SOPC 기반의 재구성 가능한 로봇제어기 구현

Implementation of SOPC-based Reconfigurable Robot Controller

최 기 홍*, 최 영 준, 박 재 현
(Kihong Choi, Youngjoon Choi, and Jaehyun Park)

Abstract : Recently, a variety of intelligent robots are developed for the personal purpose beyond the industrial application. These intelligent robots have ranges of sensors, actuators, and control algorithms to their application. In this paper we propose a reconfigurable robot controller, SR^2C (The SOPC-based Reconfigurable Robot Controller), based on SOPC (System on a Programmable Chip), that can be reconfigurable easily by software. The proposed robot controller contains not only a processing module but also robot-specific IP's. To show a feasibility of the proposed robot controller, a small entertainment robot, Wizard-4 is implemented with a single chip controller as proposed in this paper.

Keywords : robot controller, SOC (system on chip), hardware-software co-design, FPGA

I. 서론

로봇 산업은 최근 들어 괄목할만한 성장을 하고 있으며, 기존의 산업용 로봇에 국한되어있던 응용분야가 개인용 로봇, 의료용 로봇, 가정용 로봇 등 다양한 형태의 지능형 로봇으로 발전되고 있다. 이러한 여러 종류의 지능형 로봇 개발을 위해서 새롭고 다양한 종류의 센서와 액추에이터들이 이용되고 있으며, 이들에 대한 인터페이스 방식 또한 다양한 구조로 발전되고 있다. 그러나 이러한 유연하고 다양한 요구의 분출에도 불구하고 범용 마이크로프로세서 또는 마이크로콘트롤러를 사용하여 PCB 상에 구현된 기존 로봇제어기는 여러 종류의 센서와 액추에이터들이 요구하는 다양한 인터페이스를 완벽히 만족 시킬 수 없다. 특히 용도에 따라 유연하게 변화할 수 있는 지능형 로봇의 개발을 위해서는 좀 더 유연한 구조의 로봇제어기가 필요하다 하겠다.

반면, 반도체공정과 집적회로 기술이 발전하면서 PCB (Printed Circuit Board) 상에 다수의 부품들로 구성되던 시스템을 한 개의 칩 내에 집적한다는 SOC(System On a Chip) 개념이 등장하였는데, 이는 PCB 기반의 기존 시스템에 비하여 시스템을 구성하는 부품들을 칩 내에 배치하고 배선 함으로서 고주파 특성이 향상되어 시스템의 동작속도를 높일 수 있으며, 다수의 부품을 단일 칩 내에 집적하여 시스템의 부피가 작아지고, 전체 시스템의 신뢰도를 높일 수 있는 등, SOC 기반의 시스템은 여러 면에서 장점이 있다. SOC 설계에는, 복잡 다양한 시스템이 통합되는 SOC의 특성상 IP(Intellectual Property)라 불리는 기능 블록(MPU, DSP, CODEC, Peripherals)을 재사용하여 SOC를 설계하는 것이 일반적이다.

과거에 이러한 SOC는 ASIC(Application Specific Integrated Circuit) 기반에서만 구현 되었으나, 최근에는 PLD (Programmable Logic Device)의 성능과 가격 경쟁력이 향상되면서 SOC의 범주가 PLD기반의 SOPC (System On a Programmable Chip)로 까지 확장되었고, 결과적으로 SOPC는 SOC와 재구성이 가능한 PLD의 장점을 모두 지니는 새로운 개념의 지능형 시스템 설계 방법으로 사용될 수 있게 되었다[1,2,3,4,5].

본 논문에서는 유연성과 다양성이 강조되는 최근의 지능형 로봇의 제어기로서 기존 로봇제어기의 단점을 극복하는, SOPC를 적용한 새로운 형태의 로봇제어기인 SR^2C(The SOPC-based Reconfigurable Robot Controller)를 제안하고 이를 구현하기 위한 하드웨어 기반 실시간 태스크 스케줄러 및 로봇제어 전용IP들의 구조를 제안한다. 또한 제안된 SR^2C의 타당성 검증을 위하여 소형 로봇 플랫폼인 Wizard-4에 적용하였다.

II. SOPC 기반의 로봇제어기 구조

일반적인 로봇 제어기는 액추에이터를 구동하는 구동부와 각종 센서를 연결하는 센서부, 이들을 제어하고 제어 알고리즘을 구현하는 제어부로 크게 나눌 수 있다. 기존의 전통적인 로봇제어기는 이들을 각각의 독립된 상용 부품으로 구현하고, 이들을 PCB상에 연결시키는 방법을 사용하고 있으나 본 연구에서는 지능형 로봇에서 다양한 목적으로 변형하여 사용할 수 있는 유연한 구조의 로봇제어기를 구현하기 위하여 이를 SOPC 기반으로 구현한다. SOPC 내부는 마이크로프로세서와 같은 마스터 IP(master IP)와 외부 인터페이스 등을 담당하는 슬레이브 IP(slave IP), 그리고 이들 IP를 연결하는 온-칩 버스(on-chip bus)로 구성된다. 본 절에서는 SR^2C를 구성하는 각각의 요소에 대하여 설명한다.

