전력전자학술대회 논문집 2002. 7. 3 ~ 7. 6

VHDL을 이용한 전력 계측용 FFT processor 설계

이정복, 박해원*, 김수곤, 전희종
숭실대학교 전기공학과, *(주)로템

The Design of FFT Processor for Power measurement using VHDL

Jeong-Bok Lee, Hae-Won Park*, Soo-Gon Kim, Hee-Jong Jeon
Dept. of Electrical Eng. Soong-Sil Univ., *Rotem

ABSTRACT

In this paper, the FFT processor for power measurement using VHDL (Very high-speed integrated circuit Hardware Description Language) is discussed. The proposed system relies on the FFT algorithm to compute real and reactive power. The advantage of system is that harmonic analysis is carried out on a period of the input signal.[1] The proposed system is based on FFT processor which is designed using VHDL. In the design of FFT processor, radix-2^2 is adopted to reduce several complex multipliers for twiddle factor. And this processor adopt pipeline structure. Therefore, the system is able to have both high hardware efficiency and high performance.

1. 서 론

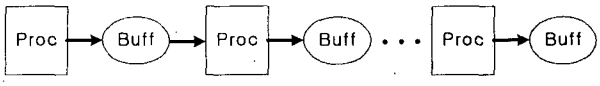
최근 전력산업의 발달로 산업현장에서 반도체 전력소자의 사용이 빈번해짐에 따라 전압, 전류 파형이 정현파를 유지하지 못하는 경우가 많아지고 있다.[2] 이때 발생되어 전력계통, 산업장비 등에 좋지 않은 영향을 끼치는 고조파를 측정, 분석하는 장치가 필요하다. 본 논문에서는 이러한 고조파 분석을 위한 FFT processor를 설계하고 검증하였다. 이 시스템을 통해 전압, 전류의 실효치, 역률, THD (Total Harmonic Distortion)등의 데이터를 얻을 수 있다.[3] 제안된 시스템을 설계하기 위해서 제어의 용이와 하드웨어의 효율성, 데이터 처리속도를 위하여 파이프라인 구조와 Radix-2^2 알고리즘을 사용하였으며 VHDL을 사용하여 SOC (System On Chip)를 적용하였다.

2. 본 론

2.1 FFT processor의 구조

FFT processor를 구현하는 방법은 크게 단일 버터플라이 연산자를 이용하는 방법, 파이프라인 구조를 이용하는 방법이 있다. 그림 1(a)는 하나의 버터플라이 연산부와 하나의 메모리를 사용하여 메모리로부터 데이터를 읽어 버터플라이 연산을 수행한 뒤 다시 같은 메모리로 결과를 저장하는 In-place 방식으로 하드웨어 비용이 적게 드는 장점이 있다. 그림 (b)는 한쪽 메모리로부터 데이터를 읽어 연산을 수행한 뒤 다른 메모리에 결과를 저장하는 방식이다. 위의 두 메모리 구조는 적은 하드웨어 구조를 가지는 반면 성능이 떨어지며 높은 주파수로 동작해야 하는 단점이 있다.

(a) 단일 메모리 구조 (b) 이중 메모리 구조



(c) 파이프라인 구조

그림 1 FFT Processor 의 구조

Fig. 1 FFT Processor structure

그림 (c)는 파이프라인 구조를 나타내는데 매 스테이지마다 버터플라이 연산부를 사용하는 구조로 하드웨어의 구조가 규칙적이므로 제어가 간단하며 순차적 입출력을 제공하기 때문에 높은 성능을 요구하는 분야에서 가장 많이 사용하는 구조이다. 따라서 본 논문에서도 데이터 처리 속도와 하드웨어 복잡도의 교환관계를 가장 잘 만족시키는 파이프라인 구조로 FFT processor의 구현하였다.

2.2 FFT 알고리즘

FFT 알고리즘은 길이 N인 수열의 이산 푸리에 변환을 보다 적은 이산 푸리에 변환으로 분해하여 계산하는 것을 기본으로 한다. 이번 절에서는 인덱스 분해법에 의한 FFT 알고리즘을 본 논문에서 적용한 파이프라인 구조에 적용하여 논한다.

