고속 펄스 모터 콘트롤러 칩의 설계 및 구현

Design and Implementation of High Speed Pulse Motor Controller Chip

김 원 호, 이 건 오, 원 종 백, 박 종 식
(Won-Ho Kim, Gun-Oh Lee, Jong-Baek Won, and Jong-Sik Park)

Abstract : In this paper, we designed and implemented a precise pulse motor controller chip that generates the pulse needed to control step motor, DC servo and AC servo motors. This chip generates maximum pulse output rate of 5Mpps and has the quasi-S driving capability and speed and moving distance override capability during driving. We designed this chip with VHDL and executed a logic simulation and synthesis using Synopsys tool. The pre-layout simulation and post-layout simulation was executed by Compass tool. This chip was produced with 100 pins, PQFP package by 0.8㎛ gate array process and implemented by completely digital logic. We developed the test hardware board of performance and the CAMC(Computer Aided Motor Controller) Agent softwate to test the performance of the pulse motor controller chip produced. CAMC Agent enables user to set parameters needed to control motor with easy GUI(Graphic User Interface) environment and to display the output response of motor graphically.

Keywords: pulse motor controller, quasi-S driving capability, override capability, VHDL, gate array, CAMC Agent

I. 서론

펄스 모터 제어 칩은 산업용 로봇, CNC 장비, 반도체 제조 장치 등의 정밀 기계 장비의 구동 장치로 광범위하게 사용되고 있는 스텝 모터[1] 및 서보 모터[2]의 제어기 칩이다. 이 칩은 스텝 모터나 서보 모터의 제어에 필요한 펄스를 프로그램된 속도로 출력하고, 센서 등의 각종 입력신호에 의해서 출력을 제어하는 기능을 수행한다. 기계부에 무리한 토크나 충격이 가해지지 않도록 일정한 가속도 및 감속도를 유지하면서 평탄한 출력을 유지하고 모터의 회전상태에 따라서 발생되는 입력 신호의 노이즈를 제거하는 디지털 필터링[3]등을 수행하기 위해서는 많은 연산량이 요구되어 범용 마이크로프로세서에 의한 실시간 제어가 거의 불가능하다. 그러므로 필요한 연산을 하드웨어적으로 처리해주는 모터 제어 전용 칩[4]-[13]이 필요하다. 모터 제어 전용 칩은 가속과 감속을 제어하고 펄스를 발생하기 위해서 많은 연산을 해야 하므로 복잡한 구조로 인해 많은 팬아웃 문제와 시간 지연요소가 따른다. 본 논문에서는 이를 줄이기 위해서 여러 입력을 비교하는 비교 제어기, 가속과 감속을 제어하는 가속 · 감속 제어기, 가속도 비를 선택하는 비율 선택기, 그리고 펄스를 발생하는 펄스 발생기로 구분하여 설계하였다.

개발된 칩의 특징은 다음과 같다.

- 제어 대상 모터는 스텝 및 서보 모터이다.
- 최고 출력 주파수는 5*Mpps*이다.
- 사용자가 설정한 각종 파라미터의 값을 저장하여 펄스 출력 시작 신호만 주면 내부적으로 모든 계산을 하여 자동으로 동작한다.
- 펄스 출력 중 속도 변경, 이동량 변경이 가능하다.
- S자 드라이브 기능이 있어서 모터의 가 · 감속 시 모터에 가해지는 기계적 로드를 감소시킬 수 있다.
- 센서 입력 신호에 의한 실시간 펄스 출력 제어가 가능하다.
- 동기 맞춤 기능이 있어서 단일 CPU에 여러 개의 칩을 사용하여 다축 제어 시스템 구현이 가능하다.
- 모터 구동 정지 후에는 인터럽트 신호를 출력하여 CPU에게 모터의 구동이 끝났음을 알릴 수 있다.

II. 펄스 발생 알고리즘

모터의 제어에 필요한 펄스를 프로그램된 속도로 출력하기 위해서 펄스 발생 알고리즘을 개발하였다.

그림 1에서 보는 것과 같이 일정한 속도(목표속도 : OBJ_Vp)를 시간에 대하여 적분하면 그 값은 발생된 펄스의 수와 같다.

$$n(0:t) = \int_0^t v_p(t')dt' \quad (1)$$

여기서, $n(0:t)$은 0에서 t까지의 시간 사이에 발생된 펄스의 합이 되며, $v_p(t')$는 시간 t'에서 발생된 펄스의 속도를 나타낸다. v_p를 목표 속도(OBJ_Vp)와 주파수 단위(F_{UNIT} : Hz)의 곱으로 나타내면 다음과 같다.

$$v_p = \text{OBJ\_Vp} * F_{UNIT} \quad (2)$$

목표 속도를 1에서 8191까지의 속도로 지정 가능할 때 F_{unit}을 나타내면 다음과 같이 표시된다.

