논문 2008-45SD-1-6

변수그룹을 이용한 효율적인 연산기 설계

(Efficient Operator Design Using Variable Groups)

김 용 은*, 정 진 균**

(Yong-Eun Kim and Jin-Gyun Chung)

요 약

본 논문에서는 곱셈기나 필터 등에서 연산을 위해 부분곱을 더할 때 더해질 변수를 그룹화하여 연산기를 설계하는 방법을 제시한다. 제안한 그룹화 알고리즘을 사용하면 기존의 full adder cell이 간단한 로직회로로 대치되고 이에 따라 면적, 전력소모, 속도면에서 효율적인 디자인이 가능하다. 제안한 방법을 7bit, 8bit 제곱기 및 FIR 필터에 사용되는 precomputer 블록에 적용한 결과 기존의 방법 보다 면적, 전력소모, 속도에서 각각 {22.1%, 20.1%, 14%}, {24.7%, 24.4%, 6.7%}, {63.6%, 34.4%, 9.8%} 의 이득 있음을 보인다.

Abstract

In this paper, we propose a partial product addition method using variable groups in the design of operators such as multipliers and digital filters. By this method, full adders can be replaced with simple logic circuits. To show the efficiency of the proposed method, we applied the method to the design of squarers and precomputer blocks of FIR filters. In case of 7 bit and 8 bit squarers, it is shown that by the proposed method, area, power and delay time can be reduced up to {22.1%, 20.1%, 14%} and {24.7%, 24.4%, 6.7%}, respectively, compared with the conventional method. The proposed FIR precomputer circuit leads to up to {63.6%, 34.4%, 9.8%} reduction in area, power consumption and propagation delay compared with previous method.

Keywords : variable group, multiplier, partial product, squarer, FIR precomputer, shift-addition

Ⅰ. 서 론

곱셈기나 제곱기와 같은 연산에서는 부분곱을 더하는 과정이 필요하며 부분곱을 더하는 과정은 연산의 속도나 면적에 큰 영향을 미친다. 부분곱을 효율적으로 더하기 위해 많은 알고리즘들이 제안되었다. 잘 알려진 부분곱 감소 방법으로는 carry-save array 방식과 트리 방식이 있다[1]. 트리 방식에는 Wallace 트리, Dadda 트리 등이 있으며, 이 방식들은 FA(Full Adder) 또는 HA(Half Adder)를 사용하여 부분곱을 감소시킨다[2~4].

트리방식은 carry-save arrary 방법보다 고속으로 부분곱감소를 수행할 수 있지만 덧셈기 셀이 불규칙적이며 지연 경로가 불균형적인 단점이 있다. 이러한 트리 방식의 단점을 개선하기 위해 4:2 compressor가 도입되었다[5].

트리 방식으로 부분곱을 연산할 때 부분곱 최대 행렬 높이는 연산 스테이지의 수와 관련이 있으므로 부분곱의 최대 행렬 높이를 줄이는 것이 효율적이다. 따라서 부분곱 행렬에서 같은 변수를 찾아 그룹화하고 변수 그룹을 논리회로로 구현하면 부분곱 행렬 높이가 줄어들어 연산 회로가 감소될 수 있다. 또한 한 입력에 대하여 출력이 여러 개일 경우 다중 출력 함수 최적화 방법으로 논리회로를 공유하여 더욱더 회로가 간단히 설계 될 수 있다.

본 논문에서는 제곱기와 FIR 필터의 precomupter[6]의 입력 변수를 효율적으로 그룹화하고 FA 또는 HA의

* 학생회원, ** 정회원, 전북대학교 전자정보공학부
(Div. of Electronic & Information Engineering Chonbuk University)

※ This work was supported by the IT R&D program of MIC/IITA.[2007-S051-1, Audio-visual Processing SoC for Intelligent Robot]
접수일자: 2007년9월14일, 수정완료일: 2008년1월7일

개수를 줄여 기존의 설계 방식에 비해 면적, 속도, 전력 소모에서 효율적인 방법을 제안한다. Ⅱ 절에서는 제곱기의 설계를 이용하여 제안하는 변수 그룹 방법을 설명하며, Ⅲ절에서는 FIR필터의 precomputer 블록 설계에 응용한다. Ⅳ절에서는 시뮬레이션 결과를 제시하고, Ⅴ절에서 결론을 맺는다.

