [15], the message length must be known beforehand, which may not be practical for all cases. Therefore, at this point, we feel that [2] still is the most powerful, flexible, and systematic approach to obtain fast retimed CRC architectures.

Table I: Frequently referenced generator polynomials.

Name	Polynomial
CRC-12	$1 + z + z^2 + z^3 + z^{11} + z^{12}$
CRC-16	$1 + z^2 + z^{15} + z^{16}$
CCITT-16	$1 + z^5 + z^{12} + z^{16}$
CRC-16†	$1 + z + z^{14} + z^{16}$
CCITT-16†	$1 + z^4 + z^{11} + z^{16}$
CRC-32	$1 + z + z^2 + z^4 + z^5 + z^7 + z^8 + z^{10}$ $z^{11} + z^{12} + z^{16} + z^{22} + z^{23} + z^{26} + z^{32}$

† denotes reversed polynomial coefficients.

III. Related Work

In this section, we briefly describe the related work contained in [2]. In our summary, we use some slightly different notations than what is used in that paper. To distinguish between the coupling matrices for the serial and parallel architectures, we mark the parallel matrices with bars, and use the letters r, n, and m to denote the bus widths of the input, state, and output vectors, respectively.

A. CRC State-Space Model

In [2], the author shows that the serial CRC LFSR architecture illustrated in Figure 1 can be described by the following state-space equations

$$\begin{aligned} \mathbf{x}_{n\times 1}(k+1) &= \mathbf{A}_{n\times n} \cdot \mathbf{x}_{n\times 1}(k) + \mathbf{b}_{n\times 1} \cdot u(k) \\ \mathbf{y}_{m\times 1}(k) &= \mathbf{C}_{m\times n} \cdot \mathbf{x}_{n\times 1}(k) + \mathbf{d}_{m\times 1} \cdot u(k), \end{aligned} \quad (2)$$

for $0 \le k \le K-1$, where $\mathbf{A}_{n\times n} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & g_0 \\ 1 & 0 & 0 & & 0 & g_1 \\ 0 & 1 & 0 & & 0 & g_2 \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & g_{n-1} \end{bmatrix}$, $\mathbf{b}_{n\times 1} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{n-1} \end{bmatrix}$, $\mathbf{C}_{m\times n} = \mathbf{I}$, $\mathbf{d}_{m\times 1} = \mathbf{0}$, and $u(k) = u_{K-1-k} \in \{0,1\}$ from (1). Then, the system described by (2) is extended to a parallel input system that processes r message bits per iteration, that is,

$$\begin{aligned} \bar{\mathbf{x}}_{n\times 1}(k+1) &= \bar{\mathbf{A}}_{n\times n} \cdot \bar{\mathbf{x}}_{n\times 1}(k) + \bar{\mathbf{B}}_{n\times r} \cdot \bar{\mathbf{u}}_{r\times 1}(k) \\ \bar{\mathbf{y}}_{m\times 1}(k) &= \bar{\mathbf{C}}_{m\times n} \cdot \bar{\mathbf{x}}_{n\times 1}(k) + \bar{\mathbf{D}}_{m\times r} \cdot \bar{\mathbf{u}}_{r\times 1}(k), \end{aligned} \quad (3)$$

for $0 \le k \le \lceil \frac{K-1}{r} \rceil$, where $\bar{\mathbf{A}}_{n\times n} = (\mathbf{A}_{n\times n})^r$, $\bar{\mathbf{B}}_{n\times r} = \begin{bmatrix} \mathbf{b}_{n\times 1} & \mathbf{A}_{n\times n} \cdot \mathbf{b}_{n\times 1} & \cdots & (\mathbf{A}_{n\times n})^{r-1} \cdot \mathbf{b}_{n\times 1} \end{bmatrix}$, $\bar{\mathbf{C}}_{m\times n} = \mathbf{C}_{m\times n} = \mathbf{I}$, $\bar{\mathbf{D}}_{m\times r} = \mathbf{0}$, and $\bar{\mathbf{u}}_{r\times 1}(k) = \begin{bmatrix} u(k\cdot r + r - 1) & u(k\cdot r + r - 2) & \cdots & u(k\cdot r) \end{bmatrix}^T$. Note that for both (2) and (3), $n = m$ as well as $\mathbf{y}_{m\times 1}(k) = \mathbf{x}_{n\times 1}(k)$ and $\mathbf{y}_{m\times 1}(k) = \bar{\mathbf{x}}_{n\times 1}(k)$, i.e., there is one flip-flop (FF) per CRC bit and the outputs have direct connections to the state FFs.

