Core-A 플랫폼을 이용한 동기형 전력제어 임베디드 시스템 설계

이우경 · 문대철*

Design of Synchronized Power Control Embedded System Based on Core-A Platform

Woo-Kyung Lee · Dai-Tchul Moon*

Department of Information and Communication Engineering, Hoseo University, Asan 336-795, Korea

요 약

본 논문은 마스터로 동작하는 32 비트 RISC 프로세서와 전력을 제어할 수 있는 다수의 슬레이브가 동기되어 동작하는 전력 제어 임베디드 시스템을 구현하였다. Core-A 플랫폼은 Core-A 프로세서, AMBA 버스, SSRAM, AC97, DMA, UART, GPIO 모듈 등으로 구성된다. 슬레이브는 4 비트의 디지털 데이터의 값에 비례하여 220V 전력을 제어할 수 있는 아날로그 회로와 마스터가 보내는 신호에 동기되어 다양한 전력제어 패턴을 생성하는 제어 시스템을 설계하였다. Core-A 플랫폼이 라이브러리로 구축된 Flowrian II를 사용하여 소프트웨어를 크로스 컴파일하고 하드웨어 회로를 시뮬레이션으로 검증하였다. 임베디드 시스템은 FPGA 검증 보드와 CPLD 칩에 구현되었고 전력제어 아날로그 보드를 제작하여 구현하였다.

ABSTRACT

This paper realize power control embedded system with one master of Core-A 32-bit RISC processor and several slaves controlling power with synchronized digital signals. Core-A platform is consisted of Core-A processor, AMBA bus, SSRAM, AC97, DMA, UART, GPIO etc. Slave is made by both digital part and analog part. The former generates various power control patterns synchronized with master signal. The latter converts 220V power proportional to 4 bit digital signals. design of Embedded system is executed in Flowrian II, in which software is cross-compiled and hardware is verified by simulation. Embedded system is implemented in FPGA board and CPLD chips as well as PCB board for analog power control.

키워드 : 임베디드 시스템, Core-A 플랫폼, 전력 제어, Flowrian II

Key word : Embedded System, Core-A Platform, Power Control, Flowrian II

접수일자 : 2014. 02. 24 심사완료일자 : 2014. 03. 23 게재확정일자 : 2014. 04. 14

* **Corresponding Author** Dai-Tchul Moon(E-mail:dtmoon@hoseo.edu, Tel:+82-41-540-5683)

Department of Information and Communication Engineering, Hoseo University, Asan 336-795, Korea

Open Access <http://dx.doi.org/10.6109/jkiice.2014.18.6.1413>

print ISSN: 2234-4772 online ISSN: 2288-4165

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright © The Korea Institute of Information and Communication Engineering.

I. 서 론

최근 디지털 제품들은 복잡하고 지적인 기능들을 간단하고 수려한 그래픽 인터페이스를 통하여 사용자에게 편리함과 이동성을 제공하고 있다. 이러한 기술발전의 근간은 고성능 프로세서와 반도체 IP들로 구성되는 하드웨어와 디바이스 드라이버, 펌웨어 및 RTOS로 구성되며 소프트웨어가 연동하는 임베디드 시스템의 발전에 근거한다. 본 논문은 임베디드 시스템과 전력 제어 회로를 결합하고 이를 디지털신호를 이용하여 전력을 제어하는 기술이다. 현재 대한민국의 총 전력 발전량에서 약 20%가 조명에 사용되고 있다. 만약 전력 소비를 감소시키는 기술을 활용하여 기기 당 20% 만을 절감하여도 100만 KW급의 원자력 발전소 2기를 절약할 수 있다.

전력 제어를 가능하게 하려면 임베디드 시스템의 하드웨어 설계와 펌웨어 프로그래밍, 특수 기능용 전자회로, 전력 제어를 위한 아날로그 회로 등의 기술이 개발되어야 한다. 또한 하드웨어와 소프트웨어 설계를 위한 개발 환경과 반도체 IP라이브러리의 관리를 위한 CAD 시스템 환경이 구축되어야 한다[1,2].

