VLSI Implementation of a Non-Linear Feedback Shift Register for High-Speed Cryptography Applications

Pey-Chang Kent Lin and Sunil P. Khatri
Department of ECE, Texas A&M University
College Station, TX 77843
k1arte@neo.tamu.edu and sunilkhatri@tamu.edu

ABSTRACT

For secure high data-rate communications, fast key generation algorithms are crucial. In this paper, we present a VLSI implementation of a Non-Linear Feedback Shift Register (NLFSR) for cryptography applications. Unlike existing cryptographic key generation techniques, our NLFSR generates multiple (64 in our implementation) key bits in each clock cycle. This enables its use in secure, high speed communications. Our NLFSR is implemented using a plurality (3 in our implementation) of LFSRs. The outputs of 64 bits from each LFSR are combined using 64 encoded majority functions, where the majority function used for any bit is changed at every clock cycle. We demonstrate that our NLFSR can generate keys which may be used for OC-768 optical fiber communication, which operates at 40 Gbps. The keys from our NLFSR pass all the tests in the NIST suite, which is a defacto benchmark used in industry to evaluate the quality of ciphers.

Categories and Subject Descriptors

B.2 [**Arithmetic and Logic Structures**]: General; E.3 [**Data Encryption**]: Public key cryptosystems

General Terms

Algoriths, Design, Security

Keywords

NLFSR, pseudo-random sequence, stream cipher

1. INTRODUCTION & PREVIOUS WORK

In high-speed, secure data communication (such as optic fiber based communication), data-rates can be extremely high. For example, the OC-768 standard for digital communication on optic fibers operates at 40Gbps. Cryptography key generation at these speeds is impossible for approaches which generate one cryptographic key bit per clock cycle. This paper proposes a Non-linear Feedback Shift Registers (NLFSR) based approach to generate cryptographic key bits for such applications.

In secure communications applications, it is customary to generate a pseudorandom sequence (called a *key*), and combine the key with the message to be sent (also called the *plaintext*). The resulting message is referred to as the *ciphertext*, and is transmitted to the recipient. The algorithm used to combine the key and the plaintext is referred to as a *cipher*. *Block ciphers* can operate on blocks of plaintext, while *stream ciphers* operate on the plaintext in a streaming fashion. The ciphertext is ideally undecipherable by unintended listeners. The intended receiver would decrypt the plaintext message by using the same key that was used during transmission. A common cipher uses a bit-wise XOR operation of key bits and plaintext bits, as shown in Figure 1. The original message is decrypted using the same cipher and key that was used during encryption.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GLSVLSI'10, May 16–18, 2010, Providence, Rhode Island, USA.
Copyright 2010 ACM 978-1-4503-0012-4/10/06 ...$10.00.

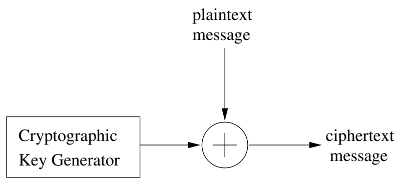

Figure 1: Cipher structure

Pseudo-random number sequences are often generated using linear feedback shift registers (LFSRs). LFSRs are attractive due to their simple implementation in hardware, and their fast performance. LFSRs possess several appealing mathematical properties. For example, LFSRs of length n whose characteristic polynomials yield maximum length sequences (m-sequences) cycle through all $2^n - 1$ nonzero states before repeating a state. Each bit of the LFSR state exhibits a uniform probability of 0 and 1, in a random sequence [3]. As a consequence, LFSRs are commonly used in VLSI testing and other non-secure applications. The main drawback of LFSRs in the context of cryptography is their linearity – through cryptanalysis, the structure of an n-bit LFSR can be inferred by observing $2n$ consecutive bits of its sequence [14]. Therefore, LFSRs cannot directly be used in cryptography applications.

