Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.1, Feb. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.1.51

분할 구조를 갖는 Leap-Ahead 선형 궤환 쉬프트 레지스터 의사 난수 발생기

박 영 규,† 김 상 춘, 이 제 훈‡
강원대학교

A Segmented Leap-Ahead LFSR Pseudo-Random Number Generator

Young-kyu Park,† Sang-Choon Kim, Je-Hoon Lee‡
Division of Electronics, Information and Communications Engineering,
kangwon National University, Rep. of Korea

요 약

스트림 암호 방식에서 사용되는 난수 발생기는 선형 궤환 쉬프트 레지스터(Linear feedback shift register, LFSR) 구조를 주로 사용한다. Leap-ahead LFSR 구조는 기존 다중 LFSR 구조와 같이 한 사이클에 다중 비트의 난수를 발생시킨다. 단지 하나의 LFSR로 구성되기 때문에 하드웨어적으로 간단하다는 장점을 갖지만, 때때로 생성되는 난수열의 최대 주기가 급격히 감소한다. 본 논문은 이러한 문제를 해결하기 위해 세그멘티드 Leap-ahead LFSR 구조를 제안한다. 수학적인 분석을 이용하여 제안된 구조를 검증하였다. 또한 제안된 구조를 Xilinx Vertex5 FPGA를 이용하여 회로 합성후 동작 속도와 회로 크기를 기존 구조와 비교하였다. 제안된 구조는 기존 Leap-ahead LFSR 구조에 비해 최대 2.5배까지 최대 주기를 향상시킨다.

ABSTRACT

A LFSR is commonly used for various stream cryptography applications to generate random numbers. A Leap-ahead LFSR was presented to generate a multi-bits random number per cycle. It only requires a single LFSR and it has an advantages in hardware complexity. However, it suffers from the significant reduction of maximum period of the generated random numbers. This paper presents the new segmented Leap-ahead LFSR to solve this problem. It consists of two segmented LFSRs. We prove the efficiency of the proposed segmented architecture using the precise mathematical analysis. We also demonstrate the proposed comparison results with other counterparts using Xinilx Vertex5 FPGA. The proposed architecture can increase 2.5 times of the maximum period of generated random numbers compared to the typical Leap-ahead architecture.

Keywords: LFSR, Galois, Pseudo-random number generator

I. 서 론

선형 궤환 쉬프트 레지스터는 데이터 스크램블, 에러 정정 코드, 데이터 암호화, 그리고 난수 발생기와 같은 다양한 어플리케이션에 사용되고 있다 [1-5]. 특히, 유비쿼터스 환경에서는 저전력 특징을 갖는 제한된 크기의 암호 회로가 필요하며, 대칭키 암호 방식에 비해 적은 면적과 저전력 특성을 갖는 스트림 암호 방식 그리고 LFSR을 이용한 난수 발생기가 주로 사용된다.

일반적으로, 난수 발생기는 특정한 크기의 난수를 발생시키기 위해 설계된 하드웨어 또는 프로그램을 의

접수일(2013년 12월 5일), 수정일(2014년 1월 10일), 게재확정일(2014년 1월 16일)
† 주저자, ing@kangwon.ac.kr
‡ 교신저자, jehoon.lee@kangwon.ac.kr (Corresponding author)

미한다. 이상적인 난수는 생성 방법이 결정되어 있지 않고, 다음에 생성될 난수 값이 전혀 예측할 수 없어야만 한다. LFSR 구조에서 생성되는 난수들은 일정한 절차에 의해 만들어지며 완전히 무작위적으로 생성되지 않기 때문에 의사 난수 발생기 (PRNG, pseudo-random number generator)라 부른다. LFSR 구조는 쉬프트 레지스터에 입력되는 값이 이전 상태값의 선형 함수로 계산되는 구조를 가지고 있다. 이 때, 레지스터가 가질 수 있는 값의 개수는 유한하기 때문에 생성되는 난수열은 특정한 주기로 반복된다. 효율적인 암호화를 위해서는 생성되는 의사난수열이 예측 불가능해야 함으로, 선형 함수를 잘 선택하여 생성되는 난수열의 주기가 길고 무작위적으로 보이는 수열을 생성해야 한다.

일반적인 LFSR 구조는 한 사이클 마다 한 비트의 난수를 생성한다. 최근 대부분의 어플리케이션은 다중 비트 난수 생성이 요구된다. 이를 위해 n개의 LFSR을 중첩시켜 구성한 다중 LFSR 구조로 *n*-비트 난수를 출력한다. 이는 하드웨어적으로 설계가 간단하나, 회로가 커진다는 단점을 갖는다. X. Gu는 이러한 문제를 해결하기 위해 Leap-ahead LFSR 구조를 제안하였다[6]. 하나의 LFSR을 이용하여 하나의 사이클에 다중 비트의 난수를 생성한다. 그러나, 이 구조는 생성난수의 비트 수와 LFSR의 레지스터 크기 관계에 따라 생성되는 난수열의 주기가 크게 감소하는 단점이 있다.