II. 제안한 방법을 이용한 제곱기 설계 방법

1. 기존의 제곱기 설계 방법

제곱기는 Booth 폴딩 방식과 재배열 방식이 사용되며 제곱기의 입력 워드 길이가 10 비트 이하 일 때 재배열 방식이 효율적이다[7~9].

부분곱을 더할 때는 일반적으로 Wallace 트리 방식을 사용한다. 그림 1-(a), (b)는 입력 워드 길이가 각각 7 비트, 8 비트인 제곱기의 부분곱을 Wallace 트리 방식을 이용하여 더하는 과정을 나타낸다.

그림 1에서 3개의 변수로 묶인 것은 FA를, 2개의 변수로 묶인 것은 HA를 사용하여 더하며 마지막 단은 vector merge adder를 사용한다.

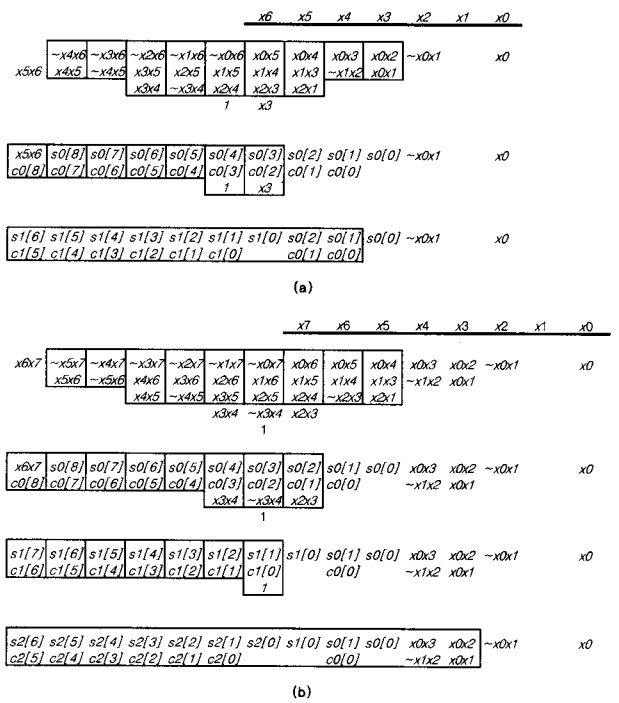


그림 1. 제곱기의 부분곱을 Wallace 트리 방식을 이용하여 더하는 과정: (a) 입력 워드길이 = 7, (b) 입력 워드길이 = 8 (s : sum, c : carry).

Fig. 1. Partial product addition process of squarers using Wallace tree method: (a) W=7, (b) W=8 (s: sum, c: carry).

2. 변수 그룹을 이용한 제곱기 설계 방법

입력이 7 비트인 제곱기를 설계할 때 입력 $x[6:0]$을 인접한 변수 $x[6:4]$, $x[4:2]$, $x[2:0]$으로 그룹화 하여 영역을 표시하면 그림 2-(a)와 같다. 각각의 영역이 더해진 결과를 캐리 아웃을 고려하여 $t2[2:0]$, $t1[3:0]$, $t0[3:0]$으로 표시 할 수 있다.

각각의 영역이 더해진 결과 $t2[2:0]$, $t1[3:0]$, $t0[3:0]$은 입력 $x[6:4]$, $x[4:2]$, $x[2:0]$의 카르노맵을 이용하여 간단한 논리회로로 나타낼 수 있다. 이들의 진리표를 작성하여 최소항을 구하면 표 1과 같다. 제안한 그룹화 방식을 이용한 7 비트 제곱기의 부분곱이 더해지는 과정은 그림 2-(b)와 같다.

표 2의 카르노맵을 이용해 표 1의 $t0[3:0]$에 대한 표현식을 구하면 표 3과 같다. 그룹화 된 $t2[2:0]$, $t1[3:0]$, $t0[3:0]$과 그룹화 되지 않은 나머지 변수를 Wallace 트리 방식으로 연산하여 최종 출력을 구한다.

