碩 士 學 位 論 文

블루투스 베이스밴드의 효율적인 분할 설계

濟州大學校 大學院

通信工學科

金 珍 淑

2002 年 12 月

목 차

Abstract

Bluetooth is standard to implement bidirectional short distance communication between portable units such as hand phone, PDA, notebook by cheaper without complicated wire. In some, negative views are coming out about bluetooth that is falling behind in market competition with wireless LAN. But bluetooth product launch is brisking gradually according to falling of chip set price and development of connected technology and target's extension to apply. Therefore, many inside and outside of the country companies are making efforts bluetooth module development and several application markets. Now there is no the company which develop bluetooth chip in the country and the SamSung Electronic, Hynix, GCT semi, MewTel etc. are developing it as developing product. Therefore it needs fundamental technology access and chip manufacture in nation can do that have large meaning.

In this thesis, the structure and basis function of baseband that is low level of bluetooth are analyzed via specification that is opened present to design bluetooth chip. Some blocks of baseband that is easy to load to MCU are implemented to hardware and firmware. And the implementation results are compared by size, implementation speed, number of creation and MCU load. Then suitable implementation plan is proposed. Also applying the proposed implementation method, baseband is designed to FPGA and firmware implementation block is made out to MCU. Finally, this thesis readies a basis of chip design by confirming packet transmission through the development board that integrates FPGA and MCU.

The hardware/firmware division implementation method proposed in this thesis can elicit methods to design little more efficient system by choosing suitable implementation method according to characteristic of baseband blocks.

Ⅰ. 서론

블루투스(bluetooth)란 핸드폰, PDA, 노트북 등과 같은 포터블(portable)한 장치들간의 양방향 근거리 통신을 복잡한 전선 없이 저가로 구현하기 위한 표준으로써, 근거리 무선 통신 기술과 제품을 총칭하여 일컫는다. 1994년 에릭슨의 이동통신그룹(ericsson mobile communication)이 휴대폰과 주변 기기들간에 소비 전력이 적고 가격이 싼 무선 인터페이스(radio interface)를 연구하면서 시작되었으며, PC·통신기기 업체인 에릭슨과 노키아, IBM, 인텔, 도시바 등이 중심이 되어 SIG(special interest group)를 결성하여 블루투스를 공개적인 표준으로 출범시켰다.(BRENT A. MILLER and CHATSCHIK BISDIKIAN, 2000)

최근 블루투스가 주목받는 이유는 블루투스에 키(key) 운영과 인증(authentication), 암호화(encryption) 등 자체 보안 기능을 가지고 있어 높은 보안이 필요한 통신 매체로 활용할 수 있다는 점이다. 초당 1600회의 빠른 주파수 호핑(frequency hopping) 방식을 통해 잡음이 있는 무선 주파수에서도 성능이 고르게 유지될 수 있다는 점 때문에 블루투스에 대한 연구가 활발히 진행되고 있다. 또한, ISM(industrial scientific medical) 대역인 2.4GHz의 주파수 대역을 사용함으로써 무선 면허 없이 사용할 수 있고, 사양이 공개되어 있어 로열티를 지불하지 않아도 된다는 장점을 가지고 있다.

기존에 널리 쓰이고 있는 IrDA(infrared data association) 방식과 블루투스를 비교해 볼 때 IrDA는 최대 데이터 전송 속도가 4Mbps로 1Mbps인 블루투스를 앞서지만 최대 전송 거리는 1m로 짧다. 블루투스가 전송 거리를 10m로 규정하고 있는 이유는 사무실 내에서 사용자가 휴대하고 있는 기기와 책상 등에 설치해 둔 기기간의 전송거리로 충분하다는 판단에 따른 결정이다. 또 블루투스는 음성부호화 방식인 CVSD(continuous variable slope delta modulation)를 채용해 문자 데이터의 전송은 물론이고, 음성 전송도 가능하다. 그러나 IrDA와 비교할 때 블루투스가 갖는 가장 큰 특징은 방해물이 있을 경우에도 통신이 가능하다는 점이다.

블루투스와 더불어 무선 통신의 총아로 떠오른 것이 무선랜인데, 당초 무선랜과

블루투스는 전혀 다른 영역에서 시작되었다. 블루투스는 가전기기나 이동(mobile) 단말기를 무선으로 연결해 저속의 데이터를 주고 받는 용도로, 무선랜은 기업이나 공중망 서비스와 같이 대량의 정보를 교환하는데 이용될 것으로 예상했었다. 그러나 정보통신 기술의 발달로 인해 무선랜이 홈네트워킹 시장까지 확장하고 블루투스의 영역을 침범하기 시작하면서 일부에서는 블루투스에 대해 부정적인 전망들이 나오기 시작했다. 그러나 점차 칩셋 가격이 낮아지고 블루투스 관련 기술 개발과 적용 대상이 확대됨에 따라 블루투스 제품 출시는 활발해지고 있는 상황이다. 이에 많은 국내외 업체들이 블루투스 모듈 개발 및 여러 응용 기기 시장에 주력하고 있다. 그러나 상위 프로토콜 스택(stack)이 오픈되어 있는데 반해 하드웨와 펌웨어 부분은 오픈된 소스가 없으며 그 구현 및 설계도 상위 레벨보다 어렵기 때문에 국내 업체들은 대부분 경쟁력이 떨어진다는 이유로 외국 칩을 사다 모듈을 구성하거나 응용 시장 위주로 개발하고 있다. 현재 국내에서 블루투스 칩을 개발한 회사는 없으며 삼성전자, 하이닉스, GCT semi, 뮤텔 등이 개발중인 제품으로만 연구가 진행되고 있는 실정이다. 또한, 국내 모듈 개발사로는 삼성전기, LG이노텍 등이 있는데 칩은 대부분 CSR 제품을 사용하고 있다. 따라서 보다 근본적 기술 접근이 필요하며, 국내에서의 자체 칩 제작은 큰 의미를 갖는다고 할 수 있다.

블루투스 프로토콜 스택은 실시간 소프트웨어의 복잡한 다중 단을 두고 있으며, 하위 계층인 하드웨어 부분은 전력과 비용 최적화에 초점을 맞춰 설계된 자동 상태제어와 시간 관리를 필요로 한다. 블루투스 베이스밴드에서 처리하는 일들은 그 양이 많기도 하거니와 블루투스의 핵심적인 부분이라고 할 수 있어, RF(radio frequency)단과 더불어 그 설계 기술은 블루투스 모듈 개발에 있어서 중요하다고 할 수 있다. 특히, 베이스밴드는 기본적인 하드웨어 부분을 제외한 비트열 처리, 억세스 코드(access code), 암호화, 암호화키 생성(encryption key generation), 주파수 호핑, 인증 등과 같은 많은 동작들이 EISC(extensible instruction set computing) 또는 ARM7과 같은 현대의 MCU(micro controller unit)에 잘 장착되기 때문에 MCU를 이용한 펌웨어로의 구현이 가능하다.(Jennifer Bray and Charles F Sturman, 2001)

본 논문에서는 자체 블루투스 칩을 설계하기 위하여, 현재 오픈된 스펙을 기준으로 블루투스의 하위 계층인 베이스밴드의 구조 및 기본 기능을 분석한다. 그리

고 베이스밴드의 여러 블록 중 MCU에 장착이 용이한 블록들을 하드웨어와 펌웨어로 구현하여 크기와 구현 속도, 생성 빈도수, MCU에 걸리는 부하 등을 기준으로 구현 결과를 비교함으로써 적절한 구현 방안을 제안한다. 또한, 실제 제안한 구현 방법을 적용하여, 베이스밴드를 FPGA(field programmable gate array)에 설계하고 펌웨어 구현 블록은 MCU 기반으로 설계한다. 마지막으로, MCU와 FPGA를 원보드화한 통합 보드를 통해 패킷 전송을 확인함으로써 칩 설계의 기반을 마련하고자 한다.

본 논문의 구성을 살펴보면 Ⅰ장에서는 본 논문의 동기와 목적을 제시하고, Ⅱ장에서는 베이스밴드의 구조 및 기본 기능에 대해 기술한다. Ⅲ장에서는 베이스밴드의 여러 블록 중 하드웨어적인 구현 뿐만 아니라 펌웨어로의 구현 또한 용이한 블록들의 구현 및 검증을 통해 적절한 구현 방안을 제시한다. Ⅳ장에서는 제안한 구현 방법을 적용하여 실제 두 디바이스 사이의 패킷 전송을 확인하고, 마지막으로 Ⅴ장에서 본 논문의 결론을 맺는다.

Ⅱ. 베이스밴드의 구조 및 기본 기능

본 논문에서 베이스밴드란 RF모듈을 제외한 블루투스 스택에서 가장 하부에 위치한 계층을 말한다. 예를 들면 일반적인 LAN에서 NIC(network interface card) 정도에 해당되며 가장 하부에서 블루투스 디바이스 간의 연결을 담당한다.

베이스밴드는 블루투스의 기반이 되는 부분으로 상당히 중요한 부분이며, 대부분 하드웨어적으로 구현된다. Fig. 1은 블루투스 베이스밴드와 링크 컨트롤러의 전체 구성 블록을 보여주고 있으며 베이스밴드에서 이루어지는 주요 기능은 다음과 같다.

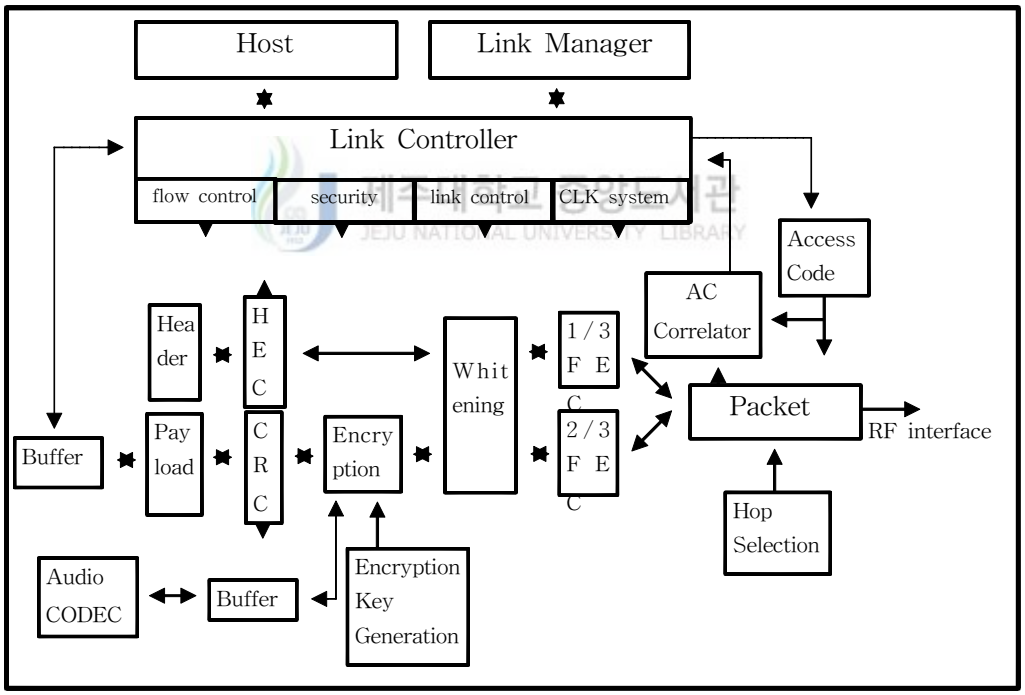


Fig. 1. Baseband/LC(link controller) architecture

1. 주파수 도약

블루투스의 주파수는 ISM 대역(2,400～2,483.5MHz)으로, 주파수 도약 방식의 스펙트럼 확산 방식을 사용한다. 전체 대역폭을 밴드폭이 1MHz인 79개의 RF 채널로 나누고 각 채널을 초당 1600회 도약을 하게 되는데, 이 채널의 정의와 도약 시퀀스 선택 등이 베이스밴드에서 이루어진다. 또한, 각 채널별로 625us의 길이를 지닌 타임 슬롯(time-slot)을 설정하여 슬롯을 통해 패킷을 교환하는 시분할이중방식(TDD:time-division-duplex)을 채택함으로써 송수신을 반복하게 된다. 이러한 시스템을 위해서는 정확한 블루투스 클럭이 요구되며, Fig. 2에서 보여지는 것처럼 클럭 구조는 28비트로 이루어져 있다. 이는 송수신기의 타이밍과 도약을 결정하는 내부적인 시스템 클럭으로서 312.5us마다 증가하는 형태이다.(Jennifer Bray and Charles F Sturman, 2001)(Bluetooth SIG, 2000)

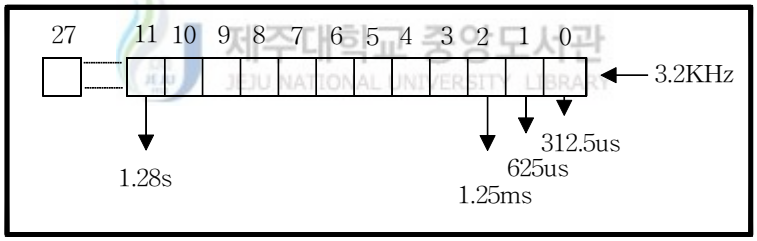


Fig. 2. Blutooth clock

도약 시퀀스의 종류에는 조회 시퀀스(inquiry sequence), 조회 응답 시퀀스(inquiry response sequence), 호출 도약 시퀀스(page hopping sequence), 호출 응답 시퀀스(page response sequence), 채널 도약 시퀀스(channel hopping sequence) 등이 있으며, 디바이스의 각 상태에 따라 사용되고 입력 제어워드가 다르다.

Fig. 3은 79 도약 시스템에서의 홉 선택 커널 블록이다. 블루투스 79 도약 시스템에서는 79 홉을 여러 개의 32 홉 세그먼트들로 나누어 세그먼트를 교환해 가며

사용하도록 되어 있다. 블록 내부의 입력인 X는 32 홉 세그먼트에서의 위상을 결정하고, Y1과 Y2는 'master-to-slave'인지, 'slave-to-master'인지의 전송 방향을 선택하게 된다. 또한 A~D는 세그먼트내의 순서를, E와 F는 도약 주파수를 결정해 주며 커널은 모든 짝수 도약 주파수 다음에 홀수 도약 주파수를 작성하게 된다. 첫 번째 가산기(ADD)는 세그먼트 내의 위상만을 바꾸는데 이 출력은 배타적 논리합(XOR)에서 B와 연산을 수행하게 된다. 다음 단계의 순열연산(PERM5)은 Fig. 4와 같고 Table 1은 제어 신호 P('1'일 때)에 따른 Z의 동작을 보여준다. 여기서 P_{0-8}은 D_{0-8}, P_{9-13}은 C_{0-4} ⊕ Y1 에 해당되며 Z는 배타적 논리합의 출력이다.