Cryptographic key generation is therefore done using a non-linear feedback shift register (NLFSR). Such a NLFSR either utilizes a non-linear function for the next state feedback and/or for the output. For NLFSRs with non-linear feedback functions, Fibonnaci or Galois implementations are commonly used. These types of NLFSR have been shown to be more secure than LFSR in cryptanlaysis attacks [5]. In a Fibonacci implementation, all bits except the last is shifted each cycle as in a normal shift register. The last bit is updated according to a non-linear function of the previous bits. The Galois implementation can be considered a generalization of the Fibonacci implementation. Each bit in a Galois implementation is updated according to its next-state function, which a non-linear function of the previous bit and up to k other bits. Galois is faster than Fibonacci because the propagation time for smaller function of individual bits in Galois is reduced compared with the large feedback function in Fibonacci. However, the mathematical properties of such NLFSRs are harder to reason about. The construction of the feedback function for large NLFSRs, with guaranteed maximal length sequences is an extremely difficult task [7].

Hence most NLFSRs implement strong pseudo-random sequences using one or more LFSRs by combining the states of these LFSRs, or by clocking the LFSRs in a modified manner (such as in the A5 cryptographic key generator [12]). A combining based scheme takes the outputs of several LFSRs (or different memory elements of a single LFSR) to produce the NLFSR output. The clock controlled NLFSRs use irregular clocking of one or more of the LFSR(s), based on some subgenerator or based on some function of the LFSR state. As an example of the latter method, the A5 cryptographic key generator consists of three LFSRs, where a specific *clocking bit* from each LFSR is used to compute a majority output. The clocking of each LFSR is done only if its clocking bit matches the majority output. The output of the A5 cryptographic key generator is the XOR of the last bit of each LFSR [12]. General problems with clock-controlled schemes are their reduced periods and hence an increased linearity of the NLFSR output.

If a NLFSR is derived using LFSRs, then the mathematical properties of LFSRs can be leveraged to derive analytical guarantees of the NLFSR performance. A LFSR also has the advantage of the high speed and ease of implementation. In this paper, we therefore derive our NLFSR by using a plurality of LFSRs, whose states are combined using a non-linear function, to generate the NLFSR state.

For our approach, we introduce an alternative type of block cryptographic key generator. Unlike stream ciphers which output 1 bit per clock cycle, our NLFSR outputs n bits per cycle. Two of the most well known block ciphers are Data Encryption Standard (DES) and Advanced Encryption Standard (AES), both developed for the US government. DES takes a 64-bit block size of plain text and transforms it through a series of iterative subsitution and permutation functions (Fiestel Network) into ciphertext of same size [4]. DES is considered insecure due to its relatively short key length (56-bit key), but still useful for some applications in RFID and financial transactions [11]. AES is considerably more secure with a block size 128-bit and key-size of 128, 192, or 256 bits. It is based on substitution permutation network and also converts a block of plain text into ciphertext and through a series of substituion, shift, mix, and combination tranformations. Hardware evaluation results show throughput of DES = 1161.31 Mbps and Rijndael [6] (AES candidate) = 1950.03 Mpbs (0.35μm process) [8]. Area in number of gates are DES = 54,405 and AES = 612,834, which is large due to complexity of the tranform functions. In contrast, our NLFSR has been simulated at speeds supporting 40 Gbps throughput and with lower device count, estimated $\sim$ 40,000. Hence our NLFSR can be used in high bandwidth secure communications such as OC-768 optic fiber based communication (which has a data-rate of 40 Gbps). This is accomplished by combining n bits from three LFSRs using multiple encoded majority (non-linear) functions. A fourth LFSR (which is clocked at the same rate as the other 3 LFSRs) randomly selects the encoded majority function that is used to generate the i^{th} bit of the NLFSR.

Cryptographic key generators are typically evaluated using the NIST test suite [1], a standard benchmark used in industry. We test our NLFSR against this test suite, and have found that the results pass each of the 12 unique tests (and the 162 variants of these tests) from the NIST test suite. The tests in the NIST test suite were developed to evaluate AES candidates for securing sensitive government data [13]. These are among the most stringent tests available in industry, and are routinely used to test cyptographic algorithms.

The key contributions of this paper are:

- We have developed a NLFSR for block cryptographic key generation. Our NLFSR generates multiple ($n = 64$) output bits per clock cycle. The NLFSR consists of 3 LFSRs, whose i^{th} bits are combined with an encoded majority function to generate the i^{th} NLFSR bit. A fourth LFSR is used to change the encoded majority function used to generate the i^{th} NLFSR bit, at each clock cycle.

