Matlab/Simulink & dSpace 보드를 이용한 유도 전동기 벡터 제어 시스템의 구현

論 文
49B-10-6

The Implementation of the Field Oriented Control of Induction Motor Using Matlab/Simulink and dSpace board

金 培 善* · 韓 宇 勇** · 李 昌 求***
(Bae-Sun Kim · Woo-Yong Han · Chang-Goo Lee)

Abstract - An easier implementation method of the field oriented control of induction machine using Matlab/Simulink and dSpace board is proposed in this paper. The total system for the control of induction motor is modeled with the help of Matlab/Simulink. These models consist of induction motor block, SVPWM inverter block and control algorithm block, etc. And this system is simultaneously simulated and experimented in Matlab/Simulink environment with dSpace borad (DS1102). It is possible that Matlab and dSpace board compiler can make '*.c' and '*.obj' file of models designed in Matlab/Simulink environment automatically. Both Simualation and experimental results are given.

Key Words : Matlab/Simulink, Induction Motor, dSpace board (DS1102)

1. 서론

시뮬레이션은 시스템에 적용될 제어 알고리즘과 설계된 회로를 미리 분석하고 그 성능을 예측하기 위해서 수행된다. 시뮬레이션 도구에는 Matlab, ACSL 과 같은 시스템 시뮬레이터와 PSPICE, SABER 등과 같은 회로 시뮬레이터로 분류된다. 이러한 시뮬레이션 도구는 시뮬레이션 하려는 대상 시스템에 따라서 그에 부합하는 적절한 도구를 선택해야 한다.

시스템 시뮬레이터는 수학적인 모델로 표현되는 시스템을 모델링하기에 적합한 것으로써 전기기계와 제어시스템의 시뮬레이션에 편리하다. 반면에 회로 시뮬레이터는 소자 수준의 시스템, 즉 전력 전자 회로의 시뮬레이션에 적합하다.

한편 전력전자 회로와 전기기계, 제어시스템을 통합하여 시뮬레이션할 경우에는 회로 내부의 동작 특성뿐만 아니라 시스템의 응답특성도 관측할 수 있어야 하므로 앞에서 제시된 두 방식의 장점을 잘 살릴 수 있는 방법이 필요하다. 그런데 회로 시뮬레이터를 사용하게 되면 전기 기계의 모델링과 복잡한 제어 시스템을 구현하는 데 있어서 많은 어려움이 따른다. 이와는 달리 시스템 시뮬레이터는 회로의 세부적 동작보다는 제어 시스템에 초점을 둔 시뮬레이션에 주로 적용된다. 이때에는 전력 전자 회로의 세부적인 동작을 근사적으로 모델링하여야 한다. 그러므로 후자의 방법을 사용하는 것이 보다 현실적이라고 할 수 있다.

Matlab은 시스템 시뮬레이터로서 수학적인 모델링이 필요하나 Simulink 툴 박스를 사용하면 비선형 · 시변 시스템의 모델링이 매우 용이하다. 특히 코딩을 사용하지 않고서도 여러 종류의 제어 알고리즘을 쉽게 구현할 수 있고 설계된 블록들을 그룹화하거나, 마스킹 할 수 있다. 뿐만 아니라 설계시 포함된 미분 항을 계산하는 데 오차를 줄이기 위해서 여러 종류의 적분 알고리즘을 사용하여 분석할 수 있다. 따라서 이러한 장점으로 인하여 현재 Matlab/Simulink는 전력 시스템을 분석하거나 유도 전동기 알고리즘을 연구하는데 적용되고 있다[1-4].

문헌 [1],[2]에서는 Matlab/Simulink 환경 하에서 유도 전동기 제어 시스템을 시뮬레이션한 결과와 마이크로 프로세서를 이용하여 실험한 결과가 유사함을 보였다. 그리고 문헌 [3]에서는 유도 전동기의 벡터 제어를 위한 자속과 회전자 저항 추정 알고리즘 등을 Matlab/Simulink 환경 하에서 구현할 수 있는 방법을 제시하였다. 또한 문헌[4]에서는 SPWM으로 구동되는 인버터 시스템을 Matlab/Simulink로 구현할 수 있는 방법을 제시하였다[4]. 그러나 이들 방법들은 대부분 Matlab/Simulink 환경 하에서 설계된 유도 전동기 제어 시스템의 시뮬레이션만을 수행하고 있고 실제적인 구현 방법은 제시되어 있지 않다.

따라서 본 논문에서는 Matlab/Simulink 환경 하에서 유도 전동기 벡터 제어 시스템을 손쉽게 구현할 수 있는 방법을 제시하였다. 유도 전동기 벡터 제어를 위한 전체 시스템의 모델은 Simulink 툴 박스를 이용하여 설계되었고, 이를 dSpace 보드(DS1102)를 통하여 제어 실험을 수행하였다. 이 방법은 Matlab과 dSpace 보드 컴파일러가 설계된 모델의 '*.c'와 '*.obj' 파일들을 자동으로 생성시켜 주기 때문에 어떤 코딩(예: c 언어로의 변환)없이도 시스템의 시뮬레이션과 실행을 동시에 수행할 수 있다. 그리고 DS1102 보드를 이용한 하드웨어 구성 방법 및 구동 시스템의 구현 방법에 대해서 서술하였고, Matlab 환경 내에서 실시된 시뮬레이션 및 실험 결과를 제시함으로써 Matlab 환경하에서도 유도 전동기 구동에 대한 연구가 수행될 수 있음을 보인다.

