濟州大學校 尖端技術硏究所 論文集, 第13卷, 1號, pp.98-106, 2002. 6
Cheju Nat'l Univ. Res. Inst. Adv. Tech. Jour., Vol.13, No.1, pp.98-106, 2002. 6

블루투스 베이스밴드의 효율적 설계 및 검증

김 진 숙* · 김 현 미* · 임 재 윤*

Efficient Design and Verification of the Bluetooth Baseband

Jin-Sook Kim*, Hyun-Mi Kim* and Jea-Yun Lim*

ABSTRACT

This paper has described the whole concept of hop sequence, access code, authentication in bluetooth baseband and studied algorithms of their types. They are implemented and verified both hardware and firmware of hop sequence, access code, authentication according to Specification ver.1.1, and compared and examined two schemes with performance, size, cost, and time-to-market, etc. This paper has suggested the efficient system division to design optimum system depending on developing environment.

Key Words : bluetooth, hop sequence, access code, authentication

Ⅰ. 서 론

블루투스(bluetooth)는 각종 전자기기간의 통신에 물리적인 케이블 없이 무선주파수를 이용하여 고속으로 데이터를 주고받을 수 있는 규격을 말한다. 2003년에 이르러 세계적으로 30억 달러 규모의 시장을 형성할 것으로 예상되고 있으며 이에 많은 업체들이 차세대 근거리 무선데이터통신으로 급부상하고 있는 블루투스 모듈 개발 및 여러 응용 기기 시장에 주력하고 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 하지만 블루투스의 핵심적인 부분이라고 할 수 있고, 따라서 RF단과 더불어 그 설계 기술은 블루투스의 중심이라고 할 수 있다. 블루투스는 그 구현이 아주 복잡하며 하위계층인 경우 하드웨어로 구현하느냐 펌웨어로 구현하느냐에 따라 그 효율성에 있어서 차이를 보이게 된다. RF단은 아직까지 하드웨어 이외의 것으로는 구현이 불가능한 반면, 베이스밴드는 하드웨어/펌웨어 구현이 개발자의 선택에 따라 결정될 수 있으므로 구현상의 최적 분할을 통해 시스템을 효과적으로 운영할 수 있다[1-4].

본 논문에서는 블루투스에서의 홉 시퀀스와 연결 설정 과정에서 중요하게 다루어지는 access code와 authentication을 하드웨어와 펌웨어로 구현해 본다. 그리고 나서 생성 빈도수, 크기, 작업이 이루어지는 시간, MCU에 걸리는 부하 등을 기준으로 하드웨어/펌웨어 구현을 비교함으로써, 가장 효과적으로 베이스밴드를 설계할 수 있는 최적의 구현 분할을 제안한다.

* 제주대학교 통신컴퓨터공학부, 첨단기술연구소
Faculty of Telecommunication and Computer Engineering, Research Institute of Advanced Technology, Cheju Nat'l Univ.

II. 블루투스 베이스밴드 일반

베이스밴드는 블루투스의 링크 컨트롤러(link-controller)에 해당하는 프로토콜로서 블루투스만의 고유한 통신 시스템 특성을 구현하는 곳이다. 따라서 그 설계를 위해서는 구성 요소 각각의 요구와 기능을 정확히 이해하는 것이 중요하다. Fig. 1은 베이스밴드의 세부적인 블록도를 보여주고 있다.

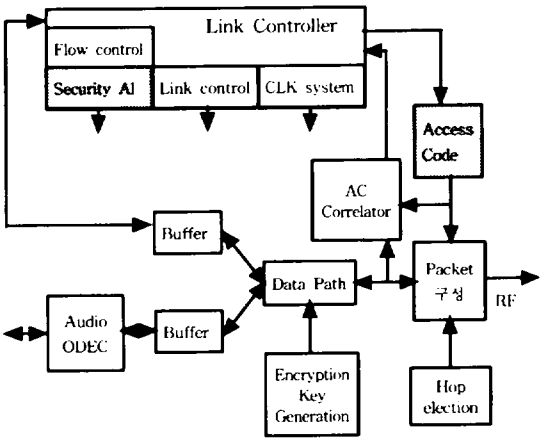


Fig. 1. Baseband/LC architecture.

우선 베이스밴드에서는 물리 채널을 정의하고 이에 대한 호핑(hopping)을 담당한다. 블루투스에서는 대역폭을 1MHz인 RF 채널 79개로 나누고, 각 채널을 초당 1600회 호핑을 하는데, 이 채널 정의와 호핑 시퀀스 선택 등이 모두 베이스밴드에서 이루어진다. 이러한 주파수 홉 설계는 장치들이 전자기 장해가 있는 지역에서도 통신 성능이 고르게 유지될 수 있도록 해 준다.

