International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 2, April 2020, pp. 1755~1763
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i2.pp1755-1763

Area efficient parallel LFSR for cyclic redundancy check

Rita Mahajan, Komal Devi, Deepak Bagai
Department of Electronics and Communication Engineering, PEC (Deemed to be University), Chandigarh, India

Article Info

Article history:

Received Jun 12, 2019
Revised Oct 23, 2019
Accepted Oct 30, 2019

Keywords:

Area time product
CRC
Critical path delay
LFSR
State-space transformation

ABSTRACT

Cyclic Redundancy Check (CRC), code for error detection finds many applications in the field of digital communication, data storage, control system and data compression. CRC encoding operation is carried out by using a Linear Feedback Shift Register (LFSR). Serial implementation of CRC requires more clock cycles which is equal to data message length plus generator polynomial degree but in parallel implementation of CRC one clock cycle is required if a whole data message is applied at a time. In previous work related to parallel LFSR, hardware complexity of the architecture reduced using a technique named state space transformation. This paper presents detailed explaination of search algorithm implementation and technique to find number of XOR gates required for different CRC algorithms. This paper presents a searching algorithm and new technique to find the number of XOR gates required for different CRC algorithms. The comparison between proposed and previous architectures shows that the number of XOR gates are reduced for CRC algorithms which improve the hardware efficiency. Searching algorithm and all the matrix computations have been performed using MATLAB simulations.

Corresponding Author:

Rita Mahajan,
Department of Electronics and Communication Engineering,
Punjab Engineering College (Deemed to be University),
Sector-12, Chandigarh, India.
Email: ritamahajan@pec.ac.in

1. INTRODUCTION

CRC (Cyclic Redundancy Check) is most popular error detection algorithm which is used in digital communication, data storage and data compression. Various digital communication standards which use CRC code are Asynchronous Transfer mode (ATM), WiMAX (IEEE 802.16) and Wi-Fi (IEEE 802.11). CRC algorithm is based on cyclic code which follow two properties that is linearity and cyclic. LFSR is used to perform CRC operation [1]. LFSR is an important circuit in field of communication which is used in encoders, decoders, cryptography, CDMA and test pattern generator. Two implementation styles of LFSR are Fibonacci and Galois [2]. Power dissipation is challenging problem in VLSI field. In order to reduce the power consumption of LFSR, various algorithms have been proposed. In [3], Concept of reducing transitions in generated test patterns has been used for LFSR. Gated clock approach was used to reduce the power consumption. [4]. The switching and testing time in LFSR was reduced in [5] for power consumption reduction. There are two Conventional serial LFSR uses Galois implementation for nth order generator polynomial as shown in Figure 1.

There are different types of CRC algorithms available. Each CRC algorithm has predefined generator polynomial which is used to generate CRC code. Figure 1 is drawn for generalized case of generator polynomial given by the equation $G(x)=1+\sum g_j.s^i+s^n$ where i, j varies from 1 to n-1 and n is generator polynomial degree. Generator polynomial is different for different types of CRC algorithms. Coefficients of generator polynomial are g_0, g_1, g_2 …. g_{n-1} in which g_0 and g_n are always equal to one.

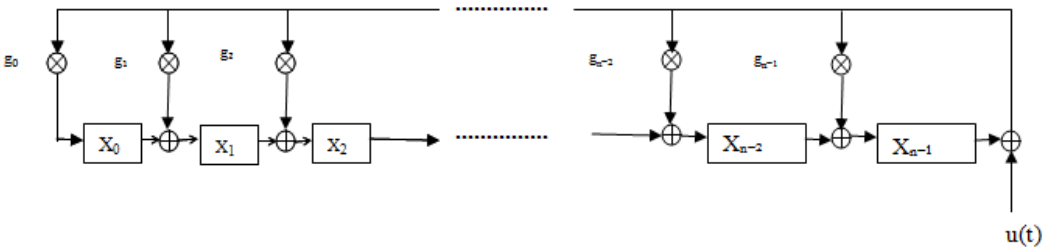

Figure 1. Conventional serial modular type LFSR architecture

The authors in [6] gave theory behind CRC and designed different hardware and software methods for CRC implementation. Parallel CRC implementations based upon mathematical deductions have been proposed in [7]. In [8-13], state space representation of serial and parallel LFSR is given. These all architectures process the data message of length 'm' which is divided into number of blocks of length 'v' each where 'v' is same as the generator polynomial degree (n). State-space similarity transformation and transformation matrix have been proposed which is dependent on some vector and exhaustive search was applied for finding the vector [10].

