論文2001-38SD-1-8

고속 페이징 시스템을 위한 FLEX 프로토콜 신호처리기의 구현

(Implementation of a FLEX Protocol Signal Processor for High Speed Paging System)

康 敏 燮 * 李 太 應 **

(Min-Sup Kang and Tae-Eung Lee)

요 약

본 논문은 휴대용 고속 페이징 시스템을 위한 FLEXTM 프로토콜 신호 처리기의 설계 및 FPGA 구현에 관한 것이다. 본 논문에서는 A/D 변환기의 입력 단에서 수신된 interleaved 4-level 비트 심볼 데이터의 동기를 위한 심볼 동기 알고리듬과 (31,21)BCH 부호에 대해 실시간 2중 오류정정이 가능한 개선된 복호 알고리듬을 제안한다. 설계된 프로토콜 신호처리기는 6개의 기능 모듈로 구성되어 있으며, 각 모듈은 VHDL(VHSIC Hardware Description Language)로 모델링을 행하였다. 제안된 프로토콜 신호기는 Axil-320 워크스테이션 상에서 Synopsys™툴을 이용하여 기능 시뮬레이션 및 논리합성(Altera 10K 라이브러리 이용)을 수행하였다. 논리합성 결과 전체 셀의 수는 약 2,631이었다. 또한, 설계된 FPGA 칩의 설계 검증을 위하여 Altera MAX+ PLUS II 상에서 타이밍 시뮬레이션을 수행하였다. PCB 상에서 testbed를 구축한 후, Logic Analyzer를 이용하여 제작된 FPGA 칩의 동작상태를 확인하였고, 실험을 통하여 제작된 칩이 정확히 동작함을 확인하였다.

Abstract

This paper presents the design and FPGA implementation of a FLEX PSP(Protocol Signal Processor) for the portable high speed paging system. In this approach, two algorithms are newly proposed for implementing the PSP which provides capabilities of the maximum 6,400bps at speed, high-channel throughput, real time error correction and an effective frame search function. One is an accurate symbol synchronization algorithm which is applied for synchronizing the interleaved 4-level bit symbols which are received at input stage of A/D converter, and the other is a modified fast decoding algorithm which is provided for realizing double error correction of (31,21)BCH signal. The PSP is composed of six functional modules, and each module is modelled in VHDL(VHSIC Hardware Description Language). Both functional simulation and logic synthesis have performed for the proposed PSP through the use of Synopsys™ tools on a Axil-320 Workstation, and where Altera 10K libraries are used for logic synthesis. From logic synthesis, we can see that the number of gates is about 2,631. For FPGA implementation, timing simulation is performed by using Altera MAX+ PLUS II, and its results will be also given. The PSP which is implemented in 6 FPGA devices on a PCB has been verified by means of Logic Analyzer.

* 正會員, 安養大學校 情報通信・컴퓨터工學部
(Div. of Computer・Electronic Engineering, Anyang University)
** 正會員, (株) 코아테크
(Core Technology Inc.)

※ 본 연구는 정보통신부의 정보통신 연구개발사업의 일환으로 추진된 공동 기술개발사업임. 또한, 반도체설계교육센터로부터 부분적인 지원을 받아 이루어 졌음.
接受日字:2000年3月2日, 수정완료일:2000年12月14日

Ⅰ. 서 론

최근, 개인용 페이징 시스템와 cellular phone들과 같은 휴대 단말기가 전세계의 통신기기 시장을 장악하고 있다. 현재 가장 많이 이용되고 있는 페이징 시스템은 CCIR 권고안 584에 규정한 POCSAG(Post Office Code Standardization Advisory Group)방식을 채용하고 있다[1].

POCSAG 방식을 채용한 페이징 시스템의 통신 속도는 최대 1200 bps(bit per second)로 동작하며, 1개의 채널 당 4-6만명의 가입자를 수용할 수 있다. 그러나 사용자들의 요구가 다양해짐에 따라 POCSAG에 기초한 기존의 저속호출기 구조는 사용자들에게 다양한 서비스 및 최신 정보에 대한 요구를 제공하기에는 충분치 못하다[2].

미국의 모토롤라사에 의해서 제안된 FLEX 프로토콜(protocol)은 전세계 통신 분야를 위한 새로운 시장과 핵심 응용(key application)들의 개발을 위한 개방형 표준으로서 설계되었으며, 프로토콜의 중요한 특징은 다음과 같다[2].

(1) 높은 채널 처리 능력을 가지고 있다. 즉 비동기식인 POCSAG 페이징 시스템에 비해서 FLEX는 동기식 호출 프로토콜이므로 FLEX 호출기는 정확한 시간에 구동하여 원하는 페이지 프레임(page frame)을 쉽게 찾을 수 있다.

(2) 다중 속도 통신(multi-speed communication)을 통하여 시스템 용량을 증가시킬 수 있다. 즉, FLEX 프로토콜은 1,600, 3,200 및 6,400 bps의 세 가지 전송속도를 지원하며, 한 개나 두 개 또는, 네 개의 1,600 bps 전송 채널들을 혼합하여 고속 전송이 가능하게 된다.

(3) 전원 수명은 POCSAG 방식과 비교해서 다섯 배 정도 길다. 즉, FLEX 호출기는 프레임 검색의 초기에 비트 동기화를 확립하고 프레임 수를 체크하여 자신의 프레임 번호들을 인식할 수 있을 뿐 만 아니라, 검색을 위한 사이클도 정확하게 알 수 있다. 따라서 평시에는 수신기의 동작을 중지시키고, 올바른 프레임 번호를 찾았을 때, 전원을 공급하므로써 소비전력을 줄일 수 있다.

본 논문은 고속 페이징 시스템을 위한 FLEX 프로토콜 신호 처리기의 설계 및 FPGA(Field Programmable Gate Array) 구현에 관한 것이다. 본 논문에서는 RF단에서 수신된 데이터와 FLEX 프로토콜 신호처리기와의 정확한 동기를 위한 심볼 동기(symbol synchronization) 알고리듬을 제안한다. 제안된 심볼 동기알고리듬은 '0' 과 '1'의 심볼 주기가 유지되는 동안 두 개의 클럭을 이용하여 각각 적분한 결과 값을 비교하여 A/D 변환기를 위한 클럭 신호를 생성하게 된다.

