情報保護學會論文誌
第14券 第4號, 2004. 8

RC4 스트림 암호 알고리즘을 위한 고속 연산 구조의 FPGA 구현 및 성능 분석

최 병 윤[a)†‡], 이 종 형[a)], 조 현 숙[b)]
동의대학교[a)], 한국전자통신연구원[b)]

FPGA Implementation and Performance Analysis of High Speed Architecture for RC4 Stream Cipher Algorithm

Byeong-yoon Choi[a)†‡], Jong-hyung Lee[a)], Hyun-sook Cho[b)]
Dong-Eui University[a)], ETRI[b)]

요 약

본 논문에서는 RC4 스트림 암호 알고리즘을 구현하는 고속 연산 구조를 제안하고, FPGA 구현 결과를 제시하였다. 기존 방식이 긴 초기화 동작이 필요하거나, S-배열 초기화 대기 시간을 제거하기 위해 S-배열을 2개 혹은 3개를 사용하는 구조를 갖는데 비해, 제안한 RC4 스트림 암호 연산 구조는 256-비트 valid-비트 엔트리 방식을 사용하여, S-배열 초기화 동작을 제거하였다. 그리고 RC4 알고리즘을 다양한 응용 분야에 사용될 수 있도록 효율적인 모듈라 연산 하드웨어를 사용하여 40 비트와 128 비트 키를 지원하도록 하였다. 제안한 RC4 스트림 암호 연산 구조를 Xilinx XCV1000E-6H240C FPGA로 구현하였다. 설계된 RC4 프로세서는 40 MHz에서 106 Mbps의 암호 비트 생성율의 성능을 갖고 있으며 WEP 프로세서와 RC4 키 검색 엔진에 적용가능하다.

ABSTRACT

In this paper a high speed architecture of the RC4 stream cipher is proposed and its FPGA implementation is presented. Compared to the conventional RC4 designs which have long initialization operation or use double or triple S-arrays to reduce latency delay due to S-array initialization phase, the proposed architecture for RC4 stream cipher eliminates the S-array initialization operation using 256-bit valid entry scheme and supports 40/128-bit key lengths with efficient modular arithmetic hardware. The proposed RC4 stream cipher is implemented using Xilinx XCV1000E-6H240C FPGA device. The designed RC4 stream cipher has about a throughput of 106 Mbits/sec at 40 MHz clock and thus can be applicable to WEP processor and RC4 key search processor.

Keywords : *RC4, Stream Cipher, WEP, Pseudo Random Number Generator*

Ⅰ. 서 론

가까운 장래에 실현될 유비쿼터스(Ubiquitous) 컴퓨팅 환경에서는 모든 사물에 컴퓨터가 내장되고 이러한 사물 간에 통신이 이루어지게 되는데, 이러한 통신은 대부분 무선 통신에 기반을 두고 이루어진다[1]. 이러한 무선 통신을 사용자가 신뢰하기 위해서는 철저한 보안이 필요하다. 반면 유비쿼터스 환경에서

접수일: 2004년 4월 29일 ; 채택일: 2004년 8월 2일
* 본 연구는 한국 전자 통신 연구원 위탁과제와 2003년도 BB21 사업 지원으로 이루어졌으며, 회로 설계에 IDEC 소프트웨어가 사용되었습니다.
† 주저자, ‡ 교신저자 : bychoi@deu.ac.kr

사물에 내장되는 컴퓨터의 경우 사용 환경 특성으로 저전력 특성을 갖는 제한된 크기의 하드웨어를 필요로 한다. 따라서 유비쿼터스 환경에서는 DES(Data Encryption Standard)와 AES(Advanced Encryption Standard)[2]와 같은 대칭키 블록 암호 방식에 비해, 적은 면적 과 저전력 특성을 갖는 스트림 암호 방식이 주목을 받고 있다. 이러한 스트림 암호 방식 중에 널리 사용되고 있는 암호 방식이 RC4 스트림 암호이다[3]. RC4 스트림 암호 알고리즘은 1987년 Ron Rivest에 의해 개발된 가변 길이의 키를 갖는 스트림 암호 방식으로서, DES와 같은 블록 암호에서 사용되는 비트의 뒤섞임(permutation) 동작이 없는 대신에, 메모리 내 데이터 위치 교환(swapping), 2의 지수승의 모듈라 덧셈(modular addition) 등의 연산으로 구성되어 있다. RC4 알고리즘은 키를 생성하는 동작 이전에 S-배열의 초기화, 키 값을 사용한 S-배열의 랜덤화(randomization) 동작이 필요하며, 이에 따른 큰 초기 대기 시간(latency)이 RC4 알고리즘의 하드웨어 응용에 많은 문제를 야기하고 있다. 현재 RC4 알고리즘의 키에 대한 몇 가지 취약점이 지적되고 있지만[4], RC4 알고리즘은 IEEE 802.11i 무선랜 보안 프로토콜 WEP(wired equivalent privacy), 로터스 사의 Notes, 오라클의 보안 SQL, PDF 문서를 만드는 acrobat 프로그램 등에서 널리 사용되고 있다. 현재 휴대용 멀티미디어 단말기와 무선 랜의 고속화와 보안 필요성이 부각됨에 따라, RC4 암호 알고리즘의 하드웨어 구현과 안전도 개선을 위한 키 길이의 확대 요구가 증대되고 있다[5]. RC4의 하드웨어 구현 연구결과는 크게 단일 입출력 포트를 갖는 S-배열 메모리를 이용한 구조[6-7]와 다중 입출력 포트를 가진 메모리 구조[8-9]로 나뉜다.

본 연구에서는 RC4 알고리즘에 대한 기존 하드웨어 구현 방식의 S-배열 초기화 동작에 따른 대기 시간 문제를 제거하고, 다양한 RC4 응용 분야에 적합하도록 40 비트와 128 비트 키 길이를 지원하는 RC4 스트림 암호 하드웨어 구조를 제안하고, FPGA로 구현한 후 성능을 분석하였다. 본 논문의 구성은 II 장에서는 RC4 스트림 암호 알고리즘을 소개하고, III 장에서는 V-비트 엔트리 방식에 의해 S-배열 초기화 동작에 기인한 초기 대기 시간을 제거한 RC4 스트림 암호 회로를 기술하였다. IV 장에서는 설계된 프로세서의 검증과 성능 분석을 기술하였으며, VI장에서는 결론을 제시하였다.

II. RC4 스트림 암호 알고리즘

본 장에서는 RC4 스트림 암호의 동작을 살펴보고 알고리즘의 하드웨어 구현시 병목이 되는 부분을 분석하였다.

