논문 2004-41SD-12-10

무선 내장형 시스템을 위한 저비용 AES의 구현

(Low-Cost AES Implementation for Wireless Embedded Systems)

이 동 호*

(Dong-Ho LEE)

요 약

AES는 인터넷 프로토콜의 대칭키 보안 알고리즘으로 널리 사용된다. 무선 내장형 시스템들이 점점 더 전통적인 유선 네트워크 프로토콜을 많이 사용하고 있으므로 이들 무선 내장형 시스템을 위한 저비용 AES 알고리즘 구현은 매우 중요하다. 가장 기본적인 AES 아키텍처는 키 스케줄을 포함하여 20개의 S-box를 사용하는 하나의 cipher 라운드로 구성되어 있다. 암호화는 동일한 라운드를 반복하여 완료된다. 근래에 이 방법의 구현 비용을 더욱 줄이기 위하여 오직 8개의 S-box만 사용하는 folded architecture가 제안되었다. 본 논문에서는 folded architecture를 이용하여 무선 통신 기술을 위한 저비용 AES 구현 구조에 대하여 연구한다. 먼저 folded architecture를 개선하여 16 바이트의 추가적인 상태 메모리 사용을 줄였다. 구현 비용을 더욱 줄이기 위하여 데이터 암호화에 하나의 S-box만 사용하는 single byte architecture를 구현하였다. Single byte architecture는 암호화에 352 클록이 소요된다. FPGA 구현 시 최대 동작 주파수는 40MHz에 도달하였다. 따라서 암호화 속도는 13Mbps 이상으로 3G 무선통신에 충분하다.

Abstract

AES is frequently used as a symmetric cryptography algorithm for the Internet. Wireless embedded systems increasingly use more conventional wired network protocols. Hence, it is important to have low-cost implementations of AES for them. The basic architecture of AES unrolls only one full cipher round which uses 20 S-boxes together with the key scheduler and the algorithm repeatedly executes it. To reduce the implementation cost further, the folded architecture which uses only eight S-box units was studied in the recent years. In this paper, we will study a low-cost AES implementation for wireless communication technology based on the folded architecture. We first improve the folded architecture to avoid the sixteen bytes of additional state memory. Then, we implemented a single byte architecture where only one S-box unit is used for data encryption and key scheduling. It takes 352 clocks to finish a complete encryption. We found that the maximum clock frequency of its FPGA implementation reaches about 40 MHz. It can achieve about 13 Mbps which is enough for 3G wireless communication technology.

Keywords: Cryptography, Computer Architecture, Wireless Communication, Security

I. 서 론

무선 handset과 같은 저비용 임베디드 시스템이 인터넷 데이터 접근이나 인터넷 응용 프로그램에 사용될 수 있게 하기 위해서 전통적인 유선 인터넷 프로토콜이 점점 더 많이 사용되고 있다. 과거로부터 무선 기반 전송 기술에서 사용되어 온 보안 프로토콜은 대부분의 경우 높은 수준의 보안을 요구하는 데이터 처리에는 충분하지 못하므로 무선 정보 통신 기기는 유선 프로토콜에서 사용되는 보안 알고리즘을 사용하게 될 것이다. 따라서 유선 프로토콜에서 널리 사용되는 대칭키 암호 알고리즘들이 저가의 정보 응용 기기에 사용자 인증, 전자 서명 생성 등의 암호 알고리즘을 위해 사용될 가능성이 높다.

데이터 통신에서 암호화 속도와 보안 능력을 향상시키기 위하여 미국의 NIST (National Institute of Standards and Technology)는 2001년 DES(Data Encryption Standard)를 대치하기 위한 AES(Advanced Encryption Standard)로 Rijndael 알고리즘을 선정하였다[2]. AES의 중요성이 증가함에 따라 AES의 ASIC 및

* 정회원, 경북대학교 전자전기컴퓨터학부
(School of Electrical Engineering and Computer Science, Kyungpook National University)
접수일자: 2004년4월30일, 수정완료일: 2004년12월2일

FPGA 구현에 관한 많은 연구들이 발표되었다[2-12]. 현재까지 대부분의 AES 구현 연구는 고급 시스템에 응용되는 고성능 설계에 관한 것들이다. 자원이 제한된 무선 응용 시스템에서 보안상 안전한 전자 데이터 교환의 필요성이 증대됨에 따라 저비용 AES의 구현이 매우 중요하다[13].