또한, 프로토콜 신호처리에서 수신된 데이터에 대해서 2중 오류정정 (31,21)BCH 부호의 복호를 위한 개선된 알고리듬을 제안한다. 기존의 복호기는 순회치환회로와 버퍼 레지스터로 구성되므로 복호에 있어서 많은 클럭수는 사이클이 요구된다. 그러나 제안하는 복호 알고리듬에서는 오류위치 탐지를 위한 회로를 조합회로로 설계함으로써 단지 1개의 클럭 사이클 동안에 수행이 가능하다.

6개의 기능 모듈로 구성된 신호처리기는 VHDL (VHSIC Hardware Description Language)[8]을 이용하여 모델링하였고, 논리 검증을 위해 기능 시뮬레이션을 수행하였다. 설계된 신호기는 Synopsys™툴을 이용하여 논리합성(Altera 10K 라이브러리 이용)을 수행하였고, Altera MAX + PLUS Ⅱ 상에서 타이밍 시뮬레이션을 수행하였다.

본 논문은 다음과 같이 구성되어 있다. 2장에 FLEX 프로토콜 및 4-level audio 신호에 대한 기본 개념을 소개한다. 3장에서는 새로운 심볼 동기 알고리듬 및 2중 BCH 복호 알고리듬을 제안하고, 제안한 알고리듬을 기본으로 한 FLEX 프로토콜 신호처리기의 설계에 대해서 기술한다. 4장에서는 FPGA 구현 및 실험 결과를 기술하고, 5장에서는 결론을 내린다.

Ⅱ. FLEX 프로토콜 및 4-level audio 신호

고속 페이징 시스템은 크게 입력신호 수신부(RF receiver), A/D 변환기부(analog to digital converter), 신호처리(디코딩)부(signal processing), 그리고 MCU부(microcontrol unit)의 4개의 블록으로 구성된다[2].

입력신호 수신부에서는 기지국에서 전송된 신호를 수신하여 디지털 신호로 변환하고, A/D 변환기부에서는 아날로그 신호를 디지털로의 변환을 수행하게 된다. 프로토콜 신호처리부에서는 변환된 신호를 분석하여

가입자의 주소와 비교하여 일치하면 정보를 수신하여 복호화를 행하고, MCU부에서는 복호된 정보를 사용자에게 알려주고, 전체 시스템을 제어하는 역할을 수행한다.

1. FLEX 프로토콜의 구조

FLEX 페이징 프로토콜은 정확한 형태의 텍스트 데이터를 제공하기 위하여 모토롤라사에 의해서 제안되었다. FLEX 페이징 시스템은 1,600, 3,200 및 6,400 bps의 전송 속도 중에서 하나를 선택할 수 있다. 이 중 가장 느린 서비스는 2-level FSK 변조(Frequency Shift Keying modulation) (여서는 phase a에 해당됨)를 사용하는 1,600 bps 속도의 단상 정보로 구성되어 있다. 2-level FSK 또는 4-level FSK를 이용하여 3,200 bps의 속도로 전송할 수 있다. 6,400 bps의 최고 속도는 4-level FSK를 사용하며, 1,600 bps에서 a, b, c 그리고 d와 같은 네 개의 동시 위상 정보로 구성된다. 그림 1은 FLEX 프레임의 기본 구성을 나타낸다[2].

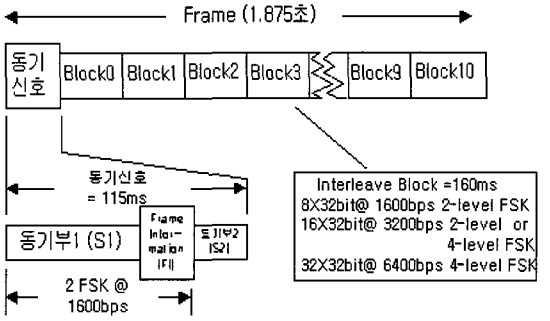


그림 1. FLEX 프레임의 기본 구조
Fig. 1. Basic structure of FLEX frame.

FLEX 프로토콜은 15개의 cycle로 구성되며, 각 cycle은 128개의 프레임(frame)으로 구성된다. 각 프레임은 동기 신호부와 11개의 블록(block)으로 구성되므로 블록에는 0부터 10까지의 번호가 주어진다. 전송속도가 1600 bps의 경우는 1 블록은 8개의 워드(word)로 구성되며, 11개의 블록은 88개의 워드로 구성된다. 3200 bps 및 6400 bps는 phase multiplexing을 행해야 하는데, 이 경우는 각 phase에 대하여 88개의 워드로 구성된다. 이때 프레임을 구성하는 11개의 각 블록은 지정 프레임 속도와 무관하게 송신시간은 160ms를 점유하고 워드 번호와 phase 순 으로 인터리빙하여 송신하게 된다(그림 2 참조). 따라서 채널의 비트 rate가 증가(지정된 송신속도의 증가)함에 따라 각 블록의 정보량도 증가하게 된다. 예를 들어서 1600bps 블록의 경우 256 비트(8 word*32)에서 MD(Multiplex Degree) =1이 되지만, 3200bps에서 1 블록은 512 비트(16 word*32)가 되어 MD=2가 된다[2].

동기 신호부는 동기부1(Sync1), FI(Frame Information), 동기부2(Sync2)로 구성되는데 FI는 4-비트로 표시되는 싸이클 번호와 7-비트로 표시되는 프레임 번호를 포함한다. 각 프레임의 Sync1(S1)은 1600 bps로 송신되는데 프레임의 타이밍 즉, 1600 bps 심볼의 타이밍 및 프레임 이외의 부분에 송신되는 신호의 속도에 대한 정보를 알려준다. 또한, sync2(S2)는 FI에서 정의된 블록의 속도에 따라 동기되어 입력되는데, message block의 다중화 신호를 정확하게 분리하여 디코딩하기 위해서도 사용된다[2].

2. 4-level 인터리빙 심볼

RF단으로부터 입력된 4-level audio 신호는 2-비트 A/D 변환기에 의해서 2-level의 디지털 신호로 변환된다. 변환된 신호는 FLEX 프로토콜 신호처리기내의 심볼 동기회로의 입력단자인 EXTS0와 EXTS1로 인가된다.

그림 2는 4-level audio 신호(0, 1, 2, 3) 및 인터리빙된 비트를 나타내며, 이 신호는 FLEX 페이징 시스템의 동기 신호부의 입력으로 사용된다[2].

