論文94-31B-10-7

32 × 32 비트 고속 병렬 곱셈기 구조

(An Architecture for 32 × 32 bit high speed parallel multiplier)

金 榮 民*, 曺 鎭 鎬* *

(Young min Kim and Jin Ho Cho)

要約

본 논문에서는 각종 컴퓨터 및 디지탈 신호처리에서 중요한 역할을 하는 32 비트 정수형 고속 병렬 곱셈기 구조를 제안한다. 본 구조에서는 Booth 알고리즘의 일종인 비트-쌍 기록법을 채택하였으며 이것은 n비트 승수부에 대하여 n/2 개의 부분곱을 생성하며 음수 양수 부호을 동일하게 처리가 가능하다. 곱셈 과정에서 부분곱을 생성시 승수부의 비트 형태에 따라 피승수부의 2진 보수가 요구되는데 본 논문에서는 부분곱을 1의 보수로하고 부분곱이 음수인 경우 2의 보수가 되기 위한 1을 더하는 과정을 나중에 병렬 계수기에서 더하는 방법을 제시하였다. 이렇게 구해진 16개의 부분곱과 한 개의 추가 부분곱은 병렬 계수기로 2개의 summand로 줄인다. 최종적으로 2개의 summand을 BLC 가산기로 더하여 최종 곱셈값을 구하였다. 0.8㎛CMOS standard cell로 simulation한 결과 30ns의 곱셈속도를 얻었다.

Abstract

In this paper we suggest a 32 bit high speed parallel multiplier which plays an important role in digital signal processing. We employ a bit-pair recoding Booth algoritham that gurantees n/2 partial product terms, which uniformly handles the signed-operand case. While partial product terms are generated, a special method is suggested to reduce time delay by employing 1's complement instead of 2's complement. Later when partial products are added, the additional 1 bit's are packed in a single partial product term and added to in the parallel counter. Then 16 partial product terms are reduced to two summands by using successive parallel counters. Final multiplication value is obtained by a BLC adder. When this multiplier is simulated under 0.8㎛CMOS standard cell we obtain 30ns multiplier speed.

I. 서 론

CCITT 규정에서 알 수 있듯이 디지탈 신호 처리에 응용되는 Digital Filtering, Fourier

*正會員, *準會員 全南大學校 電子工學科
(Dept. of Elec. Eng., Chonnam Nat'l Univ.)
接受日字 : 1994年 1月 25日

Transform, Convolution, Correlation과 같은 알고리즘은 주로 승산 및 가산등과 같은 산술적인 연산들이다. 이러한 신호 처리가 실시간에 이루어지기 위해서는 고속의 연산 처리기가 필요하다. 특히 승산부분은 가장 중요하며 대표적인 연산 요소로 실시간 처리를 만족하는 고속 병렬 곱셈기가 요구된다. [1]

초기 곱셈기는 부호 크기 곱셈기에서 시작하여 1947년에 Shaw, Burks, Goldstine, von Neumann이 2진 보수 알고리즘으로 발전 시켰다. Booth에 의해 부분곱을 절반으로 줄이고 곱셈부호에 무관하는 알고리즘을 제안하여 현재의 모든 곱셈기 구조에 쓰이고 있다. 고속 연산을 하는데 중요한 문제점은 부분곱들을 더하는 과정에서 캐리의 전달에 있다. 이 부분곱들을 더하는 과정은 1964년 C.S. Wallace에 의하여 유사 가산 트리가 제안 되었으며 1965년 L.Dadda에 의해 병렬 계수기로 이를 구현하여 곱셈 속도 증가를 크게 증가 시켰다. [2]