* 準 會 員 : 韓國電子通信研究員
** 正 會 員 : 全州工大 電氣科 副教授
*** 正 會 員 : 全北大工大電磁情報工學部 副教授
接受日字 : 2000年 4月 21日
最終完了 : 2000年 10月 14日

$$\begin{bmatrix} \dot{i}_{ds} \\ \dot{i}_{qs} \\ \dot{i}_{dr} \\ \dot{i}_{qr} \end{bmatrix} = K \begin{bmatrix} R_s L_r & -\omega_r L_m^2 & -R_r L_m & -\omega_r L_m L_r \\ \omega_r L_m^2 & R_s L_r & \omega_r L_m L_r & -R_r L_m \\ -R_s L_r & \omega_r L_m L_s & R_r L_s & \omega_r L_s L_r \\ -\omega_r L_m L_r & -R_s L_m & -\omega_r L_s L_r & R_r L_s \end{bmatrix} \begin{bmatrix} i_{ds} \\ i_{qs} \\ i_{dr} \\ i_{qr} \end{bmatrix} + \begin{bmatrix} -L_r & 0 \\ 0 & -L_r \\ L_m & 0 \\ 0 & L_m \end{bmatrix} \begin{bmatrix} v_{ds} \\ v_{qs} \end{bmatrix} \quad (1)$$

여기서 $K = \dfrac{1}{L_m^2 - L_r L_s}$

2. 유도 전동기 모델링 및 시뮬레이션

2.1 유도 전동기 모델링

유도 전동기 전기적 상태 방정식은 d-q축 고정 좌표계에서 식 (1)과 같이 정의 할 수 있다.

v_{ds}, v_{qs} 고정자 d, q 전압 , i_{ds}, i_{qs} 고정자 d, q 전류

i_{dr}, i_{qr} 회전자 d, q 전류 , R_s, R_r 고정자, 회전자 저항

L_m 상호 인덕턴스 , L_s, L_r 고정자, 회전자 인덕턴스

ω_r 회전자 속도 , J 관성 모멘트 , T_L 부하 토크

그리고 유도 전동기의 기계적인 방정식에서 토크를 전류 식으로 표현하여 식 (2)와 같다.

$$\frac{d\omega_r}{dt} = \frac{3}{8}\frac{P^2 L_m}{J}(i_{qs} i_{dr} - i_{ds} i_{qr}) - \frac{P}{2}\frac{T_L}{J} \quad (2)$$

그림 1에 나타낸 것과 같이 식 (1)과 (2)를 Matlab function을 이용하여 고정 좌표계에서 유도 전동기를 모델링할 수 있다[1]. 그림 1에서 F1 - F4 는 유도 전동기의 전기적인 상태 방정식을 표현하고 있으며 F5 는 발생 토크, F6 은 전동기 속도를 나타낸다. 입력으로는 d축 전압과 q축 전압이고 출력으로는 d축 전류, q축 전류, 발생 토크, 속도 등이며 그 밖의 필요한 제어 알고리즘 상수와 유도 전동기 파라미터는 Matlab workspace를 통해서 참조하도록 초기 파일을 작성하여 실행시켰다.

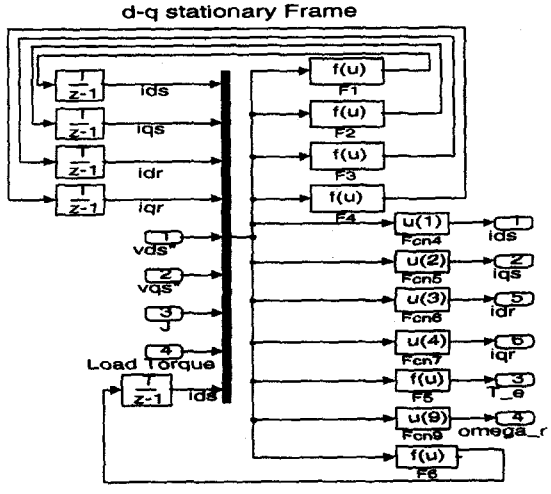

그림 1 고정 좌표계에서의 유도 전동기 모델링

Fig. 1 Induction machine model in stationary frame

F1=(Rs*Lr*u(1)-u(9)*(Lm^2)*u(2)-Rr*Lm*u(3)-u(9)*Lr*Lm*u(4)-Lr*u(5))/(Lm^2-Ls*Lr)

F2=(u(9)*(Lm^2)*u(1)+Rs*Lr*u(2)+u(9)*Lr*Lm*u(3)-Rr*Lm*u(4)-Lr*u(6))/(Lm^2-Ls*Lr)

F3=(-Rs*Lm*u(1)+u(9)*Lm*Ls*u(2)+Rr*Ls*u(3)+u(7)*(Lm^2)+u(9)*Lr*Ls*u(4)+Lm*u(5))/(Lm^2 -Ls*Lr)

F4=(-u(9)*Lm*Ls*u(1)-Rs*Lm*u(2)-u(9)*Lr*Ls*u(3)+Rr*Ls*u(4)+Lm*u(6))/(Lm^2 -Ls*Lr)

F5=3/4*P*Lm*(u(2)*u(3)-u(1)*u(4))

F6=3/8*P^2*Lm*(u(2)*u(3)-u(1)*u(4))/u(7)-u(8)*P/(2*u(7))

표 1 5[HP] 유도 전동기 파라미터

5 [HP], 220 [V], 60[Hz]			
Rs	1.8 [Ω]	J	0.3 [kgm²]
Rr	2.2 [Ω]	B	0.019[kgm²/s]
Ls	0.0557 [H]	Pole 수	4
Lr	0.0557 [H]	정격 속도	1735 [rpm]
Lm	0.0546 [H]	정격 전류	15 [A]

유도 전동기 파라미터는 표 1과 같다.