1. Avalon 온-칩 버스

온-칩 버스는 SOC를 구성하는 다양한 IP들을 서로 연결하기위한 칩 내부의 버스로서 다양한 표준이 존재한다. 그러나 대부분의 온-칩 버스 표준은 ASIC 기반 SOC 내에 사용되는 것을 주목적으로 설계하였기 때문에 PLD의 자원소모를 최소화려는 관점에서 이를 SOPC내에 그대로 적용하는 것은 비효율적이며, 성능도 좋지 못하다. 따라서 SOPC에는 전용의 온-칩 버스가 사용되는데 본 연구에서는 ALTERA에서 개발한 Avalon 온-칩 버스를 사용한다. 이는 기존의 ASIC용 온-칩 버스와 달리 PLD 자원의 소모를 최소화하고 어드레스디코딩

* 책임저자(Corresponding Author)
논문접수 : 2003. 8. 27., 채택확정 : 2004. 1. 20.
최기홍 : LG산전 자동화연구소(khchoi@lgis.com)
최영준, 박재현 : 인하대학교 정보통신공학부
(yjchoi@resl.inha.ac.kr /jhyun@inha.ac.kr)
※ 본 연구는 산업자원부지원 차세대신기술개발사업인 "퍼스널로봇을 위한 시스템엔지니어링기술개발"의 지원으로 수행되었음.

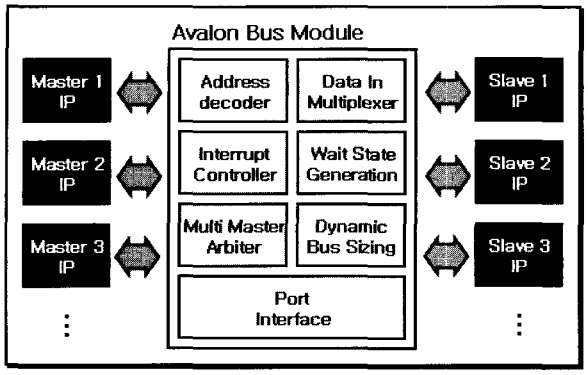


그림 1. SOPC 기반의 로봇제어기(SR^2C) 구조.
Fig. 1. Structure of SOPC-based Robot Controller.

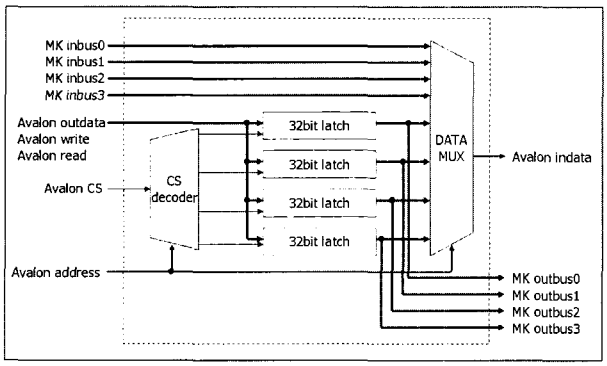


그림 2. Avalon 온-칩 버스 브리지 IP의 구조.
Fig. 2. Structure of Avalon bus bridge.

로직, 인터럽트 로직, 상태제어 로직 등을 포함한다[6]. 본 논문에서 구현한 로봇제어 전용 IP는 슬레이브 디바이스로만 동작을 하며 호스트 인터페이스가 데이터 버스만으로 이루어져 있다. 따라서 사용하려는 온-칩 버스에 따른 별도의 슬레이브 브리지가 필요하다. 이러한 마스터/슬레이브 계층구조와 별도의 브리지를 사용하는 것은 구현한 IP를 다양한 온-칩 버스에 손쉽게 재사용할 수 있도록 하기 위함이다. SR^2C 로봇제어기를 위해 구현한 Avalon 온-칩 버스 브리지 IP는 그림 2와 같다. 이는 어드레스 디코더, 32비트 데이터 래치, 32비트 데이터 멀티플렉서로 구성되어 4개의 32비트 데이터 입력버스와 4개의 32비트 데이터 출력버스를 인터페이스 할 수 있다.