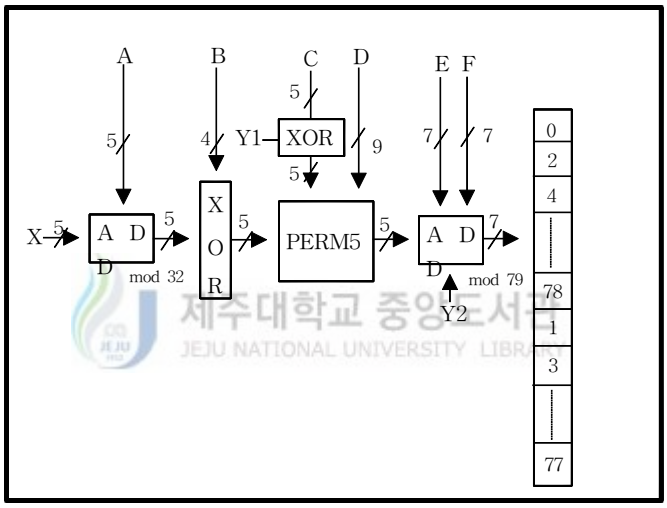


Fig. 3. Block diagram of hop selection kernel

Table 1. Control of the butterflies.

Control signal	Butterfly	Control signal	Butterfly
P_0	$\{Z_0, Z_1\}$	P_8	$\{Z_1, Z_4\}$
P_1	$\{Z_2, Z_3\}$	P_9	$\{Z_0, Z_3\}$
P_2	$\{Z_1, Z_2\}$	P_{10}	$\{Z_2, Z_4\}$
P_3	$\{Z_3, Z_4\}$	P_{11}	$\{Z_1, Z_3\}$
P_4	$\{Z_0, Z_4\}$	P_{12}	$\{Z_0, Z_3\}$
P_5	$\{Z_1, Z_3\}$	P_{13}	$\{Z_1, Z_2\}$
P_6	$\{Z_0, Z_4\}$		
P_7	$\{Z_2, Z_4\}$		

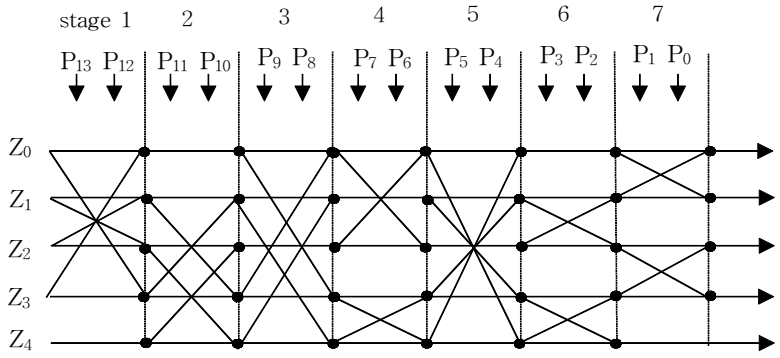


Fig. 4. Permutation operation

Table 2. Control of hop selection kernel

	Page scan/ Inquiry scan	Page/Inquiry	Page response (master/slave) and Inquiry response	Connection state
X	$CLKN_{16-12}$	$X_{p_{4-0}}/X_{i_{4-0}}$	$X_{prm_{4-0}}/X_{prs_{4-0}}$	CLK_{6-2}
Y1	0	$CLKE_1/CLKN_1$	$CLKE_1/CLKN_1$	CLK_1
Y2	0	$32 \times CLKE_1/32 \times CLKN_1$	$32 \times CLKE_1/32 \times CLKN_1$	$32 \times CLK_1$
A	A_{27-23}	A_{27-23}	A_{27-23}	$A_{27-23} \oplus CLK_{25-21}$
B	A_{22-19}	A_{22-19}	A_{22-19}	A_{22-19}
C	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0} \oplus CLK_{20-16}$
D	A_{18-10}	A_{18-10}	A_{18-10}	$A_{18-10} \oplus CLK_{15-7}$
E	$A_{13,11,9,7,5}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$
F	0	0	0	$16 \times CLK_{27-7}$ mod79

두 번째 가산기(ADD)는 32 홉 세그먼트를 각기 다른 도약 주파수들에 맵핑(mapping)시키는 역할을 하며 이 출력은 79 나머지 연산을 수행하여 79 레지스터의 뱅크에 주소로서 지정된다. 레지스터에는 0에서 78의 도약 주파수에 해당하는 합성 코드 워드를 싣게 되며 상위에는 짝수 도약 주파수를, 하위에는 홀수 도약 주파수를 구성한다.

Table 2는 선택 커널의 제어워드를 나타낸 것이고 각 상태에 따른 입력 X는 식

(1)～식 (6)에 보여준다. 이때 사용되는 클럭의 형태는 다음과 같다.

① CLK_{27-0} : 현재 피코넷의 마스터 클럭

② $CLKN_{27-0}$: 기기의 고유 클럭

③ $CLKE_{27-0}$: 호출된 기기의 고유 클럭에 옵셋을 더한 클럭

(1) 조회 상태

조회 상태(inquiry substate)에서의 X-입력은 특정 기기의 주소가 아직 지정된 상태가 아니기 때문에 조회 측의 CLKN이 사용되고, 이는 식 (1)과 같이 표현되며 이 때 k_{offset} 은 식 (2)와 같다. 주소 입력으로는 GIAC LAP와 DCI(default check initialization =0x00)의 4비트 LSB가 사용된다.

$$X_i = [\ CLKN_{16-12} + k_{offset} + (\ CLKN_{4-2,0} - CLKN_{16-12}) \bmod 16] \bmod 32 \quad (1)$$

$$k_{offset} = \begin{cases} 24 & A-train \\ 8 & B-train \end{cases} \quad (2)$$

(2) 조회 응답

조회 응답(inquiry response) 상태에서의 X-입력은 식 (3)과 같다. 여기서 $CLKN^*_{16-12}$은 CLKN과 마스터의 CLKE의 차이로 인한 링크 손실의 가능성을 피하기 위해 현재의 슬래이브 값으로 고정한 것이고, N은 응답 패킷인 FHS 패킷이 전송된 후부터 증가한다. 조회 상태와 마찬가지로 주소 입력은 GIAC LAP와 DCI의 4비트 LSB가 사용된다.

$$X_{ir} = [\quad CLKN^*_{16-12} + N] \bmod 32 \quad (3)$$

(3) 호출 상태

79 도약 시스템인 경우 송신기는 16 슬롯이나 10ms에서 16개의 다른 도약 주파수를 포함할 수가 있는데, 이렇게 호출 도약 시퀀스(paging hopping sequence)를 16개의 주파수를 갖는 두 개의 그룹 A-train과 B-train으로 나누어 사용하게 된다. 호출 상태(page substate)에서 호출 기기는 A-train(f(k-8), ... f(k), ,... f(k+7))을 사용함으로써 전송을 시작한다. 식 (2)에서 k_{offset}을 8로 설정함으로써 B-train (f(k-16), f(k-15), ... f(k-9), f(k+8), ... f(k+15))을 사용할 수 있는데, 이러한 32개의 사용 가능한 호출 주파수는 1.28s의 지속 기간을 갖는다. A-train과 B-train은 현재의 k_{offset}에 16을 더함으로써 교대로 사용할 수 있으며, 호출 상태에서의 X입력은 식 (4)와 같고 입력 주소 값으로는 대기중인 기기의 디바이스 주소가 사용된다.

$$X_p = [\ CLKE_{16-12} + K_{offset} + (\ CLKE_{4-2,0} - CLKE_{16-12}) \bmod 16] \bmod 32 \quad (4)$$

(4) 호출 응답

가) 슬래이브 응답

$CLKN^*_{16-12}$은 조회 응답 과정에서 설명한 의미와 같으며, N은 0에서 시작해서 $CLKN_1$이 0이 되는 순간마다 1씩 증가하게 된다. X-입력은 식 (5)와 같고 호출 상태와 마찬가지로 대기중인 기기의 디바이스 주소가 입력 주소로 사용된다.

$$X_{prs} = [\quad CLKN^*_{16-12} + N] \bmod 32 \quad (5)$$

나) 마스터 응답

마스터 응답은 클럭 값과 현재의 k_{offset} 값이 고정되어야하고, N은 1에서 시작

해서 $CLKE_1$이 0이 되는 순간마다 1씩 증가한다. X-입력은 식 (6)과 같다.

$$X_{prm} = [\ CLKE^*_{16-12} + k_{offset}{}^* + (\ CLKE^*_{4-2,0} - CLKE^*_{16-12}) \bmod 16 + N] \bmod 32 \quad (6)$$

(5) 연결 상태

연결 상태(connection state)에서는 마스터의 디바이스 주소와 클럭을 사용한다. 마스터는 짝수(CLK_{1-0} =00) 슬롯에서, 슬래이브는 홀수(CLK_{1-0} =10) 슬롯에서 전송을 시작한다.

2. 패킷의 정의 및 생성

베이스밴드에서는 표준 패킷을 정의하고 생성하는 일을 하는데 이것은 베이스밴드의 가장 일반적인 기능이라고 할 수 있다. 표준 패킷은 억세스 코드, 헤더(header), 페이로드(payload)로 구성되어 있고, 역할 및 링크의 종류에 따라 링크 컨트롤 패킷, ACL(asynchronous connection less) 패킷, SCO(synchronous connection oriented) 패킷으로 나누어진다. 또한, 이 3가지 패킷은 페이로드 길이, FEC(forward error correction) 방식, CRC(cyclic redundancy check) 여부 등에 따라 더 세분화된 패킷으로 나누어진다. 각 패킷의 종류에 따라 전송 속도도 달라지는데 가장 최고의 속도를 낼 수 있는 패킷은 DH5 패킷으로, 비대칭 모드로 최고 723.3Kbps, 대칭 모드로는 최고 433.9Kbps까지 낼 수 있다. Fig. 5는 베이스밴드에서 생성해 내고 있는 일반적인 패킷의 형태이다.(Jennifer Bray and Charles F Sturman, 2001)(Bluetooth SIG, 2000)

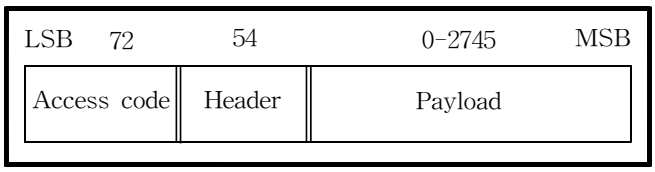


Fig. 5. Packet format

(1) 억세스 코드

억세스 코드는 모든 패킷의 시작을 알리는 접근 코드로, 같은 피코넷 안에서 전송되는 모든 패킷들은 같은 값을 사용하게 된다. 예를 들어 슬래이브는 저장된 마스터 억세스 코드와 수신한 억세스 코드를 정합시킴으로써 패킷의 존재를 검출하게 되고, 이를 통해 다른 피코넷으로부터의 데이터 전송을 막을 수가 있다. 그 구조는 프리엠블(preamble), 동기워드(sync word), 트레일러(trailer)로 구성되며, 뒤에 헤더가 올 때는 72비트의 길이를 갖고, 그렇지 않은 경우에는 트레일러를 제외한 68비트를 갖게 된다. 억세스 코드는 초기 연결 설정 과정인 조회와 호출 과정에서도 사용되는데 이때는 신호 메세지(signaling message)로 사용되는 것이며, 헤더와 페이로드는 없다. Fig. 6은 사용 목적에 따라 억세스 코드를 CAC(channel access code), DAC(device access code), IAC(inquiry access code)로 나눈 것이며, Fig. 7은 억세스 코드 동기워드의 생성 과정을 보여주고 있다.

	Preamble(4bit)	Sync Word (64bit)	Trailer(4bit)
	0101	0.... 1	0101
	1010	1.... 0	1010
CAC(72bit)	Preamble(4bit)	master's LAP	Trailer(4bit)
DAC(64bit)	Preamble(4bit)	slave's LAP	
GIAC(64bit)	Preamble(4bit)	reserved[dereved by 0x9E8B33]	
DIAC(64bit)	Preamble(4bit)	reserved	

Fig. 6. The kind of access code

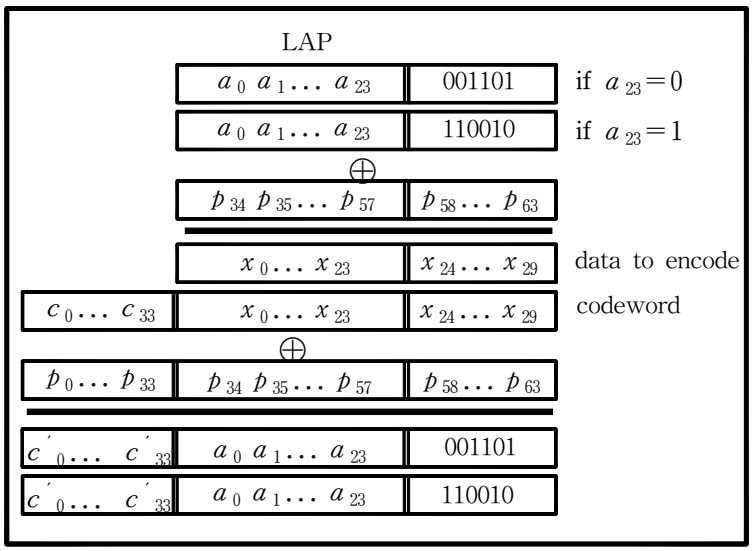


Fig. 7. Construction of the sync word

(2) 헤더

패킷의 헤더는 링크 제어 프로토콜과 링크의 중요한 동작에 관계되는 정보를 담고 있는 부분으로 HEC(header error check)를 포함하여 18비트이고, 이것은 1/3 FEC에 의해서 암호화되어 54비트를 형성한다

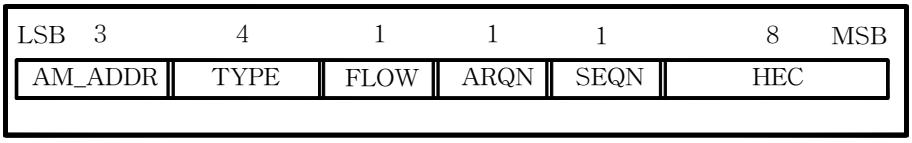


Fig. 8. Header format

헤더 포맷의 각 필드 중, 3비트 AM_ADDR은 멤버 주소(member address)로 피코넷에 참여하는 활성 멤버들을 식별하기 위해 각 슬래이브에게 임시적으로 할당해 주는 주소이다. 4비트의 TYPE 필드는 16개의 패킷 타입 중 현재 사용되는 패

킷 형태를 표시하며, 패킷이 몇 개의 타임 슬롯(time slot)을 차지하는지도 나타내준다. FLOW는 ACL 링크의 패킷 흐름 제어에 사용되며, ARQN은 CRC를 갖고 있는 정보 데이터의 성공적인 전송 여부를 알려준다. SEQN은 새로운 데이터가 전송될 때마다 인버트 되는 부분으로, 전송이 성공하면 송신측이 ACK를 받게 되고 SEQN 비트가 역전되어 새로운 정보를 전송하게 된다. 마지막으로 8비트의 HEC가 오게 되는데 이는 Fig. 9의 생성 회로에 의해 생성되며, 이때 레지스터 초기 값으로는 마스터의 8비트 UAP(upper address part)가 사용된다.