식 (1)은 DIF 의 기본식을 나타내고 있다.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)nk} \qquad k=0,1,2,\ldots,N-1$$

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

2.2.1 Radix-2 방식

Radix-2 알고리즘은 출력 시퀀스 X(k)를 연속해서 작은 시퀀스로 나누어 감으로써 구현된다. 우선 식 (2)와 같이 짝수 주파수 샘플과 홀수 주파수 샘플로 나눈다. 식(2)는 각각 N/2-point DFT에 해당된다.

$$X(2k) = \sum_{n=0}^{N/2-1} (x(n)+x(n+N/2)) W_{N/2}^{nk}$$

$$k=0,1,2,\ldots,N/2-1$$

$$X(2k+1) = \sum_{n=0}^{N/2-1} (x(n)+x(n+N/2)) W_N^n W_{N/2}^{nk}$$

$$k=0,1,2,\ldots,N/2-1 \qquad (2)$$

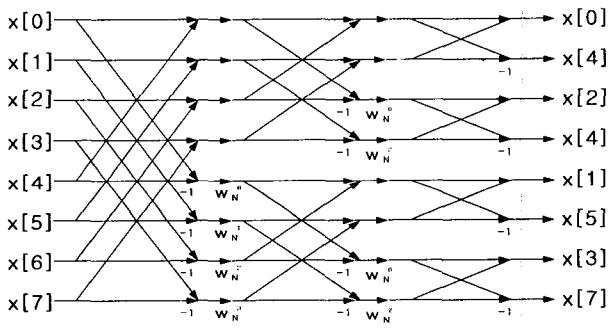


그림 2 8-point Radix-2 FFT의 신호 흐름도

Fig. 2 Signal flow graph of 8-point Radix-2 FFT

각각의 N/2-point DFT는 다시 N/4-point DFT로 나누어 질 수 있으며, 이런 방법으로 8-point DFT는 그림 2와 같이 나타낼 수 있다.

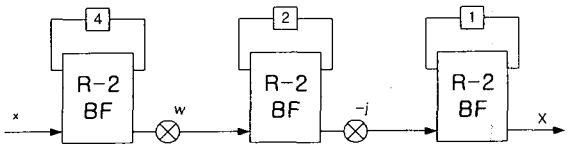


그림 3 8-point Radix-2 FFT 구조

Fig. 3 Structure of 8-point Radix-2 FFT

그림 3은 8-point Radix-2 FFT의 파이프라인 구조로써 첫 번째 단에서는 매 N/2 클럭마다 Mux 신호가 Switching 되고 다음 단에서는 N/4, 그 다음 단에서는 N/8 이런 식으로 Switching 되어진다.

2.2.2 Radix-4 방식

Radix-4 알고리즘도 Radix-2처럼 출력 시퀀스 X(k)를 연속해서 작은 시퀀스로 나누어 감으로써 구현되는데 식 (3)처럼 4단계로 나누어서 계산되어진다. 그림 4는 Radix-2보다 복잡해진 Radix-4 의 butterfly의 연산자의 구조이다.

$$X(4k) = \sum_{n=0}^{N/4-1} (x(n)+x(n+N/4)+x(n+N/2)+x(n+3N/4)) W_N^0 W_{N/4}^{nk}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1} (x(n)-jx(n+N/4)-x(n+N/2)+jx(n+3N/4)) W_N^n W_{N/4}^{nk}$$

$$X(4k+2) = \sum_{n=0}^{N/4-1} (x(n)-x(n+N/4)+x(n+N/2)-x(n+3N/4)) W_N^{2n} W_{N/4}^{nk}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1} (x(n)+jx(n+N/4)-x(n+N/2)-jx(n+3N/4)) W_N^{3n} W_{N/4}^{nk} \qquad (3)$$

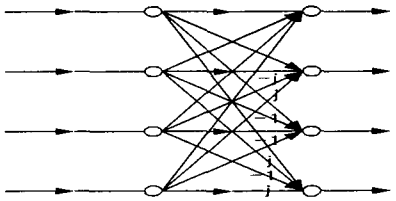


그림 4 Radix-4 버터플라이 구조

Fig. 4 Structure of Radix-4 Butterfly

Radix-4구조는 Radix-2 구조를 Radix-4 로 확장한 구조이다. 이 구조는 Radix-2와 같은 메모리를 사용하나 Radix-2보다 적은 곱셈기가 사용되어 많은 하드웨어를 줄일 수 있다. 그러나 그림 4에서 보는 바와 같이 Radix-2 구조보다 Butterfly 연산부의 구조가 복잡해 제어가 어려운 단점이 있다.