접수일자 : 1999. 2. 1., 수정완료 : 1999. 6. 14.
김원호 : 동의공업대학 전자과
이건오 : 아진전자산업 ASIC개발부
원종백, 박종식 : 경북대학교 전자 · 전기공학부

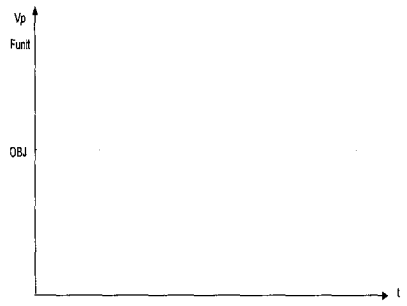


그림 1. 모터의 일정 속도 출력.
Fig. 1. Constant speed output of motor.

$$F_{UNIT} = \frac{F_{clk}}{RANGE \cdot 32,768} \simeq \frac{f_{clk}}{RANGE \cdot 32K} \quad (3)$$

여기서, f_{clk}은 입력 클럭 주파수이다.

(1), (2), (3)을 정리하면 다음과 같이 표현된다.

$$n(0:t) = \int_0^t \frac{OBJ}{RANGE*32K} f_{clk}\, dt' = \frac{OBJ}{RANGE*32K} \frac{t}{t_{clk}} \quad (4)$$

(4)는 모터의 목표 속도와 이에 해당하는 펄스 발생 수와의 관계를 나타낸다.

III. 펄스 발생부의 구현

모터의 목표 속도에 해당하는 펄스를 발생시키는 펄스 발생부를 디지털 로직으로 구현하기 위하여 기능별 블록을 구분하여 로직을 최적화하고 이를 VHDL[20]-[22]로 구현하였다.

펄스 발생부의 내부 구조는 그림 2와 같으며 이에 대한 주요 구성은 모터의 속도를 가속 및 감속하기 위한

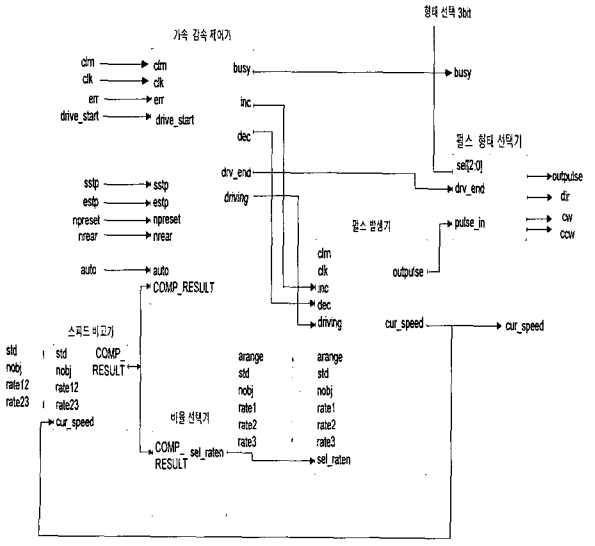


그림 2. 펄스 발생부의 블록도.
Fig. 2. Block diagram of pulse generation module.

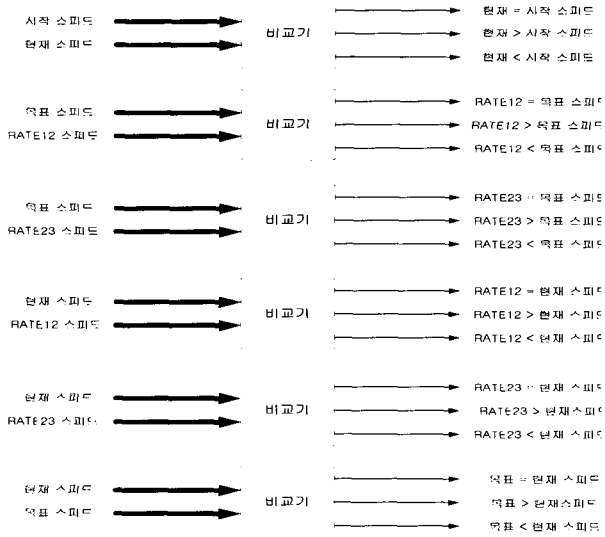


그림 3. 속도 비교 제어기.
Fig. 3. Speed compator.

가속·감속 제어기, 모터 속도의 출력 상태를 확인하기 위한 속도 비교기, 출력 속도를 선택하기 위한 비율 선택기, 펄스를 출력하기 위한 펄스 발생기, 그리고 다양한 펄스 출력을 선택하기 위한 펄스 형태 선택기로 구성되어 있다.

속도 비교 제어기에서는 각종 속도 설정 데이터를 사용하여 비교 결과를 출력해준다. 속도 비교 제어기는 시작 속도, RATE12, RATE23, 목표 스피드, 그리고 현재의 속도를 입력받아 현재 속도와 시작 속도, 현재 속도와 목표 속도, 현재 속도와 RATE12, 현재 속도와 RATE23, 목표 속도와 RATE12, 그리고 목표 속도와 RATE23의 대소관계를 출력한다.

