International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

Research Article

Comparison of Feed-forward and Feedback LFSR for High Reliability

E.K.Sharmitha[Á*], T.Sathya[Á] and B.S.Saromena Aarthi[Á]

[Á]ECE, Velalar College of Engineering and Technology, Erode, Tamil Nadu, India

Accepted 20 June 2014, Available online 30 June 2014, **Vol.4, No.3 (June 2014)**

Abstract

Error correction is one of the important technique for detecting and correcting errors in communication channels, memories etc., Errors are associated with all types of memories. But the NAND FLASH memories are competing in the market due to its low power, high density, cost effectiveness and design scalability. As far as the memory is concerned the testing should not consume more time. So, two types of LFSR is been designed to reduce the iteration bound and the critical path. Some DSP algorithms are used to overcome the delays by increasing the speed. BCH codes are widely been used for error detection and correction. The generated check bits of the BCH encoder are appended with the message bits to form a codeword. This codeword is sent to the receiver to detect any error during the transmission. One of the main components of BCH encoder is LFSR (Linear Feedback Shift Register). LFSR find its wider application in Built-in-Self-Test, signature analyzer etc., whereas here it is used to form parity bits to concatenate with message bits for the formation of a codeword. The main advantage of LFSR is that it is simple to construct and it operates at very high clock speed, but its main drawback is that the inputs are given in bit serial. To overcome these drawbacks. DSP algorithms such as unfolding and parallel processing are used by selecting the unfolding factor based on some design criteria. Selecting a better unfolding value reduces the sample period, decreases the clock cycle, and increases the speed, power and the throughput. Feed forward LFSR and Feedback LFSR are the two types of LFSRs discussed here. Each type has its own advantage and limitations. Different parameters like power, area, speed, throughput and clock cycles are compared for two types of LFSR .Based on the application any one type of LFSR can be used.

Keywords: *Bose Chaudhuri-Hocquengham (BCH), Cyclic Redundancy Check (CRC), Computational Time (CT), Feedforward LFSR (FF-LFSR), Feedback LFSR (FB-LFSR), Galois Field (GF), LFSR, MLC(Multi Level Cell),unfolding, sample period reduction*

1. Introduction

NAND Flash's high-density, low power, cost effectiveness, and scalable design make it an ideal choice to fuel the explosion of new multimedia products that are entering the market. Advances in system design techniques also enable the more cost effective NAND Flash to replace NOR Flash in a significant percentage of traditionally NOR Flash applications. NAND flash performs read and write simultaneously. Due to the efficient architecture of the NAND Flash, its cell size is almost half the size of a NOR cell. This enable NAND Flash architecture to offer higher densities with larger capacity on a given die size, in combination with a simpler production process[R. Bez *et al*, 2003].

This paper focus on NAND Flash memories since they have lower erase times, less chip area per cell which allows greater storage density, and lower cost per bit than NOR Flash memories. NAND is used or best suited for sequential data application. Physically, the NAND architecture uses smaller transistors, because it doesn't have to “pull-down” a whole bit-line. A NAND bit line is a series of transistors so each transistor only has to pass a small amount of current.

There are some inherent limitations of NAND Flash memories[J. Cooke, 2007]. These include write/read disturbs, data retention errors, bad block accumulation, limited number of writes, program disturb errors, soft errors and stress-induced leakage current . Because of these errors that occur in MLC NAND based flash memories various types of coding techniques can be applied for error detection and correction.

There are various codes available for error detection and correction. The codes are broadly classified into two types; they are a).block codes and b).conventional codes. Cyclic code is one of the classifications of block codes. The subset of the block code is BCH code.BCH code initially forms a generator polynomial by the use of finite field (GF) concept [R. Bez *et al*, 2003] and generates a parity (check) bits to be appended to the message bits to form a codeword [J. Cooke, 2007]. The main component of the encoder is simply a LFSR for generating the parity bit. The components used to form LFSR are simply registers and exor gates. Series combination of both registers and exor gates forms a LFSR. The main advantage of LFSR is it is simple to construct and it

*Corresponding author**: E.K.Sharmitha**

operates at very high clock speed, But the main drawback of the LFSR is that the bit stream applied to LFSR should be in serial. Hence, high-speed data transmission cannot be made possible. In order to increase the throughput and speed, parallel processing can be applied by unfolding concept. Parallel processing increases the number of message bits to be processed in a clock cycle (sample rate), but increases the area also.