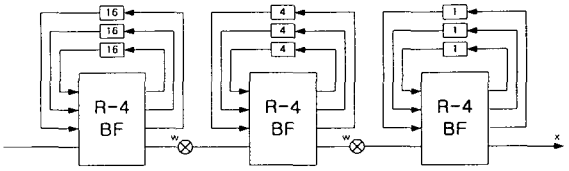


그림 5 64point Radix-4 FFT 구조

Fig. 5 Structure of 64-point Radix-4 FFT

2.2.3 radix-2^2 알고리즘

3차원 인덱스 맵에 의해 인덱스 n과 k를 다음과 같이 분해하고 식 (1)에 대입하면 식 (5)와 같이 나타낼 수 있다.

$$n = \frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3$$

$$0 \le n_1 \le 1,\ 0 \le n_2 \le 1,\ 0 \le n_3 \le \frac{N}{4} - 1$$

$$k = k_1 + 2k_2 + 4k_3$$

$$0 \le k_1 \le 1,\ 0 \le k_2 \le 1,\ 0 \le k_3 \le \frac{N}{4} - 1 \qquad (4)$$

$$X(k) = X(k_1+2k_2+4k_3)$$

$$= \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3) W_N^{(k_1+2k_2+4k_3)(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3)}$$

$$= \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \Big[BF_2(\frac{N}{4}n_2+n_3, k_1) \Big] W_N^{(\frac{N}{4}n_2+n_3)k_1} W_N^{(\frac{N}{4}n_2+n_3)(2k_2+4k_3)} \quad (5)$$

$$BF_2(\frac{N}{4}n_2+n_3, k_1) = \sum_{n_1=0}^{1} x(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3) W_N^{\frac{N}{2}n_1k_1}$$

$$= x(\frac{N}{4}n_2+n_3) + (-1)^{k_1} x(\frac{N}{4}n_2+n_3+\frac{N}{2})$$

윗 식은 radix-2 DIF 버터플라이 연산자에 해당하며, $W_N^{(\frac{N}{4}n_2+n_3)k_1}$ 는 첫단계 분해과정의 트위들 팩터에 해당한다. 이 트위들 팩터를 승산한 후 n_2 대한 연산을 수행하고 이어서 분해과정을 진행한다면, 이는 일반적인 Radix-2 DIF FFT 알고리즘이 된다. Radix-2^2알고리즘은 식 (5)에서의 트위들 팩터를 바로 연산하지 않고, 이 연산을 다음 분해과정 즉, n_2의 연산에서 수행함으로써 Radix-2와 같은 버터플라이 구조를 가지며 복소 곱셈의 수를 줄이는 알고리즘이다. 이를 수식을 통해 유도해 보면 우선 식 (5)의 트위들 팩터는 식 (6)과 같다.

$$W_N^{(\frac{N}{4}n_2+n_3)(k_1+2k_2+4k_3)} = W_N^{Nn_2k_3} W_N^{\frac{N}{4}n_2(k_1+2k_2)} W_N^{n_3(k_1+2k_2)} W_N^{4n_3k_3}$$

$$= (-j)^{n_2(k_1+2k_2)} W_N^{n_3(k_1+2k_2)} W_{\frac{N}{4}}^{n_3k_3} \quad (4)$$

식 (4)을 식 (3)에 대입하면 식 (5)를 구할 수 있다.

$$X(k) = X(k_1+2k_2+4k_3)$$

$$= \sum_{n_3=0}^{\frac{N}{4}-1} \Big\{ \sum_{n_2=0}^{1} \Big[BF_2(\frac{N}{4}n_2+n_3, k_1) \Big] (-j)^{n_2(k_1+2k_2)} \Big\} W_N^{n_3(k_1+2k_2)} W_{\frac{N}{4}}^{n_3k_3}$$

$$= \sum_{n_3=0}^{\frac{N}{4}-1} \Big[H(k_1, k_2, n_3) W_N^{n_3(k_1+2k_2)} \Big] W_{\frac{N}{4}}^{n_3k_3} \quad (5)$$

식 (5)에서 $H(k_1, k_2, n_3)$는 식 (6)과 같다.

$$H(k_1, k_2, n_3) = \sum_{n_2=0}^{1} \Big[BF_2(\frac{N}{4}n_2+n_3, k_1) \Big] (-j)^{n_2(k_1+2k_2)}$$

$$= BF_2(n_3, k_1) + (-j)^{(k_1+2k_2)} BF_2(n_3+\frac{N}{4}, k_1) \quad (6)$$

식 (6)은 2개의 버터플라이 연산으로 구성되어 진다. 따라서 식 (5)는 두 단계의 버터플라이 연산과 N/4-point DFT로 나타내어진다. 그림 6은 16- point Radix-2^2 FFT processor의 신호 흐름도를 보여주고 있다. 그림 6을 보면 첫 번째 stage의 트위들 팩터가 단순 승산인 ($-j$)로 교체됨으로써 복소 승산기의 개수가 줄어들었으며, 버터플라이 연산자의 구조는 일반적인 radix-2 DIF 버터플라이 연산자와 같음을 확인 할 수 있다.

