최적 멀티프로세서 스케줄러를 이용한 재귀 DSP 알고리듬의 구현

김형교*

Implementation of Recursive DSP Algorithms Based on an Optimal Multiprocessor Scheduler

Hyeong-Kyo Kim*

이 논문은 2005년도 한신대학교 학술 연구비 지원에 의하여 연구 되었음

요 약

본 논문은 주어진 재귀 DSP 알고리듬으로부터 최적멀티프로세서 스케줄러를 이용하여 완전한 회로도를 효과적으로 생성할 수 있는 체계적인 과정에 대하여 기술한다. 이과정은 크게 스케줄 생성 단계와 회로도 생성 단계로 구성된다. 스케줄 생성단계는 입력으로서 Fully Specified Flow Graph(FSFG)로 표현된 재귀 DSP 알고리듬을 받아서 최적 멀티프로세서 스케줄러를 생성하며 회로도 생성 단계에서는 이 스케줄러로부터 제어신호를 포함한 완전한 회로도를 생성한다. 이 회로도는 실리콘 컴파일러를 이용하여 VLSI 레이아웃으로 용이하게 변환될 수 있다. 본 논문에서는 2차 Gray-Markel Lattice 필터를 예로 사용하여 전체적인 구현과정을 보인다.

ABSTRACT

This paper describes a systematic process which can generate a complete circuit specification efficiently for a given recursive DSP algorithm based on an optimal multiprocessor scheduler. The process is composed of two states: scheduling and circuit synthesis. The scheduling part accepts a fully specified flow graph(FSFG) as an input, and generates an optimal synchronous multiprocessor schedule. Then the circuit synthesis part translates the modified schedule into a complete circuit diagram including a control specification. The circuit diagram can be applied to a silicon compiler for VLSI layout generation. This paper illustrates the whole process with an example of a second order Gray-Markel lattice filter.

키워드

DSP, design synthesis system, multiprocessor scheduler, FSFG, Silicon compiler, Gray-Markel Lattice filter

Ⅰ. 서 론

VSLI 기술의 발달에 힘입어 많은 계산량을 요구하는 DSP 알고리듬을 ASIC으로 구현하기가 용이해졌다. 주어진 상위수준 입력으로부터 ASIC의 레이아웃을 자동적으로 생성하는 과정을 디자인 합성 과정(design synthesis process) 라고하며 특정한 응용분야에 디자인 합성과정을 적용하기위해 여러 종류의 디자인 합성 시스템들이 개발되었다.[1] 대부분의 디자인 합성 시스템은 크게 스케줄 생성 모듈, 자원 할당 모듈, 그리고 실리콘 컴파일레이

* 한신대학교 정보통신학과

접수일자 : 2005. 11. 22

션(silicon compilation) 모듈과 같이 세 모듈로 구성되어있다. 스케줄 생성 모듈은 주어진 입력 알고리듬을 가장 빠른 동작속도와 최소한의 자원만을 사용하여 구현 할 수 있도록 최적의 조건을 계산하여 이를 기반으로 한 스케줄을 생성한다. 또한 자원 할당 모듈은 생성된 스케줄을 제어신호를 포함한 완전한 회로도로 바꾸고 실리콘 컴파일레이션 모듈은 이것으로부터 VLSI 레이아웃을 생성한다. 기존의 시스템에서는 최적의 스케줄을 생성하기 위해 direct graph mapping, 리타이밍, loop unrolling 그리고 소프트웨어 파이프라인 기법 등을 사용하여 주어진 입력 알고리듬을 변환하기도 하는데 이는 물론 최종 실현에 어느 정도의 개선을 가져오지만 그 어느 것도 수학적으로 엄밀한 최적성 기준을 제공하지 못하고 있다.[2]