HA를 0.5개의 FA로 간주할 때 그림 2-(b)에서 변수 그룹화를 위한 회로의 오버헤드를 제외하고 최후 2줄이 남을 때까지 사용된 FA는 5.5개이다. 같은 방법으로 그

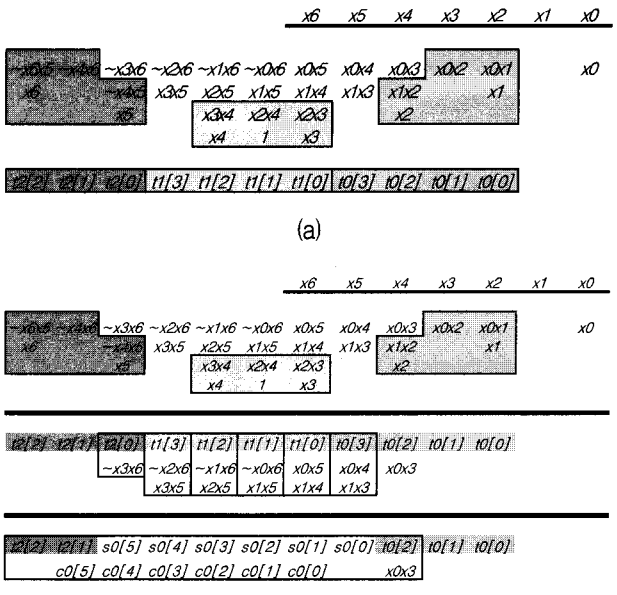


그림 2. 변수 그룹 방법으로 설계한 제곱기: (a) 제곱기의 입력 $x[6:0]$을 $x[6:4]$, $x[4:2]$, $x[2:0]$ 으로 그룹화 한 영역, (b) 그룹화를 이용한 7 비트 제곱기의 부분곱 계산과정.

Fig. 2. Squarer design using Grouping method (a) Grouping of partial products in squarer with $x[6:4]$, $x[4:2]$, and $x[2:0]$. (b) Partial Product addition computation process for W=7 using grouping method.

표 1. $t2[2:0]$, $t1[3:0]$, $t0[3:0]$의 최소항

Table 1. Minterms of $t2[2:0]$, $t1[3:0]$ and $t0[3:0]$.

$t0[3]$	$\Sigma(6, 7)$
$t0[2]$	$\Sigma(4, 5, 7)$
$t0[1]$	$\Sigma(3, 5)$
$t0[0]$	$\Sigma(2, 6)$
$t1[3]$	$\Sigma(5, 6, 7)$
$t1[2]$	$\Sigma(3,4, 7)$
$t1[1]$	$\Sigma(0, 1, 2, 4, 6, 7)$
$t1[0]$	$\Sigma(2, 6)$
$t2[2]$	$\Sigma(6, 7)$
$t2[1]$	$\Sigma(3, 4, 6, 7)$
$t2[0]$	$\Sigma(2, 6)$

표 2. 표 1의 $t0[3:0]$에 대한 카르노맵 전개

Table 2. Karnaugh map development of group $t0[3:0]$.

$x[0]$ \ $x[2:1]$	00	01	11	10
0	0	2	6	4
1	1	3	7	5

$x[0]$ \ $x[2:1]$	00	01	11	10
0	0	2	6	4
1	1	3	7	5

$x[0]$ \ $x[2:1]$	00	01	11	10
0	0	2	6	4
1	1	3	7	5

$x[0]$ \ $x[2:1]$	00	01	11	10
0	0	2	6	4
1	1	3	7	5

림 1-(a) 의 FA 개수를 계산해보면 11.5개이다. 따라서 표 1의 그룹화로 인한 논리회로의 오버헤드를 FA 2.5개로 간주 할 때 제안한 방법을 이용하여 3.5개의 FA를 감소시킬 수 있음을 알 수 있다.