비율 선택기는 펄스 발생기에서 사용하게 될 RATE1, RATE2, RATE3 중 1개의 속도 비율을 선택하게 해준다. 비율 선택기는 비교 제어기에서 출력된 비교 신호를 입력받아 현재 속도 데이터가 RATE12보다 작으면 RATE1을 선택, RATE12보다 크고 RATE23보다 작으면 RATE2를 선택, 그리고 RATE 23보다 크면 RATE 3을 선택하여 3가지의 가속 및 감속 비율을 결정한다.

가속·감속 제어기는 속도의 가속과 감속을 결정하여 준다. 비교 제어기의 현재 속도 데이터 출력과 목표 속도 데이터의 비교 값을 입력으로 받아 출력 펄스 속도의 상승, 일정, 감소를 제어하는 블록이다. 또한 구동 중 출력된 펄스의 수를 출력해 준다. 드라이브 시작 신호가 입력되면 DRV_CON(드라이브 제어)은 드라이브 신호를 활성화하고(preset_drv = '1') 드라이브 끝 신호가 입력되면 드라이브 신호를 비 활성화(preset_drv = '0') 한다.

가속·감속 제어기는 속도의 가속과 감속을 결정하여 준다. 비교 제어기의 현재 속도 데이터 출력과 목표 속도 데이터의 비교 값을 입력으로 받아 출력 펄스 속도의 상승, 일정, 감소를 제어하는 블록이다. 또한 구동 중 출력된 펄스의 수를 출력해 준다. 드라이브 시작 신호가 입력되면 DRV_CON(드라이브 제어)은 드라이브 신호를

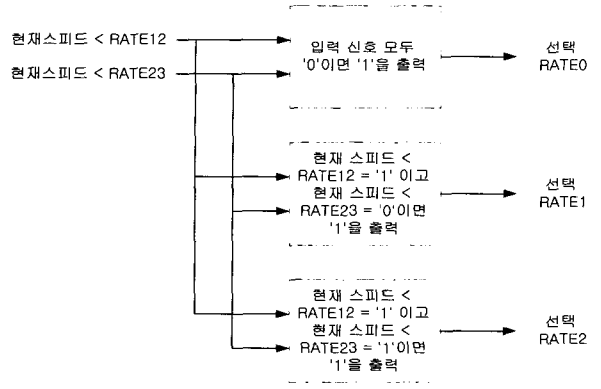


그림 4. 비율 제어기.
Fig. 4. Rate controller.

활성화하고(preset_drv = '1') 드라이브 끝 신호가 입력되면 드라이브 신호를 비 활성화(preset_drv = '0')한다.

펄스 발생기는 선택된 비율로 속도를 계산하여 펄스를 출력해 준다. 속도 설정 데이터들과 속도 가속, 등속, 감속 신호(INC, CONS, DEC)들을 참조하여 출력 펄스와 현재 속도 데이터를 출력한다. INTEGRATOR, SPEED_GEN, STEP_COUNTER 부로 구성된다.

SPEED_GEN 부는 시작 속도 데이터를 읽어서 TUNIT마다 발생되는 다음 속도 신호를 사용하여 가속 · 감속을 하고 현재 속도 데이터를 출력한다. INTER-

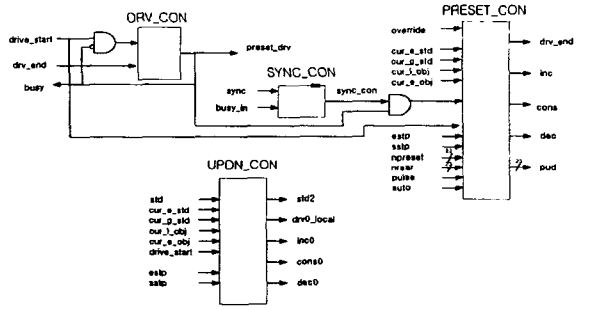


그림 5. 가속 · 감속 제어기.
Fig. 5. Acceleration and deceleration speed controller.

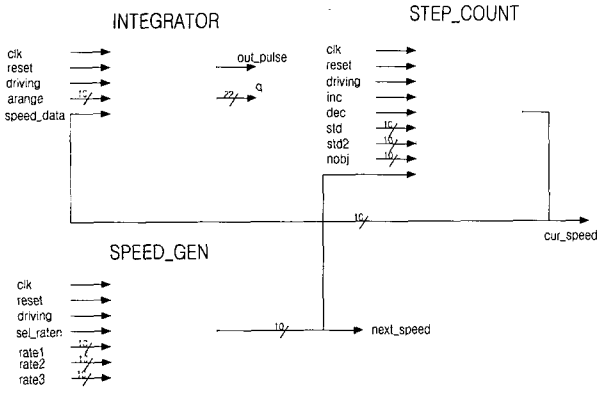


그림 6. 펄스 발생기.
Fig. 6. Pulse generator.