따라서 본 논문에서는 주어진 재귀DSP 알고리듬을 엄밀한 최적성 기준을 만족하는 멀티프로세서 스케줄러를 이용하여 효과적으로 구현할 수 있는 체계적인 방법을 기술한다. 입력으로서 재귀 DSP 알고리듬이 주어지면 세 가지 최적성 조건을 계산하여 이를 만족하는 멀티프로세서 스케줄을 생성한 후, 각 프로세서 사이의 연결을 최소화 하도록 스케줄을 수정한다. 이 수정된 스케줄은 미리 정해진 세 가지 타입의 프로세서를 이용하여 제어신호를 포함한 완전한 회로도로 변환된다. 실리콘 컴파일러를 이용하면 이 회로도부터 VLSI 레이아웃을 생성할 수 있다. II장에서는 구현방법의 전체적인 개괄을 보이며 III장은 스케줄에 대해서 기술한다. 생성된 스케줄을 이용하여 회로를 합성하는 방법을 IV장에 제안하였으며 결론은 V장에 기술하였다.

II. 개괄도

그림 1은 본 논문에서 제안된 방법의 개괄도를 보인다. 구현방법은 크게 스케줄생성 단계와 회로합성의 두 단계로 구성된다. 다른 시스템과는 달리 스케줄 생성 단계에서는 Fully Specified Flow Graph(FSFG)로 표현된 입력 알고리듬으로부터 세 가지 최적성 조건을 계산한다. 여기서 세 가지 최적성 조건이란 rate optimal, processor optimal, 그리고 delay optimal을 말한다.[3],[4] rate optimal은 주어진 알고리듬이 계산할 수 있는 입력신호의 최대 속도를, processor optimal은 알고리듬을 구현하는데 필요한 최소의 프로세서의 수를, 그리고 delay optimal은 주어진 입력과 그에 대응하는 출력을 계산하는데 소요되는 최소시간을 의미한다. 따라서 이 세 가지 최적성 조건을 만족하여 구현된 시스템은 가장 빠른 입력 속도, 최소의 프로세서 의수 그리고 최소의 입출력 지연을 보장하게 된다. 스케줄 생성 단계에서는 이 세가지 최적성 조건을 이용하여 최적 멀티 프로세서 스케줄을 생성하고 각 프로세서간의 연결을 최소화하기 위해 스케줄을 수정한다. 회로합성 단계에서는 이 수정된 스케줄을 제어신호를 포함한 완전한 회로도로 변환 한다. 이렇게 생성된 회로도로부터 실리콘 컴파일러를 이용하여 VLSI 레이아웃을 얻을 수 있다.

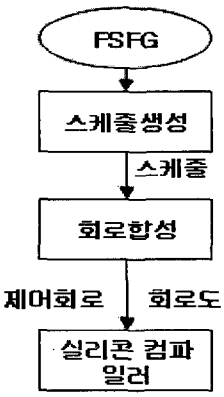

그림 1. 전체적인 개괄도
Fig. 1. System Overview

III. 스케줄 생성

앞장에서 기술한바와 같이 스케줄 생성 단계에서는 주어진 입력 알고리듬으로부터 최적성 조건을 계산한 후 이 조건들을 만족하는 최적 다중 처리 장치 스케줄러를 생성하여 처리장치간의 연결을 최소화 하기 위해 수정한다. 이절에서는 스케줄 생성에 대해서 상술한다.

3.1 입력 알고리듬 표현

스케줄 생성 단계의 입력 알고리듬은 FSFG로 표현된다. FSFG는 시 불변 그래프로 각 노드는 타겟 프로세서에서의 최소 연산단위를 나타낸다. 그림 2는 2차 Gray-Markel Lattice 필터의 FSFG인데 여기서 최소 연산단위는 덧셈과 곱셈으로 주어졌지만 일반적으로 어떠한 연산단위도 나타낼 수 있다.