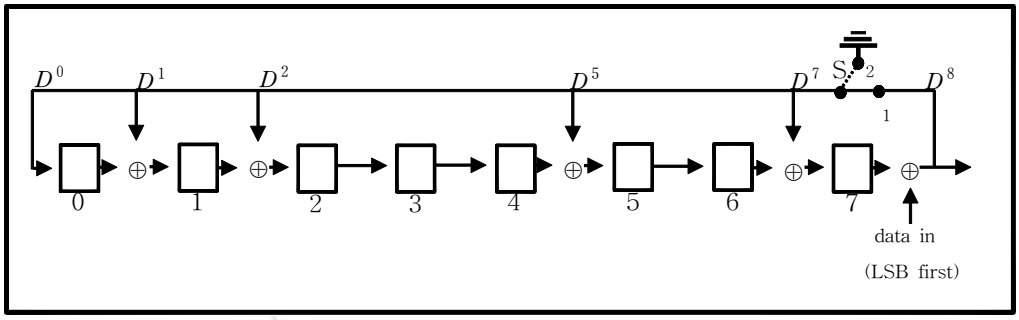


Fig. 9. the LFSR circuit generating the HEC

(3) 페이로드

페이로드는 실제 전송하고자 하는 정보 부분으로서 상위 프로토콜인 L2CAP (logical link control and adaptation protocol)나 LM(link manager)으로부터의 메세지 정보나 실제 데이터를 포함한다. Fig. 10과 같이 모든 ACL 패킷의 페이로드 필드는 페이로드 헤더, 페이로드 데이터 자체, CRC 필드의 세 부분으로 나누어지며 페이로드의 에러 검출이 Fig. 11에서 보여지는 것처럼 16비트 CRC 회로를 통해 이루어진다. HEC 생성 회로와 마찬가지로 CRC 회로의 레지스터 초기 값으로 마스터의 8비트 UAP가 사용된다.

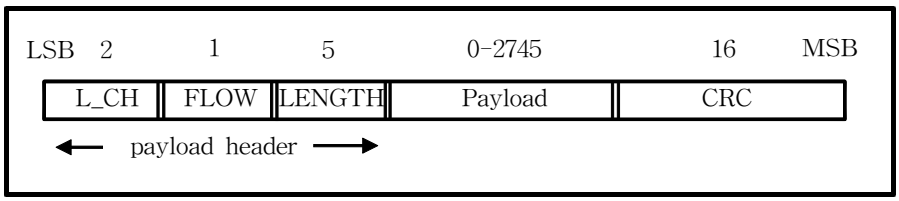


Fig. 10. Payload format

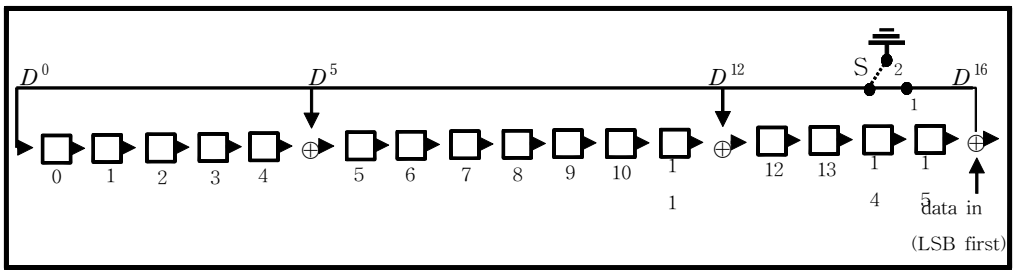


Fig. 11. the LFSR circuit generating the CRC

3. 비트 랜덤화 및 FEC

(1) 비트 랜덤화

비트 랜덤화(whitening)는 데이터를 랜덤화하기 위하여 Fig. 12를 통해 생성된 랜덤 비트 열을 데이터 열과 혼합하는 과정이다. 이것은 0이나 1이 긴 시퀀스의 가능성을 상당히 줄여주고 DC 바이어스를 제거하는 기능을 한다. 데이터의 전송 전에 회로의 쉬프트 레지스터는 마스터 클럭의 일부인 CLK_{6-1}과 '1'의 값을 갖는 하나의 MSB에 의해 초기화 되도록 되어 있다. 그러나 조회 또는 호출 스캔인 경우는 레지스터의 초기 값이 5비트의 클럭과 '1'의 값을 갖는 두 개의 MSB로 확장된다.

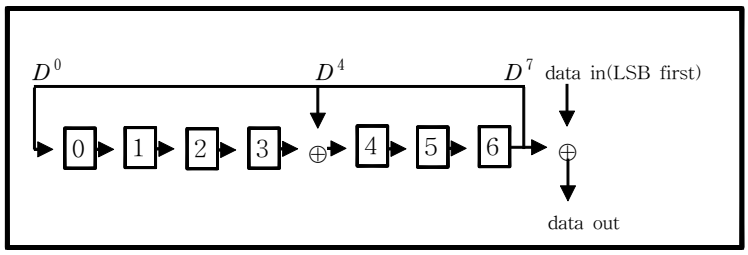


Fig. 12. Data whitening LFSR

(2) FEC

패킷과 관련되어 베이스밴드에서는 에러 정정(error correction) 및 에러 검출(error checking)의 기능도 담당한다. 블루투스에서 사용되는 에러 정정 방법은 1/3 rate FEC, 2/3 rate FEC, ARQ로 분류되며, 그 사용은 패킷의 종류에 따라 다르다. 또 에러가 발생한 패킷은 재전송을 하는데 이것은 ACL 링크에서만 가능하며, SCO 링크에서는 재전송이 이루어지지 않는다. 에러 검출은 표준 패킷의 억세스 코드, 헤더, 페이로드 각각에 대해 이루어진다.

가장 강력한 1/3 rate FEC는 단지 각 비트를 세 번 반복하여 수신측에 대한 다수결 기능을 실행하여 복호화한다. 이것은 모든 3비트 입력 중에 2비트를 검출하고 1비트를 정정하게 된다. 블루투스 패킷의 헤더는 그 속에 포함된 링크 정보가 패킷의 가장 중요한 부분이므로 1/3 rate FEC 방법을 사용하고 있다.

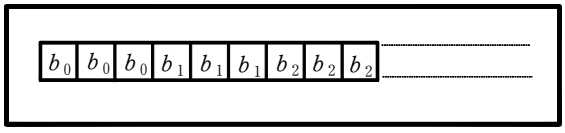


Fig. 13. Bit-repetition encoding scheme

2/3 rate FEC는 LFSR(linear feedback shift register)로 구현된 (15,10)축약 해밍(hamming)코드이다. 따라서 모든 10비트 입력에 대해서 15비트가 출력된다. 이

방법은 10비트 입력 중에서 2비트를 검출하고 1비트를 정정하게 되는데, 블루투스에서는 사용자 정보를 부호화 하는데 사용된다.

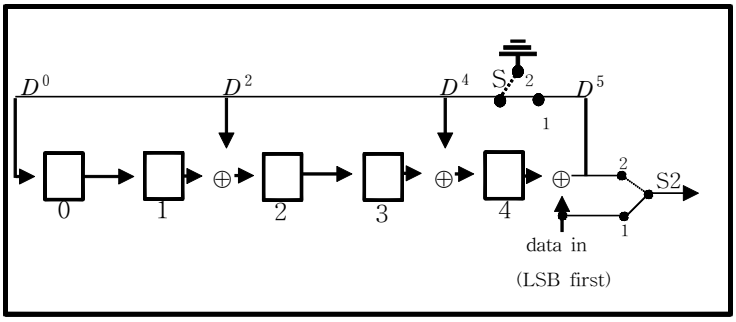


Fig. 14. LFSR generating the (15,10) shortened hamming code

4. 보안

케이블에 기반 하여 기본적인 보안이 제공되는 유선통신과는 다르게 블루투스는 무선통신이므로 손쉽게 도청이 가능하고, 도청시 인지가 불가능하다는 단점이 있다. 따라서 다른 곳으로부터의 침입을 방지하는 것이 중요한 문제라고 할 수 있다. 원하는 디바이스와의 연결만을 보장하고, 전송하고자 하는 정보를 보호하기 위해 블루투스에서는 암호화와 인증을 제공하고 있다.(Bluetooth SIG, 2000)

(1) 키 운영

블루투스에서 디바이스간의 인증과 암호화를 위해 사용되는 링크 키에는 다양한 유형이 있으며, 각 키는 다른 방법으로 생성된다. 링크키는 SAFER+ 알고리즘의 “E” 구현을 통해 생성된 128비트의 값으로서, 반영구적으로 쓰이거나 일시적으로 사용된다.

링크키에는 단위키(unit key)와 조합키(combination key), 그리고 마스터키(master key), 초기키(initialization key)가 있다. 단위키는 반영구적인 키로서 비휘발성 메모리에 저장되며, 공장 출하 시 처음 설정되고 특별한 이유가 없을 경우 거의 변하지 않는다. 조합키는 블루투스 장치간의 단위키를 서로 교환하고 두 개의 단위키를 조합하여 링크키로 사용하는 경우이다. 이 키는 두 블루투스 장치간에 더 많은 보안을 요구할 때 사용한다. 단위키와 조합키는 E_{21} 알고리즘을 이용하여 생성되며, 식 (7)과 같다. 이때, 디바이스 주소와 RAND(random number)의 적절한 조합을 통해 입력 값을 생성한다.

$$E_{21}:(RAND, address)\rightarrow Ar'(X, Y) \tag{7a}$$

$$X = RAND[0\ldots14]\bigcup(RAND[15]\bigoplus 6) \tag{7b}$$

$$Y = \bigcup address[i(\mathrm{mod}\,6)] \tag{7c}$$

마스터키는 다중 연결 설정시 사용하며 현재의 링크키를 대체하는 임시키이고, 초기키는 128비트로 하나의 세션에 대하여 사용되는 링크키로서 유닛이 초기화될 때 사용된다. 이 키들은 사용자의 PIN(personal identification number) 입력 값을 이용하여 생성되는데 보통 4자리 숫자로 입력받게 되며, 식 (8)에 표현한 것과 같이 E_{22} 알고리즘을 사용한다. 이외에도 슬래이브의 디바이스 주소, 마스터에 의해 생성되는 128비트의 RAND가 입력 값으로 들어가고, 128비트의 출력 키 값이 나오게 된다.

$$E_{22}:(\Pi N', RAND, L')\rightarrow Ar'(X, Y) \tag{8a}$$

$$X = \bigcup \Pi N'[i(\mathrm{mod}\,L')] \tag{8b}$$

$$Y = RAND[0\ldots14]\bigcup(RAND[15]\bigoplus L') \tag{8c}$$

마지막으로 암호화 단계에서 사용되는 암호화키(encryption key)가 있는데 이는 현재의 링크키로부터 유도된다. 암호화키를 생성하기 위해서는 E_3 알고리즘이 사

용되며 다음과 같이 나타낼 수 있다.

$$E_3:(K, RAND, COF) \rightarrow Hash(K, RAND, COF, 12) \quad (9)$$

이때, 입력 값은 현재의 링크키와 96비트의 COF(ciphering offset number) 그리고 128비트의 RAND를 통해 만들어지며, COF는 인증 과정에서 생성되는 ACO (authenticated ciphering offset)에 기초를 두고 있다.

이렇게 여러 가지 키를 사용하는 이유는 각 디바이스마다 메모리 저장 공간의 많고 적음의 여부와 PIN을 입력받아 사용할 것인지 아니면 고정 PIN을 사용할 것인지, 그리고 일대일 통신인지 멀티캐스팅인지에 따라 유동적으로 키를 사용하기 때문이다.

(2) 인증

본질적으로 블루투스 인증은 요구/응답(challenge-response) 구조로 되어 있다. 여기서 상대편이 공유되는 비밀키를 알고 있는지 체크하기 위해 양방향 프로토콜(2-move protocol)이 사용되는데, 기본적으로 이 프로토콜은 두 디바이스가 같은 키를 갖고 있는지, 그리고 그들이 인증 과정을 성공적으로 수행했는지를 체크하게 된다. 이 인증 과정에서 생성된 ACO 값은 양쪽 디바이스에 저장되고 나중에 암호화키를 만드는데 사용된다. Fig. 15는 두 디바이스 사이에서의 인증 과정을 보여주고 있다.

인증을 위한 함수 E1은 Fig. 16과 같이 SAFER+라고 불리워지는 Ar과 Ar의 변형 형태인 Ar′으로 구성되어 있다. E1의 입력으로는 링크키, RAND 그리고 디바이스 주소 값이 들어가고 최종적으로 SRES(signed response)와 ACO가 만들어지게 된다.

