碩士學位論文

EISC 기반 블루투스 베이스밴드의 설계 및 검증

濟州大學校 大學院

通信工學科

金 賢 美

2003 年 12 月

EISC 기반 블루투스 베이스밴드의 설계 및 검증

指導敎授 林 載 允

金 賢 美

이 論文을 工學 碩士學位 論文으로 提出함

2003 年 12 月

金賢美의 工學 碩士學位 論文을 認准함

審査委員長 金 興 洙 印

委 員 左 政 祐 印

委 員 林 載 允 印

濟州大學校 大學院

2003年 12 月

Design and Verification of Bluetooth Baseband based on EISC

Hyun-mi Kim
(Supervised by professor Jea-yun Lim)

A thesis submitted in partial fulfillment of the equirement for the degree of Master of Science

2003. 12.

This thesis has been examined and approved.

Thesis director, Heung-Soo Kim, Prof. of Telecom. Eng.

(Name and signature)

Date

DEPARTMENT OF TELECOMMUNICATION ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

목 차

Abstract

Bluetooth is a standard bidirectional communication technique for portable units such as mobile phones, PDAs and laptops without complicated wires within short distances.

In this thesis, the method of a design for a bluetooth baseband is presented based on EISC. And the baseband is realized and its efficiency is verified. The entire architecture and basic functions of baseband are analyzed on the basis of the bluetooth SIG specification 1.1. And important blocks are realized by combining the hardware with the firmware properly and the efficient implementation is found as comparing with the criterion of the frequency of utilization, its size, the performance and the stability, etc. From the comparison, the important fact is found; it is very efficient that the part of an access code generation and the authentication are realized with only firmware, the access code correlation, the encryption, and the bluetooth clock with only hardware and the hop sequence with both hardware and firmware. Connecting the baseband block to the SE3208 core of the EISC type by both the single bus bridge and the double bus bridge, the operating states by the each case on the FPGA test board are observed. From the result, the performance by the double bus bridge is superior to by the single bus bridge; 63.6% of the clock skew, 84.5% of the setup time, 97.5% of the clock to output, 98.9% of the clock period, and 81.3% of the combinational path delay. Although the connecting by the single bus bridge is better for using the number of LUT and gate, the double bus bridge is suggested because bluetooth is sensitive system to the operation timing. Bluetooth designing by the method, the bluthooth one-chip could increase the operating speed and the performance and be a stable system.

Ⅰ. 서론

블루투스(bluetooth)는 이동전화, 컴퓨터, PDA(personal digital assistant) 등 근거리 무선접속을 쉽게 해주는 기술로, 컴퓨터 및 통신 산업계의 규격이다. 이는 1994년 스웨덴의 에릭슨 이동통신그룹(ericsson mobile communication)이 휴대폰과 주변기기들 간의 소비전력이 적고 가격이 싼 무선 인터페이스 연구를 하면서 시작되어 1998년 5월에는 에릭슨, 핀란드의 노키아, 미국의 IBM과 인텔, 일본의 도시바로 구성된 블루투스 SIG(special interest group)가 발족되었다. 블루투스는 여러 가지 기술적인 장애를 극복하기 위해 개발되어 온 것으로, 5가지 장점과 3가지 목표를 세우고 있다. 여기서 5가지 장점은 이동 단말의 부가가치 증대, 케이블과 커넥터 생략, 간단한(수시로) 네트워크 구축, 모든 접속기기의 광범위한 동시 이용, 그리고 상호접속성을 고려한 신형의 제품 개발이고 3가지 목표는 저가격화, 저소비전력화, 그리고 소형경량화이다.(Jennifer Bray 등, 2001)

블루투스는 10m이내의 범위 안에서 통신하는 것을 목표로 하며 출력에 따라 100m까지 가능하다. 블루투스의 전송속도는 1Mbps로, 이는 2.4GHz대의 ISM (industrial scientific medical)대역에서 비교적 손쉽고 저렴하게 시스템을 실현할 수 있는 전송속도이다. ISM 대역은 누구나 허가 없이 사용가능하기 때문에 이 대역에서 동작하는 무선 시스템은 다양한 예측 불가능한 간섭을 견디어야 한다. 이러한 간섭은 스펙트럼에서 사용되지 않는 부분을 찾는 적응 기법(adaptive scheme)에 의해 회피될 수 있고 주파수 확산 방식(spread spectrum)에 의해 해결될 수 있다. 블루투스에서 사용되는 FHSS(frequency hop spread spectrum) 방식은 저가·저전력의 무선 장비구현을 가능하게 하고, 이 기술의 특성상 보안성을 높일 수 있다. 이는 블루투스가 휴대용 혹은 이동형 기기에 많이 응용되기 때문에 대단히 큰 장점이 된다. 사용자 보호와 정보 보안을 위해 시스템은 계층 대응 환경에 적당한 보안 기능을 제공해야 하는데, 블루투스에서는 인증(Authentication)을 위한 요구/응답(challenge-response) 루틴, 열 암호화(stream cipher) 그리고 세

션 키 생성으로 이를 구현한다. 블루투스는 음성부호화방식인 CVSD(continuous variable slope delta modulation)를 채용하고 있고 문자 데이터의 전송은 물론이고 음성 전송에도 사용할 수 있다. 블루투스가 갖는 가장 큰 특징은 전방향 무선통신이고 차폐물 투과성이 있어 공간 활용도를 높이면서 휴대 정보통신 기기를 가방이나 주머니에 넣은 채로 다른 정보통신기기와 통신할 수 있다는 점이다.(Bluetooth SIG, 2000), (http://www.bluelogic.co.kr)

이와 같은 특징을 갖는 블루투스는 미래 사회를 대변할 유망 기술로 각광 받고 있는 유비쿼터스(ubiquitous)의 주 모델이 될 가능성이 크다. 유비쿼터스는 사용자가 컴퓨터나 네트워크를 의식하지 않는 상태에서 장소에 구애받지 않고 자유롭게 네트워크에 접속할 수 있는 환경으로, 이 시대가 열리게 되면 다양한 공간에서 IT(information technology) 활용이 늘어나고 네트워크에 연결되는 컴퓨팅 사용자의 수도 늘어나는 등 IT 산업의 규모와 범위는 더욱 커지게 된다. 즉 유비쿼터스 네트워크를 위해서는 모든 전자기기에 컴퓨팅과 통신 기능이 부가되어야 하므로 저렴하고 소형인 무선 통신이 각광 받을 것이고, 블루투스가 이에 알맞은 특성을 보유하고 있다.(http://auto.ats.go.kr/ats_admin/upfiles/data/UBiquitous.pdf)

블루투스의 프로토콜 스택에서 호스트에 해당하는 상위 프로토콜 스택은 공개되어 있지만, 호스트 컨트롤러에 해당하는 하드웨어와 펌웨어는 공개된 소스가 없고 그 구현 또한 복잡하다. 그리고 외국 기업에서 이미 블루투스 칩이 생산되고 있어 국내 기업에서는 칩 경쟁력이 없다고 판단하여 외국에서 설계된 칩을 구입해서 사용하고 있는 실정이다. 따라서 보다 근본적인 하드웨어와 펌웨어의 기술 접근이 시급하며, 칩 설계의 솔루션을 찾는 일은 국내에서 중요한 의미를 갖는다. 이와 동시에 고려해야할 사항은 1998년 이전에 5칩 솔루션으로 제공되었던 블루투스 칩 설계가 임베디드화와 SoC(system on chip)화에 부합하여 요즘에는 단일칩 솔루션으로 접근되고 있다는 것이다. 이에 따라서 블루투스 칩에 장착할 CPU(central processing unit)의 성능이 중요하게 되고, 베이스밴드(baseband)의 일부 구현이 하드웨어가 아닌 펌웨어로도 가능하게 된다. 개발에 있어 가격 대 성능 대 시간에서 많은 모순점이 있는데, 이를 블루투스에서 요구하는 규격에 벗어나지 않는 범위 내에서 적절히 조절해야 한다. 또한 단일칩 설계에 있어 알고리즘을 구현하는 형태에 따라 성능과 칩 크기가 달라질 수 있는데, 여기서 베이스밴드의 블록 구현

과 전체 데이터 흐름을 제어하는 버스 브릿지의 설계가 중요하다.

본 논문에서는 블루투스 칩을 효율적으로 설계하기 위하여 블루투스 SIG 스펙 1.1을 기준으로 베이스밴드의 전체적인 구조 및 기본 기능을 분석하고, 블루투스 단일칩의 구조를 제시하며, 이를 구현하기 위한 설계 방법 및 테스트 환경을 살펴본다. 단일칩 구현에 앞서 블루투스의 주요 블록을 하드웨어와 펌웨어로 구현하여 그 결과를 크기, 성능, 그리고 안정성 등을 비교·검토함으로써 기본적인 블록에 대한 효율적인 구현 형태를 제시한다. 그리고 검증된 베이스밴드 블록을 단일 버스 브릿지와 이중 버스 브릿지의 두 가지 방식으로 EISC(extensible instruction set computing) 방식의 SE3208 코어에 패리패럴로 연결하여 FPGA(field programmable gate array)로 검증하고 그 결과를 비교·검토함으로써 적합한 버스 브릿지 방법을 제안한다.

본 논문의 구성을 살펴보면, Ⅱ장에서는 블루투스 SIG 스펙 1.1을 기준으로 베이스밴드의 구조 및 기본 기능을 기술하고, Ⅲ장에서는 블루투스 단일칩 솔루션을 구현하기 위한 구조를 제시하고 설계 방법 및 테스트 환경을 설명한다. Ⅳ장에서는 블루투스 베이스밴드의 주요 블록을 하드웨어와 펌웨어로 구현하여 비교함으로써 효율적인 구현방안을 모색하고, 이 결과를 토대로 베이스밴드를 구현하여 SE3208 코어에 두 가지 방법으로 패리패럴로 연결한다. 그리고 FPGA로 그 동작을 확인하고 구현 결과를 비교함으로써 적합한 연결 방식을 제안한다. 마지막으로 Ⅴ장에서는 본 논문의 결론을 맺는다.

Ⅱ. 베이스밴드의 구조 및 기본 기능

베이스밴드는 RF(radio frequency) 모듈을 제외하면 블루투스 스택에서 가장 하부에 위치한 계층으로, 일반적인 LAN(local area network)을 예로 들면 NIC (network interface card)정도에 해당한다. 베이스밴드는 가장 하부에서 블루투스 디바이스 간의 연결을 담당하고 그 구성도는 그림 1과 같다.

블루투스는 1MHz인 RF 채널을 79개로 나누고 초당 1600회 주파수 호핑 (frequency hopping)을 하는데, 이 채널 정의와 호핑 시퀀스 선택 등은 모두 베이스밴드에서 이루어진다. 또한 베이스밴드는 링크 및 피코넷(piconet)을 설정하며 표준 패킷을 정의하고 생성한다.(Bluetooth SIG, 2000)

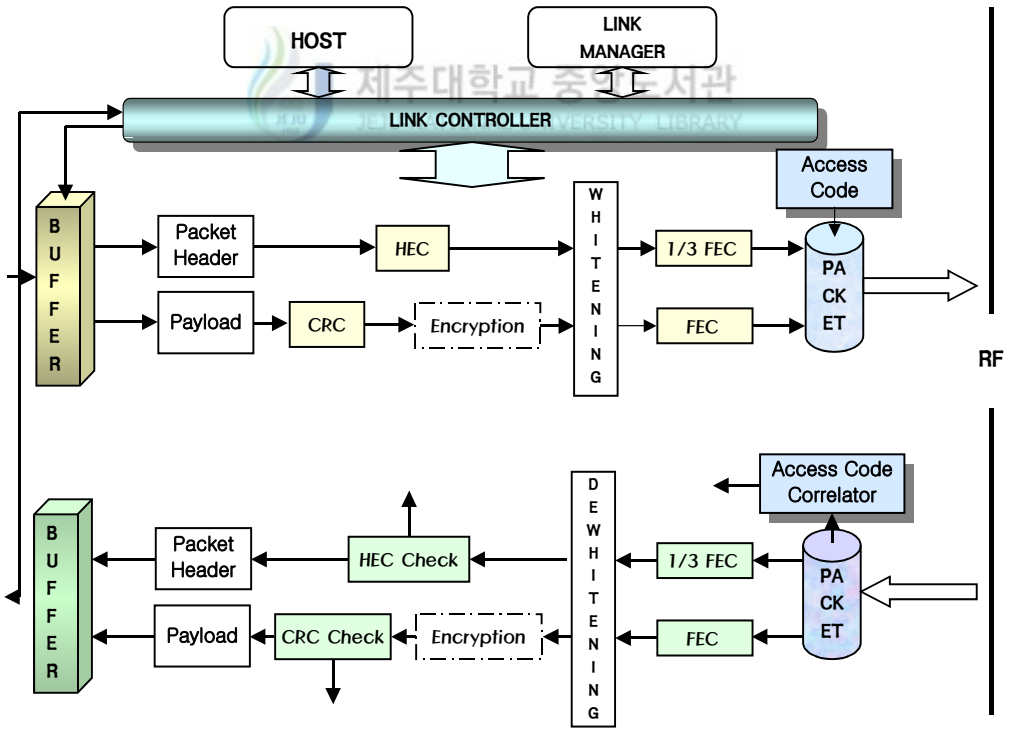


Fig. 1. Baseband/LC architecture

1. 패킷

베이스밴드에서의 표준 패킷은 그림 2와 같이 접근 코드, 헤더, 페이로드로 구성되어 있고, 그 역할 및 링크의 종류에 따라 링크 컨트롤 패킷, ACL(asynchronous connection less) 패킷, 그리고 SCO(synchronous connection oriented) 패킷으로 나뉜다. 또한 이들은 페이로드 길이, FEC(forward error correction)와 CRC(cyclic redundancy check)의 사용 여부 등에 따라 더 세분화된 패킷으로 구분된다.

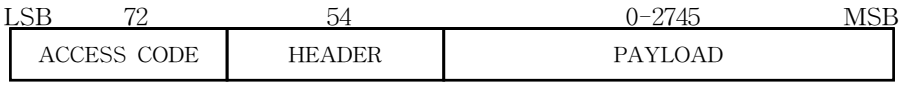


Fig. 2. Standard packet format

(1) 접근 코드

접근 코드(access code)는 모든 패킷에 포함되며 패킷의 가장 처음에 전송되고, 패킷을 수신했을 때 가장 먼저 체크되는 부분이다. 이때 접근 코드는 수신된 패킷이 자신이 속한 피코넷의 데이터인지를 판단하며 다른 피코넷의 데이터인 경우 패킷은 버려지게 된다. 또한 수신된 프레임의 시작을 검출하여 시스템이 수신 프레임의 클럭에 맞출 수 있도록 한다.

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

Fig. 3. The kind of access code

그림 3에서 볼 수 있듯이, 접근 코드는 프리엠블(preamble), 동기워드(sync word), 트레일러(trailer)로 구성되고 그 목적에 따라 사용되는 LAP(lower address part)값이 달라지며 헤더가 있는 경우는 72bits의 길이를, 없는 경우는 68bits의 길이를 갖는다. 그리고 그림 4는 그림 3의 동기워드 생성과정을 자세히 보여준 것이다.

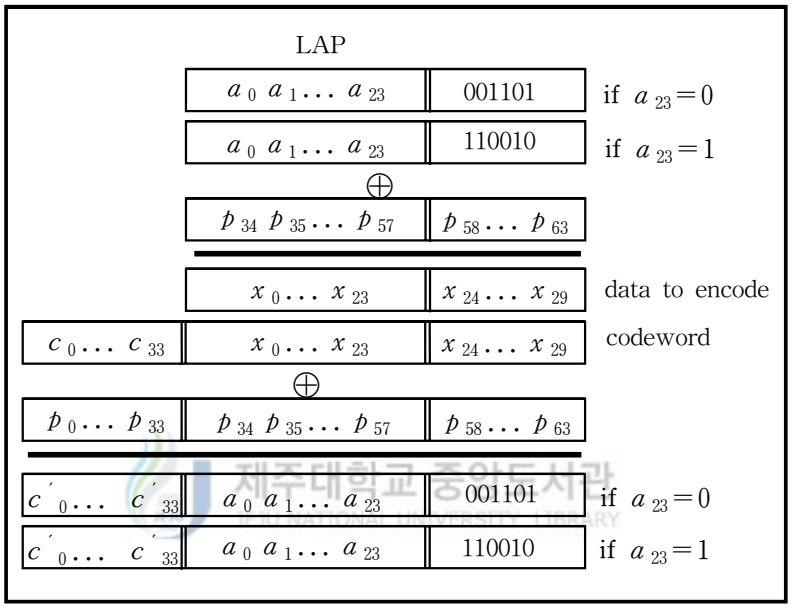


Fig. 4. Construction of the sync word

(2) 헤더

LC(link controller)정보를 포함하는 패킷 헤더는 그림 5와 같이 HEC(header error check)를 포함한 6개의 필드로 구성된다. 데이터 패스단에서는 18bits의 헤더 정보를 1/3 FEC로 부호화 하여 결과적으로 54bits의 헤더를 얻는다.