또 베이스밴드에서는 표준 패킷을 정의하고 생성하는 일을 한다. 표준 패킷은 access code와 헤더(header), payload로 구성되어 있고, 역할 및 링크의 종류에 따라 링크 컨트롤 패킷, ACL 패킷, SCO 패킷으로 나뉘어진다. 패킷과 관련되어서는 에러 정정(error correction) 및 에러 검출(error checking)의 기능도 담당하는데, 사용되는 에러 정정 방법으로는 1/3 rate FEC, 2/3 rate FEC, ARQ의 세 가지가 있으며, 그 사용은 패킷의 종류에 따라 다르다. 에러 검출은 access code, 헤더의 HEC, payload의 CRC 각각에 대해 이루어진다. Fig. 2는 Fig. 1의 data path 내부 구조를 보여주고 있다.

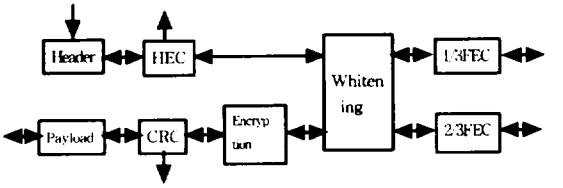


Fig. 2. Data path.

이 밖에도 베이스밴드에서는 보안(security), 링크 설정, 논리 채널(logical channel) 정의, 각 링크의 트래픽(traffic) 관리 및 흐름 제어(flow control) 등의 기능도 이루어진다.

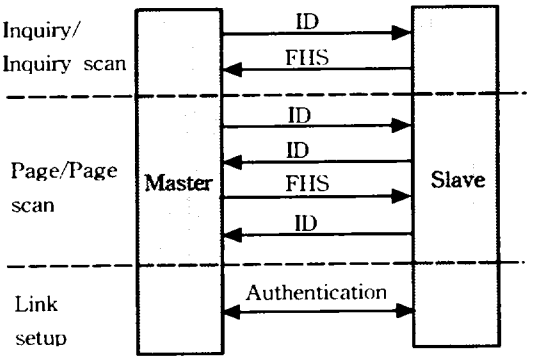


Fig. 3. Message sequence for link setup.

그러나 무엇보다 베이스밴드에서 담당하는 역할 중 중요한 것은 채널 컨트롤이라고 할 수 있다. 채널 컨트롤이란 마스터와 슬래이브 사이에 연결(connection)이 이루어지고 피코넷(piconet)이 구성되는 과정에 관련된 것으로서, 이 과정을 거친 후 마스터와 슬래이브 사이의 데이터 전송이 가능하게 된다. Fig. 3은 이를 위한 두 디바이스간의 메시지 흐름을 보여주고 있다.

III. 홉 시퀀스

3.1. 블루투스의 무선 인터페이스

블루투스의 주파수는 ISM 대역(2.400~2.483.5MHz)

으로, 주파수 호핑 방식의 스펙트럼 확산 방식을 사용한다. 이를 위한 정확한 블루투스 클럭은 Fig. 4와 같다. 채널은 79개의 RF 채널을 통한 의사 랜덤 호핑 시퀀스로, 블루투스 기기 어드레스에 의해 결정된다. 또한 이는 625μs 타임 슬롯으로 나뉘어지고 TDD (time division duplex)방식을 채택하여 시간에 따라 교대로 송수신을 반복하게 된다.

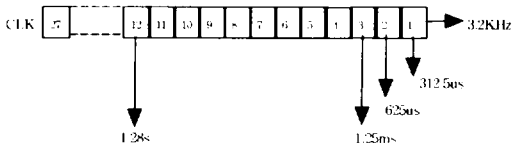


Fig. 4. Blutooth clock.

3.2. 홉 선택의 일반적인 체계

Fig. 5는 79 홉 시스템에서의 홉 선택 커널이다. X는 세그먼트에서 위상을 결정하고 Y1, Y2는 전송 방향을 선택하게 된다. 또한 A~D는 세그먼트내의 순서를, E와 F는 호핑 주파수를 결정하고 커널은 모든 짝수 호핑 주파수 다음에 홀수 호핑 주파수를 작성한다.