Ar로 표현되어지는 SAFER+는 1998년을 기점으로 표준 기한이 만료된 DES (the data encryption standard)의 대안으로 개발된 알고리즘으로서, AES (advanced encryption standard)의 후보 알고리즘 중 하나이다. 이 SAFER+ 알고

리즘 구조를 Fig. 17에 나타내었다.(James L. Massey and Gurgan H. Khachatrian and Dr. Melsik K. Kuregian, 1998)

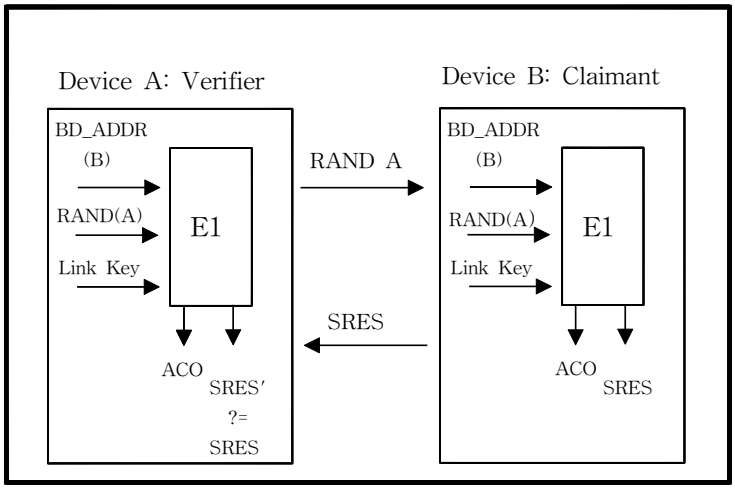


Fig. 15. Description of the authentication process

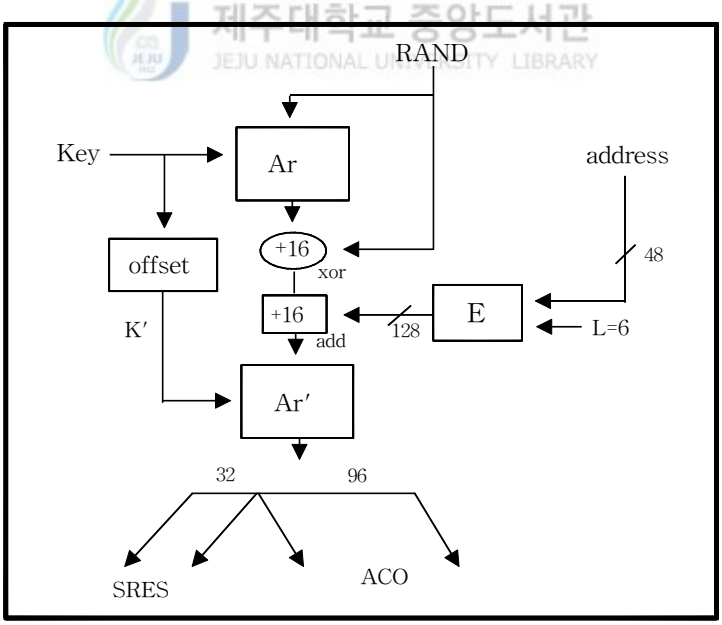


Fig. 16. Flow of data for the computation of E1

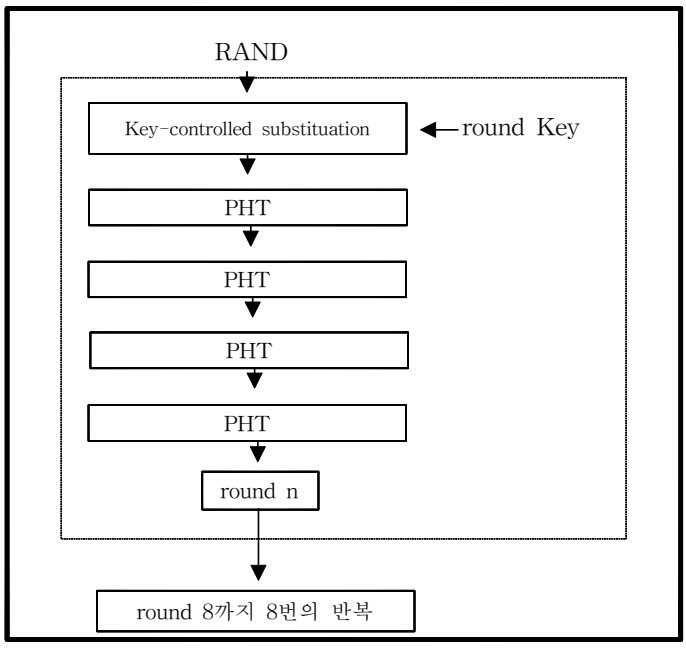


Fig. 17. SAFER+ algorithm

(3) 암호화

블루투스에서의 사용자 정보는 패킷의 페이로드 부분을 암호화함으로서 보호될 수 있다. 그러나 이때 억세스 코드와 패킷의 헤더는 암호화되지 않는다. Fig. 18에 나타난 것처럼 블루투스 정보의 암호화는 모든 정보를 재 동기화하는 E_0라 불리는 열 암호화기(stream cipher)에 의해 수행된다. 이 암호화 시스템은 세 부분으로 구성되는데, 첫 단계에서는 초기화가 이루어지고, 두 번째 단계(key stream generator)에서 생성된 키 열 비트(key stream bit)가 세 번째 단계에서 전송하고자 하는 데이터와 XOR되어 암호화된다. E_0 알고리즘에는 마스터의 블루투스 주소(master's bluetooth address), 마스터 클럭의 26비트(CLK_{26-1}), 암호화 키(K_C) 그리고 마스터에 의해 생성되어 피코넷의 슬래이브들에게 전달되는 RAND가 초기 값으로 입력되게 된다.

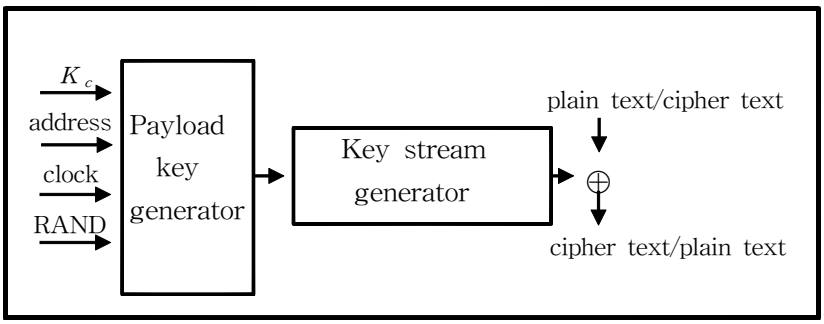


Fig. 18. Stream ciphering for bluetooth with E_0

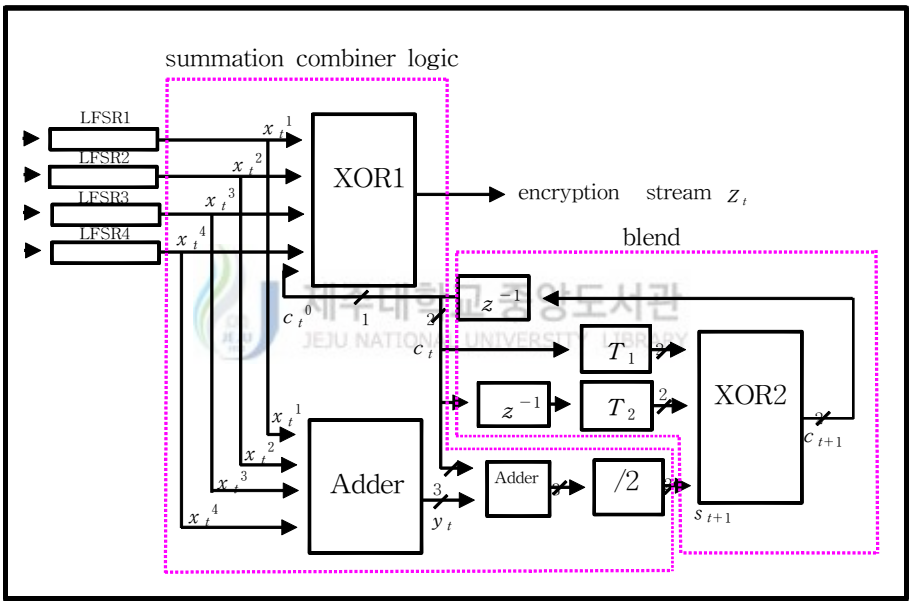


Fig. 19. Concept of the encryption engine

암호화 알고리즘의 내부 구조는 Fig. 19와 같고, LFSR과 각 단계에서의 구현 다항식은 식 (10)~식 (19)에 나타내었다. 각각의 LFSR에서 사용된 레지스터 길이는 L_1=25비트, L_2=31비트, L_3=33비트, L_4=39비트이며 사용된 전체 레지스터의 길이는 128비트이다.

Fig. 19에서 XOR1과 가산기의 입력 $LFSR_i$의 다항식은 다음과 같다.

$$LFSR_1 = t^{25} + t^{20} + t^{12} + t^8 + 1 \tag{10}$$

$$LFSR_2 = t^{31} + t^{24} + t^{16} + t^{12} + 1 \tag{11}$$

$$LFSR_3 = t^{33} + t^{28} + t^{24} + t^4 + 1 \tag{12}$$

$$LFSR_4 = t^{39} + t^{36} + t^{28} + t^4 + 1 \tag{13}$$

Fig. 19에서 가산기의 출력 y_t는 다음과 같다.

$$y_t = \sum_{i=1}^{4} x_t{}^i \tag{14}$$

식 (14)에서 입력 $x_t{}^i$는 $LFSR_t$의 출력 t^{th}를 나타낸다. 나머지 연산의 출력이면서 XOR2의 입력 s_{t+1}은 식 (15)와 같이 나타낸다.

$$s_{t+1} = (s_{t+1}{}^1, s_{t+1}{}^0) = \left[\frac{y_t + c_t}{2}\right] \in [0,1,2,3] \tag{15}$$

$$c_{t+1} = (c_{t+1}{}^1, c_{t+1}{}^0) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}] \tag{16}$$

식 (16)은 XOR2의 출력 c_{t+1}를 나타내고, Fig. 19에서 매핑 T_1과 T_2는 식 (17), (18)과 같다.

$$T_1 : (x_1, x_0) \mapsto (x_1, x_0) \tag{17}$$

$$T_2 : (x_1, x_0) \mapsto (x_0, x_1 \oplus x_0) \tag{18}$$

암호화기의 최종 출력 z_t는 다음과 같으며, 전송하고자 하는 데이터와 한 비트씩 XOR되어 데이터의 암호화가 이루어진다.

$$z_t = x_t{}^1 \oplus x_t{}^2 \oplus x_t{}^3 \oplus x_t{}^4 \oplus c_t{}^0 \tag{19}$$

Ⅲ. 베이스밴드 블록의 하드웨어/펌웨어 구현 및 검증

베이스밴드는 하드웨어로 구현하는 것이 일반적이다. 그러나 위에서도 언급하였듯이 블루투스 베이스밴드의 기본적인 하드웨어 부분을 제외한 비트 열 처리, 억세스 코드, 암호화, 암호화키 생성, 주파수 도약, 인증 등과 같은 많은 부분들이 현재 사용하고 있는 MCU에 잘 장착되기 때문에 MCU를 이용한 펌웨어로의 구현이 가능하다. 본 논문에서는 임베디드 시스템에 적합한 EISC 코어를 사용하여 펌웨어를 구현하고자 한다.

일반적으로, 하드웨어적인 구현의 장점은 MCU에 대한 부담을 크게 완화시켜 준다는 점이지만 설계 변경이 어렵고 유연성이 떨어진다는 단점이 있다. 반면, 펌웨어로의 구현은 MCU에 부하가 많이 걸린다는 단점을 갖고 있으나 설계 변경의 용이함과 유연성, 복잡하지 않은 하드웨어를 제공한다는 장점이 있다.

본 장에서는 최적의 베이스밴드를 설계하기 위해 베이스밴드의 여러 블록들을 하드웨어와 펌웨어로 실현하여 그 효율성을 비교하고 이를 통해 적절한 구현 방안을 제시하고자 한다. 하드웨어는 Verilog HDL로 Workstation 상에서 설계하여 FPGA(Altera)로 테스트하였고 펌웨어는 MCU에 장착이 용이한 C 프로그램으로 실현하였다.

1. 테스트 환경

Fig. 20과 Fig. 21은 베이스밴드의 하드웨어와 펌웨어 설계 테스트를 위한 테스트 환경을 보여주고 있다. Fig. 20은 ALTERA사의 EPF10K50RC240-3 디바이스(device)를 이용하여 여러 가지 디지털 로직을 설계할 수 있도록 제작한 트레이닝 킷(training kit)이다. 이 보드의 입력은 4개의 딥스위치로 32비트를 사용할 수 있게 구성되어 있고, 출력은 32개의 LED로 구성되어 있다. 그리고 디바이스의 모든 핀을 커넥터로 연결하여 외부 회로와의 인터페이스가 가능하도록 하였으며, 각 출력 모드는 딥스위치를 이용하여 사용 여부 선택이 가능하다. 이 보드에서 사용되는 디바이스는 게이트 어레이(gate array)를

이용한 디지털 로직을 하나의 칩으로 설계하기 위한 용도로 사용되어지며, 2,880LEs를 이용하여 50,000Gates를 설계할 수 있도록 구성되어 있다. 또한 이 디바이스는 SRAM 타입으로, 전원이 인가되지 않은 상태에서는 데이터가 손실되는 단점을 가지고 있다. 본 논문에서는 이러한 단점을 보완하기 위해 전용롬을 장착하였으며 전원이 인가되면 디바이스가 구성(configuration)되어 단독 모듈로써 동작이 가능하게 하였다 여기서, 로직 설계는 ALTERA사의 MAX+PLUSⅡ를 사용하였다.

