High-speed Parallel Architecture and Pipelining for LFSR

Vinod Mukati
PG (M.TECH. VLSI engineering) student, SGVU Jaipur (Rajasthan).
Vinodmukati90@gmail.com

Abstract— Linear feedback shift register plays an important role in many electronic circuits. LFSRs also used in BIST (Built-in self-Test) technique and as well as Design for Test (DFT). LFSRs are an important part of the CRC (Cyclic Redundancy Check) and BCH encoders. This paper has two fold. First this paper shows the mathematical proof of existence of a linear transformation to transform LFSR circuit in to equivalent state space formulation. This transformation technique has greater advantage as compare to serial architecture at the cost of an increase in hardware overhead. In the generation of the polynomials this method is used, in CRC operation and BCH encoders. In the second fold we propose a new modification of the LFSR in to the form of an infinite impulse response (IIR) filter. In this fold high speed parallel LFSR architecture based on parallel IIR filter design, pipelining and retiming algorithms. We further propose another method in which we combine the parallel and pipelining technique to eliminate the fan out effect in long generator polynomial.

***Index Terms*— BCH, cyclic redundancy check (CRC), LFSR, look-ahead computation, parallel processing, pipelining, transformation.**

I. INTRODUCTION

Linear feedback shift register (LFSR) are widely used in error detection (CRC Operation) and BCH encoders.

A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. The only linear function of single bits is xor, thus it is a shift register whose input bit is driven by the exclusive-or (xor) of some bits of the overall shift register value. CRC (cyclic redundancy check) is the important method for error detection in communication process and BCH codes are among the most extensively use codes in modern communication system. LFSRs are also used in conventional Design for Test (DFT) and Built-in-self-Test (BIST). Many parallel architectures of LFSR have been proposed in the literature for BCH and CRC encoders to increase the throughput [5]. In [5] and [6], parallel CRC implementations have been proposed based on mathematical deduction. In this paper presentation we used the recursive formulation for derived parallel CRC architectures. High-speed architectures for BCH encoders have been proposed in [6] and [12]. This architecture based on multiplication and division computations on generator polynomials. We can use the LFSR for generation

Of polynomials. They are efficient in terms of speeding up The LFSR but their hardware cost is high.

Another problem occurs with the parallel architecture is the hardware cost. In this paper the previous proposed method is CRC architecture based on state space representation [9]. The main advantage of this architecture is that the complexity is shifted out of the feedback loop. The full speed-up can be achieved by pipelining the feed forward paths. . A state space transformation has been proposed in [9] to reduce complexity but the existence of such a transformation was not proved in [9].

This paper has its two folds .first in which we present the mathematical proof of sate space transformation exists for all CRC and BCH generator polynomials. We also show in this paper that this transformation is non-unique. In this paper we proposed a new method of formulation of LFSR in terms of IIR filter. Then we propose a novel scheme based on pipelining, retiming, and look ahead computation to reduce the path in the parallel architecture base on parallel and pipelined IIR filter design. The proposed IIR filter based parallel architectures have both feedback and feed forward paths, and pipelining can be applied to further reduce the critical path.

We can say that the proposed method can achieve a critical path similar to previous design with less hardware overhead, without loss generality, binary codes are considered. This paper is an expanded version of [15].

II. LFSR ARCHITECTURE

Linear shift register is an important element for the many electronics circuit. In which we can use LFSR as hardware in polynomial generation or in the CRC (cyclic redundancy check) method of error detection. The Cyclic Redundancy check process can be easily implemented in hardware using LFSR. The LFSR divides a message polynomial by a suitably choose divisor polynomial. The remainder constitutes the FCS (Frame check sequence).

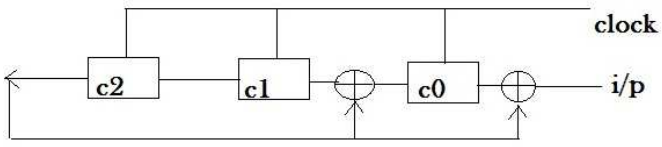

Figure 1.LFSR Architecture (1010000)

Table 1. Shows the operation for x^3+x+1 polynomial.