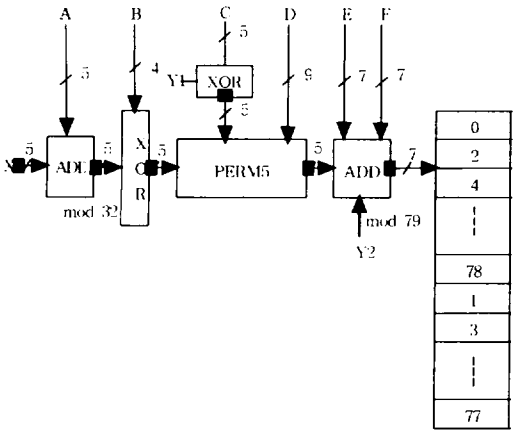


Fig. 5. Block diagram of hop selection kernel.

첫 번째 가산기(ADD)는 세그먼트 내의 위상만을 바꾸는데 이 출력은 배타적 논리합(XOR)에서 B와 연산을 수행하게 된다. 다음 단계의 순열연산(PERM5)은 Fig. 6과 같고 Table 1은 제어 신호 P('1'일 때)의 동작을 보여준다. 여기서 P_{0-8}은 D_{0-8}, P_{9-13}은 C_{0-4} ⊕ Y1 이고 Z는 배타적 논리합의 출력이다.

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}		

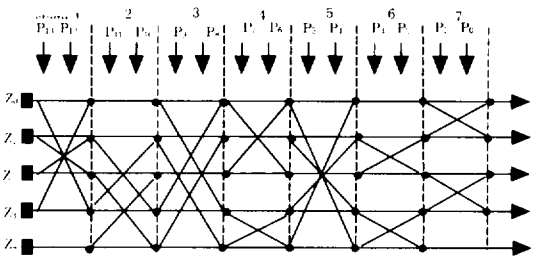


Fig. 6. Permutation operation.

두 번째 가산기(ADD)는 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 싣고 상위에는 짝수 호핑 주파수를, 하위에는 홀수 호핑 주파수를 구성한다.

3.3. 홉 시퀀스 알고리즘

Table 2는 선택커널의 제어워드를 나타낸 것이고 각 상태를 1)~6)에서 설명하였다. 이때 사용되는 클럭의 형태는 다음과 같다.

① $CLK_{27\,0}$: 현재의 피코넷(piconet)의 마스터 클럭
② $CLKN_{27\,0}$: 기기의 고유 클럭
③ $CLKE_{27\,0}$: page된 기기의 고유 클럭에 옵셋을 더한 클럭

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_{page} / X_{inq}	X_{prm} / X_{prs}	CLK_{6-2}
Y1	0	$CLKE_1$/ $CLKN_1$	$CLKE_1$/ $CLKN_1$	CLK_1
Y2	0	32 × $CLKE_1$/ 32 × $CLKN_1$	32 × $CLKE_1$/ 32 × $CLKN_1$	32 × CLK_1
A	A_{27-23}	A_{27-23}	A_{27-23}	A_{27-23} ⊕ CLK_{25-21}
B	A_{22-19}	A_{22-19}	A_{22-19}	A_{22-19}
C	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$ ⊕ CLK_{20-16}
D	A_{18-10}	A_{18-10}	A_{18-10}	A_{18-10} ⊕ CLK_{15-7}
E	$A_{13,11,9,7,5}$	$A_{14,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$
F	0	0	0	16×CLK_{27-7} mod79

1) 호출 주사(page scan)와 조회 주사(inquiry scan) 상태 : 대기중인 기기의 BD_ADDR이 입력 주소로 사용된다.

2) 호출 상태(page substate) : 호출과 대기 기기 사이에 반복되는 부적당한 가능성을 피하기 위해 주기적인 천이가 필요하다. X-입력은 식 (1)과 같고 옵셋값은 식 (2)와 같다.

$$X_{prs} = [CLKE_{16-12} + K_{offset} + (CLKE_{4-2,0} - CLKE_{16-12}) \bmod 16] \bmod 32 \quad (1)$$

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

3) 호출 응답(page response)

가) 슬래이브 응답(slave response)

$CLKN^{*}_{16-12}$은 링크 손실의 가능성을 제거하기 위해 현재 값으로 고정한 것이고 N은 0에서 시작해서 $CLKN_1$이 0이 되는 순간마다 1씩 증가하게 된다. X-입력은 식 (3)과 같다.

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

나) 마스터 응답(master response)

마스터 응답은 클럭값과 현재의 k_{offset} 값이 고정되어야하고, N은 1에서 시작해서 $CLKE_1$이 0이 되는 순간마다 1씩 증가한다. X-입력은 식 (4)와 같다.