본 논문은 Leap-Ahead LFSR로부터 생성되는 난수열의 최대 주기가 감소하는 문제를 해결하는 방법을 제안한다. 이를 위해 LFSR의 레지스터 수와 생성할 난수의 비트 수에 따라 생성되는 난수열의 최대 주기가 감소되는 영향을 분석하고, 이를 토대로 생성된 난수열의 최대주기 감소를 줄일 수 있는 분할구조를 갖는 Leap-ahead LFSR 구조를 제안한다.

II. 관련연구

스트림 암호는 그림 1에 나타낸 것처럼 비교적 단순하게 구성된다. 송신단은 평문을 암호화하기 위해 키수열 발생기에 생성된 키수열과 평문을 연산하여 암호문을 생성하고, 수신단은 입력된 암호문을 평문으로 변환한다. 키수열 생성방법이 중요하며, 생성된 키의 예측이 불가능하도록 무작위로 생성된 것처럼 주기가 긴 키수열을 생성해야만 한다. 따라서, 난수발생기는 다음과 같은 세 가지 조건을 만족해야 한다[7]. 첫째, 출력되는 키 수열의 주기가 짧으면 쉽게 예측이 가능하므로 긴 주기를 가져야 한다. 둘째, 출력 난수열은 상호간에 상관관계 없이 랜덤해야 한다. 셋째, 키 수열은 큰 성형 복잡성을 가져야 한다. 생성된 난수열이 복잡하여 예측이 어려워야 한다. 이를 위해 LFSR을 이용한 난수발생기를 이용한다.

그림 2는 n차 LFSR 구조를 나타낸다. 이는 n개의 단 (stage)과 선형 궤환 함수 (feedback function), $f(s_0, s_1, s_2, \ldots, s_{n-1})$로 구성된다. n개의 단을 각각 $s_0, s_1, s_2, \ldots, s_{n-1}$을 하나의 상태로 정의하며 $s_0s_1s_2\ldots s_{n-1}$으로 나타낸다. 또한, 선형 궤환 함수는 궤환 황수 $f(s_0, s_1, s_2, \ldots, s_{n-1})$가 식 (1)과 같이 선형적인 형태로 표시된다.

$$f(s_0, s_1, s_2, \ldots s_{n-1}) = s_0 + c_1 s_1 + \ldots + c_{n-1} s_{n-1} \quad (1)$$

식 (1)에서 $c_1, c_2, \ldots, c_{n-1}$은 모두 0과 1의 값을 취하며, c_i의 값은 그림 2에서 s_i 단의 연결 상태를 나타내는데 이를 궤환 상수(feedback constant)라고 정의한다. $s_i(t)$를 시간 t에서의 s_i 단의 내용이라 할 때 $s_i(t+1) = s_{i+1}(t+1) = s_0 + \sum_{i=1}^{n-1} c_i s_i(t)$로 나타낸다. 즉, 시간 t에서 s_i단의 내용은 i=0,1,…,n-2인 경우에는 시간 t-1에서 앞 단의 내용이 오른쪽으로 시프트되고, S_{n-1}단의 내용은 시간 t-1의 모든 단의 내용이 선형 궤환 함수의 변수로 사용되어 얻은 함수가 된다. 선형 시프트 레지스터에서 발생되는 선형 수열은 식 (2)로 나타낸다.

$$s_0, s_1, \ldots, s_{n-1} = s_0 + \sum_{i=1}^{n-1} c_i s_i, \quad s_{n+1} = s_1 + \sum_{i=1}^{n-1} c_i s_{i+1}, \ldots \quad (2)$$

이진 수열 (S_t)에서 $s_0, s_1, \ldots, s_{n-1}$을 제외한 각 항

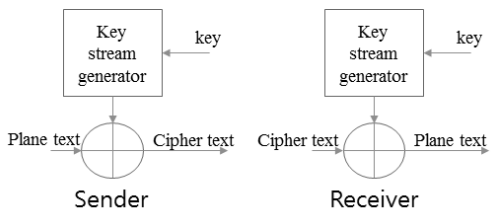


Fig.1. General stream cipher using a RNG.

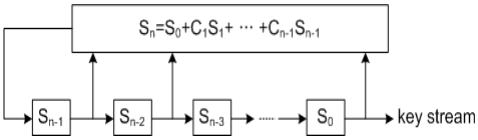


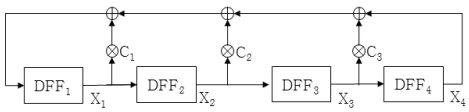
Fig.2. General architecture of LFSR RNG.