본 논문에서 구현한 곱셈기는 32×32 비트 정수형 병렬 곱셈기이다. 여기에서 채택한 것은 'Recoding' 알고리즘의 부류인 Booth 알고리즘을 기반으로 하는 비트-쌍 기록법 혹은 수정형 Booth 알고리즘으로 부분곱을 16개로 줄여 32×16 비트 곱셈 형태로 줄였으며 부호 부분은 설계 면적을 줄이기 위해 부호 확장 방법이 아닌 부호 생성 알고리즘을 사용 하였다. 또한 각 부분곱들의 합을 구하는데 전달되는 캐리을 줄이는데 효율적인 Wallace 트리 일종인 병렬 계수 방법을 사용하였으며 최종합은 BLC(Binary Lookahead Carry) 가산기를 사용하였다. 특히 이 곱셈기의 특징은 부분곱을 생성시 기록된 숫자가 -1, -2 인 경우 2진 보수가 필요하다. 그러나 본 논문에서는 이 때 부분곱을 1의 보수로하고 2의 보수가 되기위한 1을 더하는 과정을 부분곱 생성시 더하지 않고 각 부분곱에서 발생하는 것을 하나의 추가 부분곱의로 하여 병렬 계수기를 사용하여 부분곱을 더할 때 1을 더하도록 하는 방법을 제안하였다.

Ⅱ. 비트-쌍 기록법 곱셈

본 논문에서 채택한 비트-쌍 기록법은 n 비트의 승수는 n/2개의 부분곱을 산출하고 음수 양수 부호에 무관하게 동일하게 처리하며 곱셈 속도를 계산 속도가 승수부의 비트 형태에 따라 달라지는 Booth 알고리즘의 최악의 상태보다 항상 2배의 속도를 보장하는 방법이다. [3] 이 방법은 Booth 기록법으로 부터 직접 유도된다. 비트-쌍 기록법은 다섯 개의 부호 숫자 0, +1, +2, -1, -2로 기록되는데 이 기록되는 숫자는 승수부의 비트 형태에 따라 직접 얻을 수 있다. n비트의 승수부 비트 형태에 따라 n/2 부분곱을 결정하는 비트-쌍 기록표는 표 1와 같다. [1], [3]

표 1. 비트-쌍 기록표 및 제어 신호
Table 1. bit-pair recoding table & control signal

승수부 비트 형태			부분곱	con[2]	con[1]	con[0]	S1	S0
i+1	i	i-1						
0	0	0	0 × M	×	×	0	0	0
0	0	1	+1 × M	0	1	1	0	0
0	1	0	+1 × M	0	1	1	0	0
0	1	1	+2 × M	0	0	1	0	0
1	0	0	-2 × M	1	0	1	1	0
1	0	1	-1 × M	1	1	1	0	1
1	1	0	-1 × M	1	1	1	0	1
1	1	1	0 × M	×	×	0	0	0

표 1에서 기록된 숫자가 0 이면 부분곱이 0이고, 1 이면 피승수부값이 그대로 부분곱이 되며 2이면 피승수부를 왼쪽으로 한 비트 이동한 값 즉 2를 곱한 값이 부분곱이다. 기록된 숫자가 -1이면 피승수부의 2진 보수가 부분곱이 되며, -2이면 피승수의 2진 보수를 한 비트 왼쪽으로 이동한 값이 부분곱이 된다. 그러나 본 논문에서 제안한 방법은 기록된 숫자가 -1, -2인 경우 부분곱은 피승수부의 1의 보수 혹은 1비트 왼쪽으로 이동한 값이 부분곱이 된다. 2의 보수를 구할 때 1을 더해주어야 하는 것은 기록된 숫자가 -1인 경우 S0가 -2인 경우 S1이 1이 된다. 각각의 부분곱에서 발생하는 S0, S1은 하나의 추가된 부분곱을 형성하다.

5개의 기록된 숫자에 의해 부분곱이 계산 되어지는 과정은 con [2] , con [1] , con [0] 3개의 제어 신호로 피승부 M을 적당히 조작하여 부분곱을 산출한다. 이 부분곱을 생성하는 로직 다이그램은 그림 1과 같다.