2.1 RC4 스트림 암호 알고리즘

RC4 알고리즘은 RSA Data Security사의 Ron Rivest에 의해 개발된 스트림 암호 시스템으로, 외부에서 제공된 키에 따라 임의 길이의 유사 난수 시퀀스를 발생한다. 그림 1에 보는 바와 2개의 연산 단계(phase)로 구성되며j 유사 난수 발생에 2개의 256 바이트 배열(S, K)이 사용된다. 단계 1의 경우 S-배열과 K-배열의 초기화 과정(phase-1)과 S-배열의 랜덤화(randomization)과정(phase 1-b)으로 나뉜다. S-배열은 S[0]=0, S[1]=1, S[2]=2, ... 형태로 배열 인덱스와 동일한 값으로 초기화된다. K-배열은 외부 키를 사용하여 초기화된다. 단 키가 256 바이트보다 작은 길이인 N-바이트 길이를 갖고 있는 경우, K-배열의 처음 N 바이트가 키 값으로 채워지며, 나머지 256-N 바이트는 키 값이 반복 복사되어 채워진다. 이러한 2개의 배열이 초기화되면, S-배열은 단계 (1-b)에 따른 256 번의 반복 동작으로 내부 엔트리의 위치 교환(swapping) 동작을 통해 랜덤화(randomization) 된다. 이러한 반복적인 엔트리 위치 교환 과정에 2개의 주소 인덱스 (i, j)가 사용되며, j 인덱스 계산에 K-배열 값이 사용된다. 단계 2의 경우 단계 1에서 얻은 S-배열에 대해 추가의 위치 교환 동작을 거친 후 랜덤화된 키 스트림 St (= PRN[7:0])을 생성한다. 단, 단계 (1-b)의 S-배열 랜덤화 동작과 다른 점은 j-주소 인덱스 계산에 키 배열(Ki)가 사용되지 않고, i-인덱스가 0이 아닌 1부터 시작한다는 점이 다르다. 단계 2의 위치 교환에 참여한 2개의 엔트리 Si와 Sj 값을 더해서 새로운 t-주소 인덱스를 계산하여, 이 값이 가리키는 내용이 발생되는 RC4의 암호 및 복호 동작에 사용되는 키 스트림 St(=PRN[7:0])가 된다.

2.2 RC4를 사용한 암호 및 복호 동작

RC4의 경우 암호 및 복호시 동일한 동작이 적용된다. 그림 1의 단계 2(phase 2)에서 발생된 키 스트림(PRN[7:0])과 평문(plaintext), 또는 암호문

(ciphertext)과 XOR(exclusive OR) 동작을 수행하여 그림 2와 같이 암호 또는 복호 동작을 구현한다.

```
<Phase 1-a > Initialization of S-Array and K-Array
 for i=0 to 256 {
            Si=i;
            Ki=key(i mod key_length)
}
<Phase 1-b > Randomization of the S-Array
j=0
for i=0 to 256 {
            j =(j+Si+Ki) mod 256
        // swap Si and Sj
         m = Si, k = Sj
         Si = k, Sj = m
}

<Phase 2 > Generation of Pseudo-Random Stream
i=j=0
for each pseudo-random byte to be generated {
            i=(i+1) mod 256
            j=(j+Si) mod 256
          Read Si
          Read Sj
            swap Si and Sj
            t=(Si+Sj) mod 256
            pseudo-random byte is St // St=PRN[7:0]
}
```

그림 1. 유사 난수 시퀀스를 발생하는 RC4 알고리즘

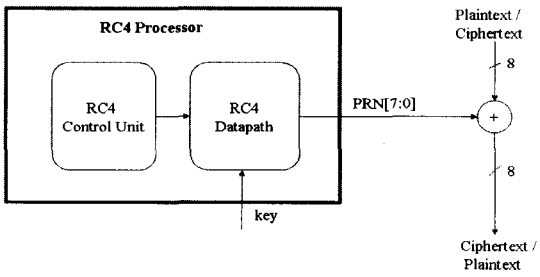


그림 2. RC4 알고리즘을 사용한 스트림 암호 및 복호 동작

Ⅲ. RC4 알고리즘의 하드웨어 설계

본 장에서는 RC4 알고리즘을 면적과 속도 측면에서 효율적인 구조로 구현하기 위해 고려한 설계 기법과 이를 바탕으로 한 하드웨어 설계를 기술하였다.

3.1 RC4 알고리즘의 하드웨어 구현 기법

RC4 알고리즘을 하드웨어로 구현할 때 면적과 속도 측면의 문제를 분석하고, 이를 해결하기 위해 본 연구에서 하드웨어 구현시 고려한 설계 기법은 다음과 같다.

첫째, 외부에서 제공하는 키를 저장하는 K-배열의 경우 그림 1의 알고리즘에서는 256 엔트리가 필요하지만(그림 3-a), 실제 키 길이에 맞게 N-바이트 길이의 기억 장치만 준비하고 K-배열에 대한 주소 인덱스에 대한 mod-N 연산을 통해 8-비트 주소 값을 $\log_2 N$ 비트로 대응시켜 그림 3-b와 같이 K-배열에 대한 면적을 감소시킬 수 있다. 이러한 기법은 K-배열의 크기는 감소시킬 수 있지만 mod-N을 위한 연산 하드웨어가 추가로 필요하다. 단, 가장 널리 사용되는 40-비트 키 길이를 사용할 경우 5 바이트의 키 길이가 필요하므로(N=5), mod-5를 위한 효율적인 모듈라 연산 회로가 필요하다. 이에 대한 체계적인 설계 기법을 3-2절에서 제시하였다. RC4 알고리즘의 경우 K-배열에 대한 접근은 순차적으로 1씩 증가하는 주소 값으로 이루어지므로, 8 비트 주소 값을 외부에서 받지 않고 그림 3-c와 같이 mod-N 카운터를 사용하는 방안도 가능하다. 단, 본 연구에서는 2가지 키 길이(128, 40-비트)만 지원하기 때문에, 2개의 mod-N 카운터를 사용하는 방식에 비해, 조합회로 구조의 mod-N 회로 방식을 채택하였다. 그러나 다양한 키 길이를 갖는 RC4 프로세서를 설계할 경우 조합 회로 구조의 mod-N 회로는 설계가 복잡하기 때문에 mod-N 카운터 방식(그림 3-c)이 효율적이다.