Fig. 20. Environment for hardware test(baseband test board)

Fig. 21은 펌웨어 설계 테스트를 위한 EISC 코어를 보여주고 있다. EISC는 국내 비메모리 반도체 업체인 에이디칩스가 자체 개발한 마이크로프로세서 기술로서, 출현 빈도가 높으면서 짧은 길이를 갖는 16비트 고정 길이 명령어(16 bit fixed length instruction)로 구성되어 있다. EISC의 특징으로는 필요한 명령어 길이(operand length)만큼 LERI(load to extension register and set E) 구조를 사용하여 명령어를 확장하는 방식을 사용하므로 프로그램 사이즈가 작아진다는 점이다. 즉 코드 밀도(code density)가 높은 장점이 있다. 또한, 한 가지 op-code에 한 개의 명령(instruction)만을 가지고 있으므로 명령의 수가 적다. 이에 명령어 길이에 따라 여러 개의 명령을 만들 필

요가 없으므로, 명령의 수가 기존의 마이크로프로세서보다 적은 특징을 갖는다. 따라서 하드웨어가 간단하고 성능이 우수한 특징을 갖는다. 결과적으로 EISC는 전력 소모가 적은 구조라고 할 수 있고 블루투스 응용 제품인 임베디드 시스템에 적합하다고 할 수 있다. 본 논문에서는 링크 컨트롤러 뿐만 아니라 베이스밴드의 여러 블록 중 펌웨어 구현이 용이한 블록들을 EISC 코어에 맞게 구현하여 하드웨어적인 기능을 분담하게 하였다.

Fig. 21. Environment for firmware test(EISC core)

Fig. 22는 하드웨어와 펌웨어를 통합 테스트하기 위해 Fig. 20과 Fig. 21의 FPGA와 MCU를 원보드화한 통합 보드이다. FPGA(하드웨어)는 블루투스 베이스밴스를 30,000Gates 내에서 설계할 수 있도록 EPF10K30RC208-3 디바이스를 사용하였으며 그 밖의 특징은 Fig. 20의 보드와 같다. FPGA와 MCU 사이의 연결은 16 비트 데이터 라인(data line)을 사용하여 8 비트는 쓰기용(write)으로, 나머지 8 비트는 읽기용(read)로 사용하였다. 그리고 8 비트의 주소(address)를 사용함으로써 256 가지의 경우의 수가 생기는데, 128 가지는 쓰기를 위한 제어 신호로, 나머지는 읽기를 위한 제어 신호로 사용하였다. 이 통합 보드는 본 논문에서 블루투스 칩을 설계하는 개발 키트로서 제작되었으나 이 자체만으로도 의미를 가지며 개발 업체나 연구소, 학교 실험실 등에서

다른 무선 통신 개발을 위한 트레이닝 키트로도 사용할 수가 있다.

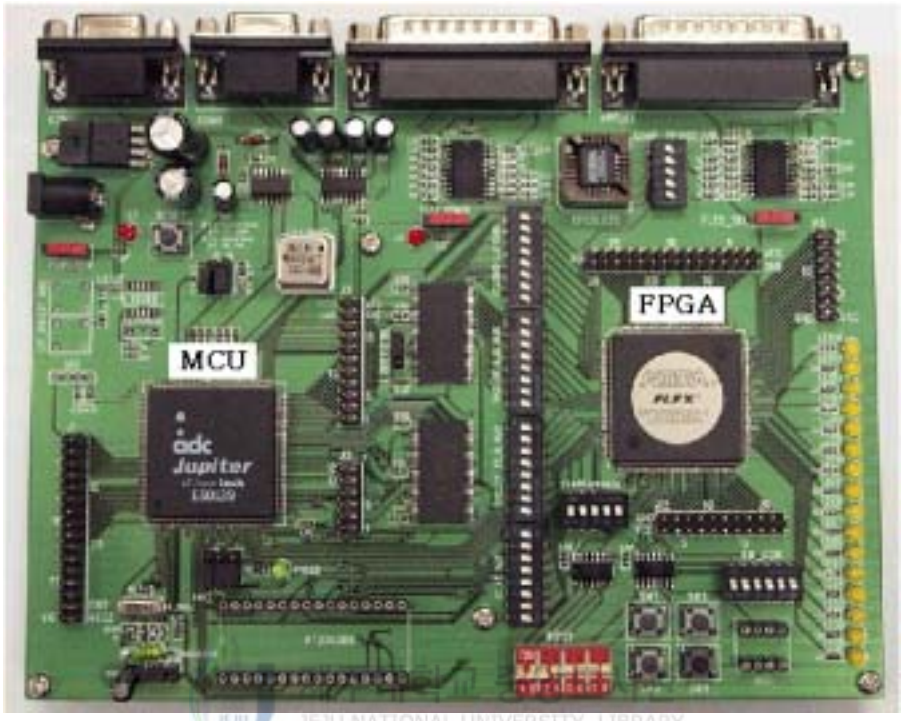


Fig. 22. The development board for bluetooth

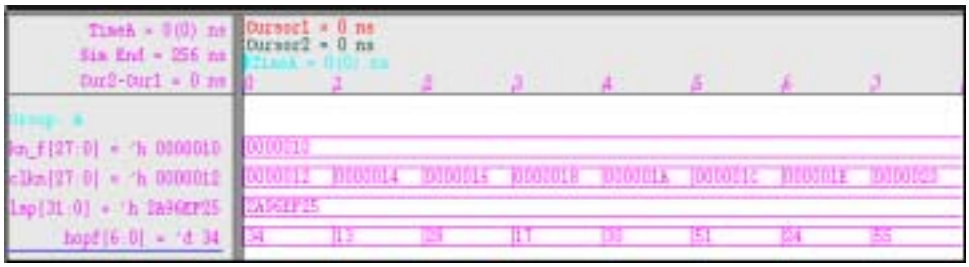
2. 도약 시퀀스의 하드웨어/펌웨어 실현 및 검증

도약 시퀀스는 블루투스 베이스밴드를 구동시키는데 있어 가장 근본적이고 중요한 부분으로서, 그 구현 및 설계가 중요하다. 각 상태에 따른 도약 시퀀스의 하드웨어적인 구현은 Fig. 3과 Table 2로부터 각 형태의 알고리즘을 기준으로 하드웨어 설계 언어인 HDL을 이용하여 실현하였다. Fig. 23은 블루투스 기기의 고유 주소 UAP&LAP가 '0x2A96EF25'라 가정하여 시뮬레이션한 결과이고 이를 블루투스

SIG(special interest group) 규격의 샘플 데이터와 비교·분석한 결과 구현의 정확성을 확인하였다.

(a) Page scan / Inquiry scan

(b) Page state / Inquiry state



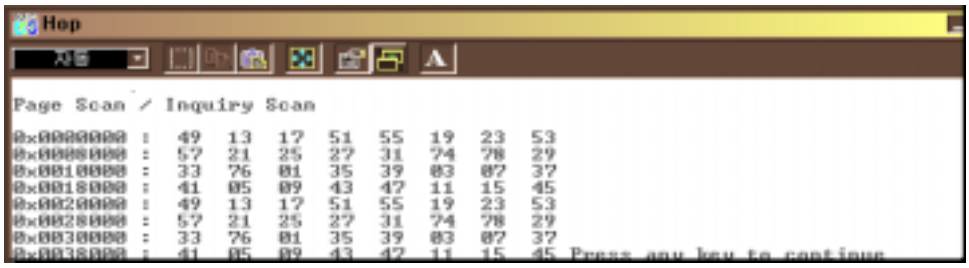
(c) Slave page response state

(d) Master page response state

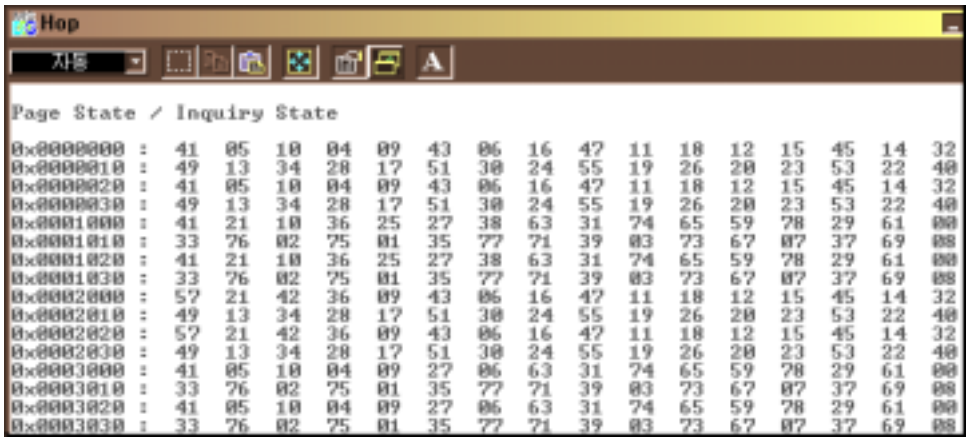
(e) Connection state

Fig. 23. The hardware simulation values of hop sequence(a)(b)(c)(d)(e)

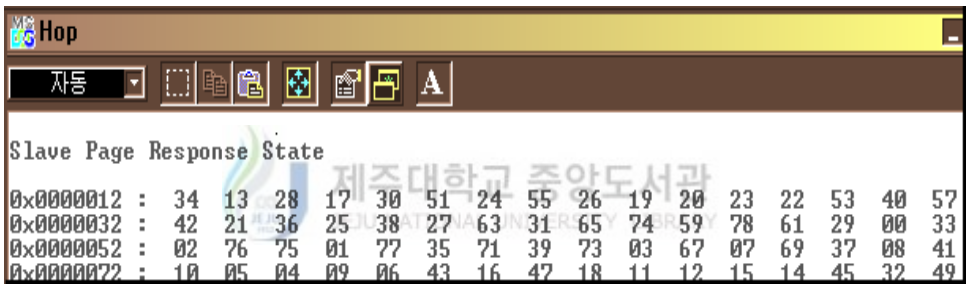
Fig. 24는 도약 시퀀스를 C 프로그램을 이용하여 펌웨어로 실현한 결과로서, 하드웨어와 마찬가지로 샘플 데이터와 같은 결과를 얻을 수 있었다.



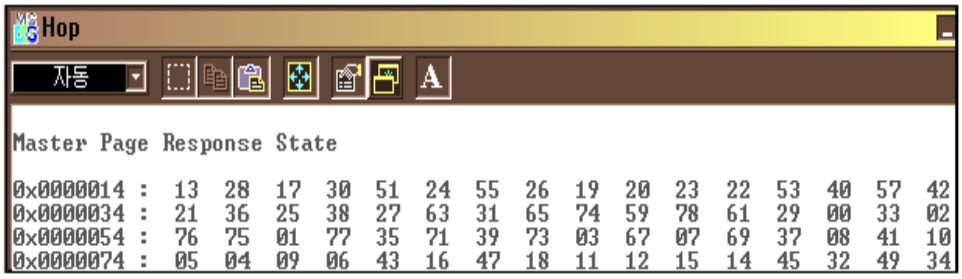
(a) Page scan / Inquiry scan



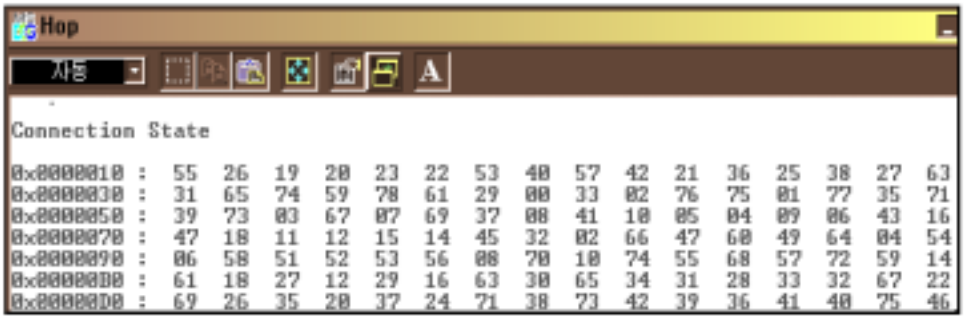
(b) Page state / Inquiry state



(c) Slave page response state



(d) Master page response state



(e) Connection state

Fig. 24. The firmware simulation values of hop sequence(a)(b)(c)(d)(e)

도약 시퀀스의 실현 결과, 도약 시퀀스 전체 크기가 하드웨어로 실현했을 때는 대략 10,000Gates 정도이고 펌웨어로 실현했을 경우는 대략 24KB를 차지하는 것으로 나타났다. 전체 베이스밴드를 20,000~30,000Gates 내외로 구현해야 하는 것을 고려해 볼 때, 하드웨어 실현 크기인 10,000Gates는 상당히 큰 크기라고 할 수 있다. 따라서 이러한 하드웨어적인 설계는 전체적인 회로의 증가를 가져오고 블루투스가 저전력, 저비용을 내세운다는 점을 생각할 때 비효율적인 설계라고 할 수 있다.

반면, Low layer protocol(베이스밴드, LM, LC)은 현재 오픈된 소스와 정확한 설계 기준이 없으므로 각 개발사마다 그 설계 기준이 다르다. 대부분의 블루투스 칩 시장을 장악하고 있는 CSR인 경우는 512KB 이하의 펌웨어 크기로 칩을 제작 ·출시하고 있다.

본 논문에서는 도약 시퀀스 블록의 효율적인 설계를 위해 하드웨어와 펌웨어 구현의 적절한 분할을 통해 설계하고자 하며 이를 위해 도약 시퀀스 블록의 구현 범위를 Fig. 25와 같이 제안하였다. 즉, Fig. 3에서 하드웨어적인 구현이 유리한 두 개의 XOR와 PERM5 블록을 하드웨어로 실현하고 나머지 부분은 펌웨어로 실현하였다. 그 결과, Table 3과 같이 하드웨어 크기는 500Gates(약1/20)로 크게 줄었고 펌웨어의 크기 또한 7.14KB(약1/3)로 축소되었다.