2.2 시뮬레이션

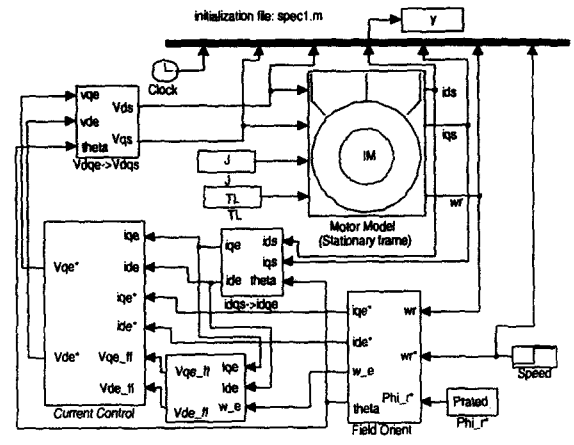

그림 2 시뮬레이션 모델

Fig. 2 simulation model

그림 2에 나타낸 것과 같이 유도 전동기 제어 시스템의 시뮬레이션 모델은 전류 제어부, 속도 제어부, 벡터 제어부, 유도 전동기 모델부로 크게 구성되어 있다. 여기서 인버터는 이득이 1인 모델로 가정하여 시뮬레이션 되었으며, 전류 제어부는 PI 제

어기로 구성되었다. 그리고 유도전동기 모델에 인가될 전압 명령은 PI 제어기의 출력과 역기전력을 보상해주는 전압의 합으로 구성하였고 또한, 속도 제어부도 PI 제어기로 구성하였다. 전류 제어기는 단순한 전류 모델의 주파수 응답을 통해 Cuf-off 주파수를 구하고 이를 통해 비례이득을 $\omega_c \sigma L_s$ 으로 적분 이득을 $R_s \omega_c$로 정하였고 속도 제어기 또한 단순한 속도 모델로부터 같은 방법에 의해 비례 이득을 $J\omega_c$, 적분 이득을 $\frac{J\omega_c}{5}$ 으로 하였다.

벡터 제어부는 회전하는 d-q 좌표에서 그림 3과 같이 슬립 속도를 이용한 간접 벡터 제어방식을 사용하였다. 간접 벡터 제어를 위해 우선, 회전 좌표계에서 회전자측 상태 방정식을 표현하면 식 (3),(4)와 같다.

$$\frac{d\lambda_{qr}^e}{dt} + R_r i_{qr}^e + \omega_{sl}\lambda_{dr}^e = 0 \tag{3}$$

$$\frac{d\lambda_{dr}^e}{dt} + R_r i_{dr}^e - \omega_{sl}\lambda_{qr}^e = 0 \tag{4}$$

여기서 $\omega_{sl} = \omega_e - \omega_r$

그리고 회전자 자속은 식 (5),(6)으로 표현되고

$$\lambda_{qr}^e = L_r i_{qr}^e + L_m i_{qs}^e \tag{5}$$

$$\lambda_{dr}^e = L_r i_{dr}^e + L_m i_{ds}^e \tag{6}$$

식 (3)~(6) 으로부터 식 (7),(8)을 얻을 수 있다.

$$\frac{\lambda_{qr}^e}{dt} + \frac{R_r}{L_r}\lambda_{qr}^e - \frac{L_m}{L_r}R_r i_{qs}^e + \omega_{sl}\lambda_{dr}^e = 0 \tag{7}$$

$$\frac{\lambda_{dr}^e}{dt} + \frac{R_r}{L_r}\lambda_{dr}^e - \frac{L_m}{L_r}R_r i_{ds}^e - \omega_{sl}\lambda_{qr}^e = 0 \tag{8}$$

회전자 자속이 일정하고 회전 좌표계의 d축과 회전자 자속축이 일치한다면 q축 자속이 0이 되므로 식 (9)와 같이 되고

$$\begin{cases} \lambda_{qr}^e = 0 \\ |\lambda_r^e| = \sqrt{\lambda_{dr}^{e\,2}} = |\lambda_{dr}^e| \end{cases} \tag{9}$$

식 (9)를 식 (7),(8)에 대입하여 정리하면 식 (10),(11)을 구할 수 있다.

$$\omega_{sl} = \frac{L_m}{\lambda_r^e}\left(\frac{R_r}{L_r}\right) i_{qs}^e \tag{10}$$

$$i_{ds}^e = \frac{\lambda_r^e}{L_m} \tag{11}$$

또한 토크 방정식을 회전자 자속과 고정자 전류로 나타내면 식 (12)와 같다.

$$T_e = \frac{3}{2}\left(\frac{P}{2}\right)\left(\frac{L_m}{L_r}\right)(i_{qs}^e \lambda_{dr}^e - i_{ds}^e \lambda_{qr}^e) \tag{12}$$

여기서, 토크 방정식에 식 (9)를 대입하여 정리하면 다음 수식을 얻는다.

$$i_{qs}^e = \frac{2}{3}\left(\frac{2}{P}\right)\frac{L_r}{L_m}\frac{T_e}{\lambda_r^e} \tag{13}$$

이상과 같이 식 (10),(11),(13)을 이용하여 그림 3과 같은 유도전동기 간접 벡터 제어 알고리즘을 모델링할 수 있다.