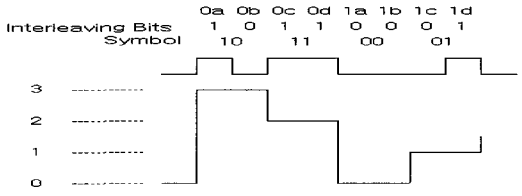


그림 2. 인터리빙 비트 및 심볼(4-level)
Fig. 2. Interleaving bits and symbol(4-level Signal).

그림 2에서 워드 0a, 0b, 0c, 0d,...,7a, 7b, 7c, 7d로 명명된 모든 워드는 row starting 하는 형태로 배치된 BCH(31,21) code+even parity 이다. 이진 비트 스트림은 직접변조(binary FSK) 이거나 또는 3200 및 6400bps의 4-level의 경우는 처음에 2-비트가 A/D 변환기를 통하여 입력된다. 입력된 심볼의 변환은 블록의

처음 2-비트가 대응하는 4-level 심볼로 된다. 3200bps 4-level의 처음의 비트 0a는 심볼의 MSB이며 0c는 LSB로 한다. 6400bps 4-level의 처음의 2-비트의 비트 0a는 심볼의 MSB이며 0b는 LSB로 한다[2].

Ⅲ. FLEX 프로토콜 신호처리기의 설계

FLEX 프로토콜 신호처리기는 수신된 아날로그 신호를 이해할 수 있는 데이터 형식으로 바꿔 주는 기능을 수행한다. FLEX 프로토콜 신호처리기는 수신기를 단계별로 워밍업하거나 동작을 중지시키기 위한 8개의 제어 라인을 통하여 RF 단을 제어한다. 그리고 2 비트 A/D 변환기의 출력인 디지털 데이타(EXTS0, EXTS1)를 76.8KHz의 시스템 주파수에 따라 처리하는 역할을 수행한다. 이 신호처리기는 38.4 KHz의 CLKOUT 신호를 발생하여 MCU와 2 비트 A/D 변환기를 동기화시킨다. 그림 3은 FLEX 프로토콜 신호처리기에 대한 블록도를 나타낸다.

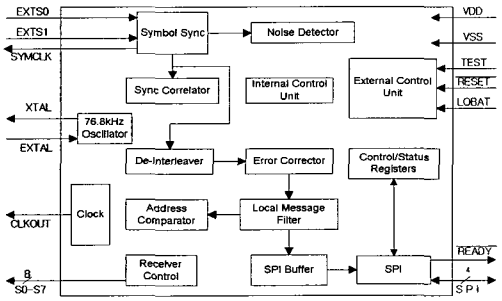


그림 3. 고속 프로토콜 신호처리기의 블록도
Fig. 3. Block diagram for FLEX protocol signal processor.

이 신호기는 크게 심볼 동기 신호부, deinterleaver부, BCH 오류 정정부, filtering부, 내부 제어부, 그리고 SPI(Serial Peripheral Interface)부 등의 6 블록으로 구성된다.

1. 심볼 동기 신호부

심볼 동기회로는 외부에서 수신된 데이터와 FLEX 신호처리기와의 정확한 동기화를 이루기 위해서 필요하다. 심볼 동기회로의 입력신호는 A/D 변환기를 통해 2-level로 변환된 디지털 신호이며, 동기회로의 입력단인 EXTS0와 EXTS1으로 인가된다.

다양한 심볼 동기기법 중에서 가장 많이 사용되고 있는 것은 Early/Late gate synchronizer 기법이다[3-4]. 이 방법은 기본 원리는 한 심볼 주기 동안 각 반 주기의 에너지 수준(energy level)을 비교하는 것이다. 제안된 심볼 동기 알고리듬은 '0' 과 '1'의 심볼 주기가 유지되는 동안 두 개의 클럭을 이용하여 각각 적분한 결과값을 비교하여 A/D 변환기를 위한 클럭신호를 생성한다.

(1) 심볼 동기 알고리듬

제안된 심볼 동기 알고리듬의 기본 개념(basic concept)는 '0' 과 '1'의 심볼 주기가 유지되는 동안에 주 클럭(main clock)으로부터 분주된 2개의 클럭을 이용하여 각각 적분한 결과값을 비교하는 것이다. 즉, Early/Late gate accumulator(그림 5에 나타난 블록)에서 적분을 수행하기 위해 각각 두개의 Sample 클럭을 사용하게 되는데, 제안된 알고리듬은 다음과 같이 4단계에 의해서 수행된다.

[단계 1] 입력되는 '0', '1'의 심볼들과 영역에 대한 비교 및 적분을 수행하기 위하여 2개의 Sample 클럭을 생성한다. 즉, 주 클럭(76.8 KHz)을 2분주한 신호인 2개의 Sample 클럭(38.4 KHz)을 각각 Sample_clk1과 Sample_clk2라 정의하고, 이 두 클럭을 각각 Early Gate 와 Late Gate라 부른다.

[단계 2] 위상 오류가 위상 한계(최대 4-비트)를 초과하는지를 check하기 위해서 2개의 Sample 클럭들을 이용하여 적분값을 서로 비교한다. 예를 들어, 입력 데이터가 0에서 2개의 Sample 클럭을 이용하여 2개의 적분값인 Falling_window_count1과 Falling_window_count2의 값을 비교하는 경우는 그림 4와 같다.

[단계 3] 이 두 클럭을 이용하여 입력 데이타가 각각 '1'과 '0'인 두 개의 적분값을 비교할 때 발생가능한 4가지 경우를 체크해야 한다.

(경우 1) Sample_clk1과 Sample_clk2 사이에 심볼이 전송되고, 심볼의 event가 이 두 클럭 가운데에서 발생했을 경우 적분값이 같으면, 타이밍 교정할 필요가 없다(그림 4의 ①).

(경우 2) 각 Sample_clk1과 Sample_clk2사이에 동일한 심볼이 전송될 경우, 적분값은 동일하게 되는데, 이것은 2개의 클럭안에서 심볼 변화가 발생되지 않은 것을 의미한다. 따라서 이 조건에서는 타이밍 교정이 필요 없다(그림 4의 ④).

(경우 3) Sample_clk1에서의 적분값(Octal 7)이 Sample_clk2에서의 적분값(Octal 6) 보다 클 경우는 심

볼 주기의 Sample_clk2에서 심볼 변화가 있다는 것을 의미한다. 이 경우 심볼 영역 의 동기를 맞추기 위해 타이밍 교정이 필요하다(그림 4의 ⑤).