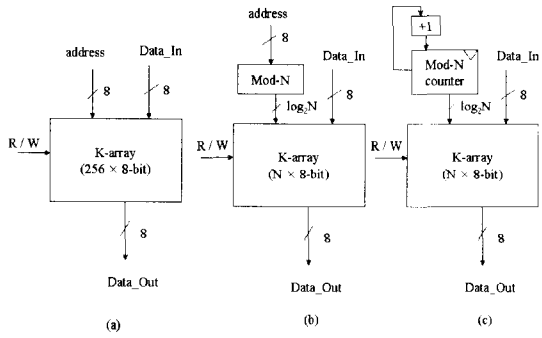


그림 3. K-배열의 크기를 감소시키는 하드웨어 구현 기법 ($N \leq 256$)
(a) 256 바이트 K-배열을 사용하는 방식,
(b) mod-N 연산 하드웨어를 내장하여 N-바이트 K-배열을 사용하는 방식(본 논문)
(c) mod-N 카운터를 사용하여 N-바이트 K-배열을 사용하는 방식

둘째, 그림 1의 단계 (1-a)에 따르면 RC4의 동작에서 S-배열의 모든 엔트리가 초기에 주소 인덱스와 동일한 값으로 초기화되어야 한다(Si = i). 이러한 동작은 RC4 알고리즘에서 S-배열의 랜덤화 동작에 앞서서 수행되어야 하는 동작으로, RC4 키 알고리즘에서 256 사이클의 초기화 대기 시간을 초래한다. 이에 대한 해결 방안으로 기존에 구현된 방식은 크게 3가지 유형이 존재한다.

- SRAM 구조의 S-배열을 순차적으로 초기화하는 방식 (그림 4-a)
- SRAM 구조의 2개의 S-배열 메모리를 갖고 있으며, 하나의 S-배열에 대해 RC4 난수 발생 동작을 수행하는 중에, 다른 하나의 S-배열에 대해 다음 RC4 동작을 위해 S-배열을 초기화하는 동작 수행 방식(그림 4-b)
  - 모드(mode) 값에 따라 사용하는 각 S-배열의 동작이 변경됨
- SRAM 구조의 S-배열이 아닌 D-플립플롭 기반의 개별 초기화 기능을 갖는 256 × 8-비트 레지스터 파일을 사용하여 전역 초기화를 하는 방식 (그림 4-c)

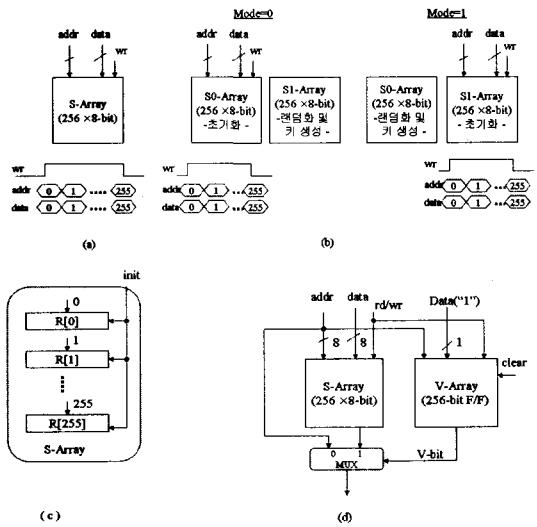


그림 4. S-배열 초기화를 위한 4가지 기법
(a) S-배열에 대한 순차적인 저장 기법
(b) 2쌍의 S-배열을 통한 S-배열 초기화 지연시간 제거 기법
(c) SRAM 구조가 아닌 D-F/F 기반의 S-배열에 대한 전역적인 초기화 기법
(d) V(valid)-비트 배열을 사용하여 S-배열 초기화 대기 시간 제거 기법 (본 논문)

그림 (4-b) 방식의 경우 그림 (4-a) 방식에 비해 S-배열의 크기가 2배인 문제와 함께, 첫 번째 RC4 동작의 경우 S-배열 초기화 대기시간이 불가피하므로, 키가 자주 변경되는 응용에만 성능 개선 효과가 있다. 그림 (4-c) 방식의 경우 1개의 클록으로 S-배열이 전부 초기화가 가능하므로 속도 측면에서 가장 큰 이점이 있지만, 그림 (4-a), (4-b)방식과는 달리 S-배열로 기존 SRAM 구조를 사용할 수 없기 때문에 면적의 증가가 매우 큰 문제가 있다.

본 연구에서는 S-배열로 SRAM 구조를 유지하면서 "Si =i"의 S-배열 초기화 동작 특성(배열 초기값이 주소 인덱스와 동일)을 활용하기 위해, 256-비트 D-F/F 구조의 V(valid)-비트 배열을 사용하여 초기화 대기 시간을 제거하는 방식을 사용하였다(그림 4-d). V-비트 레지스터 배열의 각 비트는 S-배열에 담긴 256 엔트리 내용이 타당한 정보(쓰기(write) 동작을 통해 데이터가 한 번 이상 저장된 경우 타당 의미)를 갖고 있는지를 지시한다. V-비트가 '1'인 경우 S-배열에 대한 읽기 동작시 S-배열에 저장된 내용이 읽히고, 반면 V-비트가 0인 경우 S-배열에 대한 읽기 동작시 주소 인덱스(Addr=i)가 S-배열 값 대신에 출력 값으로 선택되는 기법을 채택하였다. 이러한 V-비트 배열은 컴퓨터 시스템에서 캐쉬 메모리의 태그(Tag) 영역 V-비트와 유사한 개념이다. 따라서 이러한 방식을 사용할 경우 그림 1의 단계 (1-a)의 S-배열 초기화 동작은 제거될 수 있으며, 그림 1의 단계 1 동작이 그림 5와 같이 변경될 수 있다.

본 연구에서 사용한 방식을 기존 방식과 비교하면 다음과 같은 특징이 있다. 그림 (4-b)과 달리 하나의 S-배열을 사용함과 동시에 첫 번째 RC4 동작의 경우 S-배열 초기화에 기인한 256 클록 사이클의 초기화 대기시간(latency time)을 완전히 배제할 수 있으며, V-비트 배열로 256-비트 D-F/F만 필요하므로 그림 (4-c) 방식에 비해 면적 증가가 훨씬 작다.

셋째, 반도체 공정 기술의 발달에 따라 FPGA (Field Programmable Gate Array)와 표준 셀 라이브러리(Standard Cell Library)에서 매크로 셀(macro cell) 형태로 이중 포트(dual-port) 구조의 SRAM 메모리 블록을 제공하고 있다. 이러한 이중 포트 구조의 메모리를 사용할 경우 RC4의 동작 속도를 개선할 수 있다. 단, 이러한 메모리 블