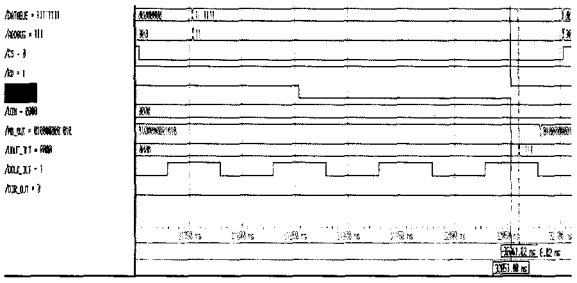


그림 7. CPU의 버스 제어신호를 이용한 출력 핀의 데이터 시뮬레이션.
Fig. 7. Data simulation of output pins using bus control signal of CPU.

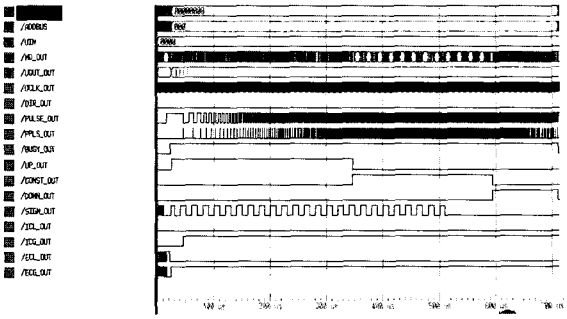


그림 8. 펄스 출력의 가속 및 감속 동작.
Fig. 8. Acceleration and deceleration speed simulation of pulse output.

GRATOR 부는 현재 스피드 데이터와 RANGE에 따라 50% duty cycle을 가지는 펄스를 출력한다. STEP_COUNTER 부는 RATE1, RATE2, RATE3 값 중에서 선택된 값을 사용하여 "TUNIT = 8*RATE" 펄스마다 1개의 펄스를 발생시켜 가 · 감속 시간을 설정 한다.

IV. 설계된 칩의 시뮬레이션

설계된 칩은 VHDL 로직 시뮬레이션 및 합성 후 시뮬레이션 결과를 확인하였다. Synopsys 시뮬레이터를 이용하여 평션 시뮬레이션을 수행하고 합성기를 이용하여 로직을 합성하였다. Compass를 사용하여 게이트 시뮬레이션을 하였으며 최종적으로 배치(place) 및 배선(route)의 과정을 거쳐 포스트(post) 시뮬레이션을 수행하였다. 또한 테스트 벡터(test vector)를 원활히 작성하고 시스템 레벨에서 시뮬레이션을 하기 위해 파형 편집기를 사용하지 않고 VHDL을 이용하여 테스트 벡터를 작성 후 시뮬레이션을 수행하였다.

그림 7에서 CPU의 버스 제어신호를 이용하여 칩의 출력 핀(UOUT_OUT)의 데이터가 변하는 것을 볼 수 있다. 출력을 "1111"로 바꾸기 위해 ADDBUS[2:0]에 "111", DATABUS[7:0]에 "11111111"을 입력한 후 WR(write) 신호를 "High-Low-High"의 순서로 변경하여 출력이 변경되는 것을 확인하였다. CPU의 버스 제어신호인 WR 신호가 끝나는 시점 보다 8.02nsec가 지연되어 변경된 UOUT_OUT 신호가 출력되는 것을 시뮬레이션에서 확인하였다.

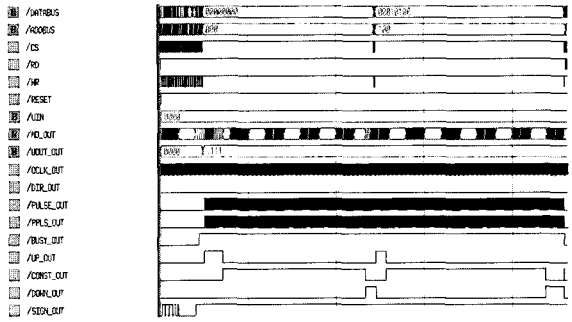


그림 9. 펄스 수 변경 시뮬레이션.
Fig. 9. Simulation for the change of pulse count.

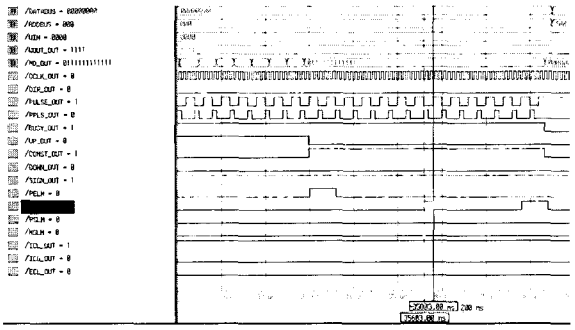


그림 10. 외부 급정지 신호를 입력했을 때의 시뮬레이션.
Fig. 10. Simulation result for emergency stop.

