멀티프로세서 구조를 이용한 Wave Digital Filter의 구현

김 형 교*

Implementation of Wave Digital Filters Based on Multiprocessor Architecture

Hyeong-Kyo Kim*

이 논문은 2006년도 한신대학교 학술 연구비 지원에 의하여 연구 되었음

요 약

Wave Digital Filter(WDF)는 그 구조상 반올림 오차에 의한 잡음에 아주 강하기 때문에 필터로 구현되는 DSP알고리듬에 있어 그 필터의 계수의 단어길이가 짧을 경우 아주 유용하게 이용될수 있다. 본 논문에서는 멀티프로세서 구조를 채택하여 입력의 샘플링 속도, 프로세서의 수, 그리고 주어진 입력에 대한 출력의 지연에 있어 최적인 WDF를 구현하고자 한다. 이 구현은 제어신호를 포함한 완전한 회로도로 주어지며, 이 화로도는 기존의 실리콘 컴파일러를 이용하여 VLSI 레이아웃으로 용이하게 변환 될 수 있다.

ABSTRACT

The round off noise properties of wave digital filters have known and desirable properties in respect to their realization with short coefficient wordlengths. This paper presents the optimal implementation of wave digital filters by employing multiprocessor archtectures in the sense of input sampling rate, the number of processors, and input-outpt delay. The implementaion will be specified by complete circuit diagrams including control signals, and can be apllied to an existing silicon complier for VLSI layout generation.

키워드

Wave Digital Filter, FSFG, IPB, PB, PDB,Cyclo-static scheduler

Ⅰ. 서 론

고속의 실시간 DSP 알고리듬 구현에 디지털 필터의 사용은 필수적이다. 따라서 응용분야에 따라특성에 맞는 다양한 형태의 디지털 필터가 사용되고 있다. 그 중에서 Fettweis에 의하여 제안된 WDF(Wave Digital Filter)[1]는 IIR 필터의 한 형태인데 오버플로우나 반올림오차 등과 같은 비선형 동작 상태나 필터 계수의 단어길이가 짧을 경우 하에서도 뛰어난 동적특성을 보여 주어 루프 연산에서도 안정성을 제공 할 수 있기 때문에 많이 이용되고 있다.[2]

일반적으로 WDF는 2단자 어댑터와 지연소자 로 구현되는데 연산의 재귀성 때문에 표준의 저전력 VLSI 기술로 구현할 때 높은 데이터 처리율을 이루기 위하여 필요한 파이프라인을 사용할 때 많은 문제점이 발생한다. 그 한 예로 비트 레벨의 systolic array로 구현한 것을 보면 클

* 한신대학교 정보통신학과

접수일자 : 2006. 9. 25

록 주파수가 높더라도 주어진 입력신호를 계산하는데 소요되는 지연시간은 필터계수의 단어길이에 관계가 있거나 지연시간은 일정하더라도 논리 구성이 아주 복잡 해진다[3].

본 논문에서는 멀티프로세서 구조를 채택하여 입력의 샘플링 속도, 프로세서의 수, 그리고 주어진 입력에 대한 출력의 지연에 있어 최적인 WDF를 구현하고자 한다. 우선 WDF를 FSFG(Fully Specified Flow Graph)[4]로 나타낸 한 후 위의 세 가지 최적성 조건을 계산하여 이를 만족하는 멀티프로세서 스케줄을 생성하고, 각 프로세서 사이의 연결을 최소화 하도록 스케줄을 수정한다. 이 수정된 스케줄은 미리 정해진 두 가지 타입의 프로세서를 이용하여 제어신호를 포함한 완전한 회로도로 변환된다. 기존의 실리콘 컴파일러를 이용하면 이 회로도부터 VLSI 레이아웃을 생성할 수 있다. II장에서 멀티프로세서 기반의 필터 설계과정을 기술하고 III장에서는 구현과정을 보였으며 결론은 IV장에 기술하였다.

II. 멀티프로세서 기반의 WDF 설계

2.1. 최적성 조건

그림 1은 4차 WDF를 FSFG로 나타낸 것이다. FSFG는 시 불변 그래프로 각 노드는 타겟 프로세서에서의 최소 연산단위를 나타낸다.

