Bluetooth Baseband의 효율적인 분할 및 구현기법

*김 현 미, *진 군 선, **임 재 윤
*제주대학교 통신공학과, **제주대학교 통신컴퓨터공학부
전화 : 064-754-3635 / 핸드폰 : 017-691-4110

The efficient division and implementation technique of Bluetooth Baseband

Hyun Mi Kim, Gun Sun Jin, Jea Yun Lim
Dept. of Telecommunication Engineering, Jeju University
E-mail : dasom0411@hanmail.net

Abstract

This paper discussed whole concept of bluetooth baseband and studied its detail algorithm. Important blocks, access code, security and clock management, are implemented and verified to hardware and firmware according to Specification ver.1.1. Then implementation results are compared and examined.

Finally, this paper suggested the efficient system implementation method. By using test board, it could confirm that suggested implementation communicated smoothly.

I. 서론

Bluetooth는 단거리 무선통신을 위한 기술 규격으로, 소형, 저가, 저전력과 좁은 구역 내의 무선 연결을 지원하는 것이 특징이다. 무선으로 데이터를 송수신할 수 있는 거리는 10m(옵션 100m)이고, 이 안에서는 1Mbps의 속도로 안정적으로 데이터를 송수신 할 수 있다. Bluetooth는 ISM 대역인 2.4GHz대를 사용하는데, 이때 발생할 수 있는 간섭을 막기 위해 FHSS (Frequency Hop Spread Spectrum) 방식을 채택한다. Bluetooth는 무선의 전 방향 통신을 하므로 각종 기기들을 분리시킬 수 있어 공간 활용도를 높이고 보안 알고리즘의 사용으로 보안관련 시스템에도 안심하고 사용할 수 있다. [1-4]

Bluetooth의 기술적인 면을 살펴보면, 상위계층의 protocol stack과 하위계층의 baseband로 크게 나눌 수 있는데, 본 논문에서는 bluetooth 코어 설계를 효율적으로 수행하기 위해, baseband의 전체적인 구조와 access code, security 그리고 clock management를 설명하고 분석하였다. 또한 이를 최선의 구현방법으로 하드웨어와 펌웨어로 분할 설계하여 이를 비교·검토함으로써 효율적인 설계방안을 제시하고, 이를 검증하였다.

II. Bluetooth Baseband

Baseband는 bluetooth 스택에서 가장 하부에 위치한 계층으로 RF링크 및 피코넷을 설정·관리하는 가장 핵심적인 프로토콜이라 할 수 있다. 그림 1은 baseband의 구조를 보여준다.

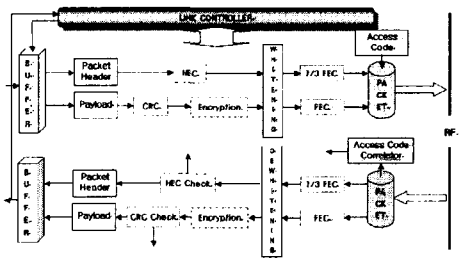

그림 1. baseband 구조

Bluetooth는 79 RF 채널로 호핑을 하는데, 이 채널의 정의와 홉 시퀀스 선택은 모두 baseband에서 담당한다. 또한 baseband는 표준패킷을 정의하고 생성하는데, 패킷은 그림 2와 같이 구성되어 있고, 그 역할 및 링크 종류에 따라 링크 컨드롤 패킷, ACL 패킷, SCO 패킷으로 나누어진다. 또 이 패킷들은 다시 payload 길이와 FEC, CRC의 사용 여부에 따라 더 세분화된 패킷으로 나누어지고, 에러 정정 및 에러 검출이 수행된다. 이 밖에도 security기능이 이루어진다.

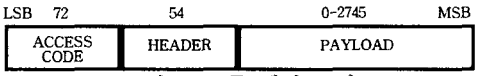

그림 2. 표준 패킷 구성

2.1 Access code

Access code는 모든 패킷에 포함되고 패킷의 가장 처음에 송수신되는 부분이다. Access code는 수신된 패킷이 자신이 속한 피코넷의 데이터인지를 판단하며 다른 피코넷의 데이터인 경우 패킷은 버려지게 된다. 또한 수신된 frame의 시작을 검출하여 시스템이 수신 frame의 clock에 맞출 수 있도록 한다.

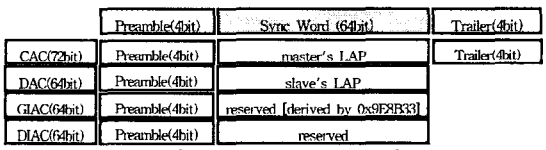

그림 3. Access code 구성

그림 3은 access code의 구성을 보여주는 것으로서, 그 사용 목적에 따라 사용되는 LAP값이 달라진다.