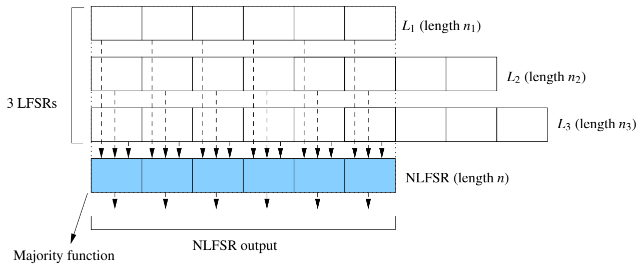

Figure 2: Block Diagram of our n-bit NLFSR

- The NLFSR utilizes 3 LFSRs (as well as a fourth LFSR which is used to select a new encoded majority function during each clock cycle), with guaranteed asymptotic maximal length properties.

- Our NLFSR (with $n = 64$) has been tested against the NIST test suite, and successfully passes all tests.

- We have implemented our NLFSR in SPICE [10] using a 65nm PTM [2] process technology. The resulting implementation operates with a delay of 380ps (pipelined) or 680ps (unpipelined). The resulting speed is 2.63 GHz (pipelined) or 1.47 GHz (unpipelined). Since our NLFSR generates 64 bits per clock cycle, it can be used to generate cryptographic keys at the rate of 168.32 GHz (pipelined) or 94.08 GHz (unpipelined), easily sustaining the data-rates for secure OC-768 optic fiber communication, which operate at 40 Gbps.

The remainder of this paper is organized as follows. Section 2 describes our NLFSR design, while Section 3 reports experimental results, including results from the NIST tests as well as SPICE simulations. Finally, in Section 4, we make concluding comments and discuss further work that needs to be done in this area.

2. OUR APPROACH

As mentioned previously, LFSRs have some key weaknesses if they are used in the context of cryptographic key generation. In particular, the LFSR polynomial can be discovered in $2n$ cycles, where n is the order of the LFSR polynomial. At the same time, however, LFSRs have several desirable properties in the context of random number generation. In particular, a maximal-length LFSR with a characteristic polynomial of order n goes through all $2^n - 1$ states before repeating a previously generated state. Also, on average there are $n/2$ transitions between consecutive cycles. Further, a maximal length LFSR has a run of length k with probability $1/2^k$. Finally, LFSRs have simple, fast and efficient hardware implementations, using shift registers with feedback connections.

For all these reasons, the proposed NLFSR utilizes LFSRs, whose outputs are combined using non-linear functions. In particular, we utilize 3 LFSRs (L_1, L_2 and L_3) of varying lengths n_1, n_2 and n_3, as shown in Figure 2. The length of the NLFSR which is generated from these 3 LFSRs is $n \leq min(n_1, n_2, n_3)$. The i^{th} bits of each LFSR (L_1^i, L_2^i and L_3^i) are connected to a majority function (shown at the bottom of Figure 2), and the output of this majority function is the i^{th} bit of the resulting NLFSR. Note that the majority function is non-linear, hence the combined state of the n majority outputs therefore results in a NLFSR. Since the LFSR have lengths n_1, n_2 and n_3, and each LFSR polynomial is selected such that the LFSRs generate maximal length sequences, the length of the m-sequence of the composite LFSR is $2^{(n1+n2+n3)}$. The 3 LFSRs are shifted every clock cycle, thereby generating n new output bits in each clock cycle. In our experiments, n is chosen to be 64.

Note that in general, the number of LFSRs can be chosen to be more than 3 (not necessarily odd). Also, the size of the NLFSR can be chosen to be greater than 64.