대부분의 무선 handheld 장치들은 매우 높은 암호화 속도를 요구하지 않는다. 현재 무선 네트워크의 최대 속도는 60Mbps 정도이며 많은 저급 응용 분야 장치들은 이보다 더 낮은 속도로 작동한다. 그러나 이러한 저급 장치들의 경우에도 보안 기능이 첨가되면 사용되는 마이크로프로세서에 상당한 계산 능력을 요구한다. 예를 들어, 무선 통신 기술이 2G로부터 2.5G, 3G로 발달하게 됨에 따라 통신 속도는 10kbps로부터 64kbps, 2Mbps 등으로 높아진다[1]. 반도체 기술이 0.35u로부터 0.13u으로 진보함에 따라 저비용 무선 handset에 사용되는 프로세서는 Intel SA-1100에서 Intel XScale 등으로 발전하였다. 반도체 기술이 0.1u로 진보하여도 저비용 무선 handset에 사용되는 프로세서의 성능은 800MIPS를 넘지 않을 것으로 보인다[1]. 2Mbps 무선 통신을 위해서는 25000MIPS 성능의 프로세서를 사용하여야 하므로 소위 "무선 보안 처리 갭"이 존재한다. SoC 설계 환경에서는 소프트웨어-하드웨어 동시 설계 기법을 이용하여 필요한 응용 분야의 보안 알고리즘의 성능을 만족하는 최소한의 하드웨어를 사용할 수 있다. 또한 무선 handset의 경우 전력 소모가 작은 암호 알고리즘이 필요하다. 하드웨어 구현이 소프트웨어 구현이 비하여 저전력에 유리하다는 점은 널리 알려져 있다. 따라서 무선 임베디드 응용 분야를 위한 저비용 AES 하드웨어 구현에 관한 연구는 매우 중요하다. 본 논문에서는 [6]에서 제안된 folded architecture를 개선하고 자원 요구를 최소화한 "single byte architecture"에 관하여 연구한다.

대부분의 다른 대칭키 암호 알고리즘처럼 AES 알고리즘도 사용되는 키의 크기에 따라 정해진 회수만큼 동일한 계산을 반복 수행한다. 이 동일한 계산을 라운드(round)라 한다. 예를 들어 128 비트 키 AES 알고리즘은 간단한 초기 라운드 이후 10 라운드를 반복 수행한다. 본 논문에서는 이 간단한 알고리즘에서 출발하여 무선 handset 등에서 사용될 저비용 AES 알고리즘 하드웨어 구현에 대하여 연구한다. 본 논문에서는 128 비트 key-agile 암호화 알고리즘에 대하여 주로 연구한다. Key-agile 암호화 알고리즘은 암호 수행 과정에 키 스케줄도 함께 수행한다. 이 방법은 키 스케줄이 완료된 후 암호화를 수행하는 key-save 알고리즘보다 면적 소요가 적다. 연구 과정에서는 11 클록 사이클에 암호화가 완료되는 기본적인 AES 알고리즘을 먼저 구현하였다. 이 구현 방법은 20 8x256 비트 ROM이 필요하므로 저비용 응용에는 부적당하다. 필요한 칩 면적을 감소시키기 위하여, 8개의 8x256 ROM이 사용되는 44 사이클 알고리즘을 구현하였다. 알고리즘 구현에 사용되는 칩 면적을 더욱 줄이기 위하여 176 사이클 설계와 352 사이클 설계를 구현하였다. Altera Quartus-II FPGA 설계 환경에서 동작 주파수 제한 없이 합성하였을 때 전자는 40MHz 이상의 최대 동작 주파수를 가지며 352 사이클 버전은 38.17MHz의 최대 동작 주파수를 가진다. 데이터 블록 크기가 128 비트이므로 이것은 13Mbps에 해당된다. 이것은 3G 무선 통신 기술에서 요구되는 전송 속도 2Mbps를 충분히 만족시킨다.

Ⅱ. 본 론

많은 기존 AES 하드웨어 설계에 관한 연구는 AES 사양에 제시된 소프트웨어 구현 알고리즘을 직접 하드웨어로 구현하였다. AES 사양은 각 라운드를 구성하는 ByteSub, ShiftRow, MixColumn, Add-Key의 연산을 $GF(2^8)$ 유한체 연산과 $GF((2^8)^4)$ 유한체 연산을 이용하여 기술하고 있다. 유한체 연산은 정수 연산과는 달리 캐리 전송이 필요하지 않으므로 하드웨어로 구현하기가 매우 편리하다.

AES 구현을 조직적으로 연구하기 위해서는 완전한 라운드를 한 클록 사이클에 수행하는 기본적인 방법에서 출발하는 것이 편리하다. 이 기본적인 연산 방법에서는 라운드 계산을 반복적으로 수행하여 전체 암호화를 완료한다. 일반적으로 소프트웨어 알고리즘을 하드웨어로 구현하면 프로세서의 명령어 처리 비용을 제거할 수 있으므로 약 3-10배의 성능 향상을 기대할 수 있다. 또한 AES 알고리즘은 본질적으로 많은 병렬성을 가지고 있으므로 많은 하드웨어 자원을 투입하면 쉽게 연산 속도를 향상시킬 수 있다[12]. 추가적으로 많은 하드웨어 자원을 투입하여 다양한 loop unrolling과 파이프라인 구조를 사용하면 프로세서의 연산 속도를 극대화할 수 있다[7,8,9]. 본 연구에서는 먼저 1 클록 1 라운드 알고리즘을 하드웨어 언어로 구현하였다. 이를 기반으로 하여 참고문헌 [6]에 제시한 folded architecture를 개선하여 무선 임베디드 시스템과 같은 저비용 시스템