록의 경우 대부분 동기 읽기 및 쓰기 (synchronous read and write) 동작을 채택하므로, 읽기 동작시 출력 데이터가 클록의 상승 에지에 동기화되므로 1 사이클 지연 효과가 있는 점을 고려해서 데이터패스가 설계되어야 한다. 또한 쓰기(write) 동작시 두 개의 주소 인덱스가 동일할 경우 데이터 충돌 현상이 발생하므로, 이러한 조건을 감지할 경우 쓰기 동작을 금지하는 제어가 필요하다. 본 연구의 RC4 동작의 경우 이중 포트에 대한 쓰기 동작은 S-배열 엔트리 교환(swapping)시에 필요한데, 주소 인덱스 i와 j가 동일한 경우 쓰기 동작을 방지해도 문제가 되지 않는다. 그 이유는 두개의 주소 인덱스가 동일한 경우 Si =Sj이므로, 데이터 이동 동작을 한 경우와 이동 동작을 하지 않은 경우가 동일하기 때문이다. 본 연구에서는 RC4 동작의 고속 동작을 위해 동기 이중 포트 메모리를 사용하였다.

```
<Phase 1-a > Initialization of S-Array and K_Array
for i=0 to 256 {
        // Si=i;  -- Si 초기화 동작 불필요
        Ki=key(i mod key_length)
}
<Phase 1-b > Randomization of the S-Array
j=0, V-array[255:0] = 0 // V-배열의 모든 비트를 0
for i=0 to 256 {
    // Si Read and j-index calculation
      if Vi == 0,
         H = i
      else
         H = Si
         j =(j + H +Ki) mod 256
    // Sj Read
      if Vj == 0,
         T = j
      else
         T = Sj
    // swap Si and Sj and update V-array
      Si = T  // Si에 쓰기 동작 및 Vi 변경
      Vi = 1   // Si에 대응하는 Vi = 1 처리
      Sj = H   // Sj에 쓰기 동작
      Vj ==1   // Sj에 대응하는 Vj = 1 처리
}
```

그림 5. S-배열의 초기화 대기시간을 제거한 연산 기법 (Phase 1의 개선)

3.2 MOD-5 연산 회로 설계

(2^n+1)의 구조를 갖는 수를 Fermat 수 라 하며, 모듈라 연산(modular arithmetic)에서는 유용한 특성을 갖는다. Fermat 수의 경우 mod (2^n+1) 연산시 다음 연산 특성이 성립한다[10].

$$
\begin{aligned}
&(A + B) \bmod M = (A \bmod M + B \bmod M) \bmod M \\
&(A \times B) \bmod M = (A \bmod M \times B \bmod M) \bmod M \\
&(-c) \bmod M = (M-c) \bmod M \\
&2^n \bmod (2^n+1) = 2^n - (2^n+1) = -1
\end{aligned}
\tag{1}
$$

A가 2n-비트 수인 경우 식 2와 그림 6과 같이 표현될 수 있다.

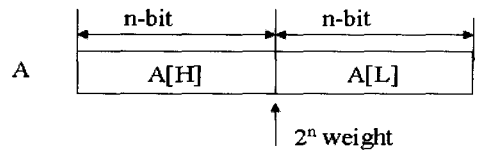


그림 6. 2n-bit 길이를 갖는 양의 정수 A

$$
\begin{aligned}
A &= \sum_{i=0}^{2n-1} 2^i a_i = A[H]2^n + A[L] \\
&= ((A div\ 2^n)2^n + (A mod\ 2^n)) \bmod (2^n+1) \\
&= (A \bmod 2^n - A \div 2^n) \bmod (2^n+1)
\end{aligned}
$$

$$
A = \begin{cases} (A[L]-A[H]), & \text{if } (A[L]-A[H] \geq 0 \\ (2^n+1+(A[L]-A[H])), & else \end{cases}
\tag{2}
$$

식 (2)에서 (A mod 2^n - A div 2^n) mod (2^n+1)이 의미하는 바는 A[L]에서 A[H]를 뺀 결과가 0보다 같거나 큰 경우, 즉 캐리 출력(Cout)이 1인 경우, 결과는 뺀 값이지만, 뺀 결과가 음수인 경우 (A[L] - A[H])에 (2^n+1)를 더한 값이 된다. 단, mod(2^n+1)의 경우 최대 결과 값이 2^n이므로

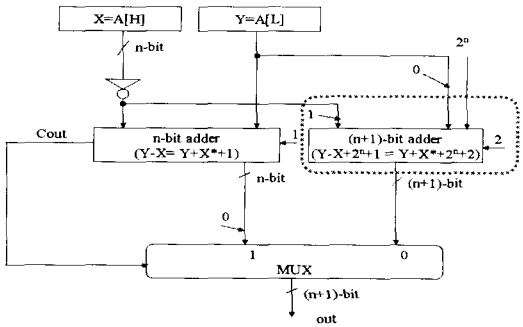


그림 7. A mod (2^n+1)에 대한 연산회로(L1_MOD 회로)

출력을 표현하기 위해 (n+1) 비트가 필요하다. 본 연구에서는 식 (2)을 구현하기 위해 기존 캐리 선택 덧셈기(carry select adder) 기법과 유사하게 그림 7과 같이 식(2)의 2가지 예상 출력을 동시에 수행한 후, (A[L]-B[H])연산의 캐리 출력 값(Cout)에 따라 결과를 선택하는 기법을 채택하였다. 단 $(2^n+1+(A[L]-A[H]))$의 경우 (n+1) 비트 출력이 발생할 수 있기 때문에 (n+1) 비트 덧셈기를 사용하였으며, 3개의 오퍼랜드 덧셈을 처리하기 위해, 캐리 보존 덧셈기(Carry Save Adder)와 캐리 전달 덧셈기(Carry Propagate Adder)를 사용하였다.

본 연구의 RC4 프로세서는 40-비트(5 바이트) 키 지원으로 A mod (2^2+1) 연산 기능이 필요하다. 본 연구의 A의 길이가 8-비트로 4n-비트 특성을 가지므로, 그림 7을 바로 적용할 수 없고 약간의 수정이 필요하다. 즉 본 연구에서 설계된 A mod-5 회로는 그림 8과 같이 2개의 레벨로 구성된다. L1_MOD회로는 $2^n(=4)$ 이하인 3-비트 출력을 생성하므로, 2개의 출력을 더한 결과는 8보다 작거나 같게 된다. 따라서 결과 값이 5이상인 경우는 -5를 추가로 수행해서 결과를 교정하는 동작이 필요하다. 따라서 이러한 순차적인 연산을 방지하기 위해 캐리 선택 가산기와 유사하게 A+B와 A+B-5를 병렬로 수행하고, A+B-5의 캐리 출력(Cout)을 검사하여, 2가지 출력 중에 하나를 선택하는 구조를 채택하였다.

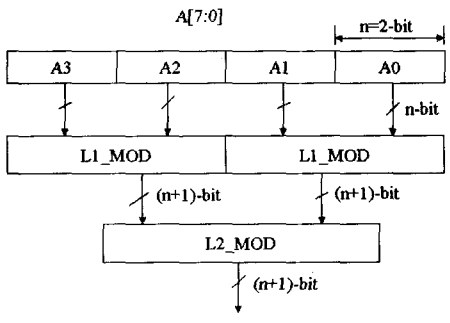


그림 8. A mod 5 회로(8-비트 A)

3.3 RC4 프로세서의 설계 사양

본 논문에서 설계한 RC4 프로세서의 설계 사양은 다음과 같다.

