2005년도 대한전기학회 EMECS학회 추계학술대회 논문집 (2005.10.20-10.22)

DSP 내의 IQ math를 이용한 회전자 위치 추정 정밀도 향상에 관한 연구

*장 중 학, 이 광 호, 홍 선 기
호서 대학교 정보제어공학과 서보기기 및 제어연구실

A study about rotor position estimation enhance using IQ math in DSP

*Joong-Hack Jang, Kwang-Ho Lee, Sun-Ki Hong
Department of Information & Control Engineering, Servo machine and control Lab., Hose University

Abstract - DSPs used at motor control are usually fixed point processor. They need scaling because they cannot excute floating point calculation. Scaling for floating point calculation makes the DSP's speed down, complex coding and etc. Therefore the IQ math is adopted. IQ math makes the fixed point processor possible to calculate the floating point math. In addition, IQ math can reduce memory usage and be more faster than that without IQ math. It seems that IQ math is appropriate in motor position control. In comparison of the position calculation between the IQ math, math function and the sine table, the method using IQ math is superior than other methods.

1. 서 론

Motor을 구동시키기 위해서는 초기 회전자 위치 추정을 해야 한다. TI사에서 나온 DSP2000 시리즈는 fixed point processor이다. DSP 28xx는 고성능을 구현할 수 있도록 CMOS 반도체 공정으로 설계 제작 되었고 내부 코어 전원이 1.9V로 저전원 칩이다. 모터를 구동시키기 위한 QEP, CAP, ADC등 여러 가지 주변회로가 내장 되어 있고 150Mhz라는 고속에서 동작한다는 장점들을 지니고 있다 [4]. 하지만 Fixed point processor는 소수점 계산 시 불리한 점을 가지고 있다. 회전자 위치 추정 시 360°이라는 각도를 나누어 보다 정확한 위치를 알아야 하기 때문에 소수점 처리는 모터 소수점 자리를 계산하기 위해 필요하다. 따라서 우리가 흔히 써오던 방법은 각각의 각도에 일정한 스케일을 한 후 그 계산된 값을 테이블 표로 만들어 그것을 프로그램 상에 저장한 후 그 테이블을 불러다 쓰는 방법 또는 10진수의 승수 형태로 스케일을 해서 소수점 이하 자리수를 없애는 방법 등을 주로 이용해 왔다. 여기서 우리가 제시할 또 하나의 방법은 바로 IQ math이다. TI사에서 라이브러리 형식으로 제공되는 계산 방법이다 [1]. 이 방법을 통해 비록 코드의 라인수가 길어지는 상황이 생길 수는 있으나 기존의 방법에 비해 메모리를 차지하는 비중을 줄이는 동시에 속도 향상의 효과도 볼 수 있을 것이라 예상 된다.

2. 본 론

2.1 회전자 위치 측정

AC 서보 시스템에서 회전자의 위치 측정은 아주 중요하다. 회전자의 위치 측정이 잘못 된다면 정확한 벡터제어를 수행 할 수가 없다. 회전자의 위치 측정은 엔코더의 U, V, W 상의 신호를 이용하여 초기 위치를 얻어오고 그 이후에는 증가형 엔코더를 사용하여 회전자의 위치를 측정하게 된다. 여기서 회전자의 위치라는 말은 자극의 위치를 말하는 것이다. 그림 1은 회전자의 위치에 따른 U, V, W 상의 신호를 전기각 360도 간격으로 나타내었다. 이 신호는 회전자의 초기 위치를 얻기 위해 쓰였고 전동기 구동 이후에는 사용하지 않았다.

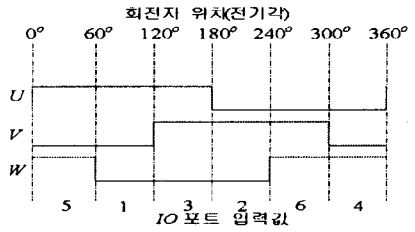

그림 1. 엔코더의 위치 신호 위치 신호

그림에서 나오는 바와 같이 60° 간격으로 초기 위치를 얻어온 후 회전자의 위치를 그 중간 값으로 취한 후 설정된 회전자의 초기 위치로 전동기를 구동하면서 IO포트의 값을 체크 한다. 60°의 간격으로 위치 정보신호가 바뀌는 점에서 회전자의 정확한 위치를 안 다음 한 바퀴가 지날 때 마다 오차를 보정한다. 그 후 엔코더 신호를 이용하여 전기각에 상응하는 sin 테이블과 연산을 통해 테이블의 값을 불러와 정확한 회전자의 위치를 파악한다. 2500 펄스의 엔코더의 경우 2/5를 곱해 거기에 상응하는 1000개의 값을 불러 와 회전자의 위치를 파악한다. 이러한 방식을 사용하면 측정한 전기각의 오차를 줄일 수 있다. 이 때 스케일 된 값을 늘린다면 더 정확한 회전자의 위치를 측정 할 수 있게 되는 것이다 [2].