을 위한 저비용의 간결한 AES 구현에 대하여 연구한다. 구체적인 연구 목표는 8 비트 마이크로프로세서에서 소프트웨어 구현과 같이 하나의 S-box 테이블을 가지면서 176 클록 또는 352 클록에 암호화를 완료하는 최적의 "single byte architecture"를 구현하고 합성 결과를 통하여 자원 요구 사항을 분석하는 것이다. 본 연구에서는 AES 사양의 병렬 연산 구조 알고리즘을 하드웨어 언어로 구현한 다음 단계적으로 병렬성을 줄이는 방법으로 single byte architecture를 얻었다. 이 방법이 소프트웨어 코드에서 직접 single byte architecture를 얻는 것보다 효과적인 설계를 얻는 방법으로 생각된다.

1. Folded architecture

본 논문에서는 간결한 AES 구현을 위해서 먼저 AES 사양에 제안된 기본적인 구현 방법을 분석한다[1]. 앞에서 말한 바와 같이 AES 알고리즘은 사용되는 키의 크기에 따라 정해진 회수만큼 라운드 계산을 반복 수행한다. 이 방법은 한 순간에 오직 하나의 데이터 블록만 처리되므로 feedback 모드와 non-feedback 모드 동작에 모두 사용될 수 있다[2]. 이 기본적인 구현 방법은 본 논문에서는 1 클록 1 라운드 알고리즘 또는 44 클록 알고리즘이라 부른다. 이 알고리즘의 AES 라운드 구조는 <그림 1>에 도시되어 있다. <그림 1>의 ByteSub과 ShiftRow 과정은 순서가 바뀔 수 있다. ByteSub는 각각의 바이트에 독립적으로 적용되고 ShiftRow는 각각의 바이트를 변경하지 않고 순서만 바꾸므로 이들 두 과정의 순서는 중요하지 않다. <그림 1>에 보인 AES 라운드는 많은 병렬성을 가지고 있다. 각 라운드는 16

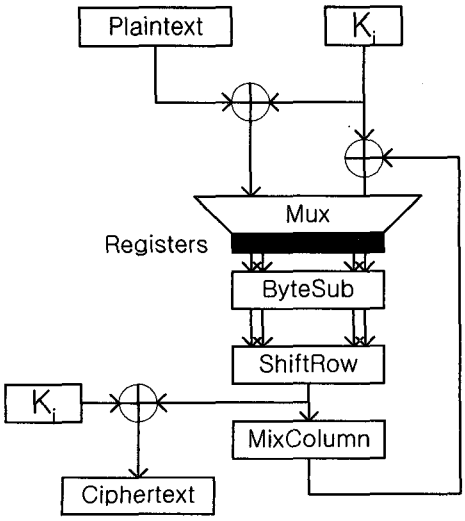


그림 1. 기본 구현 구조
Fig. 1. Basic implementation architecture.

개의 8 비트 S-box를 사용하여 ByteSub을 계산하며 4개의 32 비트 연산 MixColumn 과정이 서로 독립적으로 수행된다. 전체 128 비트를 포함하는 연산은 ShiftRow이 유일하다. 이들 16개의 SubByte와 4개의 MixColumn 연산은 자원이 충분할 경우에는 모두 동시에 수행된다.

간결한 알고리즘 구현을 위해서 오직 4개의 SubByte와 하나의 MixColumn만 사용하여 알고리즘을 구현할 수 있다. 이 방법을 사용하면 이상적으로는 구현에 필요한 자원을 1/4로 줄일 수 있다. 각 라운드 수행에는 4 클록 사이클이 사용된다. 따라서 1 클록 1 라운드 알고리즘에 비하여 약 4배 정도 성능이 저하된다. 이 변경된 라운드를 본 논문에서는 folded round라 한다. Folded round의 경우 S-box수는 1/4로 줄일 수 있으나 각 라운드에서 128 비트 데이터 블록을 transform하여야 하므로 128 비트에 대한 레지스터 공간이 필요하다.

그러나 ShiftRow 연산은 folded architecture의 경우 1 클록 1 라운드 구조보다 구현이 복잡하다. 즉, 이제까지 알려진 바로는 folded round의 구현의 경우 <그림 1>의 full round 구현과는 달리 각 AES 라운드에서 처리되는 데이터 바이트는 연산 후 상태 메모리에 직접 저장할 수 없다[3,4,5,6]. 그 이유는 새로운 바이트 값을 저장할 장소에 저장된 바이트 값들이 동일 라운드의 다음 ShiftRow 연산의 입력들로 사용되기 때문이다. 이 문제는 참고문헌 [3]에서는 간과되었다. 참고문헌 [4, 5, 6]에서는 이 문제를 고려하여 라운드 연산 결과들을 추가 메모리에 저장하였다. 특히, Chodowiec와 Gaj는 이와 같은 추가 메모리를 이용한 효율적인 folded architec-ture에 대하여 연구하였다.

