FPGA를 이용한 범용 모션 컨트롤러의 개발

Development of a General Purpose Motion Controller Using a Field Programmable Gate Array

김 성 수, 정 슬*
(Sung-Su Kim and Seul Jung)

Abstract : We have developed a general purpose motion controller using an FPGA(Field Programmable Gate Array). The multi-PID controllers and GUI are implemented as a system-on-chip for multi-axis motion control. Comparing with the commercial motion controller LM 629, since it has multi-independent PID controllers, we have several advantages such as space effectiveness, low cost and lower power consumption. In order to test the performance of the proposed controller, motion of the robot hand is controlled. The robot hand has three fingers with 2 joints each. Finger movements show that tracking was very effective. Another experiment of balancing an inverted pendulum on a cart has been conducted to show the generality of the proposed FPGA PID controller. The controller has well maintained the balance of the pendulum.

Keywords : motion controller, FPGA, VHDL, robot hand, inverted pendulum

I. 서론

일반적으로 모터를 제어하는데 있어서 가장 많이 사용되는 것은 고성능의 마이크로 프로세서나 DSP이다. 그러나 이러한 방법은 CPU의 처리속도, 하드웨어적인 제약에 의해 센서 인터페이스, 사용자 인터페이스, 다른 프로세서와의 통신 등과 같은 다수의 작업들을 수행하기에는 한계가 있다. 특히 다수의 관절을 갖는 로봇 시스템과 같이 여러 개의 모터 제어가 필요한 경우 그 한계는 명확히 드러난다. 이러한 문제점에 의해 메인 프로세서를 따로 두고 각각의 모터 제어를 위한 하위 프로세서를 둠으로써 메인 프로세서의 부담을 줄이는 방법이 많이 사용되고 있지만, 이것 역시 컨트롤러 보드의 크기를 증가시키고 많은 전력이 필요하게 되며 설계 보드의 단가가 증가되는 단점을 갖고 있다.

FPGA 는 HDL (Hardware Description Language)을 이용한 설계 방법 때문에 사용자의 의도 및 목적에 따라 최적의 설계가 가능하고 개발 시간이 단축되며 시스템의 용도, 구성, 목적의 변화에 쉽게 대처 및 변경이 용이한 장점을 갖고 있다. 또한 시스템 설계 단가 면에서도 매우 효과적이다. 이러한 FPGA의 등장으로 디지털 회로 및 여러 종류의 응용 보드가 설계자의 용도와 목적에 최적으로 적합한 설계가 가능해지고, 유연한 설계가 가능해졌다.

최근에는 FPGA를 이용한 컨트롤러 및 사용자 정의 보드 설계에 관한 연구가 많이 이루어지고 있다. 신경망[1,2], 퍼지[3]와 같은 제어기의 하드웨어적 구현, 모터 엔코더 가운터와 같은 측정 회로[4,5], 다 관절 로봇 및 이동 로봇에서의 모터 제어[6,7]와 같은 분야가 그 대표적인 예라 할 수 있겠다.

논문 [6]은 보행 로봇의 모션 제어에 적용한 경우로서, 12개의 RC 서보 모터를 제어하기 위해 FPGA를 사용하였다. 이 경우 다수의 모터 제어 신호의 생성을 위해 I/O 포트가 충분히 많은 FPGA를 사용함으로써 작은 크기의 모터 제어 보드를 설계하였다. 그러나 여기서는 오픈 루프 제어가 사용되었다. [7]은 이동 로봇의 제어를 FPGA로 구현한 경우로서 개미 로봇의 상황별 동작을 FPGA를 이용하여 설계한 경우이다.

본 논문에서는 FPGA (Field Programmable Gate Array)를 이용한 범용 다축 모션 컨트롤러의 개발에 관한 내용을 다룬다. 하나의 FPGA를 이용한 모션 컨트롤러를 사용하면 제어보드의 부피를 줄일 수 있고, 개발 비용도 많이 절감할 수 있다. 개발된 모션 컨트롤러는 DC 모터 위치 제어용 컨트롤러로서 PID 제어를 이용하였으며, PC에서 각각의 모션 컨트롤러를 제어할 수 있도록 하기 위해 PC GUI(Personal Computer Graphic User Interface)를 개발하였다. PC GUI는 모션 컨트롤러 제어 메뉴뿐만 아니라, 모터의 움직임을 실시간으로 관찰하기 위한 시뮬레이터를 포함하고 있다. 모션 컨트롤러는 여러 개의 모터를 제어할 수 있으며, 메인 프로세서와 통신하기 위한 통신 블록과, 두 개의 모터 제어 블록으로 구성된다. 각각의 모터 제어 블록은 모터 엔코더를 4채배 카운트하는 카운터 블록, PID 제어를 이용한 컨트롤 블록, 그리고 모터 구동을 위한 PWM(Pulse Width Modulation) 발생 블록으로 구성된다.

본 논문의 모션 컨트롤러는 범용 목적으로 사용 가능하도록 개발되었으며, 제어해야 할 모터의 개수에 상관없이 적용 가능하다. 이러한 모션 컨트롤러의 성능을 검증하기 위해 본 논문에서는 두 가지의 실험을 하였다. 첫 번째는 2관절 3손가락 로봇으로서 3개의 모션 컨트롤러가 사용되며, 두 번째는 1자유도를 갖는 역진자로서 1개의 모션 컨트롤러가 사용된다.

II. 전체 시스템

그림 1은 PC GUI를 이용하여 DC 모터를 제어하는 전체 시스템을 나타낸다. 전체 시스템은 PC, MCU, 모션 컨트롤러로 구성된 로봇 제어 시스템과 로봇 시스템으로 구성된다.

* 책임저자(Corresponding Author)
논문접수 : 2003. 4. 17., 채택확정 : 2003. 9. 17.
김성수, 정 슬 : 충남대학교 메카트로닉스공학과
(a741103@hanmail.net/jungs@cnu.ac.kr)

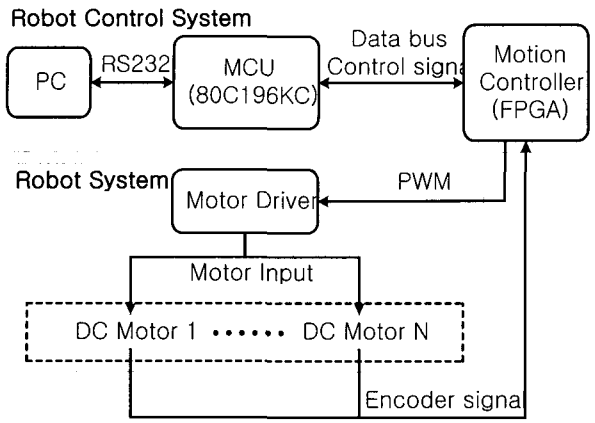


그림 1. 전체 시스템 블록도.
Fig. 1. Overall system block diagram.

