論文2000-37SD-5-8

# RB 연산을 이용한 고속 2의 보수 덧셈기의 설계

# (The Design of A Fast Two's Complement Adder with Redundant Binary Arithmetic)

李 泰 旭 *, 趙 相 福 **

(Tae-Wook Lee and Sang-Bock Cho)

요 약

본 논문에서는 CPF(Carry-Propagation-Free)의 특성을 갖는 RB(Redundant Binary)연산을 이용한 새로운 구조의 24비트 2의 보수 덧셈기를 설계하였다. TC2RB(Two's Complement to RB SUM converter)의 속도와 트랜지스터 개수를 줄이기 위해 MPPL(Modifed PPL) XOR/XNOR 게이트를 제안하고 고속 RB2TC(RB SUM to Two's Complement converter)를 사용한 두 가지 형태의 덧셈기를 제안하였다. 각 덧셈기의 특징을 살펴보면, TYPE 1 덧셈기는 VGS(Variable Group Select) 방식을 사용하여 덧셈기의 속도를 향상시켰으며 TYPE 2 덧셈기는 64비트 GCG(Group Change bit Generator)회로와 8비트 TYPE 1 덧셈기를 사용하여 속도를 향상시켰다. 64비트 TYPE 1 덧셈기의 경우 CLA와 CSA에 비해 각각 23.5%, 29.7%의 속도 향상을 TYPE 2 덧셈기의 경우 각각 41.2%, 45.9%의 속도 향상을 기대할 수 있다. 레이아웃된 24비트 TYPE 1과 TYPE 2 덧셈기의 전달지연 시간은 각각 1.4ns와 1.2ns로 나왔다. 제안한 덧셈기는 매우 규칙적인 구조를 가지고 있기 때문에 빠른 시간에 회로 설계 및 레이아웃이 가능하며 마이크로프로세서나 DSP 등과 같이 고속연산을 필요로 하는 경우에 적합하다.

**Abstract**

In this paper a new architecture of 24-bit two's complement adder is designed by using RB(Redundant Binary) arithmetic which has the advantage of CPF(Carry-Propagation-Free). A MPPL(Modified PPL) XOR/XNOR gate is applied to improve a TC2RB(Two's Complement to RB SUM converter) speed and to reduce the number of transistors, and we proposed two types adder which used a fast RB2TC(RB SUM to Two's Complement converter). The property of two types adder is followings. The improvement of TYPE 1 adder speed is archived through the use of VGS(Variable Group Select) method and TYPE 2 adder is through the use of a 64-bit GCG(Group Change bit Generator) circuit and a 8-bit TYPE 1 adder. For 64-bit, TYPE 1 adder can be expected speed improvement of 23.5%, 25.7% comparing with the CLA and CSA, and TYPE 2 adder can be expected 41.2%, 45.9% respectively. The propagation delay of designed 24-bit TYPE 1 adder is 1.4ns and TYPE 2 adder is 1.2ns. The implementation is highly regular with repeated modules and is very well suited for microprocessor systems and fast DSP units.

* 學生會員, ** 終身會員, 蔚山大學校 電氣電子 및 自動化 工學部

(School of Electrical and Automation Engineering)

※ 본 연구는 1999년 정보통신부 정보우수시범학교 지원사업과 산업자원부 반도체설계교육센터(IDEC)의 지원으로 수행되었음.

接受日字:1999年7月5日, 수정완료일:2000年3月23日

## Ⅰ. 서 론

덧셈은 마이크로프로세서나 DSP 등에 가장 중요한 연산이며 시스템의 속도에 큰 영향을 미친다. 따라서 전체 시스템의 속도를 향상시키기 위해서는 덧셈연산의 속도를 향상 시켜야한다. 특히 덧셈연산 성능을 좌우하는 캐리(Carry)의 전파를 줄이기 위한 연구가 활발

히 이루어져 왔으며, 그 대표적인 예로 RCA(Ripple-Carry-Adder), CLA(Carry-Lookahead-Adder), CSA (Carry-Select-Adder) 등을 들 수 있다[1][2]. 먼저 RCA 방식은 발생되는 캐리를 다음 단에 그대로 전달시켜 순차적으로 연산을 수행하는 방식이다. 이 방식의 장점은 회로를 작고 쉽게 구현 할 수 있다는 것이다. 그러나 비트 수가 커질수록 캐리의 전파 시간이 비트 수에 비례하여 증가하므로 덧셈 수행의 속도가 떨어진다. CLA 방식은 RCA방식의 단점인 캐리 전파 속도를 줄이기 위해 캐리를 블록 단위로 미리 계산하여 덧셈을 수행한다. 그리고 CSA는 입력되는 모든 캐리 값(0 또는 1)에 대해 캐리를 계산 한 후 그룹 캐리 신호에 의해 캐리값을 선택하는 방식이다.

최근에 들어서는 이러한 캐리 전파가 없는 CPF 특성을 이용한 덧셈기, 곱셈기 및 나눗셈기 등에 대한 연구가 활발히 이루어지고 있다[1][3][4][5]. CPF 특성을 이용한 연산은 기존의 0과 1을 사용하는 이진 수치계 대신 0과 1 그리고 -1을 가지는 RB수치계를 사용한다. RB 수치계를 이용한 덧셈의 경우 캐리의 전파는 비트수에 관계없이 항상 일정하므로 CPF 덧셈이 가능하다. 그러나 RB를 이용한 덧셈의 출력은 RB 수치계로 나오게 되므로 이를 이진 수치계로 변환하는 RB2TC회로가 필요하게 된다. 이러한 RB2TC 회로는 이진 덧셈기(Binary adder)에서 캐리 전파와 같은 변환 비트(Change bit)의 전파 지연 시간을 가지게 되어 RB연산을 사용한 시스템의 성능을 저하시킨다[4]. 따라서 본 논문에서는 이러한 변환 비트의 전파 속도를 줄이기 위한 새로운 형태의 RB2TC 회로를 제안하며 이를 덧셈기에 적용시켰다. 또한 덧셈 연산에서 가장 많이 사용되는 XOR/XNOR게이트의 성능 개선을 위해 MPPL XOR/XNOR게이트를 제안하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 RB 수치계을 이용한 덧셈 알고리즘에 대해 알아보고 Ⅲ장에서는 2의 보수 덧셈기를 효율적으로 구성하기 위한 하드웨어 구조에 대해 알아본다. 그리고 Ⅳ장에서는 제안된 덧셈기와 기존 덧셈기의 성능을 비교하며 24비트 덧셈기를 레이아웃하여 포스트 시뮬레이션(Post simulation)한다. 마지막으로 Ⅴ장에서 결론을 맺는다.