Similar to [2], we are assuming that the message length K is a multiple of the input size r, or one has the ability to prepend an appropriate number of zeros to make the message length a multiple of r. Methods to handle situations where this is not the case have been extensively discussed and we consider them to be outside the scope of this work. For further information, the reader can consult [17], where the state-space formulation is extended to handle this case.

B. State-Space Similarity Transformation

Beginning with the parallel input state-space representation in (3), consider a similarity transformation

$$\bar{\mathbf{x}}_{n\times 1}(k) = \mathbf{T}_{n\times n} \cdot \bar{\mathbf{x}}'_{n\times 1}(k),$$

which takes the matrix $\bar{\mathbf{A}}_{n\times n}$ to companion form. Then, the transformed state-space equations can be written as

$$\begin{aligned} \bar{\mathbf{x}}'_{n\times 1}(k+1) &= \bar{\mathbf{A}}'_{n\times n} \cdot \bar{\mathbf{x}}'_{n\times 1}(k) + \bar{\mathbf{B}}'_{n\times r} \cdot \bar{\mathbf{u}}_{r\times 1}(k) \\ \bar{\mathbf{y}}_{m\times 1}(k) &= \bar{\mathbf{C}}'_{m\times n} \cdot \bar{\mathbf{x}}'_{n\times 1}(k) + \bar{\mathbf{D}}_{m\times r} \cdot \bar{\mathbf{u}}_{r\times 1}(k), \end{aligned} \tag{4}$$

for $0 \leq k \leq \left\lceil \frac{K-1}{r} \right\rceil$, where $\bar{\mathbf{A}}'_{n\times n} = \mathbf{T}^{-1}_{n\times n} \cdot \bar{\mathbf{A}}_{n\times n} \cdot \mathbf{T}_{n\times n}$, $\bar{\mathbf{B}}'_{n\times r} = \mathbf{T}^{-1}_{n\times n} \cdot \bar{\mathbf{B}}_{n\times r}$, and $\bar{\mathbf{C}}'_{m\times n} = \bar{\mathbf{C}}_{m\times n} \cdot \mathbf{T}_{n\times n} = \mathbf{T}_{n\times n}$. In [2], it is shown that the matrix $\mathbf{T}_{n\times n}$ which transforms $\bar{\mathbf{A}}_{n\times n}$ to companion form $\bar{\mathbf{A}}'_{n\times n}$ in (4), can be found as

$$\mathbf{T}_{n\times n} = \left[\begin{array}{cccc} \mathbf{b}^*_{n\times 1} & \bar{\mathbf{A}}_{n\times n} \cdot \mathbf{b}^*_{n\times 1} & \cdots & \left(\bar{\mathbf{A}}_{n\times n}\right)^{n-1} \cdot \mathbf{b}^*_{n\times 1} \end{array} \right], \tag{5}$$

and $\mathbf{b}^*_{n\times 1}$ can be selected arbitrarily provided that the columns of $\mathbf{T}_{n\times n}$ are linearly independent, i.e., $\mathbf{T}_{n\times n}$ is invertible. Furthermore, it is noted that if the generator polynomial is irreducible, then any $\mathbf{b}^*_{n\times 1}$ can be chosen.