본 논문에서 구현한 회로는 크게 3부분으로 구성된다. 첫 번째는 32비트 RISC프로세서를 사용하는 Core-A 플랫폼 하드웨어와 PCM데이터를 분석하여 추출한 전력 제어 데이터를 아날로그 회로로 출력하는 기능을 제어하는 소프트웨어로 이루어진다. 두 번째는 프로세서의 명령을 해석하여 전력을 제어하기 위한 전력 구동 디지털 회로이다. 셋째는 전력을 제어하는 아날로그 회로로 구성된다.

그림 1. 제안된 시스템의 블록도
Fig. 1 Block diagram of proposed system

II. Core-A 플랫폼과 개발 환경

플랫폼은 임베디드 시스템을 개발하는데 공통, 공용의 표준화된 하드웨어 및 소프트웨어 개발 환경으로서, 모든 기능 블록들이 최적화 되고 검증되어 있기 때문에 응용 제품에 따라 기능을 추가하고 삭제함으로써 신속하게 제품을 개발할 수 있다. 이를 이용하여 임베디드 시스템의 설계 방법은 HDL 합성 기술 및 반도체 IP 재사용 설계를 뛰어넘어 플랫폼 기반 설계 (platform based design) 방식으로 발전되어 있다. 본 논문에서 사용된 FPGA에 구성된 하드웨어 프로세서는 순수 국내 기술로 만들어진 32비트 RISC 프로세서인 Core-A를 사용하였다. 그림 2는 Core-A 플랫폼 구조이다[3-6].

그림 2. Core-A 플랫폼의 구조
Fig. 2 Structure of Core-A platform

버스는 AMBA 2.0 버스 프로토콜을 사용하였다. 그 외에 데이터를 저장하기 위한 NOR-Flash and SSRAM controller, 음성 출력에 필요한 AC97 controller 및 RAM에서 빠른 음성 데이터이동을 위해 DMA가 사용되었다. 최종적으로 처리된 데이터는 전력 제어와 조명의 패턴의 정보를 가지고 있으며 GPIO를 통해 출력된다. 출력된 데이터는 자체 제작한 전력구동 CPLD칩에 의해서 디코딩 되어 디지털 회로에서 동작하며 전력제어 아날로그 회로에서 최종적으로 구동된다.

하드웨어와 소프트웨어 최종 FPGA의 검증 흐름은 아래의 그림 3과 같은 순서로 진행된다. 먼저 하드웨어의 IP들은 Verilog언어로 구현되어있다 이 언어로 되어 있는 IP들을 직접적인 게이트들로 구성하기 위해서 개발보드의 FPGA칩으로 Xilinx의 Spartan-3를 사용하였으며 Xilinx의 ISE 툴을 이용하여 합성하였다. 합성된

결과물인 net-list파일을 P&R(place and route)을 위해 사용되는 iNSPIRE에 맞도록 ngc2edif 리눅스 명령어 기반인 스크립트를 생성하여 결과 파일인 edif파일을 만들어 낸다. 생성된 edif파일을 가지고 P&R과정이 완료 되면 Core-A 개발보드에 최종적으로 다운로드 할 수 있는 eif파일이 생성된다. 하드웨어가 완성되고 난 후 이를 제어 할 수 있는 소프트웨어는 Flowrian II에서 corea-eif-as, corea-eif-gcc, corea-eif-ld를 이용하여 Core-A 개발보드에 맞는 크로스컴파일을 하여 최종적으로 개발보드에 다운로드 할 수 있는 HEX파일이 생성된다[3, 4, 7-9].

그림 3. 반도체 IP의 FPGA 검증 흐름

Fig. 3 Flowgraph of verification of semiconductor IPs for FPGA

그림 4. Flowrian II 웹 CAD 시스템의 설계 및 검증

Fig. 4 Design and verification of web CAD system, Flowrian II

여기서 사용된 Core-A 플랫폼을 구성하는 반도체 IP 라이브러리 ORION과 크로스컴파일러는 인터넷 기반

으로 동작하는 Flowrian II를 이용하여 구현되었다. Flowrian II는 그림 4와 같이 임베디드 시스템의 하드웨어와 소프트웨어를 쉽게 설계하고 검증할 수 있는 CAD 시스템으로서 SaaS(Software-As-A-Service) 방식으로 인터넷을 통하여 서버에 구축된 설계 자원을 설계자들이 공유하면서 다양한 플랫폼을 재사용할 수 있도록 구현된 툴이다. C언어로 작성된 펌웨어를 크로스 컴파일하여 추출한 코드는 Core-A 플랫폼의 RAM에 포팅한 후에 ORION 라이브러리와 연동하여 시뮬레이션하면 동작을 검증할 수 있다[10, 11].