initial	C_2 0	C_1 0	C_0 0	C_2+C_0 0	C_2+i/p 1	I/p 1
Step1	0	0	1	1	0	0

Step2	0	1	0	0	1	1
Step3	1	0	1	0	1	0
Step4	0	0	1	1	0	0
Step5	0	1	0	0	0	0
Step6	1	0	0	1	1	0
Step7	0	1	1	1	0	-

III. STATE SPACE REPRESENTATION OF LFSR

The Linear feedback Shift Register (LFSR) has its parallel architecture, which is based on state space representation. A state space representation is a mathematical model of a physical system as a set of input, output and state variable related by first order differential equation. State space representation of LFSR is shown below-

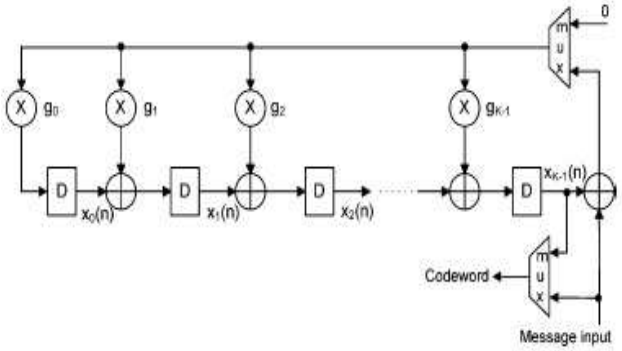

Figure. 2 Basic LFSR Architecture

The figure can be described by this equation-

$$x(n+1) = A\,x(n) + B\,u(n); \quad n>=0 \quad (1).$$

IV. STATE SPACE TRANSFORMATION

The complexity of feedback of can be reduced through the linear transformation. The State Space equation of L-parallel is given by in this manner

$$x(mL + L) = ALx(mL) + BLuL(mL); \; y(mL) = CLx(mL)$$

Where CL = I, the K £ K identity matrix. The output vector y (mL) is equal to the state vector which has the remainder at m = N=L. Consider the linear transformation of the state vector x (mL) through a constant non-singular matrix T, i.e.
x (mL) = Txt (mL).

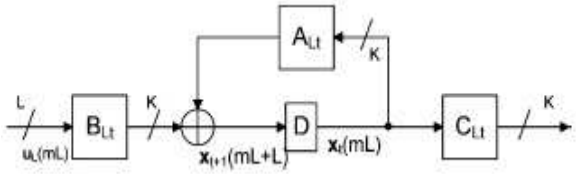

Figure3. Modified LFSR architecture using State space Transformation

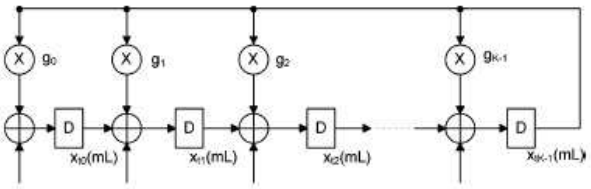

Figure 4.Modified feedback loop of fig. 3

V. IIR FILTER REPRESENTATION OF LFSR

In this section we propose a new architecture of LFSR in which general and parallel LFSR based on IIR filtering. The LFSR can be described using the following equations;

$$w(n) = y(n) + u(n)$$

$$y(n) = g_{k-1} * w(n-1) + g_{k-2} * w(n-2 + \ldots . g_0 * w(n-k)$$

Substituting (1) into (2) we get-

$$y(n)\; g_{k-1} * y(n-1) + g_{k-2} * y(n-2) + \ldots . g_0 y(n-k) + f(n)$$

Where $f(n) = g_{k-1} * u(n-1) + g_{k-2} * u(n-2) + \ldots + g_0 * u(n-k)$

In the above equation '+' denotes XOR operation.

The General Architecture of LFSR is shown below.-

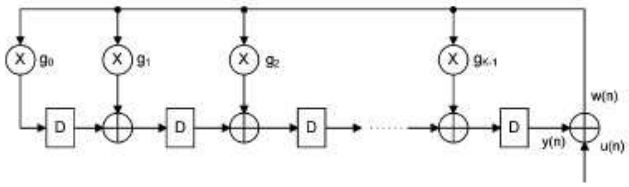

Figure 5. General LFSR Architecture

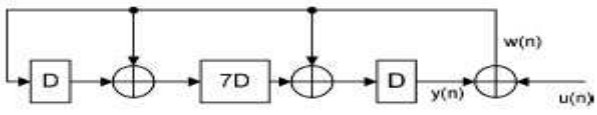

Figure 6. LFSR architecture for $g(x) = 1 + x + x^8 + x^9$.