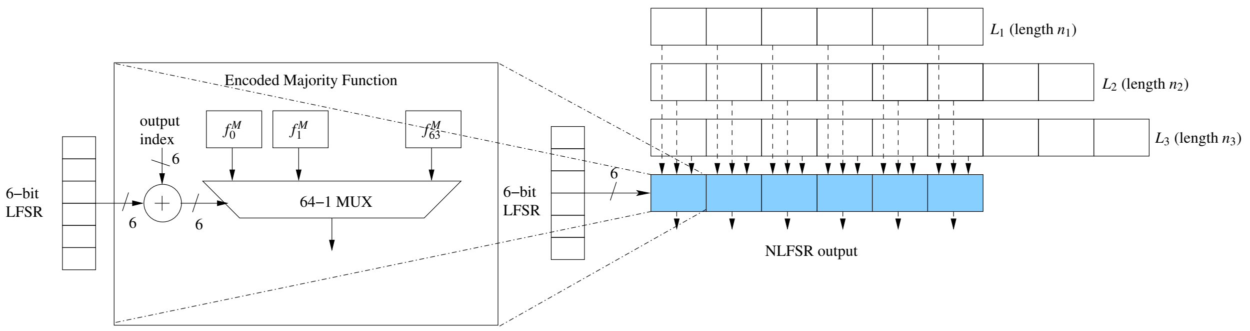

Figure 3: 64-bit NLFSR with Encoded Majority Function Blocks

We next assert that the NLFSR has an asymptotically maximal length sequence. To claim this proof, assume that in any clock cycle j, the state of LFSR L_i is V_i^j. The NLFSR operation is denoted as $N^j = M(V_1^j, V_2^j, V_3^j)$, where M is the bitwise majority function. We need to show that the probability that the NLFSR output does not repeat within the composite m-sequence of the 3 LFSRs can be made asymptotically low. In other words, we need to show that for two different clock cycles j and k in the composite m-sequence of the 3 LFSRs, $p(M(V_1^j, V_2^j, V_3^j) = M(V_1^k, V_2^k, V_3^k)) \rightarrow 0$ for an appropriately large n.

To show this, we note that for the majority function computing the output of the i^{th} bit of the NLFSR, the output will be the same across two clock cycles j and k if at least two of the sets $\{V_1^j, V_1^k\}$, $\{V_2^j, V_2^k\}$, and $\{V_3^j, V_3^k\}$ contain identical values. We denote the condition where $V_p^j = V_p^k$ $(p = 1,2,3)$ as "S", and $V_p^j \neq V_p^k$ is denoted as "D". Hence, the output of the majority function for the i^{th} bit can be equal across the j^{th} and the k^{th} clock cycles if the inputs to the i^{th} majority function are all same (SSS), or one input is different (SSD, SDS, DSS). Since the probability of the i^{th} bit changing in any of the 3 LFSRs across the j^{th} and the k^{th} clock cycles is $1/2$, the four input combinations (SSS, SSD, SDS, DSS) which cause for majority output to remain same across the j^{th} and the k^{th} clock cycles all have a probability of 1/8. Hence the probability of the majority output bit being the same across the j^{th} and the k^{th} clock cycles is $1/2$. Therefore the probability that entire NLFSR output (for all n bits) is same across the j^{th} and the k^{th} clock cycles is $1/2^n$. With large enough n, this probability can be made arbitrarily small. For our implementation, with $n = 64$, this probability is 4.21 $\times$ 10^{-20}.

With the architecture described above, initial testing of our crytographic keys against the NIST test suite failed the *BlockFrequency* test. Although the n output bits were different between adjacent clock cycles, contiguous groups of these n bits remained the same due to the shifting of the LFSR state. To avoid this, we modified our cryptographic key generation scheme, such that the majority function in the NLFSR is changed to an *encoded majority function* that is different for each bit position in any clock cycle. Further the encoded majority function for any bit position changes at each clock cycle. An encoded majority function in our approach is a function of 3 inputs, with 4 minterms mapped to the logic "1" value. If we use more than P LFSRs in our implementation instead of 3, the encoded majority functions would be functions of P inputs, with 2^{P-1} minterms mapped to the logic "1" value. As a result, P need not be an odd number.

Assuming a NLFSR output length $n = 64$ and 3 input LFSRS, we need 64 encoded majority functions ($f_0^M - f_{63}^M$). Each encoded majority function has 3 inputs (one from each LFSR). The number of unique encoded majority functions possible is ${}^{2^3}C_4 = 70$, of which 64 are chosen for our implementation at random.