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

4) 조회 상태(inquiry substate)

X-입력은 특정 기기가 주소로 지정되지 않았기 때문에 조회 측의 CLKN이 사용되고, 이는 식 (5)와 같이 표현되며 이 때 k_{offset} 은 식 (2)와 같다.

$$X_{i} = [CLKN_{16-12} + k_{offset} + (CLKN_{4-2,0} - CLKN_{16-12}) \bmod 16] \bmod 32 \quad (5)$$

5) 조회 응답(inquiry response)

X-입력은 식 (6)과 같고 N은 FHS 패킷이 전송된 후에 증가한다.

$$X_{rr} = [CLKN^{*}_{16-12} + N] \bmod 32 \quad (6)$$

6) 연결 상태(connection state)

이 상태에서는 마스터의 access code와 클럭을 사용한다. 마스터의 전송은 짝수(CLK_{1-0} =00) 슬롯에서, 슬래이브는 홀수(CLK_{1-0} =10) 슬롯에서 시작된다.

Ⅳ. Access code

access code는 패킷의 가장 앞부분에 붙는 코드로서, 모든 패킷에 포함되어 패킷을 수신했을 때 가장 먼저 체크되는 부분이다. access code의 체크는 수신된 패킷이 자신이 속한 피코넷의 데이터인지, 다른

피코넷의 데이터인지를 구분할 수 있게 해 주며, 따라서 그 값이 일치하지 않는 경우 그 패킷은 버려지게 된다.

일반적인 데이터 패킷은 access code, 헤더 그리고 payload로 구성되나 연결 설정(link setup)을 위한 inquiry나 paging 과정에서 사용되는 ID 패킷인 경우는 access code만으로 이루어진다. 따라서, 이처럼 모든 패킷의 구성 요소로서 사용빈도가 큰 access code의 효율적 구현은 중요한 문제라고 할 수 있다.

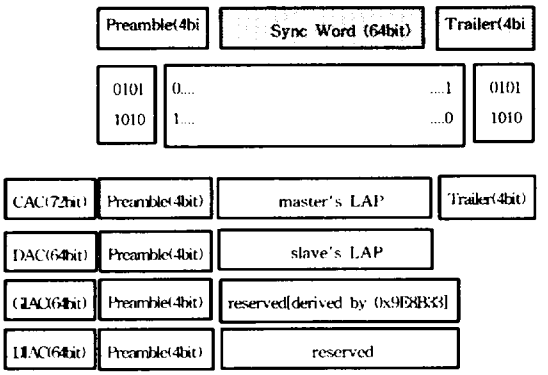


Fig. 7. Access code generation.

Fig. 7은 access code의 생성 과정을 보여주는 것으로서, 그 사용 목적에 따라 사용되는 LAP값이 달라진다.

V. Authentication

케이블에 기반하여 기본적인 보안이 제공되는 유선통신과는 다르게 블루투스는 무선통신이므로 손쉽게 도청이 가능하고, 도청시 인지가 불가능하다는 단점을 갖고 있다. 따라서 블루투스는 원하는 디바이스와의 연결만을 보장하고 전송하고자 하는 정보를 보호하기 위해 page/page scan 과정 후에 authentication 단계를 거치게 된다.

Fig. 9는 이러한 E1 알고리즘의 내부 구조를 보여주고 있다. E1의 입력으로는 링크키, RAND(random number) 그리고 디바이스 주소 값이 들어가고 최종적으로 SRES(signed response)와 ACO (authenticated ciphering offset)가 만들어지게 된다. Fig. 8에서 보여지는 것처럼 SRES의 비교를 통해 두 디바이스가 같은 키를 갖고 있는지, 그리고 그들이 authentication 과정을 성공적으로 수행했는지를 체크하게 된다.

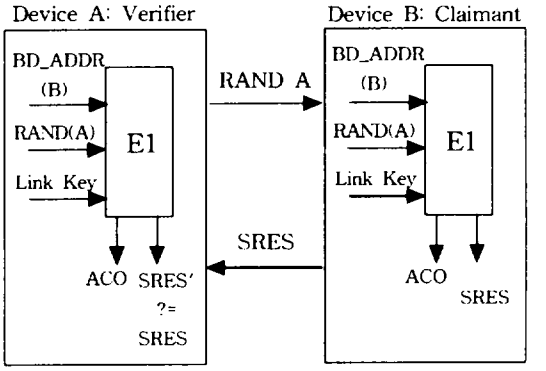


Fig. 8. Description of the authentication process.