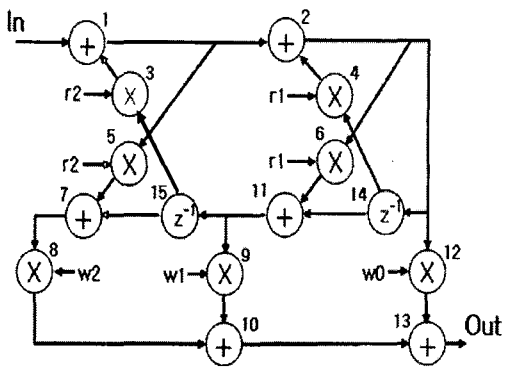

그림 2. 2차 Gray-Markel 필터의 FSFG
Fig. 2. FSFG of Second Order Gray-Markel Lattice Filter

3.2 최적성 조건

일단FSFG가 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. IPB는 알고리듬이 처리 할 수 있는 입력의 최대 속도를, PB는 알고리듬을 구현하는 경우 입력이 IPB로 인가될 때 소요되는 최소의 프로세서수를 나타내며 PDB는 하나의 입력 샘플이 계산되어 출력에 나타나는데 소요되는 시간을 의미한다. 주어진 알고리듬을 구현할 때 IPB, PB, PDB를 만족하면 rate optmal, processor optimal, delay optimal이라 한다. 예를 들어 그림2의 경우 각 노드의 계산 소요시간을 각각 1이라고 가정하면 IPB, PB 그리고 PDB는 각각 5,3,7이된다.

3.3 스케줄 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 IPB에 대하여 주기적이므로 한주기동안의 스케줄 정보만으로 충분하다. 표 1은 그림 2에 보인 2차 Gray-Markel Lattice 필터의 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. 2차 Gray-Markel 필터의Cyclo-static 스케줄
Table 1. Cyclo-static Schedule of Second Order Gray-Markel Lattice Filter

시간	1	2	3	4	5
Pr #1	4^0	2^0	6^0	8^0	3^1
Pr #2	1^0	5^0	7^0	11^{-1}	9^0
Pr #3	10^{-1}	13^{-1}	12^0	idle	idle

3.4 스케줄 변환

스케줄 변환의 목적은 VLSI로 실현할 경우에 구성하는 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 보존되는 셈이다.

생성된 Cyclo-static 스케줄을 변환하는 과정을 다음 각 단계에서 기술한다.

① 스케줄의 각 시간에서 승산기수와 가산기수를 NM_i, NA_i라고 하면 스케줄구현에 필요한 최소의 승산기수와 가산기수는 다음과 같이 주어진다.

$$PNM = \max_{i \in IPB}(NM_i)$$

$$PNA = \max_{i \in IPB}(NA_i)$$

표1의 스케줄의 경우, $PNM = 2$, $PNA = 2$가 된다.

② 각 프로세서는 한 종류만의 연산을 하도록 같은 행의 요소들을 교환하여 스케줄을 재배열한다. 표2에 재배열된 스케줄을 보인다.

표 2. 2차 Gray-Markel 필터의 재배열된 스케줄
Table 2. Rearranged Schedule of Second Order Gray-Markel Lattice Filter

시간	1	2	3	4	5
Pr #1(승산)	4^0	5^0	6^0	8^0	3^1
Pr #2(승산)	idle	idle	12^0	idle	9^0
Pr #3(가산)	10^{-1}	2^0	idle	idle	idle
Pr #4(가산)	1^0	13^{-1}	7^0	11^{-1}	idle

③ 스케줄과 FSFG로부터 연결 표를 만든다. 연결표의 각 요소는 스케줄의 특정 시간에 있어서 각 노드의 연산의 결과가 어디로 연결되는가를 기술한다. 표3에 그림2와 표1로부터 구한 연결표를 보인다.

④ 수정된 스케줄의 첫째 열에 재배열된 스케줄의 첫째열로 채운다.

⑤ 각 시간에서(시간 2에서 시작하여) 승산노드를 프로세서 #1에 할당하고 연결표를 이용하여 부가적인 연결선의 수를 구한다. 이과정을 나머지 승산기에 대해서 계속한다. 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. 계산 복잡도는 $O(PNM^2)$가 된다.

⑥ ⑤의 과정을 승산노드에 대해서 수행한다. 계산 복잡도는 $O(PNA^2)$가 된다.