## Ⅱ. RB 수치계를 이용한 덧셈 알고리즘

본 논문에서 사용한 RB 표현은 Digit set $\{-1,0,1\}$을 갖는 Radix-2 Signed Digit(SD) 수치계에 속한다. 일반적으로 SD 수치계에서 $SD_i \in \{-1,0,1\}$는 두 개의 이진 비트 $(S_i D_i)$ (단, $S_i D_i \in \{0,1\}$ $S$와 $D$는 각각 Sign과 Digit 비트를 나타냄)로 인코딩 되며 $S$와 $D$는 표1에 의해 정의된다.

표 1. Sign과 Digit 비트의 정의
Table 1. Definition of sign and digit bit.

| $SD_i$ | S | D |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| -1 | 1 | 1 |

2의 보수 덧셈기의 덧셈과정을 살펴보면 다음과 같다. 덧셈기의 입력으로 2의 보수 A, B가 들어가며 이는 TC2RB를 통해 RB SUM으로 바뀐다. RB SUM은 앞서 나타낸 것과 같이 $S_i D_i$ 두 개의 이진 비트로 인코딩된 형태며 이는 다시 RB2TC를 통해 2의 보수형태의 덧셈값을 출력하게 된다. 이를 블록 다이어그램으로 나타내면 그림 1과 같다.

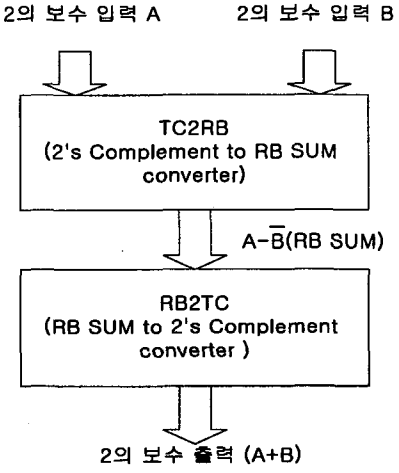



그림 1. RB 연산을 이용한 덧셈 과정
Fig. 1. Addition procedure using RB arithmetic.

1. TC2RB

2의 보수 $A=[a_{n-1}\,a_{n-2}\cdots a_1\,a_0]_2\,(a_i\in\{0,1\})$를 RB 형태의 대수 값으로 바꾸면 $-a_{n-1}*2^{n-1}$

$+\sum_{i=0}^{n-2} a_i * 2^i$ 가 된다. 그리고 2의 보수 입력 A와 B의 덧셈은 식(1)과 같이 나타낼 수 있다.

$$A+B=A-(-B)=A-(\overline{B}+1)=(A-\overline{B})-1 \quad (1)$$

$A=[a_{n-1}\ \ a_{n-2}\ \ \cdots\ \ a_1\ \ a_0]$

$B=[b_{n-1}\ \ b_{n-2}\ \ \cdots\ \ b_1\ \ b_0]$

$\overline{B}$는 $B$의 반전을 나타냄)

식(1)에서 SD를 식(2)와 같이 정의하면 식(3)으로 나타낼 수 있다.

$$SD \equiv A-\overline{B} \equiv (S,D) \quad (2)$$

(단,

$SD=[sd_{n-1}, sd_{n-2}, \cdots, sd_1, sd_0]$

$=[s_{n-1}d_{n-1}, s_{n-2}d_{n-2}, \cdots, s_1d_1, s_0d_0]$

$sd_i$는 Signed Digit number를 나타냄

$s_i, d_i$는 각각 Sign 비트 와 Digit 비트를 나타냄)

$$A+B=SD_{RB}-1 \quad (3)$$

식(3)으로부터 A+B는 SD 수치계로 나타낼 수 있으며, RB 수치계는 Radix-2 SD 수치계이므로 A+B는 RB 수치계로 바꿀 수 있음을 알 수 있다. 각 비트에 대한 $S_i$와 $D_i$ 의 부울 식을 표2 에 의해 구하면 식(4)와 같이 나타내어진다.

$$S_i=\overline{a_i}\cdot\overline{b_i}=\overline{a_i+b_i}$$
$$D_i=\overline{a_i}\cdot\overline{b_i}+a_ib_i=\overline{a_i\oplus b_i} \quad (4)$$
(단, $0 \le i < n-1$ )

표 2. Sign과 Digit의 진리표
Table 2. Truth table of sign and digit.

| $a_ib_i$ | $S_i$ | $D_i$ |
|---|---|---|
| 0 0 | 1 | 1 |
| 0 1 | 0 | 0 |
| 1 0 | 0 | 0 |
| 1 1 | 0 | 1 |

여기서 주의해야 할 점은 A와 B가 2의 보수형태를 취하고 있으므로 $a_{n-1}$, $b_{n-1}$ 는 부호 비트가 되며 그것의 대수값은 $-(a_{n-1}*2^{n-1})$, $-(b_{n-1}*2^{n-1})$이 되어 음의 부호를 가진다. 따라서 부호 비트에서의 $s_{n-1}$, $d_{n-1}$ 은 다음과 같이 생각할 수 있다.

① $a_{n-1}=1$, $b_{n-1}=1$ 일 때

$a_{n-1}-\overline{b_{n-1}}(=sd_{n-1})=-2^{n-1}$

② $a_{n-1}=0$, $b_{n-1}=0$ 일 때

$a_{n-1}-\overline{b_{n-1}}(=sd_{n-1})=2^{n-1}$

③ 그 이외의 경우 $a_{n-1}-\overline{b_{n-1}}(=sd_{n-1})=0$

위의 세 가지 경우를 조합하면 표3의 결과를 얻을 수 있으며 $s_{n-1}, d_{n-1}$은 식(5)가 된다.