In [2], it mentioned that the choice of $\mathbf{b}^*_{n\times 1}$ dictates the new state-space representation that will be formed by the transformation. Intuitively, there will exist state-space representations which result in realizations with different hardware complexities. In the following section, we determine the set of $\mathbf{b}^*_{n\times 1}$ vectors which result in realizations with minimal hardware complexity.

IV. Methodology

To reduce the hardware complexity of parallel retimed CRC computation architectures, we perform an exhaustive search of the $\mathbf{b}^*_{n\times 1}$ vector space for the frequently referenced generator polynomials when the input size is equal to the generator polynomial degree, i.e., $r = n$. Based on (4), it follows that the hardware complexity depends on the number of 1s in the coupling matrices. Consequently, we are interested in finding the $\mathbf{b}^*_{n\times 1}$ vectors that minimize the total number of 1s in the transformed coupling matrices.

A. Search

We have written a C++ program that precomputes the coupling matrices $\bar{\mathbf{A}}_{n\times n}$ and $\bar{\mathbf{B}}_{n\times r}$ in (3), for each of the generator polynomials in Table I with $r = n$. Then, after the precomputation stage, we loop over all the $2^n - 2$ possibilities[2] for the different $\mathbf{b}^*_{n\times 1}$ vectors. For each $\mathbf{b}^*_{n\times 1}$ candidate, the matrix $\mathbf{T}_{n\times n}$ is computed using (5) and then inverted using Gauss-Jordan elimination, i.e.,

$$\left[\begin{array}{cc} \mathbf{T}_{n\times n} & \mathbf{I}_{n\times n} \end{array} \right] \sim \left[\begin{array}{cc} \mathbf{I}_{n\times n} & \mathbf{T}^{-1}_{n\times n} \end{array} \right],$$

to obtain $\mathbf{T}^{-1}_{n\times n}$. Next, we verify that $\mathbf{T}_{n\times n} \cdot \mathbf{T}^{-1}_{n\times n} = \mathbf{I}$, and afterward, the transformed coupling matrices $\bar{\mathbf{A}}'_{n\times n}$, $\bar{\mathbf{B}}'_{n\times r}$, and $\bar{\mathbf{C}}'_{m\times n}$ in (4) are obtained. We then proceed to count the number of 1s in $\bar{\mathbf{A}}'_{n\times n}$, $\bar{\mathbf{B}}'_{n\times r}$, and $\bar{\mathbf{C}}'_{m\times n}$, and the set of $\mathbf{b}^*_{n\times 1}$ vectors that produce minimum values is retained.

[2]The all zero vector $\mathbf{b}^*_{n\times 1} = \left[\begin{array}{cccc} 0 & 0 & \cdots & 0 \end{array} \right]^T$ is not a valid selection.

Table II: The found $\hat{\mathbf{b}}^*_{n\times 1}$ and number of 1s in the coupling matrices for frequently referenced generator polynomials when the input size is equal to the generator polynomial degree.

$G(z)$	$\hat{\mathbf{b}}^*_{n\times 1}$	1
CRC-12	$[\texttt{0x814}]^T$	120
CRC-16	$[\texttt{0xC00D}]^T$	188
CCITT-16	$[\texttt{0x648B}]^T$, * $[\texttt{0x908C}]^T$, $[\texttt{0xC916}]^T$, * $[\texttt{0xF664}]^T$	226
CRC-16†	$[\texttt{0x00E0}]^T$, * $[\texttt{0x7401}]^T$	190
CCITT-16†	$[\texttt{0x390D}]^T$, $[\texttt{0x721A}]^T$, * $[\texttt{0xAC1F}]^T$	226
CRC-32	$[\texttt{0xD840 5018}]^T$	928

