Design of High-Speed VLSI Architecture for LFSR with Maximum Length Feedback Polynomial

Meenal Dadhe¹ Prof. Anup R. Nage²
¹²Department of Electronics and Communication Engineering
¹²GHERAE, RTMN University Nagpur, India

Abstract— The main purpose of high-speed architecture of linear feedback shift register (LFSR) based on PN Sequence generator technique. PN sequences used for communication purpose cryptography application and for designing encoder, decoder. Depend on the feedback polynomial total number of random sequence generator on LFSR. It is simple counter so its count maximum of 2^n-1 by using maximum feedback polynomial. Here in this 8, 32 bit LFSR designed by using VHDL, we implement LFSR to study the performance and analysis the behavior of randomness. We also designed proposed architecture which is based on Galois implementation. The analysis is conceded out to find number of gates, memory and speed requirement as the number of bits is increased. We proposed LFSR architecture based on serial, parallel, and also design one application of LFSR i.e. BIST for Adder to minimize delay of the system.

Key words: VLSI, LFSR

I. INTRODUCTION

Linear feedback shift register (LFSR) is widely used for generating data encryption keys, random numbers are very much useful in the various applications such as communication channel, bank security, etc.[1][3] It is used to design encoder and decoder for sending and receiving data in noisy communication channel. Random number generator is a device to generate a random sequence of numbers i.e 0 or 1. There are various methods for pseudo-random numbers are known.[1] Most of them are based on linear congruential equations and require a number of time consuming arithmetic operations. In contrast, the use of feedback shift registers permits very fast generation of binary sequences. Shift register sequences of maximum length (m-sequences) are well suited to simulate truly random binary sequences. With minimum length feedback polynomial 8, and 32-Bit LFSR based PNRG design. As we change the feedback polynomial the run-length as well randomness also changes. Here we have design 8, and 32 bit length sequences using VHDL with maximum feedback Polynomial and also design types of LFSR i.e. Galois type and Fibonacci type with 32 bit to understand the memory utilization and speed requirement. In this paper also design the application of LFSR i.e. BIST for Adder. We have presented the comparison of performance analysis based on synthesis and simulation result as well identify the simulation problem for long bit LFSR. The target device we have used Xilinx Vertex 5 and performed simulation and synthesis using Xilinx ISE 13.1.

II. LINEAR FEEDBACK SHIFT REGISTER

Linear Feedback Shift Register (LFSR) is a popular technique for generating a pseudorandom bit stream. Linear feedback shift registers are electronic circuits that generate linear recurring sequences. An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit (see Figure 1)[1][7]. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops as shown in Figure.

Fig. 1: Basic Block Diagram of LFSR

III. PSEUDORANDOM PATTERN GENERATION

Linear feedback shift registers make extremely good pseudorandom pattern generators. When the outputs of the flip-flops are loaded with a seed value (anything except all 0s, which would cause the LFSR to produce all 0 patterns) and when the LFSR is clocked, it will generate a pseudorandom pattern of 1s and 0s. Note that the only signal necessary to generate the test patterns is the clock.

IV. LFSR IMPLEMENTATION

Linear feedback shift registers can be implemented in two ways; one is the Fibonacci implementation or simple shift register generator (SSRG) and the other is the Galois implementation or multiple return shift generator (MRSRG) [11].

A. Galois Implementation

In Galois Implementation, the data flow is from left to right and the feedback path is from right to left. The polynomial increments from left to right with X 0 term (the "1" in the polynomial) as the first term. This is referred to as a Tap polynomial, as it indicates which taps are to be fed back from the shift register. Since the XOR gate is in the shift register path, the Galois implementation is also known as an in-line or modular type (Mtype) LFSR. Figure 3.1 The shift register is initially loaded with bits called the seed value (any value except all zeroes) and then clocked.

Fig. 2: Galois Implementation
B. Fibonacci Implementation

In the Fibonacci implementation[11], the data flow is from left to right and the feedback path is from right to left, similar to the Galois implementation. However, the Fibonacci implementation polynomial decrements from left to right with X 0 as the last term in the polynomial. This polynomial is referred to as a Reciprocal Tap polynomial and the feedback taps are incrementally annotated from right to left along the shift register. Since the XOR gate is in the feedback path, the Fibonacci implementation is also known as an out-of-line or simple type (S-type) LFSR [14, 15].

Figure

An interesting fact about Galois implementation and Fibonacci implementation is that the order of the Galois weights is opposite that of the Fibonacci weights. If identical feedback weights are given, the two linear feedback implementations will produce the same sequence. However, the initial states of the two implementations must necessarily be different for the two sequences to have identical phase. When implemented in hardware, modulo-2 addition is performed using EXCLUSIVE-OR (XOR) gates. The Galois form is generally faster than the Fibonacci in hardware due to the reduced number of logic gates in the feedback loop [11].

V. APPLICATION

There are several applications involving the use of LFSRs. These applications include Built in Self Test (BIST) circuits, encryption key generation, and pseudorandom number generators. The following sections describe the BIST For Adder applications in more detail.