표 3. $s_{n-1}$과 $d_{n-1}$의 진리표
Table 3. Truth table of $s_{n-1}$ and $d_{n-1}$.

| $a_{n-i}b_{n-i}$ | $s_{n-1}$ | $d_{n-1}$ |
|---|---|---|
| 0 0 | 0 | 1 |
| 0 1 | 0 | 0 |
| 1 0 | 0 | 0 |
| 1 1 | 1 | 1 |

$$s_{n-1}=a_{n-1}b_{n-1}$$
$$d_{n-1}=\overline{a_i}\cdot\overline{b_i}+a_ib_i=\overline{a_{n-1}\oplus b_{n-1}} \quad (5)$$

위의 과정을 예를 들어 설명하면 그림2와 같다.

$A=[10000110101]_2(-971)$
$B=[11000000110]_2(-506)$ 이라면, 식(2)에의해

$A\ [\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1]_2$
$-\overline{B}\ [\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1]_2$
가 된다.

$A-\overline{B}=SD_{RB}=[\bar{1}0\ 1\ 1\ \bar{1}0\ 0\ \bar{1}1\ 0\ 0]_{RB}(-1476)$

단, 첨자 2와 RB는 각각 이진수치계와 RB 수치계임을 나타낸다.

그림 2. TC2RB 생성 과정
Fig. 2. Procedure of TC2RB generation.

2. RB2TC

RB 비트를 2의 보수 비트로 변환하기 위한 알고리즘을 생각해 보자. 이진 수치계와 SD 수치계의 관계는 식(6)과 같이 나타낼 수 있다.

$$SD_{RB}=SD^+{}_{RB}-SD^-{}_{RB} \quad (6)$$

(단,

$SD_{RB} = [sd_{n-1}\, sd_{n-2} \cdots sd_1\, sd_0]$

$SD^{+}{}_{RB} = \sum_{sd_i=1} sd_i * 2^i$

$SD^{-}{}_{RB} = \sum_{sd_i=-1} |-sd_i| * 2^i$

첨자 +와 -는 각각 SD 수치계에서 양의 비트와 음의 비트의 합을 나타냄.)

여기서 $(-1)_{RB} = (-1\ 1)_{RB}$로 나타낼 수 있으므로 식(6)은 식(7)과 같이 쓸 수 있다.

$$\begin{aligned} SD_2 &= \sum_{sd_i=1} sd_i * 2^i - \sum_{sd_i=-1} |-sd_i| * 2^i \\ &= \left(\sum_{sd_i=1} sd_i * 2^i + \sum_{sd_i=-1} |-sd_i| * 2^i\right) - \sum_{sd_i=-1} |-sd_i| * 2^{i+1} \\ &= \sum_{sd_i=-1} |sd_i| * 2^i - \sum_{sd_i=-1} |sd_i| * 2^{i+1} \\ &= SD_i{}^{+}{}_2 - SD_{i+1}{}^{-}{}_2 \end{aligned} \tag{7}$$

(단,

$SD_i{}^{\pm}{}_2 = \sum_{sd_i=\pm 1} |sd_i| * 2^i$

$SD_{i+1}{}^{-}{}_2 = \sum_{sd_i=-1} |sd_i| * 2^{i+1}$

$0 \le i \le n-1$)

위 식에서 보듯이 SD 수치계를 이진 수치계로 변환시키는 방법은 SD 수치계에서 양과 음의 비트를 가지는 $SD_i{}^{\pm}{}_2$ 과 음의 비트를 가지는 $SD_i{}^{-}{}_2$를 구한 후 음의 비트를 가지는 $SD_i{}^{-}{}_2$를 왼쪽으로 1비트 이동시켜($SD_{i+1}{}^{-}{}_2$이 됨) 빼주면 된다. 이때 주의해야 할 점은 $SD_2$ 의 최상위 비트(MSB)는 부호 비트가 되므로 빼지지 않고 그대로 $|sd_{n-1}| * 2^n$ 이 된다. 마지막으로 최종 출력 A+B를 구하기 위해서는 $SD_2 - 1$ 을 해주면 된다. 위 알고리즘을 예를 들어 설명하면 그림 3과 같다.

위의 예를 살펴보면 RB 수치계에서 이진 수치계로 변환시($SD_{RB}$ to $SD_2$) 뺄셈의 과정을 거쳐야 하는데 $SD_{i+1}{}^{-}2$ = 1인 bit에서 왼쪽으로 자리 내림(borrow)이 발생하여 $SD_i{}^{\pm}{}_2$ = 1인 곳에서 멈추게 되며, 만일 $SD_i{}^{\pm}2$ = -1이 되면 자리 내림은 멈추지 않고 $SD_i{}^{\pm}2$ = 1이 될 때까지 계속 진행하게 된다. 그리고 자리 내림이 진행되는 영역안의 $SD_i{}^{\pm}{}_2$의 비트는 이진수로 변환시 반전되며(위의 예에서 점선으로 둘러 쌓인 부분), 나머지 영역은 그대로 임을 알 수 있다. 마지막으로 $SD_2 - 1$의 경우도 위와 같이 생각할 수 있으며, 특히 최하위 비트(LSB)( $s_0$)는 항상 1을 빼주어야 하므로 언제나 반전됨을 알 수 있다.

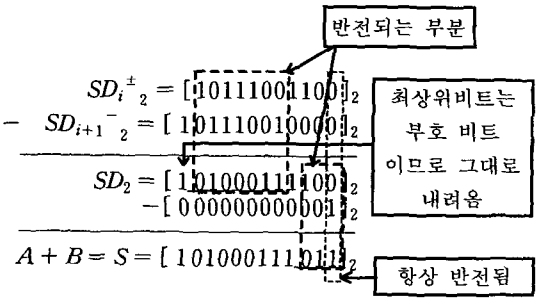



그림 3. RB2TC 생성 과정
Fig. 3. Procedure of RB2TC generation.

위의 결과에 따라 변화되는 부분을 나타내는 변환 비트(C)를 다음과 같이 정의 할 수 있다.

**정의 1** : 변환 비트의 정의

1) 전단의 $SD_i{}^{\pm}{}_2$ = -1 일 때 $C_{i+1}=1$ 이 된다.

2) 전단의 $SD_i{}^{\pm}{}_2$ = 1 일 때 $C_{i+1}=0$ 이 된다.