Two types of LFSR architectures described here is feed forward LFSR (FF-LFSR) and feedback (FB-LFSR). The FF-LFSR has greater iteration bound. The sample period is reduced by the application of unfolding and its critical path is high when comparing to that of the FB-LFSR but, the clock cycle of the FF-LFSR is very much low comparing with FB-LFSR. Whereas the FB-LFSR has lower iteration bound and smaller critical path when compare to that of FF-LFSR.

Unfolding is a transformation technique, which describes J consecutive iterations of the original DSP program. Unfolding increases the Iteration bound $T_\infty$to JT_∞. In order to reduce the sampling period it is important to calculate the iteration bound before unfolding the system to select the unfolding factor. Many important cases such as CT >T_∞ and T_∞ is not an integer must be analyzed before selecting the unfolding factor. The selected unfolding value must make CT < T_∞ and T_∞ is an integer, which automatically reduces the sampling period. Unfolding is a transformation technique that can be applied to any DSP program to create a new program, which describes more than one iteration of the original program [W. Stallings, 2004]. Large number of iterations of an original program can be made by unfolding it by an unfolding factor.

The rest of the paper is organized as follows. Section II gives the brief summary of the Feed-forward LFSR and its design procedure. Section III contains the design procedure of Feedback LFSR for BCH (31,16) and criteria for selecting the unfolding factor to propose a new unfolded structure to reduce the sample period and gives the steps for unfolding the DFG of the LFSR. Section V analyses the data flow,area, power, iteration bound, critical path and clock cycle for FF-LFSR,FB-LFSR and for its different unfolding factors.

2. Feedforward LFSR

2.1 Design Of Feed forward LFSR for BCH(31,16) Encoder

BCH codes are subset of the Block codes. BCH codes belong to a powerful class of multiple error correcting codes [Naresh Reddy et al, 2012]. BCH codes are based on well-defined mathematical properties. These mathematical properties are based on the Galois Field or finite fields. The Finite field has the property that any arithmetic operations on field elements always have results in the field only [R. Bez et al, 2003]. To provide an excellent error correcting capability, the roots of the generator polynomial of the BCH codes have to be specified carefully. With a generator polynomial of g(x), a t- error correcting cyclic codes is the binary BCH codes, with a condition that g(x) must be the least degree polynomial over Galois Field GF(2). Steps for designing the generator polynomial of BCH (31,16) is explained below,

i). Choose an irreducible polynomial p(x) =$x^5 + x^2 + 1$
ii). Construct GF (2^5)
iii). Construct the minimal polynomial using the relation $\varphi_\beta(X)=\prod_{i=0}^{l-1}(x+\beta^{2^i})$ (1)

α:$x^5 + x^2 + 1$ = m1(x)
α^3: $x^5 + x^4 + x^3 + x^2 + 1$ = m2(x)
α^5: $x^5 + x^4 + x^2 + x + 1$ = m3(x)
where β be a non-zero element of GF(2^m).
iv). Form the generator polynomial using the relation g(x)=LCM(m1(x),m2(x),m3(x)). (2)

In this work, BCH (31, 16) is taken as an example and an encoder is designed by the FF-LFSR as a main component using the generator polynomial g(x)=$x^{15} + x^{11} + x^{10} + x^9+x^8 + x^7 + x^5 + x^3 + x^2 + x$+1 and LFSR is unfolded by an unfolding factor which is selected based on some design criteria discussed in theorem[W. Stallings, 2004] to improve the design methodology. LFSR architecture for BCH encoding is shown in Fig 3. Initially the 4-bit information(1000001) has to be appended with the must be made equal to the degree of the generator polynomial. Hence, the resultant message bit is 16 bit(1000001000000000) long, which is divided by the generator polynomial to form parity bits.hence it requires 16 clock cycles for generating the parity bit.

The parity bits that are obtained from LFSR is 100101000100010. This is systematic encoding, because information and check bits are arranged together so that they can be recognized in the resulting codeword.
General equation for codeword is,

i(x).x^{n-k}=q(x).g(x) +r(x) (3)
Where,
i (x):information bit polynomial.
q (x):quotient bit polynomial.
g (x):generator polynomial.
r (x):remainder polynomial.
Encoder of BCH(31,16) comprises of parallel in serial out shift register followed by the LFSR.