본 논문에서 제안하는 folded architecture는 Chodo-wiec와 Gaj의 구현 방법을 개선한 것이므로 다음에 그들의 구현 방법에 대하여 상세히 설명한다[6].

Chodowiec와 Gaj의 구현 방법은 Xilinx FPGA에 더 적합하다. Chodowiec와 Gaj는 데이터 바이트들을 <그

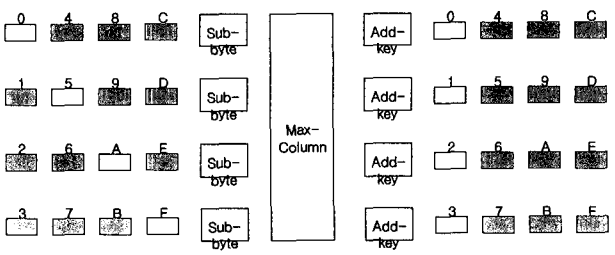


그림 2. Folded 구조
Fig. 2. The folded architecture.

림 2>에서처럼 4개의 행(row)에 배열하였다.

이 배열은 AES 사양에서 사용하는 상태(state) 표현 방식과 일치한다[2]. 이 표현 방법을 이용하여 AES 암호화 라운드는 다음과 같이 수행할 수 있다.

1. 0, 5, A, F 주소의 데이터 바이트를 읽고 ByteSub, MixColumn, AddRoundKey 연산들을 수행한 후 그 결과를 주소 0, 1, 2, 3에 기록한다. 이 과정은 <그림 2>에 흰색으로 강조되어 있다.
2. 위 과정을 주소 4, 9, E, 3의 입력에 대하여 수행하고 그 결과를 주소 4, 5, 6, 7에 기록한다.
3. 위 과정을 주소 8, D, 2, 7의 입력에 대하여 수행하고 그 결과를 주소 8, 9, A, B에 기록한다.
4. 위 과정을 주소 C, 1, 6, B의 입력에 대하여 수행하고 그 결과를 주소 C, D, E, F에 기록한다.

다음 라운드에서는 전형적인 이중 버퍼 구현에서처럼 입출력 버퍼가 교환된다. 위의 네 단계로 AES 한 라운드가 완전히 수행된다. 위의 수행 과정을 자세히 검토하면 한 행(row)에서 오직 하나의 바이트가 입력되고 출력된다. ShiftRow 연산은 각 클록마다 상태 메모리 바이트 4개를 조직적으로 읽고 씀으로써 이루어진다. Chodowiec와 Gaj는 효율적인 FPGA 구현을 위하여 dual-port RAM 기반 구현 방법과 쉬프트 레지스터 기반 구현 방법을 제안하였다. 전자는 한 FPGA 슬라이스의 두 LUT가 16x1 dual port RAM을 구현할 수 있다는 점을 이용하였다. 후자는 Xilinx FPGA의 경우 각 LUT가 임의의 위치의 출력이 가능한 16 비트 쉬프트 레지스터를 구현할 수 있다는 점을 이용한 것으로 Xilinx FPGA 구현의 경우 전자보다 두 배 이상 효율적이다.

2. Folded architecture의 개선

본 연구에서 제안하는 44 클록 AES 구현 방법은 Chodowiec와 Gaj의 folded architecture를 개선한 것으로 ShiftRow에서 double buffering을 위한 추가적인 상태 메모리를 제거하였다. 새로운 상태 메모리 접근 방법을 설명하기 위하여, 먼저 상태 메모리의 각 행(row)을 4개의 주소를 가진 조그만 메모리로 본다. 그리고 4개의 2 비트 주소 rot[0], rot[1], rot[2], 그리고 rot[3]를 각 행에 할당한다. 이들은 암호화 시작 전에 0, 1, 2, 3으로 초기화된 후 각 라운드마다 0, 1, 2, 3을 더한다. rot[i]가 2 비트 주소이므로 주소 연산은 modular 4로

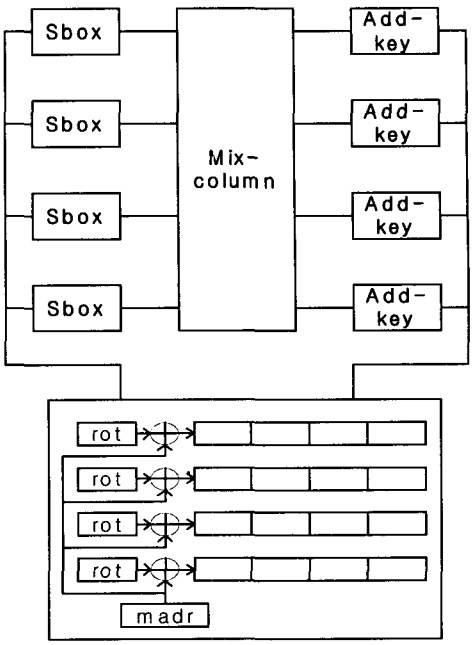


그림 3. 개선된 folded architecture
Fig. 3. The improved folded architecture.