III. 전력 구동회로 설계 및 시뮬레이션

Core-A 플랫폼과 전력 구동 디지털 회로에서 마스터와 슬레이브는 핸드셰이킹 프로토콜에 따라 직렬 통신 방식으로 데이터를 전달한다. 마스터는 능동적으로 데이터를 제공하는 역할을 담당하고, 슬레이브는 수동적으로 데이터를 받기만 한다. 마스터는 8비트의 패킷을 전송하는데 데이터의 종류를 결정하는 3비트와 데이터 4비트로 구성된다. 전자는 슬레이브의 주소, 전력제어 패턴, 전력크기, 클럭속도, 동기화여부 등을 구분할 수 있는 정보들이다. 전자의 종류에 따라 후자 데이터의 종류가 구분된다. 그림 5는 핸드셰이킹 프로토콜을 위해 master 모듈과 slave 모듈을 CPLD로 구현한 구성도이다.

그림 5. CPLD의 master 및 slave 모듈의 구성

Fig. 5 Construction of master and slave modules

표 1. 마스터가 송신하는 패킷 형식

Table. 1 Transmitted packet form with master module

7 (MSB)	6	5	4	3	2	1	0 (LSB)
Sort[2]	Sort[1]	Sort[0]	Data[3]	Data[2]	Data[1]	Data[0]	Parity

마스터에서는 표 1과 같이 8비트의 sort와 data, parity 비트를 포함한 패킷형식으로 데이터를 슬레이브에게 전송하게 된다. 이 때 동기 동작과 비동기 동작에 따라 패킷을 전달하는 순서가 중요하다. 먼저 Sort[2:0]는 패킷에 함께 전송되는 4비트의 데이터가 어떠한 정보를 제공하는지를 구한다. Data[3:0]는 Sort[2:0]에 의해 결정되는 데이터를 전달한다. 모든 통신은 Address 패킷을 처음에 전송해야 하고, 다음은 Synchronous 패킷을 전송하고, 다음은 필요에 따라 Brightness와 Clock 패킷을 전송하고, 마지막으로 Command 패킷을 전송해야 한다. 전송된 패킷의 주소와 일치된 고유 주소를 가진 슬레이브만이 연속적으로 전송되는 패킷을 받아들게 된다. 동기 동작으로는 마스터가 동기화 시킬 슬레이브들에게 모든 패킷을 전송해도 LED 램프는 동작하지 않는다. 파워 스위치는 마스터의 신호 SyncExec = '1' 이 되는 순간에 동작한다. 비동기 동작은 마스터가 Command 패킷을 전달하는 순간 LED 전구가 동작하는 방식을 의미한다. Command 패킷이 전달되자마자 파워 스위치는 작동한다[3,4].

전력 구동 회로는 Core-A 플랫폼이 전송하는 명령을 해석하여 아날로그 회로의 전력을 제어하는 역할을 수행한다. 단순히 전력 크기만을 제어하는 것이 아니고 패킷에 정의된 명령에 따라 다양한 패턴을 만들어 내는 팬시 기능을 구현한다.

표 2에서 Sort[2:0]의 값 000~100이외의 다른 address는 상황에 맞추어서 더 많은 동작들을 넣을 수 있도록 정의 하지 않았다. Address는 데이터를 받을 슬레이브의 주소를 지정하고 Command는 LED전구의 구동하는 동작에 대해서 지정하고, Brightness는 Command에 따라서 LED 전구의 빛의 밝기를 조정, Clock은 LED 전구의 빛 변화 속도를 결정, Synchronize는 여러 슬레이브 간의 동기 혹은 비동기 동작을 지정한다. 예를 들어 단순히 전력의 크기를 제어하거나 페이드인(fade-in)이나 페이드아웃(fade-out)방식으로 순차적으로 가변한다든지, 페이드 인/아웃 동작을 하면서 가변되는 변화의 속

도를 조절한다든지, 여러 개의 슬레이브들이 동기 되어 동작시킬 수 있다. 이와 같이 모든 동작들이 최대 15개의 슬레이브까지 확장할 수 있다.