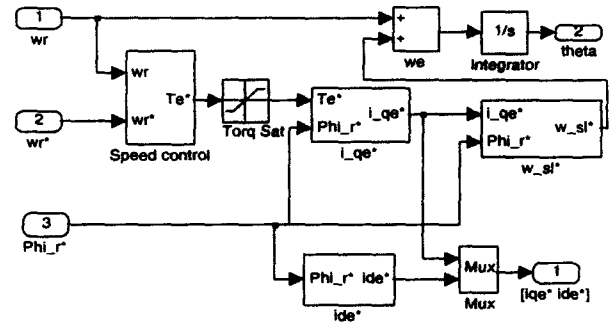

그림 3 간접 벡터 제어 알고리즘

Fig. 3 The algorithm of Indirect Field Oriented Control

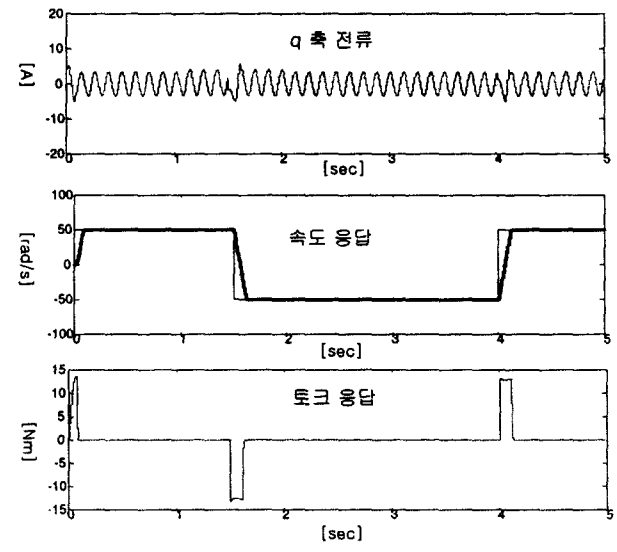

그림 4 무부하시 50[rad/s]에서 q축 전류,속도 및 토크 응답

Fig. 4 q axis current, speed and torque response with no load at 50[rad/s]

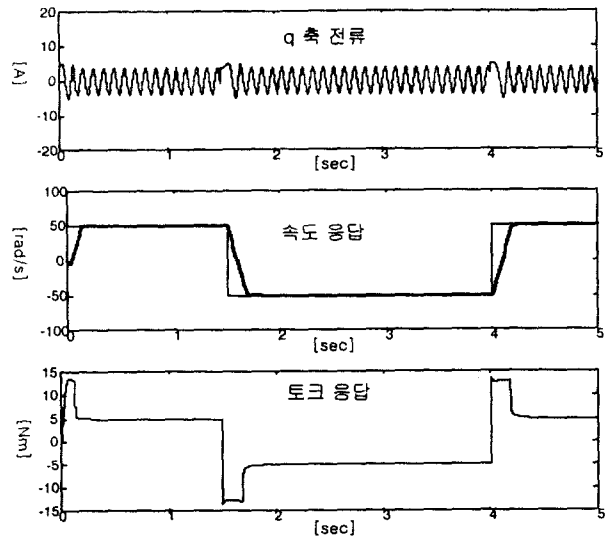

그림 5 부하 토크 5[N · m]인가시 50[rad/s]에서 q축 전류 ,속도 및 토크 응답

Fig. 5 q axis current, speed and torque response with 5[N · m] load at 50[rad/s]

그림 4와 5에서는 50[rad/s]에서 유도 전동기를 정 · 역으로 구동하였을 때 시뮬레이션 결과이다. 부하 토크를 5[N · m]인가 하였을 때 기준 속도 명령을 잘 추종하나 속도 응답이 약간 길어지는 것을 볼 수 있다.

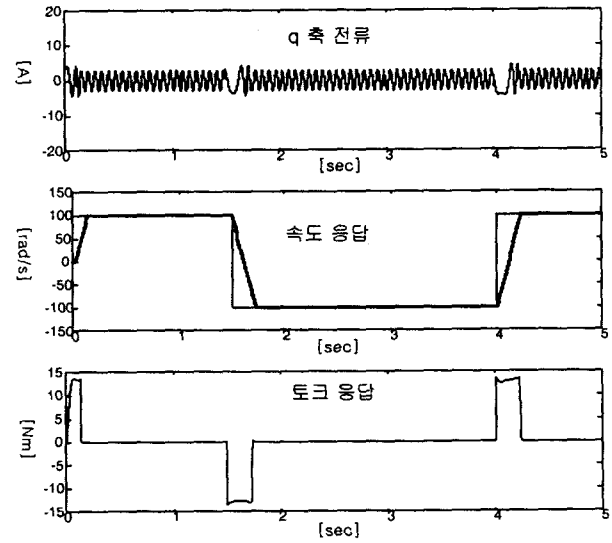

그림 6 무부하시 100[rad/s]에서 q축 전류, 속도및 토크 응답

Fig. 6 q axis current, speed and torque response with no load at 50[rad/s]

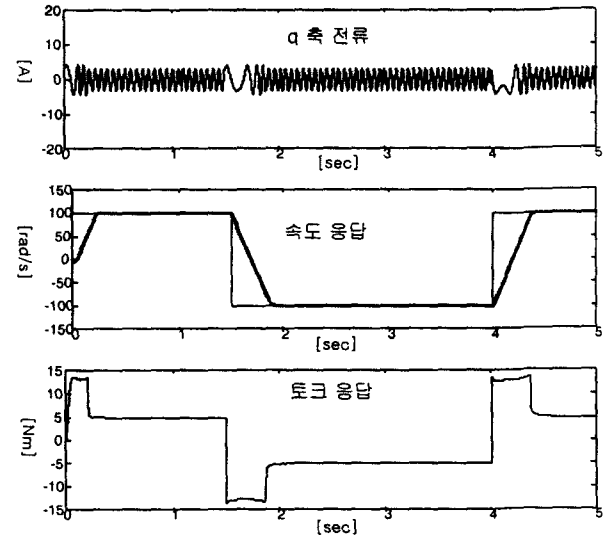

그림 7 부하 토크5[N · m]인가시 100[rad/s]에서 q축 전류 ,속도 및 토크 응답

Fig. 7 q axis current, speed and torque response with 5[N · m] load at 50[rad/s]

그림 6과 7에서는 100[rad/s]에서 유도 전동기를 정 · 역으로 구동하였을 때의 시뮬레이션 결과이다. 50[rad/s]에서의 시뮬레이션 결과와 거의 유사함을 볼 수 있다.

