Fault-Tolerant 하틀리변환 알고리즘

변 산 동* , 조 원 경
경희대학교 전자공학과

Fault-Tolerant Hartley Transform Algorithm

San-Dong Byun* , Won-Kyung Cho
Kyung Hee University,Electronic

ABSTRACT

This paper describes a fault tolerant in Fast Hartley Transform (FHT) using the redundant stage and time. FHT is similar to the Cooley Tukey FFT but performs much faster because it requires only real arithmatic computation requied by the FFT.The FHT uses the real variable as the transform kernel, while the FFT uses the complex exponential as the transform kernel. The Fault Tolerance is concurrent with normal circuit operation and allows a continuousflow of correct data when a fault is occured.

I. 서론

최근에는 VLSI 설계 기술이 발달하고 실시간 신호처리 필요성이 증가함에 따라 FFT 등과 같은 직교변환 알고리즘을 전용 프로세서로 실현하는 연구가 활발히 진행되고 있다. 이러한 연구는 주로 병렬처리(parallel processing) 와 파이프라인 구조, 시스토릭어레이(systoric array)구조로 설계하기 위해 하드웨어 알고리즘이나 직교변환 알고리즘의 연산을 줄이기 위한 것 이다. 그런데, FFT 등과 같은 직교변환에 의해 복소수로 분석, 처리되는 신호들은 시간영역 에서는 실수값만을 갖는다. 따라서, 실수 직교함수를 이용하여 신호처리를 할 수 있다면, 계산과 승산 수를 상당히 줄일 수 있다. 실수 직교함수를 이용한 변환에는 이산 코사인변환(Discrete Cosine Transform), Karhunen Leove 변환등이있는데 주로 영상 데이터의 압축이나, 필터링에 제한적으로 이용되어 왔다. FHT 는 FFT 알고리즘과 유사하며 스펙트럼 분석, 필터링 콘볼루션(Convolution) 등과 같은 모든 FFT 응용분야에 적용될 수 있다. 또한, FHT 로 부터 FFT 와 DCT 를 쉽게 얻을 수 있다. FHT 의 모든 계산과정은 실수로 이루어 지므로 FFT 보다 빠른데다, FFT 와 IFHT(inverse FHT) 는 구조가 동일하며 신호의 흐름도(signal flow graph) 가 매우 규칙적 이므로 VLSI로 설계하는데 용이하다. 그런데 이러한 신호처리 알고리즘인 FFT 나 FHT 를 실현하는 전용 프로세서에서 Fault 의 발생시 심각한 문제가 될 것이다. 그러한 시스템을 실현하고 고비도의 중요한 고 수행성(High Performance)을 갖는 시스템에서 이런 치명적인 에러가 발생 한다면 그 시스템의 동작과 출력된 데이터를 신뢰할 수 없다. 따라서 수행중인 계산된 결과가 정확하다는 것을 보증하기 위해 Fault Tolerant 알고리즘의 개발이 필요하게 되었다. 여기서 Fault Tolerance 는 Fault 가 발생할때 정상적인 동작을 수행하면서 동시에 Fault 를 검출하고, 정확한 데이터를 연속적으로 처리하는것을 말한다. 본 논문에서는 FHT 프로세서에 대해 효과적인 알고리즘 레벨의 Fault 진단법이 제안될 것이다.

FHT 에서도 FFT 와 마찬가지로 분할해 나가는 절점(sequence) 이 입력단 인가 출력단 인가에 따라 DIT(Decimation In Time) 과 DIT (Decimation In Time) 알고리즘으로 구분된다.

