UIT Report Vol.13, No.2, pp.351~358, 1982 연구논문집 제13권 제2호 pp.351~358, 1982

프로그래머블 디지탈 필터의 설계 및 제작*

박경섭, 김수운, 김종수
전기 및 전자공학과
(1982.6.30 접수)

〈요 약〉

디지탈필터의 계수 및 종류에 무관한 고속 디지탈 필터를 처리하기 위한 programmable processor를 설계하고, 이의 효율을 산출하였다.

μP를 사용하므로 필터변동에 따라 프로그램으로 처리하여 flexibility를 유지하면서 TTL logic으로 프로세서 처리속도를 실시간에서 가능하도록 하였다.

Sampling frequency, $f_s=1/(\text{number of instruction})*(\text{cycle time})$으로 cycle time에 반비례 한다. 구성된 회로의 cycle time은 210ns로, IC로제작되면 더욱 단축가능하고 일반 μP와 직접연결이 가능하다. 제안된 프로세서는 기존회로보다 2배의 증가된 효률를 얻었고, 간단한 회로구성이 가능하다.

Design and Implementation of Programmable Digital Filter

Park, Kyung Sub · Kim, Su Woon · Kim, Jong Soo
Dept. of
Electrical and Electronic Engineering
(Recieved June 30, 1982)

〈**Abstract**〉

The digital signal processing requires high speed computation capability in real time processing. Many dedicated hardware processors are based on pre-calculated coefficients or its unique architecture.

This paper describes the architecture of programmable digital filter with a distributed configuration using μP 8086.

The processor accepts the microprogram from the μP to execute the algorithm. The sequence of microprogram is executed sequentially in order to simplify the processor circuits, and multiplication is replaced by the series of addition, substraction and shift operations simultaneously.

The experimental results offer better efficiency than other multiplier in practical bit size.

This structure provides a good flexibility through, uP and high speed computation with TTL logic family.

The cycle time is 210 ns so multichannel processing could be implemented depending on the length of program.

* 본논문은 1981년도 문교부 학술연구비에 의하여 연구된 것임.

Ⅰ. 서 론

Digital filter를 구성하는데 해결 할 문제는 속도 개선이다[1,2].

근래에 반도체 집적기술의 발달로 소형 μP의 등장과 더불어 이를 이용한 filter 구성에 대한 연구가 활성화되고 있으나, μP자체의 속도제한 때문에 real time 처리가 어렵지만, flexibility 및 회로의 간소화 등의 장점을 갖고 있다[1,3].

따라서 flexibility를 유지하면서, 처리속도를 개선시키는 algorithms과 architecture 등이 요구되고, 이에 parallel processing 등의 구조를 갖는 processor가 있으나 computation loads에 따른 효율이 불균일하고 설계하기가 어렵다[7].

한편 계수를 이용한 programmable architecture의 경우, filter 변화에 따른 계수변동과 address 지터 등으로 인해 속도가 단축되며, flexibility가 제한된다.

본 논문에서는 flexibility와 real time processing의 두가지 장점을 유지하기 위하여, 16bits μP 8086을 이용하여 분산처리형태의 programmable digital filter의 구성에 대하여 연구하였다. 본 논문에서 구성한 digital filter의 특징은 다음과 같다.

① 임의의 차수를 갖는 filter를 μP에 의하여 처리할 형태의 microprogram으로 변환시킨다.

② 처리속도 단축을 위하여 TTL 회로로 구성된 arithmetic unit 및 scaler는 microprogram에 의하여 제어된다.

③ filter의 종류에 독립적인 회로를 구성한다.