2.2 Bluetooth security

(1) Key management

디바이스간의 인증과 암호화를 위해 사용되는 링크 키는 SAFER+ 알고리즘의 "E" 구현을 통해 생성된 128비트의 값으로, unit key와 combination key, 그리고 master key, initialization key가 있다.

(2) Authentication

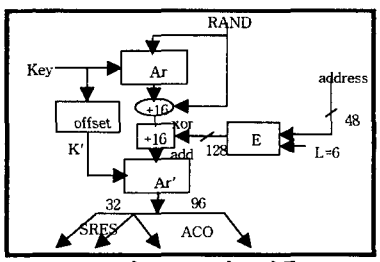

그림 4. E1 알고리즘

Challenge-response 구조로, 양방향 프로토콜이 사용되는데, 이는 두 디바이스가 같은 키를 갖고 있는지, 그리고 인증 과정을 성공적으로 수행했는지를 체크한다. 그림 4에 의해 이 과정에서 생성된 ACO 값은 양쪽 디바이스에 저장되고 암호화키를 만드는데 사용된다.

(3) Encryption

사용자 정보는 패킷의 payload 부분을 암호화함으로서 보호될 수 있는데, 이는 모든 정보를 재 동기화하는 E_0라 불리는 그림 5와 같은 열 암호화기(stream cipher)에 의해 수행된다.

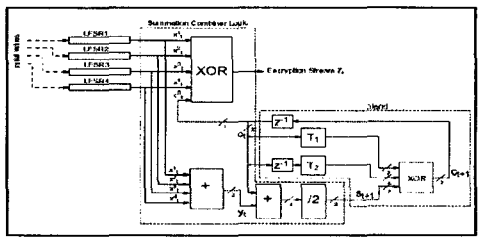

그림 5. 암호화기 알고리즘

2.3 Bluetooth clock management

Bluetooth clock은 Tx-Rx 데이터 교환을 동기시키고, 손실 패킷과 재전송 패킷사이를 구별하며, 의사 랜덤 열을 생성하도록 해준다. 그림 6의 bluetooth clock은 전원을 켤때 리셋되고 312.5us마다 증가하는 28-bit 카운터로 구성된다. 이 클럭이 CLKN이 되고, 그림 7과 같이 offset을 추가하여 CLK와 CLKE를 생성한다.

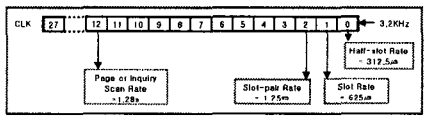

그림 6. Bluetoth Clock Generation

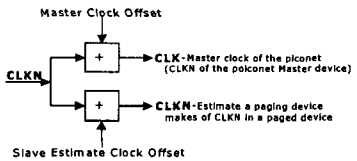

그림 7. Bluetooth clock의 개념

III. Baseband 블록 구현 및 검증

3.1 Bluetooth 기술의 구현

Bluetooth stack은 일반적으로 HCI를 기준으로 protocol을 배분하지만, 별도의 host가 존재하지 않는 'Wholly Embedded Single Architecture'의 형태로 구현되면서 baseband의 구현도 하드웨어만으로 국한되지 않고 펌웨어로의 구현도 가능해지게 되었다. 따라서 이를 장착할 MCU의 역할도 더욱 중요해지게 되는데, 본 논문에서는 다른 MCU에 비해 코드 사이즈가 적고 임베디드 시스템에 적합한 EISC(Extendable Instruction Set Computer) core를 사용하였다.

일반적으로, 하드웨어는 MCU의 부하를 크게 완화시켜 주는 반면 유연성이 떨어지고, 펌웨어는 MCU 부하

의 위험으로 전체 시스템의 효율을 떨어뜨릴 수 있으나 유연성과 복잡하지 않은 하드웨어를 제공한다. 이를 바탕으로 본 논문에서는 주요 블록을 하드웨어(HDL(Cadence)로 설계하여 FPGA(Altera)로 테스트)와 펌웨어(EISC기반의 C로 설계)로 설계하여 그 결과를 블루투스 SIG 스펙의 샘플데이터와 비교하였으며, 성능, 크기, 안정성 등을 고려하여 효율적인 구현 방안을 제시하고 그림 8의 환경에서 테스트하여 그 결과를 검증하였다.

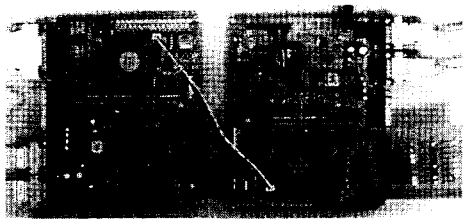

그림 8. 테스트를 위한 통합 보드(EISC-FPGA)

3.2 Access code의 구현 및 검증

Access code는 bluetooth 기기의 고유 주소 LAP가 '0x9E8B33'인 경우를 테스트 하였다.