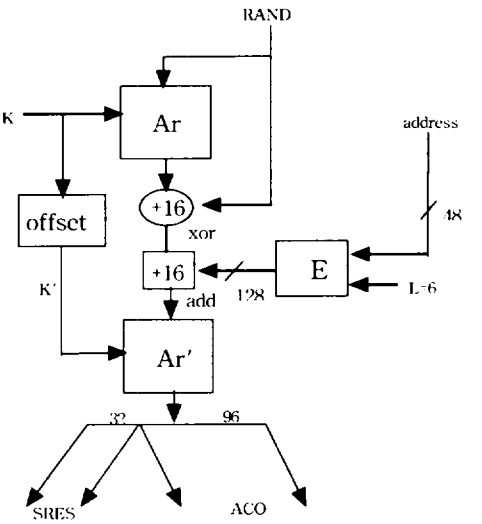


Fig. 9. Flow of data for the computation of E1.

VI. 베이스밴드의 하드웨어/펌웨어 구현 및 검증

베이스밴드는 하드웨어로 구현하는 것이 일반적이다. 그러나 비트열 처리, 암호화(encryption), 암호

화 키 생성, 주파수 호핑(frequency hopping), access code, authentication 등과 같은 많은 베이스밴드의 동작들은 ARM7과 같은 현대의 MCU에 잘 장착되기 때문에 ARM7을 이용한 펌웨어로의 구현이 가능하다.

일반적으로, 하드웨어적인 구현의 장점은 MCU에 대한 부담을 크게 완화시켜 준다는 점이지만 위험이 많고 유연성이 떨어진다는 단점이 있다. 반면 펌웨어로의 구현은 MCU에 부하가 많이 걸린다는 단점을 갖고 있으나 적은 위험성과 유연성, 복잡하지 않은 하드웨어를 제공한다는 장점이 있다.

따라서 본 논문에서는 베이스밴드의 하드웨어/펌웨어 구현의 효율성을 비교하기 위해, 홉 시퀀스와 연결 설정 과정에서 중요하게 다루어지는 ID 패킷의 access code, 그리고 authentication을 하드웨어와 펌웨어로 구현해 보았다.

홉 시퀀스의 하드웨어 구현은 Fig. 5와 Table 2로부터 각 형태의 알고리즘을 기준으로 하드웨어 설계 언어인 VHDL를 이용하여 구현하였다. Fig. 10은 이를 위한 테스트한 환경을 보여주고 있다.

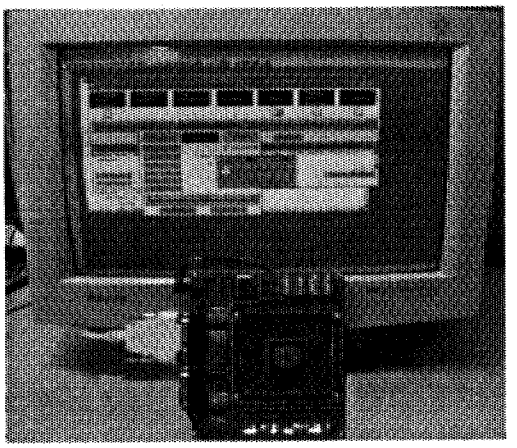
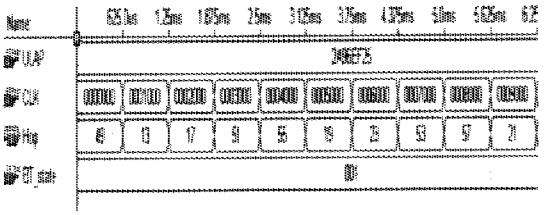
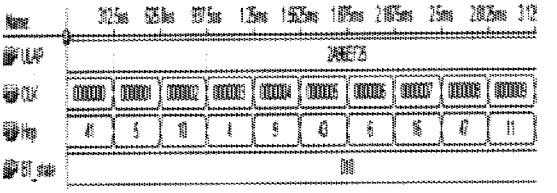


Fig. 10. Baseband test board.

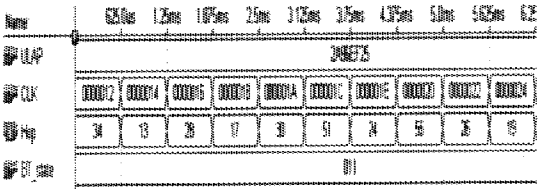
Fig. 11은 블루투스 기기의 고유 주소 ULAP를 "0x2A96EF25"라 가정하여 구현한 시뮬레이션 결과이고 블루투스 SIG(special interest group) 규격의 샘플 데이터와 비교하였다. 구현 결과 홉 시퀀스의 하드웨어 크기는 대략 10,000 게이트임을 알 수 있었다.