Ⅱ. 연산알고리즘

Digital filter를 구성하기 위한 architecture를 살펴보기 전에 간단히 이에 필요한 algorithms을 표시하면,

the Infinite Impulse Response(IIR) Filter의 경우

$$y(t_n)=\sum_{K=1}^{N}\alpha_K y(t_{n-K})+\sum_{K=0}^{R}\beta_K x(t_{n-K}) \quad (1)$$

$t_n=nT$, $n=0,1,2\cdots\cdots$

T: sample period

α_K, β_K: filter coefficient

$x(t_n)$: sample signal

the Finite Impulse(FIR) Filter의 경우

$$y(t_n)=\sum_{K=0}^{N}h(t_K)x(t_{n-K}) \quad (2)$$

$x(t_n)$을 discrete Fourier Transform으로 변형하면

$$x(f_j)=\sum_{n=0}^{N-1}x(t_n)\exp(-i2\pi f_j t_n) \quad (3)$$

$i=\sqrt{-1}$. $j=0,1,2,\cdots N-1$. $f_j=j/NT$

(3)식을 FFT로 정리하면

$$\mathrm{Ar}=\sum_{K=0}^{N-1}x_K\exp(2\pi irk/N)=\sum_{K=0}^{N-1}x_K w^{rK} \quad (4)$$

$w=\exp(2\pi i/N)$

N: number of sample

r: $0,1,2,\cdots,N-1$

Time domain과 frequency domain에서 FFT를 비교하면 sample N가 클 경우, N^2의 계산을 FFT 방법으로 감소시킬 수가 있다[7]. 결국 어느 domain에서든 filter를 수행하는데 필요한 연산은 가감산 및 곱셈으로 처리되며,

$$y_{1(n)}=\alpha_1 y_{1(n-1)}+\alpha_2 y_{1(n-2)}+\beta_0 x_{1(n)}+\beta_1 x_{1(n-1)} \quad (5)$$

의 IIR의 경우,

Discrete value x_n, y_n은 digital value로 표시가능하며, normalize한 경우

$$x_n=\sum_{K=1}^{L}x_{nK}2^{-K} \qquad \begin{pmatrix} x_{nK}: 0,1 \\ L: \text{bit length}\end{pmatrix} \quad (6)$$

(6)식을 사용하여 (5)식을 정리하면

$$y_{1(n)}=\sum_{K=1}^{L}2^{-K}(\alpha_1 y_{1(n-1)K}+\alpha_2 y_{1(n-2)K}+\beta_0 x_{1(n)K}+\beta_1 x_{1(n-1)K}) \quad (7)$$

$$=\sum_{K=1}^{L}2^{-K}a_K(y_{1(n-1)K}+y_{1(n-2)K}+x_{1(n)K}+x_{1(n-1)K}) \quad (8)$$

결국 곱셈된 수를 미리 계산하여 memory에 상수값을 저장, $y_{1(n-1)K}$, $y_{1(n-2)K}$, $x_{1(n)K}$, $x_{1(n-1)K}$를 해당 address로 하여 각각 다른 a_K를 처리하므로 multiplier없이 신속히 처리할 수 있으나, 서론에서 언급한 바와 같이 flexibility가 저하된다[13].

상수 및 변수의 곱셈은 가감산 및 shift로 처리 가능하므로, $X=Z+[C*Y]$에서 C가 상수이면 가감산으로 대행된다. 즉 $C=2.02642$인 경우

$C=2^1+2^{-5}+2^{-8}-2^{-10},\cdots$의 형태로 표시되므로

$$X=Z+[(2^1+2^{-5}-2^{-8}-2^{-10})*Y] \quad (9)$$

가 되어 변수 Y의 shift 및 가감산이 된다.

상수 C의 변환 algorithm은 다음과 같다[3].

1. V의 초기치를 0으로 set시킨다.

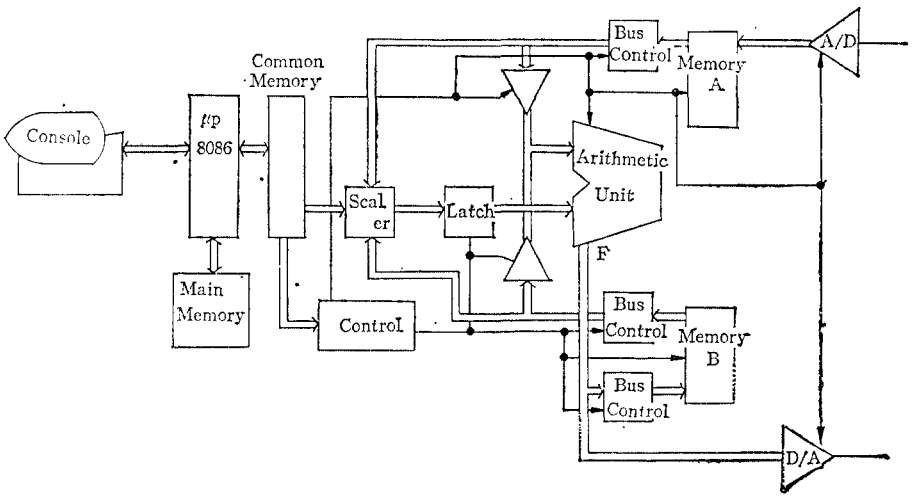


그림 1. 전체회로의 Block Diagram

2. 오차 $E=C-V$ 를 구하여 허용오차내에 존재하면 loop에서 끝난다.

3. $|T-E|$=최소값이 되는 수 T를 구하여 $V=V+T$를 결정, set시킨다.

4. 새로운 값 V로 step 2부터 반복한다. 변수와의 곱셈은 2's complement를 이용하여 계산하는 것이 회로 구성상 편리하다. X를 2's complement로 표시하면

$$X=-x^0+\sum_{i=1}^{B-1}x_i2^{-i} \qquad (10)$$

x_i: weighting number 0, 1

식 (10)을 일반적으로 표시하면

$$X=(-1)S+S\sum_{i=0}^{B}x_i2^{-i}+S2^{-B}+\bar{S}\sum_{i=0}^{B}x_i2^{-i} \qquad (11)$$

S: sign bit $\begin{cases}0 \text{ positive}\\ 1 \text{ negative}\end{cases}$

$$X=\sum_{i=0}^{B}x_i2^{-i}+2^{-B} \quad x<0,\ S=1 \qquad (12)$$

$$X=\sum_{i=0}^{B}x_i2^{-i} \quad x>0,\ S=0 \qquad (13)$$

$\hat{X}=-S+X$, $\hat{Y}=-T+Y$로 하면

$\begin{pmatrix}S,T\text{: sign}\\ X,Y\text{: magnitude}\end{pmatrix}$

$$\hat{Z}=\hat{X}*\hat{Y}=ST+XY-SY-TX$$

로 되어 shift와 가감산으로 같이 변환시킬 수 있다.

Ⅲ. 하드웨어

정확도와 안정도가 좋은 digital filter를 real time에서 설계하기 위하여 다음의 2가지를 고려하여야 한다.

1. High speed signal processing arithmetic unit.

2. Flexibility of processor unit.

따라서 본 논문에서는 제한된 microprogramm의 길이로 이 두가지 점을 충족시킬 수 있는 회로를 구성하기 위하여 16bits μP를 사용하였다.

제안된 회로는 그림 1과 같고 real time에서 처리하기 위하여 μP를 제외한 외부회로는 TTL Logic으로 구성하였다.

Memory A, B는 filter의 차수에 의하여 정해지며 A는 FIR, B는 IIR로 이용된다.

Arithmetic unit는 74S181과 look ahead carry generator를 사용하였으며 16 bit가산의 경우 $T_A=28ns$정도의 지연시간이 걸린다. processor에서 이러한 filter의 계산을 처리하기 위하여 μP에서는 다음과 같은 내용의 마이크로프로그램을 제공한다.

1. MOV, OUT, STA(Store and Addition), STB(Store and Substraction)의 Operation Code.

2. Filter의 종류에 따른 FIR과 IIR의 구분 mode bit.

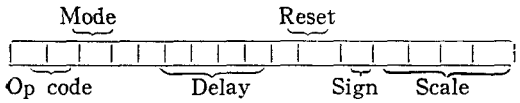
3. 곱셈을 처리하기 위한 shift size 및 방향의 sign bit.

4. Z transform에 의한 delay count.

5. Address reset (instruction restart).

마이크로프로그램의 제한 및 회로의 간소화를 위하여 program의 처리순서는, common memory의 address 0에서부터 fetch되여 reset bit이 동작될때까지 순서적으로 처리된다. 상용 μP의 1 word는 16bits이므로 이를 이용한 마이크로프로그램의 내용은 표 1과 같다.

표 1. 마이크로프로그램의 구성



다음과 같은 특성을 갖는 1KHz low pass elliptical digital filter의 경우

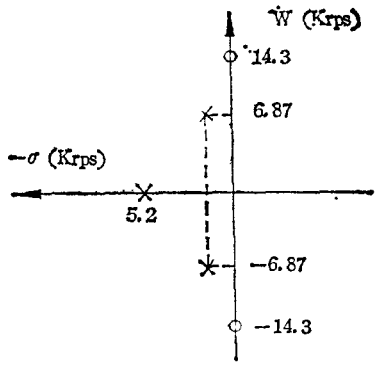
1. Pass band (DC-1KHz)에서 ripple<1 dB.
2. 2 KHz이상에서 25 dB rejection.

이의 pole 및 zero점은 그림 2와 같다.

Filter의 block diagram은 그림 3과 같다.

출력 $Y_{OUT} = A10*Y10+A11*Y11+A12*Y12$

$Y10=G1*Y00+B11*Y11+B12*Y11$



Simple pole $\sigma=0.83124$, $\omega=0$

Complex pole $\sigma=-0.31128$, $\omega=\pm1.09399$

Complex Zero $\sigma=0$, $\omega=\pm2.2701$

그림 2. Pole and Zero

표 2 3차 Elliptic Digital Filter Object Code

MOV	I.	Y01,	Y00,	R00
STA	F.	Y00,	X0,	R02
STA	I.	Y00,	Y01,	R01
STA	I.	Y00,	Y01,	R04
STA	I.	Y00,	Y01,	R05
STB	I.	Y00,	Y01,	R10
MOV	I.	Y12,	Y11,	R00
MOV	I.	Y11,	Y10,	R00
STA	I.	Y10,	Y00,	R03
STA	I.	Y10,	Y11,	R00
STA	I.	Y10,	Y11,	R02
STA	I.	Y10,	Y11,	R06
STB	I.	Y10,	Y12,	R01
STB	I.	Y10,	Y12,	R03
STB	I.	Y10,	Y12,	R05
STB	I.	Y10,	Y12,	R06
MOV	I.	Y00,	Y10,	R00
STB	I.	Y00,	Y11,	R02
STB	I.	Y00,	Y11,	R05
STB	I.	Y00,	Y11,	R08
OUT		S	Y00	L01

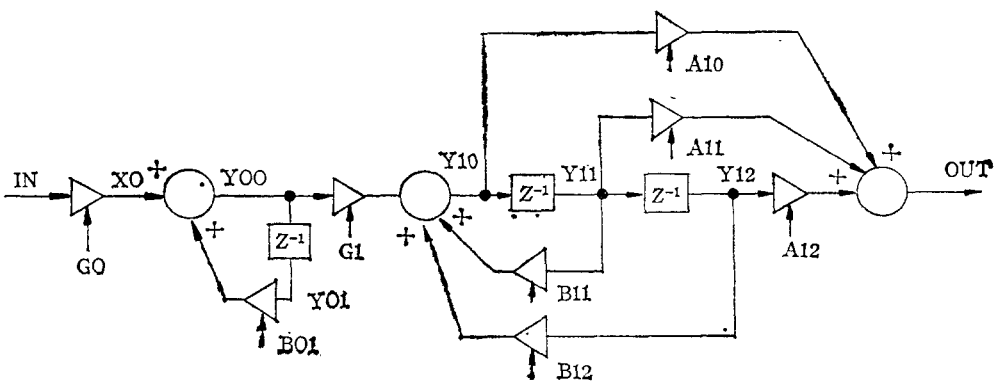


그림 3. Elliptical Digital Filter Block Diagram

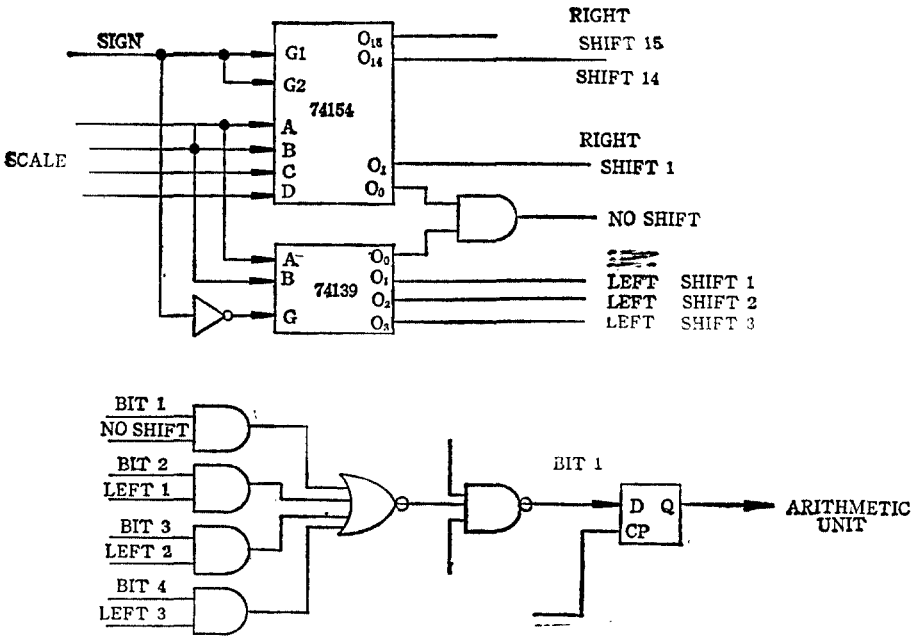


그림 4. Scaler와 Shift Magnitude Generator

$$Y00=G0*X0+B01*Y01$$

$$Y12=Y11,\ Y11=Y10,\ Y01=Y00$$

각각의 계수를 산출하여 이와 같은 필터를 처리하기 위한 object code는 표 2와 같다.

이와같은 마이크로프로그램에서 *R?? L??*은 shift의 수 및 방향으로 4 bits scale과 1 bit sign으로 구성된다. 이를 처리하기 위한 논리회로는 그림 4와 같이 common memory에서 프로그램을 입력으로 하여 74154(right shift)와 74139(left shift) multiplexer를 이용하여 shift할 크기를 얻는다.

이 출력을 invert되는 buffer를 거쳐 그림 4의 bit 1의 예처럼 7454 AOI의 입력으로 연결한다.

예로 마이크로프로그램의 sign과 scale bits이 "00010"일 경우 right shift 2이므로 74154가 동작되여 출력 O_2가 "Low"로 되고 이 신호가 invert된후 7454의 입력 bit 2와 함께 AND된다.

따라서 bit 2가 bit 1의 자리로 이동하게 된다. 본 실험에서 취한 최대 이동크기는 right 14, left 2이므로 한 instruction으로 4배의 곱셈과 $2^{-14}=0.000061$의 나눗셈을 행할 수가 있다.

gate수를 줄이기 위하여 그림 4의 scaler를 7495와 같은 36MHz의 높은 clock으로 동작되는 shift register를 이용할 경우에 최대 14bits shift를 하기 위하여 14개의 clock이 요구된다.

이는 약 0.4μs의 지연시간이 소비되므로 real time 처리 주파수가 저하되며, shift의 빈도에 따라서 처리주파수가 불균일하므로 예측주파수 범위가 필터에 따라 변경되는 단점이 발생한다.

본 논문에서 실험한 회로는 그림 5와 같다. 선간의 분포용량 등의 문제로 스위칭 지연시간에 의한 오동작을 막기 위하여 μP의 clock주파수는 2.45 MHz로, processor의 주파수는 1MHz로 동작시켰다. 실험상 공통메모리의 용량은 32×16bits로 하였다.

프로그램의 기록은 μP의 명령에의해 7474 D F·F과 함께 Tri-state gate를 이용, processor와의 연결을 차단시키고 8226을 통하여 이루어진다. 프로그램 기록이 끝난후 μP에 의해 D F·F의 clear 신호가 제거되면 위와 반대연결이 이루어지며 74193 address counter 내용이 address가 되며, 7489 메모리는 항상 read 상태이므로 이의 내용을 clock pulse에 의해 latch한다.

이와같이 저장된 마이크로프로그램은 우선 scaler를 통과하며 clock pulse가 0인 상태에서 op code

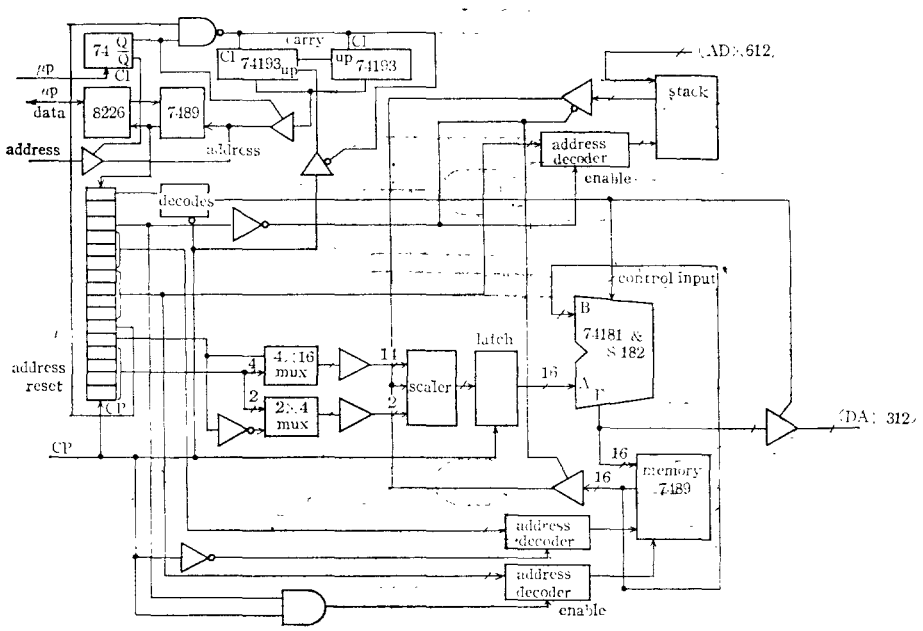


그림 5. process의 회로

가 동작되어 연산부에서 필요한 가감산을 수행하게 된다.

프로그램의 처리가 끝이 나면 reset bit이 0이 되어 μP가 processor를 제어할 때와 같이 NAND 게이트를 동작시켜 74193의 출력을 0으로 지우므로 메모리의 address 입력이 0이 되여 프로그램을 반복 수행한다. 이때 clock pulse가 동작되면 0번지의 내용을 latch할 수 없는 시간차가 존재할 경우가 발생한다. 따라서 번지수를 0으로 지울 경우에 clock을 차단시켜 이와같은 경우를 방지하였다.

회로의 stack과 메모리 7489는 F.I.R과 I.I.R.의 지연을 목적으로 구분 사용한다. 즉 모우드 비트의 1과 0에 따라 각 부분의 디코우더 동작상태가 결정되여 기록과 독출을 할 수 있다.

표 2의 STA F Y00 X0 R02의 경우에 X0의 내용인 "0000"가 어드레스 디코우더에 공급되며 동시에 모우드가 F이므로 이를 동작시키고 반면 메모리 7489의 디코우더 및 buffer를 차단시킨다. 따라서 이의 내용이 scaler에 의해 오른편으로 2비트 이동되고 clock pulse가 0인 상태에서 Y00번지에 해당하는 000값이 7489의 어드레스 디코우더에 공급되여 곱하여진 값이 저장 처리된다.

Data sheet에 의한 최대 지연시간을 기준으로 instruction의 cycle time은

OUT=194ns, MOV=116~210ns,

STA, STB=210ns.

따라서 최대 sample frequency

$f_{s(MAX)}=1/(210\times10^{-9})*(\text{Instruction No.})$

로 결정된다.

여기서 gates와 delay time과의 관계를 구하면 N bits의 Schottky TTL을 사용한 Dada's multiplier의 경우[6], 지연시간

$$T_M=18.84\log_2 N+T_A(2N)+20\text{ns} \quad (14)$$

$T_A(2N)$: $2N$ bit addition time

gate수

$$M_M=13N^2-16N \quad (15)$$

Abraham Peled에 의한 canonical signed digit code를 이용한 경우에 [1, 5]

$$T_C=\frac{N}{a_N}T_A(2N)\text{ns} \quad (16)$$

a_N: average no. of nonzero digits

$M_C=33N+3.63N\log_2 N$

본논문에서는 shift 및 가감산으로 처리하므로 scaler에 의해 좌우되여 이에대한 값은

$$T_E=\frac{N}{a_P}(T_A(2N)+T_D)\text{ns} \quad (18)$$

a_P: average No. of nonzero digits

$$M_E=\frac{1}{2}(N+1)(N+2) \quad (19)$$

Dada's multiplier에 대한 efficiency ratio를 구하면

$$E_{(E/M)}=\frac{T_E\cdot M_E}{T_M\cdot M_M}$$

$$=\frac{N(T_A(2N)+T_D)(N+1)(N+2)}{2a_P(18.84\log_2 N+T_A(2N)+20)(BN^2-16N)} \quad (20)$$

$T_D=18.84\log_2 N+20$

$a_P=3$

이라 가정하면

$$E_{(E/M)}=\frac{N(N+1)(N+2)}{6(13N^2-16N)}$$

$N=8$bits인 경우 $E_{(E/M)}(8)=0.17$

$N=16$bits인 경우 $E_{(E/M)}(16)=0.265$

따라서 Dada의 방법에 비하여 4~5배의 효율을 향상시킬 수가 있다.

Abraham Peled에 대하여

$$E_{(E/C)}=\frac{T_E\cdot M_E}{T_C\cdot M_C}$$

$$=\frac{\frac{N}{2a_P}(T_A(2N)+T_D)(N+1)(N+2)}{\frac{N}{a_N}T_A(2N)33N+3.63N\log_2 N)} \quad (21)$$

$E_{(E/C)}(8)=0.52$($T_D=3.63N\log_2 N$인 경우)

$E_{(E/C)}(16)=1.87$ ($T_D=3.63N\log_2 N$인 경우)

그러나 본회로에서는 bit수의 증가에 관계없이 T_D가 일정하므로 $T_D=3.63N\log_2 N$가 성립하지 않는다.

따라서 $T_D=50$ns로 단축시킬 경우

$E_{(E/C)}=0.56$, $T_D=111$ns 일 때 Peled와 같은 효율을 얻는다.

결국 a_P 및 T_D에 의하여 효율을 개선시킬 수가 있으며 8bits의 경우 Dada에 비해 5배를 Peled에 대하여 2배의 효율을 증가시킬 수가 있다.

Ⅳ. 결 론

Real time에서 digital signal을 처리하기 위한 processor를 마이크로프로그램에 의하여 제어하므로 flexibility 및 속도면에서 개선을 얻을 수 있었다.

μP는 필요한 마이크로프로그램을 변환하여 제공하므로 쉽게 filter를 변경시킬 수 있는 반면, 모든 처리동작은 processor에 의하여 수행되므로 sampling frequency는 프로그램의 길이에 반비례한다. 본실험에서 지연시간 T_D는 회로를 IC로 fabrication한다면 충분히 제거 가능하므로 계산된 비교 효율보다 증가 시킬 수가 있다.

Bit수가 증가함에 Peled방법보다 canonical code로 변환하기 위한 algorithm과 efficiency, 양면에서 모두 양호한 결과를 얻을 수 있었다.

구성된 회로는 filter의 종류 및 차수에 구애됨이 없이 사용할 수가 있으므로 이용도가 높으나, FIR의 경우에 차수가 높아짐에 이에 따라 instruction수가 증가하여 sampling을 처리하는 시간이 지연되는 단점이 있다.

이러한 경우에 마이크로프로그램의 길이를 확대하여 곱셈을 1회에 계수로 처리하는 방법등을 고려할 수가 있다. 본 실험 결과 일반 μP와 간단히 연결 가능한 interface부분 및 다중채널통신에 processor와 계속적인 제어관계등의 연구가 필요하다.

參 考 文 獻

1. Abraham Peled,, "On the Hardware Implemenentation of Digital Processors," IEEE Trans. Acoust., Speech, Signal Processing. Vol. ASSP-24 pp.77—86. Feb. 1976.
2. B. Liu and A. Peled, "A New Hardware Realization of High Speed Fast Fourier Trans forms," IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-23, pp.543—547, Dec. 1975.
3. Intel. "The 2920 Analog Signal Processor Design Handbook," Aug. 1980.
4. The Bell System Technical Journal Vol 60. No.7 Sept. 1981.
5. A. Peled and B. Liu, "Digital Signal Processing" John Wiley & Sons. 1976. pp.174—183.
6. A. Habibi and P.A. Wintz, "Fast Multipliers," IEEE Trans. Comput. Vol C-19, pp. 153—157, Feb. 1970.

7. Y.S. Wu, "Architectural Considerations of a Signal Processor under Microprogram Control," 1972 Spring Joint Computer Conference, AFIPS Proc., Vol.40, pp.675—683.
8. J. Allen,"Computer Architecture for Signal Processing," Proc. IEEE, Vol 63. pp.624—633. Apr. 1975.
9. H.T. Nagle, Jr., and V.P. Nelson, "Digital Filter Implementation on 16-bit Microcomputers," IEEE MICRO Vol.1. No.1 pp.23—41, Feb. 1981.
10. S. L. Freeny, "Special Purpose Hardware for Digital Filtering. Proc, IEEE Vol 63. pp. 633—648, Apr. 1975.
11. R.R. Shively. "Architecture of a Programmable Digital Signal Processor," IEEE Trans, Comput. Vol, 3-31, No.1 pp.16—22, Jan. 1982.
12. G.L. Kratz, "A Microprogrammed Approach to Signal Processing," IEEE Trans. Comput., Vol.c-23, pp.808—817, Aug. 1974.
13. 이대영, 진용옥, 허도근, "디지탈필터의 설계와 구성" 전자공학회지 Vol.17 No.4 pp.11—17. Aug. 1980.
13. Jan Zeman and H. Troy Nagle, Jr. "A High-Speed Microprogrammable Digital Processor Employing Distributed Arithmetic," IEEE Journal of Solid-State Circuits, Vol. SC-15, No. 1, Feb. 1980.