은 직전의 n개의 항과 궤환 상수로 결정되며, 생성된 난수열은 2^n-1보다 같거나 작은 주기를 갖는다.

2.1 기존 LFSR URNG

스트림 암호를 위한 LFSR은 일반적으로 갈로이스 (Galois)와 피보나치 (Fibonacci) 형태의 LFSR 구조를 사용한다. 그림 3은 일반적인 피보나치 타입의 LFSR 구조의 예를 나타낸다. 4개의 플립플롭이 궤환 입력을 갖는 LFSR로 구성되며, X_m는 각 단의 출력이 된다. LFSR의 초기 저장값은 시드 (seed)라 부른다. 본 논문에서는 LFSR의 단수를 n 그리고 생성되는 난수의 비트수를 m으로 표기한다.

그림 3에 나타낸 것처럼 LFSR에 시드값인 "1010"을 입력하면, 클럭 변화에 따라 난수가 발생된다. 1번째 클럭에서 "0101"을 출력하고 14번째 클럭에서 "0100"을 출력한 후 15번째 클럭부터 이 순차적인 난수열을 반복적으로 출력한다. 난수열의 최대 주기는 15가 되고, 이론적으로 LFSR로부터 생성된 난수열의 최대 주기 2^n-1이 된다. 그러나, 시드값에 따라 생성되는 난수열의 주기가 변화한다. 시드값을 "1010" 대신 ""1011"을 입력하면, 난수열의 최대 주기는 7로 감소된다. 따라서, 최대 주기값을 유지하기 위해서는 신중하게 시드값을 결정해야 한다. 또한, 출력된 난수



	1	2	3	4
Seed value	1	0	1	0
1 번째	0	1	0	1
2 번째	0	0	1	0
3 번째	1	0	0	1
4 번째	0	1	0	0
5 번째	1	0	1	0
6 번째	0	1	0	1
7 번째	0	0	1	0
8 번째	1	0	0	1
9 번째	0	1	0	0

	1	2	3	4
10 번째	1	0	1	0
11 번째	0	1	0	1
12 번째	0	0	1	0
13 번째	1	0	0	1
14 번째	0	1	0	0
15 번째	1	0	1	0
16 번째	0	1	0	1
17 번째	0	0	1	0
18 번째	1	0	0	1
19 번째	0	1	0	0

Fig.3. Fibonacci type LFSR RNG.

열의 예측을 막기위해 난수열간의 상관관계를 줄여야 한다. 그림 3의 예제는 15개의 난수열을 반복적으로 출력한다. LFSR의 구조가 간단하기 때문에 출력된 난수열을 통해 다음 생성할 난수의 예측이 가능하다. 난수열간의 상관관계를 줄이기 위해 Leap-Ahead LFSR 구조가 사용된다[6].

2.2 Leap-Ahead LFSR URNG

Leap - ahead 구조를 갖는 LFSR은 하나의 LFSR을 통해 다중 비트의 난수를 출력할 뿐만 아니라, 출력되는 난수열간의 상관관계를 줄인다. LFSR 구조를 갖는 난수 발생기의 인접한 두 출력은 식 (3)에 나타낸 관계를 갖는다.

$$X(t+1)=AX(t) \tag{3}$$

여기에서, $X(t)$는 현재 사이클, t에서의 출력이고, $X(t+1)$은 다음 사이클에서의 LFSR 출력을 의미한다. A는 변환 행렬이 된다. 식 (3)은 시간 t를 증가시킴에 따라, 식 (4)와 같이 변환된다.

$$\begin{aligned} X(t+m) &= AX(t+m-1)=A(AX(t+m-2)) \\ &= A^mX(t) \end{aligned} \tag{4}$$

따라서, $t+m$ 시점의 LFSR 출력은 t 시점의 출력에 A^m을 곱한 결과와 같다. 이 때, 변환 행렬 A는 식 (5)와 같이 표현된다.

$$A_{Galois}=\begin{pmatrix} 0_{1\times(n-1)} & C_{n\times1} \\ I_{(n-1)\times(n-1)} & \end{pmatrix}_{n\times n} \tag{5}$$

식 (5)에서 0은 제로 행렬을 의미하며, I는 항등행렬을 나타낸다. 이 결과를 이용하여 식 (6)과 같이 A^m을 구할 수 있다.

$$\begin{pmatrix} 0_{m\times(n-m)} & C_{n\times1}\ A\times C_{n\times1}\ \cdots\ A^{m-1}\times C_{n\times1} \\ I_{(n-m)\times(n-m)} & \end{pmatrix}_{n\times n} \tag{6}$$

식 (6)의 변환행렬을 회로로 구현하여, 매 클럭마다 생성된 난수열의 m번째 출력을 생성하여 난수들간의 상관관계를 줄인다. 그림 4는 갈로이스 타입의 LFSR 구조에서 4비트 난수 발생 결과를 나타낸다.