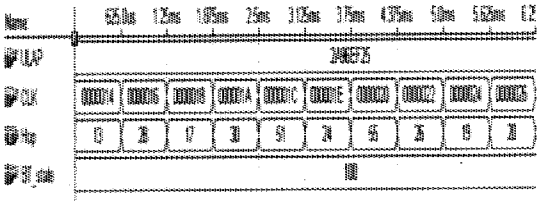
(a) Page scan / Inquiry scan



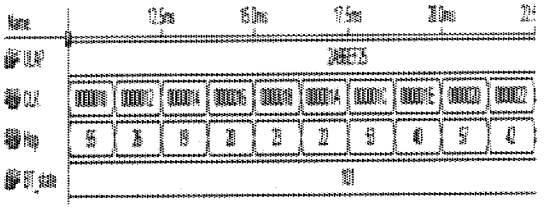
(b) Page state / Inquiry state



(c) Slave page response state



(d) Master page response state



(e) Connection state

Fig. 11. Simulation result of the hardware.

반면 홉 시퀀스의 펌웨어 구현은 MCU에 장착이 잘 되는 소프트웨어 언어인 C프로그램을 사용하였고 이

를 테스트한 환경을 Fig. 12에 나타내었다. Fig. 13은 하드웨어 구현과 같은 조건에서 시뮬레이션 한 결과이고 이 또한 샘플데이터와 비교하였다. 구현 결과 홉 시퀀스의 펌웨어 크기는 대략 28KB임을 알 수 있었다.

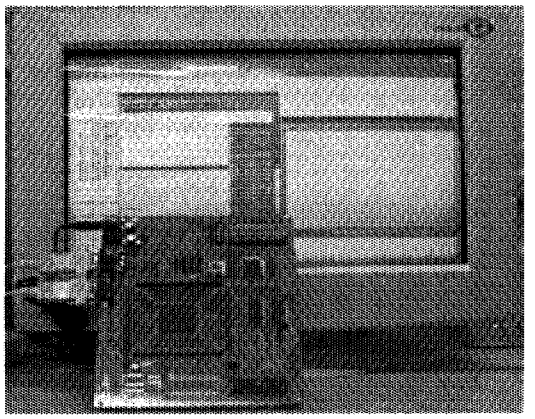
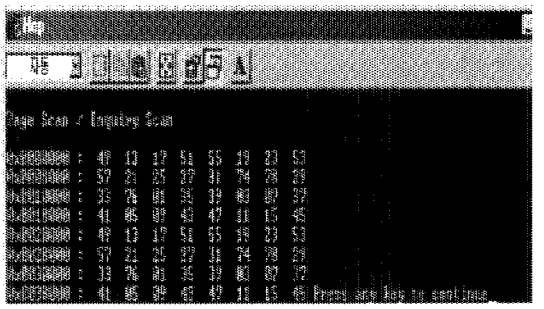
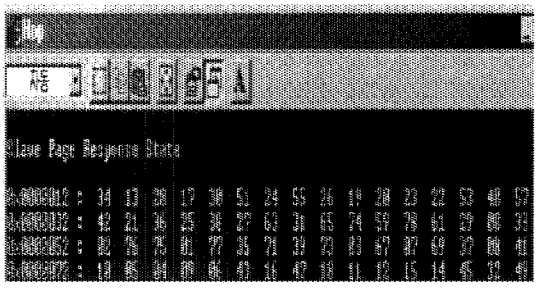


Fig. 12. Firmware test environment (EISC core).

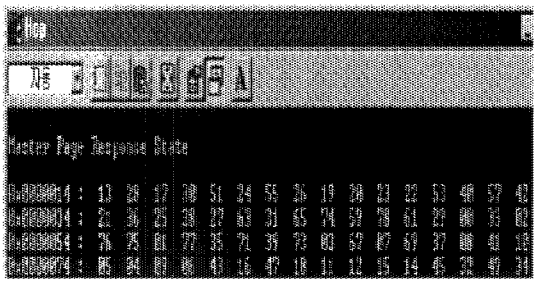


(a) Page scan / Inquiry scan

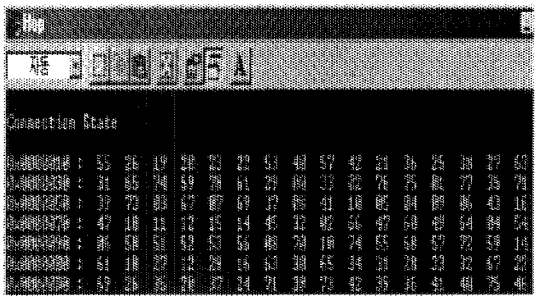
(b) Page state / Inquiry state



(c) Slave page response state



(d) Master page response state



(e) Connection state

Fig. 13. Simulation result of the firmware(a), (b), (c), (d) and (e).