3) 전단의 $SD_i{}^{\pm}{}_2$ = 0 일 때 $C_{i+1}=C_i$ 이 된다.

4) 그리고 $C_0=1$ 이다.

위 정의에 따른 C와 최종 출력 A+B(S)의 부울 식은 식(8)과 같다.

$$\begin{aligned} C_{i+1} &= s_i + \overline{d_i} C_i = s_i d_i + \overline{d_i} C_i \\ A+B=S &= SD_i{}^{\pm}{}_2 \oplus C_i \end{aligned} \tag{8}$$

(단, $s_i, d_i$ 는 각각 Sign과 Digit 비트를 나타낸다.)

위의 정의에 따라 RB2TC를 다시 계산하면 앞의 결과와 일치함을 알 수 있다.

$SD_{RB} = [\overline{1}\,0\,1\,1\,1\,0\,0\,\overline{1}\,1\,0\,0]_{RB}$

$SD_i{}^{\pm}{}_2 = [101110011100]_2$

$C_i = [111111110111]_2$

$S_i = SD_i{}^{\pm}{}_2 \oplus C_i = [101000111011]_2$

## Ⅲ. 하드웨어 구조

본 장에서는 앞장에서 제안한 알고리즘을 적용하여 고속 2의 보수 덧셈기를 설계하며 전체 덧셈기의 구조가 매우 규칙적이므로 덧셈기의 비트 수를 쉽게 확장할 수 있다. 전체 덧셈기는 2가지 TYPE으로 구현한다. 1절에서는 TYPE 1 덧셈기의 구조에 대해 기술하고, 2절에서는 TYPE 2 덧셈기의 구조에 대해 언급한다.

1. TYPE 1 덧셈기의 구조

II장에서 제안된 알고리즘을 이용하면, n비트 덧셈기는 다음과 같이 2의 보수 입력을 RB SUM 형태로 바꾸어주는 부분(TC2RB)과 RB SUM을 2의 보수 형태로 바꾸어 주는 부분(RB2TC)으로 구성됨을 알 수 있다. 각 블록에 대한 설명은 다음과 같다.

1) TC2RB의 회로

TC2RB는 그림 4와 같이 2-입력 NOR 게이트, AND 게이트 그리고 XNOR 게이트로 구성되며, n 비트의 덧셈기 구성시 n-1개의 NOR 게이트와 1개의 AND 게이트 그리고 n개의 XNOR 게이트가 필요함을 알 수 있다. 따라서 본 논문에서는 RB SUM 단에서의 게이트 수를 최소화하기 위해서 PPL(Push Pull Logic)을 변형한 새로운 형태의 XOR/XNOR 게이트를 제안한다.

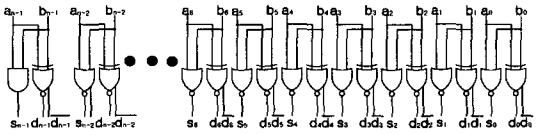


그림 4. TC2RB 회로도
Fig. 4. Circuit diagram of TC2RB.

① MPPL XOR/XNOR 게이트

제안된 MPPL은 패스 트랜지스터(Pass Transistor)로 구성되어 있다. 패스 트랜지스터는 복잡한 로직을 쉽게 구현할 수 있는 장점이 있으나 NMOS의 경우 완전한 "1"(또는 HIGH)을 통과시키지 못하며, PMOS의 경우 완전한 "0"(또는 LOW)을 통과시키지 못하는 단점이 있다. 그러나 제안된 MPPL은 NMOS M5와 PMOS M6를 사용하여 이러한 단점을 보완 하였다. 아래에 MPPL XOR/XNOR의 동작 원리를 나타내었다.

**동작 원리 :**

1. a=0, b=0 일 때

M1과 M2가 ON되며 a, b의 값이 PMOS M1과 M2를 통해 XOR 출력에 전달된다. XOR 출력은 PMOS 전달 특성에 의해 완전한 "0"이 전달되지 못한다. 한편 M3와 M4는 OFF되어 XNOR 출력에 영향을 미치지 못하나 XOR 출력 "0"에 의해 PMOS M6가 ON이 된다. PMOS M6는 VDD의 값을 손실 없이 XNOR 출력으로 흘려 보내므로 XNOR 출력은 완전한 "1"이 된다. XNOR 출력에서의 "1"은 다시 NMOS M5를 ON 시켜 GND를 손실없이 XOR 출력으로 내보내게 되므로 완전한 "0"의 값이 XOR 출력으로 나가게 된다.

2. a=0, b=1 일 때

M1은 OFF되나 M2가 ON되어 b의 값이 PMOS M2를 통해 XOR 출력으로 나가게 된다. XOR 출력은 PMOS 전달 특성에 의해 완전한 "1"이 된다. 한편 M3는 ON되나 N4는 OFF되어 a의 값이 NMOS M3를 통해 XNOR 출력으로 나가게 된다. XNOR 출력 역시 NMOS 전달 특성에 의해 완전한 "0"이 된다.

3. a=1, b=0 일 때

M2는 OFF되나 M1이 ON되어 a의 값이 PMOS M1를 통해 XOR 출력으로 나가게 된다. XOR 출력은 PMOS 전달 특성에 의해 완전한 "1"이 된다. 한편 M4는 ON되나 N3는 OFF되어 b의 값이 NMOS M4를 통해 XNOR 출력으로 나가게 된다. XNOR 출력 역시 NMOS 전달 특성에 의해 완전한 "0"이 된다.

4. b=1, b=1 일 때

M1과 M2가 OFF되어 XOR 출력에 영향을 미치지 못하나, M3와 M4가 ON되어 XNOR 출력은 "1"이 된다. XNOR 출력의 "1"은 M5를 ON시키므로 완전한 XOR 출력은 완전한 "0"이 되며, 이는 다시 M6를 ON시켜 XNOR 출력은 완전한 "1"이 된다.

위의 과정을 요약하면 표 4와 같으며 그림 5는 제안된 2-입력 MPPL XOR/ XNOR 게이트의 회로도이다. 구성된 로직의 시뮬레이션 결과는 그림 6과 같다.