In [12], parallel architecture for LFSR has been proposed on the basis of transposed serial LFSR which reduces hardware complexity as well as critical path delay (CPD). The authors in [13] gave best way to find transformation matrix and improved searching algorithm is given to obtain best vector that reduces hardware complexity. Parhi [14] and Zhang and Parhi [15] has been proposed BCH encoders high speed architecture. For elimination of fan-out bottleneck new approaches have been proposed in [14, 16]. To reduce the effect of fan-out authors in [17] has been proposed a two-step method. Pipelining methods have been proposed for high speed CRC hardware implementation in [18]. In [19, 20] approaches have been presented that are based on software based CRC computation.

Matrix based formulation have been performed for computation of CRC for the specific case when generator polynomial degree is less than the degree of parallelism [16]. For transformation from serial to parallel new methods have been described [21] and firstly applied by Patel [22] for CRC calculation. For parallelization of computation Look-ahead technique used in [23]. Parallel implementation of CRC achieved by using cascading approach [24]. Performance of CRC polynomials generated with Genetic algorithms has been evaluated in [25]. Large fan-out effect has been eliminated using new technique based on Chinese Remainder Theorem for long BCH codes [26]. In [13], improved search algorithm has been proposed which finds the best vector that results in less hardware. This paper presents the implementation and results of this search algorithm using MATLAB. Functions such as number of ones calculation, generation of kids from parent node, calculation of total number of ones (TN) for each child node in level are used to implement the search algorithm.

The main requirement is to reduce the harware complexity of the design. To solve this problem, improved technique is given in this paper to find number of XOR gates for different CRC algorithms. This method is based on sharing technique in multiple rows which results in hardware efficient architecture as compared to previous one. The remaining paper is organized as follows: In section 2, representation of serial and parallel linear feedback shift register is done using state space method, summarizes the technique and constraint to obtain transformation (T) matrix which is presented in [13], gives detail regarding implementation of search algorithm and explaination of new technique to find number of XOR gates. Section 3 gives analysis of comparison between proposed architecture and previous reported parallel architectures and conclusion is given in section 4.

2. RESEARCH DESIGN

2.1. State-space representation of LFSR

State-space equation of conventional serial LFSR implementation is given as under

$$X_{n \times 1}(t+1) = A_{n \times n} . X_{n \times 1}(t) + B_{n \times 1} . u(t) \quad (1)$$

where in equation (1), $X_{n \times 1}(t)$is (n x 1) order vector and u(t) represent single bit input at time 't' and matrix A is in terms of generator polynomial coefficients which is same as the transition matrix in Galois implementation of LFSR. A matrix is shown below.

$$\begin{bmatrix} g_{n-1} & 1 & 0 & 0 & \cdots & \cdots & 0 \\ g_{n-2} & 0 & 1 & 0 & \cdots & \cdots & 0 \\ g_{n-3} & 0 & 0 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ g_1 & 0 & 0 & 0 & \cdots & \cdots & 1 \\ g_0 & 0 & 0 & 0 & \cdots & \cdots & 0 \end{bmatrix}$$

B vector is row vector of order n x 1 which is given as:
$B= (g_{n-1}, g_{n-2}, \ldots\ldots\ldots,g1,g0)T$
According to equation (1) $X_{n\ x\ 1}(t+2)$ is given by
$X_{n\ x\ 1}(t+2) = A_{n\ x\ n} . X_{n\ x\ 1}(t+1) + B_{n\ x\ 1} . u(t+1)$
$X_{n\ x\ 1}(t+2) = A_{n\ x\ n} [A_{n\ x\ n}. X_{n\ x\ 1}(t) + B_{n\ x\ 1} . u(t)]+ B_{n\ x\ 1} . u(t+1) = A^2_{n\ x\ n} . X_{n\ x\ 1}(t) +$
$A_{n\ x\ n}. B_{n\ x\ 1} . u(t) + B_{n\ x\ 1} . u(t+1)$
Similarly, $X_{n\ x\ 1}(t+3)$, $X_{n\ x\ 1}(t+4)$ and so on can be found.
According to all these recursive relations $X_{n\ x\ 1}(t+v)$ can be given as

$$X_{nxl}(t+v)=A^v_{nxn}.X_{nxl}(t)+((A^{v-1}.B, \ldots A.B, B)\ ).[u(t), u(t+1), u(t+2) \ldots . u(t+v-1)]^T \quad (2)$$