2. Nios 소프트 프로세서 코어

로봇을 제어하는데 따르는 복잡 다양한 알고리즘과 상태머신을 처리함에 있어 마이크로프로세서를 사용하는 것은 매우 효율적이다[7]. 따라서 SR^2C에 마이크로프로세서 코어를 구현하는 것이 필요하다. SOPC에 프로세서 코어를 구현하는 방법은 크게 두 가지로 나눌 수 있으며 이는 다음과 같다.

- 하드코어프로세서 탑재형: 마이크로프로세서 블록 내장형 PLD를 사용하는 방법이다. 이는 마이크로프로세서를 별도로 설계하지 않아도 되며 마이크로프로세서 코어만을 놓고 볼 때 최적화되어 있는 장점이 있다. 하지만 유연성이 기타 방법보다 떨어지고 디바이스 가격이 고가인 단점이 있다.

표 1. 타겟팅 FPGA에 따른 Nios의 성능[11].
Table 1. Performance of Nios to the FPGA[11].

디바이스	32 비트		16 비트	
	LEs	Fmax	LEs	Fmax
Stratix	2555	125	1897	135
Mercury	2670	72	2040	73
APEX II	2599	74	2037	81
APEX 20KE	2550	62	2040	64
FLEX 10KE	2810	36	2220	44
ACEX 1K	2810	37	2220	43

표 2. Nios 프로세서의 성능비교[10,17].
Table 2. Performance comparison [10,17].

제작사	프로세서	동작 주파수	MIPS
ALTERA	Nios	50	44
ARM	7Thumb	59	53
Hitachi	H8S	25	12
Motorola	M-Core	33	31
Motorola	68360	33	8.3
Motorola	68HC16	25	2.5
Philips	XA(8051)	33	5

- 소프트코어프로세서 탑재형: PLD 벤더 또는 IP 설계업체에서 제공하거나 직접 설계한 마이크로프로세서 IP를 PLD에 구성하여 사용하는 방법이다. PLD 벤더에서 제공하는 마이크로프로세서 IP의 경우 다른 방법보다 신뢰성과 효율성이 우수한 장점이 있다[8,9].

PLD벤더가 제공하는 대표적인 소프트코어프로세서는 ALTERA의 Nios와 XILINX의 MicroBlaze가 대표적이며 본 논문에서는 ALTERA의 Nios를 사용하여 마이크로프로세서 코어를 구현하였다[10]. Nios는 LPM(Library of Parameterized Module) 방식으로 취급되기 때문에 용도에 따라 데이터패스의 크기, 레지스터의 개수, 하드웨어 곱셈기의 종류 등을 선택하여 구성할 수 있으며 설계자에 의해 임의로 설계된 커스텀 명령어 및 해당 로직을 Nios의 ALU(Arithmetic Logic Unit)에 추가 할 수 있다. 이러한 특징은 더욱 유연한 SR^2C의 설계가 가능하도록 한다. FPGA에 구성되는 Nios의 특성상 최대 연산처리속도와 적용가능 여부는 FPGA의 구조와 속도 및 크기에 따라 결정된다. 표 1은 ALTERA의 FPGA 패밀리에 Nios를 구성한 결과를 나타내며 표 2는 Nios와 일반적으로 사용되는 마이크로프로세서 코어 간의 성능 비교이다.

3. 하드웨어 기반 실시간 태스크 스케줄러 설계

제어 알고리즘의 연산 또는 센서의 샘플링과 같이 일정 주기마다 실시간으로 처리되어야만 하는 태스크가 다수일 경우에는 일반적으로 실시간 운영체제(RTOS;Real-Time Operating System)를 사용하여 소프트웨어를 설계하는 것이 일반적이다. RTOS에서 가장 중요한 역할을 담당하는 부분은 태스크 스케줄러로서, 이는 커널의 일부분으로 소프트웨어적으로 구현되어 있으며 스케줄러는 매 Clock-Tick(타이머 인터럽트)마다 실행되어 다음에 실행해야 할 태스크를 선택하는 역할을 한다.

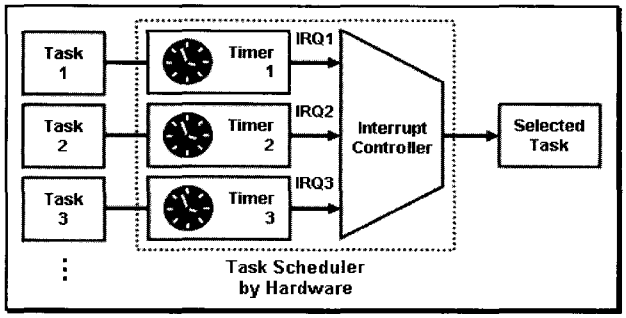


그림 3. 하드웨어기반 실시간 태스크 스케줄러의 구조.
Fig. 3. Structure of hardware-based task scheduler.