일례로, LFSR 구조에서 생성된 난수열은 {1011, 1100, 0110, 0011, 1101, 1010, 0101, 1110, 0111, 1111, 1011, 1001, 1000, 0100, 0010}이다. 생성된 난수열의 최대 주기는 15가 되고, 생성된 난수열은 한 클럭씩 인가될 때마다, 1011, 1100, … 순으로 인접한 난수를 출력한다.

그러나, Leap-Ahead LFSR 구조의 경우 생성된 난수열로부터 다음 클럭에 m-사이클 이후의 난수를 출력하여 상관관계를 줄일 수 있다. 즉, m을 2로 설정할 경우, 1011, 0110, 1000 순으로 출력되며, 최대 주기는 15를 유지한다. 따라서, 인접한 난수를 출력하지 않기 때문에 출력되는 난수열은 상호간에 상관관계를 줄일 수 있다. 그러나, m을 3으로 변경할 경우, 1011, 0011, 0010, 1101, 0111 순서로 출력된다. 따라서, 상관관계는 감소되나, 최대 주기가 15대신 5로 줄어들게 된다. 결론적으로 Leap-Ahead LFSR 구조의 경우 생성된 난수열의 상관관계와 최대 주기간의 변화를 고려하여 m을 설정하여야만 한다. 특히, 출력되는 난수들의 상관관계가 가장 적으면서 최대 주기 2^n-1을 유지하는 조합이 가장 이상적이다.

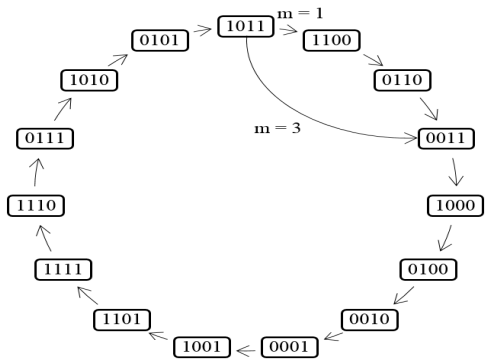


Fig.4. The generated random number sequences in a general Leap-ahead LFSR when m is 1 and 3.

표 1은 기존 Leap-ahead LFSR 구조에서 생성되는 난수열의 최대 주기를 나타낸다. 표 1에서 나타낸 것처럼 2^n-1과 m이 서로 소인 경우에 생성되는 난수열의 최대 주기가 2^n-1로 유지되고, 그렇지 않은 경우 생성되는 난수열의 최대 주기가 큰 폭으로 감소된다. 일례로 n이 16 인 경우, 출력되는 난수의 비트 수가 3 그리고 5인 경우 2^{16}-1인 65,535와 나누어지기 때문에 표 1에 나타낸 것처럼 최대 주기가 각각 21,845와 13,107로 감소된다.

따라서, 기존 Leap-ahead LFSR의 출력 난수열의 비트 수와 LFSR의 레지스터 수를 고려하여 늘 최대주기를 유지하며 상관관계를 줄여야 한다는 단점을 갖는다. 본 논문에서는 출력되는 난수들간의 상관관계는 기존 Leap-ahead LFSR 구조와 동등하나, 최대 주기 감소폭을 줄일 수 있는 새로운 Leap-ahead LFSR 구조를 제안한다.

Table 1. The maximum period of the random numbers that are generated in general Leap-ahead LFSR.

m \ n	16	32
1	65535	4294967295
2	65535	4294967295
3	21845	1431655765
4	65535	4294967295
5	13107	858993459
6	65535	4294967295
7	65535	4294967295
8	65535	4294967295

III. 제안한 Leap-ahead LFSR 난수발생기

Leap-ahead LFSR 구조는 생성하려는 난수의 비트 수와 LFSR의 스테이지 수에 따라 최대 주기가 큰 폭으로 감소하는 단점을 갖는다. 최대 주기가 감소하는 현상을 방지하기 위하여 본 논문에서는 하나의 큰 LFSR을 이용하는 대신 분할된 LFSR을 연결하는 세그먼티드 Leap-ahead 구조를 제안한다.

기존 Leap-ahead LFSR의 생성 난수열의 최대 주기는 해당 LFSR의 스테이지 수를 n, 생성된 난수의 비트 수를 m이라 할 때, 식 (7)과 같다[6].

$$T_{\max} = \frac{[2^n - 1, m]}{m} \tag{7}$$

여기서, $[2^n$-1, $m]$은 2^n-1과 m의 최소공배수를 의미한다. 따라서, 2^n-1과 m이 서로 소인 경우에만 생성되는 난수열의 최대 주기를 2^n-1이 된다. 그러나, 난수발생기 설계 특성상 LFSR의 단 수와 출력되는 난수의 비트 수를 위와 같은 조건으로 구성하기 어려워져 생성되는 난수열의 최대 주기가 짧아지게 된다. 이러한 문제를 해결하기 위해 하나의 LFSR 대신 여러 개의 작은 LFSR을 시리즈로 연결하는 분할된 Leap-ahead LFSR 구조를 제안한다.