블루투스 클럭은 전원을 켤 때 0으로 리셋되고, 그 후 312.5us를 증가시키면서 자기 발진하는 28비트 카운트이므로 홉 시퀀스는 매시간 계산되는 형태이다. 따라서 현재는 블루투스가 1Mbps의 전송속도를 갖기 때문에 계산 속도가 그리 중요하지 않다고 할 수 있으나 차후 전송속도가 증가될 경우 속도 문제는 매우 중요한 사항이 될 것으로 보인다. 또한 펌웨어로 구현할 경우 MCU 사용율이 높아지게 되고 이로 인해 부하의 위험성이 커진다는 점을 감안한다면 하드웨어 구현이 펌웨어 구현보다 속도와 시간면에서는 유리하

다고 생각할 수 있다. 그러나 홉 시퀀스를 구현 크기 면에서 고려했을 때는 하드웨어 크기가 약 10,000 게이트이고 펌웨어는 약 28KB이다. 이는 블루투스가 소형, 저가의 전제조건을 내세운다는 점을 생각했을 때, 전체 회로의 감소를 위해서는 펌웨어적인 설계가 효율적이라고 할 수 있다.

Fig. 14는 access code를 ASIC으로 구현한 시뮬레이션 결과이고, 구현 결과에서 LAP가 '9E8B33'인 경우 access code 값이 '5475C58CC73345E72A'임을 확인하였다.

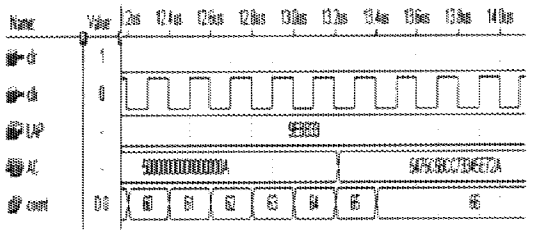


Fig. 14. Simulation result of the access code for ID packet.

반면 Fig. 15는 access code를 C프로그램을 이용하여 펌웨어로 구현한 결과이다.

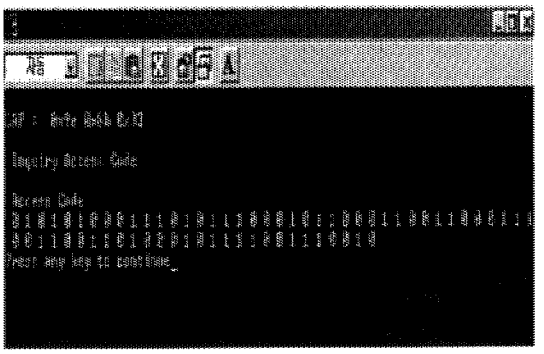


Fig. 15. Result of the access code for ID packet.

그 구현 결과를 보면, ASIC으로 구현했을 때는 전체 크기가 대략 5,000 게이트 정도가 된다. 이는 일반적으로 전체 베이스밴드를 2만~3만 게이트 내외로 구현하는 것에 비추어 볼 때 큰 크기를 차지한다고 볼 수 있다. 이러한 설계는 전체적인 하드웨어의 증가를 가져오고, 블루투스가 저전력, 저비용을 내세운다는 점을 생각할 때 비효율적이라고 할 수 있다. 반면 access code가 펌웨어적으로 구현됐을 때는 4KB 정도로 그 크기가 상당히 줄어들게 되고, 따라서 크기면에서는 펌웨어 구현이 효과적이라고 할 수 있다. 그러나 access code는 패킷이 생성될 때마다 실행되어야 하므로 펌웨어 구현은 상대적으로 MCU에 걸리는 부하가 커지는 결과를 가져오게 된다.

마찬가지로 Fig. 16은 authentication을 ASIC으로 구현한 결과이고 Fig. 17은 펌웨어 상에서의 구현 결과이다. RAND가 '00000000000000000000000000000000', 어드레스가 '000000000000' 그리고 링크키 값 역시 '00000000000000000000000000000000' 라고 했을 때 SRES가 '056C0FE6'이고 ACO는 '48AFCDD4BD40FEF76693B113'임을 보여주고 있다.

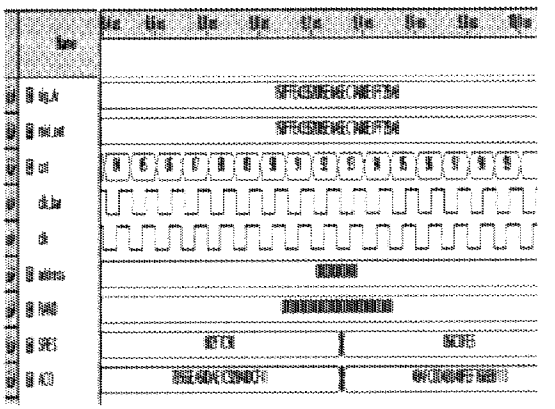


Fig. 16. Simulation result of the authentication.