3. 전동기 구동 시스템 구현

3.1 하드웨어 구성

그림 8은 본 연구에서 유도 전동기 벡터 제어 실험을 위해 사용된 전체 시스템의 구성도를 나타내고 있다. 그림 5에서 DS1102 보드의 주 프로세서는 단일 사이클 명령 실행 시간이 33.33[ns]인 32비트 floating-point DSP [TMS320C31]이고 부 프로세서는 16비트 fixed-point DSP [TMS320P14]이다. 부 프로세서는 6개의 PWM 발생 회로와 4개의 타이머, 디지털 입 · 출력 포트, 4개의 캡처 입력과 시리얼 통신을 지원한다. 128 [KWord] 크기의 zero wait SRAM은 60[MHz] 클럭 속도로 동작하고 사용자가 작성한 프로그램을 다운로드 받아 실행시킨다. 그 밖의 주요 주변회로에 12비트의 800[KHz] 샘플링 A/D 2채널, 16비트 250[KHz] 샘플링 A/D 2채널이 있으며 12비트 D/A가 4 채널, 최소 120[ns] 엔코더 펄스폭을 24비트로 카운팅하는 증분형 인코더 블록이 2채널 있다. 본 논문에서는 주 프로세서에서 제어 알고리즘을 연산하고 부 프로세서는 PWM 신호의 발생을 담당하도록 구성하였다. 또한 12비트 2채널의 A/D 컨버터를 이용하여 전류 신호를 받아 들일 수 있도록 하였다. 그리고 본 논문에서는 이용되지 않았으나 4채널 D/A컨버터는 알고리즘 구동시 모니터링 채널로 활용할 수 있으며 2채널의 증분형 엔코더 블록 중 1채널로 속도 신호를 받았다. 이들 하드웨어 장치들은 dSpace사에서 제공하는 라이브러리에 의해서 Matlab/Simulink 환경 내에서 툴박스처럼 사용하여 외부 구동부와 인터페이스가 가능하다. PC와의 인터페이스는 내부 버스를 사용한다.

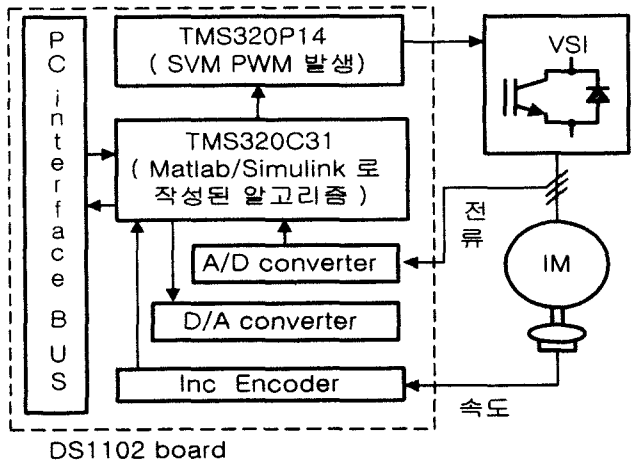

그림 8 실험에 사용된 시스템 구성도

Fig. 8 Configuration of the system used in experiment

데이터 전송의 불일치를 피하기 위해서 두 번의 16비트 PC-BUS 실행을 한 번의 32 비트 DSP-bus 전송으로 매핑시키는 변환기를 내장하고 있다[7].

3.2 구동 프로그램 구현

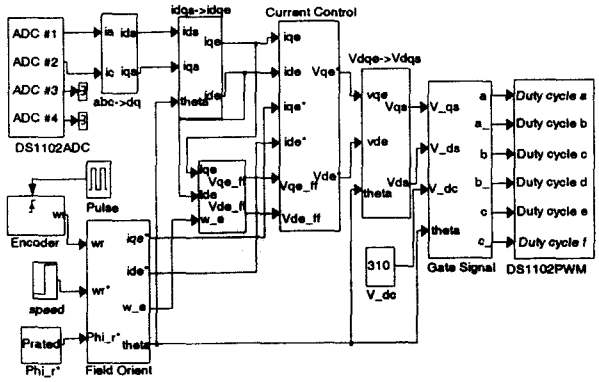

그림 9 DS1102보드에서 실행되는 구동 프로그램

Fig. 9 The driving program executing on DS1102 board

그림 9는 Matlab/Simulink 환경 하에서 작성한 유도 전동기 제어 시스템을 구동하는 프로그램이다. 이 구동 프로그램은 벡터 제어부, 속도 제어부, 전류 제어부, 공간벡터 PWM 발생부로 구성되어 있다. 공간 벡터 PWM은 S-function 함수를 사용하여 C 언어로 구현되었다. 이를 제외한 나머지 구동 프로그램은 Simulink 툴 박스를 이용하여 구현하였다. 구동 프로그램에서 외부 구동부 하드웨어와 인터페이스하는 데 사용되는 PWM 게이팅 신호를 출력하는 DS1102PWM, 전류 신호를 받는 DS1102ADC, 유도 전동기의 속도 신호를 받는 엔코더는 시뮬레이션 모델의 유도 전동기부와 유도 전동기 모델부에서 출력되는 전류 신호, 그리고 속도 신호를 대체한 것이다. Simulink 환경하에서는 구동 프로그램을 설계할 때 각기 독립적인 블록을 기능별로 그룹화하면 프로그램을 이해하기 편리할 뿐 아니라 수정하기 또한 편리하다. 이 구동 프로그램의 디버그는 모니터링 소프트웨어인 Control Desk를 이용하면 쉽게 할 수 있다. 왜냐하면 Control Desk는 구동 프로그램을 DS1102 보드에서 실행시키고 원하는 부분의 신호가 그림 14에서 처럼 실시간으로 확인되기 때문이다.