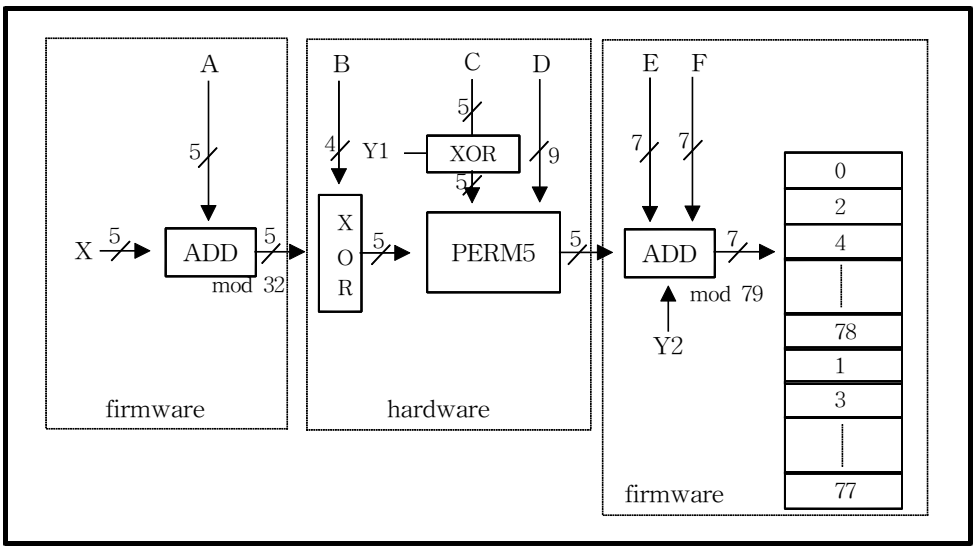


Fig. 25. The division of realization for hop sequence

Table 3. Realization size of hop sequence block

실현 방법	크기
하드웨어	10,000Gates
펌웨어	24KB
하드웨어/펌웨어 분할	HW : 500Gates FW : 7.14KB

3. 억세스 코드의 하드웨어/펌웨어 실현 및 검증

억세스 코드는 Fig. 6과 Fig. 7의 알고리즘을 이용하여 설계하였다. Fig. 26은 억세스 코드를 HDL을 이용하여 하드웨어로 실현한 시뮬레이션 결과이다. 그 결과, 블루투스 기기의 고유 주소 LAP가 '0x9E8B33'인 경우 억세스 코드 값이 MSB를 먼저 써서 '0x54E7

A2CCE331A3AE2A'임을 확인하였다.

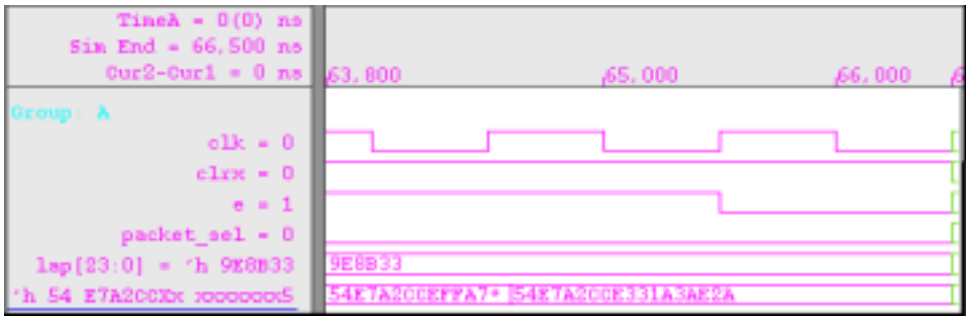


Fig. 26. The hardware simulation values of access code

Fig. 27은 억세스 코드를 C프로그램을 이용하여 펌웨어로 실현한 결과로서, 하드웨어와 마찬가지로 샘플 데이터와 같은 결과를 얻을 수 있었다.

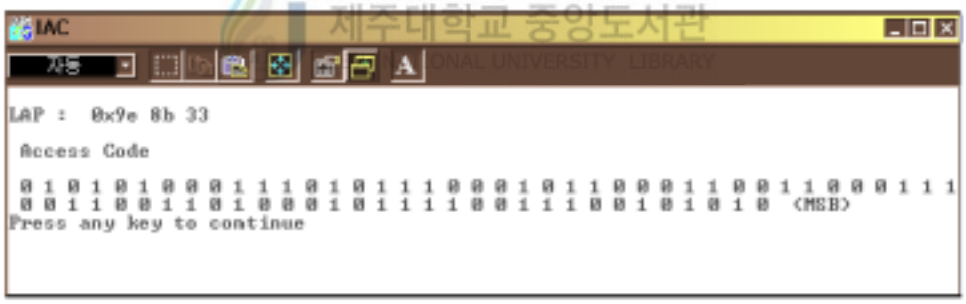


Fig. 27. The firmware simulation values of access code

억세스 코드의 실현 결과를 살펴보면, 하드웨어로 실현했을 경우 그 크기가 대략 3,000Gates를 차지하는 것을 알 수 있었고, 펌웨어로 실현했을 경우는 대략 4KB의 크기를 갖는 것으로 나타났다. 따라서 크기 면에서는 펌웨어 구현이 훨씬 효율적임을 알 수 있다. 그러나 펌웨어로 실현했을 경우 크기는 상대적으로 작아지지만 계속적으로 펌웨어에서 베이스밴드로 억세스 코드를 생성하여 내려보내야 하므로,

FPGA와 MCU를 장착하고 있는 Jupiter 칩 사이의 많은 데이터 전송으로 인해 중간 인터페이스 속도가 떨어지는 결과를 가져왔다. 현재 블루투스가 1Mbps의 전송속도를 갖기 때문에 구현 속도가 그리 중요하지 않다고 할 수 있으나 차후 전송속도가 증가될 경우 속도 문제는 매우 중요한 사항이 될 것이며, 따라서 억세스 코드의 펌웨어적인 구현은 속도 면에서 한계성을 갖는다고 할 수 있다.

이에 억세스 코드도 하드웨어와 펌웨어로의 분할 구현을 고려하여 다음 Fig. 28과 같이 Fig. 7의 비트 연산 개념이 강한 CRC34는 하드웨어로 실현하고 나머지 부분은 펌웨어로 실현하여 보았다. 그 결과 펌웨어 부분은 3.3KB로 그 크기가 다소 줄었으나, 중간 입출력 핀의 증가로 인해 하드웨어의 크기는 3,600Gates로 오히려 커지는 결과를 가져와 분할 구현은 비효율적이라는 결론을 내렸다.

따라서 본 논문에서는 억세스 코드가 연속적인 패킷의 전송에 따라 계속적으로 생성되어야 하므로 구현 속도가 빠른 하드웨어적인 방법에 의해 구현하는 것이 바람직하다고 사료된다.

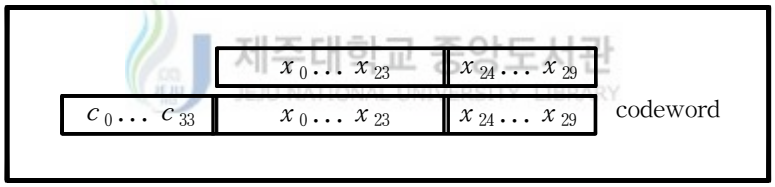


Fig. 28. The division of realization for access code

Table 4. Realization size of access code block

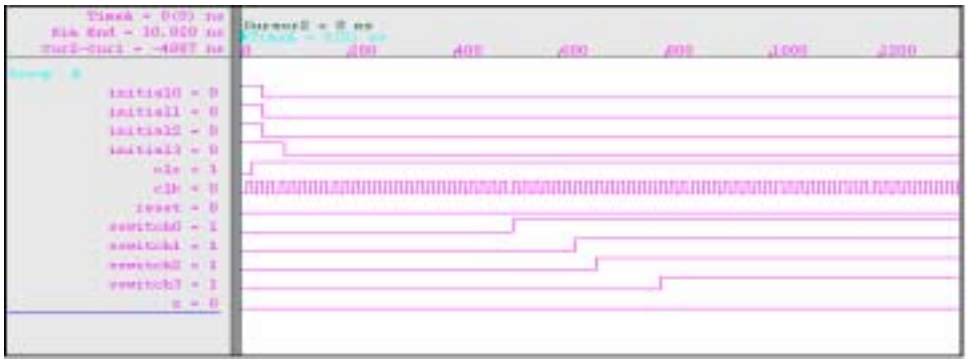
실현 방법	크기
하드웨어	3,000Gates
펌웨어	4KB
하드웨어/펌웨어 분할	HW : 3,600Gates FW : 3.3KB

4. 암호화기의 하드웨어/펌웨어 실현 및 검증

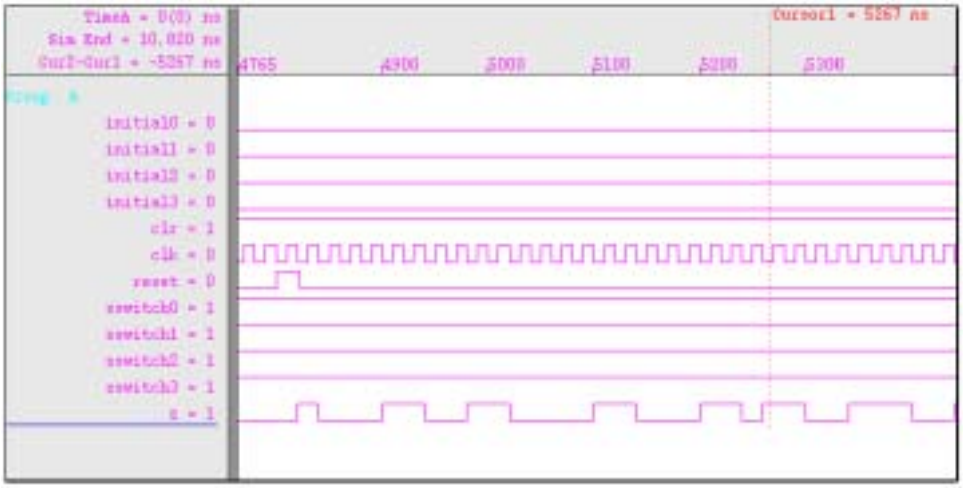
암호화기는 datapath단에서 선택적으로 구현되는 부분으로서 앞단의 결과와 XOR되어 전송하고자 하는 데이터를 암호화하게 된다. Fig. 19에서 입력 값들의 일정한 배치에 의해 생성된 초기 값들을 4개의 LFSR에 계속적으로 주면서 각각의 LFSR의 클럭이 25, 31, 33, 39일 때까지는 스위치를 “OFF” 시키고, 그 후부터는 "ON” 시킨다. 이로부터 얻게 되는 출력 값들을 끝에서부터 8비트씩 묶어 128비트를 취한 다음, 클럭이 240이 되었을 때 4개의 LFSR 레지스터에 병렬 초기 값으로 주고 다시 동작시키게 된다.

Fig. 29는 암호화기를 Fig. 19의 알고리즘에 따라 하드웨어로 실현한 결과이다. 그 결과, 크기는 Fig. 30에서 알 수 있듯이 30,000Gates의 96%인 28,800Gates로 나타났다. 따라서 이러한 하드웨어적인 구현은 전체적인 하드웨어 크기 상 적절하지 못한 것으로 결론을 내렸다. 또한, 암호화기가 datapath단의 중간에 위치하여 전송되는 데이터 비트와 계속적으로 연산이 수행되어야 한다는 점을 생각했을 때 분할 구현으로 인한 중간 인터페이스의 시간적 지연은 효율적이지 못하다.

본 논문에서는 암호화기를 펌웨어적인 방법으로 설계하고자 하며 그 구현 결과는 Fig. 31과 같다. 구현 결과 크기는 Fig. 32에서 보여지는 것처럼 대략 25KB 정도이며, 펌웨어상의 암호화기에서 생성된 암호화 열은 베이스밴드로 내려 보내지게 된다.



(a) The initial value of each LFSR and switch 'ON'



(b) The generated encryption stream

Fig. 29. The hardware simulation values of encryption(a)(b)

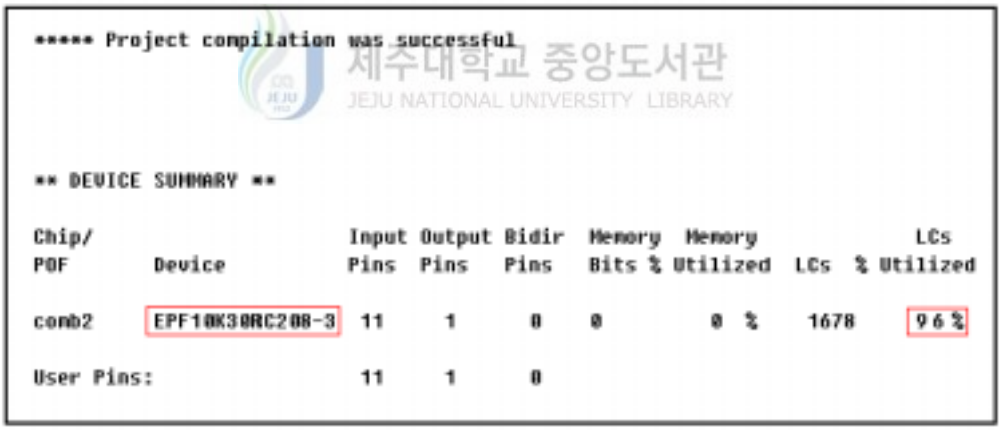
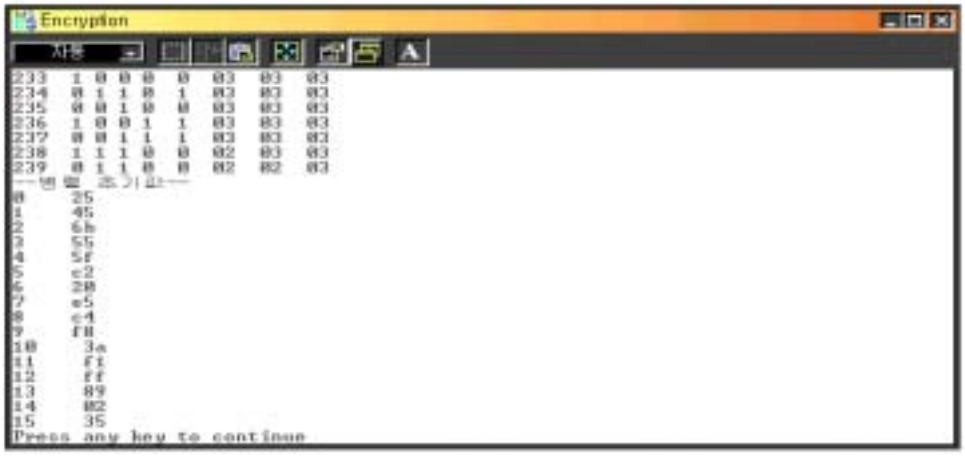


Fig. 30. Hardware realization size of encryption



(a) The 128 last generated output symbols

(b) The generated encryption stream

Fig. 31. The firmware simulation values of encryption(a)(b)

Fig. 32. Firmware realization size of encryption

5. 인증의 하드웨어/펌웨어 실현 및 검증

블루투스는 원하는 디바이스와의 연결만을 보장하고 전송하고자 하는 정보를 보호하기 위해 호출/호출 스캔 과정 후에 인증 단계를 거치게 된다. Fig. 33과 Fig. 34는 인증의 내부에 사용되는 SAFER+ 알고리즘과 인증 블록을 하드웨어적으로 실현한 결과이다. 인증 블록의 입력인 RAND가 '0x00000000000000000000000000000000'이고, 어드레스가 '0x000000000000' 그리고 링크키 값이 '0x0000000000000000000000000000000'라고 했을 때 SRES는 '0x056C0FE6'이고 ACO는 '0x48AFCDD4BD40FEF76693B113'임을 보여주고 있다. 이를 블루투스 SIG 규격의 샘플 데이터와 비교함으로써 구현의 정확성을 확인하였으며, 그 결과 인증 블록의 전체 크기는 디바이스 수용 크기인 30,000Gates를 넘는 것으로 나타났다.