(경우 4) 물론 (경우 3)과 반대의 경우, 즉 Sample_clk2에서의 적분값이 클 경우에도 타이밍 교정이 필요하게 된다.

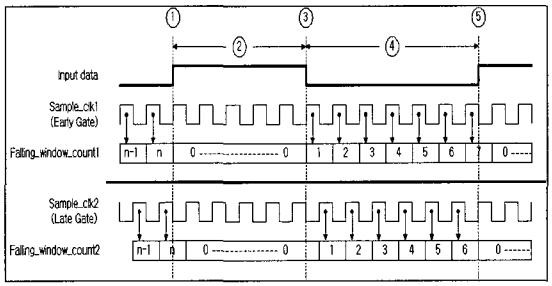


그림 4. 2개의 적분값 비교에 대한 예
Fig. 4. Comparison of two integral values.

[단계 4] 비교된 결과에 대해서 위상차를 조절하고, 오류를 체크하여 심볼 클럭을 생성한다. 이때 생성된 심볼 클럭은 A/D 변환기의 입력으로 feedback되어 A/D 변환기의 인터리빙된 입력신호를 동기화시키는데 사용된다.

(2) 심볼 동기회로의 설계

그림 5는 제안된 심볼동기 알고리듬을 기본으로 한 심볼 동기회로의 블록도를 나타낸다. 동기 회로는 크게 위상 검출기(phase detector), 루프 필터(loop filter), 그리고 Digitally Controlled Oscillator(DCO)로 구성된다.

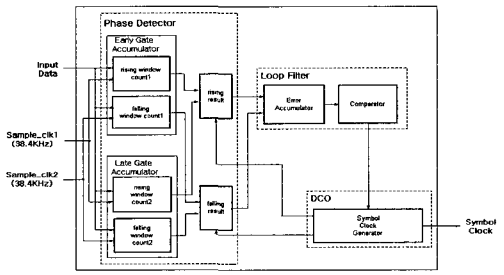


그림 6. 제안된 심볼 동기 회로의 블록도
Fig. 5. Block diagram for proposed symbol synchronizer.

위상 검출기는 Early/Late gate인 두 개의 적분기 설계를 말한다. EXTS0와 EXTS1단자에서 출력되는 입력신호와 주 클럭(76.8KHz)을 2분주한 2개의 Sample_clk1(38.4KHz)과 Sample_clk2(38.4KHz)를 받아들여 Early/Late Gate Accumulator의 rising_window_count1과 count2에서 1의 상태에 대한 적분을 수행하고, falling_window_count1과 count2는 0의 상태에 대한 적분을 수행한다(그림 4 참조). 적분된 결과값은 각각 rising과 falling 결과에 입력되어 심볼 클럭과 데이터 열 사이의 위상 차와 오류를 출력한다. Early와 Late gaate 적분기 사이의 차는 위상 오류(θe)와 비례한다. 위상 검출기의 출력은 $\theta e = |\Sigma early| - |\Sigma late|$ 와 같이 정의된다.

Loop Filter의 Error Accumulator는 위상 검출기로부터의 위상 오류의 출력이 누적된다. Early/Late gate에서 발생하는 위상 오류의 갯수가 동일할 경우 DCO에서 발생되는 심볼 클럭에는 아무런 변화가 없으며, 양단의 위상 오류 발생 갯수의 차(difference)만이 심볼 클럭에 대한 위상 오류의 원인으로 작용하게 된다. Comparator는 위상 오류가 위상 한계(최대 4-비트)를 초과하는지를 검사한다.

DCO는 위상 오류가 위상 한계를 초과했을 때, 내부에서 발생된 심볼 클럭의 위상을 조절한다. 또한, DCO의 출력은 다시 Phase Detector의 rising과 falling 결과에 입력되어 심볼 클럭과 데이터 열 사이의 위상 차와 오류를 조절한다.

2 BCH 오류 정정부

(1) 기존의 2중 BCH 복호 알고리듬

2원 BCH부호는 유한체 GF(2) 상의 원소를 기호로 갖는 부호로서 부호의 구성은 확대체 GF(2^m)을 토대로 이루어지며 복호는 부호의 구성시 부여된 대수적 구조를 통해 이루어진다[6]. 먼저 수신어 r(x)로부터 오증(syndrome)을 계산한다. 오증요소 s_k는 r(x)를 α_k($1 \le k \le 2t$)의 최소 다항식 $m_x(x)$로 나누어서 구할 수 있다. 여기서 오증요소 s_k을 이용하면 오류가 발생한 수신어에서 오류위치번호를 찾아낼 수 있는 오류위치다항식 $\sigma(x)$을 구한다. 일반적으로 t중 오류를 정정할 때 필요한 오류위치다항식은 식 (1)과 같이 표시되는데, 여기서 σ_0=1이다.

$$\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots \sigma_t x^t = \sum_{j=0}^{t} \sigma_j x^j \qquad (1)$$

다음 과정으로 $\sigma(x)$의 근을 구하기 위해 $\sigma(x)$의 계수를 구하며, 오류탐지회로를 이용하여 오류위치번호를

결정한다. 그리고 오류위치번호의 비트를 정정하면 오류정정된 수신어를 얻을 수 있다. 그림 6은 기존의 2중 오류정정 BCH 부호의 복호기를 나타낸[3,6].

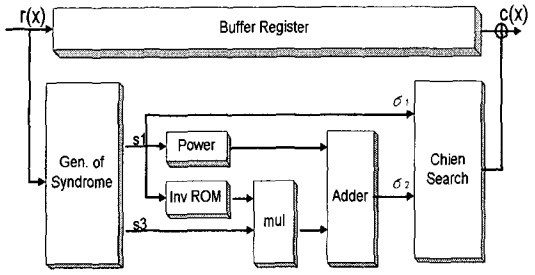


그림 6. 기존의 BCH 부호 복호기의 블록도
Fig. 6. Block diagram of conventional BCH decoder.