이러한 기존 태스크 스케줄러는 소프트웨어적으로 구현되어 처리되기 때문에 소프트웨어 처리능력의 일부를 소진하게 되며 Clock-Tick을 기반으로 하여 태스크들의 실행주기가 Clock-Tick 주기에 종속되는 단점이 있다[18,19]. 이와 같은 단점에도 불구하고 RTOS의 태스크 스케줄러가 Clock-Tick과 소프트웨어 기반으로 구현되는 주된 이유는 RTOS가 적용되는 대부분의 기존 시스템이 소프트웨어의 재구성은 자유롭지만 타이머와 인터럽트 같은 하드웨어의 재구성은 불가능하다는 제약 때문이었다. 이에 반해 본 논문에서 제안하는 SR^2C는 SOPC를 기반으로 하기 때문에 하드웨어의 재구성이 가능하여 소프트웨어를 포함한 하드웨어의 자원관리를 자유자재로 할 수 있다. 이는 기존의 소프트웨어로 구현된 태스크 스케줄러를 사용하는데 따른 문제점을 극복할 수 있는 새로운 형태의 하드웨어 기반 태스크 스케줄러 설계를 가능하게 하였다.

구현된 하드웨어 기반 실시간 태스크 스케줄러의 구조는 그림 3과 같다. 이는 주기적으로 실행되어야 하는 태스크마다 각각의 타이머와 인터럽트를 할당하고 비 주기적인 태스크에는 외부 인터럽트를 할당하여 인터럽트 처리기에 의해 태스크의 스케줄링이 하드웨어적으로 이루어지도록 하는 구조이다. 즉 태스크의 우선순위 및 실행주기는 인터럽트 우선순위와 타이머 설정 값에 따라 결정되고 스케줄링 사상은 인터럽트 처리기에 설정된다. 이러한 태스크 스케줄링에 필요한 일련의 동작은 모두 하드웨어적으로 처리되므로 다음과 같은 장점을 갖는다.

- 태스크 스케줄링을 위한 소프트웨어의 부하 감소
- 태스크 각각의 실행주기를 자유롭게 설정 가능
- 소프트웨어적 오류에 따른 교착상태 회피
- 하드웨어적인 처리를 통한 응답 속도 향상

태스크 타이머 IP는 16비트 분해능의 설정 데이터에 따라 입력클럭을 카운팅하고 카운팅이 완료되면 인터럽트를 요청한다. 태스크의 실행주기가 태스크 타이머 IP의 수용범위를 넘을 경우, 태스크 타이머 IP에 입력되는 클럭을 프리스케일링하여 사용한다. 또한 인터럽트 처리기는 사용하고자 하는 스케줄링 사상에 의해 입력된 인터럽트 요청신호 및 우선순위에 따라 해당 인터럽트 서비스 루틴을 호출한다. 이와 같은 하드웨어 기반의 태스크 스케줄러는 실시간 운영체제 내부 커널의 수행시간을 획기적으로 단축시키고 각각의 태스크에 가장 적합한 수행주기를 설정할 수 있게 하여 로봇 제어

표 3. 실험을 위한 태스크의 설정.
Table 3. Test parameters for task scheduler.

	주기 (Hz)	우선순위
Task 1	250	30
Task 2	200	33
Task 3	150	36
Task 4	100	39

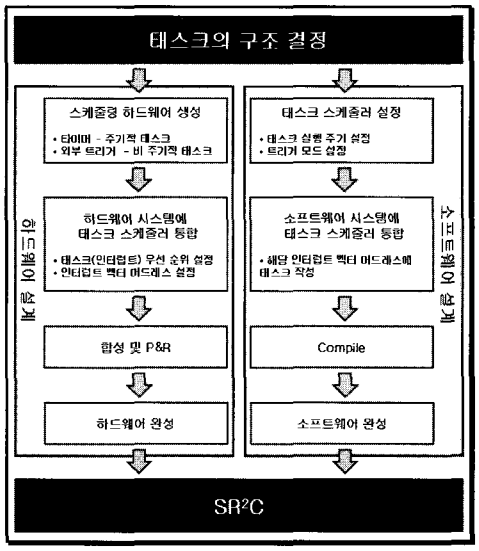


그림 4. 하드웨어기반 태스크 스케줄러의 설계과정.
Fig. 4. Design procedure of task scheduler.

프로그램을 유연하고 효율적으로 작성할 수 있도록 한다. 그림 4는 하드웨어 기반 실시간 태스크 스케줄러를 사용하기 위한 설계과정을 나타낸다. 태스크 스케줄링 하드웨어는 태스크의 종류와 개수에 따라 구성하고 소프트웨어는 사용하는 태스크의 특성을 설정한다. 태스크 루틴은 해당 태스크의 인터럽트 서비스 루틴에 작성하게 된다.