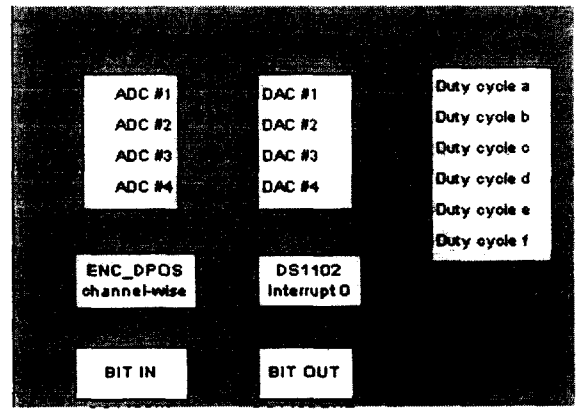

그림 10 DS1102 보드 라이브러리

Fig. 10 DS1102 board library

그림 10은 dSpace 사에서 제공하는 DS1102보드에 대한 RTI (Real Time Interface) 라이브러리의 일부분이고 이 라이브러리는 Simulink 툴 박스 처럼 사용할 수 있다. dSpace 사의 라이브러리 중에서 DS1102PWM을 이용하여 그림 11과 같이 PWM의 게이팅 신호를 간단한 게이트 로직을 통해 얻을 수 있다. 또한 Matlab/Simulink 환경에서 지원하는 S-function을 이용하여 공간 벡터 PWM 발생에 필요한 기준 전압 명령이 위치한 섹터를 판별하고 유효 벡터 인가 시간을 계산하여 섹터에 따른 게이팅 신호를 발생하도록 프로그램 하였다. 그림10의 DS1102PWM과 그림 11의 로직을 통해서 IPM 인버터의 구동 회로에 PWM 신호를 인가하였다.

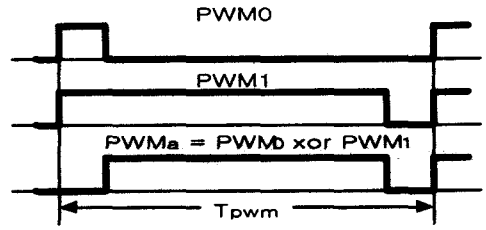

그림 11 xor 게이트로 만들어진 PWM 신호

Fig. 11 PWM signal made by xor gate

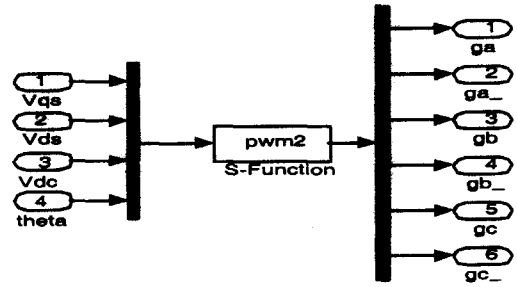

그림 12 S-function 으로 구현한 PWM

Fig. 12 PWM implemented by S-function

그림 12는 Matlab/Simulink 환경에서 S-function으로 구현된 공간 벡터 PWM 블록을 보여주고 있다.

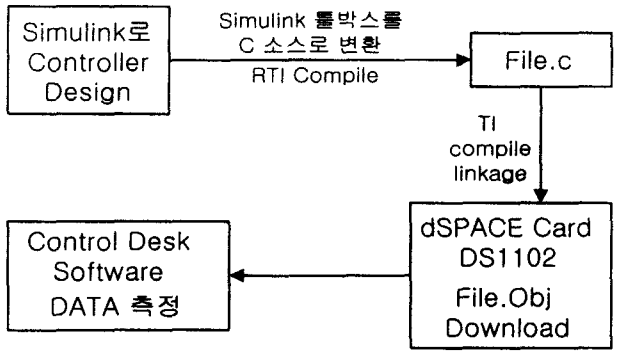

그림 13 프로그램 실행 흐름도

Fig. 13 The executing chart of program

그림 13의 프로그램 실행 흐름도는 Matlab/Simulink 환경에서 작성된 프로그램이 컴파일되고 DS1102 보드에 다운로드 되어 실행되는 흐름도를 나타낸다. 먼저 Matlab/Simulink 환경하에서 원하는 제어 알고리즘을 툴 박스를 사용하여 설계한 다음 Real Time Workshop에서 컴파일하면 배치 파일에 따라서 Simulink블럭을 '* .C' 소스로 다시 번역한다. 그런 후 TI 사의 컴파일러에 의해서 '* .obj'를 생성하고 이를 DS1102 보드에 다운로드하여 실행한다. 한 번 프로그램이 실행되면 소프트웨어 Control Desk 상에서만 정지및 재실행이 가능하다. 또한, 프로그램이 다시 컴파일 될 때까지 현재 다운로드된 프로그램을 유지하고 있어서 구동부의 손상을 예방하도록 되어 있다. 유도 전동기 제어 알고리즘이 아주 복잡할 경우와 Matlab에서 제공하는 정밀한 적분 알고리즘을 적용할 경우 유도 전동기 제어 알고리즘은 DS1102 보드에서 실행되지 못한다. 이는 주 프로세서의 연산 수행 능력이 충분히 빠르지 못할 뿐 아니라 호스트 PC 와도 데이터 통신을 해야하기 때문이다. 따라서 본 논문에서는 ode1의 적분 알고리즘을 사용하였고 유도 전동기 제어 시스템은 fixed step : 200[μsec]마다 수행되도록 하였다.