입력 워드 길이가 8 비트인 제곱기를 설계할 때 입력 $x[7:0]$을 $x[7:5]$, $x[5:2]$, $x[2:0]$으로 그룹화 하여 각각 영역을 표시하고 각각의 영역이 더해진 결과를 $t2[2:0]$, $t1[5:0]$, $t0[3:0]$이라고 하면 그림 3과 같다. $t2[2:0]$, $t1[3:0]$, $t0[3:0]$은 입력 $x[7:5]$, $x[5:2]$, $x[2:0]$의 논리식 나타낼 수 있다. 이들의 진리표를 작성하여 최소항을 구하면 표 4와 같다.

그림 3에서 최후 2줄이 남을 때까지 사용된 FA는 7.5개로, 이것은 그림 1-(b)의 17.5개보다 9.5개가 적다. 연산 stage 개수도 기존의 4단과 비교해 볼 때 제안한

표 3. $t0[3:0]$에 대한 최소항 계산 값

Table 3. Karnaugh map results of $t0[3:0]$.
($\cdot$: AND, +: OR, ^ : XOR)

	입력 $x[2:0]$에 대한 최소항
$t0[3]$	$x_2 \cdot x_1$
$t0[2]$	$x_2 \cdot (\overline{x_1} + x_0)$
$t0[1]$	$x_0 \cdot (x_2 \wedge x_1)$
$t0[0]$	$x_1 \cdot (\overline{x_0})$

표 4. $t2[2:0]$, $t1[5:0]$, $t0[3:0]$의 최소항

Table 4. Minterms of $t2[2:0]$, $t1[5:0]$ and $t0[3:0]$.

$t0[3]$	$\Sigma(6, 7)$
$t0[2]$	$\Sigma(4, 5, 7)$
$t0[1]$	$\Sigma(3, 5)$
$t0[0]$	$\Sigma(2, 6)$
$t1[5]$	$\Sigma(11, 12, 13, 14, 15)$
$t1[4]$	$\Sigma(7, 8, 9, 10, 14, 15)$
$t1[3]$	$\Sigma(4, 5, 6, 9, 10, 12, 13, 15)$
$t1[2]$	$\Sigma(0, 1, 2, 3, 6, 8, 10, 13, 14, 15)$
$t1[1]$	$\Sigma(3, 5, 11, 13)$
$t1[0]$	$\Sigma(2, 6, 10, 14)$
$t2[2]$	$\Sigma(13, 14, 15)$
$t2[1]$	$\Sigma(6, 7, 8, 9, 13, 14, 15)$
$t2[0]$	$\Sigma(4, 5, 8, 10, 13, 14)$

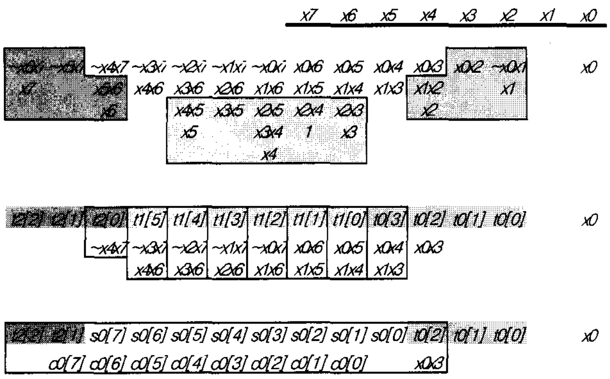


그림 3. 그룹화를 이용한 8 비트 제곱기의 부분곱 계산과정

Fig. 3. Grouping of partial products in squarer with W=7.

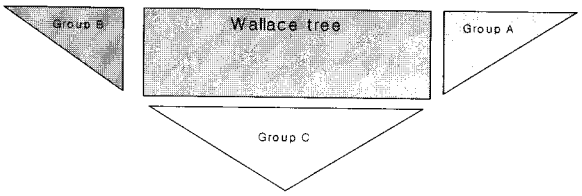


그림 4. 제곱기의 그룹화 연산 부분
Fig. 4. Grouping part of squarer's partial products.

방법이 1단 적음을 알 수 있다. 또한 $t2[2:0]$, $t1[5:0]$, $t0[3:0]$은 다중 출력 함수 이므로 내부 논리회로를 공유하는 것이 효과적이다.