Data message length is equal to 'm' and 'v' is equal to message input which is applied at the time 't'. So v divides the data message of m length. At time t, parallel input is denoted as $U_v(t)$ and given as u(tv), u(tv+1), u(tv+2) .u(tv+v-1)^T where t varies from 0,1,2,3,4… ((m/v)-1). (2) can be given as :

$$X_{nxl}(t+v)=A^v_{nxn}.X_{n\ x\ 1}(t) + B_v. U_v(t) \quad (3)$$

where Bv matrix has order n x v. Linear transformation is applied on $X_{n\ x\ 1}(t)$ to reduce the complexity. Transformation matrix used must be a nonsingular matrix. $X_{nxl}(t) = T_{nxn}$ x $X^T_{nxl}(t)$. Transformed state space equation for (3) can be written as:

$$X^T_{nxl}(t+1) = A_vT. X^T_{nxl}(t) + B_vT.U_v(t) \quad (4)$$

where $A_vT = T^{-1} .A^v. T$ and $B_vT=T^{-1}. B_v$

Table 1 given below represents generator polynomial for different CRC algorithms. Figure 2 shows block diagram of parallel LFSR based on state space transformation described by (4).

Table 1. Generator polynomial for different CRC algorithm

S.No.	Name of the code	Generator Polynomials
1	CRC-12	$s^{12} \oplus s^{11} \oplus s^3 \oplus s^2 \oplus s \oplus 1$
2	CRC-16	$s^{16} \oplus s^{15} \oplus s^2 \oplus 1$
3	CRC-CCITT (SDLC)	$^{16} \oplus s^{12} \oplus s^5 \oplus 1$
4	CRC-16 REVERSE	$s^{16} \oplus s^{14} \oplus s \oplus 1$
5	SDLC REVERSE	$s^{16} \oplus s^{11} \oplus s^4 \oplus 1$
6	CRC-32	$s^{32} \oplus s^{26} \oplus s^{23} \oplus s^{22} \oplus s^{16} \oplus s^{12} \oplus s^{11} \oplus s^{10} \oplus s^8 \oplus s^7 \oplus s^5 \oplus s^4 \oplus s^2 \oplus s \oplus 1$

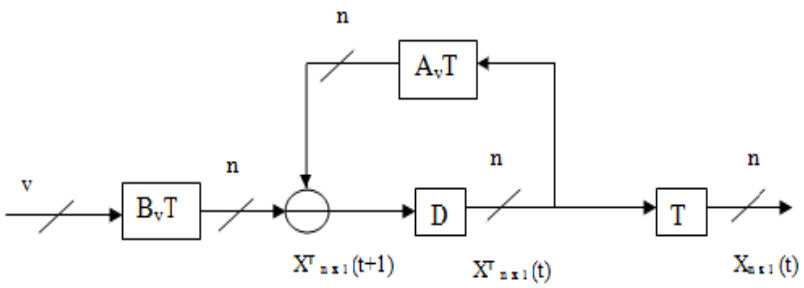

Figure 2. Parallel linear feedback shift register architecture using state-space transformation

2.2. Method to find transformation matrix

There are many methods to find best transformation matrix. In [10], transformation matrix was selected in such a way that would simplify the feedback path complexity but outside the feedback loop, circuit complexity increased. This complexity resulted in more hardware cost. In [13], new method was presented to find improved transformation matrix and following constraints were followed to find better transformation matrix.

- The transformation matrix T must be non-singular matrix that could satisfy state space transformation.
- Less number of ones in coupling matrices AvT, BvT and T matrix. Because less number of ones directly decide hardware complexity of the circuit in terms of XOR gates.
- Efficient method to search transformation matrix.

For transformation matrix, vector V of length n is used. Vector value cannot be equal to 0. Vector MSB must be equal to 1 and other elements of this vector can be 0 or 1. Vector V is defined as V = [1, v_1, v_2, v_3, ... v_{n-1}]. T matrix constructed from vector V is given below.

$$\begin{bmatrix} 1 & v_1 & v_2 & v_3 & \cdots & v_{n-2} & v_{n-1} \\ 0 & 1 & v_1 & v_2 & \cdots & v_{n-3} & v_{n-2} \\ 0 & 0 & 1 & v_1 & \cdots & v_{n-4} & v_{n-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & v_1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

2.3. Search algorithm implementation

This algorithm search for the best vector which gives smallest TN. Total levels in the searching tree is equal to number of total bits or equal to generator polynomial degree. The root node in the level 1 is given value: [1 0 0……...0] and child nodes in level n are obtained by replacing 0 with 1 in parent node of level (n-1) which means number of child nodes for specific parent node is equal to number of zeros in that parent node. Each node in Level n contains n number of ones.