⑦ ⑤와 ⑥의 과정을 재배열된 스케줄의 나머지 열에 대해서 수행한다. 따라서 전 과정의계산복잡도는 $O(IPB^*(PNM^2+PNA^2))$ 가 되며 이는 계산 가능한 범위이다.

표3. 연결표
Table 3. Communications Table

1	2	3	4	5
$1^0 \rightarrow 2^0$	$2^0 \rightarrow 6^0$	$6^0 \rightarrow 11^1$	$11^{-1} \rightarrow 9^{-1}$	$3^1 \rightarrow 1^1$
$1^0 \rightarrow 5^0$	$2^0 \rightarrow 12^0$	$12^0 \rightarrow 13^0$	$11^{-1} \rightarrow 3^0$	$9^0 \rightarrow 10^0$
$4^0 \rightarrow 2^0$	$2^0 \rightarrow 4^1$	$7^0 \rightarrow 8^0$	$11^{-1} \rightarrow 7^0$	
$10^{-1} \rightarrow 13^{-1}$	$2^0 \rightarrow 11^1$		$8^0 \rightarrow 10^0$	
	$5^0 \rightarrow 7^0$			
	$13^{-1} \rightarrow$ Out			

표 4. 2차 Gray-Markel 필터의 변환된 스케줄
Table 4. Modified Schedule of Second Order Gray-Markel Lattice Filter

시간	1	2	3	4	5
Pr #1(승산)	4^0	5^0	6^0	idle	3^1
Pr #2(승산)	idle	idle	12^0	8^0	9^0
Pr #3(가산)	10^{-1}	13^{-1}	idle	idle	idle
Pr #4(가산)	1^0	2^0	7^0	11^{-1}	idle

표4에 2차 Gray-Markel Lattice 필터의 변환된 스케줄을 보인다. 이 스케줄을 구현 하려면 2 개의 승산기 및 가산기와 4 개의 프로세서간의 연결선이 필요하다. 만약 본래의 스케줄을 실현한다면 3 개의 승산기 및 가산기와 12개의 연결선이 필요하게 된다.

Ⅳ. 회로합성

회로합성단계에서는 수정된 스케줄을 제어신호를 포함하여 평션 장치와 저장 장치로 구성된 회로도로 변환한다. 이 회로도는 실리콘 컴파일러의 입력으로 사용된다.

4.1 타깃(target) 프로세서의 구조

일반적으로 디자인 합성과정에 있어서 입력 알고리듬을 나타내는 FSFG의 각 노드는 다양한 연산을 나타낼 수 있으나 대부분의 DSP 알고리듬의 연산은 승산, 가산 및 지연으로 대표될 수 있으므로 본 논문에서는 각 연산노드를 승산 혹은 가산으로 제한하여 이를 실현할 타깃 프로세서의 구조를 다음 세 가지 형태로 제안한다. 제1형의 프로세서는 승산기, ROM, 레지스터 및 멀티플렉서로 구성되며 제2형의 프로세서는 승산기, 레지스터 및 멀티플렉서로 그리고 제3형의 프로세서는 가산기, 레지스터 및 멀티플렉서로 구성된다. 그림3에 이 세 가지 형태의 타깃 프로세서의 구조를 보인다.