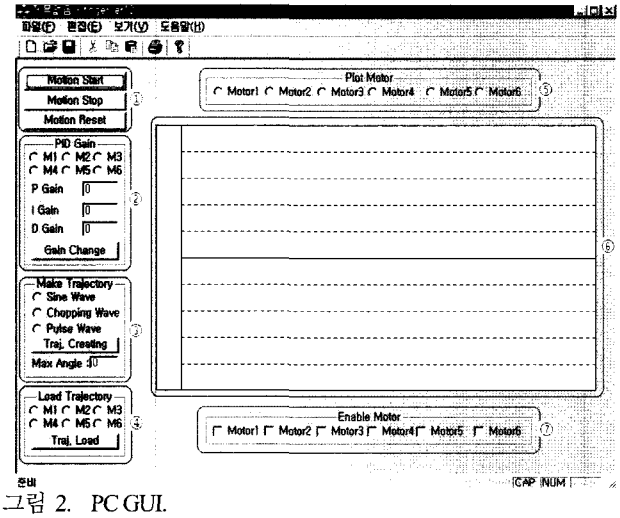


그림 2. PC GUI.
Fig. 2. PC GUI.

본 시스템은 PC GUI를 통해 로봇을 제어하는 것을 기본 목적으로 한다. PC GUI를 통해 로봇의 모션의 시작, 정지, 초기 화 등을 제어하며, 모션 컨트롤러의 PID이득값 설정, 로봇의 경로 생성, 로봇의 제어 상태 디스플레이와 같은 기능을 한다. PC와 모션 컨트롤러의 중간 버퍼의 역할로 80C196KC가 사용되며, PC로부터의 제어 명령을 모션 컨트롤러로 전송하고, 모션 컨트롤러로부터의 데이터를 PC로 전송하는 역할을 한다. 이때 PC와의 통신은 RS232 시리얼 통신을 이용하며, 모션 컨트롤러와의 통신은 8비트 데이터 버스 통신을 이용한다.

III. PC GUI

모터의 효과적이면서 시각적인 제어를 위해 모션의 움직임 제어 및 여러 파라메터 설정 또는 모니터링 할 수 있는 GUI 환경이 그림 2와 같이 제작되었다.

PC GUI의 세부적인 기능은 다음과 같다.

① 로봇 모션의 시작, 정지, 로봇 자세의 초기화와 같은 기본적인 로봇의 움직임을 제어할 수 있다.

② 모션 컨트롤러의 PID gain을 설정하는 부분으로 여러 개의 모터를 제어할 경우 모터의 특성 및 모터에 걸리는 부하 등에 따라 이득값이 다르게 설정되어야 할 필요가 있기 때문에 모터 각각의 PID 이득값을 독립적으로 설정할 수 있도록 되어 있다.

③ 로봇 조인트의 경로를 생성하는 부분으로, 일정 주기를 갖는 사인파, 삼각파, 그리고 펄스파를 생성할 수 있다. 최대 각도를 입력하여 경로를 생성하기 때문에 로봇 조인트의 움직임의 범위를 조절할 수 있다.

④ 로봇 조인트 경로를 다운로드 하는 GUI로서 로봇 조인트를 선택하여 ③에서 생성된 경로를 다운로드 할 수 있다. 이때 다운로드되는 경로는 MCU의 메모리에 저장되며, MCU가 일정시간마다 모션 컨트롤러의 제어 입력으로 설정한다.

⑤ ⑥에서 로봇의 특정 조인트의 움직임이 디스플레이 되는데, 여기에서 디스플레이 될 로봇 조인트를 선택하도록 되어 있다. 본 시스템에서 사용되는 PC와 MCU 간의 통신 방법은 RS232를 이용한 시리얼 통신이다. 통신 속도가 제한되어 있기 때문에 실시간으로 모든 조인트의 데이터를 읽어 오기에는 시간적인 부하가 많이 걸리게 된다. 그래서 특정 조인트 하나만을 선택하여 데이터를 읽어서 디스플레이 함으로써 통신상의 시간적인 부하를 최소화 하였다.

⑥ 로봇 조인트의 측정된 엔코더 값과 제어 입력 값을 그래프 상에 디스플레이 하게 된다. 실시간으로 로봇 조인트의 제어 입력값과 측정값이 동시에 그려지므로 제어 상태를 실시간으로 관측할 수 있다.

⑦ 6개의 로봇 조인트를 각각 enable/disable하는 부분이다. 위의 GUI를 통해 모션의 시작, 정지, 초기화와 같은 로봇의 기본적인 제어가 가능하며, 모션 컨트롤러의 PID gain 설정, 경로 생성 및 다운로드의 기능을 갖고 있다. 그리고 그래프를 통해 모터의 움직임을 실시간으로 디스플레이 함으로써 모터의 움직임을 정확하게 볼 수 있으며, 그래프를 통해 모터의 움직임을 관찰하면서 PID 이득값을 변경함으로써 효과적인 이득값 설정이 가능하다. 또한 여러 개의 로봇 조인트를 GUI 상에서 독립적으로 제어가 가능하기 때문에 다양한 로봇의 움직임을 표현할 수 있다.