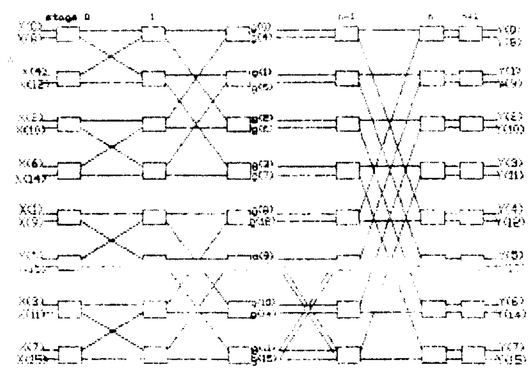

그림 1. 16-point DIT FHT 프로세서

여기서, 각 블럭은 하나의 Radix-2 Butterfly 계

산을 나타내고 있다. 여기서 실수 X 와 Y 는 다음에 의해 변환된다.

$$X_{out} = X_{in} + Y_{in} M^k \text{ ,} \quad Y_{out} = X_{in} - Y_{in} M^k$$

여기서 M^k 는 삼각함수의 합이다. 즉 cos (2πk n/N) + sin (2πkn/N) 이다. 그림에서 각 블럭은 연산모듈(Butterfly)이라 부른다.

II. 폴트 모델(Fault Model)

이 논문에서는 FHT 프로세서가 연산동작중에 발생하는 가능적인 fault 를 고려한다. 프로세서내의 링크보다 연산모듈의 계산 유니트가 더 복잡하기 때문에 fault 는 연산모듈(Butterfly) 계산 유니트에 한정되는 것으로 가정한다.

다음과 같이 가정된 fault 가 이 논문을 통해 사용된다.

- ○ fault 진단 동안 단일한 계속되는 fault.
- ○ fault 검출 동안 단일한 일시적인 fault.
- ○ 한개의 연산모듈에서 복소수 곱셈기, 덧셈기 혹은 뺄셈기에서 발생하는 함수적인 fault.
- ○ 만일 X = X + Y M 이라면 연산 모듈은 fault free.

이 모델은 연산 모듈에서만 fault 를 가정하지만, 링크에서 발생하는 fault 가 일시적으로 발생하는 fault 들 다 연산 모듈에 묶여져서 때문에 여기서 발생하는 fault 도 검출될 수 있다. fault tolerance를 위해 덧붙여진 스위치나 바이패스 링크(bypass link) 와 비교기 같은 부가적인 회로에서는 fault free라고 가정한다. 이러한 점은 그림2 에서 보는것 같이 스위치나 바이패스 링커(bypass linker) 에서 결정적인 동작사이에 쉽게 검출될 수 있다.

이 fault 모델은 이러한 이유에서 발생하는지 잡기 못하는 내부분의 fault 도 검출될 수 있다.

III. Fault Tolerant 프로세서

개요

이장 에서는 양방 통행에 의해 계산되는 N point FHT 프로세서에대한 동시 에러검출 기술이 제안된다. 여기서 a, b, d [a = 1/ 2, b = cos (π/8) , d = sin (π/8)] 는 각 연산모듈에 미리 입력되어 있다고 가정한다. 한 레이블 지연되고 중복된 계산은 FHT 프로세서에서 양방 통행을 이용하여 실행되고 그림1 에있는 FHT 프로세서는 함수식으로 대칭이 아니기 때문에 구조를 그림 2 와 같이 수정하는 것이 필요하다.

FHT 프로세서의 fault tolerant 구조는 그림2 에 보여지고 고장을 진단하기 위해서 한 단계가 더해진다. 각 연산모듈은 셀 바이패스 링크를 갖는다고 가정하고 만일 입력 스위치가 꺼지면 연산모듈로 가는 두개의 입력이 바뀌어지고 출력 스위치가 꺼지면 연산모듈로 부터 나오는 두개의 출력이 바뀌는 입 출력 스위치를 갖는다고 가정한다.

고장을 진단하기 위하여 (n) 단계와 (n + 1) 단계 사이에 비교기가 덧붙여진다. 연산모듈에 대한하드웨어 소비 해드는 O(1 / log (N +1))이다. 원래 FHT 계산은 stage 0 에서 stage n-1 까지 파이프라인 방식을 통해서 수행된다. 쌍방 통행에 의한 재계산은 한단계 늦추어져 원래의 계산을 뒤따른다. 바이패스된 stage n-1 을가지고 stage 0 에서 stage n 까지 파이프 라인을 통해 수행된다.

stage n-1 연산모듈은 함수적 대칭을 만들기위해 입력 데이터 통로를 바꾸어야 한다.