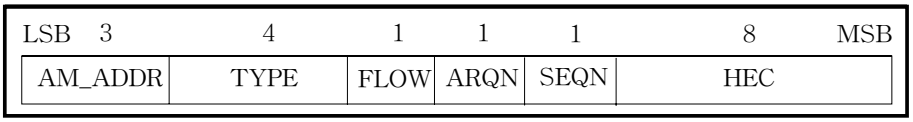


Fig. 5. Header format

헤더의 마지막 필드인 HEC의 생성 회로는 그림 6과 같고 레지스터 초기값은 마스터의 블루투스 기기의 고유 주소(BD_ADDR:bluetooth device address)중 8bits UAP(upper address part)를 사용한다.

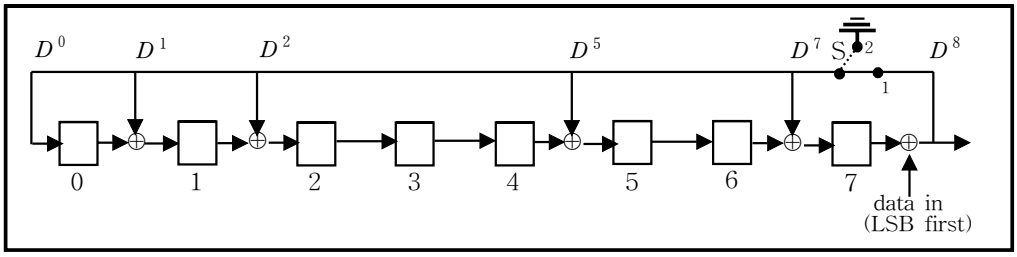
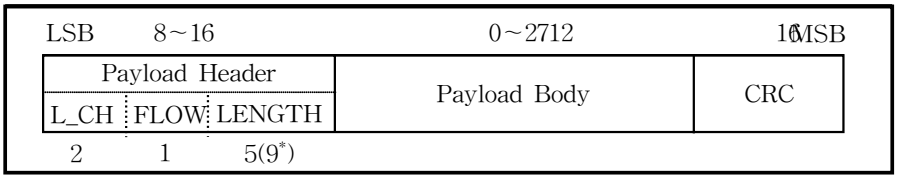


Fig. 6. The LFSR circuit generating the HEC

(3) 페이로드

페이로드는 실제 전송하고자 하는 정보 부분으로서 상위 프로토콜인 L2CAP(logical link control and adaption protocol)나 LM(link manager)으로부터의 메시지 정보나 실제 데이터를 포함한다. 페이로드는 두개의 필드로 구분되는데, ACL 패킷만이 갖는 데이터 필드와 SCO 패킷만이 갖는 음성 필드가 그것이다.

데이터 필드는 그림 7과 같이 페이로드 헤더, 바디, 필요에 따라 CRC로 구성된다. 여기서 CRC 회로는 그림 8과 같고 레지스터의 상위 8bits는 '0'으로, 하위 8bits는 마스터의 UAP로 초기화된다.



*. *In the case of multi-slot packets, length indicator is extended by four bits into the next byte*

Fig. 7. Payload format of ACL packet

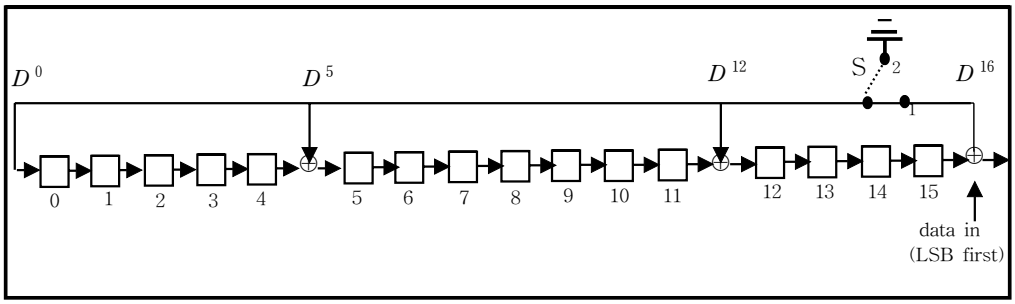


Fig. 8. The LFSR circuit generating the CRC

(4) FEC

패킷은 입력 데이터에 추가 패리티 비트를 첨가함으로써 비트 오류가 검출되고 정정된다. FEC는 패킷의 형태에 따라 non, 1/3, 2/3를 선택하여 사용한다.

가장 강력한 그림 9의 1/3 FEC는 송신 측에서 각 비트를 세 번 반복하여 부호화해서 전송하고, 수신 측에서 다수결 기능을 실행하여 복호화한다. 그렇기 때문에 패킷의 핵심인 링크정보를 포함하는 패킷 헤더는 1/3 FEC 방식을 사용한다.

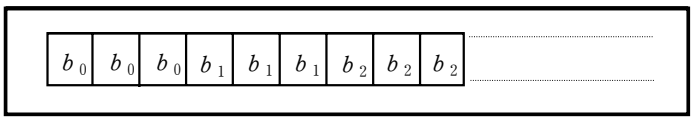


Fig. 9. Bit-repetition encoding scheme

2/3 FEC는 LFSR(linear feedback shift register)로 구현되는 (15, 10) 축약 해밍 코드이고 그림 10의 연산으로 모든 10bits 입력에 대하여 15bits가 출력된다. 이는 10bits 입력 중에서 2bits의 오류를 검출하고 1bit의 오류를 정정한다.

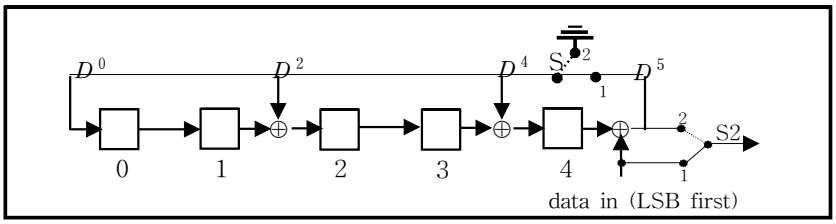


Fig. 10. LFSR generating the (15,10) shortened hamming code

(5) 비트 랜덤화

비트 랜덤화(whitening)는 데이터를 랜덤화하기 위하여 의사 랜덤(pseudo random) 비트 순서를 데이터 비트열에 더하는 것으로, 이는 '0'이나 '1'의 긴 시퀀스 가능성을 줄이고 DC 바이어스를 제거하는 역할을 한다. 그림 11은 비트 랜덤화를 위한 회로로, 각 레지스터는 마스터 클럭의 일부인 CLK_{6-1}과 1bit의 MSB가 '1'로 초기화된다. 단, FHS(frequency hop sychronisation) 패킷인 경우는 예외로 홉 시퀀스의 X 입력 값과 2bits의 MSB가 '1'로 초기화된다. 초기화 후에는 패킷 헤더와 페이로드가 스크램블 된다.

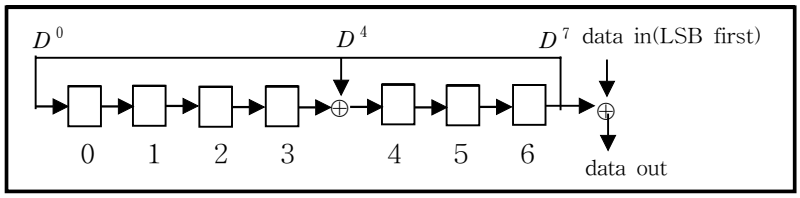


Fig. 11. Data whitening LFSR

2. 블루투스 클럭 관리

블루투스 클럭은 TX-RX 데이터 교환을 동기 시키고 손실 패킷과 재 전송 패킷사이를 구별하며 의사 랜덤 열을 생성하도록 해준다. 그림 12의 블루투스 클럭은 전원을 켤 때 리셋 되고 312.5μs마다 증가하는 28bits 카운터로 구성된다. 그림 13에서 CLKN(native clock)은 블루투스 클럭이고, 여기에 마스터 클럭 옵셋과 슬레이브 예측 클럭 옵셋을 추가하여 CLK(master clock)와 CLKE(estimated clock)를 생성한다.

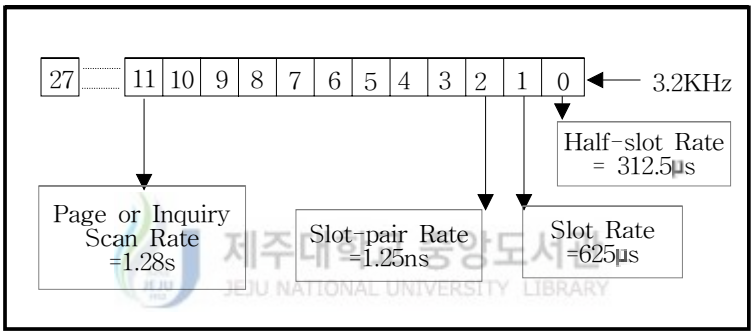


Fig. 12. Bluetooth clock generation

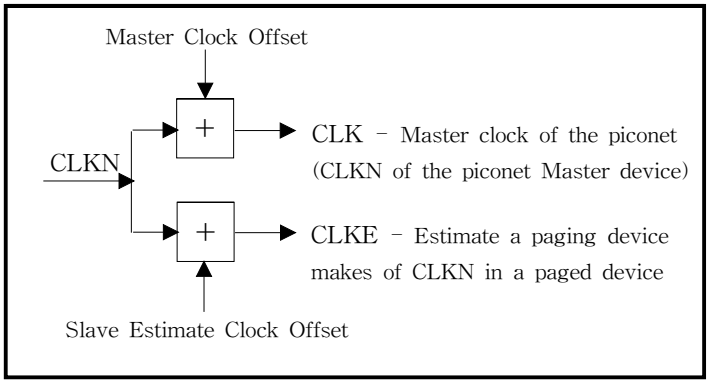


Fig. 13. The conception of bluetooth clock

3. 홉 선택

블루투스의 주파수는 ISM 대역으로, 주파수 호핑 방식의 스펙트럼 확산 방식을 사용한다. 채널은 79개의 RF 채널을 통한 의사 랜덤 호핑 시퀀스이고 이는 블루투스 기기의 고유 주소에 의해 결정된다. 또한 이는 625㎲ 타임 슬롯으로 나뉘어지고 TDD(time division duplex)방식을 채택하여 시간에 따라 교대로 송수신을 반복하게 된다.

그림 14는 홉 선택 커널로, X는 세그먼트에서 위상을 결정하고 Y1과 Y2는 전송방향을 선택하게 된다. 또한 A~D는 세그먼트내의 순서를, E와 F는 호핑 주파수를 결정하고 커널은 모든 짝수 호핑 주파수 다음에 홀수 호핑 주파수를 작성한다.

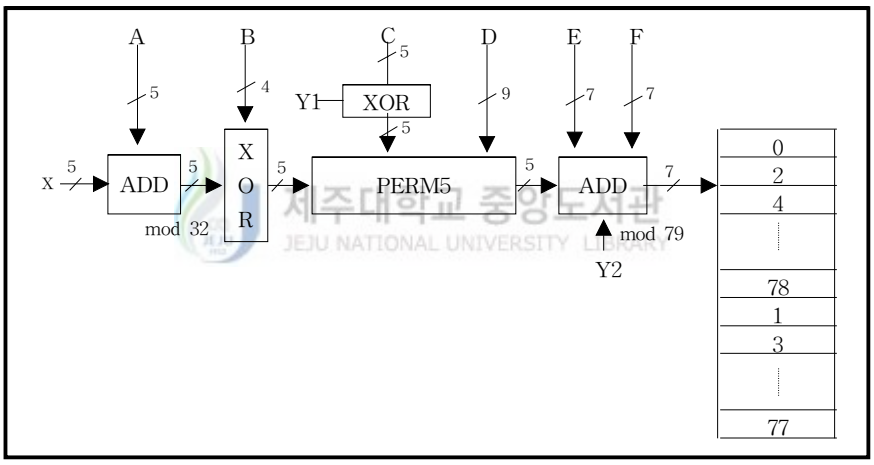


Fig. 14. Block diagram of hop selection kernel

첫 번째 가산기(ADD)는 세그먼트 내의 위상만을 바꾸는데 이 출력은 배타적 논리합(XOR)에서 B와 연산을 수행하게 된다. 배타적 논리합의 출력은 순열연산(PERM5)을 수행하고 이는 두 번째 가산기(ADD)로 입력된다. 두 번째 가산기는 세그먼트의 다른 호핑 주파수를 결정하고 이 출력은 79 나머지 연산을 수행하여 79 레지스터의 뱅크에 주소로 지정된다. 레지스터는 0에서 78의 호핑 주파수에 해당하는 합성 코드 워드를 갖고 상위에는 짝수 호핑 주파수로 구성되고 하위에는 홀수 호핑 주파수로 구성된다.

4. 보안

무선통신은 유선통신보다 도청될 가능성이 크기 때문에 보안은 무선통신시스템에서 중요한 문제점이라 할 수 있다. 그래서 짧은 거리에서 피어 투 피어(peer-to-peer) 통신을 하는 블루투스 규격에서는 사용자 보호와 정보 비밀유지를 위해 어플리케이션 계층과 링크계층 모두에서 보안이 이루어진다.

(1) 키 운영

블루투스에서 디바이스간 인증과 암호화를 위해 사용되는 링크키에는 단위키(unit key), 조합키(combination key), 마스터키(master key), 그리고 초기키(initialization key)가 있다. 링크키는 SAFER+ 알고리즘의 “E” 구현을 통해 생성된 128bits의 값으로, 반영구적으로 쓰이거나 일시적으로 사용된다. 그리고 현재의 링크키로부터 암호화 단계에서 사용되는 암호화키(encryption key)가 유도된다.

(2) 인증

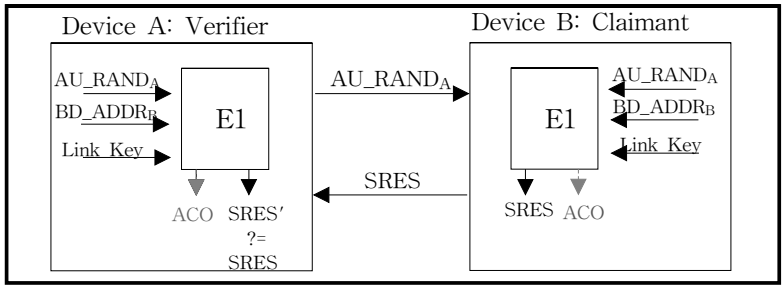


Fig. 15. Challenge-response for the bluetooth

블루투스는 그림 15와 같은 요구/응답 구조의 인증 방식을 사용한다. 인증에서는 상대편이 공유되는 비밀 키를 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. 인증 과정에서 생성되는 ACO(authenticated ciphering offset) 값은 양쪽 디바이스에 저장되고 나중에 암호화키를 만드는데 사용된다.