IV. 모션 컨트롤러 시스템

1. 전체 제어 블록

모션 컨트롤러는 MCU와의 통신을 담당하기 위한 통신 블록과 2개의 모터 위치 제어를 위한 모터 컨트롤 블록, 그리고 trigger 신호를 발생하는 블록으로 구성되며, 개발 환경에서 설계된 모션 컨트롤러의 전체 설계 화면을 나타내고 있다. 그림 3에서 통신 블록은 MCU와의 통신을 담당하는 블록으로서, MCU로부터의 여러 가지 명령을 해독하여 모션 컨트롤러의 동작을 제어하거나, 모션 컨트롤러의 데이터를 MCU로 전송하는 역할을 한다. 모터 컨트롤 블록은 2개의 모터를 제어를 위해 PID 컨트롤러가 포함된 모터의 위치 제어기가 설계되었으며, Trigger신호 발생 블록에서는 샘플링 시간마다 트리거 신호를 발생함으로써 모터 컨트롤 블록에서 PID 제어

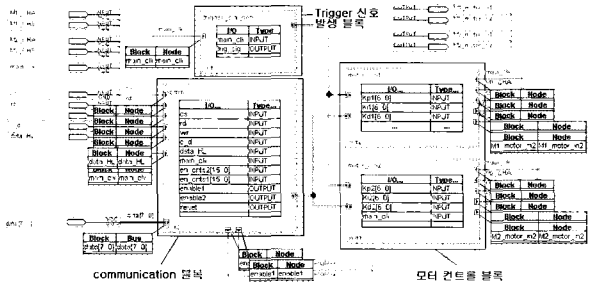


그림 3. Quartus II2.0 상에서의 모션 컨트롤러의 설계.

Fig. 3. The design of motion controller on Quartus II 2.0.

표 1. 핀 이름과 각각의 기능.

Table 1. The pin name and its function.

입,출력	핀 이름	기능
입력	M1_CHA	Motor 1 encoder phase A
	M1_CHB	Motor 1 encoder phase B
	M2_CHA	Motor 2 encoder phase A
	M2_CHB	Motor 2 encoder phase B
	main_clk	FPGA system clock
	Cs	Chip select
	Rd	Read
	Wr	Write
	c_d	Command, data 구분 신호
	data_HL	상위, 하위 바이트 구분 신호(16비트 데이터)
출력	M1_motor_in1	Motor 1 output 1
	M1_motor_in2	Motor 1 output 2
	M2_motor_in1	Motor 2 output 1
	M2_motor_in2	Motor 2 output 2
	enable1	Motor 1 enable
	enable2	Motor 2 enable
양방향	data[7..0]	8bits data bus

시기를 판단할 수 있도록 하였다. 여기에서 트리거 신호는 1 kHz의 주파수를 갖는다. 즉 PID 제어기는 1ms마다 연산을 수행하게 된다.

표 1에서는 개발된 모션 컨트롤러의 입,출력 핀과 각 핀의 기능을 보여주고 있다.

입력 신호로는 엔코더 신호, FPGA의 시스템 클럭, MCU와의 데이터 통신을 위한 제어 신호가 있고, 출력은 모터 구동 신호가 있다. 그리고 양방향 신호로서 8비트 데이터 버스가 있다.

2. 통신 블록

MCU와의 통신을 담당하는 블록으로서, MCU로부터의 여러 가지 명령을 해독하여 모션 컨트롤러의 동작을 제어하거나, 모션 컨트롤러의 데이터를 MCU로 전송하는 역할을 한다. 따라서 MCU→ 모션 컨트롤러의 데이터 전송과 모션 컨트롤러→ MCU의 데이터 전송으로 크게 나누어진다. 모든 제어 신호는 MCU에서 생성되므로 MCU에서 모션 컨트롤러로 데이터 쓰기와 MCU에서 모션 컨트롤러로부터 데이터 읽기로 나눌 수 있다.

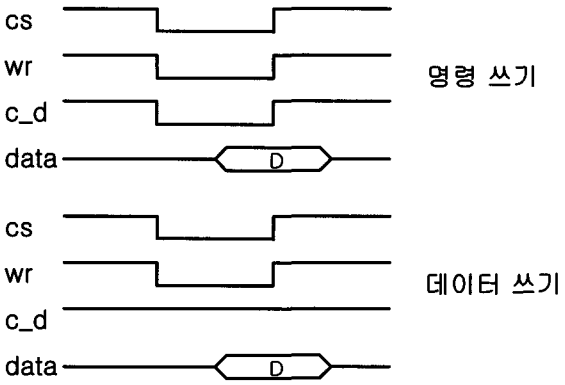


그림 4. 명령/데이터 쓰기 동작.

Fig. 4. The operation of command/data write.

표 2. 핀 이름과 각각의 기능.

Table 2. The pin name and its function.

Command	Data	Command	Data
Motor 1 Kp	0x01	Motor 2 Kp	0x11
Motor 1 Ki	0x02	Motor 2 Ki	0x12
Motor 1 Kd	0x03	Motor 2 Kd	0x13
Motor 1 Yd	0x04	Motor 2 Yd	0x14
Motor 1 enable	0x05	Motor 2 enable	0x15
Motor 1 disable	0x06	Motor 2 disable	0x16
Read encoder Motor 1	0x07	Read encoder Motor 2	0x17
FPGA reset	0x08		

2.1 MCU→모션 컨트롤러 데이터 쓰기

MCU에서 모션 컨트롤러로의 데이터 쓰기는 PID 이득값, PID 컨트롤러의 제어 입력과 같은 데이터 쓰기와 모터 enable/disable, 모션 컨트롤러 reset과 같은 제어 명령으로 나누어진다.

데이터 쓰기의 경우 다음의 순서에 의해 모션 컨트롤러에 데이터를 쓸 수 있다.

• 명령 write→ 데이터 write

표 2와 같이 쓰고자 하는 데이터에 따라 명령 데이터가 구분된다. 그리고 이어서 데이터를 따라서 쓰면 모션 컨트롤러 내부의 레지스터가 변경된다. 여기에서 명령인지 데이터인지를 구분하기 위해 그림 4와 같이c_d 핀을 이용한다. Write 동작에서 c_d핀이 'L'일 경우 명령 쓰기이고, c_d핀이'H'일 경우 데이터 쓰기가 된다.

제어 명령의 경우 위와 비슷한 방법으로 역시 표 2에 보여지는 명령 데이터로 명령 쓰기만 하면 된다.

2.2 MCU←모션 컨트롤러 데이터 읽기

MCU에서 모션 컨트롤러로부터의 데이터 읽기는 모터 1,2의 엔코더 카운터값을 읽는 것이다. 이 경우 다음과 같은 순서에 의해 모션 컨트롤러로부터 데이터를 읽을 수 있다.