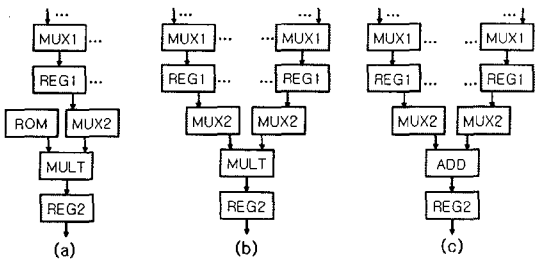

그림3.타깃 프로세서의 구조 (a) 제 1형 (b) 제 2형 (c) 제3형

Fig. 3 Prototypes of Processors (a) Type I (b) Type II (c) Type III

본 논문에서는 구현될 스케줄에 사용될 프로세서는 한 가지 형태의 연산만을 행하도록 변환되었으므로 각 프로세서는 한 가지 형태의 연산소자만을 가진다. REG1은 프로세서의 연산의 입력데이터들을 저장한다. REG1의 수는 초기에 무한대로 설정하고 디자인 최종과정에서 필요한 수만큼만 남게 될 것이다. REG2는 연산 결과를 저장하게 되며 각 프로세서에서 단 하나만 필요하다. ROM은 입력 알고리듬에 따라 미리 정해진 계수들을 저장하며 MUX1은 REG1에 저장될 값을 결정한다. MUX2는 현재 프로세서에서 수행할 연산에 사용될 데이터를 저장하고 있는 REG1를 선택한다.

대부분의 재귀형 DSP 알고리듬은 적응형 과 비적응형 알고리듬으로 나눌 수 있는데 적응형 알고리듬에서 승산 연산이 수행될 경우 그 데이터들은 다른 연산의 출력으로부터 나오므로 ROM은 필요하지 않게 된다. 따라서 적응형 알고리듬에는 제 1형 및 제3형의 프로세서가, 그리고 비적응형 알고리듬에는 제2형 및 제3형의 프로세서가 이용된다.

제어신호는 레지스터를 로딩하고 멀티플렉서를 선택하기 위한 신호인데 FSM(finite state machine)으로 기술된다.

앞에서 논의한 바와 같이 일단 재귀형 알고리듬이 FSFG로 표시되면 기본적인 연산의 종류에 관계없이 Cyclo-static 스케줄을 구할 수 있다. 현재까지 대부분 DSP 알고리듬에 이용되는 연산은 가산 및 승산으로 제한되어 있어 본 논문에 채택된 프로세서의 형태도 이에 따랐다.

4.2 하드웨어 할당

하드웨어 할당 단계에서는 수정된 스케줄과 FSFG로부터 앞에서 기술한 프로세서를 이용하여 제어신호를 포함한 완전한 회로도를 생성한다. 이렇게 생성된 회로도로부터 실리콘 컴파일러를 이용하여 VLSI 레이아웃을 얻을 수 있다. 본 절에서는 이 과정을 각 단계에서 기술한다.

프로세서 형태 결정

① 수정된 스케줄의 첫째 열과 FSFG로부터 각 프로세서의 형태를 결정한다. (승산기 혹은 가산기를 가진 프로세서) 표2의 예에서 프로세서 1과 2는 승산기를, 그리고 프로세서 3과 4는 가산기를 가진다.

승산기 형태 결정

② 앞절에서 기술한바와 같이 주어진 알고리듬이 적응형 이라면 승산기의 양쪽입력은 다른 프로세서의 출력으로부터 받으므로 제 2형의 프로세서가 채택될 것이며 비적응형 이라면 제 1형의 프로세서가 채택된다.

각 프로세서의 연결 및 REG1 할당

③ 스케줄의 첫째열의 첫째 행부터 시작하여 현재 연산의 모든 후행노드를 FSFG를 참조하여 찾는다. 각 후행노드에 대해 어떤 형태의 프로세서가 사용될 것인지 결정한다. 만약 후행노드가 하나의 선행노드를 필요로 한다면(이 경우 다른 하나의 입력은 ROM에 연결되어있다.) 현재의 연산결과를 후행 노드의 비어있는 REG1에 저장한다. 만약 후행노드가 두 개의 선행 노드를 필요로 하면 다른 하나의 선행노드를 찾는다. 그 다른 하나의 선행 노드가 이미 존재하면 현재 연산의 결과를 그 향 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,

④ ③의 과정을 현재시간에서 나머지 행의프로세서에 대해서 반복한다.

⑤ 다음시간으로 가서(스케줄의 다음 열) ③ ->④의 과정을 반복하되 각 시간에 있어서 프로세서간의 연결구조가 주기적이 될 때까지 계속한다. 모든 경우에 있어서 2*IPB*PB 시간이내에 연결 형태는 주기적이 된다.