그림 6에서 Gen. of Syndrome 블록은 오증 요소 s_1과 s_3를 계산하기 위한 회로이며 Inv ROM, Power ROM, mul, Adder 블록은 $\sigma(x)$의 계수를 구하기 위한 역원 ROM, 자승기, 곱셈기, 덧셈기를 나타낸다[4, 6]. Chien Search 블록은 $\sigma(x)$의 근을 구하여 오류위치를 판별하기 위한 회로이다. 이 회로는 순회치환회로로써 버퍼 레지스터와 순회치환회로를 사용하여 모든 위치의 오류를 순회적으로 정정한다. 그러나 이 회로는 이 과정에서 부호장이 n인 t중 오류정정 BCH부호의 경우 3tn의 클럭 사이클이 요구된다[5-7].

(2) 개선된 2중 BCH 복호 알고리듬

여기에서는 그림 6에서 설명한 기존의 방법을 개선한 2중 오류정정 (31,21)BCH 부호의 복호 알고리듬에 관하여 기술한다. 제안한 (31,21)BCH 부호의 2중 오류정정 알고리듬은 다음과 같은 절차로 수행된다.

먼저 2중 오류정정을 위해 오증요소 s_1, s_2를 구한다. 즉, 부호 생성다항식 $g(x) = m_1(x)m_3(x)$이므로, 오증요소 s_1은 식 (2)와 같이 $r(x)$를 $m_1(x)$로 나누었을 때의 나머지 $s_1(x)$에 $m_1(x)$의 근 α를 대입하여 얻어진다. 오증 요소 s_3의 경우 같은 방법을 적용하면 식 (3)과 같이 구할 수 있다.

$$s_1 = [q(x)m_3(x)]\, m_1(x) + s_1(\alpha) = s_1(\alpha) \quad (2)$$

$$s_3 = [q(x)m_1(x)]\, m_3(x) + s_3(\alpha) = s_3(\alpha) \quad (3)$$

다음으로 오류위치 다항식 $\sigma(x)$를 구한다. 여기에서는 2중 오류정정을 고려하기 때문에 $\sigma(x)$를 2차식으로 표현하면 식 (4)와 같다.

$$\sigma(x) = 1 + s_1x + (s_1^2 + s_3/s_1)x^2 \quad (4)$$

식 (4)에서 계수 σ_1은 오증 s_1과 동일하므로 쉽게 구할 수 있지만 계수 σ_2를 구하기 위해서는 오증 s_1을 제곱하기 위한 자승기, 오증 s_1의 역원(inverse element) s_1^{-1}을 구하는 역원계산기, s_3와 s_1^{-1}을 곱하기 위한 승산기 그리고 s_1^2과 s_3/s_1을 더하기 위해 exclusive-or 게이트로 이루어진 덧셈기가 필요하다. 본 논문에서는 복호 싸이클 수를 줄이기 위하여 모든 연산기를 조합 논리회로로 구성해야 한다.

(1) 자승기 및 역원 계산기 구현

유한체 $GF(2^5)$상의 다항식 표현 및 벡터표현을 이용하면 ROM으로 쉽게 구현할 수 있다.

(2) 승산기 구현

승산기의 구성식을 유도하기 위해 $GF(2^m)$상의 임의의 두 원소 A와 B가 주어졌을 때 두 원소의 곱 Z는 식 (5)와 같이 정의된다.

$$Z = A \cdot B = A \cdot \sum_{i=0}^{m-1} b_i \alpha^i = \sum_{i=0}^{m-1} b_i (A\alpha^i) \quad (5)$$

단, 두 원소 A와 B는 다음과 같이 정의한다.

$$A = a_0 + a_1\alpha + \ldots + a_{m-1}\alpha^{m-1} = \sum_{i=0}^{m-1} a_i \alpha^i$$

$$B = b_0 + b_1\alpha + \ldots + b_{m-1}\alpha^{m-1} = \sum_{i=0}^{m-1} b_i \alpha^i$$

본 논문에서는 (31,21)BCH부호를 사용하므로 원시다항식 $p(x) = 1+x^2+x^5$인 $GF(2^5)$상에서의 임의의 두 원소 A, B간의 승산식은 식 (5)에 대하여 행렬을 구한 후 이를 이용하면 식 (6-10)을 유도할 수 있다.

$$z_0 = a_0b_0 + a_4b_1 + a_3b_2 + a_2b_3 + (a_1 + a_4)b_4 \quad (6)$$

$$z_1 = a_1b_0 + a_0b_1 + a_4b_2 + a_3b_3 + a_2b_4 \quad (7)$$

$$z_2 = a_2b_0 + (a_1 + a_4)b_1 + (a_0 + a_3)b_2 + (a_2 + a_4)b_3 + (a_1 + a_3 + a_4)b_4 \quad (8)$$

$$z_3 = a_3b_0 + a_2b_1 + (a_1 + a_4)b_2 + (a_0 + a_3)b_3 + (a_2 + a_4)b_4 \quad (9)$$

$$z_4 = a_4b_0 + a_3b_1 + a_2b_2 + (a_1 + a_4)b_3 + (a_0 + a_3)b_4 \quad (10)$$

식 (6-10)을 이용하면 $GF(2^5)$상의 승산기를 조합논리회로에 의해서 쉽게 구현할 수 있다.

다음 단계로 오류위치 다항식의 근을 구한다. 본 논

문에서는 $\sigma(x)$의 근을 구하기 위하여 대입법(substitution method)을 이용하며, 이것은 식 (11)과 같이 $\sigma(\alpha^k)=0$이 되도록 GF(2^m)의 모든 원소 α^k를 대입하는 것이다.

$$\begin{aligned}
&\sigma(1)=1+\sigma_1+\sigma_2+\cdots+\sigma_t\\
&\sigma(\alpha)=1+\sigma_1\alpha+\sigma_2\alpha_2+\cdots+\sigma_t\alpha^t\\
&\sigma(\alpha^2)=1+\sigma_1\alpha_2+\sigma_2(\alpha^2)^2+\cdots+\sigma_t(\alpha^2)^t\\
&\qquad\vdots\ \alpha^{2^m-2}\\
&\sigma()=1+\sigma_1\alpha^{2^m-2}+\sigma 2(\alpha^{2^m-2})^2+\cdots+\\
&\quad \sigma t(\alpha^{2^m-2})^t \qquad (11)
\end{aligned}$$

여기에서 오류위치번호 α^{n-j}, $0\le j\le n-1$은 $\sigma(x)$의 근인 α^j의 역이므로, (31, 21)BCH부호의 경우 오류형태 e(x)는 식 (11)과 같이 표현할 수 있다.

$$e(x) = \sigma(\alpha^{30}) + \sigma(\alpha^{29})x + \cdots\cdots + \sigma(1)x^{30} \qquad (12)$$

또한, 부호어 c(x)=r(x)+e(x)이므로 (31, 21)BCH 부호의 경우에 식 (11)과 같이 $\sigma(x)$의 x에 α^k ($0\le k\le 2^m-2$)를 대입하여 근을 구하는 회로를 조합논리회로로 구현할 수 있다. 이것을 31개 사용하면 동시에 e(x)의 모든 계수를 구할 수 있으므로 r(x)와 Exclusive-or 연산하면 1 클럭 사이클 동안에 복호를 수행할 수 있다.