• 명령 write→ 데이터 read(상위)→ 데이터 read(하위)

데이터 읽기의 경우 설계된 엔코더 카운터가 16비트이기 때문에 상위, 하위 바이트로 나누어서 읽어야 된다. 여기에서 상위 바이트와 하위 바이트를 구분하기 위해 그림 5와 같이 data_HL핀을 이용한다.

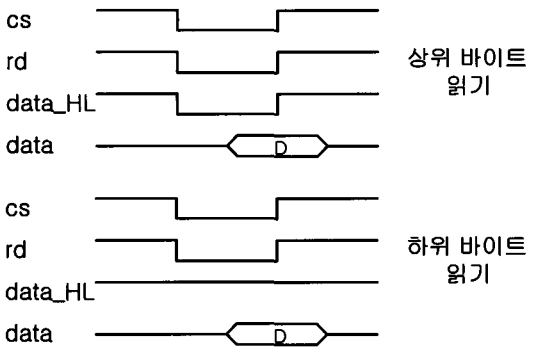


그림 5. 데이터 읽기 동작.

Fig. 5. The operation of data read.

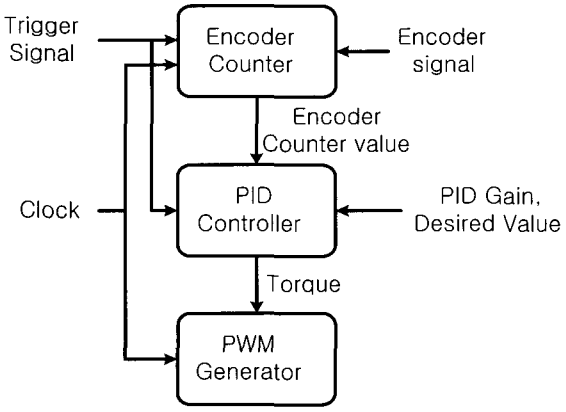


그림 6. 모터 제어 블록 구조.

Fig. 6. The structure of motor control block.

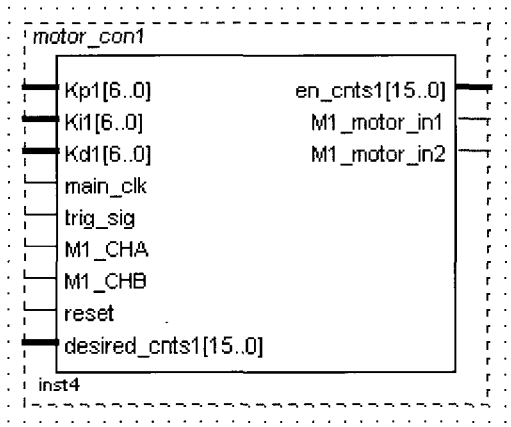


그림 7. 설계된 모터 컨트롤 블록 심벌.

Fig. 7. The designed symbol of motor control block.

3. 모터 컨트롤 블록

그림 6은 모터 제어 블록의 구조를 좀 더 세부적으로 나타내고 있다. 모터 컨트롤 블록은 엔코더 카운터 블록, PID 컨트롤러 블록 그리고 PWM 발생기 블록으로 구성된다.

그림 7은 설계된 모터 컨트롤 블록의 심벌을 나타낸다. 입력 신호로는 PID 이득값, 시스템 클럭, 트리거 신호, 엔코더 신호, PID 제어기의 기준 값, 그리고 리셋 신호이다.

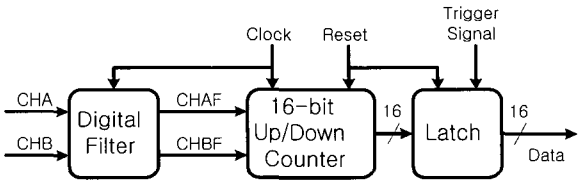


그림 8. 엔코더 카운터 구조.

Fig. 8. The structure of encoder counter.

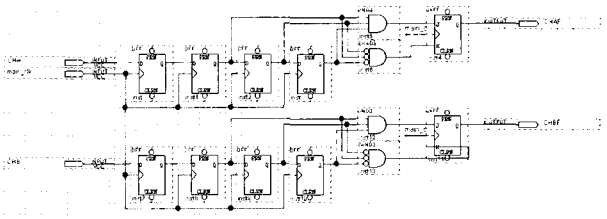


그림 9. 엔코더 블록의 디지털 필터.

Fig. 9. The digital filter of encoder counter block.

여기에서 리셋 신호는 엔코더 카운터를 초기화하기 위한 신호이다. 그리고 출력 신호로는 모터 구동 신호와 엔코더 카운터 값이 있다.

3.1 Encoder Counter 블록

본 논문에서 디자인된 엔코더 카운터는 A, B 두 상의 엔코더 신호의 위상차를 검출하여 16비트 정, 역 카운트가 가능하며, 엔코더 신호 한 주기당 4채배 카운트 된다. 그림 8은 엔코더 카운터 블록의 구조를 나타낸다. 모터 엔코더 카운터 블록은 모터와 같은 기계적인 시스템과 연결되기 때문에 노이즈 신호가 엔코더 신호에 포함될 수 있다. 이러한 노이즈를 제어하기 위한 디지털 필터가 그림 9처럼 설계되었다.

$1\,ms$ 마다 발생되는 트리거 신호에 의해 카운터 값이 출력된다.

3.2 PID Controller 블록

모션 컨트롤러는 제어 알고리즘이 간단하며, 가장 광범위하게 사용되는 PID제어기를 사용하였다. PID 제어기의 적분텀은 한계값을 두었으며, 수식들은 다음과 같다.

$$\tau(n) = K_p e(n) + K_i s(n) + K_d \{e(n) - e(n-1)\} \tag{1}$$

$$s(n) = \begin{cases} s_i & \sum e(n) > s_i \\ \sum e(n) & -s_i \le \sum e(n) \le s_i \\ -s_i & \sum e(n) < -s_i \end{cases} \tag{2}$$

$\tau(n)$ 컨트롤러의 출력, $e(n)$ 에러, K_p, K_i, K_d P, I, D 이득값, s_i 적분텀 한계값을 나타낸다. 이 블록에서는 제어 샘플링 시간마다 기준값과 엔코더 측정값의 오차를 계산하여 모터 제어량을 계산한다.

