Cyclo-static 스케줄러를 이용한 재귀형 LMS Filter의 VLSI 구조

VLSI Architecture of a Recursive LMS Filter Based on a Cyclo-static Scheduler

김 형 교*

Hyeong-Kyo Kim*

요 약

본 논문에서는 적응 필터링 분야에서 널리 쓰이고 있는 재귀형 LMS 필터의 고속연산을 위해 Cyclo-static 스케줄러를 이용하여 VLSI구현에 적합한 구조를 제안한다. 이과정은 크게 스케줄 생성 단계와 회로도 생성 단계로 구성되는데, 스케줄 생성단계는 입력으로서 Fully Specified Flow Graph(FSFG)로 표현된 재귀 DSP 알고리듬을 취하여 입력의 샘플링 속도, 프로세서의 수, 그리고 주어진 입력에 대한 출력의 지연에 있어 최적인 Cyclo-static 스케줄러를 생성하여 각 프로세서간의 연결선이 최소가 되도록 스케줄을 변환한다. 회로도 생성 단계에서는 이 변환된 스케줄러로부터 미리 정의된 두 가지 형태의 프로세서 구조를 이용하여 그것을 구성하고 있는 레지스터 및 멀티플랙서의 할당을 행하고 제어신호를 포함한 완전한 회로도를 생성한다, 이렇게 생성된 회로도는 기존의 실리콘 컴파일러를 이용하여 VLSI 레이아웃으로 용이하게 변환 될 수 있다.

Abstract

In this paper, we propose a VLSI architecture of an LMS filter based on a Cyclo-static scheduler for fast computation of LMS filteing algorithm which is widely used in adptive filtering area. This process is composed of two steps: scheduling and circuit synthesis. The scheduling step accepts a fully specified flow graph(FSFG) as an input, and generates an optimal Cyclo-static schedule in the sense of the sampling rate, the number of processors, and the input-output delay. Then the generated schedule is transformed so that the number of communication edges between the processors. The circuit synthesis part translates the modified schedule into a complete circuit diagram by performing resource allocations. The VLSI layout generation can be performed easily by an existing silicon compiler.

Keywords : LMS Filter, FSFG, IPB, PB, PDB, Cyclo-static scheduler

I. 서 론

Wdrow와 Hoff에 의해 제안된 LMS필터는 구조 및 계산의 간편성과 수렴의 안정성으로 인해 디지털 신호처리 및 통신 분야에서 사용되고 있는 적응 필터의 구현에 널리 이용되고 있다. LMS 필터는 재귀형(recusrvie) 필터와 비 재귀형(non-recursive) 필터로 나누어 질 수 있는데 재귀형 LMS 필터는 비 적응형 필터에서와 마찬가지로 조건에 따라 안정성을 보장하지 못하여 필터계수의 민감도나 수렴시간에 영향을 줄 수도 있지만 반면에 같은 차수의 비 재귀형 필터에 비해 더 나은 성능을 제공 할 수 있다. [1]

본 논문에서는 LMS 필터에 인가되는 입력의 샘플링 속도, 필터를 구현하는데 필요한 프로세서의 수, 그리고 주어진 입력에 대한 출력의 지연과 같은 세 가지 조건에 대하여 최적을 보장 할 수 있는 LMS 필터의 VLSI구현에 적합한 구조를 제안한다. 우선 LMS 필터를 FSFG(Fully Specified Flow Graph]로 나타낸 후 위의 세 가지 최적성 조건을 계산하여 이를 만족하는 Cyclo-static 스케줄[2]을

*한신대학교 정보통신학과
논문 번호 : 2006-11-7 접수 일자 : 2006. 4. 16
심사 완료 : 2007. 1. 22
※이 논문은 2006년도 한신대학교 학술 연구비 지원에 의하여 연구 되었음