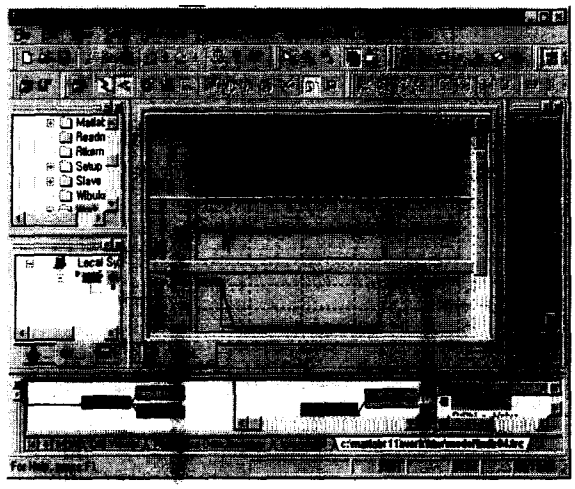

그림 14 Control Desk 소프트웨어

Fig. 14 Control Desk software

그림 14는 모니터링 소프트웨어인 Control Desk 이다. 그림 14의 중앙부에서 실험시 실시간으로 전류 및 속도, 전압 신호를 모니터링하고 있으므로 Matlab/Simulink 환경 하에서 작성된 알고리즘의 성능을 파악할 수 있다. 또한 모니터링하고 있는 신호를 실시간으로 저장할 수 있으며 알고리즘의 수행 및 정지 등 여러 기능을 제공하고 있다.

4. 실험 결과

실험은 소프트웨어로 Matlab/Simulink, 하드웨어로 DS1102 보드, IPM 인버터, 1024 펄스 엔코더, 5[HP] 유도 전동기, 그리고 부하로는 다이나모 메타를 사용하여 실시하였다. 그림 9와 같이 Matlab/Simulink 환경 하에서 구동 프로그램을 설계하여 실험을 실시하였다. 실험 결과는 Control Desk에서 실시간으로 데이터를 저장하였다.

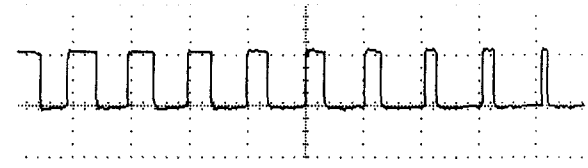

그림 15 200 [μsec] 주기의 PWM 게이팅 신호

Fig. 15 PWM gating signal at every 200 [μsec] period

그림 15에서는 200[μsec] 마다 Simulink에서 작성된 공간 벡터 PWM 블록에서 만들어지는 게이팅 신호를 보여주고 있다.

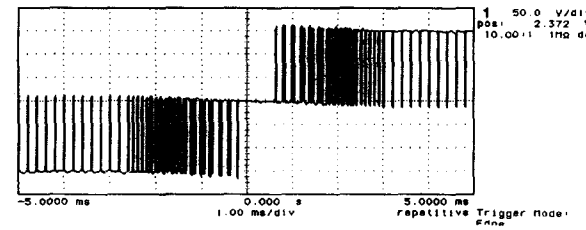

그림 16 선간 전압

Fig. 16 line to line Voltage

그림 16은 150[V]의 DC 링크 전압을 인가하여 그림 6의 프로그램으로 전동기를 구동할 때 발생한 선간 전압이다.

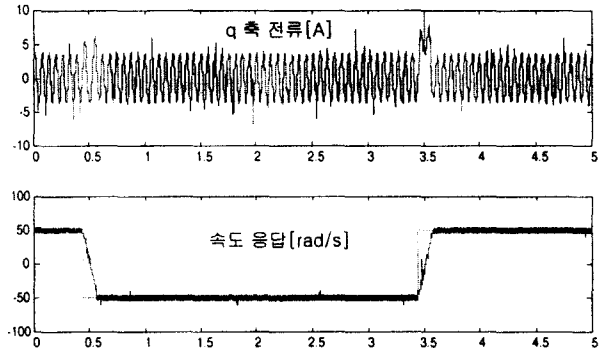

그림 17 무부하시 50[rad/s]에서 q 축 전류와 속도 응답

Fig. 17 q axis current and speed response with no load at 50[rad/s]

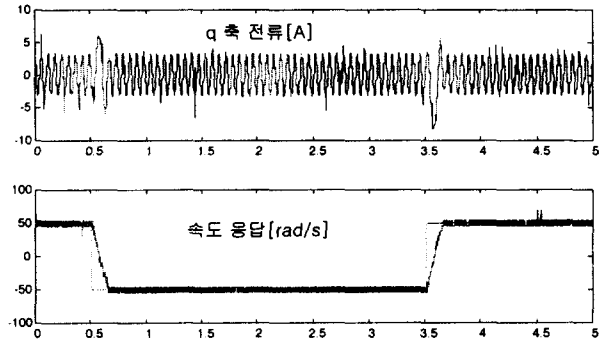

그림 18 50% 부하 토크 인가시 50[rad/s]에서 q 축 전류와 속도 응답

Fig. 18 q axis current and speed response with load at 50[rad/s]

그림 17과 18은 각각 50 [rad/s]에서 부하 토크를 인가하지 않았을 때와 50% 부하 토크를 인가하였을 때의 q축 전류와 속도 응답곡선이다. 추종 속도 곡선이 기준 속도 변화에 잘 추정하고 있음을 알 수 있다. 그림 4와 5의 시뮬레이션 결과와 비교해 볼 때 응답 곡선이 거의 유사함을 알 수 있다.