3.1)Fault Tolerant 알고리즘

<Lemma 1>

FHT processor 에서 butterfly 로 가는 두개의 입력중 하나에서 발생한 에러는 butterfly 계산에의해 감추어질수 있다.

< 증명>

덧셈기와 뺄셈기는 에러를 감출수 없기 때문에, 곱셈 유니트만 고려하는 것이 필요하다.

W, Y를 곱셈 유니트에로의 두개의 실수 입력으로 놓자. 여기서 W는 그 유니트에서 실수 이다. 만일 Y 가 에러가 있는 입력 데이타라면, 그것은 $Y = (Y_1 + d_1)$로 다시 표현할수 있고 여기서 d_1 는 에러항이다.

그 때,

$$(WY) = W_1(Y_1 + d_1) \text{ .}$$

만일 W d = 0 이면 에러는 감추어질수 잇다. W 는 영이 아니기 때문에 에러항을 갖는다. 즉, butterfly 로부터 출력 둘다 에러항을 포함한다.

<정 리1 >

Fault torlent FHT 프로세서에서, butterfly 로 들어가는 한개의 입력 에러는 항상 그의 leaf butterfly 에 comparison mismatch 를 낳는다.

< 증 명 >

원래 계산과 재계산은 그림 3 에서 보는 것과 같이 묘사될 수 있다. Lemma 1은 Block A 로부터의 출력은 Block B 로 부터의 출력과 다르다는 것을 의미한다. 여기서 재계산이 원래 계산과 같은 fault 결과를 꼭 같이 생산 하는기

안하는지 알아 보는것이 필요하다. faulty X 의 데이타를 $X' = (X_1 + e_1) + (X_2 + e_2)$ 로 놓고 faulty Y 의 데이타를 $Y' = (Y_1 + d_1) + (Y_2 + d_2)$ 로놓자. 그때 만일 $X' + YW = X + Y'W$ 라면, 그 비교방법은 틀린 것이다.

즉,

$$X_1 + e_1 + (Y_1W_1 - Y_2W_2) + [X_2 + e_2 + (Y_1W_2 + Y_2W_1)] = X_1 + [(Y_1 + d_1)W_1 - (Y_2 + d_2)W_2] + [X_2 \pm 9 (Y_1 + d_1)W_2 + (Y_2 + d_2)W_1)]$$

이것은 $e_1 = \pm(d_1W_1 - d_2W_2)$ 와 $e_2 = \pm(d_1W_2 + d_2W_1)$ 이라는 것을 의미한다. 만일 원래의 계산이 X 에서만 에러를 생산한다면, 그때 e_1 혹은 $e_2 = 0$ 이다. e_1 혹은 e_2가 두개의 값을 가질 수 없기 때문에 이것은 모순이다. 만일 Y 에서 에러를 생산한다면, 그때는 이런 조건을 만족시키기 위하여 $e_1 = e_2 = 0$ 이다. 즉, $d_1W_1 = d_2W_2$와 $d_1W_2 = -d_2W_2$ 는 또다른 모순이다. 따라서 마지막 단계에 있는 그 leaf butterfly 로부터 최소한 하나의 출력은 match 되지 않는다. 그래서 양방 통행에 의한 재계산은 항상 하나의 단일 fault 를 검출한다.

FHT 프로세서에서 butterfly 는 두개의 지수를 사용해 표시한다. B(i,j)는 stage j 에서 i 차 butterfly 를 가리킨다.

< 정 의 1 >

만일 b(i,j)이 B(i,j-1)에 직접 연결되어 있다면 그때 B(i,j-1)은 B(i,j)의 predecessor 라 불리운다. [Pred(B(i,j)]

< 정 의 2 >

B (i,j), $0 \le m < 2^j$ 인 동등한 프래스는 다음과 같은 집합으로 정의 된다.

{B(i,j), Pred(B(i,j)) | Cout[B(i,j) 에서 fault 검샘기]= Cout[B(i,j) 에서 fault 검샘기]}

대등한 class B (j) 는 comparison outcome, Cout 으로 부터 얻어지는 유일한 정보 이다. 즉, comparison mismatch와 근의 predecessor 를 갖는 leaf butterfly 의 근이다.