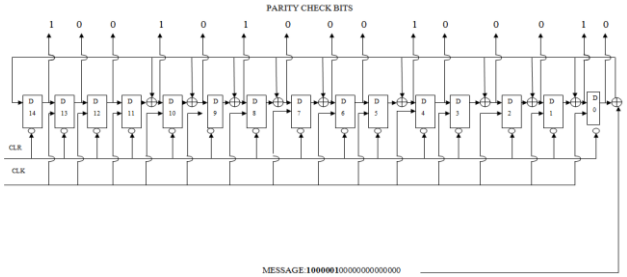

Fig.1: Existing LFSR Architecture for g(x) =1+ $x^{15} + x^{11} + x^{10} + x^9+x^8 + x^7 + x^5 + x^3 + x^2 + x$

The feed forward LFSR has only less number of delays in the feedback path. The first register's output and the input is computed and the result is feedback to the system. This

architecture generates the parity bits, if the message bit is made equal to that of the generator polynomial by appending zeros to the information bit. The architecture of the BCH(31,16) encoder's LFSR design is shown in Fig 1.

2.2 Unfolding LFSR

Unfolding algorithm is applied to the LFSR architecture in this design to increase the speed by reducing the clock cycle. The steps to be followed to unfold the encoder are,

i) Convert normal LFSR into DFG
ii) Calculate the iteration bound for the DFG

2.2.1 Formation of DFG for the LFSR

Often a DSP program is represented using the DFG. Here the nodes represent the computation and each of the node has its own computation time. The communication between the nodes is represented using edges. The DFG for the FF-LFSR is shown in Fig 2.

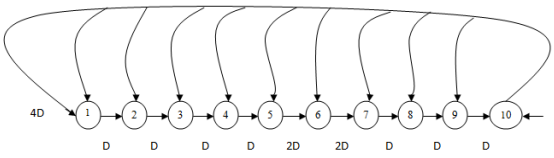

Fig 2: DFG of feedforward LFSR for g(x) =$x^{15}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{5}+x^{3}+x^{2}+x+1$

2.2.2 Calculation of Iteration Bound

Many of the DSP algorithms contain feedback loops, which impose an inherent fundamental lower bound on the achievable iteration or sample period. This bound is referred to as iteration bound. Iteration bound is the representation of the algorithm in the form of a DFG. Same algorithm but with different representations lead to different iteration bound. Iteration bound is defined as,

T_∞=max$_{l\in L}\{\frac{t1}{w1}\}$ (4)

Iteration bound is the maximum of loop bound

$T_\infty = max\{\frac{2}{1},\frac{3}{2},\frac{4}{7},\frac{5}{5},\frac{6}{7},\frac{7}{8},\frac{8}{9},\frac{9}{10},\frac{10}{11},\frac{10}{15}\}\}$

$T_\infty=\frac{2}{1}$ = 2u.t .

The iteration bound of the FF-LFSR is 2u.t. Now we can apply unfolding techniques to further reduce the sample period, critical path and power.

The loop 10 ⟶ 9 has the maximum loop bound. Synthesis report reveals that all the nodes of the DFG has CT of 1u.t. Since T_∞> CT, iteration period can be made equal to T_∞ so, any unfolding factor can be chosen in this case(i.e., T_∞> CT). Iteration bound of the unfolded DFG changes from$T_\infty toJT_\infty$. Where J stands for the unfolding factor. Similarly the sample period of the unfolded DFG is$\frac{T_\infty}{J}$. For this FF-LFSR , J=2 and 3 is chosen for comparing it with FB-LFSR.

2.3. Unfolding Algorithm

It is a transformation technique that can be applied to a DSP program in order to create a new program describing more than one iteration of the original program. Unfolding a DSP program is done by selecting an unfolding factor J, which describes J consecutive iterations of the original program. Loop unrolling is also called as unfolding [Naresh Reddy et al, 2012].

2.3.1 Algorithm Steps

1. For each node U in the original DFG, draw J nodes $U_0, U_1, U_2, \ldots\ldots, U_{J-1}$
2. For each edge U ⟶ V with w delays in the original DFG, draw the J edges $U_i \longrightarrow U_{(i+w)\%J}$ with $\lfloor \frac{i+w}{J} \rfloor$ delays for i = 0,1,……J-1. (5)

By this technique the speed of the LFSR is increased automatically by reducing the clock cycle. The main drawback of unfolding is that the area of the system increases and choosing a large value of unfolding factor leads to hardware complexity. After applying the unfolding technique with unfolding factor J=3 and 2 for Fig 1, three parallel and two parallel architectures are obtained and it is shown in Fig 3 and 4.