A. Built in Self Test

Built in self-test is the circuit which test the circuit itself. This increases the Controllability and Observability of the circuit. In BIST, the test pattern generation and the output response evaluation are done on chip itself so the hardware used for designing BIST should be minimized. The various test pattern generated by LFSR. Input test vectors are binary patterns applied to the inputs of the CUT and the resulting output patterns are observed on the outputs of the CUT. Using a comparator output responses are checked against the expected correct response data, which is obtained through simulation.[13]

Fig. 4: Built in Self Test

If all output responses match the accepted response data, the CUT has passed the test and it is 1 fault-free circuit. Based on the techniques how the test vectors are applied to the CUT and how the output responses are compared, there are two main directions to test electronic circuits: external testing using automatic test equipment (ATE) and internal testing using built-in self-test (BIST). When external testing is employed, the input test vectors and correct response data are stored in the ATE memory. Input test vectors are generated using ATPG tools, while correct response data is obtained through circuit simulation. For external testing, the comparison is carried out on BIST is a design-for-test (DFT) method where part of the circuit is used to test the circuit itself. BIST needs only an inexpensive tester to initialize BIST circuitry and inspect the final results (pass/fail and status bits).

B. Ripple-Carry Adder

It is possible to create a logical circuit using multiple full adders to add N-bit numbers. Each full adder inputs a carry-in, which is the Cn-1 of the previous adder. This kind of adder is called a ripple-carry adder, since each carry bit “ripples” to the next full adder. Note that the first (and only the first) full adder may be replaced by a half adder (under the assumption that C0n = 0). The layout of a ripple-carry adder is simple. The following diagram shows a four-bit adder, which adds the numbers A[4:0] and B[4:0], as well as a carry input, together to produce S[4:0] and the carry output.

Fig. 5: Ripple-Carry Adder

VI. LITERATURE REVIEW

An exhaustive literature review has been carried out related to the work to find out the current research. Abstracts of some of most relevant research works are reported in the following paragraph.


In this paper transformation achieved by a full-speed compare to the serial architecture. A novel method of high speed parallel implementation of linear feedback shift register based parallel IIR filter design, pipelining and retiming algorithm is proposed.
B. Chan-Bok Jeong and Dae-Ho Kim, “High-Speed Architecture for k-dimensional LFSR in H/W Implementation” Electronics and Telecommunication Research Institute Daejeon, Korea 2011 IEEE

The analysis of the proposed LFSR architecture demonstrates that the proposed k-dimensional LFSR architecture is k times as fast as a conventional LFSR architecture and the used processing time for scrambling is enough to implement scramble function for high-speed applications such as LTE-Advanced. The architecture of k bits-in and k bits-out is called k-dimension as the proposed LFSR structure. It is possible to transform the 1-dimensional LFSR architecture into the k-dimensional LFSR architecture.


LFSR based PN Sequence Generator technique is used for various cryptography applications and for designing encoder, decoder in different communication channel. It is more important to test and verify by implementing on any hardware for getting better efficient result. The total number of random state generated on LFSR depends on the feedback polynomial. As it is simple counter so it can count maximum of $2^n - 1$ by using maximum feedback polynomial. Here in this paper implemented 8, 16 and 32-bit LFSR on FPGA by using VHDL to study the performance and analysis the behavior of randomness. The analysis is conceded out to find number of gates, memory and speed requirement in FPGA as the number of bits is increased. In this paper delay in 8bit, 16bit, and 32bit LFSR is $7.271ns$ and number of slices is 4, 9, 18[7].


A new framework to construct fast and efficient pseudo-random (PN) sequence generation for bit scrambling, called a J-delayed and K-dimensional linear feedback shift register (JLFSR) has been introduced in [45]. In the proposed framework, the state of a J-shifted linear feedback shift register using one clock and K bit multiple outputs of a linear feedback shift register each clock cycle for scrambling/descrambling of large coded bits using an output of linear feedback shift register has been generated. JLFSR is highly relevant for the scrambling/descrambling process for a high-speed mass data transmission in an LTE Advanced system, as it has fast computation and supports clock-based processing. Simulation results have verified that the proposed method reduces the processing time used for generating PN sequences from $(J + DL)$ clocks to $(1 + DLK)$ clocks as compared with a conventional linear feedback shift register, where $DL$ denotes the length of a data stream.


The main purpose of this paper [36] has been to study the FPGA implementation of two 16 bit PN sequence generator namely linear feedback shift register and Blum-Blum-Shub (BBS). The logic of PN Sequence Generator presented here can be changed any time by changing the seed in linear feedback shift register or by changing the key used in BBS. The analysis has been conducted to find number of gates, memory and speed requirement in FPGA for the two methods. In this paper CPU time is taking for 4-Bit LFSR is 0.39 sec where as for BBS it is 4.11 sec. Memory utilization is more in BBS than LFSR as the number of FFs used in BBS are more[6].

VII. CONCLUSION

In this paper design 8 and 32 bit parallel and Serial LFSR Architecture with primitive polynomial for maximum length sequence. Design purposed architecture based on Galois Linear feedback shift register to minimize the delay of LFSR. Design one LFSR application ie. BIST for Adder.

REFERENCES