표 2. Sort[2:0] 신호에 따른 Date[3:0]의 동작 상태

Table. 2 Operation state of Data[3:0] by Sort[2:0] signals

Sort [2:0]	Data [3:0]의 상세한 의미 설명
000	Address "1111" == 모든 슬레이브에 해당 (Broadcasting) Others == 통신하려는 슬레이브의 주소
001	Command "0000" == Direct Setting (현재 밝기로 설정) "0010" == Delayed Off (현재 값으로 딜레이 후 Black 세팅) "0011" == Delayed On (현재 값으로 딜레이 후 White 세팅) "0100" == Fading one Black (현재 밝기에서 Black으로 가변) "0101" == Fading one White (현재 밝기에서 White으로 가변) "0110" == Fading Ever Black (현재 밝기에서 Black으로 계속 가변) "0111" == Fading Ever White (현재 밝기에서 White로 계속 가변) "1000" == Up Breathing (현재 밝기와 Black 간을 계속 숨쉬기 가변) "1010" == Saw Waving (현재 값에서 짜인 파형으로 Waving 가변)
010	Brightness Command가 사용하는 밝기나 클럭의 수
011	Clock LED 전구의 밝기 변화를 구동하는 클럭 주파수 지정
100	Synchronize "0000" == 전체 비동기 동작 "0001" == 홀수 주소의 슬레이브들이 동기화 "0010" == 짝수 주소의 슬레이브들이 동기화 "1111" == 전체 동기 동작
Others	Not Defined 아직 정해지지 않은 영역

그림 6. 전력 구동 회로의 구조

Fig. 6 Structure of power driven circuit

그림 6의 전력 구동회로는 핸드셰이킹 방식으로 패킷을 직렬 전송하면서 패리티를 검사하는 회로, 명령어를 해석하는 디코더 회로, 각각의 명령어를 해석하여 동작시키는 유한상태머신, 그리고 동작속도를 제어하는 클럭 발생회로 등으로 구성된다. 예를 들어 그림 7의 톤 카운터(tone counter) 회로는 LED 램프를 구동하는 클럭 신호의 주기를 가변시키는 회로로서 초기 값 지정이 가능한 카운터이다. 레벨1 보다 더 높은 레벨2의 값으로 카운트가 시작되도록 설계하면 terminal count가 이전 경우보다 더욱 자주 발생하게 된다. 클럭 발생회로는 톤 카운터를 응용하여 클럭을 발생시키는데 입력된 4비트 데이터의 크기에 따라 생성되는 클럭의 동작 주파수가 비례하여 가변된다. 이와 같이 톤 카운터는

카운터의 load와 terminal count 기능을 이용하여 다른 주파수의 톤을 생성하게 된다. 그림 8은 톤 카운터의 페이드 인/아웃 시의 시뮬레이션 결과이다.

그림 7. 설계된 톤 카운터의 파형
Fig. 7 Waveform of designed tone counter

그림 8. 톤 카운터의 페이드 인/아웃 시뮬레이션 결과
Fig. 8 Simulation results of tone counter for fade in/out

그림 9는 fine tuning 회로의 시뮬레이션 결과 파형을 나타낸다. 전송받은 데이터로부터 LED 전구의 밝기를 결정하는 모듈에 사용되는 컨트롤러로서 LED전구를 구동시켜야할 시간을 결정하는 모듈이다. 전체 구성은 컨트롤러와 데이터패스로 이루어지는데 컨트롤러는 2단계의 계층 구조로 이루어진다. 상위 계층의 컨트롤러 ‘ftmainctrl’는 LED전구를 구동시켜야 할 경우를 판단하는 역할을 담당하고, 하위 계층의 컨트롤러들은 Command의 개수만큼 존재하게 되는데 본 논문에서는 4개만 구현되었다.

그림 9. Fine tuning 회로의 시뮬레이션 결과
Fig. 9 Simulation results of fine tuning module

마스터 제어 모듈은 마스터 모듈의 동작 순서를 제어하는 컨트롤러로서 유한상태머신 Moore 타입의 FSM (Finite State Machine)으로 설계된다. 마스터는 Request 신호를 발생하고, 슬레이브가 데이터를 받은 후 발생하는 Acknowledge 신호를 감지하여 핸드셰이킹 프로토콜을 구현하였다. 그림 10은 마스터 모듈의 시뮬레이션 결과 파형이다.