소프트웨어로 구현된 실시간 운영체제의 태스크 스케줄러를 대체하는 하드웨어 스케줄러의 타당성을 검증하기 위하여 스케줄링 사상은 선점형으로 설정하고 1msec의 수행시간을 갖는 4개의 태스크를 표 3과 같이 할당하여 실험하였다. 실험 결과 그림 5와 같이 각 태스크들은 일정한 주기로 실행(파형의 상승 엣지)되고 우선순위에 따라 태스크의 실행순서가 스케줄링되어 종료 됨(파형의 하강 엣지)을 확인할 수 있다.

4. 로봇 제어 전용 IP의 설계

로봇 제어 전용 IP는 특정 센서와 액추에이터를 Avalon 버스 브리지를 통해 Nios 프로세서에 인터페이싱하는 역할을 한다. 이는 다양한 센서와 액추에이터를 인터페이스하고 처리하는데 있어, 범용 마이크로콘트롤러에 직접 연결하여 구현하는 기존 로봇제어기의 단점인 최적화된 분해능으로 제어 불가능, 마이크로콘트롤러 내의 기능블록이 지원하지 않는 부분을 처리하기 위한 소프트웨어의 부하 증가 등의 한계요소를 극복할 수 있도록 한다. 또한 SR^2C에 새로운 형태의 장치를 인터페이싱하기 위해서는 그에 따른 전용 IP를 설계

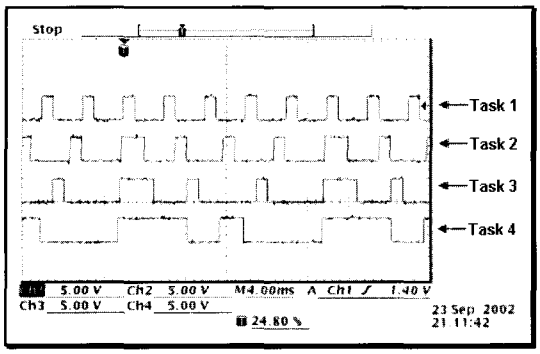


그림 5. 하드웨어기반 태스크스케줄러의 실험 결과.
Fig. 5. Test result of task scheduler.

하고 이를 SR^{2}C에 추가 함으로서 물리적인 변화 없이 새로운 제어기를 구성할 수 있다. 이는 새로운 형태의 지능형 로봇을 개발할 때 기존의 로봇 제어기를 그대로 사용할 수 있는 장점을 갖는다. SR^{2}C 내부에 구현된 로봇 제어 IP들은 VHDL(VHSIC Hardware Description Language)을 사용하여 설계되었으며 Exemplar Logic의 Leonardo spectrum에 의해 합성되었다[20,21].

4.1 펄스 폭 변조기

아날로그 신호를 디지털 데이터로 제어하기 위하여 정해진 주기에 펄스의 폭을 변조하는 PWM(Pulse Width Modulation) 모듈은 로봇 구동 모터를 구동하는데 가장 핵심 요소로서, SR^{2}C 내부에 IP로 구현하였다. 구현한 PWM IP는 8비트 분해능을 갖기 때문에 8비트 증가 카운터의 데이터와 PWM 데이터를 서로 비교하여 이진화 함으로서 해당 듀티비의 PWM 신호를 생성하게 된다. 여기서 8비트 증가 카운터는 PWM IP의 외부에 존재한다. 이는 다수의 PWM IP 사용시 8비트 증가 카운터를 공유하여 사용하도록 함으로서 자원 소모를 최소화 하도록 한다.

4.2 소나(SRF04) 인터페이스

물체 또는 거리의 감지를 위해 사용되는 Sonar 센서 중에서 작은 크기로 인해 많이 사용되고 있는 초음파 센서를 인터페이스 하기 위한 IP를 구현하였다[그림 6]. 형식은 SRF04 초음파 센서의 인터페이스 신호 형식으로 최소 10usec 동안 "H"를 유지하는 트리거 펄스를 보내면 거리에 비례하는 100usec에서 18msec 사이의 "H"를 유지하는 에코 펄스가 출력된다. 즉 사용자는 SRF04로부터의 데이터를 취득하기 위하여 에코 펄스가 "H"인 동안의 시간을 측정하면 된다. 이를 타이머가 내장된 범용 마이크로콘트롤러를 이용하여 인터페이스하는 경우에는 타이머의 기준 클럭을 소나의 에코 펄스에 최적화할 수 없으므로 분해능이 제한되며 주기적인 인터럽트로 인한 소프트웨어의 실행이 방해 받을 수 있다. 따라서 SRF04 소나 인터페이스 IP를 SOPC 내부에 구현하여 소프트웨어 실행의 효율을 높였다.