(1) Generator

그림 9와 10은 그림 3의 access code를 하드웨어와 펌웨어로 구현한 결과이다.

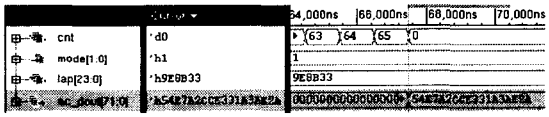

그림 9. Generator 하드웨어 시뮬레이션

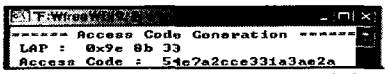

그림 10. Generator 펌웨어 시뮬레이션

(2) Correlator

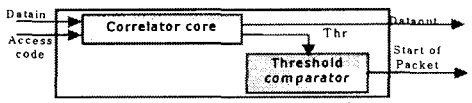

그림 11. Access code correlator 블록도

이 블록은 수신된 frame을 검출하는 것으로 그림 11과 같이 구현하여 이를 그림 12와 13에 나타내었다.

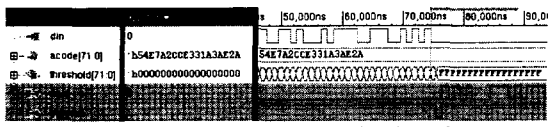

그림 12. Correlator 하드웨어 시뮬레이션

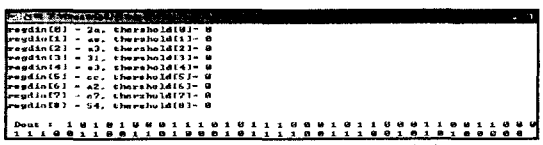

그림 13. Correlator 펌웨어 시뮬레이션

3.3 Bluetooth security 구현 및 검증

(1) Authentication(E1)

그림 14과 15는 다음의 입력 값을 이용하여 그림 4의 E1 알고리즘을 구현한 결과이다.

- ☐ rand = bc3f30689647c8d7c5a03ca80a91eceb
- ☐ address = 7ca89b233c2d
- ☐ key = 159dd9f43fc3d328efba0cd8a861fa57

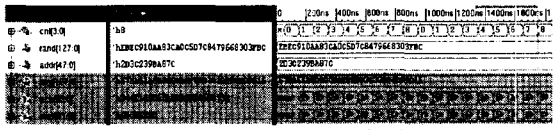

그림 14. E1 알고리즘 하드웨어 시뮬레이션

그림 15. E1 알고리즘 펌웨어 시뮬레이션

(2) Encryption

그림 16와 17은 다음의 입력 값을 이용하여 그림 5의 암호화기 알고리즘을 구현한 결과이다.

- ☐ K'c = 00000000000000000000000000000000
- ☐ addr = 000000000000
- ☐ clk = 00000003

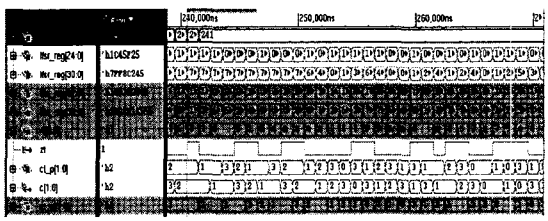

그림 16. Encryption 하드웨어 시뮬레이션

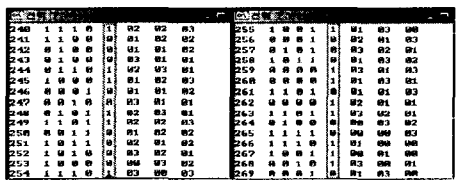

그림 17. Encryption 펌웨어 시뮬레이션

3.4 Bluetooth clock

그림 18은 2.3절에서 설명한 bluetooth clock를 하드웨어로 구현한 결과이다.

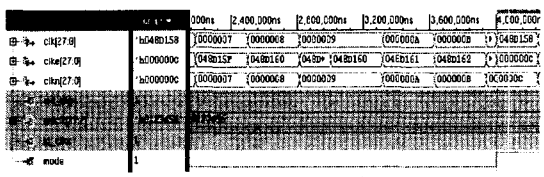

그림 18. Bluetooth clock 시뮬레이션

3.5 Baseband 구현의 비교 및 검토

위의 3.2~3.4절에서 구현한 결과, 각 블록의 크기는 표1과 같고, 3.1절에서 언급한 내용을 바탕으로 표1의 내용과 각 블록의 특성을 고려하여 다음과 같은 결론을 내릴 수 있다.