2.2 IQ math

IQ math는 fixed point processor에서 소수점의 계산을 하기 위해 TI 사에서 제공되는 것이다. Fixed point processor를 이용하여 고속 그리고 고정밀도를 유지하면서 연산을 하기 위해서 사용하는 계산방식을 말한다.
다시 말해서 어떠한 숫자를 2진수의 스케일을 해주는 라이브러리 파일을 통해서 프로세서의 하는 일을 덜어주어 메모리 용량을 줄이고 계산 속도를 보다 빠르게 해주는 것이다. 기본 적인 개념은 다음 그림 2와 같다.

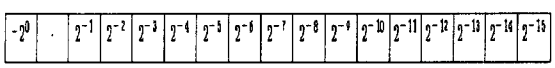

그림 2. 스케일의 방법

위와 같은 방법으로 임의의 수에 따라 그에 해당하는 2진수 스케일 값을 넣어 숫자를 분리 하게 된다. 위의 경우는 16bit의 데이터인 경우를 나타내는 것이고 32bit의 경우에는 좀 더 넓은 범위의 분할이 가능하다.

```
Q15: x.xxx xxxx xxxx xxxx -1 _ float _ 1
Q14: xx.xx xxxx xxxx xxxx -2 _ float _ 2
Q13: xxx.x xxxx xxxx xxxx -4 _ float _ 4
                .
                .
                .
Q2 : xxxx xxxx xxxx xx.xx  -8192 _ float _ 8192
Q1 : xxxx xxxx xxxx xxx.x -16384 _ float _ 16384
Q0 : xxxx xxxx xxxx xxxx  -32768 _ float _ 32768
```

그림 3. 스케일 범위 지정

위의 그림이 IQ math의 원리를 가장 잘 나타내 주고 있다. 일정한 간격의 수를 분리해서 나타내는 것이다. 얼핏 보면 10진수 스케일와 같아 보일 수 있지만 2진수를 사용한다는 의미에서 많은 성능의 향상을 가져 올 수 있다.

단순하게 생각해서 1에서 -1의 수를 약 65000개로 나누게 되는데 이것이 바로 fixed point processor를 floating point processor로 사용하는 것이다. 소수점 아래의 숫자를 2진수의 곱으로 해서 소수점 이하의 숫자를 좀 더 자세하게 나타 낼 수 있다. 위의 방법 역시 16 bit의 경우를 나타내는 것이고 32bit라면 보다 많은 수로 표현 할 수 있다. 다음의 식(1)을 이용해서 좀 더 정확한 이해를 할 수 있다.

$$QM : x \le float \le y$$

$$float \Rightarrow float \times 2^{M} \Rightarrow N(QM) \qquad (1)$$

예를 들어 소수점 숫자를 스케일 할 경우

$10.30 \rightarrow 10.30 \times 2\text{^}11 \rightarrow 21094$

$0.41 \rightarrow 0.41 \times 2048 \rightarrow 839$

$0.23 \rightarrow 0.23 \times 2048 \rightarrow 471$

이러한 형식으로 스케일이 되며 여기서의 계산은 아래의 식(2)와 같이 된다.

$$x(QN) \times y(QN) + z(QN) \qquad (2)$$

Q11로 스케일을 한 수를 계산하는 경우

$21094(Q11) \times 839(Q11) + 471(Q11)$

$21094(Q11) \times 839(Q11) = 17697866(Q22)$

$\rightarrow 17697866 / 2048 \rightarrow 8641(Q11)$

$471(Q11) + 8641(Q11) = 9112(Q11)$

이러한 방식을 이용하여 계산을 하는 것이다. 기본적인 연산에서도 알 수 있듯이 2진수에 의한 스케일이 굉장한 고분해능 연산을 수행 할 수 있도록 해준다는 사실을 알 수 있다.

2.3 일반 연산에서 IQ math로의 변환

Fixed point processor에서 분해능을 올리기 위해서 가장 많이 이용되는 방법이 바로 테이블을 사용하는 방법이다. Motor의 회전자 초기 위치 추정 시 임의의 위치를 가정하고 거기에 따른 sin값을 불러 오게 된다. 이 방법으로 비교적 정확한 초기 위치를 추정할 수 있게 된다.