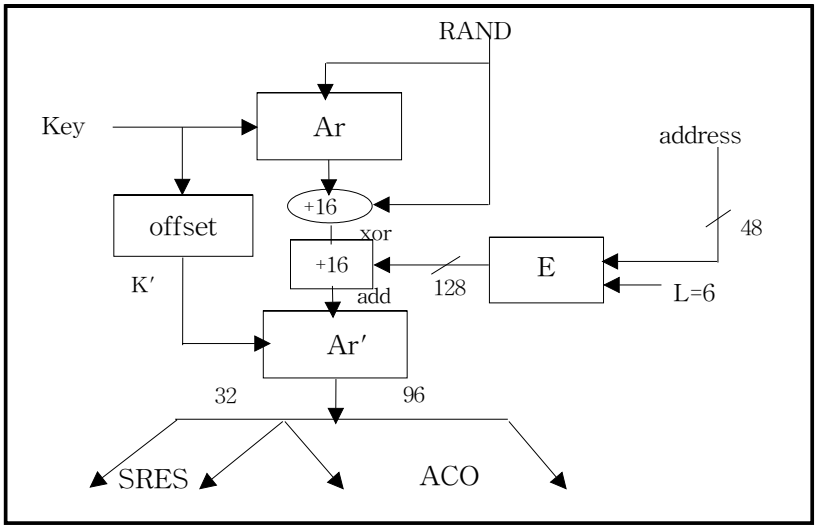


Fig. 16. Flow of data for the computation of E_1

인증을 위한 함수 E1은 그림 16과 같이 SAFER+라고 부르는 Ar과 Ar의 변형 형태인 Ar′으로 구성되어 있다. 이 E1 함수는 링크키, RAND(random number), 그리고 디바이스 주소 값이 입력으로 들어가서 최종적으로 SRES(signed response)와 ACO가 생성 된다.

(3) 암호화

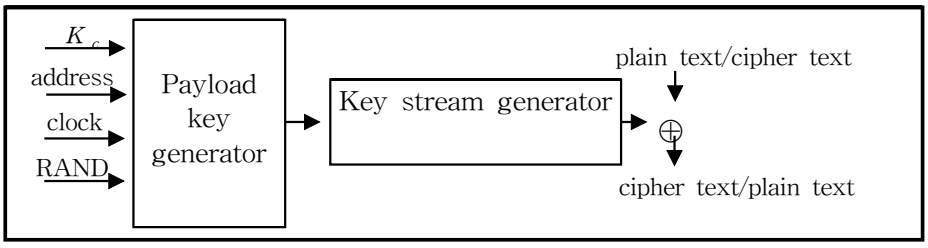


Fig. 17. Stream ciphering for bluetooth with E_0

블루투스에서의 사용자 정보는 패킷의 페이로드 부분을 암호화함으로서 보호될 수 있다. 그림 17에 나타낸 것처럼 블루투스 정보의 암호화는 모든 정보를 재 동기화하는 열 암호화기, E_0에 의해 수행된다. 암호화 시스템은 세 단계로 구성되는데, 첫 번째 단계에서는 초기화가 이루어지고, 두 번째 단계에서 키 열 비트(key stream bit)가 생성되며, 세 번째 단계에서는 전송하고자 하는 데이터가 키 열 비트와 배타적 논리합 연산을 통해 암호화된다. E_0 알고리즘에는 마스터의 48bits 블루투스 기기의 고유 주소, 26bits 마스터 클럭(CLK_{26-1}), 128bits 암호화 키(K_C), 마스터에서 생성되어 동일 피코넷의 슬래이브들에게 전달되는 128bits RAND가 초기 값으로 입력된다.

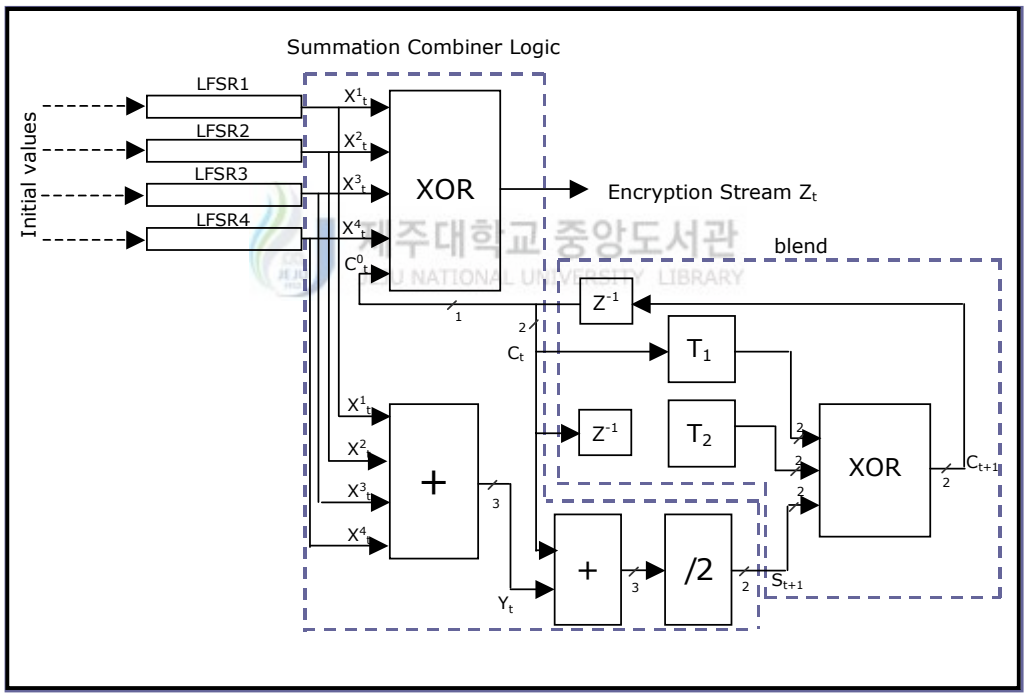
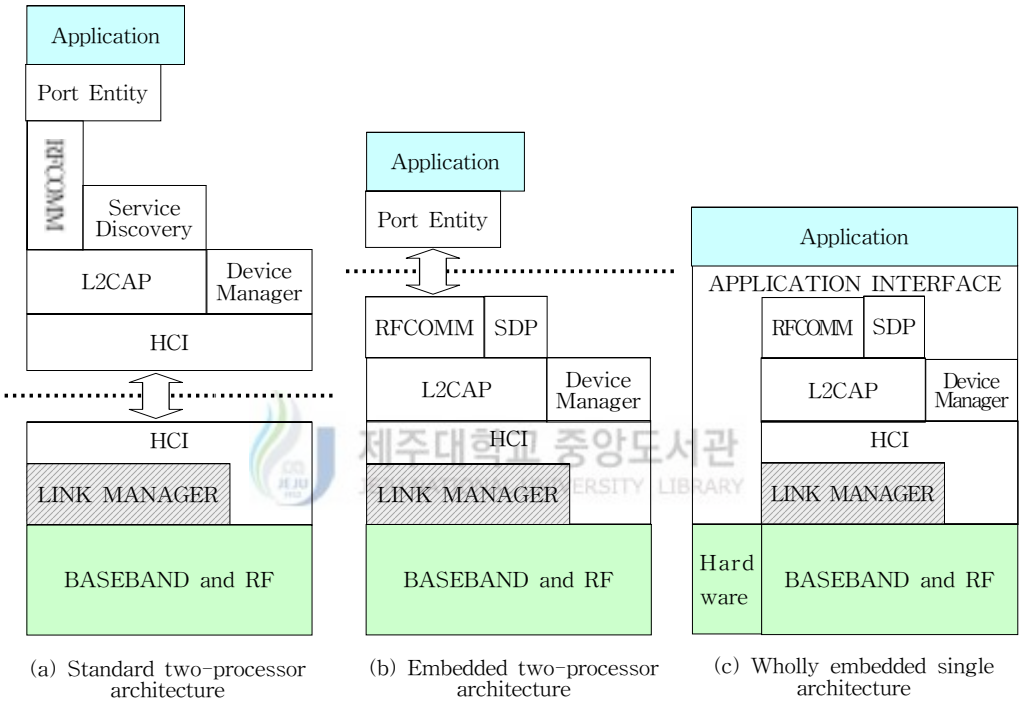


Fig. 18. Concept of the encryption engine

암호화 알고리즘의 내부 구조는 그림 18과 같고 각각의 LFSR 레지스터 길이는 $L_1 = 25$bits, $L_2 = 31$bits, $L_3 = 33$bits, $L_4 = 39$bits이며 전체 레지스터의 길이는 128bits이다.

Ⅲ. 블루투스 베이스밴드 단일칩 설계 구조 및 방법

1. 블루투스 기술의 구현



(a) Standard two-processor architecture (b) Embedded two-processor architecture (c) Wholly embedded single architecture

Fig. 19. Possible bluetooth architectures

블루투스 스택은 일반적으로 HCI(host controller interface)를 기준으로 프로토콜을 배분하지만 호스트 종류에 따라 그림 19와 같이 세 종류로 분류된다. (a)가 가장 일반적인 구조이나 이는 호스트에 걸리는 작업 로드가 크고 많은 리소스를 필요로 한다. 그리고 최근 임베디드 시스템과 SoC 산업이 두드러지면서 단일칩 솔루션이 더욱 중요해지고 있기 때문에 별도의 호스트가 존재하지 않는 (c) 구조로 블루투스를 구현하는 것이 적합하다고 할 수 있다. 이러한 이유로 베이스밴드의 구현이 하드웨어만으로 국한되지 않고 펌웨어로도 가능해지게 되었고, 이를 장착할 CPU

의 역할도 더욱 중요해졌다. 본 논문에서는 EISC방식의 SE3208 코어를 사용하였는데, 표 1을 살펴보면 EISC가 다른 프로세서에 비해 프로그램 사이즈가 작고 임베디드 시스템에 적합하다는 것을 알 수 있다.(http://www.microvision.co.kr)

Table 3. Feature of processors

	CISC	*RISC*	*EISC*
주요 특징	for high-end computers	32bits 고정된 코드	Post-PC Devices Scalable = 16, 32, 64bits Solutions
코드밀도	복잡	간단	간단
상대적 프로그램 사이즈(%)	120-140	140-220	100
성능	낮음	높음	높음
임베디드	적합하지 않음	높음	이상적

블루투스 베이스밴드를 구현하여 패킷이 제대로 생성되는지의 여부를 확인하기 위하여 LM과 L2CAP를 구현하는 것이 필요하다. 따라서 패킷을 생성할 수 있는 최소의 LM과 L2CAP를 구현하여 블루투스 베이스밴드의 동작 상태를 테스트한다.

LM은 링크 설정, 보안, 제어를 담당하는 곳으로, 주된 역할은 LMP(link manager protocol) 메시지를 이용한 통신이다. 링크, 연결 상태(park, sniff, hold) 그리고 보안 등의 설정으로 RF 및 링크를 직접 제어하는 곳은 베이스밴드이지만, 명령 및 제어를 통해 이 설정을 리모트 디바이스에게 수행하도록 하고 리모트 디바이스의 상태에 대한 정보를 얻는데 필요한 별도의 통신 수단은 LMP 메시지이다. LMP 메시지는 두 디바이스의 LM 사이에서 전달되며 상위계층으로 전달되지는 않고 페이로드를 통해 전달된다.(Bluetooth SIG, 2000)

L2CAP는 데이터 링크계층에 해당하는 것으로, ACL 링크만이 지원되고 상위계층 데이터와 하위계층 데이터의 통합・분할・조립을 담당한다. 이 프로토콜의 주 기능은 데이터의 오류 감지 및 흐름 제어이고, 그 밖의 기능은 프레임 동기화, 주

소 식별, 링크 관리 등이다. 이러한 L2CAP 스택의 기능을 처리하기 위해서 두 기기 사이에는 하나의 ACL 링크만이 존재하며 이는 LMP를 사용하여 셋업되고 베이스밴드는 항상 전 이중 통신 채널을 제공한다는 조건이 선행되어야 한다.(Bluetooth SIG, 2000)

Table 4. Logical channel L_CH field contents

L_CH code	Logical Channel	Information
00	NA	undefined
01	UA/I	Continuing L2CAP message
10	UA/I	Start L2CAP message
11	LM	LMP message

베이스밴드의 논리 채널은 표 2와 같이 페이로드 헤더의 L_CH(logical channel) 코드에 따라서 설정된다.

2. 단일칩 내부 구조

본 논문에서 설계하고자 하는 블루투스 베이스밴드는 그림 20에서 보는 것처럼 32bits SE3208 코어, 블루투스 베이스밴드, 인터럽트, 메모리와 UART (universal asynchronous receiver/transmitter) 블록으로 구성된다.

그림 20의 내부 메모리는 논리 채널을 설정·관리하는 LM과 데이터를 관리하는 L2CAP를 비롯한 상위계층, 펌웨어로 설계될 베이스밴드 블록이 저장된다. 본 논문에서는 이 내부 메모리를 칩 외부에 두어 구현하였지만 향후에는 칩 내부에 구현할 것이다.

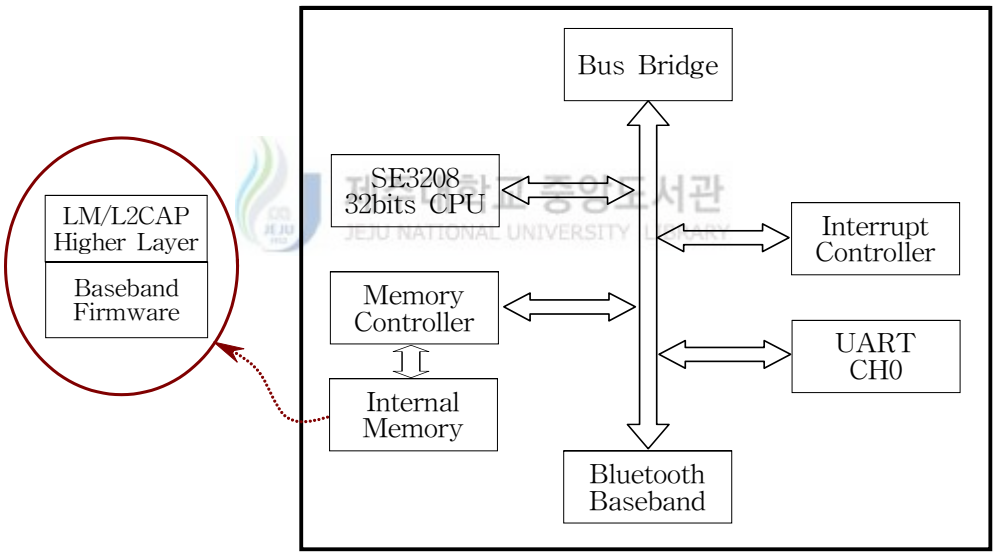


Fig. 20. Internal block diagram

(1) 블루투스 레지스터 설정

상위계층과 하위계층 사이의 데이터 처리와 베이스밴드 동작에 필요한 정보를 얻는 등 블루투스의 효율적인 동작을 위해서 레지스터를 정의하는 것은 중요하다. 따라서 블루투스 베이스밴드의 동작을 사전에 정확히 파악하여 사용 목적과 용도

에 맞게 레지스터를 분류하여 구현해야 설계시간을 단축할 수 있고 블루투스 칩을 제어하는 데도 훨씬 편리해진다.

본 논문에서 사용하는 SE3208 코어는 32bits의 주소를 가지며 총 4Gbytes의 주소 영역을 직접 접근 할 수 있다. 표 3은 구현하고자 하는 단일칩의 레지스터 종류와 메모리 영역을 나타낸 것이다.

Table 5. Register memory map

Offset Address	Block	Remark
0180 0000h	Local Memory Controller	
0180 0400h	Interrupt Controller	
0180 0800h	UART	1 Channel
0180 0C00h	System	
Bluetooth Baseband		
0181 0000h	Bluetooth Address	
0181 0400h	Bluetooth Clock	
0181 0800h	Baseband Controller	
0181 0C00h	Slave Channel Configuration	
0181 1000h	SCO Channel Configuration	
0181 1400h	Time	
0181 1800h	Baseband Interrupt	
0181 1C00h	Data	
0181 2000h	Baseband Register	

로컬 메모리 컨트롤러는 외부 메모리와 내부 메모리에 대한 제어신호를 형성하고 CPU와 메모리 디바이스를 인터페이스 해주는 역할을 한다. 만일 32bits 프로세서와 다른 크기의 데이터 버스를 사용하는 블록이 있다면, 이를 인터페이스하기 위해 CPU에서 출력되는 4bits의 바이트 인에이블 신호를 검사하여 메모리 접근 횟수를 결정하고, 그에 맞게 1byte 단위로 데이터를 교환하게 된다. 예를 들어 바이트 인에이블이 4'b0000인 경우는 4회(4bytes), 4'b1100/4'b0011인 경우는 2회(2bytes), 4'b1110 /4'b1101/4'b1011/4'b0111인 경우는 1회(1byte) 접근한다. 그리고 이 바이트 인에이블의 값은 코어 주소 최하위 2bits에 따라 결정된다.