첫째, 지원하는 키 길이로 다양한 분야 적용을 위해, 가장 보편적으로 사용되는 2가지 키, 40-비트와 128-비트 키 길이를 지원할 수 있도록 하였다.

둘째, 키를 저장하는 K-배열은 모듈라 연산 기능을 갖는 주소 인덱스를 채택함에 의해 크기가 크지 않는 점을 고려하여, 고속 동작에 적합하게 D-F/F 기반의 128-비트 레지스터와 MUX로 구성하였다.

셋째, S-배열은 초기화 대기 시간을 제거하고 연산 속도를 향상시키기 위해 이중 포트 구조의 256-바이트 SRAM 과 이중 포트 구조의 256-비트 V-비트 배열 구조를 사용하였다.

넷째, 그림 1의 S-배열 랜덤화 동작과 키 스트림 생성과 관련된 반복 루프 동작을 3개의 클록 사이클에 수행될 수 있도록 하는 기법을 채택하였다.

다섯째, 모듈라 연산을 위한 효율적인 하드웨어 기법을 통해, 모듈라 연산이 성능을 저하시키지 않도록 하였다.

여섯째, 3개 오퍼랜드 덧셈 동작은 캐리 보존 덧셈(carry save addition)과 캐리 전달 덧셈(carry propagate addition)을 결합시킨 연산 기법을 통해 지연시간을 최소화하였다.

3.4 RC4 알고리즘의 하드웨어 설계

3.1절과 3.2절에서 제시한 설계 기법과 모듈라 연산 기법을 바탕으로 설계된 RC4 프로세서의 블록도는 그림 9와 같다. RC4 프로세서는 크게 제어부, 데이터 패스부, 발생되는 유사 난수 바이트 길이를 제어하기 위한 LEN_REG 로 구성된다.

K-배열 메모리와 LEN_REG를 외부에서 초기화한 후, Start 신호로 RC4 동작을 시작한다. KR_Busy 신호는 RC4 프로세서가 그림 1의 단계 1 동작을 수행중임을 지시하는 플래그이며, Busy 신호는 RC4 프로세서가 단계 1 또는 단계 2 동작을 수행중임을 지시한다. KR_Busy신호 와 Busy 신호로 플래그를 분리한 이유는 RC4 알고리즘의 단

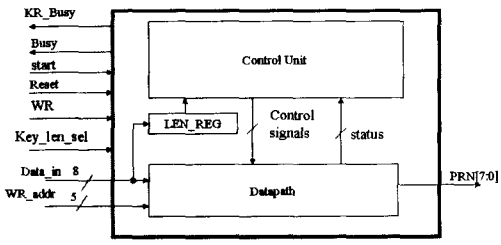


그림 9. RC4 프로세서의 블록도

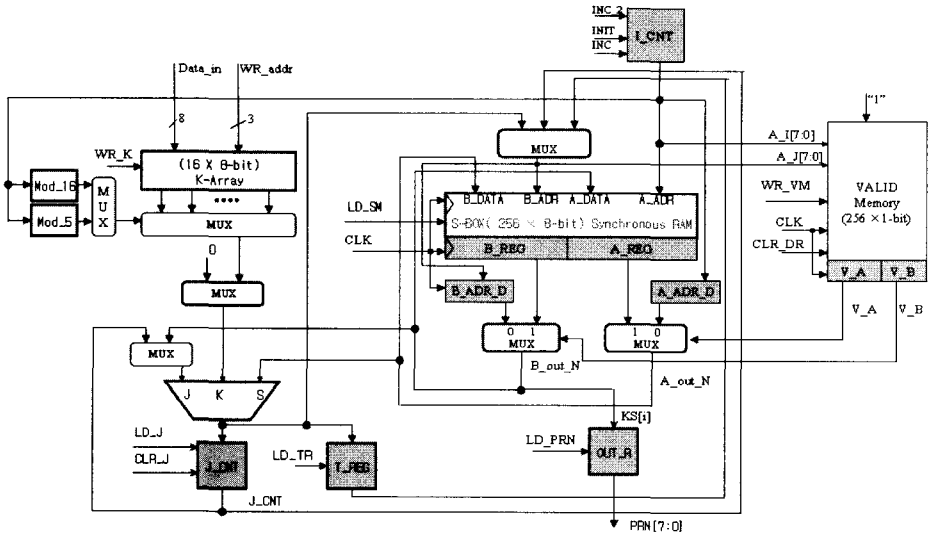


그림 10. RC4 데이터패스 블록도

계 2 동작 중에는 K-배열 메모리가 사용되지 않기 때문에, 새로운 키를 사용하는 다른 RC4 동작이 뒤에 수행되는 경우 앞선 RC4 동작 완료를 기다리지 않고, K-배열을 외부에서 자원 충돌 없이 초기화할 수 있도록 하기 위해 도입하였다. 그림 10은 RC4 프로세서 내부 데이터패스의 블록도를 나타내며, 3.1절에서 제시한 설계 기법이 반영되어 있다. 그림 11은 데이터 패스를 이용하여 RC4 동작을 수행하기 위해, 제어 회로 설계를 위해 정의한 ASM (Algorithmic State Machine)을 나타낸다. ASM 도표에 따르면 상태 T1, T2, T3은 S-배열 랜덤화 동작 제어에 사용되며, 상태 T4, T5, T6, T9는 유사 난수, 즉 키 스트림 생성 동작을 제어한다. S-배열 랜덤화 동작을 구현하기 위한 ASM 도표의 동작을 분석해 보면, T1 상태 동안 Si를 읽고 T2 상태동안 J_CNT를 변경함과 동시에 Sj를 읽게 된다. 그리고 T3 상태동안 Si와 Si간의 데이터 이동 동작과 V-엔트리 변경 동작이 수행된다. 유사한 과정이 키 스트림 생성과정(T4, T5, T6)에서 수행된다. T4, T5, T6은 키 스트림 생성을 위한 반복 루프를 형성하며, T9 상태는 T6 상태보다 1 사이클 뒤에 활성화되며, 2번째 반복 루프의 T4와 동일 타이밍에 수행된다. T4 상태 박스 내부를 보면 S 배열의 B-포트가 사용되지 않기 때문에, T9 상태 동안 자원 충돌 없이 S 배열의 B-포트를 통해 키 스트림, KS[i]를 출력할 수 있다. 따라서 이러한 기법을 통해 단계 2의 반복 루프 동작이 4 사이클이 아닌 3클록 사이클 만에 처리가 가능하다. 그림 12는 RC4 동작에 대한 타이밍 동작을 나타낸다. 키 스트림을 생성하는 단계 2 동작의 경우 2번째 반복 루프부터 T4 상태와 T9 상태가 동시에 활성화됨을 알 수 있다. 그리고 시작 신호를 인가한 후 첫 번째 키 스트림 KS[0]이 발생할 때 까지 (256 × 3) 클록 사이클의 초기 대기 시간(latency)이 존재함을 알 수 있다. 이러한 대기시간 후에는 3 개의 클록마다 유사 난수 KS[i]를 발생함을 알 수 있다. 단, S-배열 메모리를 통해 T9 상태에서 출력되는 KS[i]는 1 사이클 동안만 안정되므로, 외부에서 3 사이클 동안 안정된 신호를 사용하기 위해 OUT_R 레지스터에 저장하여 외부로 출력하도록 설계하였다. Data-path의 내부의 S-배열 메모리의 경우 초기화 동작을 제거하고 연산 사이클 수를 줄이기 위해 이중 포트 구조의 동기 메모리가 사용되었으며, V(valid)-비트 메모리 역시 이중 포트 구조에 맞게 2개의 V-비트(V-A, V-B)를 출력한다. 단, S-배열 메모리가 동기 읽기(Synchronous Read) 특성으로 데이터가 클록 상승 에지에 맞추어 1 사이클 지연되어 출력되는 특성을 갖고 있기 때문에, 2개의 주소 값 (A_ADR, B_ADR)도 1 사이클 지연시키기 위해, 외부에 1 사이클 지연 용도의 B_ADR_D와 A_ADR_D 레지스터가 사용되었다. 그리고 기존 방식[7]에서는 2개의 덧셈기를 사용하는 데 비해, 본 연구에서는 입력은 선택하는 MUX를 추가해 하나의 덧셈기로 동작이 가능하도록 하였다. 그림 10에서