We note two optimizations to reduce the time-complexity of the brute force search computation. First, since it is known that if $r = n$, then $\bar{\mathbf{A}}_{n\times n} = \bar{\mathbf{B}}_{n\times(n)}$ [2], thus one is able to first compute $\bar{\mathbf{B}}'_{n\times n} = \mathbf{T}^{-1}_{n\times n} \cdot \bar{\mathbf{B}}_{n\times n}$, and then compute $\bar{\mathbf{A}}'_{n\times n} = \bar{\mathbf{B}}'_{n\times n} \cdot \mathbf{T}_{n\times n}$. By taking this approach, one matrix multiplication operation is removed from the $\mathbf{b}^*_{n\times 1}$ simulation loop. Secondly, if the generation polynomial is irreducible, then all $\mathbf{b}^*_{n\times 1}$ vectors (excluding $\mathbf{0}_{n\times 1}$) will yield a matrix $\mathbf{T}_{n\times n}$ that is non-singular [2], therefore one need not check that $\mathbf{T}_{n\times n} \cdot \mathbf{T}^{-1}_{n\times n} = \mathbf{I}$. For the generator polynomials in Table I, CRC-32 is the only one that is non-reducible, and $\mathbf{T}_{n\times n} \cdot \mathbf{T}^{-1}_{n\times n} = \mathbf{I}$ was not verified for that case. The results of the exhaustive searches are listed in Table II.

In the early stages of the search process, we performed some preliminary verification of our algorithm by tracing the example presented in [2] for CRC-32 with $r = 32$. We noticed that the matrix $\mathbf{T}^{-1}_{32\times 32}$ reported at the end of the paper is not the inverse of $\mathbf{T}_{32\times 32}$, and the corrected inverse of matrix $\mathbf{T}_{32\times 32}$ shown in Figure 2. The reader can easily verify these claims by inputting the coefficients of the matrices into a computer algebra system (CAS) and performing the matrix multiplication $\bmod 2$.

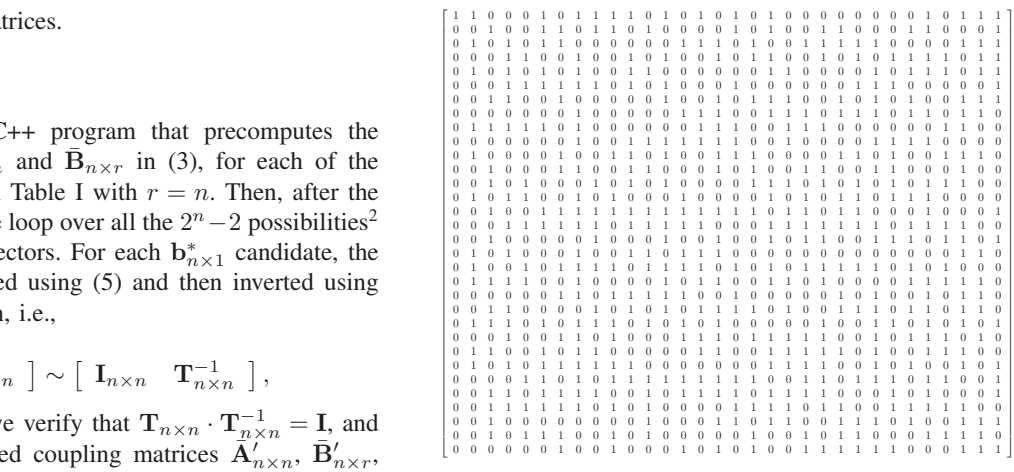

Figure 2: The matrix $\mathbf{T}^{-1}_{32\times 32}$ for CRC-32 with $r = 32$ and $\mathbf{b}^*_{32\times 1} = \left[\begin{array}{cccc} 1 & 0 & 0 & \cdots & 0 \end{array} \right]^T$.