인터럽트 컨트롤러는 내부 패리패럴과 외부 포트에서 발생되는 인터럽트 요청들

을 해당 채널의 마스크 상태와 우선순위에 따라 선택하여 CPU에 인터럽트를 요청하고 해당 인터럽트 채널의 벡터 주소를 CPU로 보내는 역할을 수행한다.

UART는 RS-232C(recommended standard 232 revision C) 인터페이스의 일반적인 PC(personal computer) 및 I/O 디바이스와의 직렬통신을 위한 기본적인 UART 기능을 가지며, 1 채널 전이중 통신방식 기능을 지원한다.

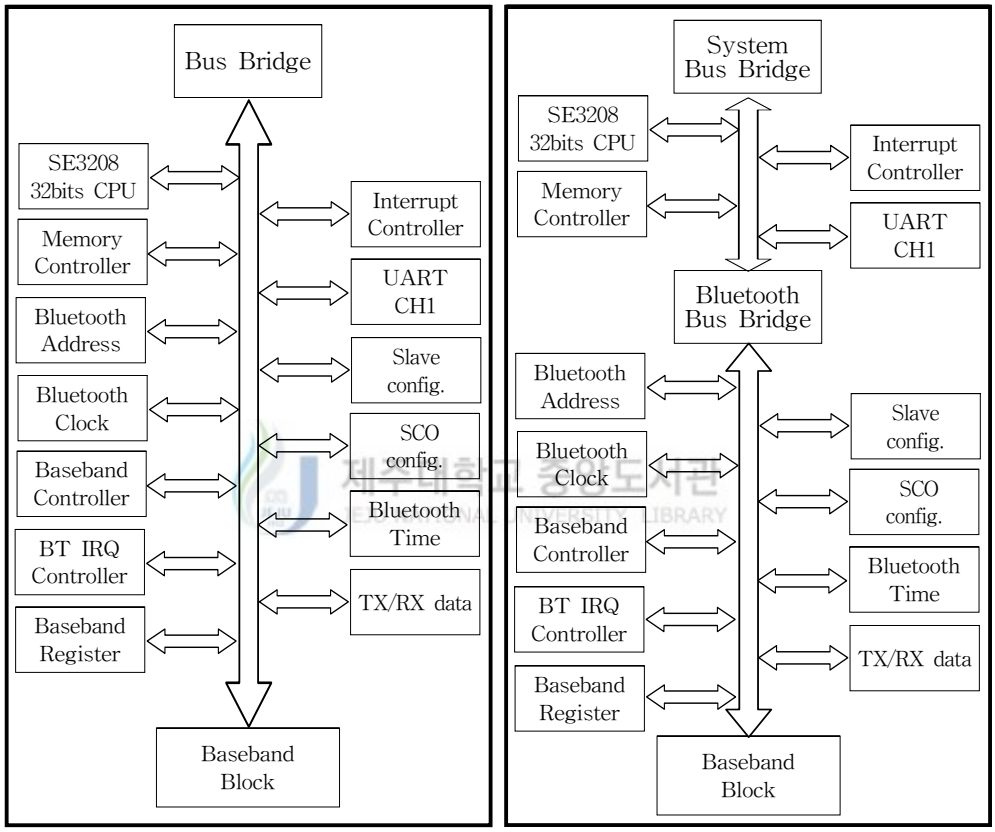
시스템은 생산된 칩에 대한 고유한 정보를 가지고 있으며, 프로그램으로 칩을 구분할 수 있는 확실한 방법을 제공한다.

블루투스 베이스밴드는 그 역할과 종류에 따라 분류된 9개의 레지스터와 그림 1의 데이터 패스 단으로 구성된다. 블루투스 주소 레지스터는 48bits의 블루투스 기기의 고유 주소를 저장하는 곳으로, 자신과 상대방 디바이스의 주소를 모두 포함한다. 블루투스 클럭 레지스터는 자신과 상대방 디바이스의 CLKN, CLK, CLKE의 값과 마스터와 슬레이브 슬롯 사이의 차이를 포함하는 영역이다. 그리고 베이스밴드 컨트롤러는 베이스밴드 동작에서 필요한 여러 가지 정보와 파라미터를 포함한다. 슬레이브 채널 구성은 디바이스가 마스터로 동작할 경우 7개의 슬레이브 중 원하는 슬레이브에 접근 할 수 있게 해주는 정보를 포함하고, SCO 채널 구성은 SCO 채널을 사용할 경우에 필요한 정보를 포함한다. 시간 레지스터는 블루투스 동작에서 사용되는 모든 시간을 관리하는 곳이고, 베이스밴드 인터럽트는 베이스밴드 동작에 필요한 인터럽트를 요청하고 수행 할 수 있는 제어를 담당한다. 데이터는 블루투스 기기간에 교환되는 데이터를 말하는 것으로, 그 값은 ACL 패킷의 페이로드에 해당한다. 마지막으로 베이스밴드 레지스터는 펌웨어로 구현할 블록의 결과를 저장하고 테스트 결과 값을 보기 위해 사용한다.

(2) 버스 브릿지 연결 방식

칩 내부를 살펴보면 CPU와 패리패럴을 중개해 주는 버스 브릿지가 존재하는데, 이는 내부 제어 신호에 따라 CPU가 여러 패리패럴 중 하나를 선택하여 두 블록간의 데이터 교환을 원활히 하도록 도와준다. 그림 21은 이러한 내부 버스 브릿지 구조의 두 가지 모델을 보여준 것으로, (a)는 블루투스 베이스밴드와 SE3208 코어를 비롯한 그 밖의 패리패럴 블록을 하나의 버스 브릿지에 연결하여 제어하는 방

식이고, (b)는 블루투스 버스 브릿지를 시스템 버스 브릿지에서 분리하여 두 개의 버스 브릿지로 제어하는 방식이다. 즉, (a)의 버스 브릿지가 (b)에서는 그 역할이 시스템 버스 브릿지와 블루투스 버스 브릿지로 분리되어 있는 형태이다. 이때 패리패럴과 CPU간에 버스 브릿지를 통해 교환되는 신호는 읽기(read)/쓰기(write) 신호와 칩 선택(chip select) 신호, 준비(ready) 신호, 그리고 주소와 데이터이다.



(a) Single bus bridge (b) Double bus bridge

Fig. 21. Two models of internal architecture

3. 동작 원리

블루투스 베이스밴드를 CPU와 연결하여 단일칩으로 구현하기 위해서는 사전에 칩 내부에서 전달되는 신호나 타이밍을 충분히 검토해야한다. 3절에서는 베이스밴드와 CPU간의 데이터 검출 방식과 읽기/쓰기 신호의 타이밍을 설명하고 이를 바탕으로 칩 내부의 신호 흐름에 대해 기술한다.

(1) 데이터 검출 방식

CPU가 비동기인 블루투스를 효율적으로 제어하려면 두 코어 사이에 데이터 송수신을 감지하는 방식이 중요하다. 이때 사용되는 방식은 일반적으로 두 가지가 있는데, 그 하나는 프로그래머가 명령어를 사용하여 입력 핀 또는 값을 계속 읽어서 변화를 알아내는 것이고 다른 하나는 CPU 자체가 하드웨어적으로 그 변화를 체크하여 변화 시 그에 맞게 대처하는 것이다. 전자를 폴링(polling) 기법이라고 하고 후자를 인터럽트(interrupt) 기법이라고 한다. 폴링은 모든 경우의 입력 또는 값의 변화에 대응하여 처리가 가능하지만 인터럽트는 하드웨어적으로 지원되는 몇 개의 입력 또는 값의 변화에만 대응 처리가 가능하다. 즉, 하드웨어 인터럽트는 CPU가 만들어질 때부터 결정이 된다. 일반적인 처리 속도는 인터럽트이 폴링방식보다 H/W로 구현되었기 때문에 더 빠르다. 따라서 사용자는 블록과 데이터의 사용 빈도수와 특성을 고려하여 두 방식을 병행해서 사용해야한다.

본 논문에서는 베이스밴드 제어를 위해 폴링 방식을 사용하여 상위계층으로부터 명령어가 내려오면 그에 맞게 동작하도록 하고, 다른 기기로부터 수신된 정보를 상위계층으로 전달하기 위해 인터럽트 방식을 이용한다.

(2) 동작 원리

CPU와 블루투스 베이스밴드는 서로 다른 클럭으로 동작하기 때문에 데이터를 주고받기 위해서는 두 블록이 공유할 수 있는 별도의 영역이 있어야 한다. 따라서 앞 절에서 설명 된 블루투스 레지스터 영역을 그림 22와 같이 CPU와 블루투스 베이스밴드 사이에 구현하여 이를 매개로 두 블록의 정보를 교환하고 공유하도록 한다. 즉, 블루투스 레지스터를 경계로 CPU와 베이스밴드는 구동 클럭과 동작이 분리되게 되고 인터럽트와 폴링방식으로 서로 데이터를 주고받게 되는 것이다.

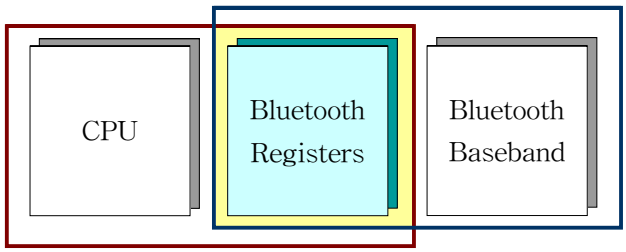
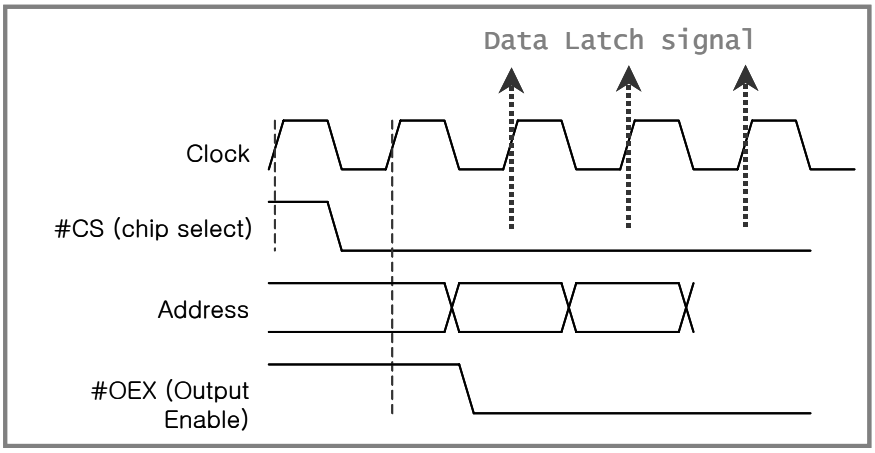
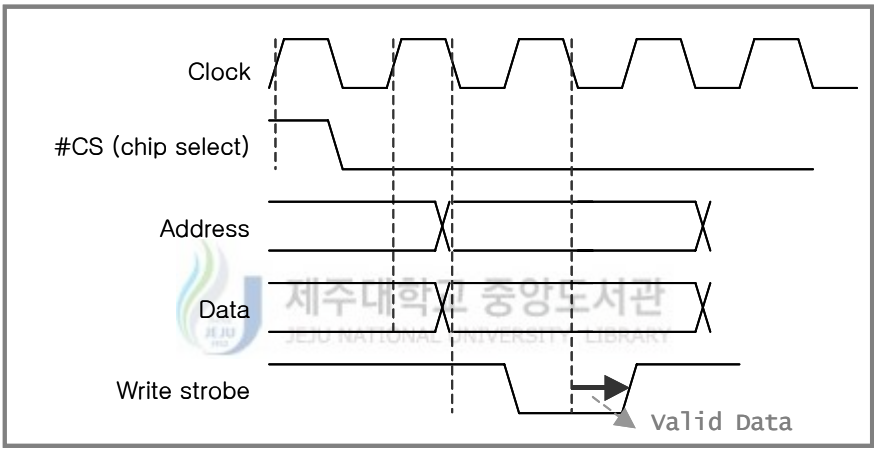


Fig. 22. Bluetooth core architecture

블루투스 레지스터와 CPU간 데이터를 안정되게 주고받기 위해서는 내부 메모리에 접근할 수 있는 신호를 정확하게 이해하여 데이터가 교환되는 시간을 놓치지 않도록 해야 한다. 데이터 읽기/쓰기를 실행하기 위하여 메모리에 접근할 수 있는 제어 신호의 타이밍도를 그림 23에 나타내었다. (a)는 읽기에 대한 타이밍도를 나타낸 것으로, 메모리 접근 주소와 출력 인에이블 신호는 칩 내부의 지연을 고려하여 일정 시간이 흐른 후에 유효한 출력 값이 나타난다. (b)는 쓰기에 대한 타이밍도를 나타낸 것으로, 메모리 접근 주소와 데이터 출력은 칩 내부 지연을 고려하여 일정 시간이 흐른 후에 유효한 출력 값이 나타나게 되고 쓰기 스트로브 신호는 메모리 제어 클럭의 폴링 에지(falling edge)에서 출력되게 된다.(ADC, 2002)



(a) Read timing diagram



(b) Write timing diagram

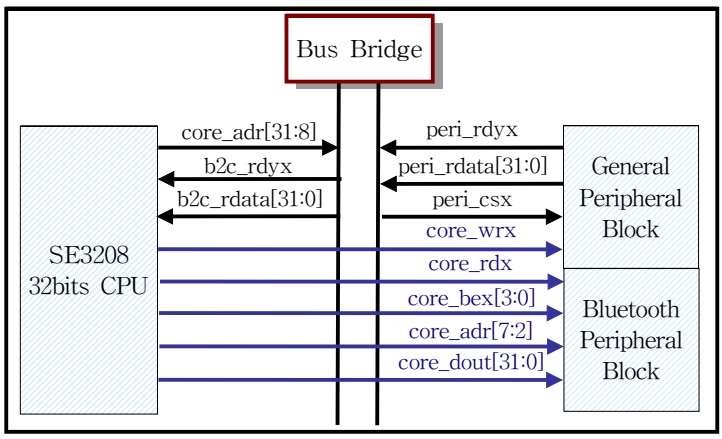
Fig. 23. External memory/internal memory SRAM timing diagram