제안된 회로는 그림 5와 같이 하나의 LFSR을 두 세그먼트로 분할했다. 제안된 구조는 2^n-1이 m으로

나누어지는 경우와 그렇지 않은 경우에 따라 두 개의 MUX를 제어한다. 그림 5의 하단의 클럭 입력에 연결된 MUX, *SEL*은 클럭 입력을 선택한다. 2^n*-1*과 *m*이 서로 소인 경우인 경우 동일한 클럭 입력을 두 세그먼트로 인가하고, 그렇지 않은 경우 분리된 클럭을 인가하여 독립적으로 동작시킨다. 상단의 MUX, DMUX는 Segment2의 마지막 출력을 2^n*-1*과 *m*이 서로 소인 경우 Segment1의 입력으로 인가하여 두 세그먼트를 하나의 큰 LFSR로 통합하고, 서로 소가 아닌 경우, Segment2의 입력으로 인가하여 두 세그먼트들을 독립적으로 분할한다. 두 세그먼트의 출력은 하나의 다중 비트 난수로 합쳐져 출력된다.

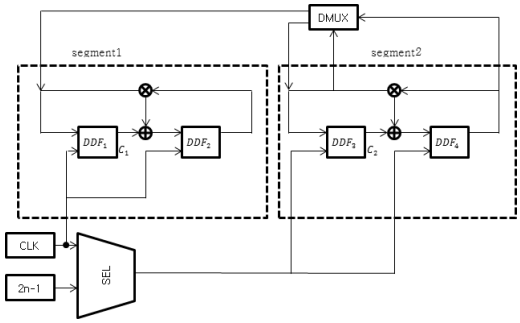


Fig.5. The architecture of the proposed segmented Leap-ahead LFSR.

따라서 제안된 분할 구조를 갖는 Leap-ahead 구조에서 생성된 난수열은 크게 2^n*-1*과 *m*이 서로 소인 경우 하나의 큰 LFSR로 동작하는 경우와, 그렇지 않고 두 개의 작은 LFSR들로 구동될 때 각각 주기 계산식이 변화한다. 첫 번째로, 하나의 큰 Leap-ahead LFSR 구조로 동작하는 경우 생성 난수열의 최대 주기, T_{max}는 식 (6)과 같이 구할 수 있다.

그러나, Segment1과 Segment2의 두 개의 세그먼트로 분할된 Leap-ahead LFSR의 주기는 Segment1의 주기를 T1 그리고 Segment2의 주기를 T2로 정의하면, 이 두 주기의 곱으로 결정된다. 이 때, 각 세그먼트는 Leap-ahead LFSR로 동작하며, 각각의 단 수를 n_1 그리고 n_2라 하면 n_1과 n_2의 합은 전체 Leap-ahead LFSR의 단수인 *n*이 된다. 마찬가지 방법으로, 각 세그먼트의 난수 출력의 비트 수를 m_1 그리고 m_2라 하면 m_1과 m_2의 합은 전체 Leap-ahead LFSR의 출력 난수 비트 수인 *m*이 된다.

분할 구조를 갖는 LFSR은 두 세그먼트의 생성 난수열의 최대 주기의 곱으로 전체 생성 난수열의 최대 주기를 계산한다. 따라서, 두 개의 세그먼트로 분할하였을 때, 생성되는 난수열의 최대 주기는 다음 세 가지 경우가 발생한다. 첫 번째 경우는 두 세그먼트들이 모두 Leap-ahead LFSR의 최대 주기를 만족하는 경우이며 이는 2^{n1}*-1*과 m_1 그리고 2^{n2}*-1*과 m_2가 모두 서로 소일 때 발생한다. 이 경우 생성 난수열의 최대 주기는 식 8과 같이 구할 수 있다. 두 번째 경우는 두 세그먼트중 하나는 생성 난수열의 최대주기를 갖고, 다른 하나는 그렇지 못할 경우에 발생되며, 식 (9)와 같이 연산된다. 마지막으로 두 세그먼트 모두 2^{n1}*-1*과 m_1 그리고 2^{n2}*-1*과 m_2가 모두 서로 소가 아닐 때 발생되며, 식 (10)과 같이 계산된다.

$$T_{\max} = (2^{n_1}-1)\times(2^{n_2}-1) \tag{8}$$

$$T_{\max} = \frac{(2^{n_1}-1)\times(2^{n_2}-1)}{m_1} \text{ or } \frac{((2^{n_1}-1)\times(2^{n_2}-1)}{m_2} \tag{9}$$

$$T_{\max} = \frac{(2^{n_1}-1)\times(2^{n_2}-1)}{m_1 \times m_2} \tag{10}$$

표 2는 제안된 세그먼티드 Leap-Ahead LFSR 구조를 사용할 때 생성되는 난수열의 최대 주기를 나타낸다. 전체 Leap-ahead LFSR을 두 개의 Leap-ahead LFSR로 분할하였다.