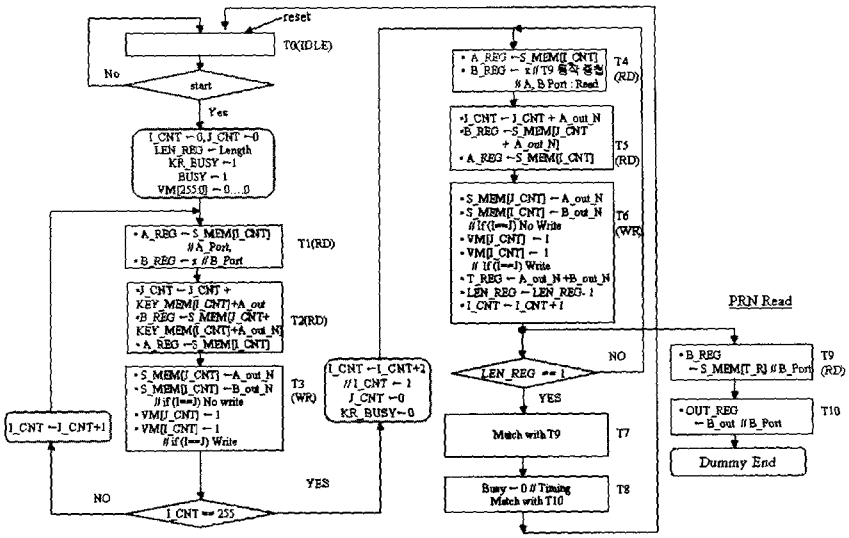


그림 11. RC4 프로세서 동작을 제어하기 위한 ASM

40 비트와 128 비트 키에 따라 K-배열의 출력을 선택하기 위해 2개의 MOD 회로(MOD-5, MOD-16)가 사용되었다. 단, MOD-16은 주소 값의 하위 4 비트를 취하면 되며 MOD-5 회로는 3.2절의 설계 기법으로 구현하였다. 그리고 3개의 오퍼랜드를 더하는 덧셈 연산의 경우 2개의 덧셈기 대신에 CSA(carry save adder)와 CPA(carry propagate adder)를 사용하여 연산 지연시간을 최소화하였다.

III. FPGA 구현 및 성능 분석

본 연구에서 설계한 암호 프로세서는 Verilog HDL 언어로 모델링을 하였다. RC4를 구현하는 C 프로그램[11]과 설계된 회로와의 결과 비교 동작을 통해 회로가 올바른 동작을 수행함을 확인하였다. 그림 12는 NC-Verilog 시뮬레이터로 검증한 40-비트 RC4 프로세서의 검증 파형을 나타낸다. 전체 회로는 Xilinx XCV1000E-6HQ240C FPGA Device로 구현하여 성능을 평가하였다. 본 연구에서 설계된 RC4 회로 내부의 S-배열 메모리를 구현하기 위해 Xilinx 사의 Core Generator 소프트웨어를 사용하여 이중 포트 구조의 동기 메모리를 생성하여 사용하였다. 그림 13은 PCI 인터페이스 보드에 탑재된 Xilinx FPGA에 구현된 RC4 프로세서의 검증 환경을 나타낸다. 단, 이러한 PCI 인터페이스를 위해 기존 RC4에 PCI 인터페이스를 위한 기본 기능을 추가하였다. PC 화면을 확대한 테스트 결과의 좌측 화면은 FPGA에서 수행된 결과이고, 우측 화면은 RC4를 소프트웨어로 구현한 결과를 나타낸다. 검증 결과 2가지 데이터가 올바른 결과를 발생함을 확인하였다.

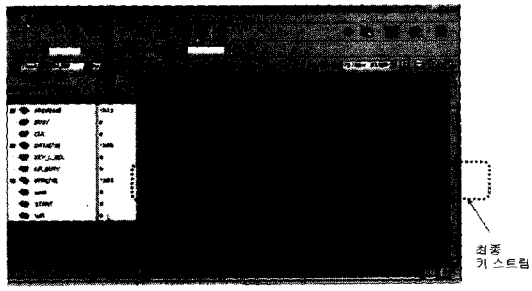


그림 12. 40-비트 키에 대한 RC4 프로세서에 대한 검증 파형 (사용한 Key 값 = 0x0123456789)

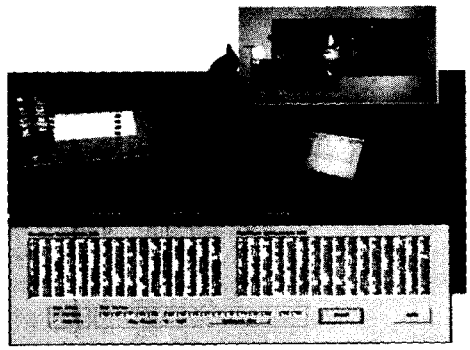


그림 13. RC4 프로세서에 대한 FPGA 검증

표 1. RC4 FPGA 구현 결과

FPGA 디바이스	Xilinx XCV1000E-6HQ240C	
Area Allocation	Used/Available	Utilizations
CLB slices	651 / 12288	5%
Slice Flip Flops	578 / 24576	2%
bonded IOBs	27/408	4%
GCLKs	1/4	25%
RAM block	256 bytes	
Frequency(MHz)	약 40 Mhz (57.4 % routing delay)	
Throughput	106 Mbps	
Latency time	3 × 256 = 768 clocks	