생성하고, 각 프로세서 사이의 연결을 최소화 하도록 스케줄을 수정한다. 이 수정된 스케줄은 미리 정해진 두 가지 타입의 프로세서를 이용하여 제어신호를 포함한 완전한 회로도로 변환되며 기존의 실리콘 컴파일러를 이용하면 이 회로도부터 VLSI 레이아웃을 생성할 수 있다. II장에서 LMS 필터의 그래프로부터 최적성을 계산하여 이를 만족하는 Cyclo-static 스케줄 생성과정을 기술하고 III장에서는 스케줄로부터 VLSI 구현을 위해 회로도로 변환하는 과정을 보였으며 결론은 IV장에 기술하였다.

II. LMS 필터의 Cyclo-static 스케줄

2.1 LMS 필터의 FSFG

그림 1은 입력과 출력의 관계식이 다음과 같이 주어지는 2차 LMS 필터의 FSFG를 보인다.[3]

$$H[n+1]=H[n]+\mu X[n]e[n]$$
$$e[n]=d[n]-X^T[n]H[n]$$

여기서

$X[n]$ 입력신호의 벡터;
$H[n]$ 필터 계수의 벡터;
μ 스텝 크기;
$e[n]$ 에러신호 샘플;
$d[n]$ 원하는 신호 샘플.

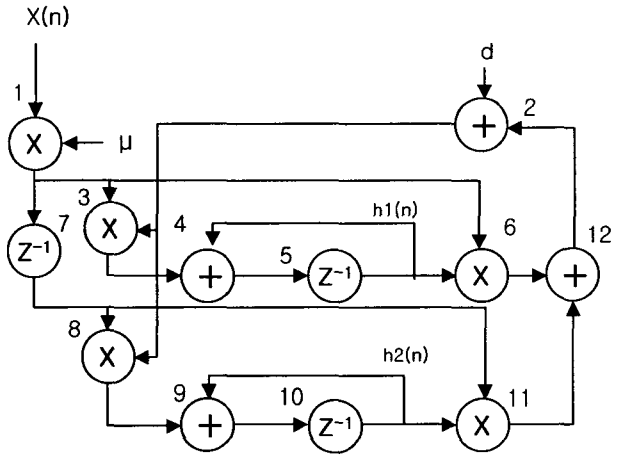

그림 1. 2차 LMS 필터의 FSFG.
Fig. 1. FSFG of Second Order LMS Filter

FSFG는 시 불변 플로우 그래프(Shift-Invariant Flow Graph)로서 각 노드는 알고리듬이 구현될 타겟 프로세서에서의 최소 연산단위를 나타내어 알고리듬의 병렬구조를 최대한 이용 가능하도록 되어있다.
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.[4] 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 l 에서 지연소자의 총 갯수;
d_j =루프 l의 j번째 노드의 연산 소요시간;
D_l =루프 l의 총 연산소요 시간;
l = 루프지수.

그림 1의 LMS 필터F에서 $d_{mmult}=d_{add}=1$라고 가정하면 IPB는 5가 된다. 스케줄이 IPB를 만족하면 rate-optimal 스케줄이라 한다.
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 모든 노드들의 연산소요시간의 총합 ;
T_0 IPB.

그림 1에서 PB는 2가 된다. PB를 충족하는 스케줄을 processor-optimal 스케줄이라 한다.
아래의 식으로 주어지는 PDB는 하나의 입력 샘플이 계산되어 출력에 나타나는데 소요되는 시간을 의미하며 이를 만족하는 스케줄을 delay-optimal스케줄이라 한다,..

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

여기서

D_p =입출력 경로 p 상의 연산소요시간의 총합;
n_p = 경로 p상의 지연소자의 수;
T_o = IPB.

그림 1에서 PDB는 2가되며 해당 입출력 경로는 1-3-4-5-6-12-2-3이 된다.