그림 8은 펄스 발생기의 내부 가속비, 감속비, 시작 속도, 목표 속도 그리고 출력 펄스 수를 입력하고 시작 명령을 내린 후 출력 펄스(PULSE_OUT)의 변화를 보여준다. 가속 신호(UP_OUT) 가 '1'인 곳에서는 출력 펄스의 출력 주기가 작아지며, 감속 신호(DOWN_OUT)가 '1'인 곳에서는 출력 펄스의 출력 주기가 길어지는 것을 시뮬레이션 결과로부터 확인할 수 있다.

그림 9는 펄스가 출력되는 중에 출력 펄스 수를 변경했을 때의 시뮬레이션 결과를 보여준다. 이 그림에서 프로그램된 펄스를 프로그램으로 변경하여 그 결과가 출력에 반영됨을 볼 수 있다. 이전 보다 더 많은 펄스 수가 설정이 되면 다시 목표 속도까지 가속(UP_OUT= '1')하여 출력되는 것을 시뮬레이션 결과로부터 확인할 수 있다.

그림 10은 펄스 출력 중에 정지 신호가 입력되었을 때의 시뮬레이션 결과이다. 이는 프로그램 된 펄스가 출력 중에도 센서의 출력 신호에 의해서 모터를 정지시킬 수 있다는 것을 보여준다. 또한 급정지 신호(NELM = '1')가 200nsec 이하일 경우 노이즈로 판단하여 제거하며 다음에 200nsec보다 큰 신호가 입력되면 펄스의 출력이 종료되는 것을 시뮬레이션 결과로부터 확인할 수 있다.

V. 칩의 성능 실험 및 결과

1. 모터 제어 칩 성능 측정 하드웨어 개발

펄스 모터 제어 칩은 VHDL(Very High Speed Integrated Circuit Hardware Description Language)로 설계하고 0.8㎛ 공정의 SOG(Sea of Gate)[15]를 이용하여 구현하였다. SOG로 구현하기 위하여 Synopsys[16]-

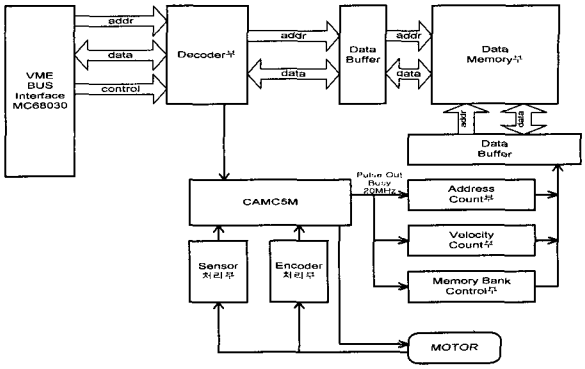


그림 11. 칩 성능 측정 보드 하드웨어 블록도.
Fig. 11. Block diagram of chip perfrmance test board.

[18]를 이용하여 로직을 합성하였으며 Compass[19]를 이용하여 배치(place) 및 배선(route) 후 사용전압이 4.75V, 동작온도가 70°C인 환경에서 최종 worst case 시뮬레이션을 수행하여 원하는 제어 성능을 검증하였다. 제작된 칩은 23,000 게이트를 가지며, 100핀 QFP 패키지를 사용하여 제작하였다.

제작된 칩의 성능을 확인하기 위하여 칩 성능 측정 보드를 개발하였다. 이 보드는 칩이 지정된 모드로 구동될 때 실제 출력되는 펄스 수의 주기를 20MHz의 메인 클럭에 동기시켜 어드레스를 증가시키면서 메인 클럭의 1/2 주기에 해당하는 시간에 발생된 펄스의 수를 데이터로 저장한다. 즉, 펄스가 출력될 때에는 어드레스 버스와 데이터 버스를 CPU 버스 신호와 분리하여 자체적으로 메모리 뱅크를 관리하면서 데이터를 저장한다. 펄스 출력이 완료되었다는 신호는 칩의 BUSY 신호를 이용하여 검출한다. BUSY 신호가 검출되면 버스 신호를 CPU 버스로 절체시킨 다음 메모리 뱅크에 저장된 데이터를 읽도록 하는 DMA 제어 방식으로 설계되었다. 이러한 제어 로직은 칩 성능 측정 보드 내에서 FPGA(Field Programmable Gate Array)[23]로 구현하였다.

그림 11은 칩 성능을 측정하기 위한 하드웨어 블록도이다.