FPGA 합성 결과는 표 1과 같다. FPGA 구현 결과 약 40Mhz의 최대 가능 동작 주파수를 가짐을 확인할 수 있었다. 설계된 회로는 그림 14에 보이는 바와 같이 S-배열 랜덤화 동작을 위한 768 클록의 초기 대기 지연(latency)후에 3 클록마다 8 비트의 출력을 생성하는 구조를 갖는다. 따라서 본 연구의 RC4 프로세서의 유사 난수(키 스트림) 생성율(throughput)과 레이턴시(latency)는 식 (3)과 같이 정의될 수 있다. 식 (3)에 따르면 본 연구의 RC4 프로세서는 약 106 Mbps의 유사 난수 생성율을 가짐을 알 수 있다.

$$Throughput\ of\ RC4\ processor(bps) = \frac{8*f}{3}$$
$$Latency(\#\ of\ clock) = 3*256$$
$$where\ \ f = clock\ frequency \qquad (3)$$

서로 다른 키를 사용하는 RC4 동작이 반복되는 응용의 경우, 본 연구에서 설계한 회로를 그림 16과 같이 수정하면, RC4 알고리즘의 단계 1과 단계 2 동작을 중첩시키는 파이프라인 동작을 통해 두 번째 RC4 동작부터 S-배열 대기시간을 제거할 수 있다. 즉 본 연구의 S-배열 메모리를 2개의 쌍으로 구성하고 모드 신호에 따라 사용하는 S-배열을 각 동작이 바꾸어 사용하게 되면 2개의 인접한 RC4 동작의 단계 1과 단계 2 동작을 중첩시킬 수 있다. 그림 (4-b) 방식과 유사하지만 차이점은 본 연구에서는 S-배열을 초기화하는 동작이 V-비트 메모리를 통해 제거되므로, 키 설정과 S-배열 랜덤화에 더 많은 시간 할당이 가능하므로 보다 효율적으로 파이프라인 처리가 가능하다. 단, K-배열은 RC4 단계-1 동작에서만 사용되므로 2 쌍이 존재할 필요가 없다. 표 2는 구현 환경에 따른 차이점을 배제하기 위해 기존 방식과 본 연구에서 제안한 방식을 구조적인 측면에서 성능을 비교한 결과를 나타낸다.

표 2. RC4 프로세서의 구조적인 측면 성능 비교

RC4 구조	참고 문헌 [7]	참고 문헌 [8]	참고 문헌 [9]	본 연구 방식
S-배열 구조 / 크기	단일포트 SRAM 구조 /(256× 8-bit)	이중포트 SRAM 구조 / (2×256 × 8-bit)	이중포트 레지스터 파일 구조 / (3×256 × 8-bit)	이중 포트 SRAM 구조 + V-배열 / (256 × 8-bit + 256 × 1-bit)
gate count (0.3um 표준 셀)	-	-	-	8,400 + dual-port memory block
S-배열 초기화 시간	256 clocks	256 clocks	0 clocks (각 레지스터를 다른 값으로 전역 초기화)	0 clock (0으로 V-메모리 전역 초기화)
S-배열 랜덤화 시간	5 *256= 1,280 clocks	3 *256= 768 clocks	3 *256= 768 clocks	3 * 256= 768 clocks
키 스트림 생성율 (f : freq.)	8 × f / 6	8 × f / 3	8 × f / 3	8 × f / 3
지원하는 키 길이	32-bit	40-bit	8 ~ 128-bit	40-bit /128-bit

표 2에 따르면 본 연구의 RC4 프로세서는 기존 방식에 비해 V-비트 배열과 이중 포트 메모리를 사

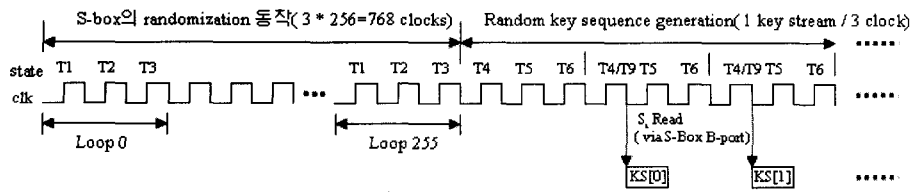


그림 14. 유사 난수 형태의 키 스트림을 생성하는 RC4 프로세서 타이밍도

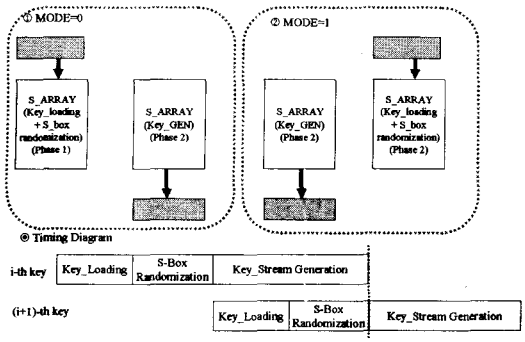


그림 15. 2개의 S-Array 배열을 사용하여 레이턴스 시간 제거 기법

용함에 의해 적은 하드웨어 추가로 S-박스 초기화 동작을 제거할 수 있으며, 체계적인 모듈라 연산 하드웨어 사용으로 40-비트와 128-비트 길이의 키를 지원한다. 그리고 설계된 회로를 IDEC MPW 삼성 0.35um 표준 셀 공정으로 합성할 경우 약 8,400개의 랜덤 게이트와 1개의 이중 포트 메모리로 구성된다. 단 랜덤 게이트 중 약 15% 정도가 V-비트 메모리에 기인한 부분이다. 또한 3개의 클록마다 8-비트의 유사난수 출력을 생성하므로 고속의 RC4 동작이 필요한 분야에 효율적으로 응용 가능하다. 그리고 본 연구에서 제안한 V-비트 배열을 사용할 경우, RC4 동작이 반복되는 응용에서 그림 15와 같이 2쌍의 S-배열을 사용하여, 키 스트림 생성 동작과 별도의 S-배열을 이용한 키 초기화 S-배열 랜덤화 동작을 중첩시킴에 의해 두 번째 RC4 동작부터 S-박스 랜덤화 동작에 기인한 대기시간을 완전히 제거할 수 있다. 단, 모드 값에 따라 키 생성과 키-초기화가 사용되는 S-배열이 바뀌게 된다. 본 연구에서 S-배열 초기화 시간을 제거하기 위해 추가되는 이중 포트 구조를 갖는 256-비트 F/F 하드웨어는 많은 면적 오버헤드를 야기하므로, 속도가 최적화되는 RC4 암호 응용과 키 검색 엔진에 적합하다.