(3) 칩 내부의 신호 흐름

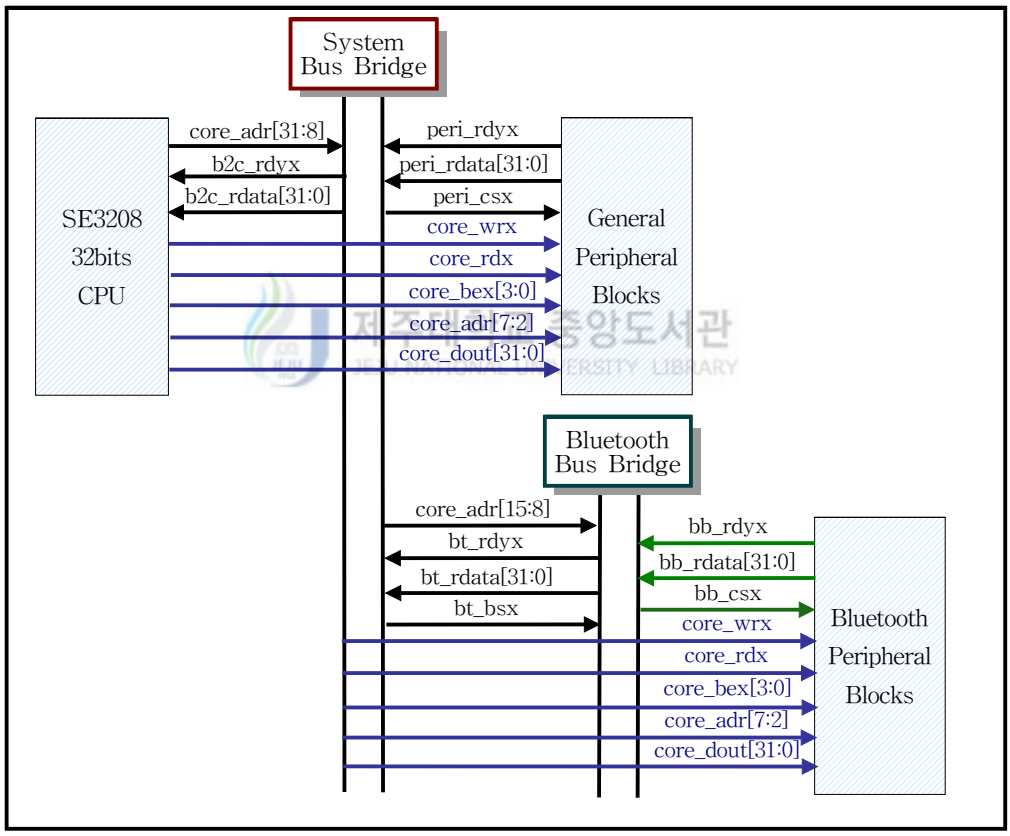
그림 24는 블루투스 칩 내부에서 교환되는 상세 신호를 나타낸 것이다. (a)인 경우, CPU에서 상위 주소 값(core_adr[31:8])이 출력되면 이를 버스 브릿지가 받아들여 칩 선택 신호(peri_csx)가 발생되고 해당 패리패럴로 입력된다. CPU에서 출력되는 읽기(core_rdx)/쓰기(core_wrx) 신호는 모든 패리패럴에 공통으로 입력되지만, 칩 선택 신호가 발생된 블록에서만 그 값이 유효하여 선택된 패리패럴에서만

데이터 교환이 이루어진다. 이때 해당 블록은 CPU로부터 입력 받은 하위 주소 값(core_adr[7:2])에 따라 각각 용도가 다른 데이터를 교환한다. 쓰기인 경우는 core_dout[31:0]을 통해 CPU에서 패리패럴로 이동하고, 읽기인 경우는 패리패럴에서 출력된 데이터(peri_rdata[31:0])가 버스 브릿지를 거쳐 b2c_rdata[31:0]를 통해 CPU로 이동하게 된다. 여기서 데이터 버스의 크기는 앞서 설명한 바이트 인에이블인 core_bex[3:0]의 값에 따라 결정된다. 그리고 읽기/쓰기가 모두 성공적으로 수행되었는지를 판단하기 위한 준비신호가 패리패럴에서 버스 브릿지(peri_rdyx)를 통해 CPU(b2c_rdyx)로 이동한다.

(b)인 경우, 시스템 버스 브릿지가 CPU에서 출력되는 core_adr[31:16]값에 따라 블루투스 버스 브릿지를 선택하는 신호(bt_bsx)를 발생하면, 블루투스 버스 브릿지는 다시 core_adr[15:8]에 따라 원하는 블루투스 패리패럴을 선택하는 칩 선택 신호를 발생한다. 이때 읽기/쓰기 신호에 따라 데이터 교환이 이루어지고, 이 과정에서 발생되는 준비신호(bb_rdyx)와 읽기 데이터(bb_rdata[31:0])가 시스템 버스 브릿지를 거쳐 b2c_rdyx와 b2c_rdata[31:0]를 통해 CPU로 입력된다. 그리고 일반 패리패럴 블록의 데이터 교환은 (a)와 같다.



(a) Single bus bridge



(b) Double bus bridge

Fig. 24. Exchanged signal between SE3208 and peripheral through bus bridge

4. 설계 IP 테스트 환경 및 방법

(1) 테스트 보드 및 환경

본 논문에서는 블루투스 베이스밴드를 설계하고 이 IP(intellectual property)의 원활한 동작을 확인하기 위해 그림 25와 그림 27의 테스트 보드를 제작하였다.

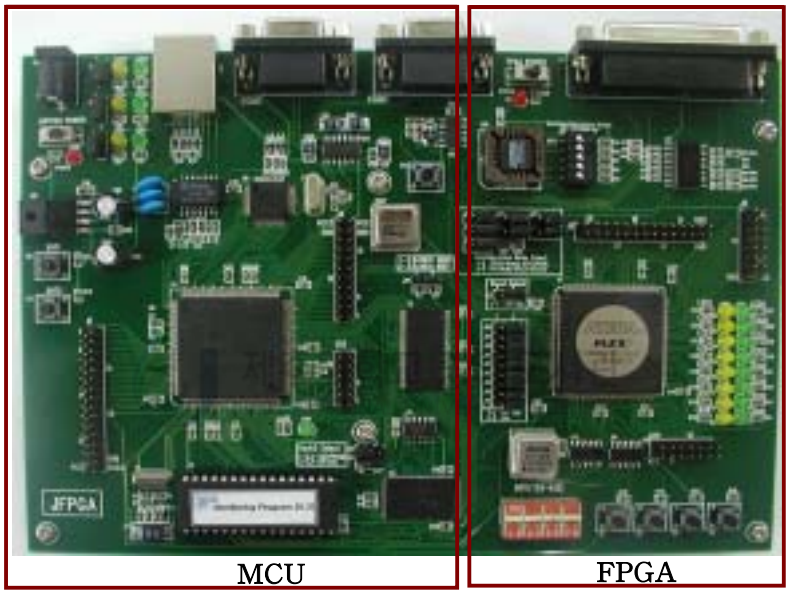


Fig. 25. Test board for baseband block design

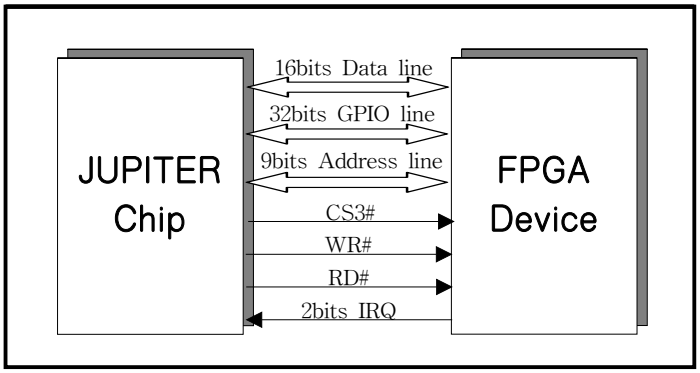


Fig. 26. Interface architecture

그림 25는 베이스밴드의 블록별 테스트를 위한 보드이고, 이 보드에서 MCU (micro controller unit)와 FPGA 사이의 인터페이스는 그림 26과 같이 CPU 내장 단일칩 설계를 고려하여 제작하였다. 이 보드에서 사용한 MCU는 SE3208 코어를 내장한 (주)에이디칩스의 Jupiter이고, FPGA는 Altera사의 EPF10K30RC208-3 (30KGates, 137개의 I/O)이다.

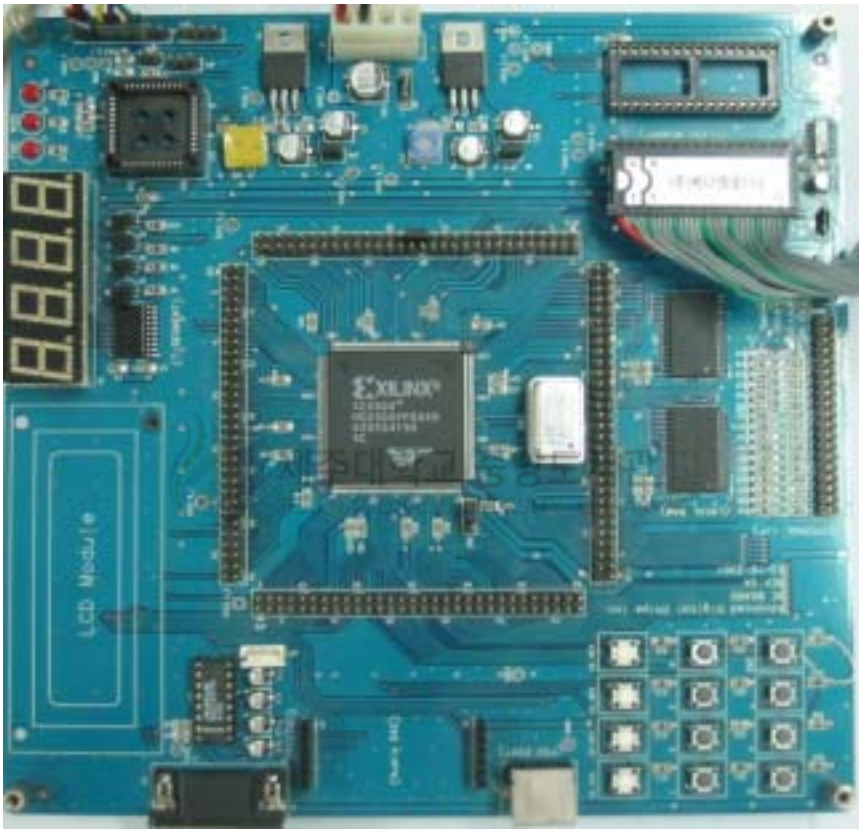
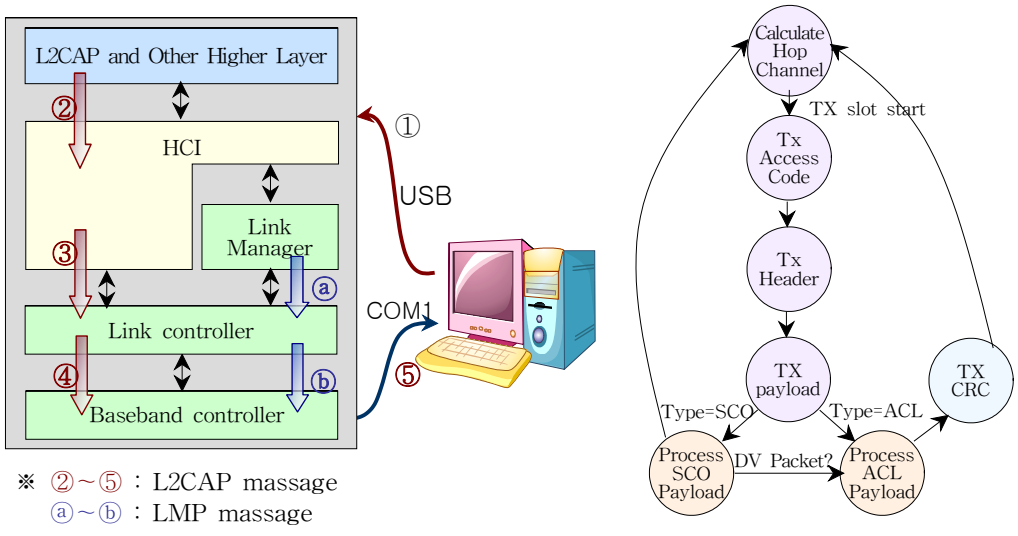


Fig. 27. Test board for bluetooth core design

그림 27은 버스 브릿지 방식을 달리하여 설계한 블루투스 단일칩의 동작상태를 검증하기 위한 것이다. 이 보드는 CPU를 내장한 단일칩의 블루투스 개발에 적합하도록 SDRAM(synchronous dynamic random access memory), 플래시메모리, 그리고 UART 등의 외부 회로로 구성되어있고, FPGA는 Xilinx사의 XCV600HQ240C(660KGates, 166개의 I/O)이다.

(2) 칩 검증 과정



(a) Data transfer process (b) LC transmit state diagram

Fig. 28. Packet generation process and verification environment

그림 28은 패킷 생성과정을 비롯하여 이를 위한 데이터 전달 과정과 본 논문에서의 테스트 과정을 나타낸 것이다. (a)는 상위계층에서 전송된 데이터가 HCI를 통해 하위계층에 도달하여 ACL 패킷을 생성하기까지의 데이터 전달 과정을 보여준다. 패킷 생성 과정에 앞서 과정 ①과 같이 USB(universal serial bus)를 통해 블루투스 동작에 필요한 정보와 데이터, 일부의 베이스밴드 블록 등을 포함한 블루투스 펌웨어를 메모리에 저장한다. L2CAP는 데이터를 블루투스 스택의 상위계층과 어플리케이션으로부터 전송받아 HCI를 통하여 스택의 하위계층으로 보내는데, ②~④의 과정이 이에 해당한다. 이렇게 하위계층으로 전달된 데이터는 (b)와 같은 과정으로 패킷이 생성되고 본 논문에서는 이 패킷을 ⑤의 과정을 통해 하이퍼터미널에서 확인하였다. ⓐ, ⓑ의 데이터 전달 과정은 블루투스가 패킷전송을 위해 링크를 설정하고 관리할 때 이루어진다.

```
#define __LOC_BD_ADDR0                  0X01810000
#define __SET_LOC_BD_ADDR0(DATA)        pokel((void*)__LOC_BD_ADDR0, (long)DATA);

long LOC_LAP = 0x9E8B33;
__SET_LOC_BD_ADDR0(LOC_LAP);
```

(a) Data transmission by CPU

```
always @ (negedge rstx or posedge clk )
    if (~rstx)  ready <= 1'b1;
    else        ready <= csx | ( regwx & regrx );

always @ (negedge rstx or posedge clk )
    if (~rstx)  ready1 <= 1'b1;
    else        ready1 <= ready;

assign adr_rdyx = ready | ( ~ready1 );

assign wr_rdyx = adr_rdyx | regwx;

wire predec = ~(|adr[7:5]);

wire r00cs = predec & ~adr[4] & ~adr[3] & ~adr[2];

wire r00wr2 = r00cs & ~wr_rdyx & ~bex[2];
wire r00wr1 = r00cs & ~wr_rdyx & ~bex[1];
wire r00wr0 = r00cs & ~wr_rdyx & ~bex[0];

always @(negedge rstx or posedge clk)
  if (~rstx)         bd_loc_lap[23:16] = 8'b0;
  else if (r00wr2) bd_loc_lap[23:16] = din[23:16];

always @(negedge rstx or posedge clk)
  if (~rstx)         bd_loc_lap[15:8] = 8'b0;
  else if (r00wr1) bd_loc_lap[15:8] = din[15:8];

always @(negedge rstx or posedge clk)
  if (~rstx)         bd_loc_lap[7:0] = 8'b0;
  else if (r00wr0) bd_loc_lap[7:0] = din[7:0];
```


(b) Data reception by baseband

Fig. 29. The example of data transfer from CPU to baseband

그림 29는 패킷 생성에 필요한 정보를 CPU에서 블루투스 레지스터로 쓸 때의 코딩 예를 보여준 것이다. 즉, __SET_LOC_BD_ADDR0가 실행되면 버스 브릿지에서 'core_adr[31:8]=0x018100'에 맞는 칩 선택 신호를 출력하게 되고 베이스밴드 레지스터에서 이 신호와 쓰기 신호를 조합하여 LAP를 CPU로부터 받아들이게 된다.

```
#define __LOC_BD_ADDR0                  0X01810000
#define __GET_LOC_BD_ADDR0(DATA)    DATA = peekl((void*)__LOC_BD_ADDR0);

__GET_LOC_BD_ADDR0(D[0]);
```

(a) Data reception by CPU

```
always @ (negedge rstx or posedge clk )
    if (~rstx)  ready <= 1'b1;
    else        ready <= csx | ( regwx & regrx );

always @ (negedge rstx or posedge clk )
    if (~rstx)  ready1 <= 1'b1;
    else        ready1 <= ready;

assign adr_rdyx = ready | ( ~ready1 );

assign rd_rdyx = adr_rdyx | regrx;

wire predec = ~(|adr[7:5]);

wire r00cs = predec & ~adr[4] & ~adr[3] & ~adr[2];

wire loc_bd_adr0_rd = r00cs & ~rd_rdyx;

always @(loc_bd_adr0_rd)
  case (vdd)        // sysnopsys parallel_case full_case
    loc_bd_adr0_rd : mregrd = {8'b0, bd_loc_lap};
    default                : mregrd = 32'b0;
  endcase
```

(b) Data transmission by FPGA

Fig. 30. The example of data transfer from baseband to CPU

그림 29와 같은 과정을 통해 블루투스 레지스터에 값이 저장되면 이를 토대로 블루투스 베이스밴드 블록에서는 필요한 정보를 축출하여 그에 맞는 동작을 하게 되고, 이때 얻어진 값은 다시 해당 블루투스 레지스터에 저장된다. 이를 다시 CPU에서 읽어 들여 하이퍼터미널로 그 값을 출력하는데 그 예를 그림 30에 보였다. 즉, __GET_LOC_BD_ADDR0가 실행되면 버스 브릿지에서 'core_adr[31:8]= 0x018100'에 맞는 칩 선택 신호를 출력하게 되고, 베이스밴드 레지스터에서 이 신호와 읽기 신호를 조합하여 CPU로 bd_loc_lap를 보낸다.