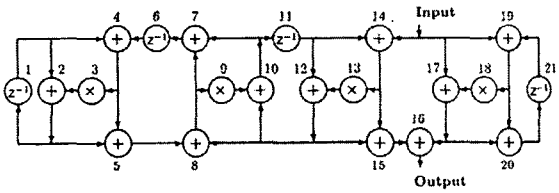

그림 1. 4차 WDF의 FSFG
Fig. 1. FSFG of 4-th Order WDF

일단 FSFG가 주어지면 그것을 멀티프로세서로 구현할 경우 성능을 결정할 세 가지 기본적인 bound가 존재한다. 그것은 Iteration Period Bound(IPB), Processor Bound (PB), 그리고 Periodic Delay Bound(PDB)로서 FSFG의 각 노드의 연산 소요시간만 알면 구할 수 있다.[5]

Iteration Period Bound(IPB): IPB는 알고리듬이 처리 할 수 있는 입력의 최대 속도를 나타내며 다음의 계산식으로 구 할 수 있다.

$$T_0 = \lceil \max \frac{D_l}{n_l} \rceil \qquad l \in \text{루프}$$

$$D_l = \sum_{j \in l} d_j$$

여기서

n_l = 루프 l에서 지연소자의 총 갯수

d_j = 루프 l의 j번째 노드의 연산 소요시간

D_l = 루프 l의 총 연산소요 시간

l = 루프지수

그림 1의 WDF에서 $d_{mmult} = d_{add} = 1$라고 가정하면 IPB는 8이된다.

Processor Bound(PB): PB는 알고리듬을 구현하는 경우 입력이 IPB로 인가될 때 소요되는 최소의 프로세서수를 나타내며 다음과 같이 주어진다.

$$P_0 \geq \lceil \frac{D}{T_0} \rceil$$

여기서

D=모든노드들의 연산소요시간의 총합

T_0=IPB.

그림 1에서 PB는 3이 된다.

Periodic Throughput Delay Bound(PDB):

아래의 식으로 주어지는 PDB는 하나의 입력 샘플이 계산되어 출력에 나타나는데 소요되는 시간을 의미한다.

$$D_0 = \max(D_p - n_p T_0), \qquad p \in i/o$$

여기서

D_p=입출력 경로 p 상의 연산소요시간의 총합

n_p = 경로 p상의 지연소자의 수

T_o= IPB.

그림 1에서 PDB는 5가되며 해당 입출력 경로는 14-13-12-15-16이 된다.

2.2. 멀티프로세서 스케줄의 생성

본 논문에서는 멀티프로세서 스케줄 생성을 위해 Cyclo-Static 스케줄러를 채택하였다. Cyclo-Static 스케줄러는 동기식 멀티프로세서 스케줄러인데 주어진 FSFG로부터 rate optimal, processor optimal, 그리고 delay optimal인 스케줄을 생성한다.[5] 또한 Cyclo-Static 스케줄러는 IPB

에 대하여 주기적이므로 한주기동안의 스케줄 정보만으로 충분하다. 표 1은 그림 1에 보인 4차 WDF의 한주기 동안의 Cyclo-Static 스케줄이다. 여기서 밑수는 노드를 그리고 지수는 현재의 입력 샘플에 대한 상대적인 시간을 나타낸다. 각 프로세서는 승산및 가산 연산을 모두 수행 할 수 있어야 한다.

표 1. 4차 WDF의 Cyclo-static 스케줄
Table. 1 Cyclo-static Schedule of Fourth Order WDF

Pr#1	14^0	13^0	12^0	15^0	16^0			
Pr#2	3^0	2^0	5^0	8^0	9^0	10^0	7^0	4^1
Pr#3		19^0	18^0	17^0	20^0			

2.3. 멀티프로세서 스케줄의 수정

스케줄 수정의 목적은 VLSI로 실현할 경우에 구성하는 프로세서의 구조를 간단히 하고 프로세서간의 연결선의 수를 최소화하기 위한 것으로 Cyclo-static 스케줄은 같은 행의 요소들을 교환하거나 열을 순환(wrap around)시켜도 최적성은 변함이 없이 프로세서간의 연결 구조만을 변경시킨다는 특성을 이용한다.

일반적으로 수정된 스케줄과 본래의 스케줄에 필요한 프로세서의 수가 다를 수 있으나 각 프로세서의 구조가 다르므로 PB는 그대로 보존된다.

FSFG로 부터 생성된 Cyclo-static 스케줄을 변환하는 과정은 다음과 같다. 우선 스케줄의 각 시간에서 필요한 최소의 승산기수 및 가산기수를 구하여 한주기 동안의 IPB에서 최대가 되는 수가 필요한 가산기 및 승산기의 수이다. 표1에서 필요한 승산기의 수는 1,가산기 수는 3이 된다. 각 프로세서는 한 종류만의 연산을 하도록 같은 행의 요소들을 교환하여 스케줄을 재배열한 것이 표2다.