C program is designed which computes the total number of nodes at each level, since the values increases a lot as computation for higher bits is started. So, it is very important to know the total number of nodes at each level otherwise CPU may take lots of time to complete the TN calculation and memory requirement also increases. So, first step is to compute the total number of nodes at each level and next step is to find the value of nodes at each level. To ease the output readability, values are shown in decimal format and converted to binary internally for computation of TN. Since output window has limited size in C application, so the output is shown for n=3 in Figure 3 and for n=5 in Figure 4. C application was first implemented to find total number of nodes and value of nodes at each level and then search algorithm is implemented in MATLAB, where saving and handling matrix data is easier as compared to C language. Search algorithm steps are described by using flowchart given in Figure 5. MATLAB code structure for search algorithm is shown in Figure 6. Description of each function and main code is given below.

```
Enter the number of bits: 3
Total Number of Nodes in 3 bits tree at each level is:
 LEVEL 1: 1
 LEVEL 2: 2
 LEVEL 3: 2
 Total Nodes are 5

 -----------------CHILD NODES AT ALL LEVELS----------------------------

LEVEL 1:  4
LEVEL 2 : 6      5
LEVEL 3 : 7      7
 --------------------DONE----------------------------


--------------------------------
Process exited after 2.128 seconds with return value 0
Press any key to continue . . .
```

Figure 3. Level Output for n=3

```
Enter the number of bits: 5
Total Number of Nodes in 5 bits tree at each level is:
 LEVEL 1: 1
 LEVEL 2: 4
 LEVEL 3: 12
 LEVEL 4: 24
 LEVEL 5: 24
 Total Nodes are 65

 -----------------CHILD NODES AT ALL LEVELS----------------------------

LEVEL 1:  16
LEVEL 2 : 24      20      18      17
LEVEL 3 : 28      26      25      28      22      21      26      22      19      25      21      19
LEVEL 4 : 30      29      30      27      29      27      30      29      30      23      29      23      30      27
  30      23      27      23      29      27      29      23      27      23
LEVEL 5 : 31      31      31      31      31      31      31      31      31      31      31      31      31      31
  31      31      31      31      31      31      31      31      31      31
 --------------------DONE----------------------------

------------------------------
Process exited after 2.483 seconds with return value 0
Press any key to continue . . .
```

Figure 4. Level Output for n=5

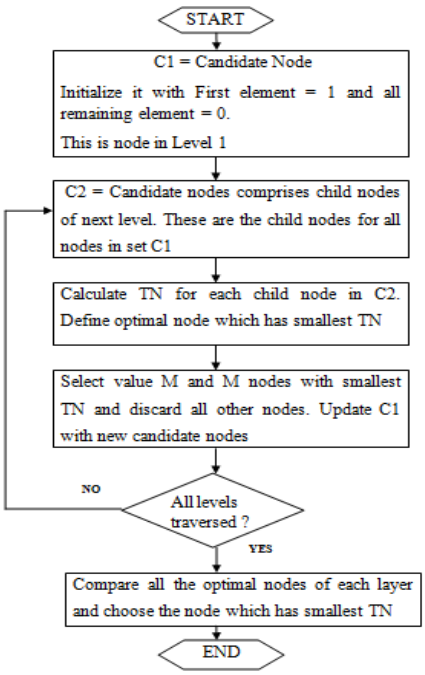

Figure 5. Flow chart for Search algorithm

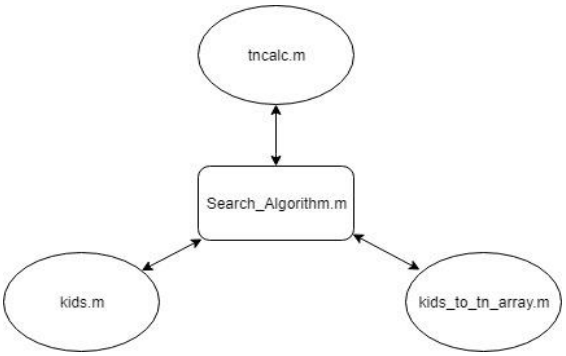

Figure 6. MATLAB code structure

- *Search_Algorithm.m*

This is the main file to get the input in the form of number of bits and 'M' value from user. It computes the parent node and calculate other elements using functions of MATLAB used in code. Final output shows oneT (number of ones in T matrix), oneA_vT (number of ones in A_vT matrix), oneB_vT (number of ones in B_vT matrix) and TN with its corresponding child nodes and optimal node at each level. Array of optimal nodes and node which has smallest value is shown at the end.