첫 번째 예로, 전체 LFSR의 단 수가 16인 경우 각각 8개의 단으로 구성된 두 개의 LFSR로 나누어 구성하였다. 표 1에서 나타낸 것처럼 출력되는 난수의 비트 수가 3과 5일 때 생성되는 난수열의 최대 주기가 감소된다. 즉, 최대주기인 2^n*-1*인 65,535를 최대공약수인 3과 5로 각각 나눈 값만큼 감소되어, 각각의 최

Table 2. The maximum period for the proposed architecture.

m \ n (j+j)	16 (8+8)	32 (16+16)
1	65535	4294967295
2	65535	4294967295
3	65025	4294836225
4	65535	4294967295
5	21675	1431612075
6	65535	4294967295
7	65535	4294967295
8	65535	4294967295

대 주기가 21,845 그리고 13,107로 감소한다. 반면에, 제안된 구조는 m이 3과 5일 때 각각 최대 주기의 크기가 65,025와 21,675가 된다. 따라서, m이 3인 경우 최대 주기는 65,535로 유지되고 m이 3인 경우에도 기존 Leap-ahead 구조에 비해 최대 주기의 감소폭을 크게 줄일 수 있다.

IV. 시뮬레이션 결과

본 논문에서 제안된 구조를 VHDL로 합성한 후, Xilinx사의 FPGA를 이용하여 동작을 검증하였다. 실험을 위해 설계된 난수 발생기는 매 사이클마다 32, 64, 그리고 128 비트의 난수열을 하나씩 출력한다. 성능 비교를 위해 기존 Leap-ahead LFSR과 제안된 구조를 동일한 설계과정으로 합성 후 성능을 비교하였다. 제안된 분할 구조를 갖는 Leap-ahead 구조는 기존 Leap-ahead 구조에서 사용되는 LFSR의 절반 크기를 갖는 두 LFSR를 연결하여 구성하였다. 32-비트 난수발생기는 16단의 LFSR을 두 개 연결하여 구성하였고, 마찬가지 방식으로 64 및 128비트 난수발생기는 32단 및 64단의 LFSR을 각각 연결하여 회로를 구성하였다.

첫 번째로, 회로의 하드웨어적인 복잡도와 동작 속도를 비교하였다. 표 3은 기존 Leap-ahead 구조와 제안된 구조를 비교 결과를 나타낸다. 표 3에 나타낸 것처럼, 회로 크기는 제안된 세그멘티드 Leap-ahead 구조가 기존 구조에 비해 약 2배 증가함을 알 수 있다. 그러나 동작 속도면에서, 제안된 구조는 32비트에서 5.48ns, 64비트에서 5.75ns 그리고 128비트에서 6.51ns의 최대 임계 지연 시간을 갖는다. 이 결과는 기존 Leap-ahead 구조와 거의 유사한 결과를 나타낸다. 따라서, 여러 개의 LFSR을 연결하는 구조는 하드웨어적인 복잡도는 증가하나, 회로 성능은 기존 Leap-ahead 구조와 거의 유사함을 나타낸다.

Table 3. Comparison results in hardware complexity and length of critical path between the proposed architecture and conventional one.

Conventinal LFSR URNG			Proposed LFSR URNG		
bit	ALUT	Critical path	bit	ALUT	Critical path
32	4	5.503ns	32	10	5.481ns
64	5	5.868ns	64	8	5.753ns
128	5	6.634ns	128	10	6.518ns

그림 6과 기존 16-bit Leap-ahead LFSR 구조에서 발생되는 최대 주기 감소를 극복하기 위해 하나의 16-bit LFSR 대신 두 개의 8-bit LFSR들을 연결하여 구성한 난수발생기에서 생성한 난수열의 최대 주기를 비교한 결과를 나타낸다. 실험은 m의 값을 1에서 15까지 증가시켜 최대 주기 변화율을 구하였다. 기존 Leap-Ahead LFSR 구조는 2^{16}-1과 나누어지는 m값을 갖는 3, 5, 6, 9, 그리고 15인 부분에서 최대 주기가 $(2^{16}-1)/3$, $(2^{16}-1)/5$, $(2^{16}-1)/6$, $(2^{16}-1)/9$, $(2^{16}-1)/15$만큼 감소되는 결과를 나타낸다. 두 개의 8-bit LFSR들로 나누어 구성한 경우, 8-비트 LFSR의 최대 주기인 2^8-1과 나누어지지 않는 m값을 갖는 6과 9인 경우 생성되는 난수열의 최대 주기가 이론적으로 가장 긴 2^n-1에 근접할 정도로 회복됨을 알 수 있다. 또한 m이 3, 5인 경우에는 2개로 분할할 경우 최대 주기가 오히려 감소한다. 따라서, LFSR의 크기와 이의 출력 비트 수를 주의깊게 조정하여 설계해야 한다.