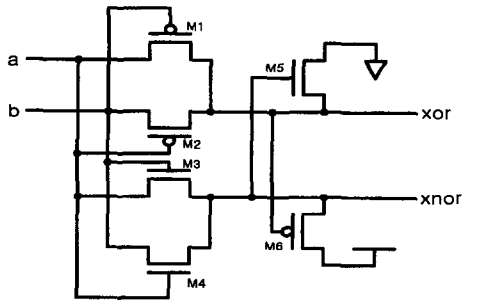



그림 5. 제안된 2-입력 MPPL XOR/XNOR 게이트
Fig. 5. Proposed 2-Input MPPL XOR/XNOR gate.

표 4. 입력에 따른 TR의 동작 상태
Table 4. Operation of TR with inputs.

| 입력 상태 | | 출력 상태 | | 동작 TR |
|---|---|---|---|---|
| a | b | xor | xnor | |
| 0 | 0 | 0 | 1 | M1, M2, M5, M6 |
| 0 | 1 | 1 | 0 | M2, M3 |
| 1 | 0 | 1 | 0 | M1, M4 |
| 1 | 1 | 0 | 1 | M3, M4, M5, M6 |

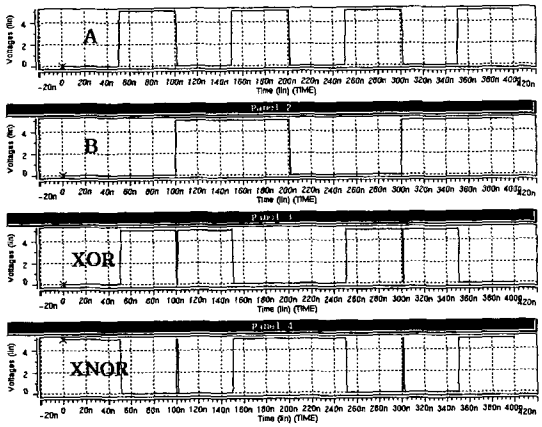



그림 6. MPPL XOR XNOR 게이트의 시뮬레이션 결과
Fig. 6. Simulation result of MPPL XOR/XNOR gate.

MPPL XOR/XNOR게이트는 PPL구조와 비슷한 모양을 가지나 상보 입력이 필요 없으므로 2개의 인버터를 줄일 수 있으며 출력으로 XOR/XNOR 동작을 동시에 수행 할 수 있으므로 상보 입력이 필요한 CPL, PPL, SRPL등에 적용하기 쉽다. 또한 트랜지스터의 수가 표 5에서 알 수 있듯이 다른 로직에 비해 적으므로 면적을 최소화 할 수 있는 장점이 있다.

표 5. 로직에 따른 트랜지스터 수 비교(With buffer)
Table 5. Comparisons of the number of transistor with buffer.

| | PMOS | NMOS | 전체 TR | Output |
|---|---|---|---|---|
| CMOS | 6 | 6 | 12 | XOR |
| CPL | 6 | 8 | 14 | XOR/XNOR |
| PPL | 7 | 7 | 14 | XOR/XNOR |
| MPPL | 5 | 5 | 10 | XOR/XNOR |

2) RB2TC

RB2TC 회로는 크게 CBG(Change Bit Generator)와 TCC(Two's Complement Converter)로 구성된다. CBG 단은 RB SUM의 출력 $S_i, D_i$를 받아 변환 비트를 생성해 주며 TCC 단은 $C_i, D_i$ 의 입력을 받아 2의 보수 형태의 SUM 값을 출력한다.

① CBG 단

변환 비트는 식(8)과 같으며, 이를 정리하면 아래와 같다.

$$\begin{aligned} C_0 &= 1 \\ C_1 &= s_0 + \overline{d_0} C_0 = s_0 d_0 + \overline{d_0} C_0 \\ C_2 &= s_1 + \overline{d_1} C_1 = s_1 d_1 + \overline{d_1} C_1 \\ C_3 &= s_2 + \overline{d_2} C_2 = s_2 d_2 + \overline{d_2} C_2 \\ &\vdots \\ C_{n-1} &= s_{n-2} + \overline{d_{n-2}} C_{n-2} = s_{n-2} d_{n-2} + \overline{d_{n-2}} C_{n-2} \\ C_n &= s_{n-1} + \overline{d_{n-1}} C_{n-1} = s_{n-1} d_{n-1} + \overline{d_{n-1}} C_{n-1} \end{aligned} \quad (9)$$

위 식에서 보는 바와 같이 변환 비트는 최상위 비트에서 최하위 비트로의 신호 전파를 발생시키며, 이 신호 전파에 의해 전체 덧셈기의 성능은 크게 저하된다. 따라서 본 절에서는 변환 비트의 전파 속도를 줄이기 위해 VGS방식을 적용한 새로운 형태의 CBG회로를 제안한다.

식(9)를 변환 비트 입력에 따른 변환 비트 출력으로 나타내면 다음과 같이 나타낼 수 있다.

1) $C_i = 0$일 때

$$\begin{aligned} C_{i+1}{}^0 &= s_i \\ C_{i+2}{}^0 &= s_{i+1} + \overline{d_{i+1}}\, C_{i+1}{}^0 \\ C_{i+3}{}^0 &= s_{i+2} + \overline{d_{i+2}}\, C_{i+2}{}^0 \\ &\vdots \\ C_k{}^0 &= s_{(k-1)} + \overline{d_{(k-1)}}\, C_{(k-1)}{}^0 \end{aligned}$$

2) $C_i = 1$일 때

$$\begin{aligned} C_{i+1}{}^1 &= s_i + \overline{d_i} \\ C_{i+2}{}^1 &= s_{i+1} + \overline{d_{i+1}}\, C_{i+1}{}^1 \\ C_{i+3}{}^1 &= s_{i+2} + \overline{d_{i+2}}\, C_{i+2}{}^1 \\ &\vdots \\ C_k{}^1 &= s_{(k-1)} + \overline{d_{(k-1)}}\, C_{(k-1)}{}^1 \end{aligned} \quad (10)$$

(단, $k \le n$)

위의 2가지 경우를 고려하여 $C_i$에서 $C_k$로의 그룹 CBG(Group CBG) $C_{k:i}$를 정의하면 식(11)과 같다.

$$C_{k:i} = C_{k:i}{}^{0}\overline{C_i} + C_{k:i}{}^{1}C_i \quad (11)$$

식(11)에서 $C_{k:i}{}^{0}$은 $C_{k:i}{}^{1}$에 포함되므로 $\overline{C_i}$는 무시 가능하다. 따라서 식(11)은 식(12)와 같이 쓸 수 있으며 첨자 $k, i$ 그리고 그룹 크기(Group Size)는 표6에 나타내었다.

$$C_{k:i} = C_{k:i}{}^{0} + C_{k:i}{}^{1}C_i \quad (12)$$

(단, $C_{k:i}$는 $C_{i+2}$, $C_{i+3}$, …, $C_k$를 나타냄.)