4.3 R/C서보 모터 제어기

로봇에서 널리 사용되는 위치 감지/조정용 기기로 R/C서보 모터(이하 서보 모터)가 널리 사용되는데 이는 서보 펄스 형태로 입력되는 지령 치에 따라 모터의 회전각이 제어된다. 서보 모듈은 DC 모터, 기어박스, 포텐셜미터, 제어기로 이루

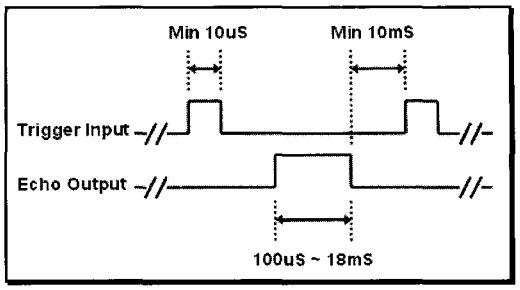


그림 6. SRF04 초음파 센서 인터페이스 신호 형식.
Fig. 6. Signal format for sonar sensor.

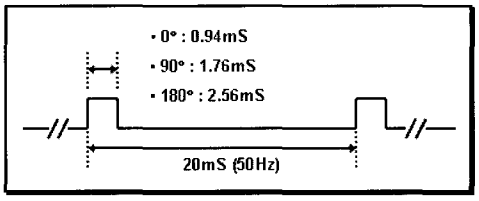


그림 7. 서보 모터의 제어신호 형식.
Fig. 7. Signal format of servo motor.

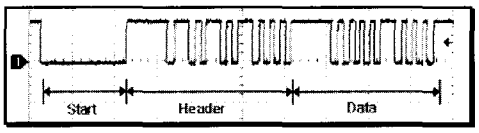


그림 8. 리모컨 데이터의 수신 파형.
Fig. 8. Signal format of remote controller data.

어져 있으며 자체적인 서보 시스템을 사용하여 피드백 제어를 하므로 센서, 제어기 등의 부가적인 부품이 필요하지 않다. 그림 7은 서보 모터를 제어하기 위한 신호의 형식을 나타낸다. 서보 모터의 제어 신호 주기는 20msec이고 회전각은 제어 신호의 듀티비에 따라 결정된다. 이 듀티비는 최소 4.7% 에서 최대 12.8%로, 0%에서 100%의 듀티비를 갖는 일반 PWM 신호와는 다른 것이며 이러한 특징으로 인해 범용 마이크로콘트롤러를 이용하여 서보 모터를 제어하는 경우에는 타이머와 인터럽트를 사용한다. 이러한 인터럽트의 사용은 소프트웨어의 실행을 방해하기 때문에 효율적인 서보 모터의 제어를 위하여 입력된 8비트 데이터에 따라 서보 모터의 제어신호를 생성하는 IP를 설계하였다.

4.4 리모컨 디코더

산업용 로봇과 달리 개인용 로봇은 사람과 유기적으로 인터페이스가 되어야 하며, 인간과의 인터페이스 중 가장 쉬운 수단 중 하나가 적외선 리모컨을 사용하는 것이다. 적외선(InfraRed, IR) 리모컨은 다른 광원(light source)이 리시버가 데이터를 수신하는 것을 간섭하지 못하도록 하기 위하여 IR LED를 30KHz에서 40KHz 사이의 주파수로 구동하여 리시버(receiver)에게 데이터를 전송한다. 이러한 적외선 리모컨에서 전송되는 데이터는 이진 코드의 형태이며 제조사에 따라 각 비트의 위상과 길이뿐만 아니라 코딩방법 또한 다르다.

본 논문에서 사용한 리모컨의 데이터 수신 파형은 그림 8과 같다. 헤더는 리모컨 제조사의 고유번호이며 실제 리모컨 버튼에 대한 정보는 데이터에 있다. 이러한 리모컨 데이터를

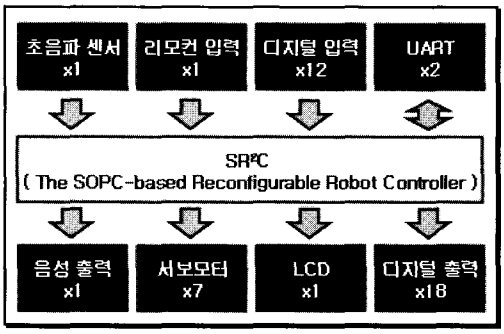


그림 9. Wizard-4용 SR²C 내부 구조.
Fig. 9. Structure of SR²C for Wizard-4.