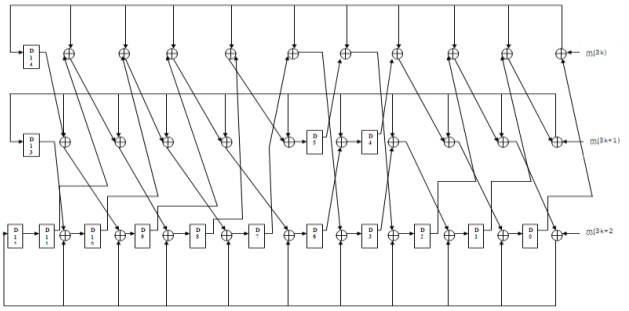

Fig. 3: Three parallel FF-LFSR for g(x) =$x^{15}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{5}+x^{3}+x^{2}+x+1$ after Unfolding by a factor of 3

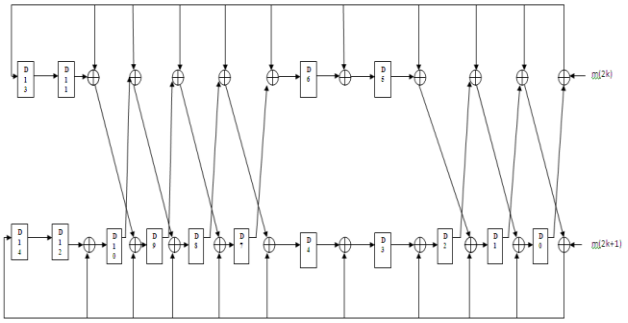

Fig 4: Three parallel LFSR for g(x) =$x^{15}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{5}+x^{3}+x^{2}+x+1$ after Unfolding by a factor of 2

After the application of unfolding, the clock cycle is reduced. This in turn increases the speed, hence automatically the speed is increased and the iteration bound is increased from 2 to 4 for J=2 and 2 to 6 for J=3.

FF-LFSR has 2Txor critical path and has larger iteration bound (T_∞).Pipelining must be done for this structure to reduce the critical path of this LFSR.

3. Feedback LFSR

Compare to feed forward LFSR, the feedback LFSR has the critical path of Txor and the iteration bound is less when compared with the feed forward LFSR architectures.

This unfolded LFSR architecture uses sample period reduction technique to achieve more speed. In order to achieve this objective, some important criteria has to be considered for selecting the unfolding factor to improve the design methodology. Data flow table for the proposed system and conventional system is tabulated and compared then, area analysis and power analysis of the FF-LFSR and FB-LFSR is graphically shown to analyze the depth on the hardware and power overhead. Different levels of unfolding factors are introduced to check the hardware complexity and speed of the design.

3.1 Design of FB-LFSR for BCH(31,16)

FB-LFSR generates parity bit for 22 bit information. Initially the information bit must be added with the generator polynomial degrees and the required amount of zeros must be appended to the message bit to form the information bit stream(1000001000000000000000).this information bit stream is divided by the generator polynomial for generating the parity bit. Hence it requires 22 clock cycle for forming the parity bit. The FB-LFSR design for BCH(31,16) is shown in Fig 5.

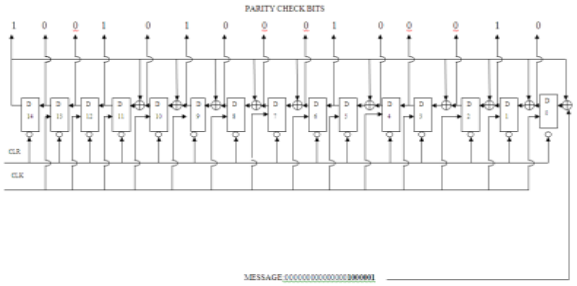

Fig. 5: LFSR Architecture for g(x) $=x^{15}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{5}+x^{3}+x^{2}+x+1$

3.2 Unfolding LFSR

Unfolding algorithm is applied only to the LFSR architecture in this design to increase the speed by reducing the sampling period.
The steps to be followed to unfold the encoder are,

iii) Convert normal LFSR into DFG
iv) Calculate the iteration bound for the DFG
v) If T_∞< CT of a node, apply unfolding to make the sampling period to be equal to T_∞. This technique is called as sample period reduction.

3.2.1 Formation of DFG for the LFSR

Often a DSP program is represented using the DFG.

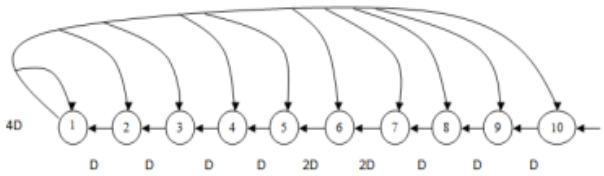

Fig.6:DFG of LFSR for g(x) $=x^{15}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{5}+x^{3}+x^{2}+x+1$

Here the nodes represent the computation and each of the node has its own computation time. The communication between the nodes is represented using edges. The DFG for the FB-LFSR is shown in Fig 6.