- *Kids.m*

This is MATLAB function which takes the input in the form of parent node and number of bits. It generates multiple possible child nodes of parent node, saves the same in kid array and returns it to the main function.

- *tncalc.m*

This function computes the TN, oneA_vT, oneB_vT and oneT value from each child node. It takes the input in form of child nodes and number of bits. These calculations are based on constant element such as A (transition matrix) and B vector which are different according to the CRC algorithm.

- *kids_to_tn_array.m*

This function converts the kid array into tn array using tncalc function. It is required to compute the TN value of each child node which helps in computing the optimal node at each level. The process of computing optimal node includes calculation of lowest TN value at each level. These optimal values are saved in matrix and at the end of code, the lowest value from all optimal values is sorted. Final optimal value location is identified in code which helps in finding the corresponding level and the node.

After calculating best vector which has smallest TN the next way to improve hardware efficiency is to find number of XOR gates in the matrices of different CRC algorithm. If number of ones in any row is equal to n and n is greater than 1, number of XOR gates needed to perform modulo-2 addition is (n-1) [10]. If single one is present in any row, then there is no requirement of XOR. XOR gates are calculated based on the subexpression sharing technique. In [13], sharing technique is applied between two rows but in the proposed method, sharing technique is applied in multiple rows.

Technique to find number of XOR gates is explained below. Matrices AvT, BvT and T for all CRC algorithms are computed using MATLAB and then observe number of ones present in each row and column. If number of ones are present at same columns in multiple rows, then number of XOR required for one row are shared by other rows. By using this multiple sharing technique, less number of XOR gates are required which further reduces hardware complexity over previous architecture [13].

3. RESULTS AND ANALYSIS

Table 2 shows vector V used, total number of ones and XOR gates required for T, AvT and BvT matrices for all CRC algorithms. The comparison is given for previous model [13] and proposed method. Number of XOR gates reduced from 53 to 35, 97 to 74, 82 to 77, 90 to 82, 461 to 255 for CRC-12, SDLC, CRC-16 Reverse, SDLC Reverse and CRC-16 respectively and percentage reduction in XOR is 33.96%, 23.71%, 6.09%, 8.88% and 44.68% for above types of CRC. Percentage reduction in XOR is shown in Table 2 rightmost column. It is observed that in proposed method number of XOR gate reduced for every type of CRC instead of CRC-16. It should be noted that percentage reduction for CRC-32, 44.68% is very high means very large reduction in hardware complexity is achieved. Finally, comparison between previous architectures done on the basis of AT for frequently referenced generator polynomial given in Table 1. Area Time Product (AT) depends on delay element (DE), total no. of XOR and CPD. AT is calculated as:

$$AT=(1.5 \cdot DE + XOR). CPD$$

Table 3 gives the comparison between various architectures and proposed one. For the proposed method, number of XOR gates are reduced for every CRC generator polynomial which means AT value is reduced for every CRC. Relative value of AT is given in the rightmost column and the value is normalized with the proposed one. It is observed that relative value is more than one for all previous architectures except for CRC-16. All computations are based on the assumption that the level of parallelism 'v' is same as the generator polynomial degree.