3.3 PWM Generator 블록

PID Controller에서 생성되는 모터 제어량을 모터의 입력 신호로 변환한다. 모터의 입력 신호는 PWM으로 만들어지며, 그 주파수는 12㎑이다. 다음 그림은 VHDL로 설계된 PWM generator를 보여주고 있다.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY pwm_generator IS
    PORT
    (   main_clk : IN STD_LOGIC;
        pwm_mag : IN integer range 0 to 255;
        pwm_signal : OUT STD_LOGIC  );
END pwm_generator;
ARCHITECTURE pwm_generator_architecture OF pwm_generator IS
    signal      pwm_count   :   integer range 0 to 256*8-1;
    signal      pwm_buff    :   std_logic;
BEGIN
    process(main_clk)
        variable    mag :   integer range 0 to 256*8-1;
    begin
        if main_clk = '1' and main_clk'event then
            mag := pwm_mag * 8;
            if pwm_count >= mag then
                pwm_buff <= '0';
            else
                pwm_buff <= '1';
            end if;
            pwm_count <= pwm_count + 1;
            pwm_signal <= pwm_buff;
        end if;
    end process;
END pwm_generator_architecture;
```

그림 10. PWM발생기의 VHDL코드.
Fig. 10. VHDL code of PWM generator block.