위 식과 표 6을 식(9)에 적용시켜 전개시키면 아래와 같이 나타낼 수 있다.

$C_0 = 1$

$C_1 = S_0 + \overline{D_0}C_0$

$C_2 = S_1 + \overline{D_1}C_1$

GROUP 1

$C_3 = C_3{}^{0} + C_3{}^{1}C_2, \quad C_4 = C_4{}^{0} + C_4{}^{1}C_2$

GROUP 2

$C_5 = C_5{}^{0} + C_5{}^{1}C_4, \quad C_6 = C_6{}^{0} + C_6{}^{1}C_4,$

$C_7 = C_7{}^{0} + C_7{}^{1}C_4$

(13)

GROUP 3

$C_8 = C_8{}^{0} + C_8{}^{1}C_7, \quad C_9 = C_9{}^{0} + C_9{}^{1}C_7,$

$C_{10} = C_{10}{}^{0} + C_{10}{}^{1}C_7, \quad C_{11} = C_{11}{}^{0} + C_{11}{}^{1}C_7$

GROUP 4

$C_{12} = C_{12}{}^{0} + C_{12}{}^{1}C_{11}, \quad C_{13} = C_{13}{}^{0} + C_{13}{}^{1}C_{11},$

$C_{14} = C_{14}{}^{0} + C_{14}{}^{1}C_{11}, \quad C_{15} = C_{15}{}^{0} + C_{15}{}^{1}C_{11},$

$C_{16} = C_{16}{}^{0} + C_{16}{}^{1}C_{11}$

⋮

위 식을 살펴보면 비트가 커질수록 그룹의 크기가 커져 변환 비트 전파속도를 줄일 수 있음을 알 수 있다. 또한 변환 비트 생성(CG; Change bit Generator)의 부울 방정식이 모두 A+BC형태를 가지므로 회로를 매우 쉽게 구성할 수 있는 장점이 있다. 그림7에 제안한 CBG회로의 구조를 나타내었다.

표 6. 첨자 $i, k$, 그룹 크기와의 관계

Table 6. Relationship of subscript $i, k$ and group size.

| $i$ | 2 | 4 | 7 | 11 | 16 | 22 | … | 137 |
|---|---|---|---|---|---|---|---|---|
| $k$ | (3,4) | (5,6,7) | (8,…,11) | (12,…,16) | (17,…,22) | (23,…,29) | … | (138,…,154) |
| Group Size | 2 | 3 | 4 | 5 | 6 | 7 | … | 17 |

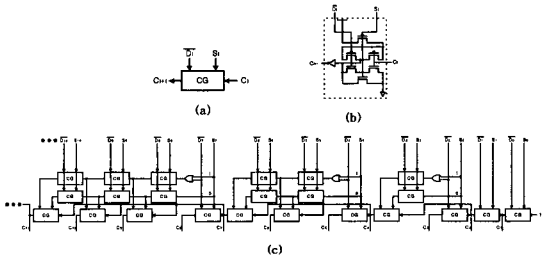



그림 7. (a)&(b) CG 셀; (c) TYPE 1 방식의 CBG 회로

Fig. 7. (a)&(b) CG Cell; (c) CBG circuit of TYPE 1.

② TCC

변환 비트 생성단에서 나온 $C_i$와 RB SUM단에서 나온 $D_i$를 이용해 덧셈 결과를 2의 보수 형태로 출력하는 부분이다. A+B의결과는 식(8)에 나타내어지며, Sign 비트는 다른 부가 회로 없이 최상위 변환 비트 ( $C_n$ )를 그대로 출력시키면 된다. 그림 8에 TCC 회로를 나타내었다.

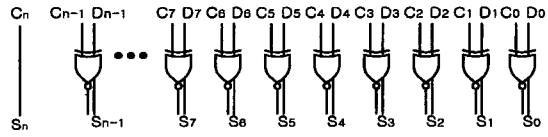



그림 8. TCC 회로도

Fig. 8. Citcuit diagram of TCC.

2. TYPE 2 덧셈기의 구조

앞 절에서 구현한 덧셈기는 하나의 변환 비트가 여러 게이트를 구동시켜야 되므로 비트 수가 커질수록 팬 아웃(fan-out)이 큰 인버터를 필요로 하며, 팬 아웃

이 어느 이상 되면 회로 전체의 전달지연시간이 길어지게 된다. 따라서 그룹의 크기 증가에 따라 구동능력이 큰 버퍼를 삽입하여야하는 단점이 있다.

따라서 본 절에서는 TYPE 1 방식을 응용한 8비트 CBG회로를 사용하여 변환 비트의 전파 속도를 줄인다. TYPE 2 방식의 변환 비트식은 식(14)와 같으며 선택 신호로 $d_i, \overline{d_i}$를 사용하므로 반전된 신호가 필요 없으며 MUX를 사용할 수 있는 장점이 있다.