표 1. 블록별 구현 크기

Block		Size	
		HW(Gates)	FW(KB)
Access Code	Generator	3,000	4
	Correlator	3,300	23
Encryption		7,200	25
Authentication (E1)		?	28
Clock Management		4,500	X

*Access code*는 *inquiry/page* 상태에서는 새로운 값을 만들어내야 하지만 일단 *connection* 상태로 진입하면 master의 LAP에 의해 만들어진 *access code* 값을 유지하게 된다. 따라서 generator인 경우 하드웨어로의 구현을 고집하기 보다는 표 1에서도 볼 수 있듯이 크기면에 유리한 펌웨어로 구현하는 것이 좋을 것이다. 그러나 correlator인 경우는 데이터가 들어오는 즉시 검출을 시작하고 실시간으로 동작해야하기 때문에 하드웨어로의 구현이 적합하겠다. 또한 표1에서의 구현된 크기를 보더라도 이는 명백해질 것이다.

E1 알고리즘은 하드웨어로 구현한 경우, key 생성과 *SAFER+* 알고리즘(*Ar/Ar'*)에서 각 단계를 루프로 연산하여 삽입해야 할 세부 블록을 많이 줄이긴 하였으나 입출력 port와 연산이 복잡하여 설계하고자 하는 크기에 적합하지 않았다. 반면, 펌웨어로의 구현은 E1 알고리즘의 순차적이고 메모리적인 요소가 많기 때문에 하드웨어보다 구현 절차와 크기면에서 유리하다 할 수 있다. 또한 이 블록은 두 디바이스간의 인증 절차에 사용되고 그 후 실제 정보 암호화는 다른 알고리즘을 사용하므로 펌웨어로 설계하여도 무방할 것으로 사료된다.

암호화기는 선택적으로 구현되는 부분으로서 앞단의 결과와 XOR되어 전송하고자 하는 데이터를 암호화하게 된다. 따라서 이는 실시간으로 암호화데이터가 생성이 되어야 하고 표 1의 구현 크기를 보더라도 하드웨어로 구현하는 것이 적당할 것이다.

Bluetooth는 대부분의 동작을 실시간 클럭에 동기시키는데, 데이터 교환시 piconet에서 master와 slave 사이에 동기화(clock/slot)가 이루어지게 된다. 이를 위해서는 bluetooth clock이 매순간 업그레이드 되어야하고 access code correlator와도 직접 연결이 되어야한다. 따라서 bluetooth clock은 하드웨어로 구현하는 것이 적합하다 하겠다.

Ⅳ. Bluetooth 통신

4장에서는 3장에서 설계하고 검증한 블록을 토대로 ID 패킷과 NULL 패킷의 송수신을 구현하였다. 그 구현한 결과를 그림 8의 bluetooth 개발 보드를 이용하여 그림 19, 20과 같이 그 기능을 검증하였다.

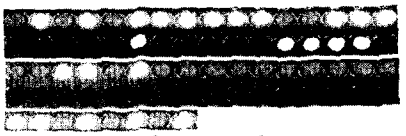

그림 19. ID 패킷

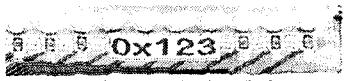

그림 20. NULL 패킷

Ⅴ. 결론

본 논문은 *bluetooth core* 개발의 기반을 마련하기 위한 연구로, 소형, 저가, 그리고 저전력을 실현하면서 기기간의 원활한 통신을 가능하도록 하는데 목적이 있다.

이를 수행하기 위하여, bluetooth baseband의 전체적인 흐름을 이해하고, 주요 블록의 알고리즘을 분석하였다. 그리고 나서 access code와 security, bluetooth clock을 bluetooth SIG 스펙에 맞추어 알고리즘을 하드웨어와 펌웨어로 구현하였고 이를 FPGA와 MCU를 이용하여 테스트하고 검증하였다. 임베디드 시스템 개발과 SoC 산업이 두드러지고 있는 요즘, 'Wholly Embedded Single Architecture' 형태의 구현을 고려할 수 있는데, 이와 함께 여러 가지 환경과 요구조건을 적절히 고려하여 각 블록의 구현 결과를 비교·검토함으로써 적합한 구현 방안을 제시하였고, 이를 토대로 구현한 *ID* 패킷과 Null 패킷의 송수신 통신이 원활함을 확인할 수 있었다.

본 논문은 bluetooth 전체 시스템의 성능을 향상시키고 bluetooth 특징에 부합되도록 설계하기 위한 기초자료로 활용될 것으로 사료된다.

참고문헌

[1] Jennifer Bray and Charles F Sturmam, 2001, "BLUETOOTH Connect without Cables", Prentice-Hall, pp.399-421, pp(33, 63-64)

[2] Specification of the Bluetooth System Version 1.1 *"Part B : Baseband Specification"* pp.47-178, pp.127-138, 143-148

[3] Jamil Khatib, 2002, "Bluetooth IP Core Specification" pp.8-11,14-15