< 정 의 3 >

FHT 프로세서에서 butterfly 의 쌍,[B(i,j),B(k,j)]는 만일 그들이 stage j-1 에서 predecessor 를 공유 한다면 "buddy" 성질을 갖고 있다고 말한다.

그림 2에서 예를 들면, [B(0,2), B(2,2)]는 그들이 B(0,1)과 B(2,1)을 공유하고 있기 때문에 buddy 성질을 갖고 있다.

< Lemma 2 >

B(i,j)를 $0 \le m < 2^j$ 에 의해 $B_m(j)$ 의 요소로 놓자. 그때 B(i,j)의 buddy 는 fault-free이다.

< 증 명 >

B (j) 는 B(m=2), j-1,2,...,N-1, 만 포함하고 있다. B(i,j)의 buddy 는 $B(i+2^{j-1},j)$ 이기 때문에 이것은 $B_m(j)$ 의 요소인지 아닌지 알수 없다.

< 정 의 4 >

만일 이것이 B(i,j)가 faulty인지 fault-free인지 정확하게 구별할 수 있다면 butterfly , B(k,j)는 butterfly ,B(i,j) 의 식별자라고 부른다 (discriminator) .

< 정 리 2 >

B(i,j)를 검사할 수 있는 FHT 프로세서에서 어떤 $m < 2^j$ 에 대해 $B_m(j)$ 의 요소라 놓자. 그때, log N+3사이클 내에 B(i,j)가 faulty인지 fault-free인지 결정 하는 B(i,j)에 대한 식별자는 항상 존재 한다.

< 증 명 >

만일 j=0 라면, 그때 stage 0 의 모든 butterfly 는 같은 그룹에 있다. stage 1 의 모든 butterfly 는 단일 fault 를 가정 할때 fault-free 이기 때문에 바이패스 된 stage 를 갖는 것에 의해 간단한 식별자(discriminator) 로 사용될 수 있다.

만일 j > 0이면, 그때 Lemma 2 에 의해 B(i,j)의 buddy 는 fault-free이다. B(i,j-1)과 B(i+2 , j-1)의 출력 스위치를 1 로 놓는 것에 의해 같은 입력 데이타는 판별자로 적용될 수 있다. 따라서, satge 0로부터 stage j 로 가는 원래 계산 결과와 한 stage 에 의해 기여된 판별자로 부터 계산된 결과는 바이패스된 stage n 을 통해 stage j+1 을 갖음으로서 얻어질수 있다.

< Corollary 1 >

B(i,j-1)을 어떤 $m < 2^j$에 대해 $B_m(j)$의 요소로 놓자. 그때, fault 덧샘기, 뺄샘기는 두개의 부가적 싸이클내에 장소가 정해질 수 있다.

< 증 명 >

정리2 를 기초로, stage j의 연산모듈이 faulty이거나 stage j-1 의 덧샘기, 뺄샘기가 faulty이다. B(i,j-1)의 판별자를 이용하는 것에의해, faulty덧샘기, 뺄샘기를 갖는 연산모듈은 파이프라인 양식에서 같은 방법을 적용한는 것에 의해 두개의 부가적인 싸이클 내에 장소를 정할 수 있다.

따라서, 어떤 faulty 연산모듈도 log(N+1)+5싸이클 이내에 고정 된다.

IV. 구성

플트를 검출하고 정상적인 동작을 수행하는

시스템은 다음과같이 이어진다. 만일 몰티 연산 모듈이 stage n-1 에서stage n 에 있다면 전체시스템은 아직 동작 중이다. 만일 몰티 연산모듈이 stage 0 와 stage n-2 내에 있다면 [illegible] 보내유지' [illegible] 보시 정상적인 동작을 할 수 있다. 상기 [illegible] 0 에서 stage n-2 까지 [illegible] 중에 결함은 두번 사용될 수 있다. 압축 계산 결과는 완전한 변환을 위해 stage n 에 저장된다. 시간조건(timing condition) 을 만족 시키기 위해, 처음 N/2 입력되는 동안 stage n-1 까지 계산된 결과는 stage n 의 추가되는 싸이클 동안 유지되어야 한다. 이에 필요한 부가적 장치는, 두부분으로 입력이 나누어지는 외부 데이터 분리기, 재 시도 동안 입력데이터를 저장하기 위한 약간의 버퍼메모리 ,그리고 한 싸이클동안 계산을 지연시키기 위한 내부 제어회로이다. 몰티 연산모듈을 갖는 일을 제거함으로 이루어지는 완전한 재구성은 stage n-1 과 stage n 사이에 스위칭 기능을 부과함 으로서 실현된다.