2.2 LMS 필터의 Cyclo-Static 스케줄 생성

VLSI 구현의 전단계로서 주어진 알고리듬으로부터 세 가지 bound를 계산한 후 스케줄을 생성하게 되는데 본 논문에서는 Cyclo-Static 스케줄러를 채택하였다. Cyclo-Static

스케줄러는 동기식 멀티프로세서 스케줄러로서 주어진 FSFG로부터 rate optimal, processor optimal, 그리고 delay optimal인 스케줄을 생성한다.[4] 또한 Cyclo-Static 스케줄러는 IPB에 대하여 주기적이므로 한주기동안의 스케줄 정보만으로 충분하다. 표 1은 그림 1에 보인 2차 LMS 필터의 한주기 동안의 Cyclo-Static 스케줄이다. 여기서 밑수는 노드를 그리고 지수는 현재의 입력 샘플에 대한 상대적인 시간을 나타낸다. 각 프로세서는 승산 및 가산 연산을 모두 수행 할 수 있어야 한다.

표 1. 2차 LMS 필터의 Cyclo-static 스케줄

Table. 1 Cyclo-static Schedule of Second Order LMS Filter

시간	1	2	3	4	5
pr #1	6^0		1^1	3^0	4^0
pr #2	11^0	12^0	2^0	8^0	9^0

2.3 LMS 필터의 Cyclo-Static 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 보존된다.

생성된 Cyclo-static 스케줄을 변환하는 과정은 다음과 같다. 우선 스케줄구현에 필요한 최소의 승산기수와 가산기수를 결정한 후 각 프로세서는 한 종류만의 연산을 하도록 같은 행의 요소들을 교환하여 스케줄을 재배열한다. 그리고 스케줄과 FSFG로부터 연결 표를 생성하는데 연결표의 각 요소는 스케줄의 특정 시간에 있어서 각 노드의 연산의 결과가 어디로 연결되는가를 기술한다. 이 연결 표를 이용하여 각 승산노드와 가산노드를 각 프로세서에 할당하였을 때 필요한 연결선수를 구하여 이것이 최소가 되도록 한다. 이 과정을 각 단계에서 상술하면 다음과 같다.

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.

표2. 2차 LMS 필터의의 재배열된 스케줄

Table 2. Rearranged Schedule of Second Order LMS Filter

시간	1	2	3	4	5
pr #1(mult)	6^0			8^0	
pr #2(mult)	11^0		1^1	3^0	
pr #3(add)		12^0			4^0
pr #4(add)			2^0		9^0

③. 스케줄과 FSFG로부터 연결 표를 만든다. 연결표의 각 요소는 스케줄의 특정 시간에 있어서 각 노드의 연산의 결과가 어디로 연결되는가를 기술한다. 표3에 그림1과 표1로부터 구한 연결 표를 보인다.

표3. 연결표

Table 3. Communications Table

1	2	3	4	5
$6^0 \rightarrow 12^0$	$12^0 \rightarrow 2^0$	$2^0 \rightarrow 3^0$	$3^0 \rightarrow 4^0$	$4^0 \rightarrow 6^1$
$11^0 \rightarrow 12^0$		$2^0 \rightarrow 8^0$	$8^0 \rightarrow 9^0$	$4^0 \rightarrow 4^1$
		$1^1 \rightarrow 3^1$		$9^0 \rightarrow 11^1$
		$1^1 \rightarrow 6^1$		$9^0 \rightarrow 9^1$
		$1^1 \rightarrow 8^2$		
		$1^1 \rightarrow 11^2$		

④. 수정된 스케줄의 첫째 열에 재배열된 스케줄의 첫째 열로 채운다.

⑤. 각 시간에서(시간 2에서 시작하여) 승산노드를 프로세서 #1에 할당하고 연결 표를 이용하여 부가적 인 연결선의 수를 구한다. 이 과정을 나머지 승산기에 대해서 계속한다. 어떤 노드가 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 복잡 도는 $O(PNM^2)$가 된다.