3.2.2 Calculation of Iteration Bound

Many of the DSP algorithms contain feedback loops, which impose an inherent fundamental lower bound on the achievable iteration or sample period. This bound is referred to as iteration bound. Iteration bound is the representation of the algorithm in the form of a DFG. Same algorithm but with different representations lead to different iteration bound. Iteration bound is defined as,

$$T_\infty=\max_{l\in L}\{\frac{t1}{w1}\} \quad (4)$$

Iteration bound is the maximum of loop bound
$T_\infty=max\{\frac{1}{4},\frac{2}{5},\frac{3}{6},\frac{4}{7},\frac{5}{8},\frac{6}{10},\frac{7}{12},\frac{8}{13},\frac{9}{14},\frac{10}{15}\}$
$T_\infty=\frac{10}{15}=\frac{2}{3}$.

The iteration bound of the FF-LFSR is 2u.t.whereas Without applying the sample period reduction technique the FB-LFSR architecture has a lower iteration bound of 0.66. The critical path of the FB-LFSR is only Txor. Now we can apply unfolding techniques to further reduce the sample period, critical path and power.

3.2.1 Sample Period Reduction

The loop 10 ⟶ 9 ⟶ 8⟶ 7 ⟶6 ⟶5 ⟶4 ⟶3 ⟶ 2 ⟶ 1 has the maximum loop bound. Synthesis report reveals that all the nodes of the DFG has CT of 1u.t. Since T_∞< CT, iteration period cannot be made equal to T_∞. In such a case retiming can be applied but it cannot be used to reduce the CT of the critical path of the DFG to T_∞. Selection of the unfolding factor is an important criterion in sample period reduction. Unfolding factor is chosen using the relation, J=$\lceil\frac{t_u}{T_\infty}\rceil$ = 2. Iteration bound of the unfolded DFG changes from$T_\infty toJT_\infty$. where J stands for the unfolding factor. Similarly the sample period of the unfolded DFG is$\frac{T_\infty}{J}$. One more case exist is, if $T_\infty$is not an integer. The LFSR of g(x) $=x^{15}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{5}+x^{3}+x^{2}+x+1$ satisfies both the cases. Because its CT is greater than the iteration bound and the iteration bound is not an integer. Hence J must be selected in such a way that JT_∞ is an integer and JT_∞> node CT. The only value of J that satisfies both the condition is 3. This is clearly specified by a theorem[W. Stallings, 2004].

3.3 Unfolding Algorithm

After applying the unfolding technique for FB-LFSR with unfolding factor J=3 and 2 for Fig 3, three parallel and two parallel architectures are obtained and it is shown in Fig 7 and 8. Unfolding increases the total hardware size. The total exor gate count is increased whereas the register count remains constant. Hence power is increased for the greater unfolding values.

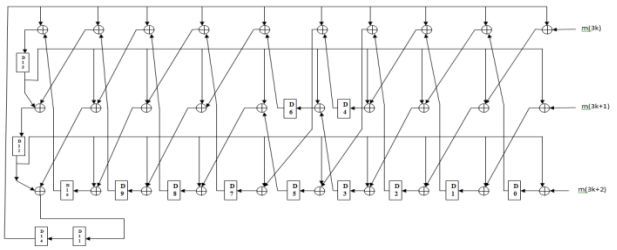

Fig. 7: Three parallel LFSR for g(x) =$x^{15}+x^{11}+x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+x+1$ after Unfolding by a factor of 3

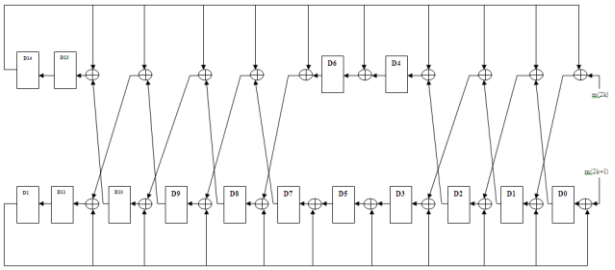

Fig. 8: Three parallel LFSR for g(x) =$x^{15}+x^{11}+x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+x+1$ after Unfolding by a factor of 3

After the application of unfolding, the sample period is reduced and the iteration bound is increased from 0.66 to 1.32. So that J T_∞> CT. This sample period reduction is one of the application of the unfolding algorithm.