하지만 이 table이 차지하는 용량은 숫자 하나에 1 word씩하여 1000배의 스케일로 잡는다 치면 1 Kword의 용량을 차지하게 된다. TMS320F2812 같은 경우 플래시 메모리 용량이 128K Word라고는 하나 개발 과정에서는 플래시 메모리를 거의 사용하지 않고 플래시 메모리 영역은 download시 많은 시간을 소모하기 때문에 권장하지 않는 방법이다. 때문에 대부분의 경우 RAM으로 download를 하는데 이러한 요건으로 따져 볼 때 1K Word는 프로그램 상에서 너무나 많은 용량을 차지한다. 다음에 나오는 table은 연구실에서 쓰는 sin-table의 일부이다.

```
int SIN_TBL
[]= { 0,    6,   12,   19,   25,   32,   38,   45,   51,   57,  //10
     64,   70,   77,   83,   89,   96,  102,  109,  115,  121,  //20
    128,  134,  141,  147,  153,  160,  166,  172,  179,  185,  //30
    191,  198,  204,  210,  217,  223,  229,  235,  242,  248,  //40
    254,  260,  267,  273,  279,  285,  291,  298,  304,  310,  //50
                                .
                                .
                                .
    316,  310,  304,  298,  291,  285,  279,  273,  267,  260,  //460
    254,  248,  242,  235,  229,  223,  217,  210,  204,  198,  //470
    191,  185,  179,  172,  166,  160,  153,  147,  141,  134,  //480
    128,  121,  115,  109,  102,   96,   89,   83,   77,   70,  //490
     64,   57,   51,   45,   38,   32,   25,   19,   12,    6,  //500
  //100 TO 270°
```

그림 4 sin 테이블 일부분

이렇게 긴 문장이 되어 차지하는 메모리의 양이 1K Word가 된다. 하지만 IQ math를 이용하면 메모리의 양이 줄어들게 된다. 지금 쓰고 있는 sin-table은 개발자가 직접 계산을 한 후 그 값을 입력해야 한다. 사용상의 번거로움뿐만이 아니라 DSP자체도 연산을 통해 값을 얻는 것이기 때문에 당연히 속도 지연이 될 수밖에 없다. IQ math를 이용하는 방법은 다음과 같다.

표 1. IQ math Library & trig functions

operation	Floationg - Pint	"IQmath" in C
type	float A, B	_iq A, B;
constant	A = 1.2345	A = _IQ(1.2345)
multiply	A * B	_IQmpy(A,B)
divide	A / B	_IQdiv(A,B)
add	A + B	A + B
subtraction	A - B	A - B
boolean	>, >=>, <, <=, ==, !=, &&	>, >=>, <, <=, ==< !=, &&
trig functions	sin(A), cos(A)	_IQsin(A),_IQcos(A)
	sin(A,2pi)	_IQsinPU(A)
	cos(A*2pi)	_IQcosPU(A)
	atan(a)	_IQatan(A)
	atan2(A,B)	_IQatan2(A,B)
	atan2(A,B)/2pi	_IQatan2PU(A,B)
	sqrt(A),1/sqrt(A)	_IQsqrt(A),_IQisqrt(A)
	sqrt(A*A+B*B)	_IQmag(A,B)
saturation	if(A>Pos)A= Pos if(A<Neg)A=Neg	_IQsat(A,Pos,Neg)
iq to iqN	A	_IQtoIQN(A)
iqN to iq	A	_IQNtoIQ(A)
integer(iq)	(long)A	_IQint(A)
fraction(iq)	A-(long)A	_IQfrac(A)
iq*iq=long	A*(floast)B	_IQmpyI32(A,B)
integer(iq*long)	(long)(A*(float)B)	_IQmpyI32int(A,B)
fraction(iq*long)	A-(long)(A*(float)B)	_IQmpyI32frac(A,B)
qN to iq	A	_QNtoIQ(A)
iq to qN	A	_IQtoQN(A)
string to iq	atof(char)	_atoIQ(A)
IQ to float	A	_IQtoF(A)

위의 비교를 통해 새로운 코드를 만들 수 있다. 뿐만이 아니

라 속도에서도 차이가 많이 나게 된다. 위의 표에서 알 수 있듯이 변환 하는 방법도 그리 어렵게 느껴지지 않는다.

2.4 일반 연산과 IQ math의 계산 결과 비교

일반적인 연산과 IQ math를 이용한 연산과의 비교를 통해 IQ math가 얼마나 적은 용량을 차지하는지 그 속도 차이는 어떠한지 알아 봐야 한다. 일단은 0°부터 90° 사이의 sin파를 만들어 비교하고 그 속도의 차이와 코드의 크기 그 값에서 차이는 얼마나 나는지 비교를 해본다.