멀티플렉서 (MUX1 및 MUX2)의 할당

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

⑦ 각 프로세서는 두 그룹의 REG1이 있는데 각 그룹에

서 둘 이상의 REG1이 있으면 REG1과 연산장치 사이에는 MUX2가 필요하게 된다.

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

그림 4와 표 5는 2차 Gray-Markel Lattice 필터를 이상과 같이 Cyclo-Static 스케줄러를 이용하여 구현 한 회로도와 이와 관련된 제어 신호를 보인다. 이렇게 생성된 회로도로부터 실리콘 컴파일러를 이용하여 VLSI 레이아웃을 얻을 수 있다.

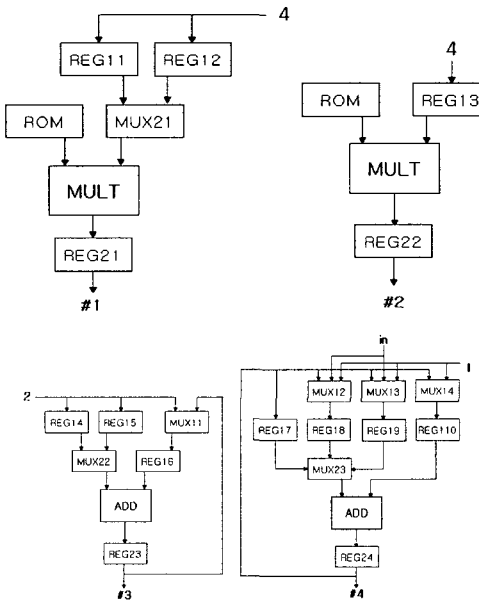

그림 4. 2차 Gray-Markel Lattice 필터의 회로도
Fig. 4. Circuit Diagram of Schedule of Second Order Gray-Markel Lattice Filter

표5. 그림4에 보인 회로도에 대한 제어신호
Table 5. Control Signals for Circuit Diagram shown in Fig.4

시간	1	2	3	4	5	6	7	8	9	10
R11	0	1	1	0	0	0	1	1	0	0
R12	X	X	X	X	1	X	X	X	X	1
R13	X	X	X	1	1	X	X	X	1	1
R14	0	0	X	1	0	0	0	X	1	0
R15	0	X	X	X	1	0	X	X	X	1
R16	1	1	X	X	X	1	1	X	X	X
R17	0	0	0	X	1	0	0	0	X	1
R18	1	1	1	0	0	0	0	0	0	X
R19	0	0	0	0	X	1	1	1	0	0
R110	1	1	1	1	X	1	1	1	1	X
M 11	2	3	X	X	X	2	3	X	X	X
M 12	in	1	4	X	X	X	X	X	X	X
M 13	X	X	X	X	X	in	1	4	X	X
M 14	1	4	1	1	X	1	4	1	1	X
M 21	R11	R11	R11	X	R12	R11	R11	R11	X	R12
M 22	R15	R14	X	X	X	R15	R14	X	X	X
M 23	R18	R18	R17	R19	X	R19	R19	R17	R18	X

V. 결 론

본 논문에서는 주어진 재귀DSP 알고리듬을 최적멀티프로세서 스케줄러를 이용하여 효과적으로 구현할 수 있는 체계적인 방법을 기술하였으며 2차 Gray-Markel Lattice 필터를 예를 들어 보였다. 특히 최적성을 수학적으로 엄밀한 방법으로 계산하였기 때문에 리타이밍 방법과 같은 입력 알고리듬변환을 행할 필요가 없다. 본 논문에서 제안된 방법으로 얻어진 회로도는 실리콘컴파일러를 이용하면 VLSI 레이아웃을 생성할 수 있다.

참고문헌

[1] V. Madisetti and C. Arpnikanondt, *A Platform-Centric Approach to System-On-Chip(SOC) Design*, Springer, 2005

[2] S.Y. Kung, *VLSI Array Processor*, Prentice-Hall, 1988

[3] A.Fettweis,"Realizability of Digital Filter Networks," *Arch..Elek. Ubertragung*, Vol. 30, pp.90-96, Feb. 1976.

[4] 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.

[5] 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.

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