IV. 결 론

본 논문에서는 다양한 응용 분야에 적용되고 있는 RC4 알고리즘에 대한 효율적인 하드웨어 구현 기법을 제안하고 이를 FPGA로 구현한 후 성능을 분석하였다. 설계된 회로는 기존 방식과 달리 V(valid)-비트 메모리를 채택하여, S-배열 초기화 동작에 따른 대기 시간을 제거하였으며, 이중 포트 구조의 S-배열 메모리를 채택하여 3 사이클마다 키 스트림을 생성할 수 있는 구조를 갖는다. 또한 체계적인 모듈라 연산 하드웨어를 사용하여 40-비트와 128-비트 키 길이를 지원할 수 있다. 그리고 제안한 모듈라 연산 구조는 (2^n+1) 모듈러스(modulus)를 갖는 다른 RC4 키에 적용가능하다. 단, (2n+1) 모듈러스(modulus) 특성을 갖지 않은 다양한 키 길이의 RC4 프로세서를 설계하는 경우 MOD-N 카운터 구조를 사용하는 방식이 타당할 것으로 판단된다. 설계한 RC4 프로세서를 Xilinx XCV1000E-6HQ240C FPGA Device로 구현할 결과 약 40 Mhz의 동작주파수를 가지며, 약 106 Mbps의 유사 난수 발생 율을 가지며, 768 클록의 S-배열 랜덤화 초기 대기 시간(latency)을 갖고 있다. 최근에 유비쿼터스 환경과 무선 통신 연구가 활발함에 따라 저전력 특성과 적은 면적을 차지하는 스트림 암호 알고리즘을 이러한 분야에 적용하는 연구가 확대되고 있다. 따라서 본 논문에서 설계한 RC4 암호 프로세서는 병렬로 구성되어 RC4 키 검색 엔진에 적용되거나 IP(Intellectual Property) 형태로 가공되어 무선 랜 표준인 WEP와 RC4 알고리즘이 필요한 멀티미디어 단말기 환경 등에 보안 모듈로 사용될 수 있을 것으로 판단된다.

참 고 문 헌

[1] 박춘식, "유비쿼터스 네트워크와 시큐리티 고찰", 정보보호학회지, 제14권 제 1호, pp.12-20, 2004.2.

[2] 안하기, 신경욱, "AES Rijndael 블록 암호 알고리듬의 효율적인 하드웨어 구현", 정보 보호 학회 논문지, 제 12권 2호, pp.53-64, 2002. 4.

[3] Bruce Schneir, Applied Cryptography-Protocols, Algorithms and Source Code in C, 2nd Edition, John Wiley and Sons, 1996.

[4] S. Fluher, I.Mantin, and A. Shamir, "Weakness in the key scheduling algorithm of the RC4," In Proc. 8th Workshop on Selected Area in Cryptography, LNCS 2259, Springer-Verlag, 2001.

[5] Panu Hamalainen, Marko Hannikainen,

Timo Hamalainen, and Kukka Saarinen, "Hardware Implementation of the Improved WEP and RC4 Encryption Algorithms for Wireless Terminals", The European Signal Processing Conference (EUSIPCO '2000), September, pp.2289-2292, 2000.

[6] I. Goldberg and D. Wagner, "Architectural consideration for cryptographic hardware", http://www.cs.berkeley.edu/~iang/issac/hardware/, 1996.

[7] Paul D. Kundarewich, Steven J.E. Wilton, Alan J. Hu, "A CPLD-Based RC-4 Cracking System", IEEE Proceeding of the Canadian Conference on Electrical and Computer Engineering, May 1999.

[8] K.T.Tsoi, K.H.Lee, and P.H.W. Leong, "A Massively Parallel RC4 Key Search Engine", Proc. of the 10th Annual IEEE Symposium on Field- Programmable Custom Computing Machines (FCCM'02), pp.13-21, 2002.

[9] P.Kitsos, G.Kostopoulos, N. Sklavos, and O. Koufopavlou, "Hardware Implementation of the RC4 Stream Cipher", 46th IEEE Midwest Symposium on Circuits & Systems '03, Egypt, 2003.

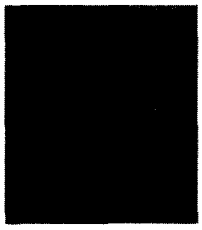
[10] Reto Zimmermann, "Efficient VLSI Implementation of Modulo (2n ± 1) Addition and Multiplication, Proceedings 14th IEEE Symposium on Computer Arithmetic, pp.158-167, 1999.

[11] Confirmed Test Vector and Program for RC4, on line available at http://www.qrst.de/html/dsds/rc4.htm, 2004.

〈著者紹介〉

최 병 윤 (Byeong Yoon Choi) 종신회원
1985년 2월 : 연세대학교 전자공학과 졸업
1987년 2월 : 연세대학교 전자공학과 석사
1992년 8월 : 연세대학교 전자공학과 박사
1997년 8월~1998년 8월 : 미국 일리노이 주립대, 객원 교수
1993년 2월~현재 : 동의대학교 컴퓨터공학과 정교수
〈관심분야〉 정보 통신용 VLSI 설계, RISC와 암호 프로세서 설계, 그래픽 프로세서 설계

이 종 형 (Jong-Hyung Lee)
1987년 2월 : 연세대학교 전자공학과 (공학사)
1990년 2월 : 연세대학교 전자공학과 (공학석사)
2000년 5월 : Ph. D., Electrical & Computer Engineering, Virginia Tech
1990년 1월~1994년 6월 : (주)대우 반도체 사업본부
2000년 5월~2001년 5월 : Principal R&D Engineer, Advanced Technology Lab., Sprint
2001년 5월~2002년 2월 : System Engineer, Opthos, San Carlos, CA
2002년 3월~현재 : 동의대학교 전자공학과 조교수
〈관심분야〉 광통신, VLSI 설계, 정보보호



조 현 숙 (Hyun-Sook Cho)
1998년~2001년 : 충북 대학교 전자 계산 학과 (이학 박사)
1982년~2002년 : 한국 전자 통신 연구원(ETRI) 지상 SW 실장
한국 전자 통신 연구원(ETRI) 정보 보호 시스템 연구 부장
한국 전자 통신 연구원(ETRI) 정보 보호 연구 본부장 역임
2003년~현재 : 한국 전자 통신 연구원(ETRI) 차세대 시큐리티 기술 개발 사업 단장
〈관심분야〉 Network Security, 이동인터넷 보안, Conditional Access