조합논리회로를 이용한 연산기는 회로연결(wire routing)이 불규칙적이고 GF(2^m)상의 m이 커지면 회로가 복잡해지는 단점을 가진 반면, 연산에 소요되는 지연시간이 매우 짧은 장점을 가지고 있어 m=5인 (31,21)BCH부호의 복호기에 적합하다.

(2) BCH 오류 정정부의 설계

일반적인 BCH 부호의 복호기는 앞에서 기술한 것처럼 복호에 있어서 많은 클럭 사이클을 요구하는 순회치환회로와 버퍼 레지스터로 구성된다[3, 6] 여기에서는 복호기의 연산부분을 완전한 조합회로로 설계하여 처리속도를 개선한 2중 오류정정 (31,21)BCH 복호기 설계에 관하여 기술한다.

그림 7은 제안된 복호 알고리듬을 이용하여 설계된 개선된 복호기의 블록도를 나타낸다. 먼저 31-비트 r(x)가 입력되면 오증 생성기(Gen. of Syndrome)를 거쳐 오증요소 s_1, s_3를 구한다. 다음으로 오류위치다항식 $\sigma(x)=1+s_1x+(s_1^2+s_3/s_1)x^2$의 2차항의 계수를 구하는 σ_2 Calculation BLK을 거쳐 계수 σ_2를 구한다. 여기에서 Inv ROM 블록은 s_1의 역원을 구하고, Power ROM 블록은 s_1의 자승을 구하며, mul 블록은 s_3와 s_1의 역원과의 승산기이다. 그림 7에서 알 수 있듯이 Power ROM과 Inv ROM의 크기는 공히 2^5*6이 된다.

다음 단계는 대입법을 이용하여 $\sigma(x)$의 근을 구함으로써 오류위치 번호 β_i, $0\le i\le n-1$을 결정(Error Decision Circuit)하여 31-비트 E(x)를 구성(Gen. of E(x))한다.

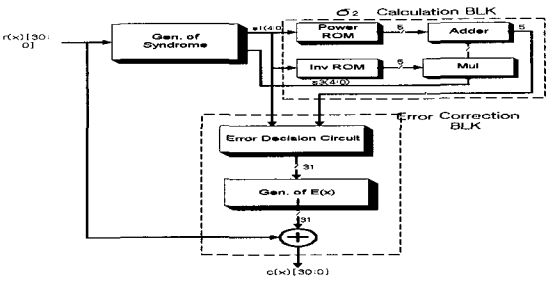


그림 7. (31, 21)BCH부호의 개선된 복호기 블록도
Fig. 7. Block diagram of modified decoder of(31, 21)BCH code.

마지막으로 E(x)와 31-비트로 입력받은 r(x)를 exclusive-or하여 오류정정된 c(x)를 출력한다. 제안된 복호기는 오류 정정과정에서 많은 클럭 사이클이 요구되는 순회치환회로와 버퍼 레지스터를 사용하는 기존의 방법과 달리 모든 연산기를 조합논리회로로 설계하므로써 고속 연산이 가능하다.

3. Deinterleaver 부 및 Filtering 부

Deinterleaver부는 FLEX 프로토콜에 따라 인터리빙되어 입력되는 신호를 다시 디인터리빙 하는 역할을 수행하며, 쉬프트 레지스터로 구성된다. 다중화된 phase가 복수가 아닌 경우는 가변 수신 싸이클, frame 계속, 및 복수회 송신을 제외하고 각 phase의 내용은 서로 독립적이다. 페이저는 ID-ROM에 할당된 0, 1, 2 또는 3의 phase 값을 갖고 있다. 예를 들어 6400 bps에서는 할당된 phase 값이 0이면 'a', 1이면 'b', 2이면 'c', 3이면 'd'를 수신한다. 또한, 3200 bps에서는 phase 값이 0과 1은 'a', 2와 3은 'c'를 수신한다. 수신기의 phase 지정방법은 ID-ROM에 의해 규정되고 기지국측 설비는 이와 같은 규정에 맞는 phase로 송신한다.

Filtering부의 주요기능은 디코더 내부의 BCH 오류정정부에서 오류 정정된 32 비트 자료를 프로토콜의

순서에 맞게 FLEX 프로토콜에 따라 주소, 벡터, 메시지 등의 패킷으로 변환하여 호스트(MCU)에 전달하기 위해 SPI 버퍼로 전송하는 블록이다[2]. 페이징 시스템은 사업자가 정한 고유한 캡 코드(Cap address)를 가지고 있으며, 신호처리기는 수신한 데이터가 자신의 캡 코드와 일치하는 지를 검사하여 일치하는 경우에만 데이터 필터링을 수행하게 된다.

Filtering 블록은 크게 카운터의 블록, 주소 비교 블록, 주소 카운터, 주소 패킷 필터링 블록, 벡터&메시지 패킷 필터링 블록으로 나눌 수 있다. 카운터의 블록은 BCH 오류정정부로 부터 전송되어진 32 비트의 자료를 하나씩 받을 때마다 내부에서 워드를 1씩 증가시키는 카운터 역할을 수행한다. 제어신호인 word_st를 입력으로 하여 카운트가 시작되며, PS(Phase Select)에 따라 각 전송속도에서 프로토콜 신호기가 디코드 해야 할 phase(a, b, c, d)가 결정된다(표 1).

표 1. Phase의 선택
Table 1. Selection of Phase.

PS 값		디코드된 Phase(bps)		
PS1	PS0	1600	3200	6400
0	0	a	a	a
0	1	a	a	b
1	0	a	c	c
1	1	a	c	d

주소 비교 블록에서는 내부 버퍼에 임시 저장되어 있는 주소와 DATA_IN으로부터 들어오는 주소와 비교하여 주소 필터링 블록으로 전송한다. 주소는 모두 그 기능별로 6가지 type의 종류[2]를 가지며, 각각의 고유한 범위를 가지게 된다. 주소는 1개의 워드를 가진 짧은 주소(short address)와 2개의 워드로 구성된 긴 주소(long address)로 구성된다.