Table 2. Comparison of paper [13] with the proposed one in terms of no. of ones and XOR for T, AvT and BvT matrices

Name of the code	Algorithm	Vector b_2⋆ Vector V	AvT 1	AvT XOR	BvT 1	BvT XOR	T 1	T XOR	Total TN	Total XOR	%age Reduction in XOR
CRC-12	[13]	[0xA01]	29	19	25	17	23	5	77	53	33.96%
	Proposed		29	13	25	11	23	11	77	35	
CRC-16	[13]	[0xC001]	35	23	33	15	32	6	100	60	No reduction
	Proposed		35	16	33	15	32	16	100	63	
CRC-CCITT (SDLC)	[13]	[0x8408]	88	46	45	25	31	10	164	97	23.71%
	Proposed		88	26	45	17	31	15	164	74	
CRC-16 REVERSE	[13]	[0xC002]	154	32	73	19	33	15	260	82	6.09%
	Proposed		154	29	73	15	33	17	260	77	
SDLC REVERSE	[13]	[0x8810]	84	41	38	20	33	13	155	90	8.88%
	Proposed		84	34	38	15	33	17	155	82	
CRC-32	[13]	[0x80000212]	414	221	425	198	49	10	888	461	44.68%
	Proposed		414	107	425	99	49	17	888	255	

Table 3. Several architectures comparison in terms of AT

Name of the code	Algorithm	XOR	DE	CPD	AT
CRC-12	[11]	109	36	5	815 (2.86)
	[12]	103	24	4	556 (1.95
	[13]	53	24	4	356 (1.25)
	Proposed	35	24	4	284 (1.00)
CRC-16	[11]	113	50	5	940 (1.69)
	[12]	94	32	5	710 (1.27)
	[13]	60	32	5	540 (0.97)
	Proposed	63	32	5	555 (1.00)
CRC-CCITT (SDLC)	[11]	139	50	5	1070 (2.92)
	[12]	97	32	3	435 (1.18)
	[13]	97	32	3	435 (1.18)
	Proposed	74	32	3	366 (1.00)
CRC-16 REVERSE	[11]	126	50	5	1005 (2.01)
	[12]	82	32	4	520 (1.04)
	[13]	82	32	4	520 (1.04)
	Proposed	77	32	4	500 (1.00)
SDLC REVERSE	[11]	127	50	5	1010 (1.94)
	[12]	90	32	4	552 (1.06)
	[13]	90	32	4	552 (1.06)
	Proposed	82	32	4	520 (1.00)
CRC-32	[11]	794	96	5	4690 (2.67)
	[12]	675	64	5	3855 (2.19)
	[13]	461	64	5	2785 (1.58)
	Proposed	255	64	5	1755 (1.00)

4. CONCLUSION

This paper presented a new method to find number of XOR gates in transformation and coupling matrices for generator polynomial of different types of CRC. Implementation details of search algorithm are given. Based upon this new improved method of finding XOR, the proposed method achieve smaller AT as compared to previous architectures. Reduced AT value means parallel architecture of less hardware complexity is obtained.

ACKNOWLEDGEMENTS

The authors would like to thank Punjab Engineering College for support, technical assistance and appreciation.

REFERENCES

[1] G. Campobello, “Parallel CRC Realization,” *IEEE Transactions On Computers*, 52(10), pp.1312-1319, 2003.
[2] M. Goresky, A.M. Klapper. Fibonacci and Galois Representations of Feedback-With-Carry Shift Registers[J]. IEEE Transactions On Information Theory, 48 (11), pp. 2826-2836, 2002.
[3] B. Singh, A. Khosla and S. Bindra, "Power Optimization of Linear Feedback Shift Register (LFSR) for Low Power BIST," *2009 IEEE International Advance Computing Conference*, Patiala, pp. 311-314, 2009.
[4] W. Aloisi and R. Mita, "Gated-Clock Design of Linear-Feedback Shift Registers," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 6, pp. 546-550, June 2008.