```
+---------------------------------+-----------------------+
|Processing status                |Fitting Successful     |
|Chip name                        |motion_controller      |
|Device name                      |EP20K300EQC240-2       |
|Total logic elements             | 2236 / 11520 ( 19 % ) |
|Total pins                       | 24 / 157 ( 15 % )     |
|Total ESB bits                   | 0 / 147456 ( 0 % )    |
+---------------------------------+-----------------------+
```

그림 11. 사용된 FPGA의 리소스.
Fig. 11. Whole resource amount used in FPGA.

본 시스템에서 모션 컨트롤러는 Quartus II 2.0 환경에서 VHDL로 제작되었으며, 하드웨어는 Altera사의 APEX II Series중 EP20K300EQC240이 사용되었다. 다음 그림 11은 모션 컨트롤러 개발에 사용된 LE(Logic Element)와 핀 수를 나타내고 있다.

3.4 LM 629와 비교

본 논문에서 제안하는 PID 제어기는 사용자 측면에서 편리하게 구성되었다는 장점이 있다. 일반적으로 대표적인 상용 모션 컨트롤러인 LM 629가 정확하고 function이 많은 장점이 있으나, 모터를 제어할 경우에는 그러한 function들이 불필요하므로 효율성이 떨어진다. LM629와 본 논문에서 개발된 모션 컨트롤러 시스템과의 차이는 가,감속 기능이다[8]. 그러나 모션 컨트롤러 시스템의 구성이 PC를 기반으로 하여 PC와 FPGA간에 실시간 통신이 이루어지기 때문에 PC에서 경로를 생성하여 FPGA로 다운로드 함으로써 가,감속이 가능해진다. 뿐만 아니라 PC GUI를 통한 제어가 이루어지기 때문에 여러 파라미터의 설정이 용이하며, 효과적인 제어가 가능하다.

표 3은 LM 629와 FPGA PID 제어기와의 기본적인 차이를 비교하여 보여준다.

표 3. LM 629와 비교
Table 3. comparison with LM 629

	제안하는 FPGA	LM 629
Pan outs	6 or more/chip	1/chip
GUI	GUI 제공	없음
속도 지정	PC에서 설정	칩에서 직접 설정
가격	effective	expensive
Functions	적음	불필요한 많음
사용법	쉬움	어려움
확장성	있음	없음

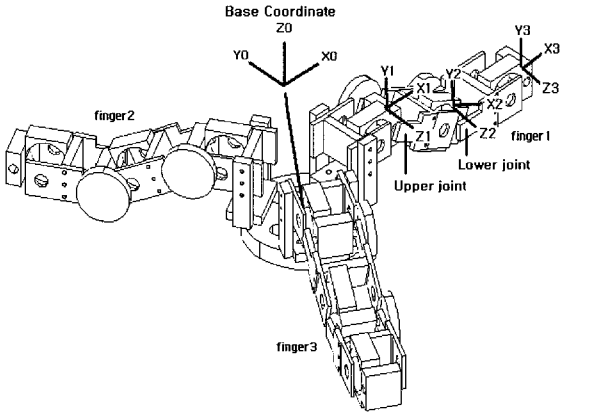


그림 12. 손가락 로봇의 좌표계 정의.
Fig. 12. Coordinate definition of finger robot.

표 4. 손가락 로봇의 DH 파라미터.
Table 4. DH parameter of finger robot.

Joint		θ_i(°)	d_i(㎜)	α_i(°)	a_i(㎜)
Finger1	1	0	37.5	90	62
	2	-25	0	0	55
	3	25	0	0	55
Finger2	1	120	37.5	90	62
	2	-25	0	0	55
	3	25	0	0	55
Finger3	1	240	37.5	90	62
	2	-25	0	0	55
	3	25	0	0	55

V. 실험

1. 손가락 로봇 시스템

로봇의 좌표계는 첫 번째 손가락의 경우 다음 그림 12와 같이 설정되었으며, 두 번째, 세 번째 손가락의 경우 역시 같은 방법으로 정의된다. 단지 두 번째 손가락의 경우 기준 좌표계를 중심으로 120도, 세 번째 손가락은 240도가 각각 회전하였다. DH 변수는 표 4와 같다.

그림 13은 손가락 로봇 시스템의 사진이다. FPGA 제어기와 손가락이 연결되어 있고 모니터에는 각 손가락 조인트의 움직임 실시간으로 그려지도록 되어 있다.

손가락 로봇 시스템은 먼저 로봇이 물체를 집는 형상을 보일 수 있는 경로를 생성하여 로봇에 적용하였다.

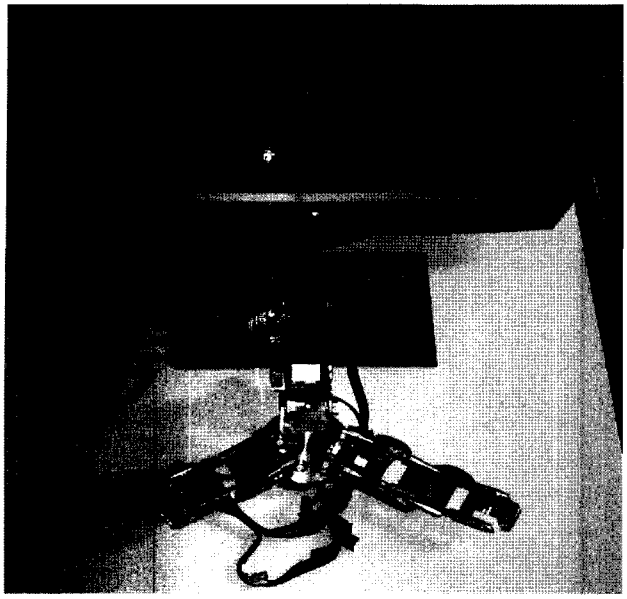


그림 13. 손가락 로봇 시스템.
Fig. 13. Finger robot system.

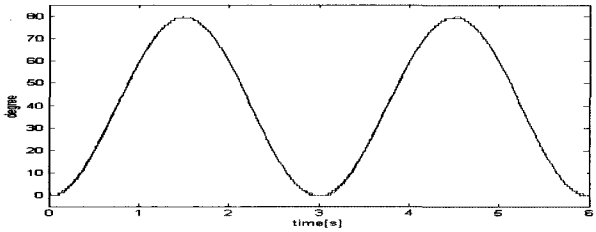


그림 14. 첫 번째 손가락 상위 조인트의 경로.
Fig. 14. The trajectory of upper joint of first finger.

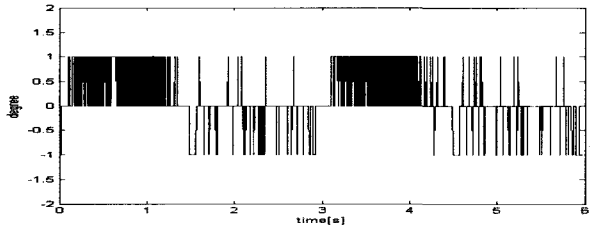


그림 15. 첫 번째 손가락 상위 조인트의 에러.
Fig. 15. The error of upper joint of first finger.

로봇이 물체를 집는 형상을 나타내기 위해 0~80도의 크기를 갖는 사인파를 로봇 손가락의 상위 조인트에 적용하고, 하위 조인트에는 0~40도의 크기를 갖는 사인파를 적용하였다. 사인파의 주파수는0.333Hz이다. 그림 14~25는 각 조인트의 실제 경로와 측정된 엔코더 값, 그리고 그 오차를 나타낸 그림이다. 그림14~19는 상위 조인트를 나타내며, 그림 20~25는 하위 조인트를 나타낸다. 실제 그림 14에서는 잘 추종하여 원하는 경로와 실제 움직임을 잘 구별할 수 없다.