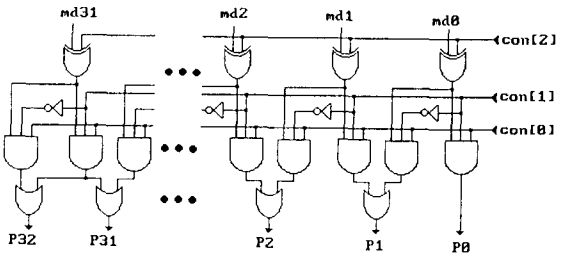


그림 1. 부분곱을 생성하는 로직 다이그램
Fig. 1. Logic diagram for partial product terms generation.

그림 1에서 기록된 숫자가 0일때 부분곱은 0이되는 과정은 con [0] 신호가 0로 con [2] , con [1] 에 무관하게 부분곱이 0이 된다. 기록된 숫자가 1이면 con [2] 가 0, con [1] 이 1, con [0] 가 1로 부분곱은 피승수부 값과 같다.기록된 숫자가 2이면 con [2] , con [1] 이 0, con [0] 가 1이 되어 피승수부를 좌측으로 한 비트 이동한 값이 부분곱이 된다. 기록된 숫자가 -1이면 con [2] , con [1] , con [0] 모두 1으로 피승수부에 보수를 취하는 값이 부분곱이되다. 기록된 숫자가 -2이면 con [2] 가 1, con [1] 이 0, con [0] 가 1이 되어 피승수부를 보수를 취하고 한 비트 왼쪽으로 이동한 값이 부분곱이 되다. 기록된 숫자가 2, -2 일 때 좌측으로 한 비트 이동함에 따라 부분곱을 나타내는데는 33 비트가 필요하다. S0는 기록된 숫자가 -1일 때 부분곱이 피승수부의 1의 보수가되고 1을 더하는 것을 피하고 병렬 계수기로 보내게 된다. S1은 기록된 숫자가 -2일 때 1이 되어 병렬 계수기로 1을 더하는 것을 보내게 된다.

그림 1에서 부분곱을 생성 하도록 하는 제어 신호 con [2] , con [1] , con [0] , S1, S0을 승수부의 비트 형태로 직접 나타낼 수 있다. 그림 2는 승수부 i-1, i, i+1 비트가 제어신호를 발생하는 로직 다이그램을 나타낸다.

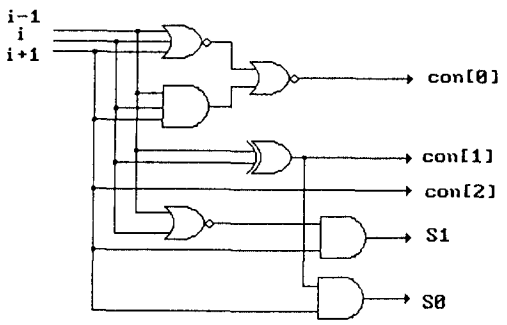


그림 2. 제어 신호의 로직 다이그램
Fig. 2. logic diagram of control signal.

Ⅲ. 부호 확장의 처리 방법

앞에서 설명한 것과 같이 32 비트 곱셈기의 부분곱을 표시 하는데는 33 비트가 필요하며 33 번째 비트는 부호를 표시한다. 부분곱을 더하는 과정에서 부호의 처리가 필요하는데 부호 처리는 부호 확장 방법과 부호 생성 방법이 있다. 본 논문에서는 곱셈기 설계시 설계 면적을 줄이기 위해 부호 생성 방법으로 부호 문제를 처리하였다. 부호 생성 방법은 다음과 같다. [1]

부분곱들의 부호 부분 전체의 값을 간략하게 표시하면 다음과 같다.

$$S = S_0\sum_{i=32}^{63}2^i + S_1\sum_{i=34}^{63}2^i + S_2\sum_{i=36}^{63}2^i + \cdots + S_{15}\sum_{i=62}^{63}2^i \tag{1}$$