V. 결론

많은 양의 데이터를 고속으로 처리하는 하틀리변환 알고리즘에대하여 알아보고, 이 변환을 Fault 에 대응할 수 있는 알고리즘을 이용한 시스템을 구성해 보았다. 이는 신호처리에서 에러발생에 치명적인 부분에 응용될 수 있을 것이다.

참 고 문 헌

[1]. R. N. Bracewell, "Discrete Hartley Transform" Proc. IEEE, vol. 73, no. 12, Dec. 1983

[2]. Hsieh S. Hou, "The Fast Hartley Transform Algorithm" IEEE,Trans. on Comput, vol. c 36, no. 2,Feb. 1987

[3]. P. Duhamel and M. Vetterli , "Improved Fourier and Hartley Transform Algorithm: Application to cyclic convolution of real data," IEEE Trans. on ASSP, vol. ASSP -35, no.6, June 1987

[4]. Y. H. Choi, and M. Malek, "A Fault Tolerant FFT Processor"IEEE Trans. comput. vol. 37, no. 5, June 1987

[5] . K. H. Huang, and A. Abraham, "Algorithm-Based Fault Tolerance for Matrix OPerations" IEEE Trans. Comput, vol. c-33 no. 6, June 1984

[6] . J. Y. Jou, and J. A. Abraham, " Faullt-Tolerant FFT Nutwork IEEE Trans. Comput, vol. c-31, no. 5, May 1988

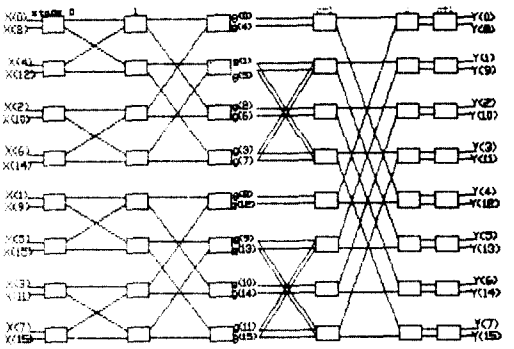

그림 2. Fault Tolerant FHT 프로세서

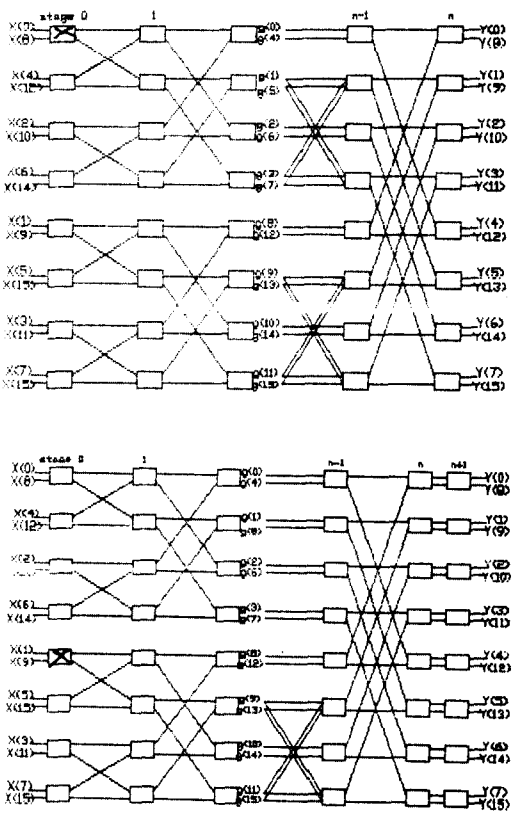

그림 3. Logical View of two computations