2. 모터 제어칩 성능측정 소프트웨어(CAMC Agent) 개발

모터 제어 칩을 그래픽 환경에서 편리하게 사용하기 위하여 모터 제어 칩 성능 측정 소프트웨어인 CAMC Agent[24]를 개발하였다. 이 CAMC Agent는 모터를 제어하기 위해 필요한 파라미터들을 PC의 GUI(Graphic User Interface) 환경에서 설정할 수 있으며, 실행 결과 모터의 출력 응답을 그래프로 볼 수 있도록 해준다.

CAMC Agent의 주요 기능은 다음과 같다.

- 설정 데이터에 의한 계산 파형과 실제 구동 데이터에 의한 실제 측정 파형을 비교 분석하는 기능
- Step Motor, DC 및 AC Servo 모터 등 각종 모터 구동 시험을 위한 모드 지원
- 설정된 파라미터를 이용한 C-소스 코드 자동 생성 기능
- HTML을 이용한 온라인 사용자 매뉴얼 기능

• 펄스 모터 제어 칩의 내부 레지스터 상태 도시 기능

CAMC Agent를 실행하기 위한 기본 환경으로는 Windows 95(Internet Explorer 4.01이 설치되어야 함), Windows 98, Windows NT 4.0 이상 중 하나가 설치되어 있어야 한다. 또한 모터 제어 칩 성능 측정 보드와는 직렬 포트에 의해 연결되며, 화면상의 데이터를 프린터로 출력할 수 있다.

3. 모터 제어 칩 측정 실험 결과

1) 지정 펄스 구동 방식

지정 펄스 구동 방식은 지정된 펄스 수만큼 모터를 구동시키는 방식으로 그림 12는 지정 펄스 수 구동 모드에서 파라미터를 설정하기 위한 CAMC agent 화면이다.

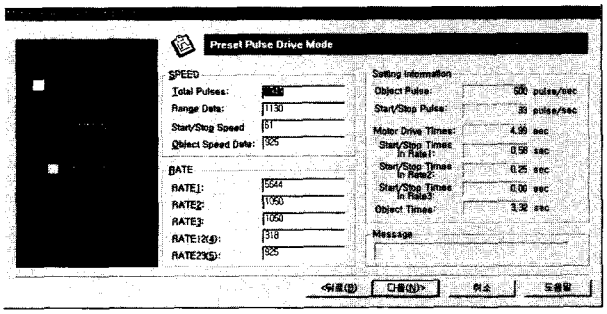


그림 12. 지정펄스구동 모드에서 파라미터 설정 화면.

Fig. 12. Parameter set in preset pulse drive mode.

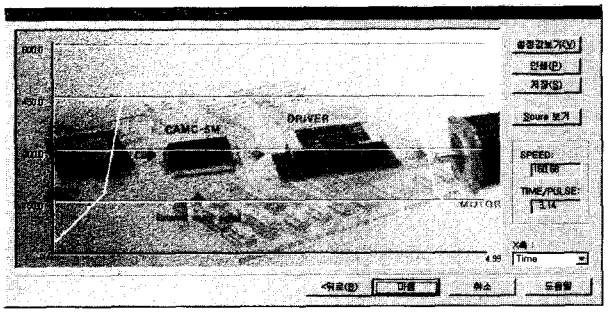


그림 13. 지정펄스구동모드에서 이론치로 계산한 모터의 출력 응답.

Fig. 13. Computed output response in preset pulse drive mode.

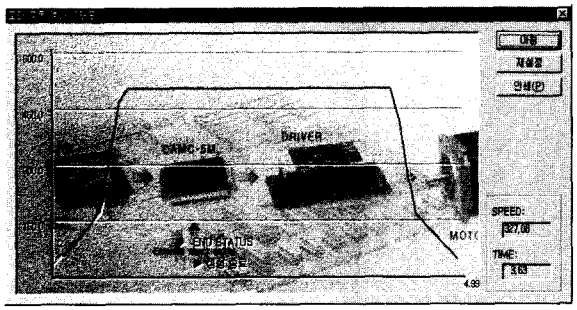


그림 14. 펄스 지정 방식에서 모터의 실제 출력 응답.

Fig. 14. Measured output response in preset pulse drive mode.

"Range Data"에 의해 출력 주파수의 단위 주파수가 생성되고, 시작 및 정지 속도 데이터, 목표 속도 데이터에 단위 주파수를 곱하여 출력 속도를 설정한다. 기존 "RATE" 데이터를 설정하여 가속·감속 시간 설정 단위를 결정할 수 있으며, 이 가속·감속 시간 설정 단위를 이용하여 시간에 대한 속도 변화를 계산할 수 있다. 시간(x축) 대 속도(y축) 그래프에서 구간별 면적의 합은 이동거리를 나타내며 이것은 모터를 구동할 때 발생되는 총 펄스 수와 같다.

그림 12의 파라미터 설정에 의해 이론치로 계산한 모터의 출력 응답은 그림 13과 같다.