식 (1)에서 S는 부분곱의 부호확장 부분만 전체 더한 값이고 S_0 , S_1 ,…, S_{15}는 각 부분곱의 33번째 비트 즉 부호의 값이다. 다음 2개의 등가식은 자명한 것이다.

$$\sum_{i=j}^{k}2^i = 2^k - 2^j \tag{2}$$

$$S^i = 1 - \overline{S_i} \tag{3}$$

식 (2)와 식 (3)을 식 (1)에 대입하면

$$\begin{aligned} S &= (1-\overline{S_0})(2^{64}-2^{32}) + (1-\overline{S_1})(2^{64}-2^{34}) + \cdots + (1-\overline{S_{15}})(2^{64}-2^{62}) \\ &= [15-(\overline{S_0}+\overline{S_1}+\cdots+\overline{S_{15}})]2^{64} + 2^{64} - (2^{32}+2^{34}+\cdots+2^{62}) \\ &\quad + (\overline{S_0}2^{32} + \overline{S_1}2^{34} + \cdots + \overline{S_{15}}2^{62}) \end{aligned} \tag{4}$$

그런데 2^{64}을 식(5)와 같이 쓸 수 있다.

$$2^{64} = 2^{63} + 2^{62} + 2^{34} + \cdots + 2^{32} + 2^{32} \tag{5}$$

이 식을 (4)식에 대입하여 정리하면

$$\begin{aligned} S &= [15-(\overline{S_0}+\overline{S_1}+\cdots+\overline{S_{15}})]2^{64} + 2^{63} + \overline{S_{15}}2^{62} + 2^{61} \\ &\quad + \overline{S_{14}}2^{60} + 2^{59} + \cdots + \overline{S_1}2^{34} + 2^{33} + (1+\overline{S_0})2^{32} \end{aligned} \tag{6}$$

식 (6)에서 첫번째항은 65번째 비트로 곱셈의 결과에는 아무런 영향을 주지않으므로 생략하면 부호의 값은 다음 식과 같이 나타낼 수 있다.

$$S = 2^{63} + \overline{S_{15}}2^{62} + 2^{61} + \overline{S_{14}}2^{60} + 2^{59} + \cdots + \overline{S_1}2^{18} + 2^{17} + (1+\overline{S_0})2^{16} \tag{7}$$

식 (7)에서 부호를 결정하는데는 각 부분곱의 부호 비트의 보수를 취하여 더하고 부호 비트 왼쪽에 1을 더하며 첫번째 부분곱 부호에 1을 더함으로 부호의 처리가 끝난다.

Ⅳ. 부분곱의 덧셈

비트-쌍 기록법에의해 16개의 부분곱이 생성되는데 이 부분곱을 더하는 과정은 병렬 계수 방법을 이용하

$\leq 2^m - 1$) 입력이 들어보면 m 개의 출력을 갖는 논리회로이다. 여기에서 m은 입력 n에 들어오는 "1"의 갯수를 2진수로 코드화한 값이다. [4], [5], [6]

다음 그림 3은 입력 n이 4인 경우의 병렬 (4,3) 계수기의 블럭 다이그램이다. 이 방법은 입력이 n개일때도 확장할 수 있다.

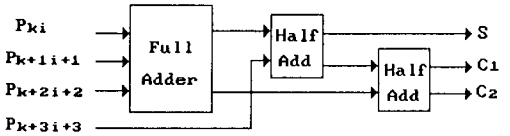


그림 3. 병렬 (4,3) 계수기 블럭 다이그램
Fig. 3. Block diagram of parallel (4,3) counter.

위에서 설명한 병렬 계수기을 이용하여 부분곱을 더하는 과정을 16×16 비트 곱셈기에 대하여 그림 4에 설명하였다. 여기에서 설명한 방법과 동일하게 32 32 비트 곱셈에 대해서도 확장할 수 있다.