제곱기의 사이즈가 커질 때에는 각 그룹 내의 변수의 개수를 적절히 선택하여 그룹화로 인한 오버헤드의 증가를 최소화 시키는 것이 필요하다. 그림 4는 제곱기에서 그룹화 시킬 부분을 보여준다. 제곱기의 경우 같은 변수가 밀집되어 있는 부분을 Group A(하위), Group B(상위), Group C(중앙) 부분 즉 3개의 부분으로 나눌 수 있다. 이 그룹들을 카르노맵을 이용하여 제곱기의 부분곱의 줄 수를 줄이고 남겨진 부분곱과 그룹의 출력을 Wallace 트리 방식을 이용하여 연산한다.

제안한 변수그룹을 이용한 부분곱 감소 알고리즘을 요약하면 다음과 같다.

1. 동일한 변수가 밀집된 부분을 찾는다.
2. 그룹화 할 변수 개수를 4개 이내로 하여 변수를 그룹화하고 각 그룹의 출력을 카르노맵을 이용하여 간단한 조합회로로 변환한다.
3. 각 그룹의 출력과 그룹에 포함되지 않은 변수들을 Wallace트리와 같은 부분곱 감소 알고리즘을 이용하여 연산한다.

Ⅲ. 변수 그룹을 이용한 FIR precomputer 설계

1. 기존의 FIR 필터 precomputer 블록 설계방법

그림 5는 피연산자를 공유하여 입력 x에 대해 $1x$, $3x$, $5x$, $7x$, $9x$, $11x$, $13x$, $15x$ 값을 미리 계산해 두고 이를 쉬프트 하여 더하는 방식으로 곱셈기를 구현한 피연산자 공유 FIR 필터 구조이다[6]. 피연산자 공유 FIR 필터의 precomputer 블록에서는 입력 x의 홀수 배수 $1x$, $3x$, $5x$, $7x$, $9x$, $11x$, $13x$, $15x$의 출력을 만든다. 그림 5에서는 x에 11100100이 곱해지는 예로 든다. 상위 4 비트 1110은 shifter 블록에서 111로 오른쪽으로 1번 쉬프트되며 111이므로 precomputer에서 계산된 $7x$

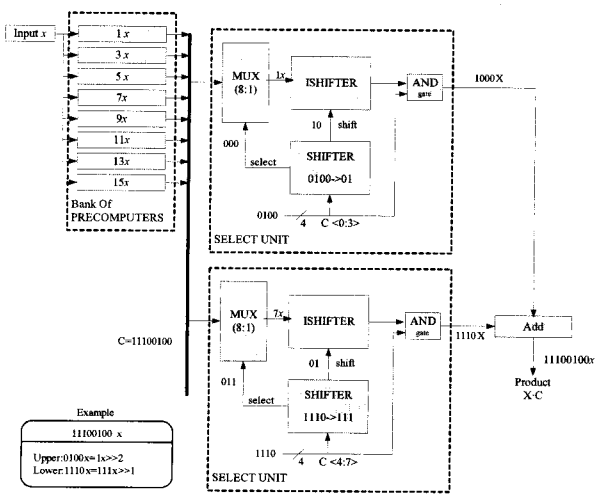


그림 5. High performance FIR 필터
Fig. 5. High performance FIR filter.

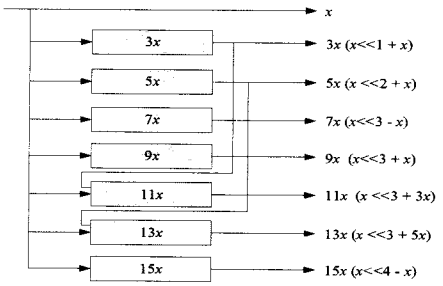


그림 6. Shift_add를 이용한 precomputer 구조
Fig. 6. Precomputer structure using shift_add.