Ⅳ. EISC 기반 블루투스 베이스밴드 설계

1. 블루투스 베이스밴드 설계 개요

본 장에서는 베이스밴드의 블록 중 접근 코드, 블루투스 클럭, 홉 시퀀스와 보안을 하드웨어와 펌웨어로 구현하여 블루투스 SIG 스펙의 샘플데이터와 비교하였고, 그 결과를 여러 환경과 요구 조건을 바탕으로 서로 비교·검토함으로써 효율적인 구현 형태를 제시한다. 그리고 이 결과를 토대로 블루투스 베이스밴드를 단일칩으로 설계하는데, 이때 상이한 버스 브릿지, 즉 단일 버스 브릿지와 이중 버스 브릿지를 이용하였고 그 결과를 분석하여 효율적인 버스 브릿지 방식을 제안한다.

3절에서 보여주는 블루투스 단일칩의 패킷 생성은 베이스밴드의 블록별 분할 설계에서 얻은 결과를 토대로, ACL 패킷 중 DM1 패킷과 블루투스 클럭 관리를 구현하였다. DM1 패킷의 페이로드는 그 길이가 0~17bytes이고, 2/3 FEC와 CRC를 통과한다. 베이스밴드의 동작을 위해서 필요한 제어 신호 및 정보의 교환은 표 3에서 보여준 블루투스 레지스터를 통해서 이루어진다. 즉 접근 코드, 패킷 헤더 및 페이로드는 블루투스 주소 레지스터와 베이스밴드 컨트롤 레지스터를 사용하고 블루투스 클럭 관리는 블루투스 클럭 레지스터와 베이스밴드 컨트롤 레지스터를 사용한다. 접근 코드와 블루투스 클럭 관리는 분할구현에서 구현한 것을 그대로 이용하고, 패킷 헤더는 그림 5의 정보를 수집하여 10bits의 정보를 HEC와 비트 랜덤화 회로, 1/3 FEC를 차례로 통과시켜 54bits를 생성하게 된다. 그리고 페이로드는 DM1 패킷이므로 1byte의 페이로드 헤더와 2bytes의 사용자 정보를 조합한 ACL 데이터를 시스템 버퍼를 통해 CRC와 비트 랜덤화 회로, 2/3 FEC를 차례로 통과시켜 60bits를 생성한다.

본 논문에서 하드웨어는 Verilog HDL(hardware description language)을 이용하여 설계하였고 펌웨어는 EISC기반의 C언어를 이용하여 설계하였다.

2. 베이스밴드 블록의 분할 설계 및 검증

(1) 접근 코드의 설계 및 검증

다음은 블루투스 기기의 고유 주소 중 LAP가 '0x9E8B33'인 접근 코드의 구현 결과를 나타내었다.

ⅰ) 발생기(generator)

그림 31과 그림 32는 그림 3의 접근 코드를 하드웨어와 펌웨어로 구현한 결과이다.

Fig, 31. Hardware simulation of access code generator

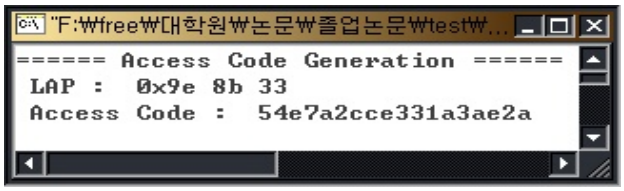


Fig. 32. Firmware simulation of access code generator

ⅱ) 상관기(correlator)

이 블록은 수신된 프레임을 검출하는 것으로 그 동작 원리는 그림 33과 같다.

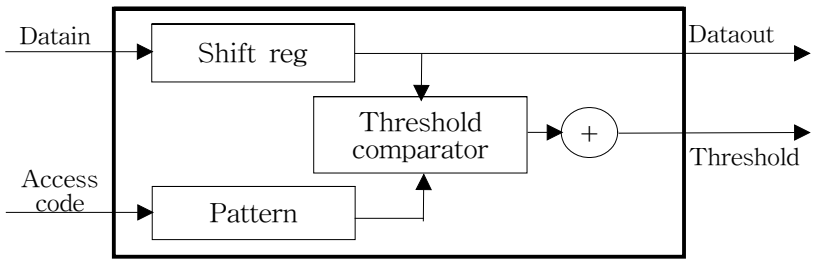


Fig. 33. Block diagram of access code correlator

여기서 접근 코드는 발생기에서 만들어진 값을 사용하고, 문턱 값에 의해 발생되는 트리거 신호로 클럭을 동기화시킨다. 그림 34와 그림 35가 이를 구현한 시뮬레이션 결과이다.

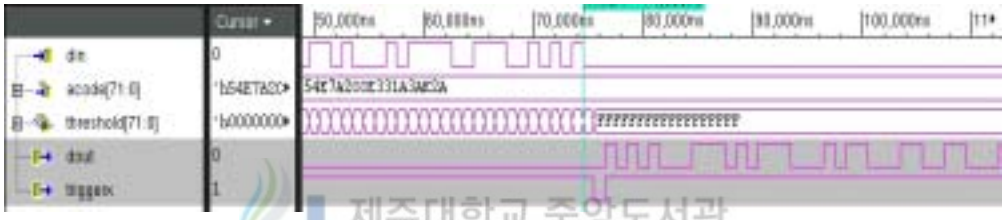


Fig. 34. Hardware simulation of access code correlator

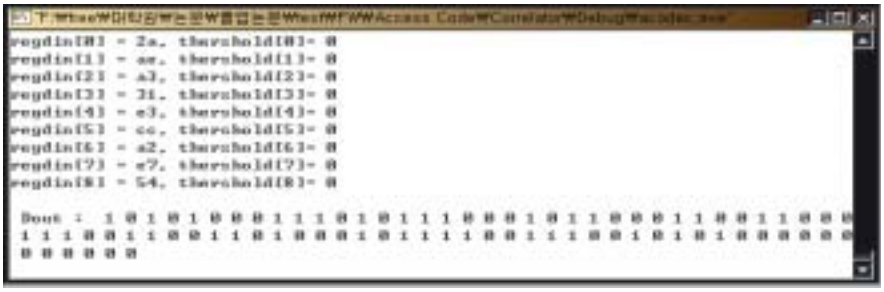


Fig. 35. Firmware simulation of access code correlator

(2) 블루투스 클럭의 설계 및 검증

그림 36과 그림 37은 그림 12와 그림 13의 블루투스 클럭을 하드웨어로 구현한 결과이다. 이 그림에서 ext_trigx 신호를 볼 수 있는데, 이는 접근 코드 상관기에

서 출력되는 신호로 시스템이 수신 프레임의 클럭에 맞출 수 있게 하는 신호이고, ld_clkx는 FHS 패킷을 보내는 디바이스의 CLKN값을 로드 시키는 신호이다.

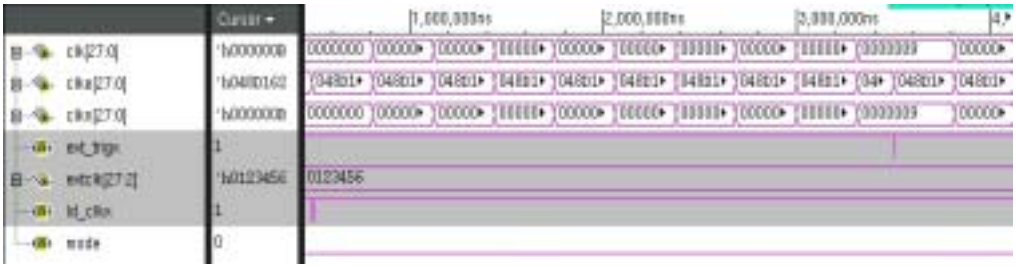


Fig. 36. Hardware simulation of bluetooth clock in master

Fig. 37. Hardware simulation of bluetooth clock in slave

(3) 홉 시퀀스의 설계 및 검증

그림 14의 알고리즘을 이용하여 8가지 상태를 설계하여 검증하였다. 이때 블루투스 기기의 고유 주소 UAP&LAP를 '0x2A96EF25'라 가정하였고, 마스터 호출 응답 상태일 때의 시뮬레이션 결과를 그림 38과 그림 39에 나타내었다.

Fig. 38. Hardware simulation of hop sequence

Fig. 39. Firmware simulation of hop sequence

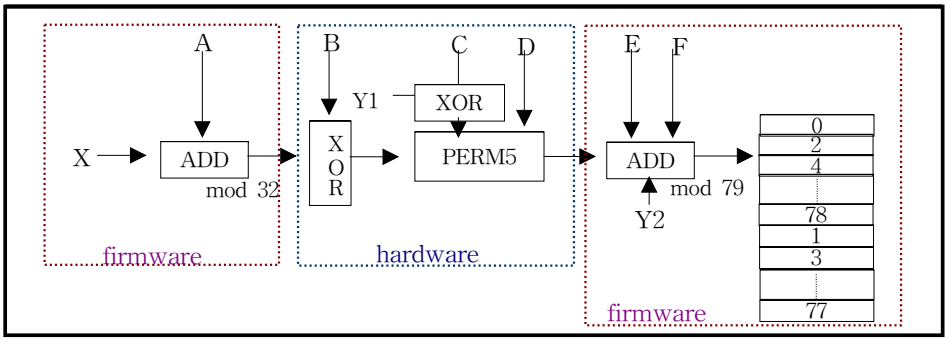


Fig. 40. Divided block diagram of hop selection kernel

그림 40은 그림 14의 알고리즘을 하드웨어와 펌웨어로 구현하기에 적합한 부분으로 분할한 것이다. 즉 두 개의 배타적 논리합과 순열연산 블록을 하드웨어로, 나머지 부분을 펌웨어로 나누어 구현하였고, 이를 테스트한 결과를 그림 41에서 보이는 것과 같이 하이퍼터미널 출력으로 확인하였다.

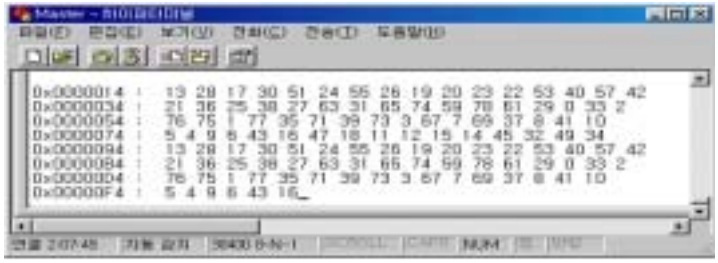


Fig. 41. Simulation of divided hop sequence

(4) 블루투스 보안의 설계 및 검증

ⅰ) 인증(E1)

그림 42와 그림 43은 rand가 '0xBC3F30689647C8D7C5A03CA80A91ECEB', address가 '0x7CA89B233C2D', 그리고 key가 '0x159DD9F43FC3D328EFBA0CD8A861FA57' 일 때 그림 16의 E1 알고리즘을 구현한 결과이다.

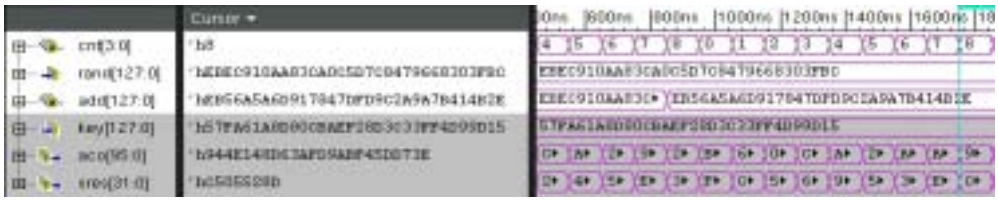


Fig. 42. Hardware simulation of E1 algorithm

Fig. 43. Firmware simulation of E1 algorithm

ii) 암호화

그림 44와 그림 45는 K'c가 '0x00000000000000000000000000000000', addr이 '0x000000000000', 그리고 clk가 '0x00000003' 일 때 그림 18의 암호화기 알고리즘을 구현한 결과이다.

Fig. 44. Hardware simulation of encryption

Fig. 45. Firmware simulation of encryption

(5) 분할 설계의 비교 및 검토

베이스밴드의 구현 결과를 분석하기에 앞서 일반적인 하드웨어와 펌웨어 구현의 장단점을 비교하여 시스템에 미칠 수 있는 영향을 미리 파악해야 한다. 하드웨어는 CPU의 부하를 크게 완화시켜 주고 소비전력과 동작 지연은 적은 반면에 유연성이 떨어지는 단점이 있고, 펌웨어는 작업이 많을 때 CPU의 부하를 가중시켜 전체 시스템의 효율을 떨어뜨리고 소비전력이 증가하는 반면에 유연성이 좋고 하드웨어 복잡도를 줄인다는 장점이 있다.

베이스밴드 블록의 효율적인 분할을 위해서는 이와 함께 구성요소의 특성과 요구사항을 정확히 알아야 한다. 즉 블루투스는 기본적으로 소형, 저가, 저전력이 요구되기 때문에 이를 실현하기 위해서는 구현한 블록의 성능, 크기, 속도, 안정성, 소비전력 등을 고려해야 한다.

(1)~(4)절에서 구현한 결과 각 블록의 크기는 표 4와 같다. 이때 하드웨어와 펌웨어의 크기는 절대적으로 비교될 수는 없지만 앞에서 언급한 내용을 바탕으로 각 블록의 특성을 고려하여 비교·검토하는 과정에서 참고가 될 수 있다. 그리고 이를 통해 다음과 같은 결론을 내릴 수 있다.

Table 6. Block size of implementing baseband

Block		Size		
		HW(Gates)	FW(KB)	HW(Gates)/FW(KB) Division
Access Code	Generator	3,000	2	-
	Correlator	3,300	1	-
Clock Management		4,500	-	-
Authentication (E1)		46,300	6	-
Encryption		7,200	2	-
Hop sequence		10,000	4	HW : 500 FW : 3

그림 3에서 볼 수 있듯이 접근 코드는 각 상태에 따라 다른 LAP를 생성하기 때문에 블루투스가 새로운 연결이 되기 전까지는 다른 접근 코드가 사용된다. 하지만 일단 새로운 연결 상태로 진입하면 접근 코드는 연결이 끊길 때까지 마스터의 LAP에 의해 생성되기 때문에 일정하게 유지된다. 따라서 연결 상태에서 패킷을 송수신할 때 접근 코드를 메모리에 저장해 두었다가 사용해도 무방하다. 이 점을 감안하면, 접근 코드 발생기인 경우 하드웨어로의 구현만을 고집할 필요 없이 표 4에서 볼 수 있듯이 구현 크기도 적합한 펌웨어로 구현하는 것이 좋을 것이다. 그러나 상관기인 경우는 데이터가 들어오는 즉시 검출을 시작하고 실시간으로 동작해야하며 블루투스 클럭과도 밀접한 연관이 있기 때문에 펌웨어로 구현하는 것은 적합하지 않다. 비록 펌웨어로 구현할 경우 그 크기가 적어 유리하지만 무선으로 전송되는 데이터를 펌웨어로 전송시킨 후 다시 하드웨어로 가지고 와야 하는 비효율성과 CPU의 부하가 가중될 때에 전송 지연이 발생하여 패킷 손실을 초래할 수 있다는 점을 감안하면 하드웨어의 구현이 적합하다.