수신하여 디코딩하기 위해서 범용 마이크로콘트롤러를 이용하는 경우에는 타이머와 인터럽트를 사용하는 것이 일반적이다. 그러나 이러한 방법은 소프트웨어의 실행을 방해할 수 있기 때문에 리모컨 데이터 디코더 IP를 구현하여 하드웨어적으로 리모컨 데이터를 디코딩하도록 하였다. 구현한 리모컨 디코더 IP는 리모컨 수신 패킷의 시작을 감지(하강 엣지 검출)하면 샘플링 클럭을 수신데이터에 동기 시키고 샘플링 클럭을 카운트하여 헤더 및 데이터의 시작과 끝을 감지한다. 헤더가 시작되면 샘플링 클럭의 상승에지 마다 리모컨 수신 패킷을 쉬프트레지스터에 저장하여 리모컨데이터를 디코딩한다. 리모컨 디코더 IP는 데이터 수신 완료 시 인터럽트 어크놀러지가 입력될 때까지 Avalon 버스에 인터럽트를 요청할 수 있다.

III. 로봇 제어기의 구현

1. Wizard-4 로봇 플랫폼

본 논문에서 제안한 SR²C의 타당성을 검증하기 위하여 SR²C 기반의 소형 로봇 플랫폼 Wizard-4를 구현하였다. 이는 소나 센서와 적외선 센서 그리고 한 쌍의 팔과 두 다리로 이루어진 엔터테인먼트 로봇으로 이를 구성하는 모든 센서 및 액추에이터는 SR²C로 제어된다. 현재 사용되는 대부분의 FPGA가 SRAM기반이고, 아날로그 회로의 내장이 불가능하여 SR²C는 FPGA, Flash-ROM, SRAM 그리고 기능확장용 SDRAM 모듈과 PMC(PCI Mezzanine Connector) 인터페이스로 구성된 ALTERA의 Nios 시스템 개발 보드 상에 AD 변환기를 추가하여 플랫폼을 구현하였다. 본 연구에서는 ALTERA의 APEX EP20K200EFC484-2 FPGA 상에 IP를 집적하여 SR²C를 Wizard-4 로봇의 제어기로 구성하였고, 내부 구조는 그림 9와 같다. SR²C는 이와 같은 센서 및 액추에이터를 통합하고 제어 함으로서 Wizard-4가 의도한 기능을 수행할 수 있도록 한다. 또한 SR²C 내부 설계를 위하여 ALTERA의 SOPC 개발 소프트웨어인 SOPC Builder와 Quartus-II를 사용하고 소프트웨어 설계는 Nios-GCC 컴파일러를 사용하였다 [22,23].

2. Wizard-4 구현 결과

SOPC로 구현된 로봇 제어기의 성능은 제어기의 최대 동작 속도와, 사용하고 있는 내부자원(Logic Element), 외부 핀 수 등을 종합하여 평가한다. 본 논문에서 구현된 SR²C의 동작속도 및 사용 자원은 표 4와 같다.

표 4. Wizard-4를 SR²C에 구성한 결과.
Table 4. Implementation result of SR²C.

Max Freq	45.67 MHz
LEs	4196 (50%)
Pins	127 (33%)
ESBs	26496 (24%)

표 5. Wizard-4를 제어하기 위한 태스크.
Table 5. Control tasks for Wizard-4.

	주기 (Hz)	우선순위	역할
Task 1	150	30	일반 서보모터 제어
Task 2	400/130	33	허리 서보모터 제어
Task 3	40	36	소나 센서 샘플링
Task 4	15	39	디버그

SR²C상에 구현된 Wizard-4의 제어 프로그램은 표 5와 같이 총 4개의 태스크로 이루어져 있다. 이를 하드웨어 기반 실시간 태스크 스케줄러를 통해 스케줄링 함으로서 개발 시간 단축, 성능 향상 등의 효과를 볼 수 있었다. 또한 사용하는 센서 및 액추에이터에 따라 각각의 IP를 할당하여 소프트웨어에 걸리는 부하를 최소화하였다.

VI. 결론

본 논문에서는 SOPC 개념을 로봇제어기에 적용한 새로운 형태의 로봇제어기인 SR²C(The SOPC - based Reconfigurable Robot Controller)의 구조를 제안하고 로봇 제어에 필요한 전체 시스템을 FPGA에 통합하여 설계 및 구현하였다. SR²C는 로봇 제어에 널리 사용되는 다수의 전용 IP와 Nios 소프트코어 프로세서를 Avalon 온-칩 버스를 통해 구성하였으며 하드웨어 기반 실시간 태스크 스케줄러를 통해 소프트웨어 처리의 효율을 높였다. 구현된 SR²C는 기존 로봇 제어기에 비해 하드웨어의 물리적인 변경 없이 소프트웨어만으로 재구성이 가능하여 다양한 형태의 지능형 로봇에 사용될 수 있을 뿐 아니라, 내부에 로봇 제어 전용 IP를 다수 포함하고 있어 높은 성능을 보인다. 또한 소프트웨어와 하드웨어의 통합 설계를 가능하게 함으로써 개발기간을 단축하고 재구성이 자유로운 장점을 갖는다.