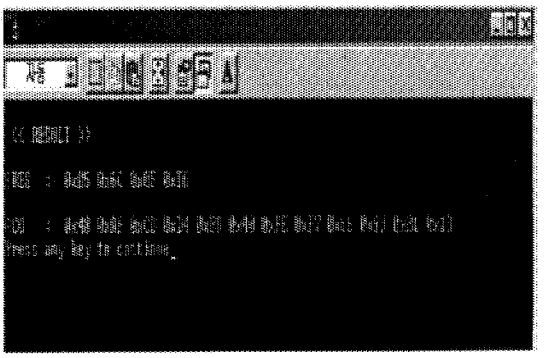


Fig. 17. Result of the authentication.

하드웨어 구현시 전체적인 회로의 크기를 줄이기 위해 기존의 SAFER+ 알고리즘을 두 개의 MUX와 카운터를 사용하여 구현하였다. 그럼에도 불구하고 그 크기가 2만 게이트나 된다. 따라서, 블루투스에서의 authentication이 두 디바이스간의 연결 설정 과정에서 한번만 수행된다는 점을 고려했을 때, 하드웨어적인 구현으로 인한 회로의 증가와 그에 따른 가격의 상승은 비효율적인 설계라고 할 수 있다. 반면 이를 펌웨어로 구현했을 때는 Fig. 17과 같은 결과를 얻을 수 있으며 그 크기는 32KB 정도가 된다. 결과적으로 크기와 빈도수면에서만 생각했을 때는 펌웨어로 구현하는 것이 효율적이라 할 수 있다.

그러나 이러한 크기에 따른 가격상의 비교는 상대적인 것이며, 시장 변화에 따라 메모리와 칩 가격이 달라지기 때문에 그에 따른 개발자의 적절한 선택이 필요하다.

다음으로 크기와 함께 중요한 문제가 구현 속도라고 할 수 있다. 일반적으로는 하드웨어상에서의 구현이 그 속도가 빠르다고 할 수 있으나, 속도와 시간의 문제는 시스템 상에서의 위치와 기능에 따라 그 중요성이 달라진다고 볼 수 있다. 본 논문에서의 access code인 경우는 연속적인 패킷의 전송에 따라 매번 생성되어야 하므로 시간적인 문제 즉 구현 속도가 중요하다고 할 수 있다. 그러나 authentication인 경우는 데이터의 전송과 관계없이 마스터와 슬래이브간의 초기 연결 설정 과정에서 독립적으로 실행되는 부분이므로 구현 속도의 문제는 그리 중요하지 않으며, 펌웨어적인 구현으로 인한 시간의 지연은 별 문제가 되지 않는다. 현재 블루투스가 1Mbps의 저속이라는 점을 감안할 때 access code 또한 펌웨어로 구현한다고 해도 그 지연이 전체 시스템에 미치는 영향은 적을 것으로 보인다.

Ⅷ. 결 론

본 논문에서는 블루투스에서의 홉 시퀀스, access code 그리고 authentication에 대한 전체적인 개념과 생성 방법에 대해 기술하였고, 이를 블루투스 1.1 규격의 절차에 따라 하드웨어와 펌웨어로 구현하였다. 그리고 나서 FPGA로 설계한 베이스밴드 테스트 보드를 통해 ASIC으로 구현한 결과를 확인하였고, ARM7 core와 ESIC core를 이용하여 펌웨어 구현을 테스트하였다. 크기, 구현 속도, 생성 빈도수, 부하 문제를 중심으로 하드웨어와 펌웨어 구현의 장·단점을 비교하여, 베이스밴드를 효율적으로 설계할 수 있는 최적의 구현 분할을 유도하는 하나의 지표를 마련하였다. 향후 홉 시퀀스, access code, authentication 뿐만 아니라 암호화 등도 이러한 과정을 통해 구현 분할을 결정한다면 블루투스가 지향하고 있는 저가(low-cost), 저전력(low -power)의 장점을 더욱 강화하는 결과를 가져올 것으로 기대된다.

참고 문헌

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

2) Nathan J. MULLER, 2000, "BLUETOOTH DEMYSTIFIED", McGraw-Hill Companies, 104pp.

3) Miller and Brent A, 2000, "Bluetooth Revealed", Prentice-Hall,

4) Specification of the Bluetooth System Version 1.1 "Part B : Baseband Specification" pp.47-178, pp.127-138