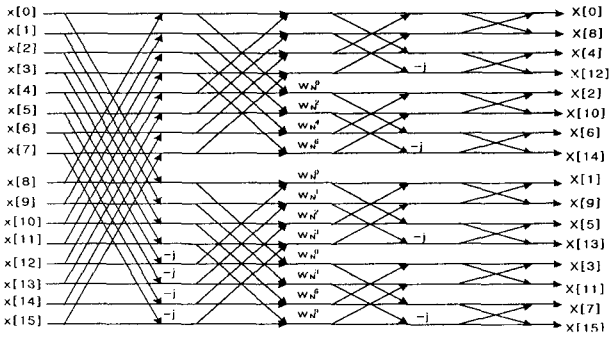


그림 6 16-point radix-2^2 FFT의 신호 흐름도

Fig. 6 Signal flow graph of 16-point Radix-2^2 FFT

Radix-2^2 구조는 기본적으로 radix-2 의 구조와 같다, 이 구조는 Radix-2 DIF FFT 알고리즘에서 Twiddle Factor의 위치를 변형하여 Pipeline화 한 것이다. Radix-2 와 설계 시 비슷하지만 두 종류의 Butterfly의 구조를 사용함으로써 Radix-2 보다 곱셈기의 숫자를 반으로 줄일 수 있고 같은 수 의 곱셈기를 사용하는 Radix-4 구조의 Butterfly 보다 간단한 Radix-2 Butterfly를 사용하여 제어가 용이한 장점이 있다.

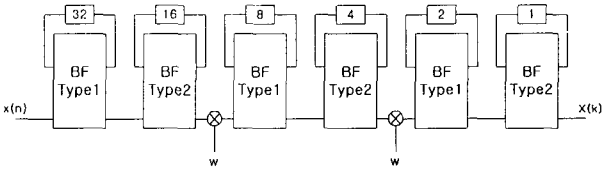


그림 7 64-point Radix-2^2 FFT 구조

Fig. 7 Structure of 64-point Radix-2^2 FFT

따라서 본 논문에서는 복소 곱셈기의 수를 줄이고 속도를 향상시켜 Radix-4 만큼 효율적인 하드웨어를 사용하고 Radix-2구조를 사용하므로 제어도 간단히 할 수 있는 radix-2^2 알고리즘을 사용하였다.

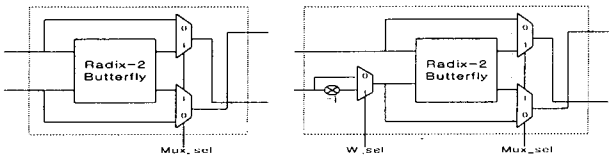
2.3 FFT Processor 의 설계

FFT 프로세서의 구현에 주로 이용되는 수 체계로는 부동 소숫점 방식(floating point)과 고정소숫점 방식(fixed point)이 있다. 본 논문에서는 하드웨어 자원의 효율적인 사용을 위해 고정소숫점 방식을 채택하였으며 고정 소숫점 방식 중 양수와 음수의 표현방식이 가장 명확한 2의 보수법을 사용하였다. 또한 비트 수 결정에 있어서 데이터의 크기를 12비트로 결정하였으며 2의 보수법에 의해 부호 비트를 추가하여 총 13비트로 결정하였다.

2.3.1 Butterfly 설계

Radix-2^2 구조의 Butterfly는 Radix-2 와 설계 시 비슷하지만 두 종류의 Butterfly의 구조를 사용함으로써 Radix-2 보다 곱셈기의 숫자를 반으로 줄일 수 있다

그림 8은 복소수 곱셈 연산이 들어가지 않은 기본적인 두 가지의 Butterfly 구조이다. type 1 은 그림 6에서 stage 2와 stage 4 의 Butterfly로써 Radix-2 Butterfly 의 연산결과를 출력하고 type2는 stage 1와 stage 3 의 Butterfly로써 select 신호에 의해 Twiddle factor -j의 곱셈이 이루어진다.