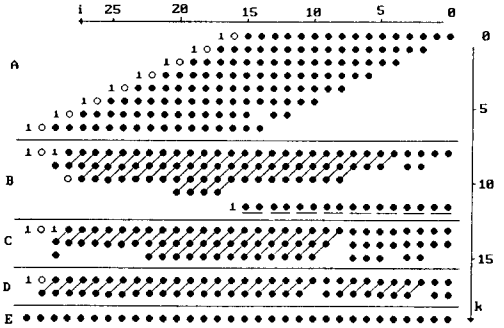
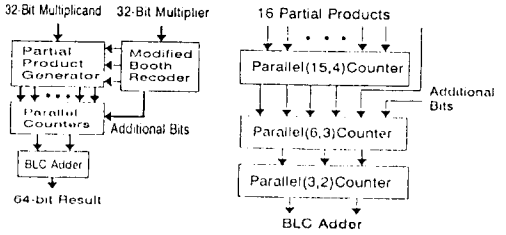


그림 4. 16×16 비트 곱셈 과정
Fig. 4. 16×16 bit multiplication process.

그림 4에서 부분곱의 '0'이나 '1'은 "·"로 표시했으며 부호 생성방법에서 설명한 것과 같이 부호 비트의 보수를 취했으며 "◦"로 표시 했다. Ⅲ절의 식 (7)에서 짝수항 $\overline{S_{13}}2^{62}, \overline{S_{12}}2^{60}, \cdots, \overline{S_1}2^{34}, \overline{S_0}2^{32}$ 항이 이에 해당한다. 그리고 부호 비트 왼쪽에 1을 더하는 것은 "1"로 표시하여 나타냈다. [1], [4] Ⅲ절의 식 (7)에서 홀수항 2^{63}, 2^{61}, 2^{59}, $\cdots$, 2^{33} 항이 이에 해당한다. A단에서 B단으로는 병렬 (8,4) 계수기와 다른 계수기가 사용된다. B단에서 5번째 열(k=12)은 비트-쌍 기록법에서 기록된 숫자가 -1, -2일 때 부분곱 생성시 피승수의 2진 보수를 구해야 하는데 부분곱 생성시 피승수의 보수만 취했을 뿐 1을 더하지 않았기 때문에 여기에서 더하는 것을 표시한 것이다. A단 k=0의 부분곱에 생성될 때 B단 i=0,1의 추가 비트가 생성되고 k=1일 때 i= 2,3의 추가 비트가 생성된다. k=7까지 8개의 부분곱이 생성되는 동안 i=15까지 추가 비트가 생성된다. 그리고 i=16의 1은 식(7)에서 2^{32} 항에 해당하는 것을 표시한다. E단은 D단에서 구한것을 마지막 결과 값을 얻기위해 BLC (Binary Lookahead Carry) Adder을 사용하여 32비트의 최종 곱셈 결과를 얻었다. 고속 가산기의 기본적인 형태인 CLA(Carry Lookahead Adder)는 가산 속도는 빠르나 가산 비트수가 많아지면 carry evaluation block이 복잡해지고 게이트의 입출력 부하의 문제로 부적합하다. [7]

Ⅴ. 곱셈기의 구성 및 시뮬레이션

2장에서 설명한 방법들을 사용하여 구성한 전체 곱셈기의 블럭 다이그램은 그림 5와 같다.



(a) 전체 블럭 다이그램 (b) 병렬계수기의 내부
(a) Total block diagram (b) In parallel counters

그림 5. 32 32 비트 곱셈기의 블럭 다이그램
Fig. 5. Block diagram of 32 32 bit multiplier.