참고문헌

[1] R. Hartenstein, "A decade of reconfigurable computing: a visionary retrospective," *Proceedings of IEEE Conference and Exhibition on Design, Automation and Test in Europe*, pp. 642-649, 2001.

[2] P. C. French and R.W. Taylor, "A self-reconfiguring processor," *Proceedings of IEEE Workshop on FPGA for Custom Computing Machines*, pp. 50-59, 1993.

[3] A. Kongmunvattana and P. Chongstivatana, "A FPGA-based behavioral control system for a mobile robot," *Proceedings of IEEE Asia-Pacific Conference, Circuits and Systems*, pp. 759-762, 2000.

[4] B. Garbergs and B. Sohlberg, "Implementation of a state space controller in a FPGA", *Proceedings of IEEE Mediterranean*

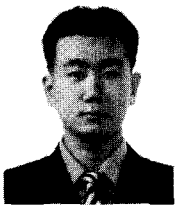
Electrotechnical Conference, pp. 566-569, 1998.
[5] J. Li and B. Hu, "The architecture of fuzzy PID gain conditioner and its FPGA prototype implementation", *Proceedings of IEEE Conference on ASIC*, pp. 61-65, 1996.
[6] *Avalon Bus Specification Manual*, Altera Inc. 2002.
[7] J. G. Bollinger and N. A. Duffie, *Computer Control of Machines and Processes*, Addison-Wesley, 1989.
[8] J. Becker, T. Pionteck, and M. Glesner, "Adaptive system-on-chip: architectures, technologies and applications", *Proceedings of IEEE Symposium on Integrated Circuits and Systems Design*, pp. 2-7, 1993.
[9] S. Knapp and D. Tavana, "Field configurable system-on-chip device architecture", *Proceedings of IEEE Conference on Custom Integrated Circuits*, pp. 155-158, 2000.
[10] *Nios 2.1 CPU, Datasheet*, Altera Inc., 2002.
[11] Altera Inc., "Estimating nios resource usage & performance," *Application Note*, 2002.
[12] *ARM7TDMI Core Processor, Datasheet*, ARM Inc., 2001.
[13] *Microcomputer Products. 16bit CISC H8S Family, Datasheet*, Hitachi Inc., Chapter 7, 2003.
[14] *MMC2001 M.CORE Integrated Microcontroller, Datasheet*, Motorola Inc., 1998.
[15] *MC68360 QUad Integrated Communication Controller, Datasheet*, Motorola Inc., 1993.
[16] *16bit Modular Microcontroller, Datasheet*, Motorola Inc., 1996.
[17] *XA 16-bit microcontroller family, Datasheet*, Philips Inc., 2000.
[18] S. A. Ito and L. Carro, "A comparison of microcontrollers targeted to FPGA-based embedded applications," *Proceedings of IEEE Conference on Integrated Circuits and Systems Design*, pp. 397-402, 2000.
[19] D. E. Simon, *An Embedded Software Primer*, Addison-Wesley, 2001.
[20] C. H. Roth, Jr., *Digital Systems Design Using VHDL*, Thomson Learning, 1998.
[21] Navabi, *Analysis and Modeling of Digital Systems*, McGraw Hill, 1998.
[22] S. Cardelli, M. Chiodo, P. Giusto, A. Jurecska, L. Lavagno, and A. Sangiovanni-Vincentelli, "Rapid-prototyping of embedded systems via reprogrammable devices", *Proceedings of IEEE Conference on Rapid System Prototyping*, pp. 133 -138, 1996.
[23] *Quartus II Software Quick Start Guide for Quartus II Software Version 2.1, Manual*, Altera Inc. 2002.

최 기 홍

1977년 4월 11일생. 2001년 인하대 자동화공학과(공학사). 2003년 동 대학원 자동화공학과(공학석사). 2003년 ~ 현재: LG산전 자동화연구소 연구원. 관심분야는 내장형 시스템, SOC설계, 로봇 제어, 실시간 OS.

박 재 현

1963년 10월 8일생. 1986년 서울대 제어계측공학과(공학사). 1988년 동 대학원(공학석사). 1994년 동 대학원(공학박사). 1995년~현재 인하대 정보통신공학부, 부교수. 관심분야는 내장형시스템, 실시간 시스템, 컴퓨터 네트웍, 이산현상 시스템.



최 영 준

1977년 3월 20일생. 2002년 인하대학교 자동화공학과 졸업 (공학사). 현재 인하대학교 정보통신대학원 재학. 관심 분야는 내장형 시스템, SOC설계, 실시간 통신, 실시간 OS.