표 2. 4차 WDF의 재배열된 스케줄
Table. 2 Rearranged Schedule of Fourth Order WDF

시간	1	2	3	4	5	6	7	8
Pr#1(승산)	3^0	13^0	18^0	idle	9^0	idle	idle	idle
Pr#2(가산)	14^0	2^0	5^0	15^0	16^0	10^0	7^0	4^1
Pr#3(가산)	idle	19^0	12^0	8^0	20^0	idle	idle	idle
Pr#4(가산)	idle	idle	idle	17^0	idle	idle	idle	idle

다음은 스케줄과 FSFG로부터 연결 표를 만든다. 연결표의 각 요소는 스케줄의 특정 시간에 있어서 각 노드의 연산의 결과가 어디로 연결되는가를 기술한다. 표3에 그림1과 표1로부터 구한 연결표를 보인다 이것을 참조하여 수정된 스케줄의 첫째 열에 재배열된 스케줄의 첫째열로 채운다. 각 시간에서(시간 2에서 시작하여) 승산노드를 프로세서 #1에 할당하고 연결표를 이용하여 부가적인 연결선의 수를 구한다. 어떤노드가 NP개의 선행노드와 NS개의 후행 노드에 연결되어야 한다면 최대 NP+NS개의 연결선이 필요하다. 만약 이 노드를 어떤 프로세서에 할당할 때 NE개의 부가적인 연결선이 필요하다면 실제로 필요한 부가적인 연결선의 수는 NE-NP-NS가 된다. 따라서 그 노드를 각 프로세서에 할당 하여 NE-NP-NS가 최소가 되는 프로세서에 할당한다. 이 과정을 각 시간의 모든 요소들이 채워질 때까지 모든 노드에 대해서 수행한다. 이 과정을 가산노드에 대해서 수행한 후 재배열된 스케줄의 나머지 열에 대해서 수행한다. 상세한 것은 [6]에 기술되어 있다.

표 3. 4차 WDF의 연결표
Table 3. Communications Table of Fourth ORDER WDF

1	2	3	4	5	6	7	8
$3^0 \rightarrow 2^0$	$13^0 \rightarrow 12^0$	$18^0 \rightarrow 17^0$	$15^0 \rightarrow 16^0$	$9^0 \rightarrow 10^0$	$10^0 \rightarrow 7^0$	$7^0 \rightarrow 4^1$	$4^1 \rightarrow 3^1$
$14^0 \rightarrow 13^0$	$2^0 \rightarrow 5^0$	$12^0 \rightarrow 8^0$	$17^0 \rightarrow 16^0$	$16^0 \rightarrow$Out	$10^0 \rightarrow 12^1$		$4^1 \rightarrow 5^1$
$14^0 \rightarrow 15^0$	$2^0 \rightarrow 4^1$	$12^0 \rightarrow 10^0$	$17^0 \rightarrow 20^0$	$20^0 \rightarrow 19^1$	$10^0 \rightarrow 14^1$		
	$2^0 \rightarrow 2^1$	$12^0 \rightarrow 15^0$	$8^0 \rightarrow 7^0$				
	$19^0 \rightarrow 20^0$	$5^0 \rightarrow 8^0$	$8^0 \rightarrow 7^0$				
	$19^0 \rightarrow 18^0$						

표4에 4차 WDF의 수정된 스케줄을 보인다. 이 스케줄을 구현 하려면 1 개의 승산기 및 3개의 가산기, 그리고 7개의 프로세서간의 연결선이 필요하다. 만약 본래의 스케줄을 실현한다면 3 개의 승산기 및 가산기와 12개의 연결선이 필요하게 된다.

표 4. 4차 WDF의 수정된 스케줄
Table 4. Modified Schedule of Fourth Order WDF

시간	1	2	3	4	5	6	7	8
Pr#1(승산)	3^0	13^0	18^0	idle	9^0	idle	idle	idle
Pr#2(가산)	14^0	2^0	12^0	15^0	16^0	10^0	idle	idle
Pr#3(가산)	idle	19^0	5^0	8^0	20^0	idle	7^0	4^1
Pr#4(가산)	idle	idle	idle	17^0	idle	idle	idle	idle

Ⅲ. VLSI 구현을 위한 회로 합성

3.1. 프로세서의 구조

WDF를 FSFG로 표시하면 각 노드의 연산은 는 승산 혹은 가산이므로 이를 실현할 프로세서의 구조는 다음과 같다. 제 I 형 프로세서는 승산기, ROM, 레지스터 및 멀티플렉서로 구성되며 제 II 형 프로세서는 가산기, 레지스터 및 멀티플렉서로 구성된다. 그림2에 이 프로세서의 구조를 보인다.