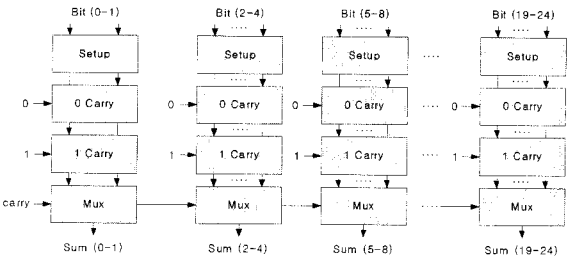


그림 7. Precomputer 내부 덧셈기 구조
Fig. 7. Adder structure of precomputer.

를 출력하고 하위 4비트 0100은 shifter에서 오른쪽으로 2번 쉬프트 되어 1이 되므로 x가 출력된다. ishifter에는 오른쪽으로 쉬프트 되었던 값만큼 precomputer의 출력을 왼쪽으로 쉬프트하여 값을 복원하고 Add블럭에서는 상위 비트에서 출력된 값과 하위 비트에서 출력된 값을 더한다.

홀수 배수 출력은 shift-add를 이용하여 그림 6과 같이 계산할 수 있다. $13x$의 출력을 구하기 위해서는 $5x$의 출력과 $8x$의 출력을 더하므로 $13x$가 가장 긴 계산 시간을 요구한다.

계산 시간을 단축시키기 위해 [6]에서는 FIR

precomputer의 덧셈기에 그림 7과 같은 square root carry select 가산기를 사용하였다.

2. 변수 그룹 방법을 이용한 FIR 필터 precomputer 블록 설계방법

제안한 변수 그룹 방법을 이용하여 FIR precomputer 블록을 효율적으로 설계할 수 있다. 예를 들어 13× 의 경우 II절과 같은 방법으로 $x[7:4]$, $x[3:0]$으로 그룹화 시키면 그림 8과 같다. 그림 8에서 각각의 그룹 변수의 덧셈의 결과는 $t0[6:0]$, $t1[7:0]$으로 표현하였고 II절에서와 같이 $t0[6:0]$, $t1[7:0]$를 계산하는 회로는 카르노맵을 이용하여 설계할 수 있다. 따라서 $13x$는 그림 9와 같이 변수그룹 회로와 8 비트 리플 가산기(3비트 : HA, 5비트 : FA)가 이용된다.

지연 시간은 변수 그룹 회로, 3개의 FA, 4개의 HA으로 기존의 $13x$의 지연시간 즉 $8x$ + $5x$에서 사용된 square root carry select 지연시간 (FA 12개, Mux 2개) 보다 작고 면적도 작다. 그림 9는 제안한 $13x$ 연산 회로이다.

제안한 알고리즘을 이용하여 $1x$, $3x$, $5x$, $7x$, $9x$, $11x$, $13x$, $15x$의 $t0$, $t1$을 회로를 설계 할 때 $t0$, $t1$은 입력 x에 대한 다중 출력이므로 다중 출력 함수의 최적화 기법을 이용하면 더욱더 최적화할 수 있다.

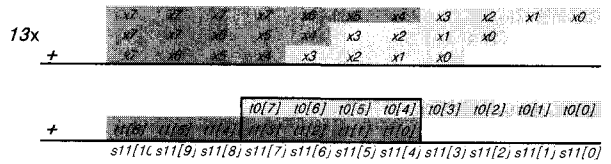


그림 8. 변수 그룹 된 13×의 부분곱
Fig. 8. Grouping of 13× partial product.

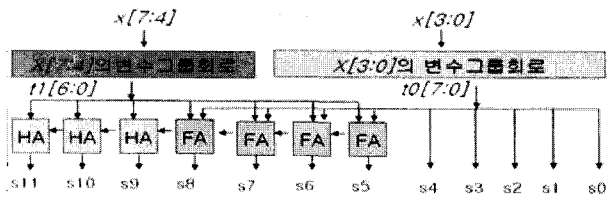


그림 9. 제안한 13× 덧셈기 회로
Fig. 9. Proposed adder for $13x$.

IV. 실험결과

입력이 각각 7비트, 8비트인 제곱기와 FIR 필터의 precomputer 블럭을 기존의 방법과 제안한 방법을 이용하여 설계하고, 삼성 0.35μ 라이브러리를 이용하여 Synopsys로 합성한 결과는 표 5-(a),(b),(c)와 같다. 7비트 제곱기는 면적, 파워, 속도면에서 각각 22.1%, 20.1%, 14%의 이득을 얻었고, 8비트 제곱기의 경우 각각 24.7%, 24.4%, 6.7%의 이득을 얻었다. 그룹화 부분의 논리회로는 Sysnopsys사의 다중 출력 함수 최적화 알고리즘을 이용하여 최적화 시켰다.