(a) Butterfly type 1 (b) Butterfly type 2

그림 8 Radix-2^2의 Butterfly 형태

Fig. 8 Type of Radix-2^2 Butterfly

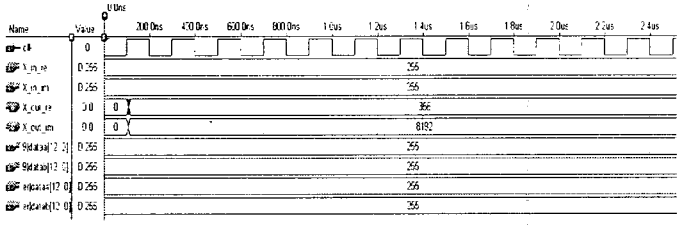


그림 9 Butterfly 시뮬레이션도

Fig. 9 Simulation diagram of Butterfly

2.3.2 복소 곱셈기의 설계

설계된 FFT Processor는 트위들 팩터 W_N^i 를 위한 복소 곱셈기가 필요하다. Processor에서 가장 중요한 모듈인 곱셈기는 하드웨어가 가장 클 뿐만 아니라 성능을 결정하는 중요한 요소이다. FFT에 사용되는 곱셈기는 두 개의 실수와 허수 입력에 대한 복소수 곱셈기이다.

$$e+jf=(a+jb)(c+jd)=(ac-bd)+j(bc+ad)$$

하나의 복소 곱셈기는 위와 같이 4개의 실수 곱셈기와 2개의 실수 가산기가 필요하나 다음 식과 같이 변형시켜 구현하면

$$e+jf=a(c+d)-d(a+b)+j(a(c+d)-c(a-b))$$

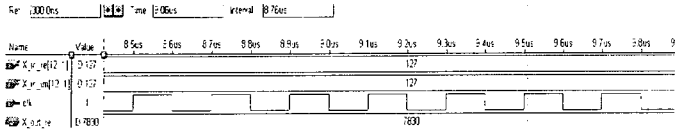


그림 10 복소곱셈기의 시뮬레이션도

Fig. 10 Simulation diagram of Complex multiplier

실수 곱셈기 3개와 가산기 5개가 사용되므로 한 개의 곱셈기를 줄여 하드웨어 크기를 줄일 수 있다. 그림 10은 복소곱셈기의 시뮬레이션도를 나타낸다.

2.3.3 FFT processor의 설계

본 논문에서는 효율적으로 하드웨어를 사용하고 제어도 용이한 Radix-2^2 알고리즘을 적용한 파이프라인 구조로 FFT processor를 설계하였다. 설계는 VHDL을 이용하여 모델링 하고 기능을 검증하였으며 ALTERA사의 Max+PlusⅡ에서 설계하였다. 그림 11은 FFT 프로세서의 시뮬레이션 결과를 보여 주고 있다.

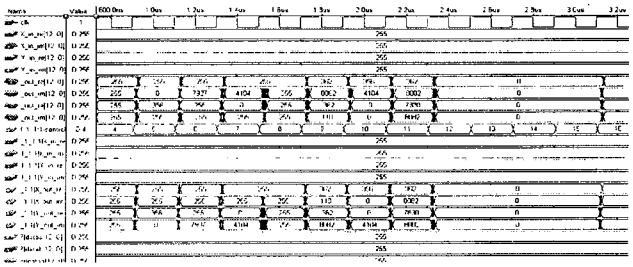


그림 11 FFT processor의 시뮬레이션도

Fig. 11 Simulation diagram of FFT Processor

3. 결 론

본 논문에서는 비 선형적인 특성을 갖는 전력계통에서 전력 품질을 분석하기 FFT processor를 제안하였다. 제안한 시스템은 SOC (System On Chip)을 적용함으로써 범용 마이크로 프로세서와 DSP등에서는 얻을 수 없는 하드웨어 자원의 효율성과 시스템의 최적화 및 데이터 처리 속도를 얻을 수 있었고 Processor 구조는 파이프라인 구조를 택하였으며 3차원 인덱스 분해법에 의한 radix-2^2 알고리즘을 적용함으로써 하드웨어자원에서 가장 많은 비중을 차지하는 복소 곱셈기의 수를 줄이고 효율적인 하드웨어의 사용을 가능하게 하였다.

또한 VHDL로 시스템을 설계함으로써 다른 ASIC 칩과 호환이 가능하며 부가적으로 기존 VHDL 설계를 새로운 응용에 적합한 기술로 합성할 수 있는 장점을 가지고 있다.

참 고 문 헌

[1] G. Bucci and C. Landi, " On Line Digital Measurement for the Quality Analysis of Power Systems under Nonsinusoidal Conditions," IEEE Transactions on Instrumentation and Measurement, Vol. 48, No 4, pp. 853-857, August 1999.

[2] Scott C. Smith M.S.E.E and Dr.Michael J.Devaney " Fourier Based Three Phase Power System," 2000 IEEE, pp. 30-35, 2000.

[3] G. Bucci and C.Landi, " On-line power measurement in nonsinusoidal condition by using A FRLS Algorithm," in Proc. IMEKO-TC4,95 Prague. Czech Republic, Sept. 13-14, 1995.