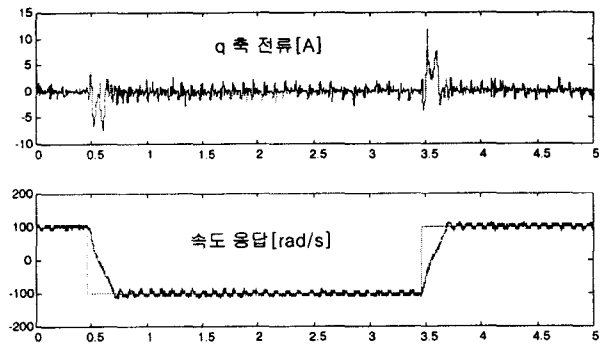

그림 19 무부하시 100[rad/s]에서 q 축 전류와 속도 응답

Fig. 19 q axis current and speed response with no load at 100[rad/s]

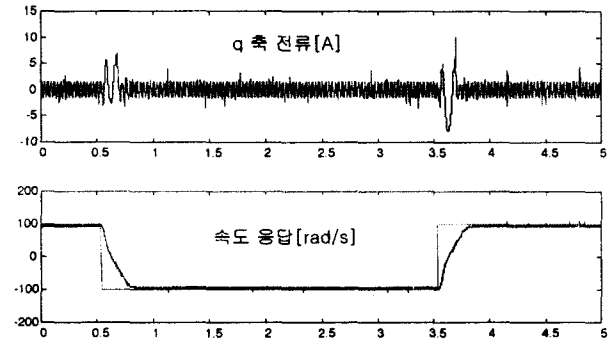

그림 20 50% 부하 토크 인가시 100[rad/s]에서 q축 전류와 속도 응답

Fig. 20 q axis current and speed response with load at 100[rad/s]

그림 19와 20은 각각 100 [rad/s]에서 부하 토크를 인가하지 않았을 때와 50% 부하 토크를 인가하였을 때의 q축 전류와 속도 응답 곡선이다. 부하 토크를 인가하지 않았을 때나 인가하였을 때도 기준 속도 변화에 잘 추종하고 있음을 알 수 있다. 실제 속도 응답 곡선이 그림 6과 7의 시뮬레이션 결과와 거의 유사함을 알 수 있다. 따라서 이상과 같은 실험 결과를 통하여 Matlab/Simulink와 DS1102 보드를 이용하면 코딩없이 제어 시스템의 시뮬레이션과 실험을 동시에 수행할 수 있음을 확인하였다.

5. 결론

본 논문에서는 Matlab/Simulink 환경 하에서 유도 전동기 제어 시스템을 모델링하고 이를 직접 DS1102 보드를 통하여 유도 전동기 제어 실험을 수행할 수 있는 방법을 제시하였다. 제시된 방법에 의하여 모델링된 시스템을 시뮬레이션하고 이를 실험한 결과 다음과 같은 장점이 있음을 확인하였다.

첫째, Simulink 툴 박스를 사용기 때문에 코딩을 할 필요가 없어 제어 알고리즘의 설계가 간편하다.

둘째, 시뮬레이션된 알고리즘이 Matlab/Simulink 환경 하에서 컴파일되어 실행되므로 알고리즘 성능을 즉시 확인할 수 있다.

셋째, 기존의 구현 과정, 즉 시뮬레이션으로 알고리즘을 검토한 후 다시 C 또는 어셈블리어로 프로그램하여 실험하는 과정이 생략되기 때문에 제어 알고리즘 개발에 필요한 시간이 많이 단축된다.

이런 관점에서 볼 때, 본 연구에서 제시한 방법을 이용하면 유도 전동기 제어 알고리즘의 연구 및 실험이 Matlab 환경에서 효과적으로 이루어질 수 있음을 입증하였다.

감 사 의 글

본 논문은 한국 과학 재단, 전라북도 지정 지역 협력 연구 센터인 메카트로닉스 연구 센터의 지원 하에 이루어졌습니다.

참 고 문 헌

[1] R. Teodorescu, "A Simulink Approach to Power Control Electronics Simulations", EPE'95, pp 3954-3958, 1995

[2] Liu Guohai, "The Simulation of a Motor-Inverter System with Matlab", IPEMC'97, pp 136-140, 1997

[3] Luis F. A. Pereira, "A Simulation Framework for Flux Estimation and Vector Control of Induction Machines", IEEE IECON'98, pp 1587-1591, 1998

[4] B. K. Lee, "A Simplified Functional Model for 3-Phase Voltage Source Inverter Using Switching Function Concept", IEEE IECON'99, pp 462-467, 1999

[5] B. K. Bose, "Power Electronics and AC Drivers", Prentice Hall, pp 264-276, 1986

[6] "TMS320C3x : User's Guide", Texas Instrument, 1994

[7] "Floating-Point Controller Board DS1102", dSpace GmbH, pp 1-48, 1995

[8] "controlDesk:Experiment Guide", pp 50-215, dSpace, 1999

[9] "Real-Time Interface(RTI and RTI-MP) : Implementation Guide", pp23-131, dSpace, 1999

저 자 소 개

김 배 선 (金 培 善)

1972년 6월생. 1998년 전북대 물리학과 졸업. 2000년 전북대 제어계측공학과 졸업(석사). 2000년 5월~현재 한국전자통신연구원(ETRI)

Tel : 016-606-1154, Fax : 063-270-2451

E-mail : xstar007@hanmail.net

한 우 용 (韓 宇 勇)

1964년 5월 23일생. 1986년 전북대 전기공학과 졸업. 1990년 동 대학원 전기공학과 졸업(석사). 1994년 동 대학원 전기공학과 졸업(공박). 현재 전주공업대학 전기과 부교수

Tel : 063-220-3834, Fax : 063-220-

E-mail : wyhan@jtc.ac.kr

이 창 구 (李 昌 求)

1958년 12월 25일생. 1981년 전북대 전기공학과 졸업. 1991년 동 대학원 전기공학과 졸업(공박). 1983년~1992년 한국전자통신연구소 선임연구원. 1996년 Alberta 대학 방문교수. 현재 전북대 공대 전자정보공학부 부교수.

Tel : 063-270-2476, Fax : 063-270-2451

E-mail : changgoo@moak.chonbuk.ac.kr