수행된다. 각 ShiftRow 연산을 수행하는 도중 상태 메모리는 4개의 열에서 동시에 읽어내며 그 주소는 madr +rot[i]가 된다. 이때 madr은 4개의 ShiftRow 연산을 수행할 때 0에서 3까지 순차적으로 증가한다. 이를 구현한 연산 구조가 <그림 3>에 주어져 있다. 즉 ShiftRow 연산 결과 4 바이트는 원래의 입력한 장소에 되돌려 출력된다.

예를 들어 <그림 2>에서 0, 5, A, F 번지에서 읽은 입력을 ShiftRow 연산한 후 0, 1, 2, 3에 출력되지 않고 0, 5, A, F 번지에 출력된다. 한 라운드 연산이 종료된 후 rot[i]들을 0, 1, 2, 3 만큼 각각 증가시키면 연산 결과를 0, 5, A, F에 출력하였더라도 <그림 2>에서처럼 0, 1, 2, 3 번지에 저장한 것과 같은 효과를 얻을 수 있다. 이러한 간단한 주소 생성 방법으로 [4,5,6]에서 사용된 추가적인 상태 메모리를 제거하여 Chodowie와 Gaj가 제안한 folded architecture의 장점을 최대한 얻을 수 있다. 개선된 주소 생성 방법은 44 클록 AES 알고리즘 뿐만 아니라 176 클록 AES 알고리즘과 352 클록 알고리즘에서도 효과적으로 사용된다.

3. 키 스케줄

AES의 구현을 최적화하려면 암호화 부분뿐만 아니라 키 스케줄 부분의 면적 요구도 최적화하여야 한다. AES 키 스케줄링에는 두 가지 방법이 있다. On-the-fly 방법은 암호화와 키 계산을 병행하는 방법이고, 사

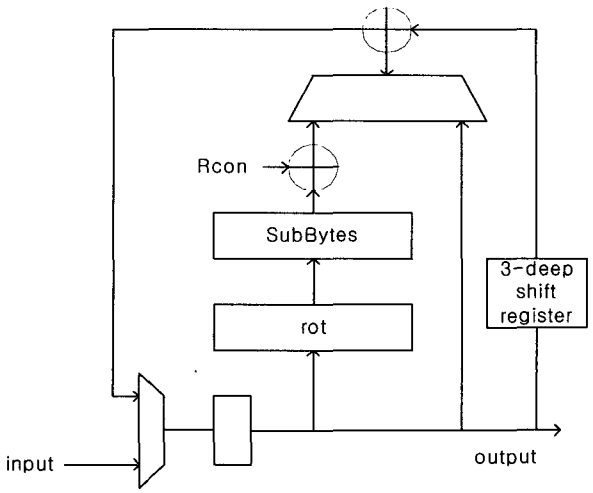


그림 4. AES 키 스케줄 알고리즘
Fig. 4. AES key scheduling algorithm.

전 계산 방법은 미리 계산된 확장된 키들을 저장했다가 암호화 수행 중에 순차적으로 접근하여 사용하는 방식이다. On-the-fly 방식으로 키를 계산하는 것은 키 스케줄 latency를 없앨 수 있으므로 key-agile 방식이라고도 하는데 데이터 블록 암호화시 키를 매번 다시 계산하여야 하기 때문에 전력 소모가 많은 장점이 있다. <그림 4>에 AES 키 스케줄링 데이터 패스가 주어져 있다.

그림에 나타나는 모든 레지스터와 조합회로 연산은 모두 4 바이트 32 비트이다. 사전 계산 방식을 채택하지 않는 경우 최소한 16 바이트의 레지스터 메모리가 소요된다. 물론 사전 계산 방식을 사용하면 확장된 키를 저장하는 많은 메모리가 소요된다.

4. Single byte architecture

Folded architecture는 요구되는 S-box 수를 20에서 8로 감소하게 한다. 제한된 자원을 사용하는 무선 임베디드 시스템에서는 folded architecture가 주어진 칩 공간에 구현될 수 없는 경우가 있다. Single byte architecture를 채택하면 구현 면적 요구를 더욱 줄일 수 있다. Single byte architecture는 암호화 중 상태 메모리에서 한 클록에 한 바이트씩 데이터를 읽어서 처리한다. 이 경우에도 앞에서 논한 바와 같이 암호화 계산 상태와 키 스케줄 계산 상태를 저장하기 위한 레지스터 메모리의 크기는 16 바이트로 고정되어 있다.

또한 S-box가 AES 구현에서 가장 비용이 많이 드는 부분이므로 single byte architecture는 사용되는 S-box 수를 감소시키는 것을 주목적으로 하여 <그림 3>의 folded architecture에서 S-box를 하나만 사용한다. 이 경우 4 바이트 워드를 처리하는 데에 4 클록이 사용된다. Key-agile 구현에서는 1 바이트 키를 얻기 위해서 S-box를 1번 사용하기 때문에 합계 2개의 S-box를 필요로 한다. 각 라운드는 16 사이클이 소요되며 전체 암호화는 176 사이클이 걸린다. 만약 S-box를 하나만 사용하면 S-box를 키 스케줄러와 데이터 암호 회로가 공유하므로 전체 암호화에 352 클록이 소요된다.