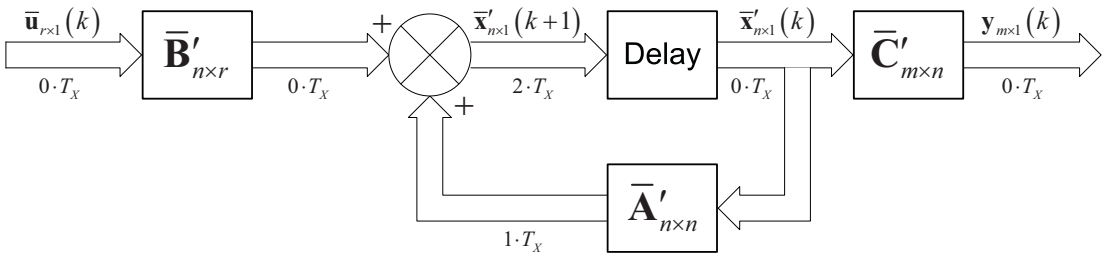

Figure 3: Illustration of the state-space transformed retimed architecture.

B. Realization

From the transformed state-space coupling matrices of a given system, the number of XOR gates and FFs required to implement the system described in (4) are calculated and the number of pipeline stages (PSs) is determined. The block diagram of the retimed architecture that implements (4) is shown in Figure 3. Here, we assume that only two-input XOR gates are available and the message input wires are clocked and have delay $0 \cdot T_X$. The architecture is retimed such that the outputs of the input and output coupling logic blocks are clocked, and any additional required retiming FFs are placed at the roots of the XOR trees.

Since we are considering two-input XOR gates, to sum up n terms for $n > 1$, $n-1$ XOR gates are required. As noted in [2], the XOR gate count of an implementation can be obtained by summing the number of 1s in the rows of the coupling matrices and subtracting 1 from each row subtotal, plus the n XOR gates required to perform the addition

$$\bar{\mathbf{x}}'_{n\times 1}(k+1) = \bar{\mathbf{A}}'_{n\times n} \cdot \bar{\mathbf{x}}'_{n\times 1}(k) + \bar{\mathbf{B}}'_{n\times r} \cdot \bar{\mathbf{u}}_{r\times 1}(k). \quad (6)$$

Now, because $\bar{\mathbf{A}}'_{n\times n}$ is a companion matrix, the best critical path delay of the retimed transformed system that one can obtain is $2 \cdot T_X$. In other words, the maximum delay of the output wires from the $\bar{\mathbf{A}}'_{n\times n} \cdot \bar{\mathbf{x}}'_{n\times 1}(k)$ block is T_X, then another level of XOR gates is required to perform addition in (6) and obtain $\bar{\mathbf{x}}'_{n\times 1}(k+1)$, which results in wires with delays of $2 \cdot T_X$. Thus, to retime the logic in the $\bar{\mathbf{B}}'_{n\times r} \cdot \bar{\mathbf{u}}_{r\times 1}(k)$ and $\bar{\mathbf{C}}'_{m\times n} \cdot \bar{\mathbf{x}}'_{n\times 1}(k)$ blocks, one can use a combination of the retiming blocks shown in Figure 4. We note that this retiming approach is slightly different than the one reported in [2].

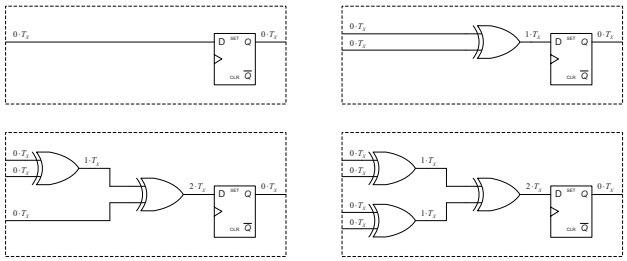

Figure 4: The retiming blocks used in our implementations.