따라서 그림 15에 오차를 따로 나타내었다. 오차는 대략 1도 정도가 되는 것을 알 수 있다.

위 실험에 사용된 PID이득값은 상위 조인트의 경우 1.875, 0.04, 1.875가 각각 사용되었고, 하위 조인트의 경우 0.6, 0.007,

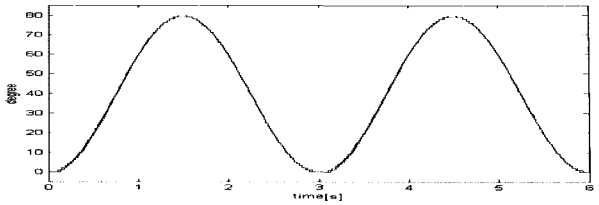


그림 16. 두 번째 손가락 상위 조인트의 경로.
Fig. 16. The trajectory of upper joint of second finger.

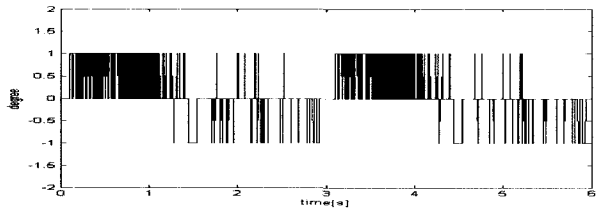


그림 17. 두 번째 손가락 상위 조인트의 에러.
Fig. 17. The error of upper joint of first finger.

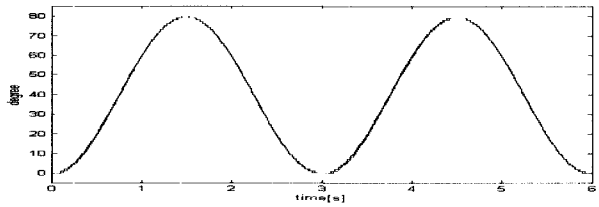


그림 18. 세 번째 손가락 상위 조인트의 경로.
Fig. 18. The trajectory of upper joint of third finger.

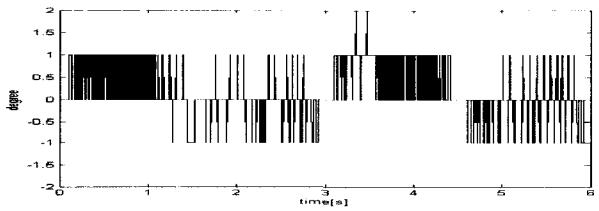


그림 19. 세 번째 손가락 상위 조인트의 에러.
Fig. 19. The error of upper joint of third finger.

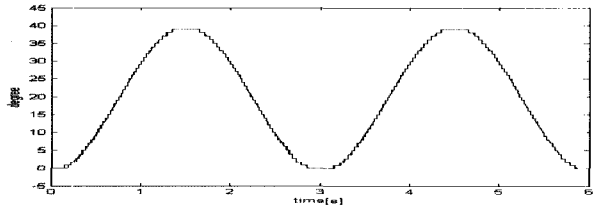


그림 20. 첫 번째 손가락 하위 조인트의 경로.
Fig. 20. The trajectory of lower joint of first finger.

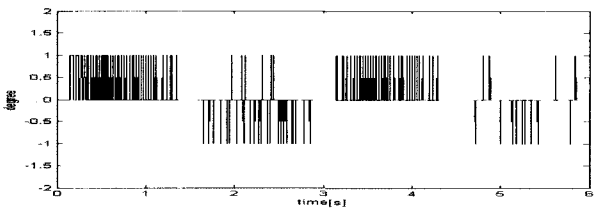


그림 21. 첫 번째 손가락 하위 조인트의 에러.
Fig. 21. The error of lower joint of first finger.

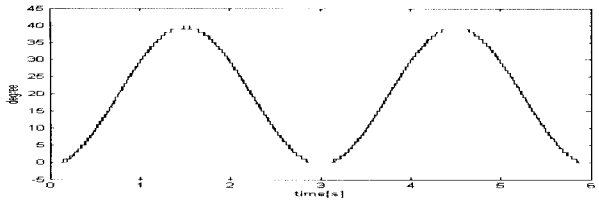


그림 22. 두 번째 손가락 하위 조인트의 경로.

Fig. 22. The trajectory of lower joint of second finger.

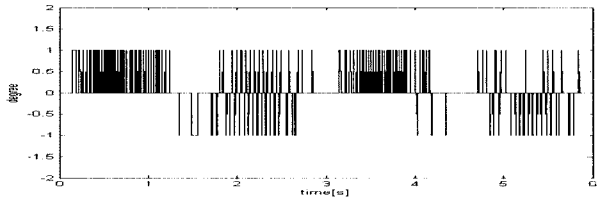


그림 23. 두 번째 손가락 하위 조인트의 에러.

Fig. 23. The error of lower joint of second finger.

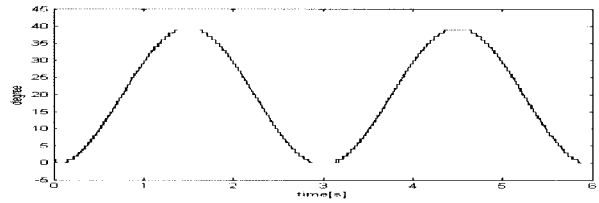


그림 24. 세 번째 손가락 하위 조인트의 경로.

Fig. 24. The trajectory of lower joint of third finger.

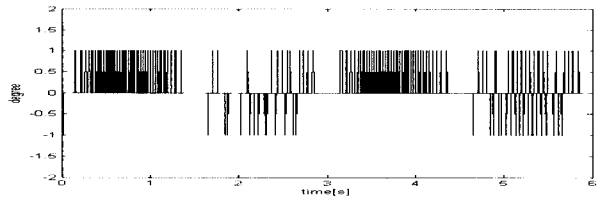


그림 25. 세 번째 손가락 하위 조인트의 에러.

Fig. 25. The error of lower joint of third finger.

0.6이 사용되었다. 상위 조인트와 하위 조인트에 걸리는 부하가 다르게 때문에 위와 같이 각각의 이득값을 설정하였다. 모든 조인트의 에러 데이터에서 보듯이 결과에서 보듯이 ± 2도 이내의 오차를 갖고 제어 입력을 추종하고 있다.

PD제어기를 사용하였을 때는 오차가 어느 정도 발생하였는데, PID제어기를 사용함으로써 오차를 영으로 수렴할 수 있었다.

위의 데이터를 바탕으로 손가락 로봇의 기구학을 해석하여 3차원으로 그려보면 다음 그림 26과 같은 결과를 얻을 수 있다. 로봇 손가락으로 물체를 잡는 움직임을 구현하였음을 알 수 있다.

또 다른 실험으로 주기적인 사인파의 주파수 변화에 따른 로봇의 추종 정도를 살펴보았다. 로봇의 경로로 0-60도의 값을 갖는 다른 사인파를 생성하여 로봇에 적용하였다. 사용된 주파수는 각각 0.667Hz, 1.000Hz이다. 다른 주파수를 갖는 사인파를 로봇의 상위, 하위 조인트에 각각 적용한 결과를 다음 그림 27-28에 나타내고 있다. 이 실험에 사용된PID 이득값은

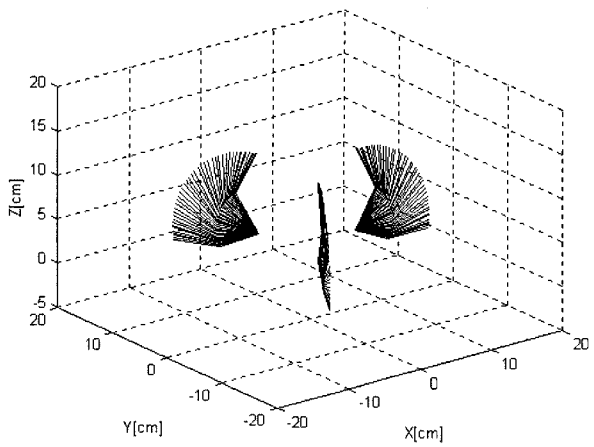


그림 26. 3차원 공간에서의 손가락 로봇의 움직임.

Fig. 26. The movement of finger robot in 3-dim space.

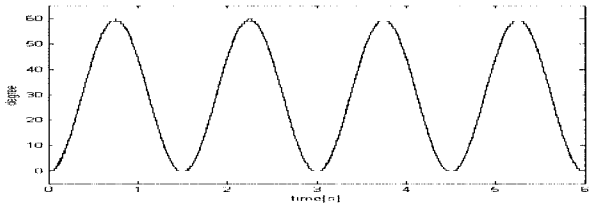


그림 27. 하위 조인트의 경로, 주파수 0.667Hz.

Fig. 27. The trajectory of lower joint, 0.667Hz.

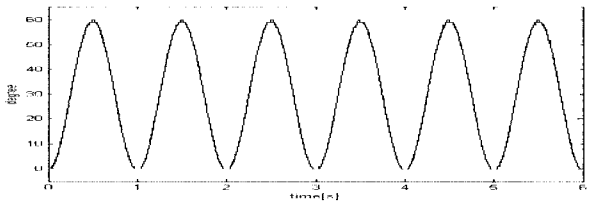


그림 28. 하위 조인트의 경로, 주파수 1.000Hz.

Fig. 28. The trajectory lower joint, 1.000Hz.

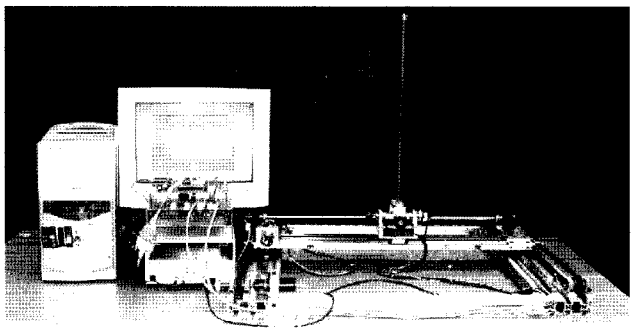


그림 29. 역진자 시스템 사진.

Fig. 29. Inverted pendulum system.

첫 번째 실험과 같은 값이 사용되었다.

주파수 변화에 상관없이 모든 경우에 대해서 ± 2도 이내의 최대 오차를 갖고 기준 제어 입력을 추종하고 있음을 보여 주고 있다.

2. 역진자 시스템

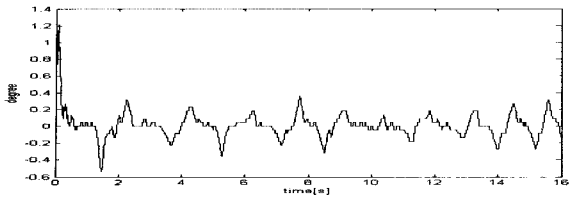


그림 30. 제어된 역진자의 각도.
Fig. 30. The angle of inverted pendulum.

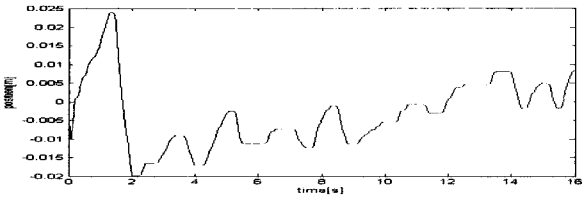


그림 31. 카트의 위치.