[5] S. Hussain and V.M. Rao, "An Approach to Measure Transition Density of Binary Sequences for X-Filling based Test Pattern Generator in Scan based Design," *International Journal of Electrical and Computer Engineering*, vol. 8 no. 4, pp.2063-2071, August 2018.
[6] T. V. Ramabadran and S. S. Gaitonde, "A tutorial on CRC computations," *in IEEE Micro*, vol. 8, no. 4, pp. 62-75, Aug. 1988.
[7] T. -. Pei and C. Zukowski, "High-speed parallel CRC circuits in VLSI," in *IEEE Transactions on Communications*, vol. 40, no. 4, pp. 653-657, April 1992.
[8] M. Ayinala and K. K. Parhi, "Efficient parallel VLSI architecture for linear feedback shift registers," *2010 IEEE Workshop On Signal Processing Systems*, San Francisco, CA, pp. 52-57, 2010.
[9] J. H. Derby, "High-speed CRC computation using state-space transformations," *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, San Antonio, TX, pp. 166-170 vol.1, 2001.
[10] C. Kennedy and A. Reyhani-Masoleh, "High-speed CRC computations using improved state-space transformations," *2009 IEEE International Conference on Electro/Information Technology*, Windsor, ON, pp. 9-14, 2009.
[11] M. Ayinala and K. K. Parhi, "High-Speed Parallel Architectures for Linear Feedback Shift Registers," in *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4459-4469, Sept. 2011.
[12] J. Jung, H. Yoo, Y. Lee and I. Park, "Efficient Parallel Architecture for Linear Feedback Shift Registers," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 11, pp. 1068-1072, Nov. 2015.
[13] G. Hu, J. Sha and Z. Wang, "High-Speed Parallel LFSR Architectures Based on Improved State-Space Transformations," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 1159-1163, March 2017.
[14] K. K. Parhi, "Eliminating the fanout bottleneck in parallel long BCH encoders," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 3, pp. 512-516, March 2004.
[15] X. Zhang and K. K. Parhi, "High-speed architectures for parallel long BCH encoders," in *Proc. ACM Great Lakes Symp. VLSI*, Boston, MA, USA, pp. 1–6, Apr. 2004.
[16] C. Kennedy and A. Reyhani-Masoleh, "High-speed parallel CRC circuits," in *Proc. 42nd Annu. Asilomar Conf. Signals, Syst., Comput.*, pp. 1823–1829, Oct. 2008.
[17] C. Cheng and K. K. Parhi, "High speed VLSI architecture for general linear feedback shift register (LFSR) structures," in *Proc. 43rd Asilomar Conf. on Signals, Syst., Comput.*, Monterey, CA, pp. 713–717, Nov. 2009.
[18] C. Cheng and K. K. Parhi, "High-Speed Parallel CRC Implementation Based on Unfolding, Pipelining, and Retiming," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 10, pp. 1017-1021, Oct. 2006.
[19] Y. Do, S. Yoon, T. Kim, K. E. Pyun and S. Park, "High-Speed Parallel Architecture for Software-Based CRC," *2008 5th IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, pp. 74-78, 2008.
[20] M. E. Kounavis and F. L. Berry, "Novel Table Lookup-Based Algorithms for High-Performance CRC Generation," in *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1550-1560, Nov. 2008.
[21] M. Y. Hsiao and K. Y. Sih, "Serial-to-Parallel Transformation of Linear-Feedback Shift-Register Circuits," in *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 6, pp. 738-740, Dec. 1964.
[22] A. M. Patel, "A multi-channel CRC register", in *AFIPS Conference Proceedings*, vol. 38, pp. 11-14, Spring 1971.
[23] 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.
[24] M.Sprachmann, "Automatic Generationof Parallel CRC Circuits," *IEEE Design and Test of Computers*, vol. 18, no. 3, pp. 108–114, 2001.
[25] A.S. Khirbeet and R.C. Muniyandi, "New Heuristic Model for Optimal CRC Polynomial," *International Journal of Electrical and Computer Engineering*, *7*(1), pp. 521-525, February 2017.
[26] H. Chen, "CRT-based high-speed parallel architecture for long BCH encoding," *IEEE Trans. Circuits Syst. II: Expr. Briefs*, vol. 56, no. 8, pp. 684–686, Aug. 2009.

BIOGRAPHIES OF AUTHORS

Rita Mahajan was born in India, Punjab, in 1965. She did her B.E. (E&EC) Degree from Thapar University, Patiala in 1986. She did her M.E. (Electronics) and Ph.D from Punjab Engineering College Chandigarh, India in 1993 and 2016 respectively. She has got 28 years of teaching experience. She has published more than 20 papers in International and National journals and conferences. Her research interests include neural networks and cognitive radios, Digital VLSI Design.

Komal Devi was born in India, Haryana, in 1990. She did her B. Tech (ECE) Degree from Kurukshetra University, Kurukshetra in 2011. She is pursuing her M.Tech (VLSI Design) from Punjab Engineering College Chandigarh, India. She has got 1.5 years of teaching experience. Her research interests include digital VLSI design.

Deepak Bagai was born in India, Chandigarh, in 1963. He did his B.E. (E&EC), M.E. (Electronics) and Ph.D from Punjab Engineering College Chandigarh, India in 1985, 1992, and 1997 respectively. He has got 28 years of teaching experience. He has published more than 40 papers in International and National journals and conferences. His research interests include telecommunications and wireless communications.