V. Results

This section outlines the results of our C++ simulations and ASIC implementations. The exhaustive searches for the $n = 12$ and $n = 16$ generator polynomials are easily computed with a desktop computer in a reasonable short time. However, for the CRC-32 polynomial, our simulation ran continuously for a close to a month to obtain the reported $\mathbf{b}^*_{n\times 1}$ value. Since our first metric is the total number of 1s in the transformed coupling matrices, for half of the generator polynomials there are multiple $\mathbf{b}^*_{n\times 1}$ vectors which give the same minimized 1s count. For these cases, we choose the $\mathbf{b}^*_{n\times 1}$ that results in the fewest number of XOR gates. If there still is a tie, then the number of FFs was considered, and finally the number of PSs. We denote the $\mathbf{b}^*_{n\times 1}$ vectors which produce minimized state-space representations with hats as $\hat{\mathbf{b}}^*_{n\times 1}$, and to save space we use a compact hexadecimal notation to represent the $\hat{\mathbf{b}}^*_{n\times 1}$ vectors. For example, $\hat{\mathbf{b}}^*_{n\times 1}$ for CRC-12 with $r = 12$ is found to be $\hat{\mathbf{b}}^*_{12\times 1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}^T$, and we denote the vector $\hat{\mathbf{b}}^*_{12\times 1}$ as $[\mathtt{0x814}]^T$.

Tables IIIa and IIIb give the number of XOR gates, FFs, and PSs required to realize a state space generated from the simple $\mathbf{b}^*_{n\times 1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}^T$ compared to an $\hat{\mathbf{b}}^*_{n\times 1}$, respectively. The $\hat{\mathbf{b}}^*_{n\times 1}$ vectors that correspond to results in Table III are marked with asterisks in Table II. There is a case where multiple $\hat{\mathbf{b}}^*_{n\times 1}$ vectors result in minimized implementations when ranked by our metrics, and both vectors are marked. One notices significant reductions in the number of XOR gates and FFs when $\hat{\mathbf{b}}^*_{n\times 1}$ is used to construct the transformation matrix $\mathbf{T}_{n\times n}$ for all of the frequently referenced generator polynomials. Finally, we observe that for all of these experiments, the number of PSs is equal for both transformed state spaces.

We note that the two boldfaced results in Table IIIa for CRC-32, match the values reported in [2]. Furthermore, if the matrix $\mathbf{T}^{-1}_{32\times 32}$ reported at the end of [2] is used to construct $\bar{\mathbf{B}}'_{32\times 32} = \mathbf{T}^{-1}_{32\times 32} \cdot \bar{\mathbf{B}}_{32\times 32}$, then we have determined that the implementation of $\bar{\mathbf{B}}'_{32\times 32} \cdot \bar{\mathbf{u}}_{32\times 1}(k)$ requires $485-32 = 453$ XOR gates. Thus, it is likely that a transcription error occurred when the matrix $\mathbf{T}^{-1}_{32\times 32}$ was typed.

Experimenting with other input sizes, we found that for small values of r with the frequently referenced generator polynomials, it is possible to obtain transformed systems with $\bar{\mathbf{B}}'_{n\times r}$ matrices that have an entire row of 0s. In other words, the transformed system has states that are not coupled to the inputs. Those cases are advantageous because they reduce the

Table III: Retimed hardware requirements for frequently referenced generator polynomials when the input size is equal to the generator polynomial degree with: (a) simple $\mathbf{b}^*_{n\times 1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}^T$ used in [2] for CRC-32, (b) $\mathbf{b}^*_{n\times 1} = \hat{\mathbf{b}}^*_{n\times 1}$ found in this paper and presented in Table II.

(a)

$r=n$	$\bar{\mathbf{B}}'_{n\times r}$				$\bar{\mathbf{A}}'_{n\times n}$				$\bar{\mathbf{C}}'_{m\times n}$				Architecture Total			
$G(z)$	1	XOR	FF	PS	1	XOR	FF	PS	1	XOR	FF	PS	1	XOR	FF	PS
CRC-12	58	46	40	2	20	8	12	1	58	46	39	2	136	112	91	5
CRC-16	92	76	53	2	18	2	16	1	108	92	62	2	218	186	131	5
CCITT-16	104	88	68	2	18	2	16	1	116	100	68	2	238	206	152	5
CRC-16†	102	86	57	2	18	2	16	1	130	114	66	2	250	218	139	5
CCITT-16†	118	102	70	2	18	2	16	1	112	96	69	2	248	216	155	5
CRC-32	498	**466**	281	3	45	13	32	1	488	**456**	249	3	1031	967	562	7