In our implementation, the i^{th} bit of each LFSR is used as an input to the i^{th} output function block. Figure 3 illustrates (on the left) the structure of the left-most output function block. As shown in Figure 3, in each output function block, the inputs are connected to all 64 encoded majority functions and a 64-1 multiplexor selects one of these functions for the i^{th} bit NLFSR output. The selection signal is controlled by XOR-ing the bit index position with the 6-bit vector output of a 6-bit LFSR (shown on the left of Figure 3). The 6-bit LFSR is clocked every cycle and the output is shared by all 64 output function blocks (only one such block is shown on the left of Figure 3), so that all bit positions select a unique and different encoded majority function every cycle.

3. EXPERIMENTAL RESULTS

Several bitstreams of random numbers (with a sequence length of q =200,000) with different LFSR lengths (>64) and initial seeds, were generated and run through the NIST test suite. The NIST test suite consists of several statistical tests, each checking for a different type of randomness. The 12 major tests are listed in Table 1. In addition, there are variants of these tests (totalling 162), which are included in the test suite. The results of these tests are reported as a set of *p-values*. The pass/fail criteria is set by a significance level α. If the p-value of any test is less than α, then the cryptographic key sequence should not be accepted as random. NIST recommends that $\alpha = 0.01$ be used for all tests. Default test settings were used, with exception of block size, which was set to 20K ($0.1 \times q$ as recommended by [9]) and 64 bits (which is the size of our NLFSR). All 162 NIST statistical tests passed, and the results are reported in Table 1. For brevity, only a subset of the results are shown.

Statistical Test	P-VALUE
Frequency	0.739918
BlockFrequency	0.122325
CulmativeSums	0.350485
Runs	0.066882
LongestRun	0.350485
Rank	0.534146
FFT	0.213309
OverlappingTemplate	0.035174
ApproximateEntropy	0.534146
Serial	0.350485
LinearComplexity	0.739918

Table 1: NIST Analysis Results

The NLFSR output function block was implemented to determine its delay and gate count. For this paper, a 65nm process