Look ahead technique can be used in the derivation of parallel architecture. To derive parallel system for a given LFSR. Parallel architecture for a simple LFSR described in the previous section is discussed first. Consider the design of 3-parallel architecture for the LFSR in Fig. 6. In the parallel system, each delay element is referred to as a block delay where the clock period of the parallel system is 3 times the original sample period (bit period). Therefore, instead of (15), the loop update equation should update y (n) using inputs and y (n-3). The loop update process for the 3-parallel system is shown in Fig. 6.1, where y(3k+3), y(3k+4), and y(3k+5) are computed using y(3k), y(3k+1), and y(3k+2) . By iterating the recursion or by applying look-ahead technique,

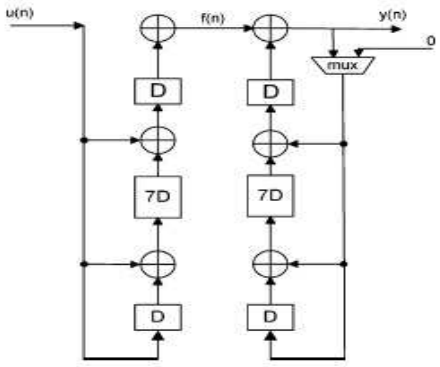

Figure 7. LFSR architecture for g(x) =1+x+x^8+x^9 after the proposed formation.

Table 2. Data Flow of Fig. 7 When the Input Message is 101011010.

Clock	U(n)	F(n)	Y(n)
1	1	1	1
2	0	0	1
3	1	1	0
4	0	0	0
5	1	1	1
6	1	1	0
7	0	0	0
8	1	0	0
9	0	1	0
10	0	1	0
11	0	1	0
12	0	1	1
13	0	0	1
14	0	1	0
15	0	1	1
16	0	1	1
17	0	0	0

We get

y(n)= y(n-1)+y(n-8)+y(n-9)+f(n)

= y (n-2) +y (n-8) +y (n-10) +f (n-1) +f (n)

=y(n-3)+y(n-8)+y(n-11)+f(n-2)+f(n-1)+f(n)

Substituting n=3k+3, 3k+4, 3k+5 in the above equations,
We have the following 3 loop update equations:

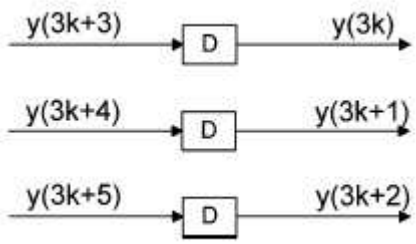

Figure 6.1 look update equations for block size L=3

y(3k+3)= y(3k+2)+y(3k-5)+y(3k-6)+f(3k+3)

y(3k+4)=y(3k+2)+y(3k-4)+y(3k-6)+f(3k+3)+f(32k+4)

y(3k+5) =y(3k+2)+y(3k-3)+y(3k-6)+f(3k+3)+f(3k+4)+f(3k+5)

Where

f (3k+3) = u (3k+2) +u (3k-5) +u (3k-6)

f (3k+4) = u (3k+3) +u (3k-4) +u (3k-5)

f (3k+5) = u (3k+4) +u (3k-3) +u (3k-4).

VI. COMBINING PARALLEL PROCESSING AND PIPELINING

The critical path can be reducing by the combination of parallel processing and pipelining process using IIR filter architecture. We use the two step look ahead computation compare to one step look ahead to generate the filter equation. We need to compute this equation as an example y (3k+8), y (3k+7), y (3k+6), instead of y (3k+5), y (3k+4), and y (3k+3). By this we can get two delays in the feed- back loop. Now, the loop update equations are

y(3k+3)= y(3k+2)+y(3k-2)+y(3k-6)+f(3k+3)+…+f(3k+6)
…. (2).

y(3k+4)= y(3k+2)+y(3k-1)+y(3k-6)+f(3k+3)+…+f(3k+7)
…… (3).

y(3k+5)= y(3k+2)+y(3k)+y(3k-6)+f(3k+3)+…+f(3k+8)
……. (4).