그림 10. 마스터 모듈의 시뮬레이션 결과
Fig. 10 Simulation results of master module

그림 11은 슬레이브 전체 회로로서 마스터로부터 핸드셰이킹 직렬 통신을 통해 데이터를 수신하는 모듈로서, 전송받은 데이터로부터 LED 램프 구동 데이터를 생성하는 모듈 `ftmain`를 포함하여 구성된다. 슬레이브 제어 모듈은 마스터 모듈이 Request 신호 `req = '1'`이 되면 데이터를 받기 시작한다. 패리티 비트를 포함하여 8비트를 핸드셰이킹 프로토콜로 차례로 수신한다. 8비트 전송 후에 패리티를 검사해서 에러가 발생한다면 신호 `err = '1'`을 출력하여 재전송을 요청한다. 데이터 전송이 수행되는 동안은 신호 `done = '0'`을 유지한다. 패리티 오류가 발생했으면 상태 `SERR`로 이동하면서 오류 신호 `err = '1'`을 가하면 현재 받은 데이터를 저장하지 않고 다음 패킷을 새로 받기 위하여 처음 상태로 돌아간다. 그림 11과 그림 12는 구현된 슬레이브 모듈의 회

로도 및 시뮬레이션 결과이며, 그림 13은 마스터와 슬레이브 모듈 사이의 통신 시뮬레이션 결과이다.

그림 11. 구현된 슬레이브 모듈의 회로도
Fig. 11 Schematic of slave module

전력 구동 디지털 회로는 Verilog 언어의 RTL 레벨에서 Core-A 플랫폼을 기반으로 설계되었으며 처리된 패턴과 명령어를 GPIO로 출력하여 디지털 데이터를 받아 동작한다. 그림 14는 전력 구동 디지털 회로를 구현한 회로도이다.

그림 14. 구현된 ftmain 모듈의 회로도
Fig. 14 Schematic of implemented ftmain module

그림 15는 슬레이브 모듈과 전력 구동 디지털 회로 간의 인터페이스를 보여주며, 최종적으로 마스터와 슬레이브 그리고 ftmainctrl을 Altera CPLD 칩으로 구현하여 에뮬레이션 검증하였다.

그림 12. 구현된 슬레이브 모듈의 회로도 및 시뮬레이션 결과
Fig. 12 Circuit and simulation results of slave module

그림 13. 마스터와 슬레이브의 통신 시뮬레이션 결과
Fig. 13 Simulation results of communication between master and slave modules

그림 15. 구현된 Slave 및 ftmain 모듈의 구성

Fig. 15 Construction of implemented slave and ftmain modules

IV. 전력제어 데이터 생성 소프트웨어

본 연구에서는 입력된 음악 데이터에 맞추어 빛이 춤을 추듯이 LED 패턴을 넣어 전력을 구동하므로 PCM 데이터가 입력 정보가 되어 표현되고 아날로그 회로를 구동하는 전력 구동 디지털 회로가 받아들이는 명령어 세트가 출력 정보가 된다. 그림 16과 같이 PCM 데이터를 가지고 변환을 수행하는 소프트웨어를 개발하였는데 PCM 데이터를 읽어 파형으로 표현하고 일정한 간격으로 전력제어 패턴을 수행하는 명령어를 16진수 형식으로 저장한다. 이 기능을 수행하는 소프트웨어는 PC Windows 환경에서 Visual Studio 툴을 이용하여 C++로 개발 되었다.

그림 16. PCM 데이터의 파형 출력 소프트웨어

Fig. 16 Software for waveform output display of PCM data

음성 데이터가 입력되면 PCM 데이터의 헤더를 기반으로 하여 음성데이터의 총 길이와 샘플링 등, 노래의 정보를 기반으로 특정 시간의 박자와 패턴을 알 수 있도록 설계되어 있다[12].