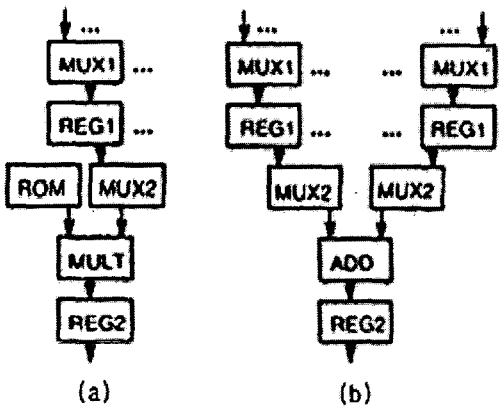

그림 2. 프로세서의 구조 (a)제 I 형 (b) 제 II 형
Fig. 2. Architecture of Processors (a)Type I (b) Type II

3.2. 회로합성

본 절에서는 수정된 스케줄과 FSFG로부터 앞에서 기술한 프로세서를 이용하여 제어신호를 포함한 완전한 회로도를 생성하는 과정을 기술한다. 이렇게 생성된 회로도로부터 실리콘 컴파일러를 이용하여 VLSI 레이아웃을 얻을 수 있다. 더 자세한 것은 [6]에 기술되어 있다.

① 표4와 그림 1로부터 각 프로세서의 형태를 결정한다. 프로세서 1은 제 I형 프로세서를, 그리고 프로세서 2, 3그리고 4는 제 II. 형 프로세서가 된다.

② 스케줄의 첫째열의 첫째 행부터 시작하여 현재 연산의 모든 후행노드를 FSFG를 참조하여 찾아서 현재의 연산결과를 후행 노드의 비어있는 REG1에 저장한다. 만약 후행노드가 두 개의 선행 노드를 필요로 하면 다른 하나의 선행 노드를 찾는다. 그 다른 하나의 선행 노드가 이미 존재하면 현재 연산의 결과를 그 선행 노드연산의 결과가 저장된 그룹의 반대편 REG1에 저장하고 그렇지 않는 경우에는 왼쪽 그룹의 REG1에 저장한다, 이 과정을 현재연산의 모든 후행 노드에 대해서 반복하고 각 데이터의 저장기간을 계산한다,

③ ②의 과정을 현재시간에서 나머지 행의프로세서에 대해서 반복한다.

④ 다음시간으로 가서(스케줄의 다음 열) ②->③의 과정을 반복하되 각 시간에 있어서 프로세서간의 연결구조가 주기적이 될 때까지 계속한다. 모든 경우에 있어서 2*IPB*PB 시간이내에 연결 형태는 주기적이 된다.

⑥ REG1은 하나 이상의 프로세서에 연결될 수 있다. 이 경우 MUX1으로 저장될 데이터를 결정한다.

⑦ 각 프로세서는 두 그룹의 REG1이 있는데 각 그룹에서 둘 이상의 REG1이 있으면 REG1과 연산장치 사이에는 MUX2가 필요하게 된다.

⑧ 각 시간에서의 프로세서 값으로부터 REG1를 로딩하고 MUX1 및 MUX2를 선택하는 제어신호를 발생 한다. 이 제어신호는 주기적인 진리표로 나타낼 수 있다.

그림 3과 표 5는 4차 WDF를 멀티프로세서 기반의 Cyclo-Static 스케줄러를 이용하여 구현한 회로도와 이와 관련된 제어 신호를 보인다. 회로도에서 MUX또는 REG의 입력에 표시된 숫자는 그것이 연결된 프로세서 번호를 나타낸다. 이렇게 생성된 회로도로부터 실리콘 컴파일러를 이용하여 VLSI 레이아웃을 얻을 수 있다.