The feedback part of the architecture is shown in Figure 8. We can see from this figure that we can reduce the critical path in the feedback by applying the retiming in the feedback section.

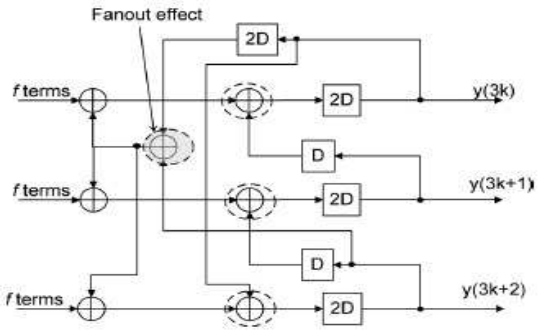

Figure 8. Loop update for combined parallel pipelined.

Poly(l)	Algo.	# XOR	#D.E.	C.P.	A.T.
CRC-12	[9]	113	35	5	1.01
	[13]*	276	47	3.15	1.70
	[5]	52	12	10	0.86
	Proposed	109	36	5	1
CRC-16	[9]	187	48	5	1.37
	[13]*	400	76	4	2.18
	[5]	72	16	15	1.53
	Proposed	113	50	5	1
SDLC 16	[9]	207	48	5	1.31
	[13]*	400	54	5.44	2.69
	[5]	88	16	8	0.84
	Proposed	139	50	5	1
CRC-16 Reverse(16)	[9]	219	46	5	1.43
	[13]*	592	68	5.97	4.14
	[5]	154	16	15	2.66
	Proposed	126	50	5	1.43
SDLC Reverse(16)	[9]	217	48	5	1.43
	[13]*	233	76	7.4	2.74
	[5]	84	16	8	0.86
	Proposed	127	50	5	1.18
CRC-32 (32)	[9]	968	96	5	1.18
	[13]*	6496	344	16.13	25.42
	[5]	452	32	17	1.81
	Proposed	794	96	5	1
BCH (255,223) (32)	[9]	903	96	5	1.24
	[13]*	4832	276	14	14.58
	[5]	588	32	24	2.80
	Proposed	863	96	5	1

Figure 9.Loop update for combined parallel pipelined for LFSR after retiming.

VII. COMPARISON AND ANALYSIS

In the error detection technique the commonly generator polynomial for CRC and BCH encoders that is shown in table 3.A comparison between the previous high speed architectures and the proposed ones is shown in Table IV for different parallelism levels of different generator polynomials. The comparison is depending upon the required number of XOR gates.

Table 3. common used generator polynomial.

CRC-12	$X^{12}+x^{3}+x^{2}+x+1$
CRC-16	$X^{16}+x^{15}+x^{2}+1$
SDLC	$X^{16}+x^{12}+x^{5}+1$
CRC-16 REVERRSE	$X^{16}+x^{14}+x+1$
SDLC REVERSE	$X^{16}+x^{11}+x^{4}+1$
CRC-32	$X^{32}+x^{26}+x^{23}+x^{22}+x^{16}+$ $x^{12}+x^{11}+X^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$

Table 4. Comparison to Previous LFSR Architecture and the proposed one.

Table 5. Comparison of C.P and xor gates of the proposed design and previous parallel long BCH (8191, 7684) Encoder for L-parallel Architecture.

VIII. CONCLUSION

In this paper we show the mathematical proof to show that a transformation exists in state space. By which help we can reduce the complexity of the parallel LFSR feedback loop. This paper present a new novel method for high speed parallel implementation of linear feedback shift register based on IIR filtering and this is the proposed method. This proposed method can reduce the critical path and the hardware cost at the same time. This design is applicable for any type of LFSR architecture. In the combined pipelining and parallel processing technique of IIR filtering , critical path in the feedback part of the design can be reduced. For the future work we can use this proposed design with combined parallel and pipelining for long BCH codes.

IX. REFRENCES

[1] T. V. Ramabadran and S. S. Gaitonde, “A tutorial on CRC computations,” IEEE Micro., Aug. 1988.

[2] R. E. Blahut, Theory and Practice of Error Control Codes. Reading, MA: Addison-Wesley, 1984.

[3] W. W. Peterson and D. T. Brown, “Cyclic codes for error detection,” Proc. IRE, vol. 49, pp. 228–235, Jan. 1961.