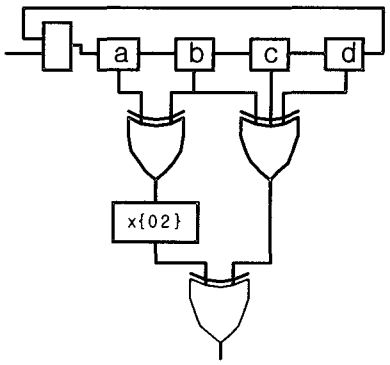


그림 5. 4 클록 MixColumn 구현
Fig. 5. Four clocks MixColumn implementation.

352 클록 사이클 구현에서는 하나의 S-box만 사용될 뿐만 아니라 MixColumn을 4 클록에 수행할 수 있다. 이 경우 각 열(column)의 4 바이트에 S-box를 이용하여 SubByte를 수행한 후 <그림 4>에 보인 바와 같이 쉬프트 레지스터에 저장한다. 4개의 MixColumn 연산은 서로 대칭성이 있다. 제안된 352 클록 암호화 알고리즘에서는 4개의 MixColumn 연산 결과를 <그림 5>에서처럼 쉬프트 레지스터를 4 클록 동안 회전하면서 얻을 수 있다. 얻어진 출력은 순차적으로 상태 메모리에 저장된다.

본 논문에서 제안하는 single byte architecture 구현은 <그림 3>의 folded architecture에서 S-box와 Add-Key 블록을 하나씩만 사용하고 MixColumn 블록은 <그림 5>의 회로로 대치하고 <그림 4>의 키 스케줄러 회로, 간단한 제어기 및 호스트 인터페이스 회로 등을 결합하여 완성된다. <그림 4>의 키 스케줄 알고리즘에서 SubBytes를 통과하지 않는 경우에는 S-box를 사용할 필요가 없다. 이 경우에는 MixColumn를 병렬로 수행하면 4 클록만에 4 바이트를 처리할 수 있다. 본 논문에서 이 알고리즘은 구현하지 않았다.

III. 실 험 결 과

실험에서는 제안된 여러 가지 AES 구현 구조의 효

표 1. AES11의 FPGA 구현 (43.49MHz)
Table 1. FPGA implementation of AES11 (43.49MHz).

function blk	# logic cells	# registers	# mem. bits
aes	769	256	32768
round	(410)	(0)	(32768)
control	18	8	0
front	762	441	0
host_int	29	6	0
key	274	128	8192
amba slave	79	15	0
Total	1881	854	40960

율성을 검증하기 위하여 ALTERA EXCALIBUR FPGA를 이용한 합성 결과를 분석한다. ALTERA EXCALIBUR FPGA는 ARM922T를 포함하고 있으나 프로그램 가능한 부분 ALTERA APEX-II 계열의 FPGA들과 동일하다. 다만 내장된 ARM922T는 외부 메모리를 사용하는 경우 4G 메모리를 접근할 수 있고 FPGA 논리회로와는 AMBA 버스를 통하여 통신할 수 있다. 먼저 Verilog HDL를 이용하여 알고리즘을 프로그램하고 ModelSim 논리회로 시뮬레이터를 이용하여 검증하였다. QUARTUS-II FPGA 설계 환경에서 제공하는 bus function 모델을 사용하여 AMBA 버스로 키와 평문을 공급하고 암호문을 얻어 소프트웨어 구현과 비교하였다. 시뮬레이션 상에서 설계가 정확하게 동작하면 QUARTUS-II의 논리회로 합성기와 설계 및 배치 도구를 이용하여 합성 결과를 얻고 키트에 다운로드하여 정확한 동작을 확인하였다. 만약 더 좋은 합성 소프트웨어를 사용하면 최대 동작 주파수를 증가시키고 사용된 FPGA 자원 요구를 감소시킬 수 있을 것이다.

FPGA를 이용한 AES 구현 연구에는 주로 Xilinx FPGA를 사용하였으나 참고문헌 [3]에서는 여러 종류의 ALTERA FPGA를 사용하여 키 스케줄 부분을 제외하고 암호화 부분만 합성한 결과를 제시하였다. 본 논문에서는 먼저 [3]에서 제시한 방법을 직접 구현한 AES11의 합성 결과와 다른 구현 합성 결과와 비교하였다.