4. Results and Analysis

Initially the codeword is formed by the generation of parity bits 100100010100010. This parity bit formation is coded in VHDL. Each of the unfolded architecture is coded in VHDL, simulated and implemented using Xilinx92i to analyze the area and speed. For the message bits: 0000000001000001 and for the generator polynomial g(x) =$x^{15}+x^{11}+x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+x+1$ the normal BCH encoder using FF-LFSR and FB-LFSR simulation result is shown in Fig 10 and 11. The same architecture but with unfolding factor of 2 and 3 is simulated and verified as shown in Fig 11 and 12. From the results shown in Table 1, 2, 3, 4,5 and 6, it is clear that the clock cycle of the FF-LFSR decreases from 16 to 8 and 5 for the unfolding factors J=2 and J=3 and for the FB-LFSR it is reduced from 22 to 11 and 8 for the unfolding factors J=2 and J=3because of sample period reduction technique in . Hence unfolding speed up the LFSR operation by decreasing the clock cycle. As far the memory is concerned, the error detection and correction must not take much time because it decreases the throughput of the system.

4.1 Data Flow Table

Table 1: Data flow table for FF-LFSR

Clock	Message bit	y(14 to 0)
1	1	100011111010111
2	0	110010000111100
3	0	011001000011110
4	0	001100100001111
5	0	100101101010000
6	0	010010110101000
7	1	101010100000011
8	0	110110101010110
9	0	011011010101011
10	0	101110010000010
11	0	010111001000001
12	0	101000011110111
13	0	110111110101100
14	0	011011111010110
15	0	001101111101011
16	0	100101000100010

Table 2: Data flow table for FB-LFSR

Clock	Message bit	y(14 to 0)
1	1	000000000000001
2	0	000000000000010
3	0	000000000000100
4	0	000000000001000
5	0	000000000010000
6	0	000000000100000
7	1	000000001000001
8	0	000000010000010
9	0	000000100000100
10	0	000001000001000
11	0	000010000010000
12	0	000100000100000
13	0	001000001000000
14	0	010000010000000
15	0	100000100000000
16	0	000110110101111
17	0	001101101011110
18	0	011011010111100
19	0	110110101111000
20	0	101010101011111
21	0	010010100010001
22	0	100101000100010

Table 3: Data flow table for FF-LFSR with unfolding factor of 3

Clock	m(3k)	m(3k+1)	m(3k+2)	y(14 to 0)
1	100			011001000011110
2	000			010010110101000
3	100			011011010101011
4	000			101000011110111
5	000			001101111101011

Table 4: Data flow table for FB-LFSR with unfolding factor of 3

Clock	m(3k)	m(3k+1)	m(3k+2)	y(14 to 0)
1	100			000000000000100
2	000			000000000100000
3	100			000000100000100
4	000			000100000100000
5	000			100000100000000
6	000			011011010111100
7	000			010010100010001
8	000			100101000100010

Table 5: Data flow table for FF-LFSR with unfolding factor of 2

Clock	m(2k) m(2k+1)	y(14 to 0)
1	10	110010000111100
2	00	001100100001111
3	00	010010110101000
4	10	110110101010110
5	00	011011010101011
6	00	010111001000001
7	00	110111110101100
8	00	100101000100010

Table 6: Data flow table for FB-LFSR with unfolding factor of 2

Clock	m(2k) m(2k+1)	y(14 to 0)
1	10	000000000000010
2	00	000000000001000
3	00	000000000100000
4	10	000000010000010
5	00	000001000001000
6	00	000100000100000
7	00	001000001000000
8	00	000110110101111
9	00	011011010111100
10	00	110010101011111
11	00	100101000100010

4.2 Area, Clock Cycle and Power Analysis

The critical path and iteration bound of the FF and FB-LFSR is shown graphically in Fig 9 .Power analysis of FF and FB-LFSR is shown graphically in Fig 10 and 11. The design is analyzed for different levels of unfolding factors in order to discuss the hardware and power overhead involving in different parallelism levels. The power analysis, delay and device utilization analysis of different unfolding factors in FF-LFSR and FB-LFSR is shown graphically in Fig 9, 10, 11, 12, 13 and 14. These analysis are is done by implementing the LFSR in XilinxISE9.2i