CAMC agent에서 파라미터를 설정하여 모터 제어 보드로 데이터를 전송하면 모터 제어 보드에서는 각 파라미터를 분석하여 모터 제어 칩 레지스터를 제어하고, 모터 제어 칩은 자동적으로 해당된 수만큼의 펄스를 출력시킨다.

그림 14에서 실제의 모터 출력과 이론치로 계산한 모터 출력이 일치하는 것을 확인할 수 있다.

2) 연속 구동 방식

연속 구동 방식은 프로그램에 의해 정지 명령이 입력될 때까지 구동한다. 가·감속 구동 및 일정 속도 구동이 가능하다. "Range Data"와 "RATE" 데이터를 설정하여 총 펄스 수와 모터 구동 시간이 결정된다. 그림 15와 같이 설정된 파라미터에 의해 이론적으로 계산하면 모터의 출력 응답은 그림 16과 같다.

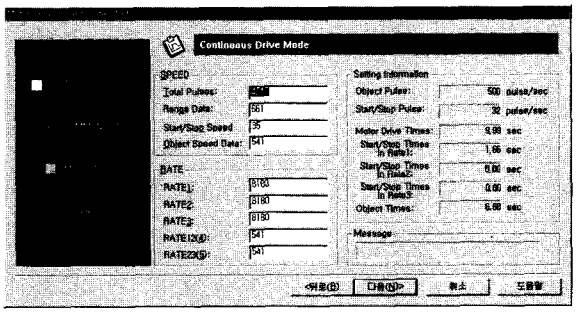


그림 15. 연속 구동 방식의 파라미터 설정 화면.

Fig. 15. Parameter set in continuous drive mode.

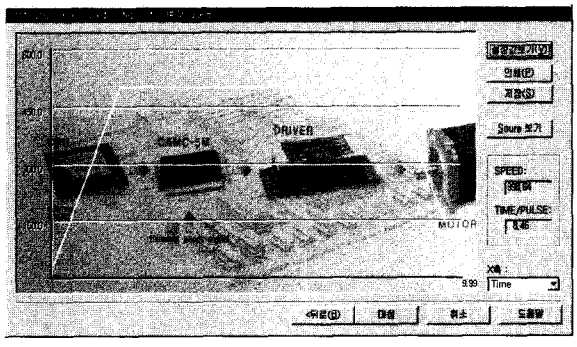


그림 16. 연속 구동 방식에서 이론치로 계산된 모터의 출력 응답.

Fig. 16. Computed output response in continuous drive mode.

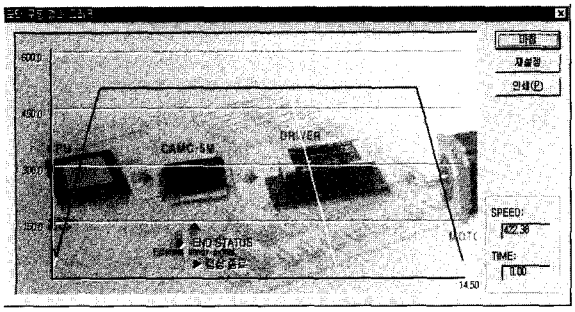


그림 17. 연속 구동 방식에서 모터의 실제 출력 응답.
Fig. 17. Measured output response in continuous drive mode.

연속 모드로 동작 시 "Slow Down Stop" 명령이 수신되면 "Start/Stop Speed" 데이터까지 감속한 후 모터는 정지한다. "Emergency Stop" 명령이 수신되면 모터는 즉시 정지한다.

그림 17과 같이 연속 모드로 동작할 때, 실제 모터의 출력 응답은 "Slow Down Stop" 제어 명령이 수신된 후 "Start/Stop Speed"까지 감속한 후에 정지한다

VI. 결론

본 논문에서는 산업용 로봇, CNC 선반, 반도체 장비 등 고속 정밀 제어 메카트로닉스 장비 등에 요구되는 정밀 모터 제어 칩을 설계하고 제작하였다. 전체 회로를 VHDL로 구현하였고, 논리 시뮬레이션을 먼저 실행하여 모든 동작을 검증할 수 있었다. 합성기는 Synopsys를 이용하였고 합성 후는 Compass로 시뮬레이션하여 그 결과가 논리 시뮬레이션과 같음을 확인하였다. 펄스 모터 제어 칩은 가속과 감속을 제어하고 펄스를 발생하기 위해서 많은 연산을 해야하므로 복잡한 구조로 인해 많은 팬 아웃 문제와 시간 지연요소가 따른다. 이를 줄이기 위해서 여러 입력을 비교하는 비교 제어기, 가속과 감속을 제어하는 가속·감속 제어기, 가속도 비를 선택하는 비율 선택기 그리고 펄스를 발생하는 펄스 발생기로 구분하여 설계하였다. 또한 모터의 속도 제어를 위해서 반드시 필요한 주파수 변조를 완전히 디지털 방식으로 구현하였으며, 모터의 속도를 제어할 때 매우 정밀한 속도로 동작하는 것을 실험을 통해 확인하였다. 칩의 성능을 측정하기 위해 성능 측정 하드웨어 보드와 칩 성능 측정 소프트웨어인 CAMC agent를 개발하였으며. 마이크로 스텝 모터 구동 시스템에 적용하여 개발된 칩의 성능을 실험을 통해 확인하였다.