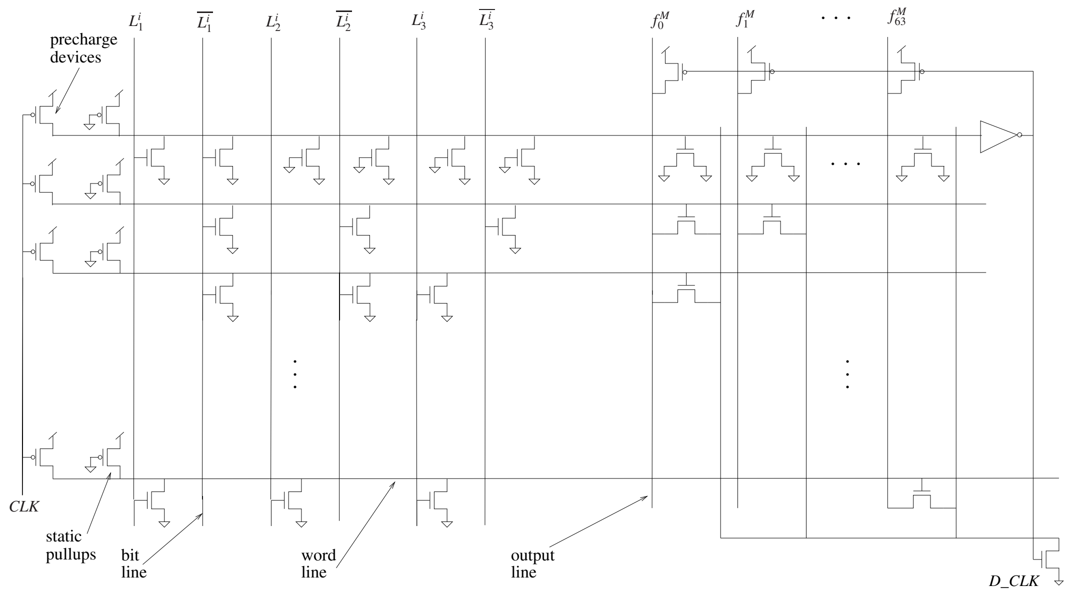

Figure 4: PLA Implementation for Output Function Block

PTM [2] technology, with copper interconnect and low-K dielectric is assumed. The circuit for the output function block consists of a dynamic NOR-NOR PLA (Figure 4) which produces all 64 encoded majority functions, and a 64-1 MUX, implemented using NMOS passgates. In the PLA, each complemented minterm is implemented in the ”AND” plane, and drives the horizontal word lines in the figure. For each input variable, there are two vertical bit-lines, one for each of its literals. The outputs are implemented in the ”OR” plane, on the right of the figure. A self-timing word line wire (the topmost wordline of Figure 4) is added to provide a delayed clock for the OR plane. This prevents the shorting of the outputs during the precharge phase of the clock. The multiplexor is a straightforward design using NMOS pass transistors. The total number of devices in the output function block is 562 (i.e. the total number of devices in the MUX and the PLA). For the PLA, the delay was measured from clock to the output using SPICE [10] simulations, and was determined to be 300ps. The multiplexor delay was measured to be 380ps. Hence the resulting implementation operates with a delay of 380ps (if it is pipelined) or 680ps (if it is left unpipelined). The resulting speed is 2.63 GHz (pipelined) or 1.47 GHz (unpipelined). Since our NLFSR generates 64 bits per clock cycle, it can be used to generate cryptographic keys at the rate of 168.32 GHz (pipelined) or 94.08 GHz (unpipelined), easily sustaining the data-rates for secure OC-768 optic fiber communication, which operate at 40 Gbps. The total estimated device count for our NLFSR is $\sim$ 40,000, which is significantly smaller than existing block ciphers [8].

4. CONCLUSIONS

We have present an implementation of a NLFSR for high throughput cryptographic key generation. Our NLFSR generates 64 key bits in each clock cycle, enabling its use in secure, high bandwidth communications. Our NLFSR is implemented using several LFSRs. The outputs of 64 bits from three LFSRs are combined using 64 encoded majority functions, and the majority function used for any bit position is changed at every clock cycle determined by a fouth LFSR. We demonstrate that our NLFSR can generate keys which passes the NIST suite, which is the industry benchmark to evaluate quality of ciphers and pseudo-random number generation. In addition, we detail and simulate a hardware implementation which can be used for OC-768 optical fiber communication, which operates at 40 Gbps. As the demand for data communication increases, our NLFSR is an effective and efficient approach towards fast key generation necessary for secure high data-rate communications.

5. REFERENCES

[1] NIST computer security resource center. http://csrc.nist.gov/groups/ST/toolkit/rng/index.html.

[2] PTM website. `http://www.eas.asu.edu/~ptm`.

[3] P. Bardell, W. McAnney, and J. Savir. *Built-In Test for VLSI - Pseudorandom Techniques*. John Wiley & Sons, Inc., 1987.

[4] W. E. Burr. Data encryption standard. http://nvl.nist.gov/pub/nistpubs/sp958-lide/250-253.pdf.

[5] A. Canteaut. Open problems related to algebraic attacks on stream ciphers. In *WCC*, pages 120–134, 2005.

[6] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[7] E. Dubrova, M. Teslenko, and H. Tenhunen. On analysis and synthesis of (n, k)-non-linear feedback shift registers. In *DATE '08: Proceedings of the conference on Design, automation and test in Europe*, pages 1286–1291, New York, NY, USA, 2008. ACM.

[8] T. Ichikawa, T. Kasuya, and M. Matsui. Hardware evaluation of the aes finalists. In *In The Third Advanced Encryption Standard Candidate Conference*, pages 279–285, 2000.

[9] C. Kenny. Random number generators: An evaluation and comparison of random.org and some commonly used generators, 2005.

[10] L. Nagel. Spice: A computer program to simulate computer circuits. In *University of California, Berkeley UCB/ERL Memo M520*, May 1995.

[11] A. Poschmann, G. Le, K. Schramm, and C. Paar. A family of light-weight block ciphers based on des. In *Proceedings of FSE 2007, LNCS*. Springer-Verlag, 2006.

[12] B. Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., 1996.

[13] J. Soto. Statistical testing of random number generators, nist, 2000.

[14] E. Zenner. Cryptanalysis of lfsr-based pseudorandom generators- a survey, 2004.