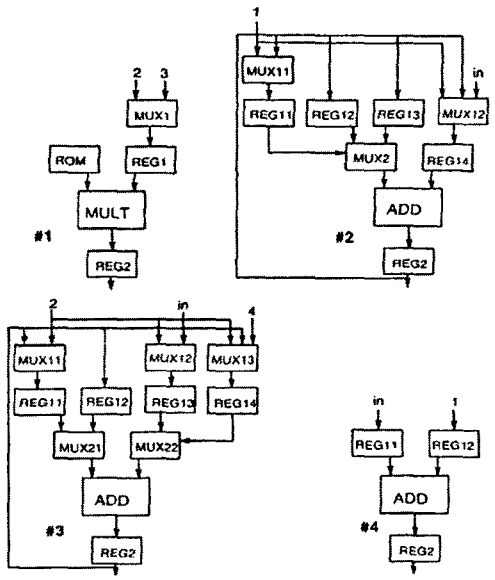

그림 3. 4차 WDF의 회로도
Fig. 3. Circuit Diagram of Fourth Order WDF

표 5. 그림3에 보인 회로도에 대한 제어신호
Table 5. Control Signals for Circuit Diagram shown in Fig.3

시간		1	2	3	4	5	6	7	8
pr #1	R1	1	1	1	X	X	1	X	X
	M1	3	2	3	X	3	X	X	X
pr #2	R11	0	0	0	X	1	1	1	X
	R12	X	1	0	0	X	X	X	X
	R13	0	0	1	0	0	0	0	0
	R14	1	1	1	1	0	0	X	X
	R15	X	X	X	X	1	X	X	X
	M11	X	X	X	X	2	1	2	X
	M12	1	1	2	X	X	X	X	X
	M21	R11	R13	R11	R12	R11	R11	X	X
	M22	R14	R14	R14	R14	R14	R14	X	X
pr #3	R11	1	0	0	1	1	0	0	1
	R12	0	0	1	0	0	1	0	0
	R13	1	0	1	0	0	0	0	0
	R14	X	X	X	1	1	X	1	X
	M11	3	X	X	2	3	X	X	3
	M12	X	X	X	3	4	X	2	X
	M13	in	X	2	X	X	X	X	X
	M21	X	R12	R11	R11	R12	X	R11	R11
	M22	X	R13	R13	R14	R14	X	R14	R13
pr #4	R11	1	0	0	0	X	X	X	X
	R12	X	X	X	1	X	X	X	X

V. 결 론

본 논문에서는 멀티프로세서 구조를 채택하여 입력의 샘플링 속도, 프로세서의 수, 그리고 주어진 입력에 대한 출력의 지연에 있어 최적인 WDF를 구현하였다. 특히 주어진 FSFG로부터 최적성을 수학적으로 엄밀한 방법으로 계산하였기 때문에 리타이밍(retiming), 언폴딩(unfolding)과 같은 입력 알고리듬변환을 행할 필요가 없다. 본 논문에서 제안된 방법으로 얻어진 회로도는 기존의 실리콘 컴파일러를 이용하면 VLSI 레이아웃을 용이하게 생성할 수 있다.

참고문헌

[1] A. Fettweis, "Some Principles of designing digital filters imitating classical filter structure," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 314-316, Mar. 1971.

[2] S. S. Lawson and A. R. Mirzai, *Wave Digital Filters*, Ellis-Horwood, 1990.

[3] S. S. lawson and S. Summerfield, "The design of wave digital filters using fully pipelined bit-level systolic arrays," *J. VLSI Signal Processing*, vol2 ,pp. 49-62,190

[4] D.A.Schwartz and T.P.Barnwell, "Cyclo- Static Solutions:Optimal Multiprocessor Realization of Recursive Algorithms," in S.Y.Kung, et al. (Eds.), *VLSI Signal Processing II*, IEEE Press, pp. 11-128, 1986.

[5] M. REnfors and Y. Neuvo, "The Maximum Sampling Rate of Digital Filters Under Hardware Speed Constraints," *IEEE Trans. on Circuits and Systems*, pp.196-202, 1981.

[6] 김형교, "최적 멀티프로세서 스케줄러를 이용한 재귀 DSP 알고리듬의 구현", 한국해양정보통신학회 논문지, 제10권 2호,pp 228-234, 2006,2.

저자소개

김 형 교(Hyeong-Kyo Kim)

1978년 2월 서울 대학교 전기공학과 공학사

1980년 2월서울 대학교 전자공학과 공학석사

1993년 3월 Georgia Institute Technology, School of Electrical Eng. Ph.D.

1993년 7월 ~ 1995년 2월 한국전자통신 연구원 선임연구원

1995년 3월 ~ 1997년 3월 상명대학교 정보과학과 전임강사

1997년 3월 ~ 현재 한신대학교 정보통신학과 부교수

※관심분야 : DSP, VLSI Signal Procressing, System Identification