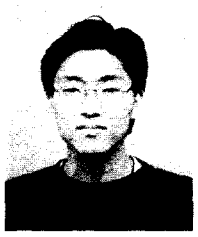
정밀 모터 제어 칩은 고속으로 동작하면서 모터의 속도를 부드럽게 가속·감속하는 것이 매우 중요하다. 개발된 모터제어 칩은 3 단계로 속도 가속·감속을 변화시켜 S자 드라이브를 구현하였으나, 앞으로는 완벽한 S자 드라이브 동작을 구현할 수 있는 연구가 있어야 할 것이다.

참고문헌

[1] A. C. Leenhouts, "Art and practice of stepmotor control", 1987.
[2] Y. Dote, "Servo motor and motion control : using digital signal processors", Prentice Hall, 1990.
[3] M. A. Bayoumi, *VLSI Design Methologies for Digital Signal Processing Architectures*, Kluwer Academic Publishers, Boston/Dordrecht /London, 1994.
[4] PCL-240AK, PCL-3AM "User's manual", (주)パルスモーター, 1995.
[5] "PCL-832 : 3-axis servo motor control card, command library and user's manual", Advantech, 1998.
[6] "General purpose motion control ICs : HCTL-1100 series technical data", Hewllett Packadd, 1996.
[7] "LM628/LM629 precision motion controller application note", National Semiconductor, 1997.
[8] "New high torque 5 phase geared step motor with compact driver", Oriental Motor U.S.A Corp., 1998.
[9] "Stepping motor controller PC card M20", Spindler & Hoyer GmbH, 1998.
[10] "2 axis, 2-D controller for stepper motor driven machines", TOMOMOTEK Product Information Sheet, 1998.
[11] "TEKCEL motion controller", Tekcel, 1997.
[12] "PC/104 module : AIM104-MOTION-1", Arcom.
[13] "PIC-SERVO motion control chipset", JR KERR.
[14] "EMC-XYYZ-Parallel port CNC stepper motor controller user's manual", Super-Tech, 1997.
[15] "0.8-Micron gate array libtary" , ETRI.
[16] "Synthesis shortcut user manual", Ver. 3.4b, Synopsys, 1997.
[17] "(V)DHL compiler reference manual", Ver. 3.4a, Synopsys, 1996.
[18] "Design compiler tutorial", Ver.3.4, Synopsys, 1996.
[19] VHDL for the ASIC Synthesizer User Guide, COMPASS Design Automation, 1994.
[20] J. R. Armstron, F. Gail Gray, *Structured Logic Design With VHDL*, Prentice Hall, New Jersey. 1993.
[21] N. H. E. Weste, Kamran Eshrghian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1993.
[22] Wayne Wolf, *Modern VLSI Design A Systems Approach*, PTR Prentice Hall, 1994.
[23] "XILINX foundation series V1.4", XILINX, 1998.
[24] "CAMC-5M user manual", (주)아진전자산업, 1999.

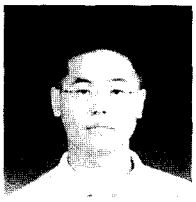
김 원 호

1985년 경북대 전자공학과(공학사). 1988년 동대학원 전자공학과(공학석사). 1999년 동대학원 전자공학과(공학박사). 1988년-1993년 전자통신연구소 연구원. 1993년-현재 동의공업대 전자과 조교수. 관심분야는 마이크로프로세서 응용, 정밀모터제어, ASIC 설계.



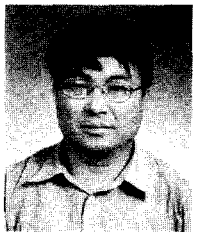
원 종 백

1997년 경북대 전자공학과(공학사). 1999년 경북대 전자공학과(공학석사). 1999년-현재 동대학원(박사과정). 관심분야는 디지털신호처리, VLSI 설계.



이 건 오

1999년 동서대 전자공학과(공학사). 1995년-현재 (주)아진전자산업 선임연구원. 1999년-현재 부산대 산업대학원(석사과정). 관심분야는 마이크로프로세서 응용, 정밀모터제어, ASIC 설계.



박 종 식

1976년 서울대 물리학과(이학사). 1978년 한국과학기술원 물리학과(이학석사). 1987년 미 Florida 대 전자공학과(공학박사). 1987년-현재 경북대 전자·전기공학부 교수. 관심분야는 VLSI 설계, 디지털신호처리, 계측 시스템.