<표 1>에는 참고문헌 [3]에서 사용한 AES의 기본 구조 구현의 FPGA 논리 합성 결과를 나타내고 있다. 기본 구조 구현의 경우 각 라운드는 1 클록에 수행되므로 11 클록에 암호화가 완료되며 20개의 S-box가 메모리 비트로 표현되어 있다. 하나의 S-box는 2048 비트로 구현된다. <표 1>에서 round는 조합 논리회로 블록으로 SubByte와 MixColumn을 구현한다. Aes 블록은

표 2. AES44의 FPGA 구현 (47.15MHz)
Table 2. FPGA implementation of AES44 (47.15MHz).

function blk	# logic cells	# registers	# mem. bits
aes	816	257	8192
round	(127)	(0)	(8192)
control	20	10	0
front	1096	569	0
host_int	28	6	0
key	196	128	8192
amba slave	79	15	0
Total	2235	985	16389

표 3. AES44-1의 FPGA 구현 (44.22MHz)
Table 3. FPGA implementation of AES44-1 (44.22MHz).

function blk	# logic cells	# registers	# mem. bits
aes	575	133	8192
round	(127)	(0)	(8192)
control	20	10	0
front	1096	569	0
host_int	27	6	0
key	199	128	8192
amba slave	79	15	0
Total	1996	861	16389

round 블록을 포함한다. 따라서 round 블록의 자원 값들은 괄호로 표시되어 있다. Front 블록과 host_int 블록은 함께 호스트 프로세서 인터페이스를 구현한다. Amba slave 블록은 AMBA 버스 Slave 설계로 QUARTUS-II의 설계 예제로부터 얻을 수 있다. Front 블록의 크기가 비교적 큰 이유는 3개의 128 비트 쉬프트 레지스터를 포함하기 때문이다. 하나는 8 비트 버스 입력을 128 비트 워드로 축적하는 역할을 하며 나머지 둘은 평문과 키를 각각 저장한다. 기본적인 아키텍처 알고리즘은 다른 제한된 자원을 사용하는 구조와 비교하기 위하여 설계되었으나 다른 구현을 디버그하는 데에도 매우 유용하게 사용되었다.

<표 2>에는 Chodowiec과 Gaj 알고리즘의 합성 결과가 나타나 있다[6]. 마지막 열을 보면, S-box의 수가 8임을 알 수 있다. S-box ROM 메모리 요구는 60% 줄었으나, 레지스터 메모리 요구는 128 비트 가량 증가하였음을 알 수 있다. Front 블록의 크기가 증가한 이유는 출력을 축적하는 128 비트 쉬프트 레지스터가 추가되었기 때문이다. Key 블록의 크기는 다소 줄어들었다. 그 이유는 키 스케줄 계산에서 32 비트 연산 일부가 제거되었기 때문이다.

<표 3>에는 개선된 folded architecture의 합성 결과

표 4. AES176의 FPGA 구현 (47.15MHz)
Table 4. FPGA implementation of AES176 (47.15MHz).

function blk	# logic cells	# registers	# mem. bits
aes	703	157	2048
round	(157)	(24)	(2048)
control	31	12	0
front	598	313	0
host_int	27	5	0
key	229	152	2048
amba slave	79	15	0
Total	1669	654	4096

표 5. AES352의 FPGA 구현 (38.17MHz)
Table 5. FPGA implementation of AES352 (38.17MHz).

function blk	# logic cells	# registers	# mem. bits
aes	718	317	2048
round	(55)	(32)	(0)
key	(230)	(152)	(0)
control	30	13	0
front	461	313	0
host_int	27	5	0
amba slave	79	15	0
Total	1317	663	2048

가 나타나 있다. Aes 줄에서 보인 바와 같이 레지스터 수가 128 가량 감소하였다. 최대 동작 주파수는 다소 감소하였다. <표 2>의 AES44 설계의 경우와 비교하면 전체 로직 셀의 수가 약 20% 정도 감소하였음을 알 수 있다. 사용된 FPGA 자원의 절반 정도가 호스트 인터페이스에서 사용되었다. 시스템 온 칩 설계에서는 버스 인터페이스 회로를 효과적으로 설계하는 것이 매우 중요하다.

<표 4>에는 176 클록 single byte architecture의 논리회로 합성 결과가 나타나 있다. 마지막 열에서 보는 바와 같이 사용된 S-box 수는 2개이다. 이 계산 구조는 암호화를 176 사이클에 완료한다. Round 블록을 포함한 aes 블록의 크기가 다소 증가하였다. 이는 이전까지 조합회로였던 round가 순차회로로 바뀌면서 버퍼 레지스터가 추가되었기 때문이다. 전체적인 logic cell의 수가 감소한 이유는 front 블록의 크기가 감소하였기 때문이다. Front 블록의 8 비트 입력 데이터를 128 비트 내부 워드로 변환하는 쉬프트 레지스터가 8 비트씩 쉬프트하는 평문 쉬프트 레지스터와 공유되었기 때문이다. 만약 마이크로프로세서와의 인터페이스가 32 비트라고 가정하면 AES176의 front 블록 크기가 AES44의 경우보다 클 것이다.

<표 5>에는 암호화 완료에 352 클록이 필요한 하나의 S-box만 사용하는 AES 구현의 합성 결과를 제시하였다. 이 구현 구조는 <그림 5>에서 제시한 MixColumn 구조를 이용하여 round 줄에서 보인 바와 같이 논리 셀 자원을 다소 절약한다. Aes줄의 로직 셀 수가 <표4>의 경우보다 많은 이유는 key 블록이 aes 블록의 내부 블록으로 이동하였기 때문이다. 다른 설계의 경우 key 블록은 key-save 구현 시를 대비하기 위하여 별도로 두었다.