(b)

$r=n$	$\bar{\mathbf{B}}'_{n\times r}$				$\bar{\mathbf{A}}'_{n\times n}$				$\bar{\mathbf{C}}'_{m\times n}$				Architecture Total			
$G(z)$	1	XOR	FF	PS	1	XOR	FF	PS	1	XOR	FF	PS	1	XOR	FF	PS
CRC-12	54	42	41	2	20	8	12	1	46	34	31	2	120	96	84	5
CRC-16	80	64	53	2	18	2	16	1	90	74	56	2	188	156	125	5
CCITT-16	106	90	65	2	18	2	16	1	102	86	58	2	226	194	139	5
CRC-16†	80	64	52	2	18	2	16	1	92	76	57	2	190	158	125	5
CCITT-16†	106	90	65	2	18	2	16	1	102	86	58	2	226	194	139	5
CRC-32	447	415	235	3	45	13	32	1	436	404	210	3	928	864	477	7

number of retiming FFs and eliminate an XOR gate from the implementation of the addition in (6). An example of this instance is CRC-12 with $r = 6$ and $\mathbf{b}^*_{12\times 1} = [0\text{x}255]^T$, where the states $\bar{x}_1(k)$ and $\bar{x}_{10}(k)$ are not coupled to any input. However, since the columns of $\mathbf{T}_{n\times n}$ are linearly independent and $\bar{\mathbf{C}}_{m\times n} = \mathbf{I}$, all of the outputs are always coupled to a state. In other words, it is not possible to have an entire row of 0s in the $\bar{\mathbf{C}}'_{m\times n}$ matrix.

From our C++ search code, we developed an application that takes a generator polynomial, input size, and vector $\mathbf{b}^*_{n\times 1}$ as inputs and outputs complete VHDL files which describe that implementation. We note that an extra r FFs are added to clock the input wires of the design and they are not reported in Table III. The correctness of these VHDL files was then verified using vector waveform simulations with the Altera® Quartus® II Web Edition software. These VHDL files were then deployed on 0.18μ CMOS ASIC technology using the Synopsys® Design Analyzer®. The optimization effort was set to medium with a target clock period of 5.0 ns and the area (μm^2) and timing (ns) was obtained for each of the designs. The results from our ASIC experiments are summarized in Table IV. For CCITT-16 where there are multiple marked $\hat{\mathbf{b}}^*_{n\times 1}$ vectors, the smaller hexadecimal number is implemented.

Table IV: Retimed ASIC implementation results for frequently referenced generator polynomials when the input size is equal to the generator polynomial degree.

$r=n$	$\mathbf{b}^*_{n\times 1}$		$\hat{\mathbf{b}}^*_{n\times 1}$	
$G(z)$	area (μm^2)	delay (ns)	area (μm^2)	delay (ns)
CRC-12	11107	2.79	10156	2.79
CRC-16	16811	2.93	15299	2.72
CCITT-16	17905	2.73	17458	2.64
CRC-16†	18751	2.87	15445	2.71
CCITT-16†	18616	2.73	17470	2.49
CRC-32	72917	4.23	66412	3.16

These implementation results verify the expected reduction in area, and also demonstrate improvements in timing. The improvements in timing could be attributed to the reduction in area, which reduces wiring complexity and shortens the length of global wires in the design.