그림 7은 제안된 16-bit 난수발생기를 구성하기 위해 두 개의 LFSR의 단수를 8-bit로 하는 대신 5비트와 11비트로 나누어 구성하였다. 이는 그림 6에 나타낸 것처럼 동일한 크기인 8비트로 분할하였을 때, m이 6과 9일 때 최대주기에 근접하게 회복되나, 오히려 m이 3과 5일 때 급격히 감소되는 현상을 나타낸다. 서로 다른 크기로 나누었을 때, 전체적인 생성 난수열의 최대 주기는 동일한 크기를 갖는 두 개의 세그먼티드 LFSR 구조에 비해 5% 가량 감소한다. 그러나, m값이 3과 5일 때 나타났던 최대 주기의 급격한 감소는 없어지고, 전체적으로 m값의 변화에 따른 최대 주기의 분산이 크게 감소한다는 장점을 갖는다. 제

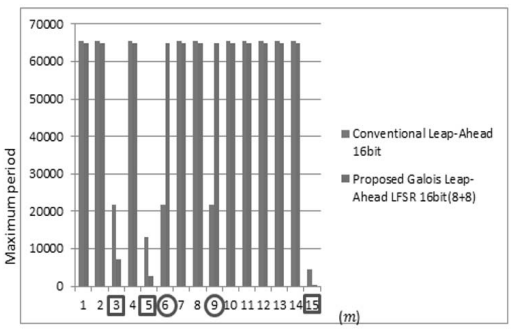


Fig.6. The maximum period of the generated random numbers when m is changed from 1 to 16. (8-bit LFSR and 8-bit LFSR)

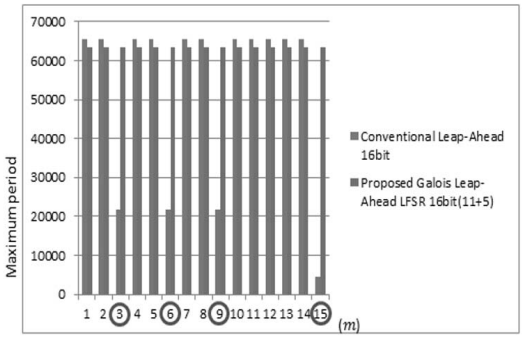


Fig.7. The maximum period of the generated random numbers when m is changed from 1 to 16. (5-bit LFSR and 11-bit LFSR)

안된 난수발생기 구조를 적용하기 위해서는 LFSR의 비트 수 n과 출력 난수열의 비트 수 m을 주의깊게 결정해야하며, 분할되는 LFSR들의 크기도 중요하다.

마지막으로, 분할된 LFSR의 크기에 따른 최대 주기 변화율을 실험하였다. 64-bit 난수발생기를 설계하고, 생성 난수열의 최대주기 변화를 실험하였다. 두 개로 분할된 LFSR은 같은 32-bit 크기인 경우와 서로 다른 크기를 갖는 경우의 출력을 비교하였다. 특히, 서로 다른 크기의 LFSR은 모두 소수로만 연결 가능한 조합을 구한 후 그 차이가 가장 적은 조합으로 구성하였다. 즉, 64-bit LFSR을 두 개의 LFSR로 분할할 경우, 소수로만 가능한 조합은 (41, 23), (47+17), (53+11) 이렇게 3가지 경우가 발생하고 두 LFSR의 크기 차이가 가장 적은 (41, 23)을 기준하여 설계하였다. 실험 결과는 16-bit LFSR을 이용하여 설계한 예와 유사한 결과를 나타낸다. 즉, 두 개의 32-bit LFSR의 크기가 같은 경우, 2^{32}-1이 나누어지는 m값 (3, 5, ...)의 경우 $(2^{32}-1)(2^{32}-1)/m^2$으로 최대 주기가 크게 감소하고, 그 외의 경우 최대 주기가 기존 64-비트 Leap-ahead LFSR의 최대 주기인 2^{64}-1 이거나 거의 유사한 값을 갖는다. 표 4에 나타낸 것처럼 m의 변화에 따라 최대 주기의 평균 감소율은 평균 11%이나, 분산오차의 크기가 14%이다. 두 LFSR의 크기를 소수인 41과 23으로 설계하면, m의 변화에 따라 최대 주기는 평균 16% 감소하나 분산오차의 크기가 균등분할에 비해 적은 11%가 된다. 이 결과는 128-bit와 256-bit를 출력하는 난수발생기에서도 동일한 현상이 발생한다. 따라서, 제안된 난수 발생기는 기존 Leap-ahead LFSR에 비해 생성되는 난수열의 최대주기의 감소율을 크게 줄일 수 있고, 동일한 크기로 분할하였을 때, 각 LFSR의 단 수, n의 2^n-1과 출력 난수열의 비트수 m이 서로 소인 경우 동일한 크기로 구성할 경우 생성되는 최대 주기의 손실이 가장 적고, 만일 서로 소가 아닌 경우 두 LFSR을 소수 중 가장 차이가 적은 크기로 구성할 경우 평균적으로 생성 난수열의 최대 주기가 가장 적게 감소함을 확인하였다.