그림 17. Wave 파일 포맷의 구성

Fig. 17 Construction of wave file format

Wave 파일 포맷은 멀티미디어 파일의 저장을 위한 마이크로소프트의 RIFF(Resource Interchange File Format) 사양의 하위집합이다. Chunk ID는 파일의 포맷을 정의하기 때문에 Wave 파일의 포맷인 RIFF가 52, 49, 46, 46이 ASCII 코드로 표현되어있다. Chunk Size는 $36 + \text{Sub Chunk2 Size}$ 또는 $4 + (8 + \text{Sub Chunk1 Size}) + (8 + \text{Sub Chunk2 Size})$ 부분의 크기를 의미하며 little endian으로 표현하여 2084가 된다. Format은 Chunk ID와 같이 Wave 의 ASCII로 57, 41, 56, 45로 표현된다. Subchunk1 ID는 문자 fmt + NULL을 포함하고 subchunk1size는 현재 subchunk1size의 크기를 의미한다. AudioFormat는 Wave 파일이 어떤 데이터의 형식인지를 판별하는데 현재 사용하는 형식이 PCM 데이터이기 때문에 1의 값을 가진다. NumChammels는 Wave 파일의 모노(1), 스테레오(2)의 값을 나타내며, Sample Rate는 1초, 동안에 나타낼 수 있는 샘플링을 나타낸다. ByteRate는 $\text{SampleRate} * \text{NumChammels} * \text{BitsPar Sample} / 8$ 으로 초당 저장하는 바이트 수를 표현한 것이고, BlockAlign은 $\text{NumChammels} * \text{BitsParSample} / 8$ 으로 전체 채널을 보관하는 한 샘플의 크기를 나타낸다. 마지막으로 BitsPerSample은 Wave 파일의 비트 형식인 8비트, 16비트로 나타낸다. Subchunk2 ID는 그림17과 같이 ASCII 코드로 데이터를 표현하고 Subchunk2Size는 2048를 나타낸다. 그 후 나오는 데이터들은 실제적으로 출력되는 PCM 데이터 이다. Wave 파일을 가지고 개발한 소프트웨어에 입력파형으로 사용하면 헤더의 정보를 가지고 파형으로 나타내게 된다. 데이터가 표현

되는 수는 최대값과 최소값을 각각 1000개씩 총 2000개를 표현하게 된다. 이 정보들을 가지고 타이밍에 맞추어 일정 비트마다 노랫소리에 동기된 패턴을 직접 정하여서 노래에 맞는 패턴을 제어 할 수 있다. 예를 들어 1분 35초부터 2분 27초에 구간을 확대 하게 되면 PCM데이터의 헤더부분에 Sample Rate값을 가져와 1분 35초를 95초로 바꾸어 Sample Rate * 95번째부터 Sample Rate * 147번째까지의 데이터를 이용하여 2000개의 샘플을 그리도록 구현하였다. 이를 통해 그림 18과 같이 각각의 원하는 타이밍에 몇 번째 슬레이브로 어떤 패턴을 입력 할지 보내주게 된다. 또한 받은 데이터를 종합하여 최종적으로 Core-A 플랫폼에 다운로드 할 수 있는 노래의 헤더와 패턴 그리고 시간 등의 각각의 필요한 정보들이 하나의 HEX파일 형식의 데이터를 Windows 환경에서 C++로 개발한 소프트웨어를 통해 출력된다.

그림 18. 소프트웨어의 패턴입력 화면
Fig. 18 Patterns input display of software

V. 결 론

본 논문은 Core-A 플랫폼이 마스터로 동작하면서 전력을 제어하는 명령을 발생하여 직렬 통신으로 다수의 슬레이브에 전송하여 전력을 제어하는 임베디드 시스템을 구현하였다. 임베디드 시스템은 FPGA 검증 보드와 CPLD칩에 구현되었고 전력 제어하는 아날로그 보드는 아날로그 파트와 디지털 파트를 각각 독립 할 수 있게 제작 하여 아날로그 회로의 어떤 조명이든 디지털 부에 역전압이 흐르게 되어도 디지털 부에 문제가 없도록 구현되었다. 또한 Core-A플랫폼의 각종 IP들을 ORION라이브러리로 구축해서 각종 IP를 쉽게 재사용

하여 구현 할 수 있도록 SaaS(Software as a Service) CAD 시스템인 Flowrion II 툴을 이용하여 구현하였다.