$C_0 = 1$

$C_1 = S_0 D_0 + \overline{D_0} C_0$

$C_2 = S_1 D_1 + \overline{D_1} C_1$

GROUP 1

$C_3 = C_3^0 D_3 + C_3^1 C_2, \quad C_4 = C_4^0 D_4 + C_4^1 C_2,$

$C_5 = C_5^0 D_5 + C_5^1 C_4$

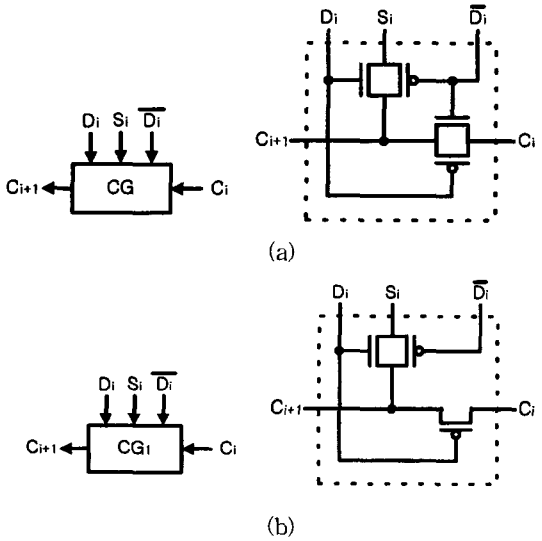
(14)

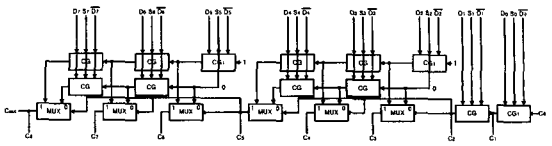
GROUP 2

$C_6 = C_6^0 D_6 + C_6^1 C_4, \quad C_7 = C_7^0 D_7 + C_7^1 C_4,$

$C_8 = C_8^0 D_8 + C_8^1 C_7$

그림 9는 MUX를 사용한 변환 비트 생성 셀과 8비트 CBG의 회로도를 나타낸다.



(c)

그림 9. (a) CG 셀; (b) 입력이 1인 CG 셀; (c) 제안된 8비트 CBG 회로도

Fig. 9. (a) CG Cell; (b) CG Cell with input 1; (c) Proposed 8-bit CBG circuit

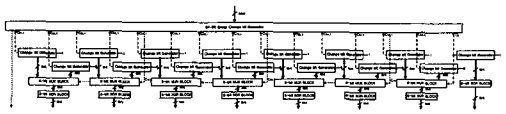


그림 10. 64비트 2의 보수 덧셈기의 블록도

그림 10.64비트 2의 보수 덧셈기의 블록도

Fig. 10.Block diagram of 64-bit two's complement adder.

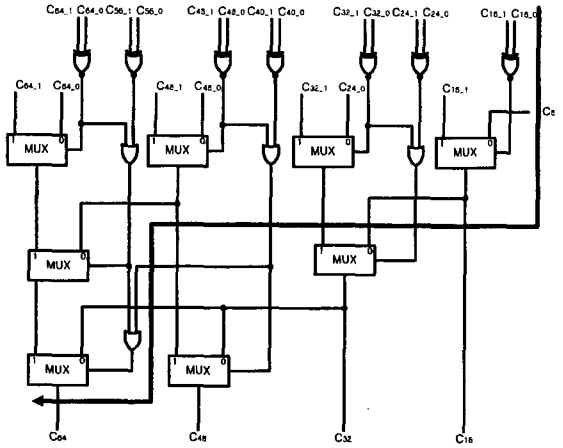



그림 11. 64비트 GCG회로

Fig. 11. Circiut diagram of 64-bit GCG.

그림 10은 64비트 2의 보수 덧셈기의 전체 블록도이며 전체적으로 캐리 선택 방식을 응용한 구조를 가지고 있다. 변환 비트 0과 1의 입력을 갖는 8비트 CBG에 의해 생성된 변환 비트는 64비트 GCG에 들어가 $C_{16}$, $C_{32}$, $C_{48}$, $C_{64}$를 생성한다. 한편 $C_{24}$, $C_{40}$, $C_{56}$은 $C_{16}$, $C_{32}$, $C_{48}$에 의해 생성되고 이렇게 생성된 변환 비트는 Digit 비트와 XOR함으로서 최종 합을 구한다. 64비트 GCG회로는 그림 11에 나타낸바와 같이 $C_{64}$를 생성하는데 $C_8$보다 3개의 MUX만 더 거치면 되므로 변환 비트의 생성속도와 전체 덧셈기의 속도를 줄일 수 있다.

## Ⅳ. 덧셈기의 성능비교 및 레이아웃

### 1. 성능비교

2의 보수 덧셈기의 성능 비교를 위해 2-1 MUX의 게이트 지연은 2-입력 NOR나 NAND 그리고 XOR 게이트의 지연 시간과 같다고 가정하였다.[1] 또한 임계경로(Critical Path)를 줄이기 위해 비트가 16비트보다 적을 때는 블록 크기를 4비트로 하였으며 16비트보다 클 때는 8비트로 하였다.

그림12에 제안한 덧셈기와 기존 덧셈기의 비트수에 따른 게이트 지연을 비교하여 나타내었다. 그림12에 나타낸 게이트 지연은 임계경로에서의 논리 게이트 숫자이며 정확한 게이트 지연의 계산을 위해 CLA[6]나 CSA[7]의 경우는 임계경로에서의 게이트 숫자를 2로 나눈 후 이를 1.5배하여 나타내었다. 이는 실제 CLA나 CSA 덧셈기 구현시 게이트들의 지연은 2-입력 논리 게이트 지연의 1.5배로 나타낼 수 있기 때문이다.[1]

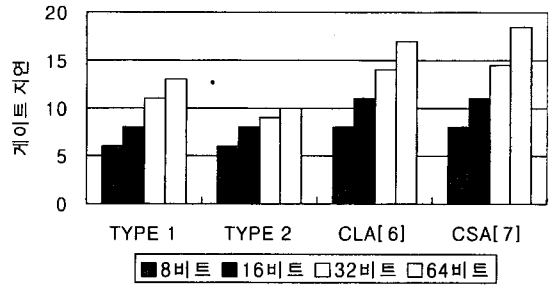



그림 12. 비트수에 따른 게이트 지연 비교
Fig. 12. Comparison of gate delay with bits.

8비트 TYPE 1과 TYPE 2 덧셈기는 CLA와 CSA에 비해 25%의 속도 향상을 기대할 수 있으며 16비트인 경우 CLA와 CSA에 비해 27.3%의 속도 향상을 기대할 수 있다. 또한 32비트 TYPE 1 덧셈기는 CLA에 비해 21.4%, CSA에 비해서는 24.1%의 속도 향상을 기대할 수 있으며 TYPE 2 덧셈기는 CLA와 CSA에 비해 각각 35.7%와 37.9%의 속도 향상을 기대할 수 있다. 마지막으로 64비트인 경우 TYPE 1 덧셈기는 CLA와 CSA에 비해 각각 23.5%, 29.7%의 속도 향상을 TYPE 2 덧셈기는 각각 41.2%, 45.9%의 속도 향상을 기대할 수 있다.