Table 4. The average reduction ratio of the maximum period of the generated random numbers and its variance error rate.

		평균주기 감소율	분산오차
64비트	균등분할	11%	14%
	소수분할	16%	11%
128비트	균등분할	11%	14%
	소수분할	16%	11%
256비트	균등분할	11%	14%
	소수분할	16%	11%

V. 결 론

스트림 암호에서는 회로 크기가 적고 동작 속도가 빠른 LFSR 난수발생기가 주로 사용된다. 하나의 LFSR로 다중 비트 난수를 생성하는 Leap-ahead LFSR 구조는 회로 크기면에서 유리하나, 생성되는 난수열의 최대 주기가 급격히 감소하는 단점을 갖는다. 본 논문은 이를 위해 하나의 큰 LFSR을 사용하는 대신 여러 개로 분할된 Leap-ahead LFSR 구조를 제안하였다. 또한, 분할된 LFSR의 크기에 따라 생성 난수열의 최대 주기 변화를 분석하였다. 이를 통해 두 개로 분할된 LFSR의 경우 LFSR의 단 수, n의 2^n-1과 출력 난수열의 비트수 m이 서로 소인 경우 동일한 크기로 구성하면 이론적인 최대 주기 2^n-1에 가깝게 난수열을 생성할 수 있고, 만일 그렇지 않은 경우 두 LFSR을 차이가 가장 적은 두 개의 소수로 크기를 결정하면 평균적으로 11%의 손실만이 발생함을 확인하였다.

References

[1] J. C. Lin, S. J. Chen, and y. H. Hu, "Cycle-efficient LFSR implementation on word-based micro-architecture," *IEEE Trans. on Computers*, 62(4), pp. 832-838,

Apr. 2013.
〔2〕 J. Glossner et al., "A software-defined communications baseband design," *Proc. IEEE Comm. Magazine*, 41(1), pp. 120-128, 2003.
〔3〕 A. K. Panda, P. Rajput and B. Shukla, "FPGA implementation of 8, 16 and 32 bit LFSR with maximum length feedback polynomial using VHDL," *Proc. of CSNT 2012*, pp. 769-771, 2012
〔4〕 N. M. Thamrin, G. Witjaksono, A. Nuruddin and M. S. Abdullah, "An enhanced hardware-based hybrid random number generator for cryptosystem," *Proc. of ICIME2009*, pp. 152-156, 2009
〔5〕 P. L'Ecuyer, "Random numbers for simulation," *Communications of the ACM*, 33(10), pp. 85-97, 1990
〔6〕 X. Gu and M. Zhang, "Uniform random unber generator using Leap-ahead LFSR architecture," *Proc. of ICCCS 2009*, pp. 150-154, 2009.
〔7〕 J. H. Lee, M. J. Jeon, and S. C. Kim, "Uniform random number generator using Leap-ahead LFSR architecture," Proc. of ASEA and DRBC, pp. 28-2, 2012.

〈저 자 소 개〉

박 영 규 (Young-kyu Park) 학생회원
2012년 2월: 강원대학교 공학대학 정보통신공학과 학사
2012년 3월~현재: 강원대학교 공학대학 정보통신공학과 석사 재학 중

김 상 춘 (Sang-choon Kim) 종신회원
1986년: 한밭대학교 전자계산학과 학사
1989년: 청주대학교 전자계산학과 석사
1999년: 충북대학교 전자계산학과 박사
1983년~2001년: 한국전자통신연구원 정보보호연구단
2001년~현재: 강원대학교 정보통신공학과 주교수

이 제 훈 (Je-Hoon Lee) 정회원
1998년 8월: 충북대학교 정보통신공학과 학사
2001년 2월: 충북대학교 정보통신공학과 통신회로 및 시스템공학 석사
2005년 2월: 충북대학교 정보통신공학과 통신회로 및 시스템공학 박사
2005년 4월~2006년 4월: Univ. of Southern California Viterbi School 박사후연구원
2006년 8월~2009년 8월: 충북대학교 BK21 충북정보기술사업단 초빙조교수
2009년 8월~현재: 강원대학교 삼척캠퍼스 전자정보통신공 학부 조교수