주소 카운터 블록은 지정된 주소에 정확히 대응하여 위치하는 벡터의 위치를 찾기 위한 주소 카운터이다. 주소 패킷 필터링 블록은 비교된 주소를 받아 호스트에 전송하기 위한 32 비트의 주소 패킷을 생성해 주는 블록이다.

벡터 및 메시지 필터링 블록에서는 주소에 대응하는 벡터 field부분의 자료를 전송받아서 블록 내부에서 32 비트의 패킷으로 세팅해서 버스로 보내준다. 벡터 필터링에서 주소가 여러 개일 경우에는 대응되는 복수의 벡터가 존재하므로 주소에 대응하는 벡터 번호를 포함하고 있어야 한다.

4. SPI 부 및 내부 제어부

SPI의 주요기능은 신호처리기 내부에서 32-비트씩 병렬로 처리된 자료를 1-비트씩 MCU로 전송하고, 반대로 MCU로부터 1-비트씩 직렬로 전송되어진 자료를 32-비트 병렬로 바꾸어서 신호처리기로 전송하는 블록이다[2]. 또한 신호처리기에서 MCU로 전송할 때 동기를 맞추기 위해 내부적으로 버퍼를 두어서 자료를 저장한다. 그리고, 신호처리기는 외부로부터 신호를 받아 어떠한 자료를 선택하여 MCU로 보내어 줄지를 결정한다. 이를 위해서 SPI내부적으로 제어할 수 있는 블록이 필요하고 자료를 선택적으로 내보내 줄 수 있는 기능이 필요하다. MCU와 신호처리기간의 정확한 송수신을 위해서 hand-shaking이 필요하다.

내부 제어부에서는 신호처리기에서 요구하는 각 패킷들에 대한 제어 신호와 호스트인 MCU의 양방향 통신을 위한 제어를 수행한다. 외부에서 수신되는 신호와 호스트에서 전달되는 신호에 의해서 신호처리기의 전체 회로가 동작하게 된다.

신호처리기의 내부 제어부는 신호처리기에 대한 전반적인 제어신호를 각각의 블록으로 전달하여 필요한 처리를 수행하도록 하며, 여기서 동기 신호부에 대한 프레임 신호를 전달하여 신호처리기가 수신하여야 할 신호의 위치를 판단하게 된다. 내부 제어부는 크게 9개의 부분 블록으로 구성되며, 호스트에서 송신된 데이터 패킷을 저장한다. 신호처리기에서 호스트로 임의의 패킷을 보낼 경우 패킷정보들은 처음에 MSB부터 보내야 한다.

Ⅳ. FPGA 구현 및 실험결과

본 논문에서 제안된 프로토콜 신호처리기는 크게 6개의 기능 모듈로 구성되어 있으며 Axil-320 워크스테이션 상에서 VHDL(VHSIC Hardware Description Language)을 이용하여 설계하였다. Front-end 설계에서는 Synopsys™툴을 이용하였고, 논리합성은 Altera 10K 라이브러리를 이용하였다. 표 2는 6개의 블록에 대해서 합성된 결과를 나타낸다.

표 2. 주요 블록의 합성 결과
Table 2. Synthesis results.

각 기능 모듈	셀의 수
심볼 동기 신호부	371
Deinterleaver 부	587
BCH 오류 정정부	600
Filtering 부	457
내부 제어부	375
SPI 부	241
전체 셀의 수	2,631

테스트_벡터를 구성한 다음 Altera MAX+PLUS II 상에서 각각의 1600, 3200, 6400 bps에 대하여 타이밍 시뮬레이션을 수행하였다. 그림 8과 그림 9는 각각 ALTERA™의 타이밍 시뮬레이터를 이용하여 심볼 동기신호부와 BCH 오류 정정부에 대한 타이밍 시뮬레이션결과를 나타낸다.

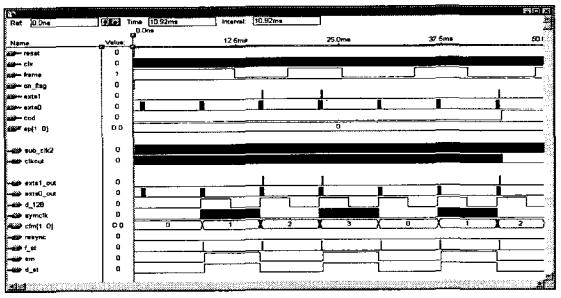


그림 8. 심볼 동기회로에 대한 타이밍 시뮬레이션 결과
Fig. 8. Timing simulation result for symbol synchronizer.

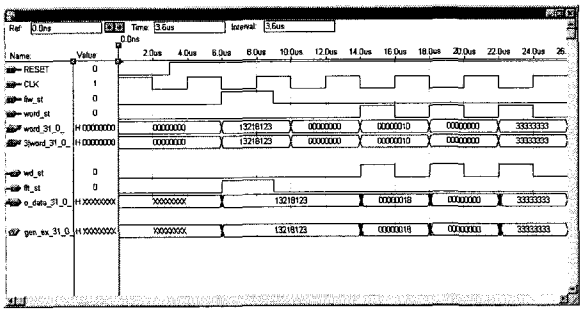


그림 9. BCH 오류정정부의 타이밍 시뮬레이션 결과
Fig. 9. Simulation result for BCH Error Correction Part.

BCH 오류 정정부에서는 Deinterleaving부에서 전달되어진 신호를 받아 BCH 오류정정을 수행한 후 결과인 32 비트의 co_data를 flt_st신호에 의해 Filtering부로 전달한다. 그림 9에서 word_31_0는 에러 비트가 포함된 원시 코드이며, wd_st신호에 따라 복호를 시작하게 된다. gen_ex_31_0는 실제 오류정정된 코드 결과를 나타내며, co_data_31_0는 코드결과를 다른 블럭에 전달하기 위한 포트이다.

시뮬레이션 결과에서 클럭 1주기(14.0 - 16.0 μs) 동안에 오류가 포함된 데이터인 0x00000010(정상 데이터는 0x00000018 임)가 들어올 경우 co_data_31_0에서 정확히 오류 비트가 정정됨을 알 수 있다. 다른 주기 동안은 오류가 포함되지 않은 경우이다.