따라서 블루투스에서의 인증이 두 디바이스간의 연결 설정 과정에서 한번만 수행된다는 점을 고려했을 때, 하드웨어적인 구현으로 인한 전체 회로의 증가와 그에 따른 전력 소비는 비효율적인 설계라고 할 수 있다. 결과적으로 본 논문에서는 인증 블록의 크기와 생성 빈도수를 고려하여 펌웨어로 구현하는 것이 효율적이라는 결론을 내렸다. 이를 펌웨어로 실현했을 때는 Fig. 35와 같은 결과를 얻을 수

있으며 블록의 전체 크기는 Fig. 36에서 보여지듯이 대략 28KB인 것으로 나타났다.

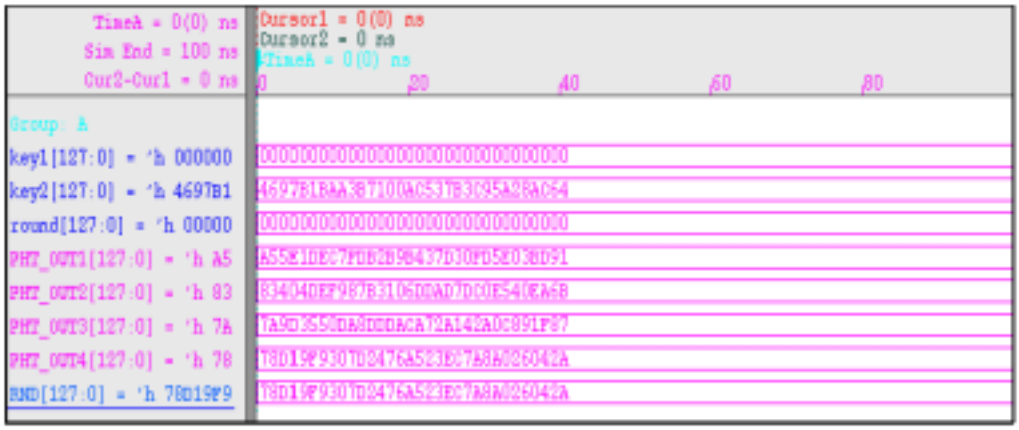


Fig. 33. The hardware simulation values of SAFER+

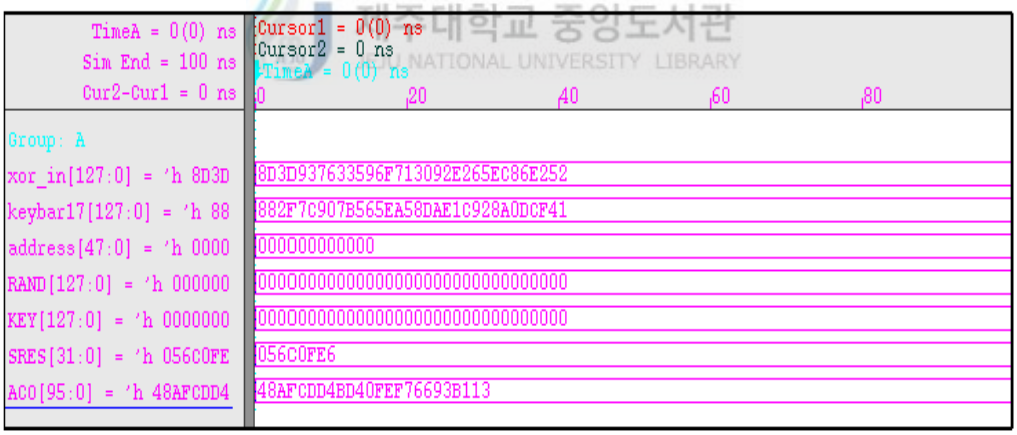


Fig. 34. The hardware simulation values of authentication

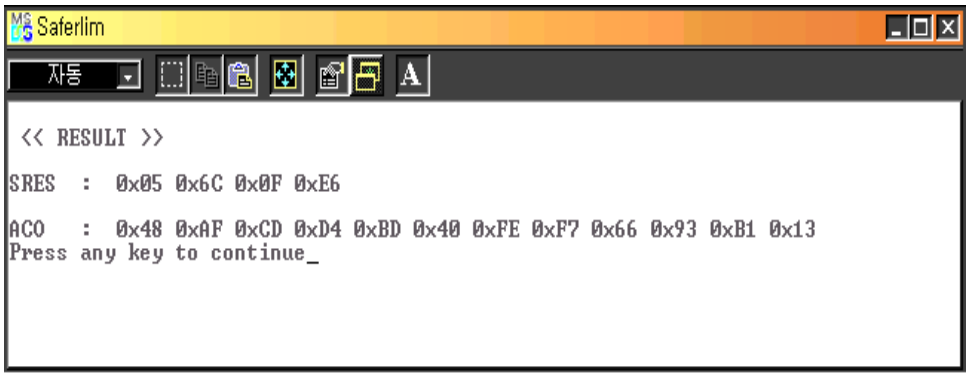


Fig. 35. The firmware simulation values of authentication

Fig. 36. Firmware realization size of authentication

Ⅳ. 베이스밴드 설계 및 통합 보드를 통한 전송 테스트

본 장에서는 3장에서 구현 및 검증을 통해 베이스밴드의 블록들을 분할한 것을 바탕으로 통합 보드 FPGA에 베이스밴드를 설계하고 펌웨어 부분은 EISC 코어를 가진 Jupiter 칩을 이용하여 작성하였다. 두 디바이스 간의 통신 가능 여부를 알아보기 위해, 두 개의 통합 보드를 제작하고 이를 사용하여 블루투스에서 사용되는 가장 기본적인 ID패킷과 NULL 패킷의 전송을 테스트하였다. RF 모듈은 논외로 두고 보드와 보드 사이는 커넥터로 연결하였다.

1. ID 패킷 전송 테스트

Fig. 37. Transmission test environment

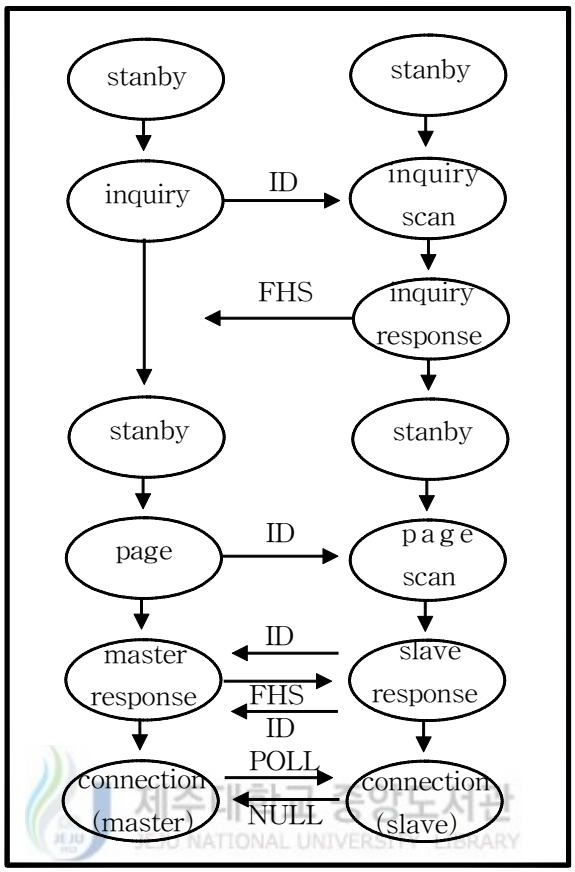


Fig. 38. State transition from standby into connection

베이스밴드에서 담당하는 역할 중 가장 중요한 것은 바로 채널 컨트롤이다. 채널 컨트롤이란 마스터와 슬래이브 사이에 커넥션이 이루어지고 피코넷이 구성되는 과정에 관련된 것으로, 이러한 과정은 스테이트(state)로 구분 지어진다. 블루투스에서의 상태는 2개의 메이저 스테이트(major state)와 7개의 서브 스테이트(substate)로 나뉘게 된다. 2개의 메이저 스테이트는 대기(standby)와 연결(connection) 상태를 나타내고, 7개의 서브 스테이트에는 조회, 조회 스캔, 조회 응답, 호출, 호출 스캔, 마스터 응답, 슬래이브 응답이 있다. 각 스테이트 간의 천이도를 Fig. 38에 나타내었다.

연결 상태가 되기 위해서는 조회와 호출 과정을 거쳐야 하는데, 조회란 주위에

서 연결할 수 있는 블루투스 디바이스를 찾고자 할 때 이루어지는 과정을 말한다. 이렇게 하여 연결할 수 있는 블루투스 디바이스를 찾아내면 어드레스와 클럭 정보 등으로 도약 시퀀스를 동기화하여 실제 커넥션을 수행하게 되는데, 이것이 호출이다. 이러한 호출과 조회는 마스터에서 수행하며 IAC와 DAC를 이용한다. 따라서 슬래이브는 IAC와 DAC를 수신할 수 있는 준비가 되어 있어야 하며 이 상태를 조회 스캔/호출 스캔이라 부른다. Fig. 37은 이러한 조회/호출 과정에서 사용되는 ID 패킷의 두 보드간 전송 테스트를 위한 환경을 보여주고 있다.

우선 조회 과정이 시작되면 송신단 상위 프로토콜인 HCI(host controller interface)로부터 LC로 "hci_inquiry_command"가 내려온다. HCI는 소프트웨어 사양에 관련된 계층으로 상위 프로토콜과 블루투스 하드웨어 모듈 사이에 서로 주고받는 패킷의 포맷과 절차를 정의하고 있다. 즉, 블루투스 모듈이 이해할 수 있는 표준 포맷으로 데이터를 만들어 내려 보내주고, 블루투스 모듈이 보내온 결과 또한 표준 패킷으로 만들어 호스트로 올려 보내는 방법을 정의하고 있다. HCI 계층으로부터 내려온 조회 명령을 감지한 LC에서는 억세스 코드 생성에 필요한 24비트 LAP를 8비트씩 세 차례에 걸쳐 베이스밴드로 내려보내도록 하였다. 이를 받아들인 하드웨어의 억세스 코드 생성기(access code generator)는 Fig. 39와 같이 억세스 코드를 생성한 후, 이를 시리얼(serial)로 상대 디바이스에 송신하게 된다.

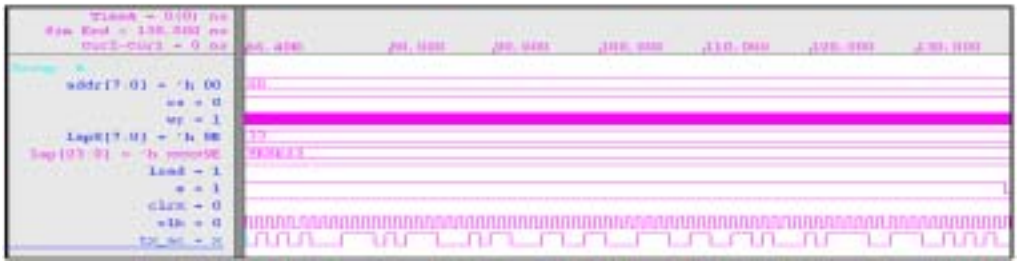


Fig. 39. The transmission of serial access code

수신단에서는 송신 과정의 역 과정을 차례로 수행하게 된다. 수신한 비트 열을 확인한 후 64비트만 추려 PN코드와 XOR한 후에 CRC34를 통과시킨다. 이때 CRC

회로의 나머지가 모두 '0'이면 성공적인 수신이며 이를 알리는 신호를 발생시키도록 하였다. Fig. 40은 수신한 시리얼 데이터 중 ID 패킷의 LAP인 '0x9E8B33'만을 추려낸 결과이고 Fig. 41은 이를 수신측 LED를 통해 확인한 결과이다.