Fig. 31. The position of a cart.

다음 실험으로 역진자 제어를 수행하였다. 제안한 PID 제어기가 역진자 시스템에도 사용됨을 검증하기 위함이다. 그림 29는 역진자 시스템의 실제 모습이다.

역진자 시스템의 경우 진자의 각도와 위치의 제어 입력을 영으로 하는 실험을 하였다. 따라서 2개의 PID 위치 제어기가 필요하므로 1개의 모션 컨트롤러를 사용하였다. 그림 30과 31은 16초간의 역진자의 각도와 위치를 나타낸다.

역진자의 각도는 ± 0.4도 이내의 값으로 진동하고 있으며, 위치는 ± 0.02m이내의 값으로 진동한다. 하지만 진자의 각이 0을 기준으로 균형을 유지하고 있음을 알 수 있다. 비선형 SIMO시스템인 역진자의 경우 진자의 각도는 선형 제어기인 PID 제어를 이용하여 안전하게 제어가 가능하지만, 카트의 위치의 경우 PID 제어기로는 완전하게 제어되지는 않는다. 위치 제어와 각도를 동시에 제어하기위해 신경 회로망을 사용한 예가 있다[9].

VI. 결론

본 논문에서는 사용자 편의를 위해 FPGA를 이용한 PID 모션 컨트롤러를 개발하여, 손가락 로봇 시스템과 역진자 시스템에 적용하여 모션 컨트롤러의 성능을 검증하였다. FPGA 기반의 모션 컨트롤러를 개발함으로써 손가락 로봇과 같은 다 관절 로봇의 모션 제어의 경우 제어 보드의 부피를 현저히 감소할 수 있었고, 간단하게 설계를 할 수 있었다. 또한 VHDL을 이용한 개발 방법에 의해 시스템의 변경에도 유연하게 대처할 수 있었다. 본 논문에서 두 가지의 실제 시스템의 모션 제어에 FPGA제어기를 적용함으로써 실제 성능을 검증할 수 있었다. 개발된 모션 컨트롤러는 다양한 분야에 범용적으로 적용될 수 있음을 증명하였다.

참고문헌

[1] T. Aoyama, Q. Wang, R. Suematsu, R. Shimizu, U. Nagashima, "Learning algorithms for a neural network in FPGA", *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 1, pp. 1007–1012, 2002.

[2] M. Krips, T. Lammert, A. Kummert, "FPGA implementation of a neural network for a real-time hand tracking system", *Proceedings of the First IEEE International Workshop on Electronic Design, Test and Applications*, pp. 313–317, 2002.

[3] M. Cirstea, J. Khor, M. McCormick; "FPGA fuzzy logic controller for variable speed generators", *Proceedings of the 2001 IEEE International Conference on Control Applications*, pp. 301–304, 2001.

[4] A. K. Oudjida et al, "A reconfigurable counter controller for digital motion control application", *Microelectronics Journal*, vol. 28, no. 6-7, 1997.

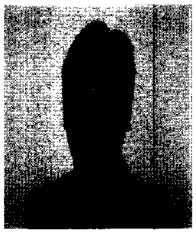
[5] F. Thomas et al, "Design and implementation of a wheel speed measurement circuit using field programmable gate arrays in a spacecraft", *Microprocessors and Microsystems*, pp. 553-560, 1999.

[6] S. N. Oh et al, "Design of a biped robot using DSP and FPGA", *Proceeding 2002 FIRA Robot World Congress*. 698-701.

[7] A. Kongmunvattana, P. Chongstivatana, "A FPGA-based behavioral control system for a mobile robot", *The 1998 IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 759-762, 24-27, Nov, 1998.

[8] "LM628/LM629 precision motion controller", *National Semiconductor Corporation*, Nov, 1999.

[9] 정슬, 임선빈, "비선형 시스템을 위한 신경망 제어기 구현", 제어 자동화 시스템 공학회, pp. 918-926, 제 7 권 제 11 호, 2001.



김 성 수

1974년 12월 22일생. 2001년 경일대학교 제어계측공학과 졸업. 2003년 충남대학교 메카트로닉스공학과석사. 관심분야는 S.o.C 제어기 설계, DSP 및 마이크로프로세서 응용, 로보틱스.

정 슬

1964년 9월 11일생. 미국 웨인 주립대학 전기 컴퓨터 공학과 졸업(1988), 미국 캘리포니아 데이비스 대학 전기공학과 석사 박사 (지능 로봇 전공)졸업(1996), 충남대 메카트로닉스공학과 부교수(1997-현재) 관심분야는 로봇 설계, 제작 및 제어, 지능 시스템 및 감성공학, S.o.C 제어기 설계.