제안된 복호기는 복호과정에서 클럭 사이클을 줄이기 위해서 모든 연산기를 조합논리회로로 설계하였고, 시뮬레이션을 수행한 결과 결과 단지 1클럭 사이클을 사용해서 r(x)로부터 c(x)를 구할 수 있음을 확인하였다.

그림 10은 프로토콜 신호처리기에 대한 전체 타이밍 시뮬레이션 결과를 나타내며, 타이밍 검증을 통하여 프로토콜 신호처리기가 정확히 동작함을 확인하였다.

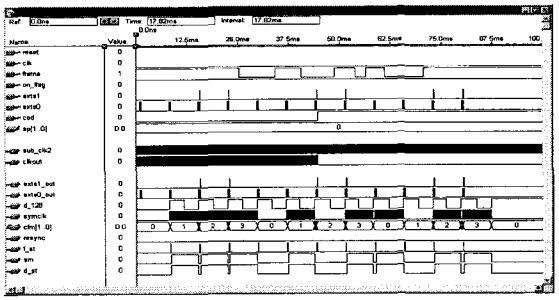


그림 10. 프로토콜 신호처리기의 시뮬레이션 결과
Fig. 10. Simulation result for the protocol signal processor.

설계된 칩의 검증을 위하여 PCB 상에서 Testbed를 구축한 후, 6개의 FPGA(EPF10K20RC 208-4)를 장착하여 데이터를 down-load하여 동작상태를 실험하였다. 테스트 벡터의 생성을 위하여 2개의 칩이 사용되었다. 즉 한 개의 칩에서는 interleaveing된 2비트의 입력 데이터를 생성하며, 다른 1개의 칩에서는 MCU를 제어하고 FLEX 프로토콜에서 제공되는 다양한 패킷을 구성하기 데이터를 생성한다.

V. 결 론

본 논문에서는 고속 페이징 시스템에서 FLEX 프로토콜 신호와의 정확한 동기를 위한 심볼 동기기법 및 실시간 오류정정이 가능한 개선된 복호 알고리듬을 제안하고, 제안한 알고리듬을 기본으로 한 FLEX 신호처리기의 설계 및 FPGA 구현에 관하여 기술하였다.

구현된 신호처리기는 FLEX 프로토콜 사양에 정의된 주파수인 76.8KHz로 동작하며, 최고 6400 bps의 통신속도를 갖는다. 또한, 실시간 오류 정정 기능과 보다 효율적인 프레임 검색 기능들을 제공한다. 그러나 제안된 조합논리회로를 이용한 오류정정 알고리듬은 연산에 소요되는 지연시간이 매우 짧지만 회로연결이 불규칙적이고 GF(2^m)상의 m이 커지면 회로가 복잡하게 된다.

제안된 프로토콜 신호기는 Axil-320 워크스테이션 상에서 VHDL(VHSIC Hardware Description Language)로 모델링하였고, Synopsys^{TM}툴을 이용하여 논리합성(Altera 10K 라이브러리 이용)을 수행하였다. 논리합성 결과 전체 셀의 수는 약 2,631이었다. 설계된 FPGA 칩의 설계 검증을 위하여 Altera MAX+PLUS II 상에서 타이밍 시뮬레이션을 수행하였다. 또한, 6개의 FPGA(EPF10K20RC 208-4)를 장착한 PCB 상에서 Testbed를 구축한 후, Logic Analyzer를 이용하여 제작된 FPGA 칩의 동작상태를 확인하였고, 실험을 통하여 제작된 칩이 정확히 동작함을 확인하였다.

한편, 1개의 칩으로 ASIC화를 위해서 삼성 0.6μ SOG(Sea-Of-Gate) 라이브러리(KG-7500)을 이용하여 논리합성을 수행하였고 논리합성 결과 게이트 수는 약 40,611이었다.

참 고 문 헌

[1] CCIR, *The Book of the CCIR Radiopaging Code*, Nr. 1-CCIR 584, Radio Paging Code Standard Group, 1986.

[2] Motorola Inc., *FLEX^{TM} : Protocol Specification and FLEX^{TM} Encoding and Decoding Requirements*, 1996.

[3] John G. Proakis, "*Digital Communications*," Third Edition, Vol 6, pp. 333-371, 1995.

[4] Darrel R. Judd "Data Synchronization Simulation Using The MATHWORKS Communications Toolbox." *Proc. of IEEE International Conf. on Comm.*, pp. 706-710, June 1996.

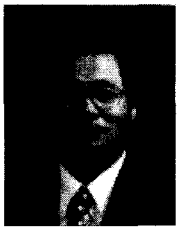
[5] Yi Chang Cheng, Erl Huel Lu, To Chang and Po Chiang Lu, "A New Step-by-step Decoder for Double-error Correcting Primitive Binary BCH Codes in Normal Basis", *International Journal of Electronics*, Vol. 80, No. 4, 1996.

[6] 이만영, "BCH 부호와 Read-Solomon 부호", 민음사, 1990

[7] Alain Poli, and Llorenc Huguet, E*RROR CORRECTING CODES, THEORY AND APPLICATIONS*, Prentice Hall, 1992.

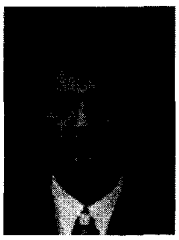
[8] DASC of the IEEE, *IEEE Standard VHDL Language Reference Manual*, June 6, 1994.

저 자 소 개



康 敏 燮(正會員)

1979년 광운대학교 전자통신공학과 졸업(공학사). 1984년 한양대학교 전자공학과 졸업(공학석사). 1992년 일본 오사카대학교 전자공학과 졸업(공학박사). 1984년~92년 한국전자통신연구원 선임연구원. 1986년~1987 일본 오사카대학교 연구원. 1993년~현재 안양대학교 정보통신·컴퓨터공학부 부교수. 관심분야 : ASIC 설계, VLSI 테스팅, 암호보안, 무선 인터넷, 병렬처리 시스템



李 太 應(正會員)

1997년 안양대학교 컴퓨터학과 졸업(공학사) 1999년 안양대학교 첨단산업기술대학원. 전산·정보학과 졸업(공학석사). 1999년~현재 (주)코아테크 개발팀 연구원. 관심분야 : ASIC 설계, I/O device 설계, 암호보안