⑥. ⑤의 과정을 가산노드에 대해서 수행한다. 승산노드의 경우와 마찬가지로 계산 복잡 도는 $O(PNA^2)$가 된다.

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

표4에 2차 LMS 필터의 변환된 스케줄을 보인다. 이 스케줄을 구현 하려면 2 개의 승산기 및 가산기가 필요하다.

표4. 2차 LMS 필터의 변환된 스케줄
Table 4. Modified Schedule of Second Order LMS Filter

시간	1	2	3	4	5
pr #1(mult)	6^0		1^1	3^0	
pr #2(mult)	11^0			8^0	
pr #3(add)		12^0	2^0		4^0
pr #4(add)					9^0

III. 스케줄을 이용한 VLSI구조설계

3.1 프로세서의 구조

LMS 필터를 FSFG로 표시하면 각 노드의 연산은 승산 혹은 가산이므로 이를 실현할 프로세서의 구조는 다음과 같다. 제 I 형 프로세서는 승산기, 레지스터 및 멀티플렉서로 구성되며 제 II 형 프로세서는 가산기, 레지스터 및 멀티플렉서로 구성된다. 그림 2에 이 프로세서의 구조를 보인다.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. MUX1은 REG1에 저장될 값을 결정한다. MUX2는 현재 프로세서에서 수행할 연산에 사용될 데이터를 저장하고 있는 REG1를 선택한다.

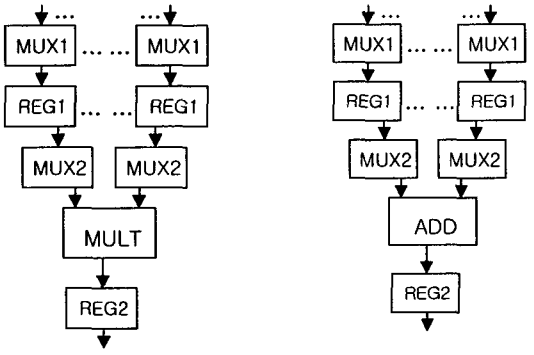

그림 2. 프로세서의 구조 (a)제 I 형 (b) 제 II 형
Fig. 2 Architecture of Processors (a)Type I (b) Type II

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 제한되어있어 본 논문에 채택된 프로세서의 형태도 이에 따랐다.

3.2 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. 하드웨어할당의 전체과정은 수정된 스케줄의 각 행에 해당하는 프로세서의 형태를 결정한 후 각 프로세서의 연결 및 REG1 할당을 행하고 멀티플렉서 할당을 한다. 이 과정을 상술하면 다음과 같다.

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.

각 프로세서의 연결 및 REG1 할당

② 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 선행 노드를 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,

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.

멀티플렉서 (MUX1 및 MUX2)의 할당

⑤ REG1은 하나 이상의 프로세서에 연결될 수 있다. 이 경우 MUX1에 저장될 데이터를 결정한다.

⑥.각 프로세서는 두 그룹의 REG1이 있는데 각 그룹에서 둘 이상의 REG1이 있으면 REG1과 연산 장치 사이에는 MUX2가 필요하게 된다.

⑦ 각 시간에서의 프로세서 값으로부터 REG1를 로

딩하고 MUX1 및 MUX2를 선택하는 제어신호를 발생 한다. 이 제어신호는 주기적인 진리표로 나타낼 수 있다.

그림 3과 표 5는 2차 LMS 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.

IV 결론

본 논문에서는 멀티프로세서 구조를 채택하여 입력의 샘플링 속도, 프로세서의 수, 그리고 주어진 입력에 대한 출력의 지연에 있어 최적인 LMS필터를 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.