그림 5 (a)에서 Bit-Pair Booth Recorder은 제어신호를 로직으로 구현한 그림 2의 회로가 들어간다. Partial Product Generator는 그림 1의 로직이 16개 들어가 제어신호에 따라 16개의 부분곱을 생성해 낸다. 그림 5에서 병렬 계수기로 들어가는 16개의 부분곱들은 모두 1의 보수이고 추가 비트는 표1과 그림 2에서 부분곱 생성시 발생하는 S1, S0들이 병렬 계수기로 들어가는 것을 보여준다. Parallel counter는 부분곱의 합을 구하는데 BLC 가산기에 들어가기전에 캐리의 전달을 줄여준다. 이 병렬계수기의 구조는 (b)의 그림과 같다. BLC Adder는 Parallel Counter에서 두 수의 합의로 줄어든 부분

곱을 더하여 최종 64 비트의 곱셈결과값을 얻는다.

그림 6은 본 논문에서 구현한 곱셈기의 시뮬레이션 결과를 text형태로 표현한 그림이다.

```
#cell2 * smlt_txt * 6 any 0 v8r4.4
# "9-May-94 GMT" "2:29:22 GMT" "9-May-94 GMT" "2:29:22 GMT" jhcho * .

Time = 0:0   [0 ns]
 md  =  'HFFFFFFFF  [0]  (input)(display)
 mr  =  'H55555555  [0]  (input)(display)
 mo --> 'HFFFFFFFFAAAAAAAB  [29.28]

 md --> 'HEEEEEEEE  [100]
 mr --> 'HAAAAAAAA  [100]
 mo --> 'H05B05B060B60B60C  [127.05]

 md --> 'H00000009  [200]
 mr --> 'H00000007  [200]
 mo --> 'H0000000000000003F  [226.11]
Time = 0:0 +300ns   [300 ns]
```

그림 6. 시뮬레이션 결과
Fig. 6. Simulation result.

그림 6에서 md는 피승수부, mr은 승수부를 나타내고 mo는 md와 mr의 곱셈값을 나타낸다. 'H는 16진수를 나타낸다. 피승수부가 -1로 16진수로 'HFFFFFFFF 이고 승수부가 16진수로 'H55555555 일때 곱셈기의 곱셈시간이 가장 길다. 시간이 0ns 에서 md, mr에 이 값들이 주어지면 29.28ns 후에 곱셈값 mo가 계산되어 나온다. 이 지연 시간은 약 30ns 이므로 이 곱셈기가 33Mhz로 동작할 수 있음을 보여준다. 이때의 critical path는 Booth recoder, 부분곱 생성기, 병렬(15,4)계수기, 병렬(6,3)계수기, 병렬(3,2)계수기,BLC가산기가 된다. 다른 수에 대한 곱셈 결과, 시간이 100ns일 때 md='HEEEEEEEE, mr=AAAAAAAA이 주어지면 곱셈값 mo가 127.05ns에 나오므로 이 때 걸리는 속도는 27.05ns 이다.

시뮬레이션은 VLSI Tech.사의 COMPAASS Tool을 사용하였으며 0.8μ 표준 CMOS 기술로 시뮬레이션 환경은 Vdd = 4.65V, Vss = 0.1V, 온도는 70℃의 조건하에서 이루어졌다.

Ⅵ. 결 론

디지탈 신호 처리에 있어서 중요한 역할을 하는 곱셈기를 비트-쌍 기록법의 알고리즘을 이용하여 설계하였다. 이 방법은 부분곱을 절반으로 감소시키며 곱셈 부호에 무관하게 계산하므로 곱셈기를 설계하는데 주로 사용되는 알고리즘이다.