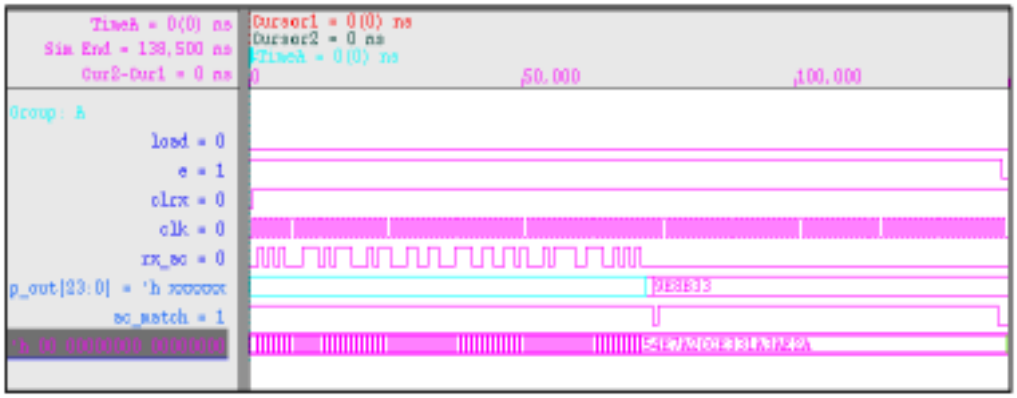


Fig. 40. The received serial access code

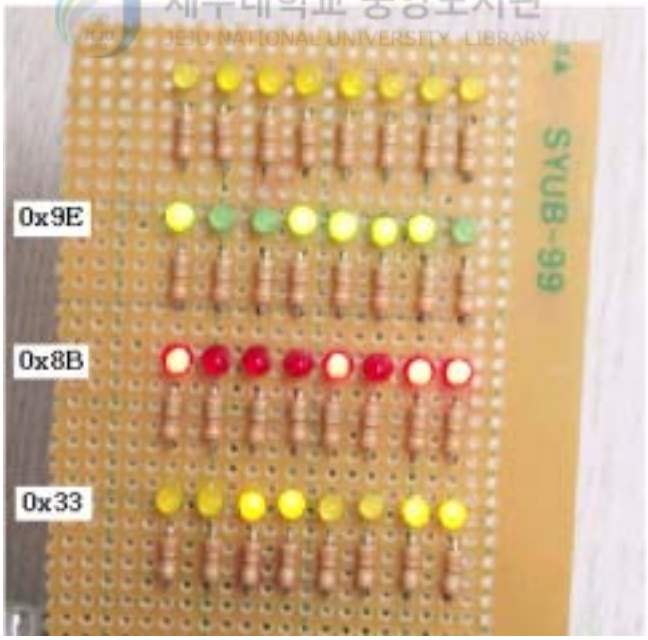


Fig. 41. The transmission result of ID packet(LAP)

2. NULL 패킷 전송 테스트

상위 프로토콜인 L2CAP/LM으로부터의 메세지 정보나 실제 데이터는 베이스밴드에서 Fig. 42의 datapath단을 거치게 된다. 전송하고자 하는 데이터는 패킷의 종류에 따라 CRC와 암호화기, 2/3 FEC의 사용 여부가 결정되며 헤더는 HEC와 비트 랜덤화기, 1/3 FEC에 의해 보호를 받게 된다. 이미 연결이 완료된 상태에서는 CAC 즉 마스터의 LAP를 사용하여 억세스 코드를 생성하게 되며, 상위 LC로부터 내려오는 헤더 정보에 의해 패킷 헤더를 생성한다. 최종적으로 억세스 코드, 헤더 그리고 페이로드 순으로 시리얼 전송을 하게 된다.

Fig. 43과 Fig. 44는 Fig. 37의 테스트 환경에서 NULL 패킷의 전송을 테스트한 결과이다. NULL 패킷은 페이로드 없이 억세스 코드와 헤더만으로 구성되어 링크의 정보를 전송하는 패킷으로, 이전 전송에 대한 성공 여부나 RX 버퍼의 상태 등을 말해준다. Fig. 43은 헤더 정보(0x123)가 HEC와 비트 랜덤화기, 1/3 FEC에 의해 코드화되어 송신되는 상태를 나타낸 것이며, Fig. 44는 수신측에서 수신한 시리얼 데이터 중 10비트의 헤더 정보만을 추려 통합 보드 LED를 통해 보여주는 것으로 헤더 각 필드의 값을 확인할 수 있다.

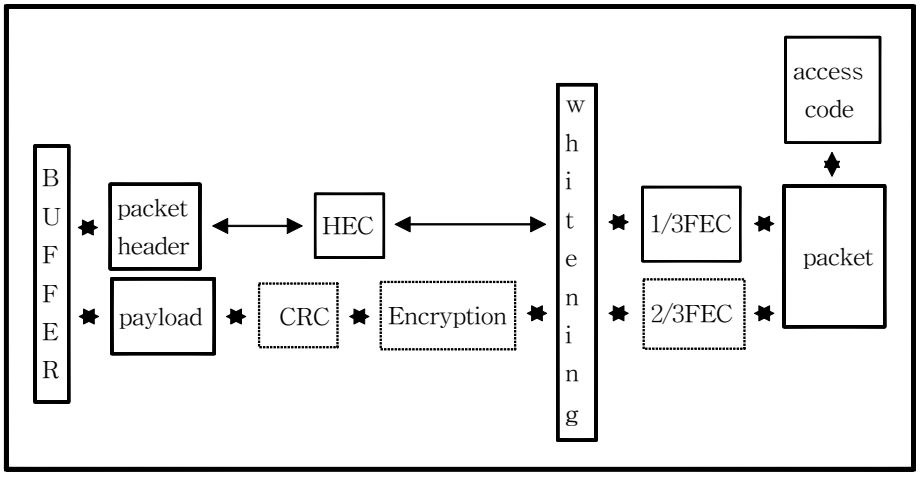


Fig. 42. Datapath process

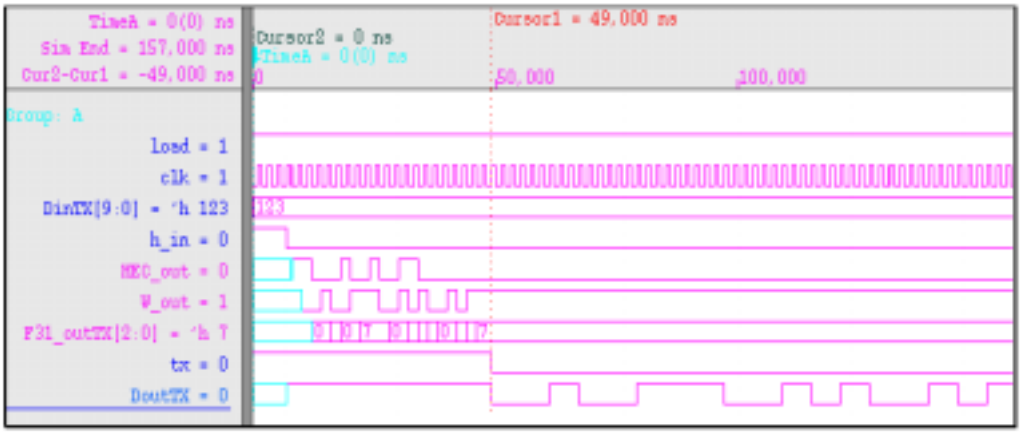


Fig. 43. The transmission of serial header information

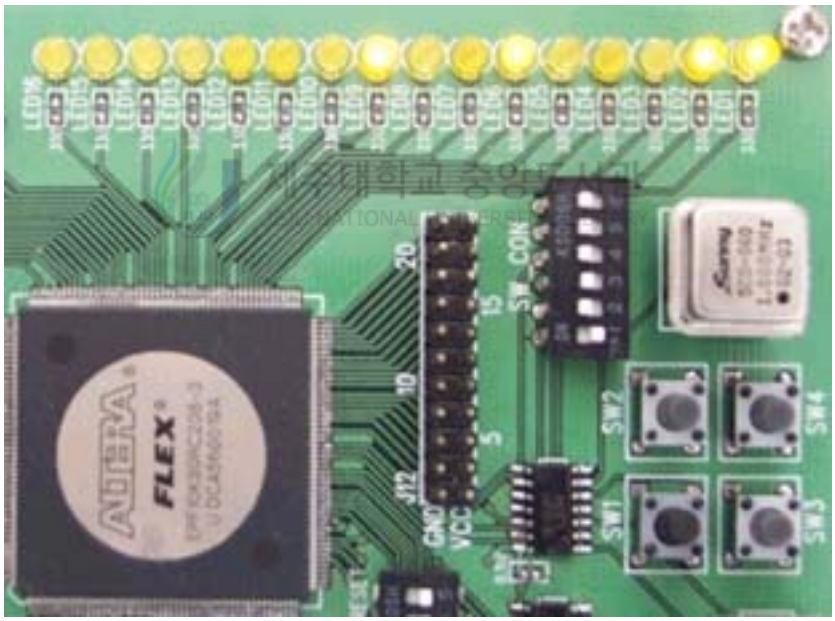


Fig. 44. The transmission result of NULL packet(header=0x123)

Ⅴ. 결론

본 논문에서는 블루투스 칩 설계의 기반을 마련하기 위해, 베이스밴드의 효율적인 분할 설계 방안을 제안하였다. 블루투스 SIG 규격 1.1에 따라 베이스밴드의 구조 및 기본 기능을 분석하였고, 베이스밴드의 블록들 중 MCU에 장착이 용이한 블록들을 하드웨어와 펌웨어로 구현하여 비교함으로써 적절한 구현 방안을 제안하였다. 또한, 이를 바탕으로 실제 베이스밴드의 일부 블록들을 펌웨어로 설계함으로써 MCU에서 베이스밴드의 기능을 분담하게 하였다.

도약 시퀀스의 경우 알고리즘의 분할 구현 방법을 제안하여 하드웨어 구현이 유리한 부분은 FPGA에 설계하고, 펌웨어 구현이 유리한 부분은 MCU에 기반 하여 작성하였다. 도약 시퀀스 블록을 하드웨어 또는 펌웨어만으로 실현했을 때와 비교해서 분할 실현에 의한 크기가 각각 5%와 33.33%로 줄어드는 결과를 가져왔다. 암호화기와 인증 블록은 하드웨어로 실현하였을 경우 그 크기가 암호화기는 28,800Gates이고, 인증 블록은 30,000Gates를 넘는 것으로 나타났다. 따라서 이러한 하드웨어적 설계는 전체 베이스밴드의 구현 크기 상 적절하지 못한 것으로 판단하여 펌웨어적으로 실현하였고, 각각 25KB와 28KB의 크기를 갖는 것으로 나타났다.

분할 구현 기법의 실현 여부를 검증하기 위해 MCU와 FPGA를 통합 보드화하여 MCU 기반 개발 키트를 제작하였으며, 제안한 구현 방법을 적용하여 베이스밴드를 FPGA에 설계하고 베이스밴드의 일부 블록과 링크 컨트롤러 부분은 펌웨어로 작성하였다. 마지막으로, 두 개발 키트 사이에서 블루투스의 가장 기본적인 ID 패킷과 NULL 패킷의 전송을 확인함으로써, 분할 설계에 의해 제작된 개발 보드가 블루투스 장치로 동작함을 확인하였다.

본 논문에서 제안한 하드웨어/펌웨어 분할 구현 방법은 베이스밴드의 고정적인 하드웨어 설계에서 벗어나, 베이스밴드 블록들의 특징에 따라 하드웨어와 펌웨어를 적절히 융합하여 최적의 구현 방법을 찾음으로써 좀 더 효율적인 시스템을 설계할 수 있는 방안이 될 것으로 사료된다.

참고 문헌

- BRENT A. MILLER, CHATSCHIK BISDIKIAN, 2000, Bluetooth Revealed, Prentice-Hall, pp.83-122
- Jennifer Bray and Charles F Sturman, 2001, BLUETOOTH Connect without Cables, Prentice-Hall, pp.41-90, pp.291-331, pp.399-421
- Nathan J.MULLER, 2000, BLUETOOTH DEMYSTIFIED, McGraw-Hill Companies,
- SIG, 1999, Specification of the Bluetooth System Version 1.0B Part B : Baseband specification, pp.41-178.
- SIG, 2000, Specification of the Bluetooth System Version 1.1 Part B : Baseband specification, pp.41-178.
- Tranter, William H/Woerner, Brian D/Reed, Jeffrey H/Rappaport, Theodore S/Robert, Max, 2000, Wireless Personal Communications, Kluwer Academic Pub
- 이문수, 강치운, 오종택, 이정재, 변건식, 2001, 한국어판 블루투스 Connect without Cables, 홍릉과학출판사, pp.45-101, pp.313-358, pp.431-454
- 조경연, 1999, A Study on Extendable Instruction Set Computer 32 bit Micro Processor, 부경대학교
- 한국 무선국 관리 사업단, 2000, 블루투스(BLUETOOTH) 기술 전파 제 96 호

URL :

- http://pbt.co.kr/bluetooth/bluetooth3/4.htm
- http://sun.uos.ac.kr/network/bluetooth/bt_03.htm
- http://www.ee.princeton.edu/~rblee/safer+
- http://www.korwin.co.kr/home/

감사의 글

많은 고민과 생각 끝에 어렵게 결정했던 대학원 진학이었던 만큼 누구보다 열심히 하고 싶었고 많은 것을 할 수 있으리라 생각했습니다. 그러나 2년이라는 시간은 어느덧 다 끝나가고 졸업을 앞두게 되었습니다. 배워야 할 것은 아직도 많은데 이렇게 끝내게 되어 아쉬움이 남습니다.

2년의 시간동안 이 한편의 논문을 완성하기 위해 준비하고 노력해 왔는데 지금 되돌아보면 좀 더 성실히 노력하지 못한 것이 못내 아쉽습니다. 항상 여러 일로 바쁘시면서도 많은 지도를 아끼지 않으셨던 임재윤 지도 교수님께 가장 먼저 고마움을 전하고 싶습니다. 농담 섞인 질책이 오히려 더 열심히 하는 계기가 되었고, 교수님께서 심어주신 할 수 있다는 자신감은 제가 살아가는데 큰 힘이 되리라 생각합니다. 교수님 감사합니다. 그리고 바쁜신데도 논문을 쓰는데 많은 도움을 주신 이용학 교수님과 강진식 교수님께 깊은 감사를 드립니다. 학부 때부터 저의 학문의 길을 열어주신 문건 교수님, 김홍수 교수님, 양두영 교수님께도 감사의 마음을 전합니다. 교수님들의 건강이 하루 속히 회복되시길 기도하겠습니다.

연구실은 달라도 항상 선배로서의 모범을 보여주셨던 강부식 선배님, 홍성욱 선배님, 이권익 선배님 그리고 어설픈 농담이 귀여운 봉수 오빠에게도 고마움을 전합니다. 저의 실수를 꾸짖어 주고 그러면서도 대학원 생활을 잘 해나갈 수 있도록 인도해 준 성익 오빠에게도 고마운 마음을 전합니다. 연구실 선배로서 많은 도움을 준 재필 오빠와 애교 많고 따뜻한 영애 언니에게도 고마움을 전하고 싶습니다. 또 2년 동안 서로의 어려움을 어루만져 주면서 고락을 함께 했던 종국 오빠, 창윤 오빠, 은진 언니, 철우 오빠도 새로 시작하는 사회 생활을 잘 해 나가길 바랍니다. 열심히 자기의 길을 가는 영배 오빠, 사랑스런 나의 후배 현미, 그리고 혼자 두고 졸업하는게 못내 섭섭한 나의 착한 친구 수미, 광식 오빠, 영길 오빠, 재오・성민 오빠에게도 고마움을 전합니다. 마지막으로, 내가 여기 있기까지 사랑과 격려를 아끼지 않으셨던 고마우신 나의 부모님과 오빠・언니들, 그리고 사랑하는 나의 조카 현지・현에게도 저의 깊은 사랑을 전하고 싶습니다.