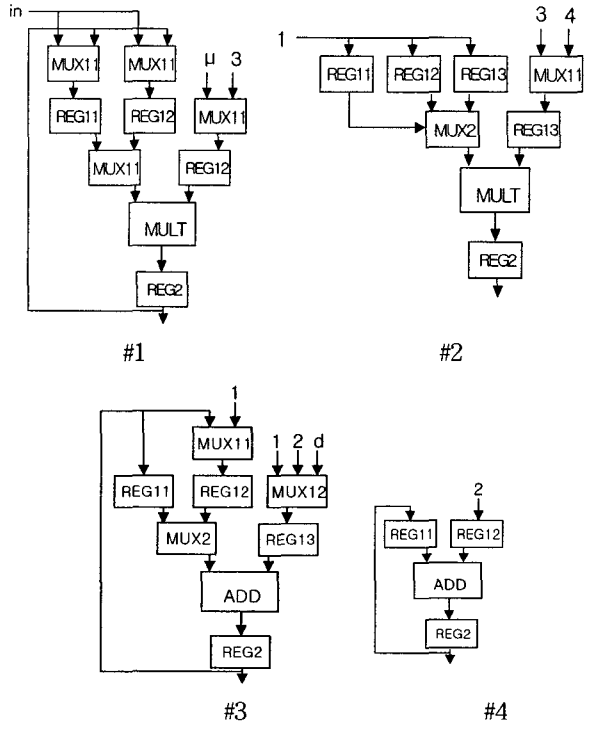

그림 3. 2차 LMS 필터의 회로도

Fig. 3. Circuit Diagram of Schedule of Second Order LMS Filter

표5. 그림4에 보인 회로도에 대한 제어신호

Table 5. Control Signals for Circuit Diagram shown in Fig. 3

time		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
pr# 1	R11	X	X	1	10	0	0	0	0	0	X	repeat				
	R12	0	0	0	0	X	X	X	1	1	0	repeat				
	R13	1	X	1	1	X	1	X	X	X	X	repeat				
	M11	X	X	in	1	X	X	X	X	X	X	repeat				
	M12	X	X	X	X	X	X	X	in	1	X	repeat				
	M13	3	X	μ	3	X	3	X	μ	3	X	repeat				
	M2	R12	X	R11	R12	X	R11	X	R12	R11	X	repeat				
pr# 2	R11	0	0	0	0	0	0	0	0	0	X	X	X	X	1	0
	R12	X	X	X	1	0	0	0	0	0	0	0	0	0	0	X
	R13	0	0	0	0	X	X	X	X	1	0	0	0	0	0	0
	R14	1	X	X	1	X	1	X	X	1	X	1	X	X	1	X
	M1	4	X	X	3	X	repeat									
	M2	R13	X	X	R13	X	R11	X	X	R11	X	R12	X	X	R12	X
pr# 3	R11	1	0	0	0	0	repeat									
	R12	X	1	1	X	X	repeat									
	R13	X	1	1	X	1	repeat									
	M11	X	1	3	X	X	repeat									
	M12	X	2	d	X	1	repeat									
	M2	X	R12	R12	X	R11	repeat									
pr# 4	R11	1	0	0	0	0	repeat									
	R12	X	X	X	X	1	repeat									

참 고 문 헌

[1] B. Widrow and S.D.Sterns, *Adaptive Signal processing*. Englewood Cliffs, NJ: Prentice Hasll, 1985

[2] 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.

[3] B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," *WESCON conv. Rec.*, Vol. 4, 1960,pp. 96-140

[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.

김 형 교(Hyeong-Kyo Kim)

1978년 2월 서울 대학교 전기공학과 공학사

1980년 2월 서울 대학교 전자공학과 공학석사

1993년3월 Georgia Institute Technology, School of Electrical Eng. Ph.D.

1993년 7월 ~ 1995년 2월 한국전자통신연구원 선임연구원

1995년 3월 ~ 1997년 3월 상명대학교 정보과학과 전임강사

1997년 3월 ~ 현재 한신대학교 정보통신학과 부교수

관심분야 : DSP, VLSI Signal Processing, System Identification