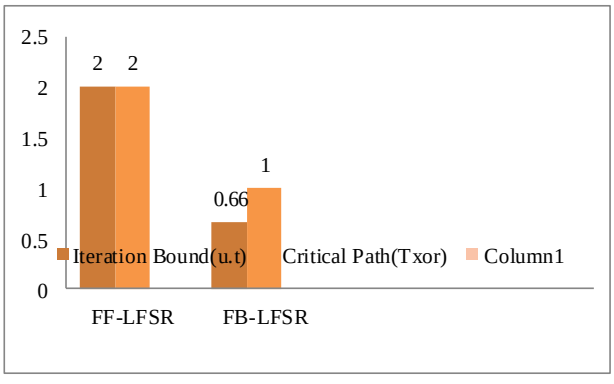

Fig 9: Comparison of iteration bound and critical path for FF and FB- LFSR

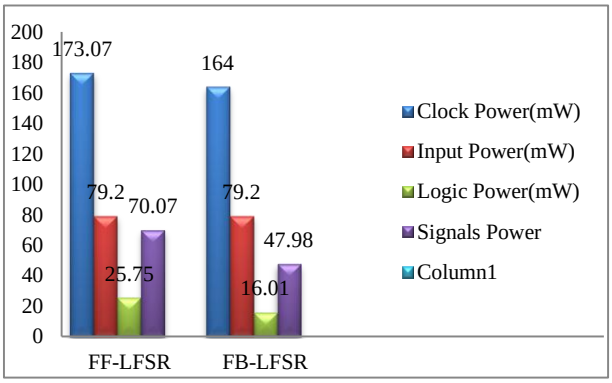

Fig 10: Comparison of different powers for FF and FB-LFSR

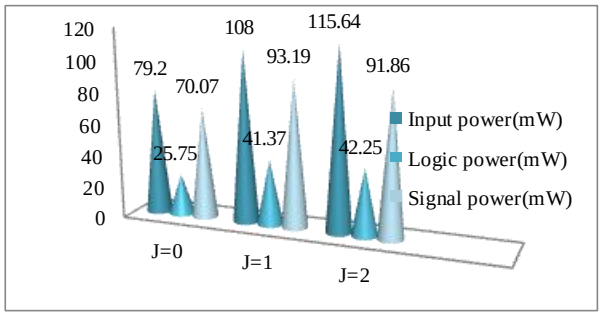

Fig 11: Comparison of various powers for different unfolding factors for FF-LFSR

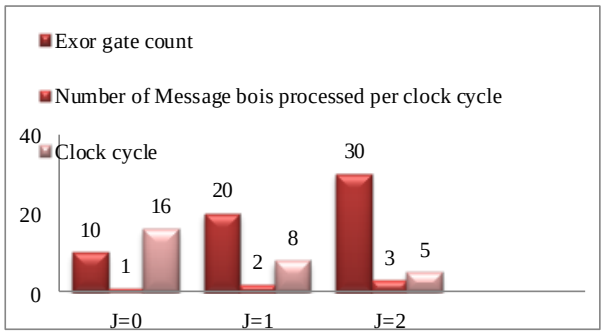

Fig 12: Comparison of Area and Speed parameters for different unfolding factors for FF-LFSR

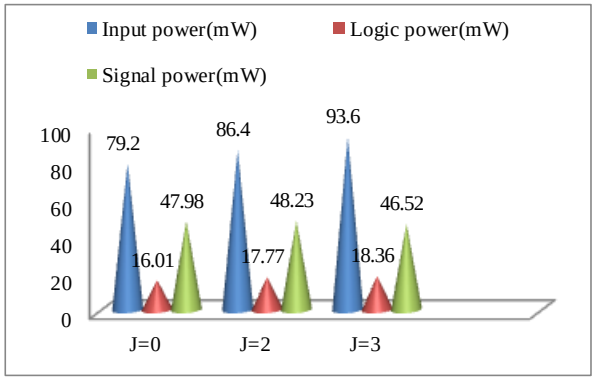

Fig 13: Comparison of various powers for different unfolding factors for FB-LFSR

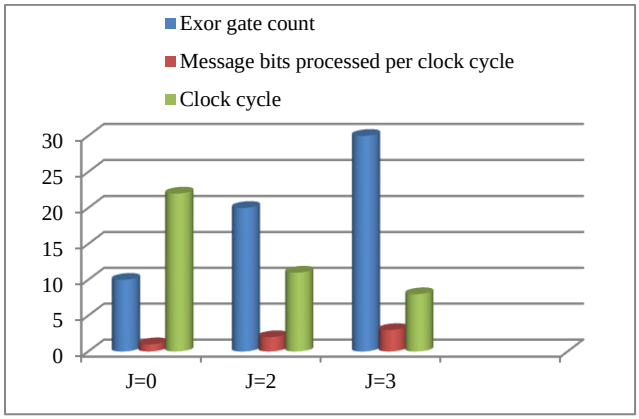

Fig 14: Comparison of Area and Speed parameters for different unfolding factors for FB-LFS