Ⅳ. 결 론

본 논문에서는 무선 임베디드 시스템의 보안 응용을 위한 저비용 AES 구현에 대하여 연구하였다. 먼저 간단한 1 클록 1 라운드 구조를 구현하였다. 그 다음 Chodowiec와 Gaj가 제안한 한 라운드를 4 클록에 완료하는 알고리즘을 구현하였다. 다음에 Chodowiec와 Gaj의 알고리즘에서 상태 메모리를 줄이는 방법을 제시하였다. 그리고 구현 면적을 줄이는 single byte architecture를 제안하였다. 기본적인 single byte architecture의 경우 AES 한 라운드를 완료하는 데에는 16 클록이 소요되며 암호화를 완료하는 데에는 176 클록이 소요된다. 하나의 S-box만을 사용하는 구현은 암호화 완료에 352 클록이 사용된다. 이 경우에도 FPGA를 이용한 실험 결과 최대 허용 주파수가 38MHz이상이므로 최대 암호화 속도 13Mbps를 얻을 수 있으므로 3G 무선 통신에 충분하다.

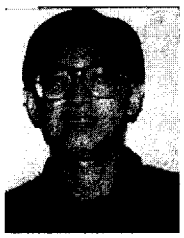
ASIC 구현의 경우 복합체를 사용하면 구현 면적을 더욱 줄일 수 있다. 마지막으로 AES 구현에서 가장 흥미로운 계산 구조적인 최적화는 Rijmen이 제한한 복합체를 사용하는 방법이다. 이를 발전시켜서 Morioka는 ASIC 설계를 위한 저면적 S-box 설계를 제안하였다[4]. 본 논문에서는 ROM을 이용하여 S-box를 구현하는 AES 구현 구조를 주로 연구하였다. ASIC 구현의 경우에는 Morioka의 복합체를 이용한 S-box 구현을 그대로 사용할 수 있다. FPGA의 구현의 경우 복합체를 사용하는 장점이 많지 않다. 차후 연구로는 Single byte architecture의 ASIC 구현에 대하여 연구할 예정이다.

참 고 문 헌

[1] S. Ravi, A. Raghunathan, and N. Potlapally, "Securing Wireless Data: System Architecture

Challenges," ISSS'02, October 2-4, Kyoto, Japan
[2] National Institute of Standards and Technology: FIPS 197: Advanced Encryption Standard, November 2001.
[3] V. Fischer and M. Drutarovsky, "Two Methods of Rijndael Implementation in Reconfigurable Hardware," Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001 Proceedings, pp. 77-92
[4] Satoh A., Morioka S., Takano K., Munetoh S.: A Compact Rijndael Hardware Architecture with the S-Box Optimization, Theory and Application of Cryptology and Information Security (ASIACRYPT 2001), Gold Coast, Australia, 2001.
[5] McMillan S. and Patterson C.: JBits Implementations of the Advanced Encryption Standard (Rijndael), Field-Programmable Logic and Applications (FPL 2001), Belfast, Northern Ireland, UK, 2001.
[6] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm," Cryptographic Hardware and Embedded Systems, Cologne, Germany, September 8-10, 2003 Proceedings, pp. 319-333
[7] I. Verbauwhede I., P. Schaumont, and H. Kuo, "Design and performance testing of a 2.29GB/s Rijndael processor", IEEE Journal of Solid-State Circuits, Volume: 38 Issue:3, March 2003.
[8] T. F. Lin, C.P. Su, C.T. Huang, C.W. Wu, "A high-throughput low-cost AES cipher chip," IEEE Asia-Pacific Conference on ASIC, 2002.
[9] U. Mayer, Oelsner C., Kohler T., "Evaluation of different rijndael implementations for high end servers," IEEE International Symposium on Circuits and Systems, 2002.
[10] Morioka S. and Satoh A., "An Optimized S-Box Circuit Architecture for Low Power AES Design," Cryptographic Hardware and Embedded Systems, San Francisco Bay, CA, 2002.
[11] A.J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volum: 9 Issue 4, August 2001.
[12] M. McLoone and McCanny J. V., "High Performance Single-Chip FPGA Rijandael Algorithm Implementations, "Cryptographic Hardware and Embedded Systems, Paris, France, 2001.
[13] V. Niemi and K. Nyberg, UMTS SECURITY, 2003, John & Wiley & Sons, Ltd.

저 자 소 개



이 동 호(정회원)
1979년 서울대학교 전자공학과 학사 졸업.
1981년 KAIST 전산학과 석사 졸업.
1992년 (미) Iowa대학 컴퓨터과학과 박사 졸업.
1981년~1992년 ETRI 선임연구원
1992년~1993년 (미) Motorolar senior CAD engineer
1993년~현재 경북대학교 전자전기컴퓨터학부 부교수
<주관심분야: 컴퓨터 구조, 프로그래밍 언어>