블루투스는 대부분의 동작을 실시간 클럭에 동기 시키고 데이터 교환 시 피코넷에서 마스터와 슬레이브 사이에 클럭과 슬롯의 동기화가 이루어지게 된다. 이를 위해서는 블루투스 클럭이 매순간 갱신 되어야하고 접근 코드 상관기와도 직접 연결이 되어야한다. 블루투스 클럭은 모든 블루투스 동작에 관계되기 때문에 정확성을 잃지 말아야하는데, 만일 펌웨어로 구현된다면 CPU의 성능과 작업 부하에 따라 클럭이 변동되어 정확성을 잃게 될 것이다. 따라서 블루투스 클럭은 펌웨어로

의 구현은 부적합하고 하드웨어로의 구현이 적합하다.

홉 시퀀스는 패킷을 무선으로 안전하게 전송해주기 위해 매우 중요한 부분이다. 이는 패킷 전송 시 매번 다른 값을 생성해야하므로 하드웨어 구현이 적합하지만, 표 4에서 보이듯이 하드웨어 단독으로 구현했을 때는 그 크기가 1만 게이트에 달하게 된다. 이를 개선하기 위해 본 논문에서는 그림 40과 같이 하드웨어와 펌웨어로 나누어 구현하였다. 이때 하드웨어와 펌웨어간 데이터 전송에서 생길 수 있는 과중부하를 고려한다 하더라도 SoC 단일칩 특성상 극복될 수 있다. 비록 펌웨어만으로 구현하는 방법도 가능하겠지만 이 경우는 분할 설계에 비해 전체 연산을 수행하는 CPU의 작업부하가 증가하여 지연 또한 더욱 커지게 된다. 따라서 구현 크기를 줄이면서 일정한 성능을 유지할 수 있는 방법으로 분할 구현이 적합한 방법이다.

E1 알고리즘은 하드웨어로 구현한 경우 키 생성과 SAFER+ 알고리즘(Ar/Ar′)에서 각 단계를 루프로 연산하여 삽입해야 할 세부 블록을 많이 줄이긴 하였으나, 입출력 포트와 연산이 복잡하여 설계하고자 하는 크기에 적합하지 않았다. 반면 펌웨어로의 구현은 E1 알고리즘의 순차적이고 메모리적인 요소가 많기 때문에 하드웨어보다 구현 절차와 크기 면에서 유리하다 할 수 있다. 또한 이 블록은 두 디바이스간의 인증 절차에 사용되고, 두 디바이스의 LM 사이에서 처리되므로 펌웨어로 설계하는 것이 좋다.

선택적으로 사용가능한 암호화기는 그림 17에서 설명되는 것과 같이 생성된 암호화 스트림을 앞단의 결과와 배타적 논립합으로 연산하여 전송하고자 하는 데이터를 암호화하는데, 이는 실시간으로 암호화 스트림이 생성되어야 함을 의미한다. 이와 함께 표 4의 구현 크기는 하드웨어로 구현하는 것이 적합하다는 것을 입증해준다. 또한 이 알고리즘의 구성이 대부분 LFSR로 되어 있으므로 하드웨어 구현이 더욱 적합하다.

표 5는 위에서 분석한 블록 구현결과를 정리한 것이고, 3절에서는 이 결과를 기반으로 단일칩의 블루투스 베이스밴드를 설계한다.

Table 7. Baseband block implementation summary

Block		Design Method	Size		Remark
			HW(Gates)	FW(KB)	
Access Code	Generator	Firmware	-	2	수행 빈도수 및 크기감안
	Correlator	Hardware	3,300	-	타이밍에 민감
Clock Management		Hardware	4,500	-	타이밍에 민감
Authentication (E1)		Firmware	-	6	알고리즘 구성 및 크기감안
Encryption		Hardware	7,200	-	타이밍에 민감
Hop sequence		HW/FW	500	3	타이밍에 민감하지만 하드웨어 단독 구현 시 크기가 너무 크다.

3. 버스 브릿지 모델 설계 및 검증

(1) 단일 버스 브릿지의 설계 및 검증

단일 버스 브릿지 모델은 그림 21(a)에서 보듯이 하나의 버스 브릿지에 모든 레지스터와 베이스밴드 블록을 모두 연결시켜 구현한 것이다.

그림 46은 패킷을 생성한 결과를 보여준 것이고, 그림 47은 마스터와 슬레이브에서 생성되는 블루투스 클럭을 보여준 것이다.

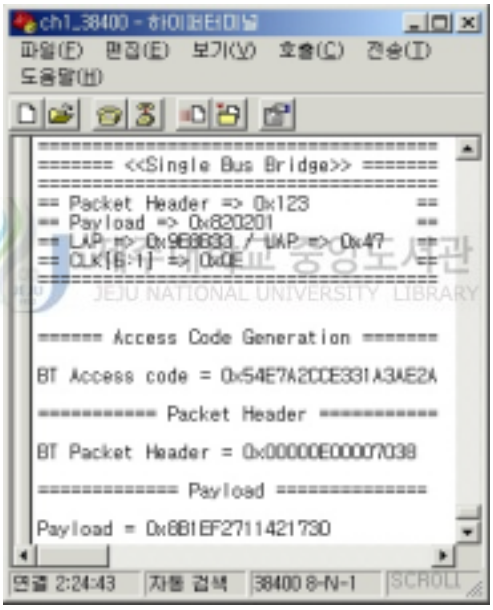


Fig. 46. Packet generation result of single bus bridge

패킷 헤더는 그림 5와 같이 구성되어 있는데, AM_ADDR(active member address)은 '0x3', TYPE(type code)은 '0x4', FLOW(flow control)와 SEQN (sequence number)은 '0x0', 그리고 ARQN(acknowledge indication)은 '0x1'이라 가정하여 생성하였다. 이를 조합하면 '0x123'이고 HEC를 통과하면 '0x01923'이 출력되게 된다. 그리고 비트 랜덤화 회로의 초기값으로 사용되는 CLK_{6-1}은 블루투스

클럭 관리에서 생성된 값으로 그림 46에서는 '0x0E'일 때 패킷 헤더를 생성하므로 그 결과는 다음과 같다.

Whitening stream : 00 0001 0001 0011 0001
HEC output : 00 0001 1001 0010 0011

Whitening output : 00 0000 1000 0001 0010

비트 랜덤화 회로에서 출력된 값은 1/3 FEC를 통과하므로 최종 패킷 헤더는 '0x00000E00007038'이 된다.

페이로드는 일반 L2CAP 메시지로 페이로드 헤더는 1byte이고 바디는 2bytes라 가정하고 테스트 하였다. 페이로드는 그림 7과 같이 구성되는데, 표 2에서 보듯이 헤더의 L_CH 값은 '0x2', FLOW는 '0x0', LENGTH는 '0x2'라 하고, 바디를 '0x0201'이라 하여 전체 페이로드를 '0x820201'이라 가정하였다. 이 데이터가 CRC를 통과하여 '0x145F820201'이 되고, 이 값은 다시 비트 랜덤화 회로를 거치게 되는데 이때 패킷 헤더와 같은 시기에 페이로드를 생성하기 때문에 CLK_{6-1}값은 패킷 헤더와 같고 그 결과는 다음과 같다.

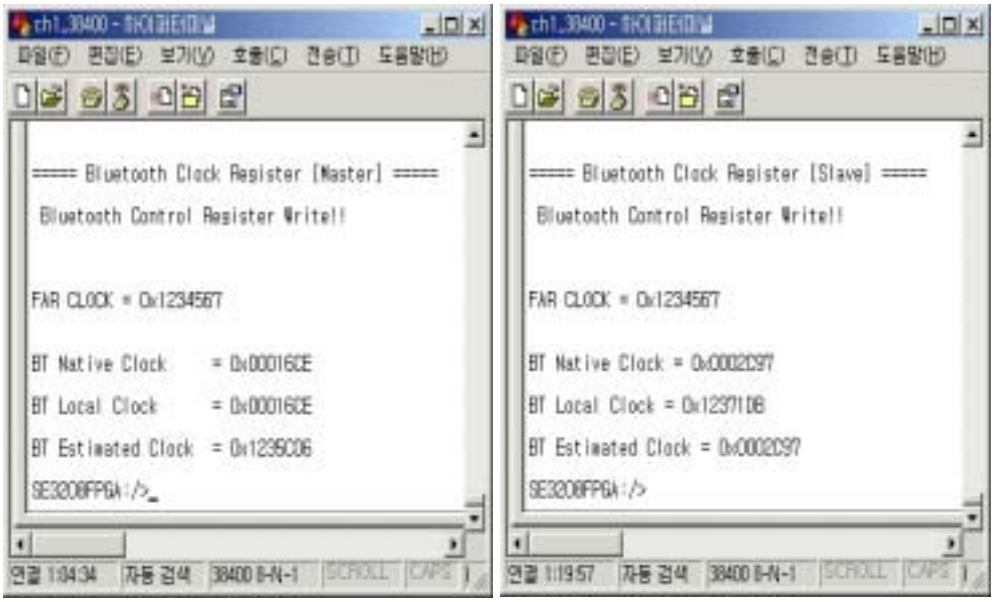
Whitening stream: 0111 0111 1001 0110 0100 1000 0001 0001 0011 0001
HEC_dout : 0001 0100 0101 1111 1000 0010 0000 0010 0000 0001

Whitening output : 0110 0011 1100 1001 1100 1010 0001 0011 0011 0000

비트 랜덤화를 통하여 얻은 '0x63C9CA1330'은 2/3 FEC를 통과하여 최종적으로 다음과 같은 60bits의 페이로드를 얻게 된다.

10001 *0110 0011 11* 01111 *00 1001 1100* 01000 *1010 0001 00* 00101 *11 0011 0000*

위의 데이터 중에 밑줄이 있는 값은 정보 데이터이고 나머지는 2/3 FEC 코드이며 송신되는 페이로드의 최종 값은 '0x8B1EF2711421730'이다.



(a) Master (b) Slave

Fig. 47. Bluetooth clock of single bus bridge

블루투스 클럭은 전원을 켤 때 0으로 리셋 되고 그 후 반 슬롯인 312.5μs마다 1씩 증가는 28bits 카운터로, CLKN(BT Native Clock)이 이에 해당한다. 그림 47의 (a)는 마스터에서의 클럭이고, (b)는 슬레이브에서의 클럭이다. 마스터인 경우 CLK(BT Local Clock)는 CLKN과 같고, CLKE(BT Estimated Clock)는 FHS 패킷을 통해 들어오는 슬레이브의 $CLKN_{27-2}$(Far CLOCK)을 기준으로 수신 시점부터 312.5μs마다 1씩 증가하게 된다. 그리고 $CLKE_{1-0}$는 슬레이브로부터 수신된 접근 코드가 검출되면 '01'로 교정된다. 슬레이브인 경우는 CLKE가 CLKN과 같고, CLK는 FHS 패킷을 통해 들어오는 마스터 디바이스의 $CLKN_{27-2}$을 기준으로 수신된 시점부터 312.5μs마다 1씩 증가하게 된다. 그리고 CLK_{1-0}는 마스터 디바이스로부터 수신된 접근 코드가 검출되면 '00'으로 교정된다.

(2) 이중 버스 브릿지의 설계 및 검증

이중 버스 브릿지 모델은 그림 21(b)에서 보듯이 베이스밴드와 블루투스 레지스

터가 연결되어있는 별도의 블루투스 버스 브릿지가 존재하고, 이것이 시스템 버스 브릿지와 통신 하면서 동작 하도록 구성되어있다. 그림 48은 패킷을 생성한 결과를 보여주는 것이고, 그림 49는 마스터와 슬레이브에서 생성되는 블루투스 클럭을 보여주는 것이다.

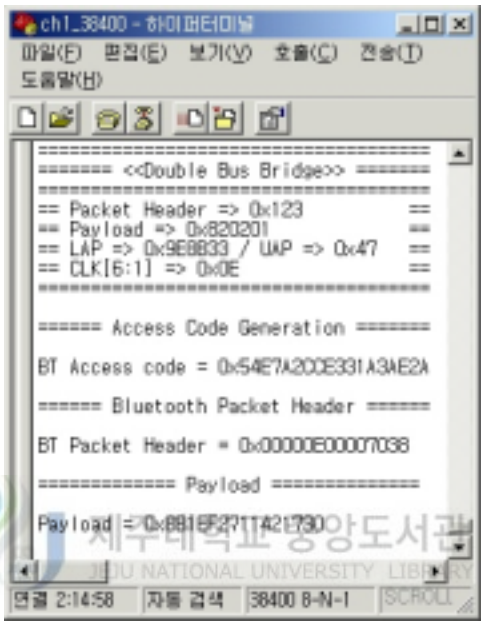
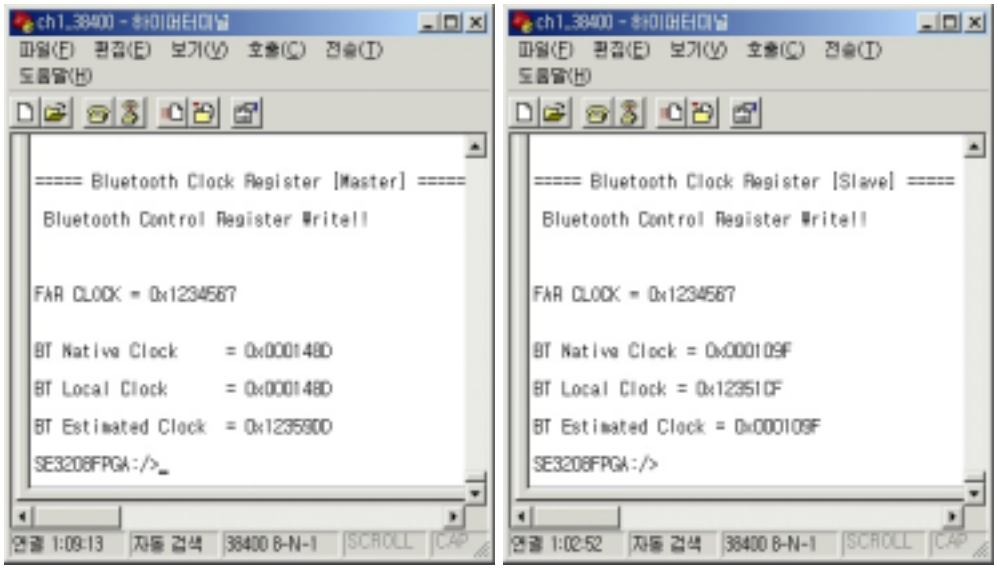


Fig. 48. Packet generation result of double bus bridge

패킷 헤더와 페이로드 생성에 필요한 정보는 단일 버스 브릿지와 같고, 비트 랜덤화 회로에 사용되는 CLK_{6-1}은 블루투스 클럭 관리에서 생성되므로 패킷 생성 시기에 따라 그 값이 항상 다르게 된다. 본 논문에서는 패킷 생성을 CLK_{6-1}이 단일 버스 브릿지와 같을 때 실행하였으므로 패킷 헤더는 '0x00000E00007038'이 되고, 페이로드는 '0x8B1EF2711421730'이 된다.

이중 버스 브릿지의 블루투스 클럭은 단일 버스 브릿지에서 설명한 것과 동일하다.



(a) Master (b) Slave

Fig. 49. Bluetooth clock of double bus bridge

(3) 버스 브릿지 방식 비교 및 검토

타이밍 분석을 함으로써 각 설계 경로와 설계에 대한 최대 시스템 클럭 속도를 알 수 있는데 이를 위하여 그 설계의 셋업 시간 및 홀드 시간, 출력 지연 시간(clock-to-output), 클럭 스큐, 최대 클럭 주파수 등을 살펴볼 필요가 있다. 이는 설계한 칩의 성능과 시간 왜곡을 말해주는데 발견되지 않은 왜곡은 타이밍 해저드(hazard)나 경쟁상태(race conditions)를 야기할 수 있으므로 이들은 회로를 오동작 시킬 수 있는 주원인 중의 하나이다.