VI. Conclusions

This paper has revisited the state-space transformation approach to obtain high-speed CRC computation architectures. We have performed exhaustive searches and found the vectors that result in improved state-space representations with minimized hardware complexity, for frequently referenced generator polynomials when the input size is equal to the generator polynomial degree. Constructing the transformation matrix using these found vectors results in a reduction of the number of 1s in the transformed coupling matrices by an average of more than 12.2%, as compared to the previously reported one. The theoretical results stated in this paper are easily verified with a computer algebra system that can perform matrix inversions mod2. Moreover, the ASIC implementations demonstrate improvements in both area and timing. Finally, since the transformed state and output coupling matrices are unchanged and portions of input coupling matrix are maintained for different input sizes, we conclude that the vectors listed in Table II are good choices for a fixed generator polynomial.

Acknowledgments

This work has been supported by a Natural Science and Engineering Research Council (NSERC) Discovery Grant, awarded to A. Reyhani-Masoleh. The authors wish to acknowledge the work of Mehran Mozaffari Kermani who assisted our ASIC experiments.

REFERENCES

[1] W. W. Peterson and D. T. Brown, "Cyclic Codes for Error Detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[2] J. H. Derby, "High-Speed CRC Computation Using State-Space Transformations," in *the Proceedings of The 2001 IEEE Global Telecommunications Conference (GLOBECOM '01)*, vol. 1, 2001, pp. 166–170.

[3] M. A. Patel, "A Multi-Channel CRC Register," in *the Proceedings of The Spring Joint Computer Conference*, 1971, pp. 11–14.

[4] M. C. Nielson, "Method for High Speed CRC Computation," *IBM Technical Disclosure Bulletin*, vol. 27, no. 6, pp. 3572–3576, 1984.

[5] G. Albertengo and R. Sisto, "Parallel CRC Generation," *IEEE Micro*, vol. 10, no. 5, pp. 63–71, 1990.

[6] T.-B. Pei and C. Zukowski, "High-Speed Parallel CRC Circuits in VLSI," *IEEE Transactions on Communications*, vol. 40, no. 4, pp. 653–657, 1992.

[7] R. J. Glaise and X. Jacquart, "Fast CRC Calculation," in *the Proceedings of The IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'93)*, 1993, pp. 602–605.

[8] R. J. 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.

[9] 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 Engineering*, vol. 17, no. 3, pp. 445–461, 2001.

[10] M. Sprachmann, "Automatic Generation of Parallel CRC Circuits," *IEEE Design and Test of Computers*, vol. 18, no. 3, pp. 108–114, 2001.

[11] F. Monteiro, A. Dandache, A. M'sir, and B. Lepley, "A Fast CRC Implementation on FPGA Using a Pipelined Architecture for the Polynomial Division," in *the Proceedings of The 8th International IEEE Conference on Electronics, Circuits, and Systems (ICECS 2001)*, vol. 3, 2001, pp. 1231–1234.

[12] J. M. N. Serrano, "5x4 Gbps 0.35 Micron CMOS CRC Generator Designed with Standard Cells," in *the Proceedings of The 11th IEEE Mediterranean Electrotechnical Conference (MELECON'02)*, 2002, pp. 215–219.

[13] G. Campobello, G. Patane, and M. Russo, "Parallel CRC Realization," *IEEE Transactions on Computers*, vol. 52, no. 1, pp. 1312–1319, 2003.

[14] C. Cheng and K. K. Parhi, "High-Speed Parallel CRC Implementation Based on Unfolding, Pipelining, and Retiming," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 10, pp. 1017–1021, 2006.

[15] M. Walma, "Pipelined Cyclic Redundancy Check (CRC) Calculation," in *the Proceedings of The 16th IEEE International Conference on Computer Communications and Networks (ICCCN 2007)*, 2007, pp. 365–370.

[16] C. Kennedy and A. Reyhani-Masoleh, "High-Speed Parallel CRC Circuits," in *the Proceedings of The 42nd Annual Asilomar Conference on Signals, Systems, and Computers (to appear)*, 2008.

[17] J.-S. Lin, C.-K. Lee, M.-D. Shieh, and J.-H. Chen, "High-Speed CRC Design for 10 Gbps Applications," in *the Proceedings of The 2006 IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, 2006, pp. 3177–3180.