4.3 Screen Shots

The Simulation of the FF-LFSR and FB-LFSR without unfolding and after unfolding by a factor of 2 and 3 is shown in the Fig 15, 16, 17, 18 and 19

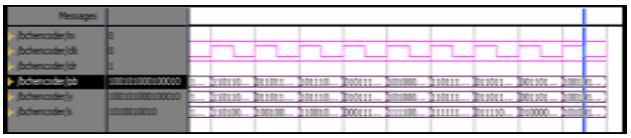

Fig 15: FF- LFSR before unfolding

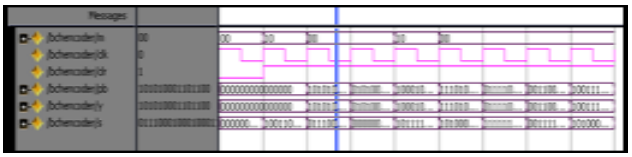

Fig 16: FF-LFSR after unfolding with J= 2

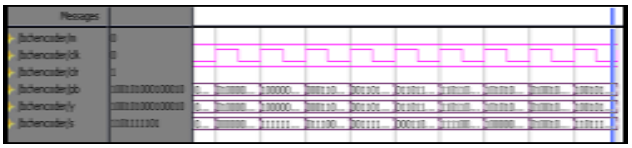

Fig 17: FB-LFSR before Unfolding

Fig 18: FB-LFSR after Unfolding with J=3

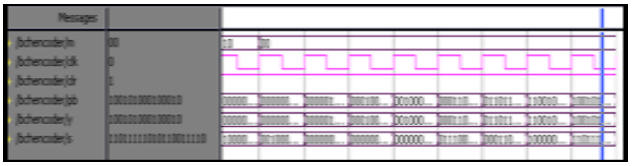

Fig 19: FB-LFSR after Unfolding with J=2

Conclusion

Since the NAND FLASH memories require less delay encoders, a high throughput encoder is designed by analyzing the operation of the two types of LFSRs and based on the application anyone type can be selected .Moreover area , clock cycle and power is analyzed by simulating the design in ModelSim tool by VHDL language and by implementing the design in XilinxISE9.2i.The obtained results reveal that the critical path, power and the iteration bound of the FB-LFSR is lower when compare to that of the FF-LFSR .the clock cycle for the FF-LFSR is less when comparing with the FB-LFSR.Hence depending on the application in memories any LFSR design can be adopted to develop a reliable encoder for MLC NAND based FLASH memories

Future Work

Different-pipelining techniques can be introduced to reduce the critical path of the encoder of BCH. Retiming also can be applied to further increase the speed and to reduce the power consumption and area.

Acknowledgment

The authors acknowledge the contributions of the students, faculty of Velalar College of Engineering and Technology for helping in the design of test circuitry, and for tool support. The authors also thank the anonymous reviewers for their thoughtful comments that helped to improve this paper. The authors would like to thank the anonymous reviewers for their constructive critique from which this paper greatly benefited.

References

R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti (Apr. 2003.) Introduction to flash memory, Proc. IEEE, vol. 91, no. 4, pp. 489–502.

J. Cooke (2007), The inconvenient truths about NAND flash memory, presented at the Micron MEMCON Presentation, Santa Clara, C.

William Stallings (2004), Cryptography and Network Security-Principles and Practices, Introduction to Finite Fields, 3rd edition.

Ranjan Bose, Information Theory, Coding and Cryptography.

K.K.Parhi VLSI Digital Signal Processing Systems-Design And Implementation.

Wei Liu, Junrye Rho, and Wongong Sung, Low- Power High throughput BCH error correction VLSI Design for Multi-Level cell NAND Flash Memories.

Keshab K. Parhi (march 2004.), Eliminating the Fan out Bottleneck in Parallel Long Bch Encoders in proc IEEE, vol.51.No.3.

Naresh Reddy, B.Kiran Kumar and K.monishaSirisha (2012), On the Design of High Speed Parallel CRC Circuits Using DSP Algorithms in IJCSIT, vol.3 (5).

Chao Cheng and KeshabParhi (October 2006.), High-Speed Parallel CRC Implementation Based On Unfolding, Pipelining And Retiming, in proc, IEEE, vol.53, No.10.

John G.Proakis Masoud Salehi (2008),Digital-Communications-Linear block codes, cyclic codes, BCH codes, Reed-Solomon codes, 5th Edition.