표 5. 시뮬레이션 결과: (a) 7 비트 제곱기, (b) 8 비트 제곱기, (c) precomputer
Table 5. Simulation results: (a) squarer with W=7, (b) squarer with W=8, (c) precomputer.

(a)

구 분	7 비트 제곱기		
	기존 방법	제안 방법	이 득(%)
면적(cell)	177.6	138.4	22.1
파워(mW)	13.0	10.3	20.1
속도(ns)	4.2	3.6	14

(b)

구 분	8 비트 제곱기		
	기존 방법	제안 방법	이 득(%)
면적(cell)	247.1	186.2	24.7
파워(mW)	19.0	14.4	24.4
속도(ns)	4.5	4.2	6.7

(c)

구 분	FIR precomputer		
	기존 방법	제안 방법	이 득(%)
면적(cell)	1119	407.3	63.6
파워(mW)	61.1	40.1	34.4
속도(ns)	4.1	3.7	9.8

V. 결 론

제곱기나 FIR필터의 precomputer를 설계할 경우 덧셈이 이루어져야하는 변수를 제안한 방법으로 그룹화시켜 FA또는 HA를 이용하지 않고 논리회로로 설계하면 연산량이 줄고 면적이 감소된다. 추후 변수 그룹 방식이 효율적으로 적용되는 연산회로를 찾고 변수 그룹이 효율적으로 이루어지는 그룹 방법에 대한 연구가 필요하다.

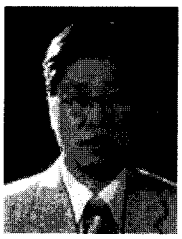
참 고 문 헌

[1] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design.* New York: Wiley, 1979.

[2] C. S. Wallace, "A suggestion for a fast multiplier", *IEEE Trans. on Electron. Comp.*, vol. 13, pp. 14-17, 1964
[3] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349-356, 1965.
[4] K. A. C. Bickerstaff, M. Schulte, and E. E. Swartzlander, "Reduced area multipliers," in *Proc. Int. Conf. on Application-Specific Array Processors*, PP. 478-489, 1993.
[5] G. Goto, T. Sato, M. Nakajiama, and T. Sukemura, "A 54×54 regulary structured tree multiplier," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1229-1236, Sept. 1992.
[6] J. Park, K. Muhammad and K. Roy, "High performance FIR filter design based on sharing multiplication," *IEEE Transaction on VLSI systems*, vol. 11, pp. 244-253. April. 2003.
[7] J. Pihl and E. Aas, "A multiplier and squarer generator for high performance DSP applications," *in Proc. IEEE 39th Midwest Symp. on Circuit and Systems*, 1996, pp. 109-112.
[8] R. K. Kolagotla , W. R. Griescbach, and H. R. Srinivas, "VLSI implementation of a 350 MHz 0.35um 8 bit merged squarer," *Electronic Letters*, vol. 34, pp. 47-48, Jan. 1998.
[9] K. E. Wires, M. J. Schulte, L. P. Marquette, and P. I. Balzola, "Combined unsigned and two's complement squarers," *in 33rd Asilomar Conference on Signals, Systems, and Computers*, pp. 1215-1219, 1999.

저 자 소 개

김 용 은(학생회원)
2005년 전북대학교 전자공학과 학사 졸업
2007년 전북대학교 정보통신 공학과 석사 졸업
2007년~현재 전북대학교 전자 정보공학부 박사
<주관심분야 : 통신, 신호처리, 반도체>



정 진 균(정회원)
1985년 전북대학교 전자공학 학사 졸업
1989년 미국 미네소타 주립대학 전기공학 석사 졸업
1991년 미국 미네소타 주립대학 전기공학 박사 졸업
<주관심분야 : 통신, 컴퓨터, 신호처리, 반도체>