[4] N. Oh, R. Kapur, and T. W. Williams, “Fast speed computation for reseeding shift register in test pattern compression,” IEEE ICCAD, pp. 76–81, 2002.

[5] T. B. Pei and C. Zukowski, “High-speed parallel CRC circuits in VLSI,” IEEE Trans. Commun., vol. 40, no. 4, pp. 653–657, Apr. 1992.

		L= 8	L= 16	L= 24	L= 32
C. P. (T_{xor})	Prop.	9	9	9	9
	[13]*	3.5	7.03	10.2	13.63
	[12]*	4.167	7.769	11.111	14.034
XOR gates	Prop.	2012	4096	6125	8229
	[13]	2360	4032	8520	10592
	[12]	2845	5469	9532	12512

[6] K. K. Parhi, “Eliminating the fanout bottleneck in parallel long BCH encoders,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 51, no. 3, pp. 512–516, Mar. 2004.

[7] C. Cheng and K. K. Parhi, “High speed parallel CRC implementation based on unfolding, pipelining, retiming,” IEEE Trans. Circuits Syst. II, Expr. Briefs, vol. 53, no. 10, pp. 1017–1021, Oct. 2006.

[8] G. Campobello, G. Patane, and M. Russo, “Parallel CRC realization,” IEEE Trans. Comput., vol. 52, no. 10, pp. 1312–1319, Oct. 2003.

[9] J. H. Derby, "High speed CRC computation using state-space transformation," in Proc. Global Telecommun. Conf. (GLOBECOM'01), vol. 1, pp. 166–170.

[10] G. Albertengo and R. Sisto, "Parallel CRC generation," IEEE Micro, vol. 10, pp. 63–71, Oct. 1990.

[11] S. L. Ng and B. Dewar, "Parallel realization of the ATM cell header CRC," Comput. Commun., vol. 19, pp. 257–263, Mar. 1996.

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

[13] 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, Nov. 2009.

[14] R. J. Glaise, "A two-step computation of cyclic redundancy code CRC-32 for ATMnetworks," IBM J. Res. Devel., vol. 41, pp. 705–709, Nov. 1997.

[15] M. Ayinala and K. K. Parhi, "Efficient parallel VLSI architecture for linear feedback shift registers," in Proc. IEEE Workshop on SiPS, Oct. 2010, pp. 52–57.

[16] A. M. Patel, "A multi-channel CRC register," in Proc. AFIPS Conf. ,1971, vol. 38, pp. 11–14.

[17] 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.

[18] F. Liang and L. Pan, "A CRT-based BCH encoding and FPGA implementation," in Proc. Int. Conf. Inf. Sci. Appl. (ICISA), Apr. 2010, pp. 21–23.

[19] C. Kennedy, J.Manii, and J. Gribben, "Retimed two-step CRC computation on FPGA," in Proc. 23rd Canadian Conf. Elect. Comput. Eng. (CCECE), May 2–5, 2010, pp. 1–7.

[20] C. Toal, K. McLaughlin, S. Sezer, and X.Yang, "Design and implementation of a field programmable CRC circuit architecture," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 8, pp. 1142–1147, Aug. 2009.

[21] K. Septinus, T Le, U. Mayer, and P. Pirsch, "On the design of scalable massively parallel CRC circuits," in Proc. IEEE Int. Conf. Electron., Circuits Syst., Dec. 11–14, 2007, pp. 142–145.

[22] C. Kennedy and A. Reyhani-Masoleh, "High-speed CRC computations using improved state-space transformations," in Proc. IEEE Int. Conf. Electro/Inf. Technol., Jun. 7–9, 2009, pp. 9–14.

[23] A. Doring, "Concepts and experiments for optimizing wide-input streaming CRC circuits," in Proc. 23rd Int. Conf. Architect. Comput. Syst., Feb. 2010.

[24] Y. Do, S. R. Yoon, T. Kim, K. E. Pyun, and S. Park, "High-speed parallel architecture for software-based CRC," in Proc. IEEE Consumer Commun. Netw. Conf., Jan. 10–12, 2008, pp. 74–78.

[25] M. E. Kounavis and F. L. Berry, "Novel table lookup-based algorithms for high-performance CRC generation," IEEE Trans. Comput., vol. 57, no. 11, pp. 1550–1560, Nov. 2008.

[26] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation.