論文 90-27-2-18

이중 완전 Shuffle을 이용한 Radix-4 FFT 프로세서의 설계

(Design of Radix-4 FFT Processor Using Twice Perfect Shuffle)

黃 明 夏*, 黃 好 正*

(Myoung Ha Hwang and Ho Jung Hwang)

要 約

본 논문에서는 레딕스(radix)-2 FFT 알고리듬에 이용하였던 완전 셔플(shuffle)을 확장하여 새로이 얻은 이중 완전 셔플을 적용하여 레딕스-4 FFT 프로세서를 설계하였다. 이 FFT 프로세서는 버터플라이 연산 회로, 입, 출력값과 계수의 번지 발생기, 입, 출력값을 일시 저장하는 레지스터와 제어 회로로 구성된다. 또한 입, 출력값과 계수를 저장하기 위해 외부 RAM과 ROM을 필요로 한다. 버터플라이 회로는 12개의 곱셈기와 덧셈기, 뺄셈기, 딜레이 시프트 레지스터(delay shift register)로 되어 있다. 25MHz two phase 클럭으로 동작하는 이 프로세서는 256-절 FFT를 6168 클럭, 즉 247 us에 계산을 하며 또한, 사용자가 4, 16, 64, 256-점까지 임의의 점을 선택할 수 있는 유연성을 갖는다.

그리고 2-um 이중 메탈 CMOS 공정을 이용하여 28000 여개의 트랜지스터와 55개의 패드를 8.0× 8.2㎟면적에 설계할 수 있었다.

Abstract

This paper describes radix-4 Fast Fourier Transform (FFT) Processor designed with the new twice perfect shuffle developed from a perfect shuffle used in radix-2 FFT algorithm. The FFT Processor consists of a butterfly arithmetic circuit, address generators for input, output and coefficient, input and output registers and controller. Also, it requires the external ROM for storage of coefficient and RAM for input and output. The butterfly circuit includes 12 bit-serial (16x8) multipliers, adders, subtractors and delay shift registers. Operating on 25 MHz two phase clock, this processor can compute 256 point FFT in 6168 clocks, i.e. 247 us and provides flexibility by allowing the user to select any size among 4,16, 64, and 256 points. Being fabricated with 2-um double metal CMOS process, it includes about 28000 transistors and 55 pads in 8.0 x 8.2 mm^2 area.

I. 서 론

최근 디지탈 신호 처리 분야에서는 사용자의 필요에 따라 필터링, 콘벌루션(convolution), 코리레이션(correlation)등을 선택적으로 수행할 수 있는 범용 DSP 칩과 고유의 한 기능만을 수행하는 전용 DSP 칩의 설계를 위한 연구가 계속 되고 있다.

본 논문에서는 1965년 Cooley-Tukey의 개발 이후 시스템 분석, 디지탈 필터링, 스펙트럼 분석[1]등 여

*正會員, 中央大學校 電子工學科
(Dept. of Elec. Eng., Joongang Univ.)

接受日字 : 1989年 1月 25日

러 분야에서 이용하여온 Fast Fourier Transform (FFT)수행 칩을 연구, 설계하였다.

일반적인 FFT 흐름도로 FFT를 직접 실행할때 수치의 입, 출력 경로가 복잡함으로, 이를 해결하기 위해 다른 FFT 실행 알고리듬 보다 데이타의 경로가 간결하며,[2] 성능이 우수한 시스템을 얻을 수 있는 완전 셔플을 확장하여, 새로이 제안하는 이중 완전 셔플을 레딕스-4 FFT 알고리듬에 적용하였다. 그래서 모든 단 (stage)의 입, 출력 양식이 같은 FFT흐름도를 얻어 제어가 용이하고, 고속 연산이 가능케 하였다. 또한 사용자의 필요에 따라 4, 16, 64, 256 중 임의의 point로 FFT 연산을 수행할 수 있는 유연성을 갖도록 FFT 프로세서를 설계하였다.

1. 레딕스-4 FFT 알고리듬

레이딕-4 FFT 알고리듬의 원리는 N-점 DFT를 4개의 N/4-점 DFT로 나눈뒤, 그 출력을 조합하여 N-점 DFT의 최종 출력을 얻고 다시 각각의 N/4-점 DFT를 4개의 N/16-점 DFT로 나누는 방식으로 최종 4-점 DFT로 분할될 때까지 위 방법을 계속함으로써 레딕스-4 FFT 흐름도를 얻을 수 있다. 그림 1 은 64-점 레딕스-4 FFT 흐름도이다.

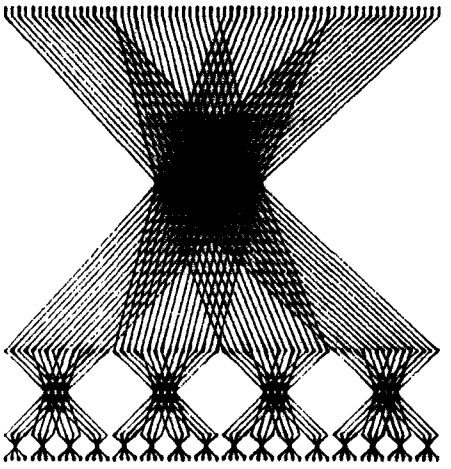


그림 1. 64-point radix-4 FFT 흐름도
Fig. 1. 64-point radix-4 FFT flow graph.

위와 같은 레딕스-4 알고리듬에서 단의 수와 각 단의 버터플라이 수는 각각 $\log_4 N$과 N/4으로 총 버터플라이 수는 $N/4\cdot\log_4 N$이지만, 레딕스-2 알고리듬의 버터플라이 수는 $N/2\cdot\log_2 N$이다. 예를들어 64-점 FFT 연산을 할때 레딕스-2 알고리듬은 $64/2\cdot\log_2 64 = 192$의 버터플라이가 필요하나, 레딕스-4 알고리듬은 $64/4\cdot\log_4 64 = 48$의 버터플라이만이 필요하다. 이처럼 레딕스-4 알고리듬은 레딕스-2 알고리듬에 비해 버터플라이수를 1/4로 줄일 수 있으므로 더 빠르게 연산을 수행할 수 있는 알고리듬이다.

2. 버터플라이 연산

그림 1 에서 처럼 FFT 흐름도는 그림 2 와 같은 버터플라이 기본 요소로 구성된다. 이때 수치들은 모두 복소수이므로 이 버터플라이는 12번의 곱셈과 15번의 덧셈과 뺄셈이 각각 필요하다. 그런데 레딕스-4 알고리듬으로 FFT를 수행할 때 최종 출력은 digit-reversed 순이 되는데 버터플라이에서 X(1) 과 X(2)를 교환함으로써 레딕스-2와 같은 bit-reversed 순서로 배열된 출력을 얻을 수 있다.[6][7]

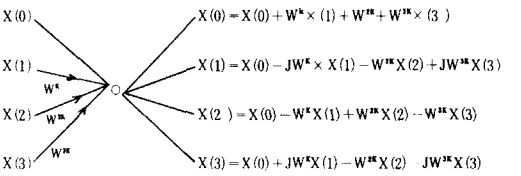


그림 2. Butterfly 연산
Fig. 2. Butterfly Arithmetic.

II. 레딕스-4 FFT 알고리듬과 버터플라이 연산

스펙트럼 분석, 음성신호 처리 등의 디지탈 신호처리 분야에서 중요한 계산인 이산 퓨리에 변환(Discrete Fourier transform(DFT))은 식(1)과 같이 나타낼 수 있다.[1][3][4]

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W^{nk} \qquad (1)$$

$$W = \exp(-j(2\pi/N))$$

$$k = 0, 1, \cdots, N-1 \quad 0 \le n \le N-1$$

위와 같은 N-점 DFT를 직접 계산할 경우 N^2의 복소수 곱셈이 필요한데 N이 큰값일 경우 계산수의 큰 증가로 빠른 계산이 불가능하다. 그래서 계산수를 줄여 빠른 연산이 가능하도록 개발된 알고리듬이 Fast fourier transform(FFT)이다.

FFT 알고리듬은 Decimation-in-time(DIT), Decimation-in-frequency(DIF) 알고리듬과 레딕스에 따라

여러 알고리듬을 포함하는데 본 논문은 레딕스-4 DIT 알고리듬을 이용하였다.

Ⅲ. 이중 완전 셔플을 이용한 FFT 알고리듬

1. 이중 완전 셔플 연결도

이중 완전 셔플 연결도는 다음 원리에 의해 구성되어 진다. $N=4^{k/2}$ (K : dit 수, 짝수)이고 0, 1, 2, … N−1 까지의 노드(node)로 구성될때 어떤 노드 i를 이진수 $a_{k-1}a_{k-2}\cdots a_1a_0$로 나타낼 수 있다. 이때 이중 완전 셔플 연결에서는 $a_{k-1}a_{k-2}=0\ 0$이면 $i<N/4$이고 $a_{k-1}a_{k-2}=0\ 1$이면 $N/4\le i<2N/4$, $a_{k-1}a_{k-2}=1\ 0$이면 $2N/4\le i<3N/4$, $a_{k-1}a_{k-2}=1\ 1$이면 $3N/4\le i<N$을 의미한다.

이때 완전 셔플 연결에서는[5] $a_{k-1}=0$이면 노드 i는 2i에, $a_{k-1}=1$이면 2i−(N−1)에 연결되지만, 이중 완전 셔플 연결에서는 $a_{k-1}a_{k-2}=00$이면 노드 i는 4i에, $a_{k-1}a_{k-2}=01$이면 4i−(N−1)에, $a_{k-1}a_{k-2}=10$이면 4i−2(N−1)에, $a_{k-1}a_{k-2}=11$이면 4i−3(N−1) 노드에 이중 완전 셔플로 연결된다. 이를 이진수로 나타낼 때 노드 i가 노드 j에 연결되면, 노드 i와 노드j를 다음 식(2)와 같이 나타낼 수 있다.

$$i=a_{K-1}2^{K-1}+a_{K-2}2^{K-2}+\cdots+a_1 2+a_0 \qquad (2)$$
$$j=a_{K-3}2^{K-1}+a_{K-4}2^{K-2}+\cdots+a_1 2^3+a_0 2^2+a_{K-1}2+a_{K-2}$$

예를들어, k=4, N=16경우 1(0001)노드는 4(0100)에, 6(0110)노드는 4×6−15=9(1001)노드에 연결된다. 그림3은 N=16 일때 구성되어지는 새로운 이중 완전 셔플 연결도이다.

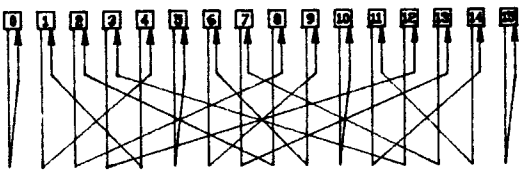


그림 3. N=16 일때의 이중 완전 Shuffle 연결도
Fig. 3. Twice Perfect Shuffle connection graph for N=16.

2. 이중 완전 셔플을 적용한 FFT 알고리듬

그림1에서 볼 수 있듯이 FFT 흐름도에서는 단간의 데이타 전송경로가 단마다 모두 달라 하나의 버터플라이 연산회로로 FFT 연산을 수행할 때는 입, 출력 제어가 매우 어렵게 된다. 그래서 다음과 같이 이중 완전 셔플을 이용하여 모든 단 간의 입, 출력 양식이 같은 흐름도를 얻을 수 있다. 먼저 FFT 초기 입력이나, 단(stage)에서 입, 출력 하나를 한 노드로 정의하고 노드 순으로 4개의 노드로 한 셀(cell)을 구성할 때 N−점 FFT의 경우, N개의 노드와 N/4개의 셀로 한 단을 나타낼 수 있다. 예를 들어 64-점 FFT의 경우 각 단은 64개의 노드와 16개의 셀로 나타낼 수 있다.

그림4는 일반적인 FFT 흐름도에서 이중 완전 셔플을 적용시켜 제어 구조가 간단하게 될 수 있도록 셀의 번호를 지정하는 과정을 나타내고 있다. 첫단의 출력이 0, 1, 2, 3 노드로 구성된 0번 셀, 16, 17, 18, 19 노드로 된 4번 셀, 32, 33, 34, 35 노드의 8번 셀과, 48, 49, 50, 51 노드로 구성된 12번 셀에서 각각의 두번째 노드(그림4. 의 굵은 선)인 1(000001), 17(010001), 33(100001), 49(110001)노드가 연결된 두번째 단의 셀은 위의 노드 번호들을 각각 이중 완전 셔플시킨 4(000100), 5(000101), 6(000110), 7(000111) 노드로 구성된다. 그래서 4개의 노드로 한 셀을 만드는 과정에서 4, 5, 6, 7노드는 1번 셀에 해당되므로 이 셀을 1번이라고 지정하였다.

첫째단의 나머지 출력 노드를 이중 완전 셔플시켜 상기와 같은 방법으로 둘째단의 다른 모든 셀에 번호를 지정하였으며 셋째단은 둘째단에서 같은 방법으로 이중 완전 셔플시켜 셀 번호를 결정하였다. 그리고 그림4의 각 단에 있는 셀을, 붙인 번호 순에 따라 나열하면 각 단의 데이타의 흐름이 같은 모양을 하고 있는 그림5를 얻을 수 있다. 이 그림에서 단 간에 상호 연결된 노드의 번호를 살펴보면 0→0→0→0, 1(000001)→4(000100)→16(010000)→1(000001), 및 17(010001)→5(000101)→20(010100)→17(010001)등과 같은 모양이 나타남을 알 수 있다. 이것은 처음 시작하는 각 노드를 다음 단에 연결할 때는, 그 노드의 이진수값에서 좌우 두번 이동하여 생성되는 값이 연결되는 노드의 번지로 표현되는 이중 완전 셔플 형태를 이루고 있다. 따라서, 레딕스-4 FFT 연산을 위해 본 논문에서 제안하는 이중 완전 셔플을 이용함으로써 모든 단에서 데이타의 입, 출력 흐름이 같은 흐름도를 얻을 수 있어, 단지 한 단만을 위한 제어 알고리듬만으로도 전체 FFT를 수행할 수 있는 제어의 용이성을 발견할 수 있다.

Ⅳ. FFT 프로세서의 설계

1. FFT 프로세서의 구조

본 논문의 FFT 프로세서는 그림6에서 처럼 계수를 저장하는 외부 ROM과 입, 출력 값을 저장하는

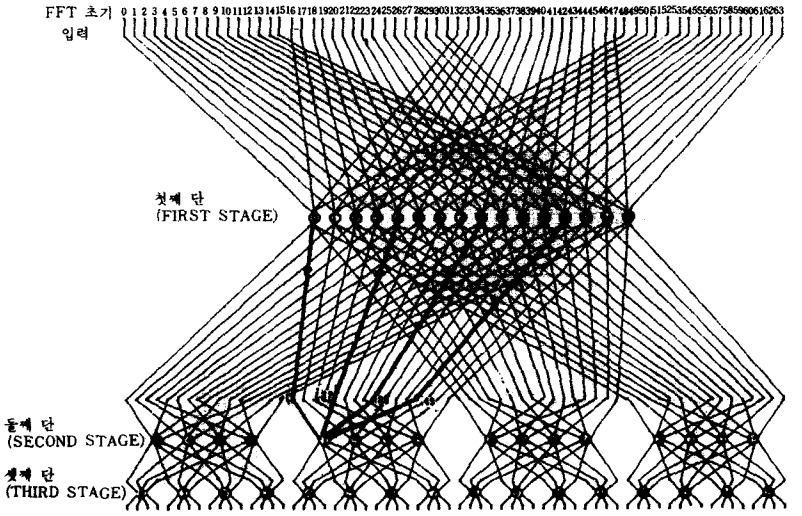


그림 4. 이중 완전 Shuffle 양식으로 번호를 붙힌 64-point radix-4 FFT flow graph.
Fig. 4. 64-point radix-4 FFT flow graph numbered by Twice Perfect Shuffle.

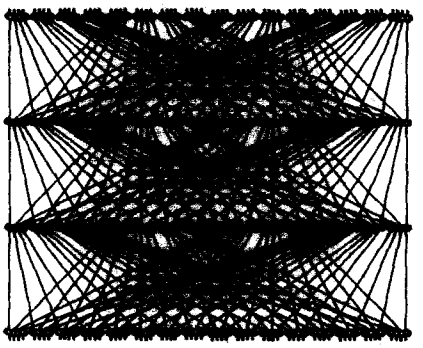


그림 5. 이중 완전 Shuffle 양식으로 재배치한 FFT flow graph
Fig. 5. FFT flow graph realigned by Twice Perfect Shuffle.

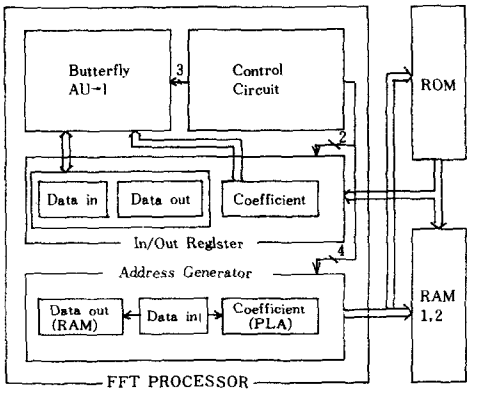


그림 6. 전 system의 Block도
Fig. 6. Block diagram of all system.

두개의 외부 RAM과 bit-parrallel 방식으로 데이타의 전송을 하며 FFT 연산을 실행한다.[6]

이 프로세서는 bit-serial 연산을 하는 버터플라이 연산회로와 입력치와 계수를 일시 저장하는 두개의 PISO (parallel-in/serial-out) 레지스터와 출력값을 위한 SIPO (serial-in/parallel-out) 레지스터로 구성되며, 또한 PLA로 설계한 계수 번지 발생기, D 플립-플롭을 이용한 동기 카운트와 점의 가변을 위한 조합회로로 구성한 입력값 번지 발생기, (9×16) RAM 출력값 번지 발생기와 전 시스템을 제어하는 제어 회로를 포함한다.

그림7은 제어 회로이며 그림8과 그림9는 각각 입력값과 계수의 번지 발생기 회로이다. 이때 그림8에서 처럼 카운터의 두번째, 세번째 출력을 입력 번지의 첫번째, 두번째 MSB로 함으로써 이중 완전 셔플과 일치하는 데이타 전송 경로를 갖게 되는 것이다.

2. 버터플라이 회로

여기서 FFT 프로세서의 가장 중요한 회로인 버터

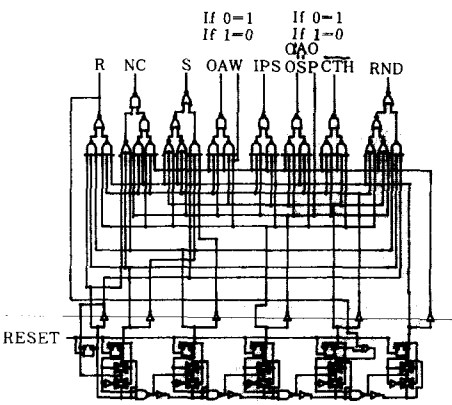


그림 7. Control 회로
Fig. 7. Control circuit.

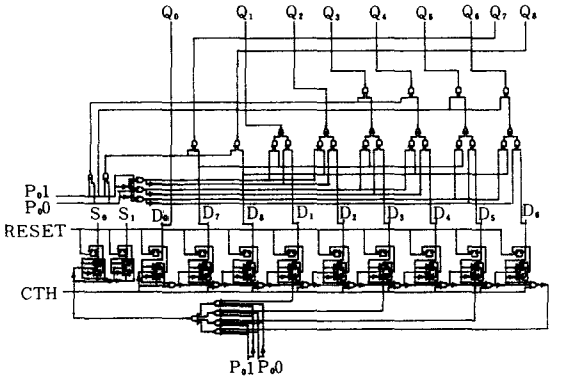


그림 8. 입력값 번지 발생기
Fig. 8. Address generator for input.

플라이는 그림10과 같이 15개의 전가산기와 전감산기, 2개의 딜레이 시프트 레지스터와 12개의 곱셈기가 필요하다. 여기서 딜레이 시프트 레지스터는 다른 입력(X1r-X3i)으로 곱셈을 수행하는 동안 두 입력값(X0r, X0i)을 지연시키는 기능을 한다. 그리고 곱셈기는 다른 bit-serial 곱셈기에 비해 레이아웃 하였을 때의 면적과 그 연산 시간을 약 1/2로 줄일 수 있는 Modified Booth 알고리듬을 이용하여 그림11과같이 설계할 수 있었다.[9] 이 그림에서 계수(CO)를 임의의 bit로 확장하려면 총 stage 수가 bits/2이 되도록 first stage 만을 증가시키면 된다.

본 논문에서 입, 출력치는 16 bit, 계수는 8 bit로 고정 소수점 형식으로 나타냈으며 입, 출력치는 $-2 \leq x < 2$로 제한하였으며, 버터플라이 연산회로의 출력

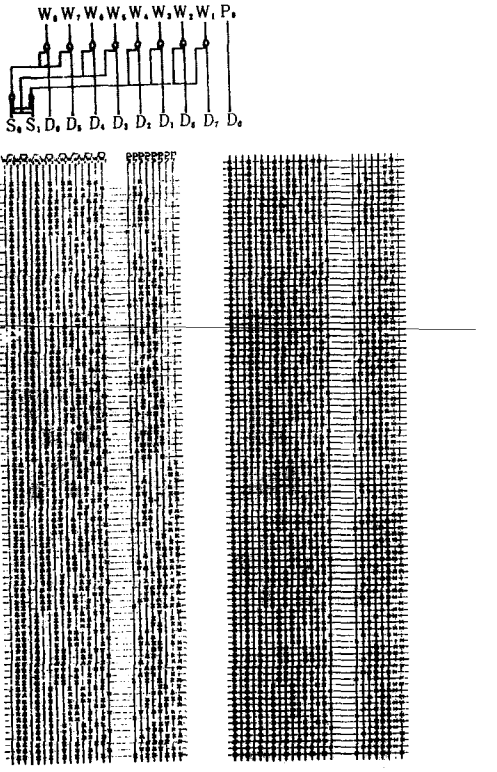


그림 9. Coefficient 번지 발생기
Fig. 9. Address generator for coefficient.

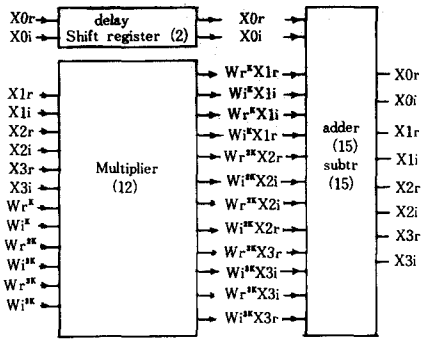


그림10. Butterfly 연산회로의 Block도
Fig. 10. Block diagram of Butterfly arithmetic circuit.

값을 그냥 truncation 할때 생기는 에러를 줄이고자 Rounding 방식을 이용하였다.[3]

3. FFT 연산의 수행

그림12와 같은 전 시스템의 제어 신호도에서 보듯이

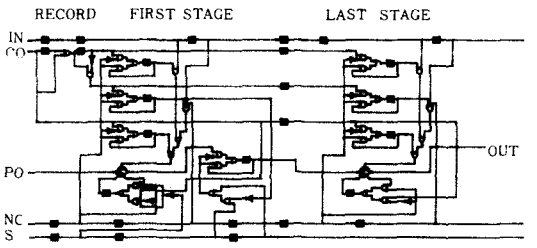


그림11. Modified Booth 알고리듬 곱셈기
Fig. 11. Multiplier designed by Modified Booth's Alorithm.

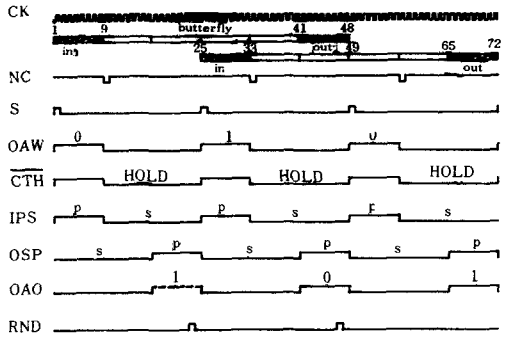
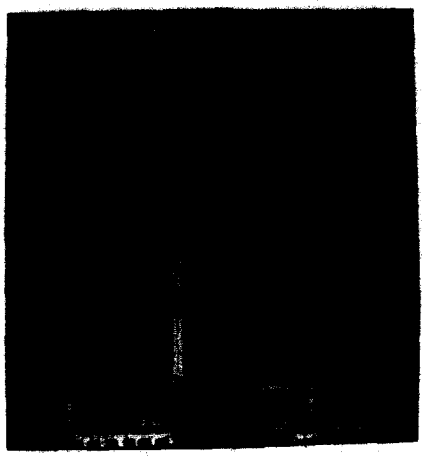


그림12. 전 system의 control timing 도
Fig. 12. Control timing diagram of all system.



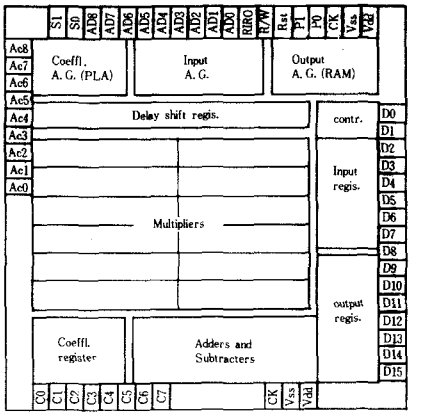


그림13. FFT Processor의 layout 사진과 floor plar
Fig. 13. Photograph and floor plan of FFT Processor.

프로세서는 다음과 같이 연산을 수행한다. 먼저 프로세서는 입력값과 계수의 번지를 이중 셔플 방식으로 발행하여 외부 RAM(1)과 ROM에서 입력값 8개와 계수 6개를 PISO 레지스터로 받아 들이고 동시에 출력값의 번지를 내부 RAM에 저장한다(1-8 클럭). 이렇게 입력된 값들은 bit-serial 방식으로 버터플라이 연산회로를 거쳐 SIPO 레지스터에 차례로 저장된다(9-40클럭).

이때 bit-serial 버터플라이 연산은 다음과 같다. 모든 입력을 받아들인 상태에서 입력들을 serial로 곱셈기에 입력시키고 난 5클럭 뒤에(13번째 클럭) 첫 곱셈 결과값이 출력되고 34번째 클럭까지 총 22bit의 곱셈 결과값이 출력된다. 그리고 ADD/SUB단을 지나 SIPO 레지스터에 입력된다.

이렇게 한 버터플라이 연산을 마치면 내부 RAM의 출력값 번지를 발생시키면서, 레지스터에 저장된 결과값은 외부 RAM(2)에 차례로 저장된다(41-48 클럭). 이와 같은 FFT 연산에서 다음 단의 계산은 전단에서 결과를 저장시킨 외부 RAM에서 데이타를 받아들여 연산을 한다.

이런 방식의 FFT 연산은 그림12에서 처럼 한 버터플라이 연산을 수행하는 중에 또 다른 입력들을 받아들여(25 클럭부터) 버터플라이 연산을 하는 파이프라인(pipeline) 방식을 적용함으로써 연산 속도를 크게 증가시킬 수 있었다.

V. 프로세서의 성능과 특징

25 MHz two phase 클럭으로 동작하는 본 프로세서는 이중 매탈 CMOS 공정으로, 약 28000개의 트랜지스터를 8.0×8.2㎟의 면적으로 설계할 수 있었으며 55개의 functional 및 power 핀이 필요하다. 이때 power 감쇠나 클럭의 skew로 인한 전체 시스템의

오동작을 막기 위해 power와 클럭 핀을 각각 두개 및 한개를 더 첨가하여 각 핀수가 네개와 두개로 되게 하였다.

그리고 이 FFT 연산은 그림12에서 처럼 총 48클럭이 필요한 버터플라이 연산을 수행하는 도중, 즉 8개의 데이타를 받아들이고 16 bits의 데이타를 serial로 곱셈기에 입력시키고 난 24클럭 후 25클럭부터 새로운 데이터들을 받아들여 버터플라이 연산을 하는 파이프라인 방식을 적용하였다.

그래서 256개의 버터플라이 연산이 필요한 256-점 FFT 연산은 이를 한 버터플라이 연산이 끝난 후 새로운 입력들을 받아들여 연산을 하는 난 파이프라인(non-pipeline)방식을 적용할 경우 12288클럭(256×48)이 필요하나, 파이프라인 방식의 적용으로 단지 6168클럭(247us)이 필요해 연산 속도를 약 2배 증가시킬 수 있었다.

Ⅵ. 결 론

본 논문은 레딕스-2 FFT 알고리듬에 적용했던 완전 셔플을 두번 수행하여 새로이 얻은 이중 완전 셔플을 소개하였으며, 이를 레딕스-4 FFT 알고리듬에 적용하였다. 그래서 FFT 흐름도를 직접 이용하였을 경우 모든 단에 각각 다른 입력 번지를 발생시키는 회로가 필요하게 됨으로써 프로세서의 면적이 커지고, 입,출력 제어가 복잡해 진다. 그러나 이중 완전 셔플의 적용으로 단(stage) 간의 모든 입,출력 경로가 같은 FFT 흐름도를 얻어 데이타의 번지 발생이 간결해짐으로써 시스템의 제어가 용이하고, 제어회로의 면적이 감소하게 된다.

그리고 입력값 번지 발생기에 조합회로를 첨가함으로써 사용자의 필요에 따라 4, 16, 64에서 256-점까지 가변하면서 연산을 할 수 있도록 FFT 프로세서를 설계하였다.

그래서 256-점 FFT를 247us로 수행할 수 있었으며, 다른 알고리듬을 적용한 FFT[6][7][8][10]보다 더 효율적인 제어로 연산을 수행할 수 있었다.

參 考 文 獻

[1] A.V. Oppenheim and R.W. Scafer, "Digital signal processing," Prentice-Hall 1975.

[2] C.D. Thompson "Fourier transforms in VLSI," *IEEE Trans. Comput*, vol. C-32 no. 11, pp. 1047-1057, Nov. 1983.

[3] L.R. Rabiner and B. Cold, "Theory and application of digital signal processing," Prentice-Hall. 1975.

[4] P. Denyer and D. Renshaw, "VLSI signal processing: a bit-serial approach," Addison-Wesely, 1985.

[5] H. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Computers*, pp. 156-161, Feb. 1971.

[6] R.W. Linderman. P.M. Chau, W.H. Ku and P.P. Rensens, "CUSP : A-2um CMOS digital signal processor," *IEEE J. Solid-state circuits*, pp. 761-769, June, 1985.

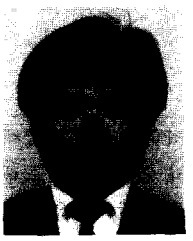
[7] M.J. Corinthios, K.C. Smith and J.L. Yen "A parallel radix-4 fast fourier transform computer," *IEEE Trans. Compters*, pp. 80 92 January, 1975.

[8] J. Fox, G. Surace and P.A. Thomas, "A self-testing 2-um CMOS chip set for FFT application," *IEEE J. Solid-State Circuits*, pp. 15-19, Feb. 1987.

[9] R.F. Lyon, "Two's complement pipelin multiplier," *IEEE Trans Computers*, pp. 418-425 April. 1976.

[10] 박세현, 황호정, "Shuffle-exchange graph를 적용한 Single chip 32-point FFT 프로세서의 설계"대한 전자 공학회 1987년 추계종합 학술대회 논문집.

著 者 紹 介



黃 明 夏(正會員)

1965年 3月 30日生. 1987月 2月 중앙대학교 전자공학과 졸업. 1989年 2月 중앙대학교 대학원 전자공학과 졸업. 공학석사학위 취득. 1989年 1月～현재 금성이렉트론 제2연구실 SRAM그룹 연구원 근무. 주관심분야는 SRAM, DSP 등임.

黃 好 正 (正會員) 第26卷 第7號 參照
현재 중앙대학교 전자공학과 조교수·부교수