본 논문에서 설계한 곱셈기는 이 알고리즘을 기반으로하여 여기에서 발생하는 부호확장은 부호 생성 방법을 적용하여 부호확장시 Routing 면적을 줄였다. 부분곱 생성시 필수적인 피승수부의 2진 보수 문제는 피승수부의 각 비트에 보수를 취하고 1을 더하는 과정은 부분곱을 더하는 과정에서 처리하여 부분곱을 생성하는 연산시간을 단축하였다. 부분곱을 더하는 과정은 Wallace 트리인 병렬 계수 방법으로 부분곱을 2개의 summands로 줄인 후 BLC 가산기로 최종 곱셈 결과값을 얻었다. 곱셈을 수행하는데 걸리는 지연시간은 30ns이다. 이것은 COMPASS Library가 제공하는 곱셈기가 같은 조건하에서 60ns가 걸리는 것과 비교하면 고속곱셈기임을 알 수 있다. 또한 설계한 곱셈기는 634 Gates 면적을 가지며 COMPASS 의 곱셈기의 크기는 673 Gates의 크기를 갖는다. 곱셈기의 크기는 약간 앞서지만 속도면에서는 큰 차이를 보인다.

만약 이 구조를 Full custom layout을 사용하여 설계 된다면 훨씬 빠른 동작속도를 얻을 것으로 기대된다.

參 考 文 獻

[1] (주)서두로직 " 디지탈 음성 통신용 ADPCM ASIC 개발 " p73 - 122

[2] 송홍종, 김영민 " 32비트 정수형 고속 병렬 곱셈기 구조" 1993년도 추계 종합 학술 발표 대회 논문집 pp. 583 - 587

[3] VoCarl Hamarcher, Zvonko G. Varneesic, Satwat G. Zaky " Computer Organization" McGraw-Hill Publishing Company pp. 282 - 291.

[4] Earl E. Swartzlander, Jr. " Computer Arithmatic" Valume 1 IEEE Computer Society Press pp. 96 - 154.

[5] Earl E1. Swartzlander, Jr. " Computer Arithmatic" Valume 2 IEEE Computer Society press pp. 189 - 249.

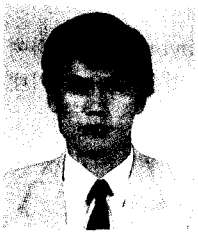
[6] 강정엽, 이원형, 권오영, 한탁돈 " 4-2 콤프레서를 이용한 승산기 모듈 생성기의 설계" 1993년도 추계 종합 학술 발표애회 논문집 pp. 388 - 392

[7] Neil H.E. Weste, Kamran Esharaghian "Principles of CMOS VLSI Design A system Perspective" Addion-Wesley Publishing Company pp. 320-331.

[8] Jalil Fadavi-Ardekani "MxN Booth Encoded Multiplier Generator Using Optimized Wallace Trees" IEEE Transations on Very Large Scale Intrgration (VLSI)Systems, VOL. 1, NO.2, June 1993 pp.120-125.

[9] Shoji Kawahito, Makoto Ishida, Tetsuro Nakamura, Michitaka Kameyama, and Tatsuo Higuchi "High-Speed Area-Efficient Multiplier Design Using Multiple-Valued Current-Mode Circuits" IEEE Transations on omputers, VOL. 43, NO.1, January 1994 pp.34-42.

著者紹介



金 榮 民(正會員)

1954年 4月 18日生. 1976年 2月 서울대학교 전자공학과 졸업(공학사). 1978年 2月 한국과학원 전기 및전자공학과 졸업(공학석사). 1978年 3月 ~ 1979年 7月 한국선박해양 연구소(주임 연구원). 1979年 8月 ~ 1982年 7月 국방과학연구소(연구원). 1986年 오하이오 주립대학교 전기공학과(공학박사). 1987年 1月~1988年 5月 노스 캐놀라이나 AT&T 주립대학교 교수. 1988年 6月 ~ 1991年 8月 한국전자통신연구소(실장). 1991年 9月 ~ 현재 전남대학교 전자공학과 교수. 주관심 분야는 영상처리, VLSI설계, 신경회로망, DSP 등임.

曺 鎭 鎬(準會員)

1967年 10月 5日生. 1993年 2月 전남대학교 전자공학과(공학사). 1993年 3月 ~ 현재 전남대학교 전자공학과 대학원 석사과정. 주관심 분야는 VLSI 설계, 영상처리, DSP 등임.