본 논문에서 제안된 전력을 제어하는 기술은 에너지를 절약하기 위한 시스템에 각종 전력을 자동으로 전력을 차단 할 수 있는 기능을 갖고 있기 때문에 대형 광고판의 자동 전력 차단 회로, 사무실에서 사용하는 PC 기반 IT 제품의 전력차단 장치, 한 개의 서버에서 비전 센서를 이용하여 건물 전체 전력 제어 및 가로등이나 지하 주차장 등에 비전 임베디드 시스템을 이용한 전력 제어 등에도 응용이 가능하다.

REFERENCES

- [1] In-Kyu Jang, Dai-Tchul Moon, Hyoung-Kie Yun, Jae-Min Jang, Jeong-Seop Seo, "Design of Vision Embedded Platform with AVR", *2013 9th International Conference on Computing Technology and Information Management, ICCM 2013*, 2013. 6.
- [2] Dai-Tchul Moon, In-Hag Park, Hyoung-Kie Yun, Xu Gang, In-Kyu Jang, Woo-Kyung Lee, Jae-Min Jang, Ki-Young Jung, "An FPGA Implementation of Energy Saving Embedded System with Multiple Vision Sensor", *17th Asia-Pacific Conference on Communications, APCC 2011*, 2011. 10.
- [3] Ahn-Do Ki, *Platform Design using Core-A Processor*, Hong-Rung Pub., 2010. 3.
- [4] Jong-Yeol Lee, Kang Lee, Si-Ho Kim, Byeong-Yoon Choi, Kwang-Yeob Lee, *Design System-on-Chip Based on CORE-A Processor*, Hong-Rung Pub., 2010. 11.
- [5] Kwang-Yeob Lee, "An architecture for a High-performance Core-A Embedded Processor", *2009 Conference on SoC, IEIE*, pp. 184-187, 2009. 5.
- [6] Chang-Su Ha, Byeong-Yoon Choi, "Hardware Design of AES Cryptography Module Operating as Coprocessor of Core-A Microprocessor", *KIICE Journal*, No. 13(12), pp. 2569-2578, 2009. 12.
- [7] Xuelong Xu., Jingzhe Xu, Seung-Pyo Jung, Ju-Sung Park, "The Design of Multi-media SoC Platform Based on Core-A Processor", *IEIE Journal*, No. 50(6), pp. 99-104, 2013. 6.
- [8] Xuelong Xu., Jingzhe Xu, Seung-Pyo Jung, Ju-Sung Park, "Design and Verification of Efficient On-Chip Debugger for Core-A", *IEIE Journal*, No. 47(4), pp. 50-61, 2010. 4.

- [9] Hyung-Bae Park, Jeong-Hoon Ji, JINGZHE XU, Gyun Woo, Ju-Sung Park, “Core-A Debugger and ICE”, *Conference on IEK*, pp. 192-195, 2009.
- [10] Ming Ma, Hyoung-Kie Yun, Woo-Kyung Lee, Jung-Seop Seo, In-hag Park, Dai-Tchul Moon, “Web-Based IP Reuse CAD System, Flowrian”, *2010 International Conference on Signal Processing, Image Processing and Pattern Recognition*, SIP 2010, 2010. 12.
- [11] Ming Ma, Hyoung-Kie Yun, Woo-Kyung Lee, Jung-Seop Seo, In-hag Park, Dai-Tchul Moon, “Engineering-SaaS CAD System for Embedded System Design, FlowrianII”, *IPSOC 2010*, 2010. 12.
- [12] In-Hag Park, Kwang-Ki Ryu, Jong-Hwa Na, Hae-Wook Choi, *Design and Implementation experiment for Application HW/ SW with Openrisc Platform*, Brain Korea Pub., 2008. 10.

이우경(Woo-kyung Lee)

2011 호서대학교 정보통신공학과 공학사
 2013 호서대학교 정보통신공학과 공학석사
 ※ 관심분야 : SoC 설계, 임베디드 시스템 설계, 하드웨어 반도체 IP 설계

문대철(Dai-tchul Moon)

1987 고려대학교 전자공학과 공학박사
 1994 North Carolina State Univ. 연구교수
 2005 Minnesota State Univ, Duluth 객원교수
 1984~현재 호서대 정보통신공학과 교수
 ※ 관심분야 : DSP 및 영상신호처리 칩 설계, SoC 설계, 임베디드 시스템 설계, VLSI 신호처리