표 6은 (1), (2)절에서 설계한 HDL을 합성하여 얻은 결과로 두 방식은 타이밍에서 서로 다른 결과를 보임을 알 수 있다. 여기서 입력 도달 시간은 셋업 시간을 말하는 것으로 데이터나 인에이블 입력을 통하여 레지스터에 도달하는 데이터는 레지스터의 클럭 신호가 인가되기 전에 입력 핀에 도착하여야 한다. 즉, 셋업 시간은 액티브 클럭 에지 전에 이 데이터가 도착해야 하는 최소한의 시간 길이이다. 출력 지연 시간은 액티브 클럭 에지 후에 출력 핀에서 유효한 출력을 얻어내는데 필요한 시간이다. 최대 클럭 주파수는 설계 클럭이 내부 셋업과 홀드 시간 요건을

위반하지 않고 작동될 수 있는 가장 빠른 속도를 말한다. 내부 최대 주파수를 결정하려면 회로의 클럭 주기가 계산되어야 하는데 클럭 주기는 데이터 경로 지연이나 레지스터간의 클럭 스큐, 소스 레지스터의 출력 지연 시간, 목표 레지스터의 셋업 시간에 달려 있다. 그리고 블루투스의 동작은 항상 실시간 클럭에 의존하기 때문에 조합 경로 지연이 짧아야 좋은 성능을 유지할 수 있다. 표 6을 살펴보면, 최대 주파수, 입력 도달 시간, 출력 지연 시간, 조합 경로 지연 모두 이중 버스 브릿지인 경우가 유리함을 알 수 있는데, 이는 내부 지연이 작아 회로 동작의 속도를 향상 시킬 수 있음을 의미한다.

Table 8. Synthesis report

Timing Summary	Single Bus bridge	Double Bus bridge
Minimum period (Maximum Frequency)	92.925ns (10.761MHz)	91.908ns (10.880MHz)
Minimum input arrival time before clock	23.817ns	20.134ns
Maximum output required time after clock	88.525ns	86.330ns
Maximum combinational path delay	28.196ns	22.936ns

표 7은 합성 후 구현하여 얻은 결과로, 설계 후에 FPGA에서 사용된 LUT(look up table)와 게이트 수를 나타낸 것이다. 4 입력 LUT는 16bits RAM을 이용하여 논리 회로를 구성한 것으로 16개의 1bit 메모리 셀에 '1' 또는 '0'의 값을 넣어 놓고 주소 입력 4bits의 값에 따라 16가지 패턴 중 하나를 출력할 수 있어 4bits 입력의 조합 논리 회로를 완벽하게 커버할 수 있다. 표 6에서 보면, 큰 차이는 아니나 단일 버스 브릿지로 설계한 경우가 더 적은 LUT와 게이트 수를 사용하였음을 알 수 있는데, 이로 인해 칩 면적과 비용을 줄일 수 있다.

Table 9. Implementation report

Design Summary	Single Bus bridge	Double Bus bridge
Total Number 4 input LUTs	5,721	5,742
Total equivalent gate count for design	60,956	61,028

클럭은 칩의 동작에 있어 메모리, 플립플롭, 그리고 래치 등 많은 클럭 게이트를 구동하기 위한 동작 타이밍의 기준이 되는 신호로, 이 클럭 신호로 인해 칩 내부의 동기화가 가능하게 된다.

그림 50의 클럭 스큐는 클럭 소스로부터 클럭을 필요로 하는 각 단자(sink)까지의 '지연시간의 최대 차'로 정의한다. 즉, 소스에서 각각의 단자까지 클럭 신호가 도달하는 시간은 비슷해야하며 '0'에 가까워야 한다. 이러한 클럭 스큐는 회로의 속도 향상에 있어 중요한 제약요소가 되고 있고, 이 시간차는 두 개의 클럭 신호 경로가 서로 다른 길이를 갖고 있을 때 발생한다.

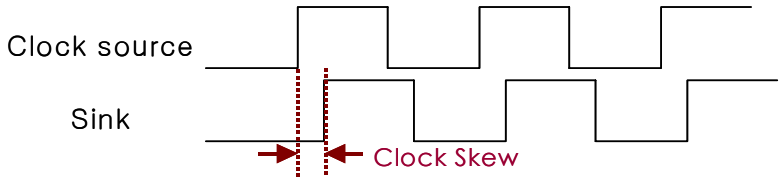


Fig. 50. Clock skew

표 8은 설계한 블루투스의 구현 결과 얻어진 클럭 스큐에 관한 정보로, 스큐의 평균값과 식 (1)로 계산한 표준편차를 이용하여 단일 버스 브릿지와 이중 버스 브릿지를 비교해 보았다. 그 결과 평균 클럭 스큐는 단일 버스 브릿지인 경우 2.170ns이고, 이중 버스 브릿지인 경우 1.380ns이므로 이중 버스 브릿지가 각 단자에 도달 하는 클럭 신호의 시간차가 더 짧다는 것을 알 수 있다. 그리고 클럭 스큐의 표준 편차를 살펴보면, 단일 버스 브릿지인 경우 1.479ns이고, 이중 버스 브릿지인 경우 1.037ns이기 때문에 이중 버스 브릿지가 도달 시간 또한 더 비슷함을 알 수 있다. 이는 이중 버스 브릿지에서 같은 클럭을 사용하는 블록끼리 묶어서 구현하였기 때문에 클럭 패턴길이를 줄일 수 있고, 이로 인해 회로의 속도 향상에 도움을 준 것으로 보인다.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - m)^2} \qquad (1)$$

Table 10. Generating clock report

Clock Net	Max Skew(ns)	
	Single Bus bridge	Double Bus bridge
DIV_clk_b	4.285	1.700
DIV_clk_b_1	3.645	2.535
DIV_clk_b_2	3.484	2.240
BT_BB_extclk_reg_0__n0001	1.801	1.580
BT_BB_AC_COR__n0004	4.436	3.596
UART_CH0_tx16cnt<3>	2.496	0.934
UART_CH0_tx_empty	0.000	0.000
UART_CH0_bclk	1.719	0.208
UART_CH0_rxclk	1.119	0.831
UART_CH0_rx_full	0.000	0.000
UART_CH0_clk_src	0.836	1.650
SE3208_TOP_MAIN_CTRL_PIPELINE_CTRL_cs<8>	2.170	1.228
Average	**2.170**	**1.380**
Standard Deviation	**1.479**	**1.037**

표 6과 표 7, 표 8에서 보여준 결과를 표 9에 정리하였다. 타이밍 측면에서는 단일버스 브릿지에 대한 이중 버스 브릿지의 백분율을 구한 것이고, LUT와 게이트 사용면에서는 이중 버스 브릿지에 대한 단일 버스 브릿지의 백분율을 구한 것이다.

Table 11. Implementation result comparison of two bus bridges

Factor / Architecture	Clock skew (%)	Setup time (%)	Clock to output (%)	Clock period (%)	combina-tional path delay (%)	Total Number LUTs (%)	Total equivalent gate count (%)
Single bus bridge	100.0	100.0	100.0	100.0	100.0	99.6	99.9
Double bus bridge	63.6	84.5	97.5	98.9	81.3	100.0	100.0
Superiority	*Double*	*Double*	*Double*	*Double*	*Double*	*Single*	*Single*

앞에서도 설명하였듯이 칩을 설계함에 있어 타이밍이 회로 동작에 미칠 수 있는

영향은 매우 크고, 더욱이 블루투스는 실시간 동작으로 타이밍에 민감하기 때문에 타이밍이 어긋날 때는 데이터 유실을 초래할 수 있다. 즉, 클럭 스큐, 입력도달시간, 출력지연시간, 클럭 주기, 조합논리회로간의 지연은 회로 동작에 있어 중요한 요소가 된다. 표 9에서 확인해 보면 이 요소들은 이중 버스 브릿지가 우수하고 이 구조로 구현하는 것이 타이밍 측면에서는 더 우수하다. 비록 LUT와 게이트, 즉 칩 면적과 비용면에서는 단일 버스 브릿지가 우수하지만 이는 표 9에서 볼 수 있듯이 단일 버스 브릿지와 이중 버스 브릿지의 차이가 그다지 크지 않기 때문에 이중 버스 브릿지로 구현해도 무방하다. 또한 이 크기의 차이는 이중 버스 브릿지의 구조가 단일 버스 브릿지 구조에서 버스 브릿지 역할이 둘로 분리되면서 나타난 것이기 때문에 나중에 전체 시스템을 설계해도 그 값의 차이는 변하지 않을 것이다.

따라서 향상된 성능과 속도를 가진 블루투스 시스템을 설계하기 위해서는 이중 버스 브릿지로 설계하는 것이 더 효율적이다.

Ⅴ. 결론

본 논문은 SoC 개념을 더한 블루투스 칩 개발의 기반을 마련하기 위한 연구로 저가격화, 저소비전력화, 그리고 소형경량화를 실현하면서 효율적인 블루투스 칩의 설계 방안을 제시하는데 목적이 있다.

이를 수행하기 위하여, 본 논문에서는 블루투스 베이스밴드의 전체적인 흐름을 이해하고 각 블록의 알고리즘을 분석하였으며, 설계하고자 하는 베이스밴드 단일 칩의 내부 구조와 이를 구현하기 위한 설계 방법 및 테스트 환경을 보여주었다. 단일칩 구현에 앞서 베이스밴드 블록 중에 중요한 몇 가지 블록을 구현하고, 이를 SoC 형태에 초점을 두어 여러 환경과 특성을 고려하여 비교·검토하였다. 그 결과 접근 코드와 인증은 수행 빈도수와 크기를 감안하여 펌웨어로 설계할 것을 제안하였는데 그 크기는 각각 2KB와 6KB이다. 그리고 접근 코드 상관기, 암호화기, 그리고 블루투스 클럭은 타이밍에 매우 민감하기 때문에 하드웨어로 설계할 것을 제안하였고 그 크기는 각각 3,300Gates, 4,500Gates 그리고 7,200Gates를 차지하였다. 홉 시퀀스 또한 시간에 의존하므로 하드웨어로 구현하는 것이 좋지만 그 크기가 너무 크기 때문에 하드웨어와 펌웨어로 분할하여 구현하는 것을 제안하였고 이 결과 하드웨어의 크기는 단독 구현 크기의 5%로 줄어들어 500Gates가 되었고, 펌웨어는 3KB를 얻었다.

이렇게 검증된 베이스밴드 블록을 EISC 방식의 SE3208 코어에 단일 버스 브릿지와 이중 버스 브릿지 각각의 방식으로 연결하여 구현하였고, 실제 FPGA (Xilinx) 보드 레벨에서 그 동작을 확인하였다. 그리고 합성·구현하는 과정에서 얻어진 데이터를 서로 비교·검토한 결과, 본 논문에서 제시한 두 모델 중에 이중 버스 브릿지 구조가 클럭 스큐, 입력도달시간, 출력지연시간, 클럭 주기, 그리고 조합논리회로간의 지연이 각각 단일 버스 브릿지 값의 63.6%, 84.5%, 97.5%, 98.9%, 81.3%에 해당함을 알 수 있었다. 비록 LUT와 게이트 사용면에서는 단일 버스 브릿지가 유리하지만 그 차이는 크지 않을 뿐 아니라 블루투스가 타이밍에 민감한 시스템이고 타이밍이 회로 동작에 있어 중요한 문제이기 때문에 이중 버스 브릿지

로 구현하는 것이 회로 동작의 속도와 성능을 향상 시킬 수 있고, 보다 안정적인 블루투스 시스템을 설계할 수 있다.

본 논문은 블루투스 단일칩을 설계하기 위한 한 과정이며, 향후에는 외부에 부착한 메모리를 칩 내부에 구현하고 RF 및 베이스밴드의 수신단을 포함하여 제어 부분을 보완해 나갈 것이다. 본 논문은 블루투스 베이스밴드를 기존의 구현방법에서 벗어나 보다 효율적으로 시스템을 구성할 수 있도록 새로운 각도로 접근하여 전체 블루투스 시스템의 성능을 향상시키고 블루투스 특징에 부합되는 코어 설계의 기초 자료로 활용될 것으로 사료된다. 또한 이렇게 구현된 칩은 소형, 저가를 선호하는 미래의 유비쿼터스 시장에서 매우 유용하게 사용될 것이다.

참고 문헌

- Jennifer Bray and Charles F Sturman, 2001, BLUETOOTH Connect without Cables, Prentice-Hall, pp.41-90, pp.291-331, pp.399-421
- SIG, 2000, Specification of the Bluetooth System Version 1.1 Part B : Baseband specification, pp.41-178.
- Jamil Khatib, 2002, Bluetooth IP Core Specification, pp.8-11,14-15.
- Advaned Digital Chips Inc., 2002, MUSE work book Version 0.1, pp.29-44,
- Advaned Digital Chips Inc., 2002, SE3208 Core Manual, Extendable Instruction Set Computer Version 2.0
- 이문수, 강치운, 오종택, 이정재, 변건식, 2001, 한국어판 블루투스 Connect without Cables, 홍릉과학출판사, pp.45-101, pp.313-358, pp.431-454

URL :

- http://www.bluelogic.co.kr/korean/technology.htm
- http://pbt.co.kr/bluetooth/bluetooth1/6.htm
- http://rsiwin.com.ne.kr/docu/DATA/introduce1.htm
- http://www.microvision.co.kr/bluetooth/lecture/lecture_protocol_1.htm
- http://www.bluelogic.co.kr/korean/core.htm
- http://sun.uos.ac.kr/network/bluetooth/bt_00.htm
- http://www.pldworld.com/_altera/html/apps/qtiming.htm
- http://auto.ats.go.kr/ats_admin/upfiles/data/UBiquitous.pdf

감사의 글

대학 4년을 지내면서 항상 생각해왔던 대학원에 들어온 지 벌써 2년이 지나 학위논문을 완성하기에 이르렀습니다. 아직 해야 할 공부도 많고 하고 싶은 것도 많은데 지금 이 시점에 서고 보니 아쉬움이 참 많이 남습니다.

대학 2학년 때 인연을 맺어 지금까지 저를 지도해 주신 임재윤 교수님께 깊은 감사의 말씀을 드립니다. 교수님께서 부족한 저에게 항상 격려를 아끼지 않으시고 많은 기회를 주셨기 때문에 제가 이만큼의 결과를 얻고 졸업을 할 수 있게 되었습니다. 그리고 바쁘신 시간을 쪼개어 제 논문이 완성되도록 지도해 주신 김홍수 교수님과 좌정우 교수님께 감사드립니다. 통신에 대한 많은 매력을 주시고 항상 따뜻하게 대해주신 문건 교수님과 이용학 교수님께도 감사드립니다. 그리고 학부 때부터 많은 가르침을 주셨던 양두영 교수님과 강진식 교수님 감사합니다. 통신공학과의 모든 교수님 항상 건강하십시오.

연구실은 달라도 많은 조언을 해주시는 마음 따뜻한 부식 선배님, 성욱 선배님, 권익 선배님, 그리고 대학원 2년 동안 많은 의지가 되어준 봉수 오빠 고맙습니다. 과사에 있으면서 많은 도움을 준 성익 오빠와 윤희에게도 고마운 마음을 전합니다. 연구실 선배인 재필오빠, 영애언니, 창윤오빠, 논문 쓸 때마다 많은 도움을 받았는데 너무 고마웠습니다. 대학원 생활동안 많은 의지가 되고 친동생처럼 아껴주고 챙겨준 은진언니와 진숙언니, 수미언니 덕분에 대학원 생활이 한결 편했습니다. 올해 후배로 들어와서 선배처럼 의지가 많이 된 진아언니, 군선오빠, 1년 남은 대학원생활 열심히 해서 좋은 결실 맺길 바랍니다. 그리고 가을학기에 입학해서 항상 웃게 해준 상보오빠도 좋은 결실 맺기 바라고, 예비대학원생 훈철오빠 대학원생활 열심히 하리라 생각됩니다. 한 연구실에서 동기로 서로 격려와 의지가 된 영배오빠 서울 직장 생활 잘 하세요. 그리고 어려움을 함께 해 온 학부 졸업동기이자 대학원 동기인 영길오빠, 새로운 사회생활을 잘 해나가길 바라고 입학동기지만 직장 생활로 졸업이 늦춰진 광식오빠, 좋은 논문 쓰길 바랍니다. 그리고 논문 쓸 때 도움을 준 (주)인터에프씨 직원 모두에게도 고마움을 전합니다. 마지막으로 항상 사랑과 격려를 아끼지 않고 저를 믿어준 부모님, 오빠들과 언니들, 현수 그리고 예쁜 조카 효주, 성화, 채은이 그리고 친구들 모두 너무 사랑합니다.