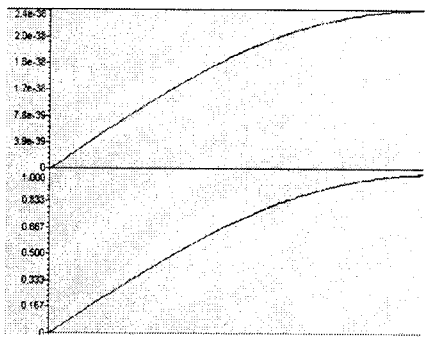

그림 5. IQ math와 일반연산 비교 그래프

표 2. 일반연산시 sin 테이블 값

[9]	0.1564345	[50]	0.7660444
[10]	0.1736482	[51]	0.7771459
[11]	0.190809	[52]	0.7880107
[12]	0.2079117	[53]	0.7986355
[13]	0.224951	[54]	0.8090169
.			
.			
.			
[43]	0.6819983	[84]	0.9945219
[44]	0.6946583	[85]	0.9961946
[45]	0.7071068	[86]	0.9975641
[46]	0.7193398	[87]	0.9986295
[47]	0.7313537	[88]	0.9993908
[48]	0.7431448	[89]	0.9998477

표 3. IQ math 연산시 sin 테이블 값

[9]	2624528	[51]	13038324
[10]	2913326	[52]	13220605
[11]	3201236	[53]	13398859
[12]	3488171	[54]	13573031
[13]	3774044	[55]	13743070
.			
.			
.			
[43]	11442013	[84]	16685302
[44]	11654412	[85]	16713368
[45]	11863262	[86]	16736343
[46]	12068498	[87]	16754220
[47]	12270057	[88]	16766993
[48]	12467880	[89]	16774659

위의 비교에서는 같은 계산을 하고 있다. 연산 시에 들어가는 값이 틀린 것은 IQ math를 이용한 그래프와 표에서 다시 역변환이 들어가지 않았기 때문이다. 위의 계산을 다한 후 code composer에서 프로 파일러 기능을 [5] 이용해 코드의 크기 및 연산을 하는 동안 소비하는 총 수행 사이클을 비교해 보았다. 두 그래프를 같은 평면에 그리지 못하는 이유는 IQ math를 사용해서 얻은 결과의 수가 일반 연산에 의한 수에 비해 크게 나오기 때문이다. IQ math후 다시 원래대로 스케일을 돌리면 그래프가 일치하기 때문에 그래프 자체에 대한 비교가 어렵게 된다.

표 4. 일반 연산과 IQ math 성능 비교

	일반연산	IQ math 연산
코드크기	26	23
총 수행 사이클	214294	8743

그냥 보기에도 매우 큰 차이의 연산 소비 수행 사이클이다. 수행 사이클은 150MIPS의 속도 동안 소비한 사이클 수를 나타낸 것이다. 뿐만 아니라 계산해야 하는 양이 늘어날 경우 더 큰 속도의 차이를 보이게 된다. 보다 정밀하게 회전자의 초기 위치 추정을 원하는 단계에서는 보다 많은 계산이 필요하게 될 것이고 그렇게 되면 더욱더 IQ math의 필요성이 대두 되게 될 것이다.

3. 결 론

이러한 방법을 통해 IQ math를 이용해 AC servo motor 구동시 비교적 정확한 rotor의 초기 위치 파악이 가능해 질 뿐 아니라 성능의 향상에 비해 적은 용량과 빠른 성능을 기대 할 수 있게 될 것이다. 기존의 일반 연산 방법보다 많은 장점을 가지고 있는 IQ math는 앞으로 여러 방면의 수학적 연산 및 fixed point processor를 floating point processor로서 사용하게 되는 큰 계기가 될 것이다. Floating point processor에 비하여 가격이 저렴한 fixed point processor로 floating point processor의 부동 소수점 계산 속도를 낼 수 있다는 것은 가격의 하락을 가져 올 것으로 기대한다. 차후에 2812의 상용화 버전인 2808이나 2810의 모델들을 사용할 때에도 적은 플래시 메모리 용량 때문에 sin 테이블을 삽입할 수가 없는 경우 IQ math를 이용하면 좀 더 효과 적인 메모리 관리가 될 것이다.

[참 고 문 헌]

[1] TEXAS INSTRUMENT " IQ math Library(A Virtual Floating Point Engine: C28X Foundation Software)".

[2] 최 치영, 홍 선기, 김 수길 " 고성능 DSP 기반의 FA 용 AC 서보 시스템에 관한 연구" 조명 학회 논문지, 2003

[3] 설 승 기 "전기기기 제어론" , 도서출판 브레인 코리아, 2002.

[4] "TMS320f28X cpu 핸드북" 도서출판 싱크윅스, 2005.

[5 TEXAS INSTRUMENT "TMS320C2XX/24X Code Composer User's Guide".