그림 13은 각 덧셈기의 비트수에 따른 트랜지스터 수를 비교하여 나타내었다. 그림에서 알 수 있듯이 TYPE 1 덧셈기의 트랜지스터 개수는 CSA에 비해 58.2%~64%가 적으며 CLA와는 비슷하다. 그러나 TYPE 2 덧셈기의 트랜지스터 개수는 CSA에 비해 35.1%~47.3% 감소하나 CLA보다는 41.4%~62%의 많은 트랜지스터를 필요로 하여 면적이 그만큼 커지게 된다. 따라서 TYPE 2 덧셈기는 면적과 게이트 지연을 고려한 설계를 필요로 한다.

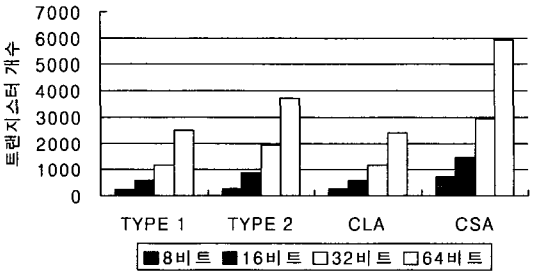



그림 13. 비트수에 따른 트랜지스터 개수 비교
Fig. 13. Comparison of the number of transistors with bits.

### 2. 24비트 TYPE 1 TYPE 2 덧셈기의 레이아웃

본 절에서는 앞 절에서 비교된 성능을 재확인하기 위해 24비트 TYPE 1, TYPE 2 덧셈기(24비트 2의 보수 덧셈기)를 LG 0.6㎛ 1-poly 3-metal CMOS 공정과 CADENCE Tool을 사용하여 레이아웃 하였다. 레이아웃 후 정확한 지연시간을 측정하기 위한 방법으로

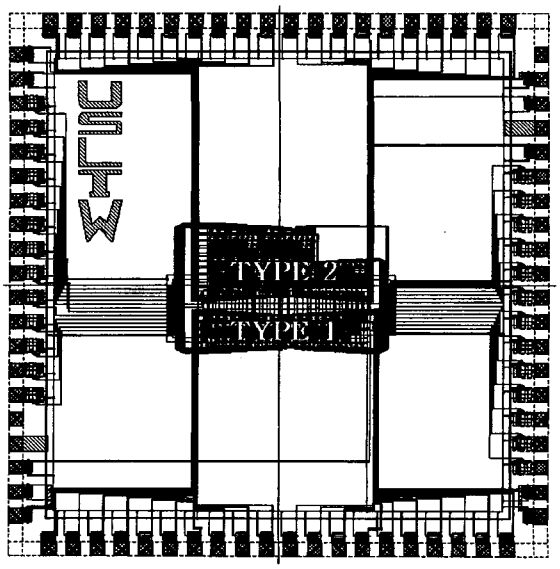



그림 14. 24 비트 2의 보수 덧셈기의 레이아웃

HSPICE를 사용하여 포스트 시뮬레이션 하였으며 이때 입력 신호는 최악 조건의 덧셈 시간을 측정할 수 있도록 입력하였다. 그림14는 전체 덧셈기의 레이아웃이다. 설계된 TYPE 1과 TYPE 2 덧셈기는 24비트의 입력과 25비트 출력을 가진다. TYPE 1 덧셈기의 덧셈 수행 시간은 1.4ns이며 코어의 크기는 약 $1.165 \times 0.189 mm^2$이다. 또한 TYPE 2 덧셈기의 덧셈 수행 시간은 1.2ns이며 코어의 크기는 약 $1.135 \times 0.425 mm^2$이다. 전체 덧셈기의 패드(pad)부분을 제외한 크기는 $1.616 \times 0.924 mm^2$이다. 표 7에 전체 덧셈기의 성능을 나타내었다.

표 7. 덧셈기의 성능비교
Table 7. Performance comparison of adders.

| | Bit | Technology | Delay(ns) | Area($\mu m \times \mu m$) |
|---|---|---|---|---|
| Ref.[8] | 32 | 0.5$\mu m$ BiCMOS | 1.7 | |
| Ref.[9] | 16 | 1.5$\mu m$ CMOS | 12.2 | |
| Ref.[10] | 32 | | 12.2 | 2369×611 |
| Ref.[11] | 16 | 1.5$\mu m$ CMOS | 2.95 | 536×536 |
| Ref.[12] | 32 | 0.9$\mu m$ CMOS | 3.1 | |
| Ref.[13] | 32 | 1.2$\mu m$ CMOS | 7.4 | 1704×1580 |
| Ref.[14] | 16 | 2$\mu m$ BiCMOS | 20 | 184×713 |
| Ref.[15] | 32 | 0.25$\mu m$ CMOS | 1.5 | 2160×260 |
| Ref.[16] | 32 | 3$\mu m$ CMOS | 20 | 1107×1107 |
| Ref.[17] | 32 | 1.2$\mu m$ CMOS | 2.1 | |
| TYPE 1 | 24 | 0.6$\mu m$ CMOS | 1.4 | 1165×204 |
| TYPE 2 | 24 | " | 1.2 | 1135×425 |

## V. 결 론

본 논문에서는 CPF의 특성을 갖는 RB연산을 이용하여 24비트 2의 보수 덧셈기를 구현하였다. RB SUM 단의 트랜지스터 개수를 줄이기 위해 MPPL XOR/XNOR 게이트를 제안하였으며, RB SUM 을 2의 보수로 변환하는데 변환 비트를 사용하였다. 제안된 덧셈기는 두 가지 형태로 설계하였으며 각 덧셈기의 특징은 다음과 같다. TYPE 1 덧셈기는 VGS 방식을 사용하여 변환 비트의 전파속도를 줄임으로서 덧셈기의 속도를 향상시켰으며 TYPE 2 덧셈기는 TYPE 1 덧셈기의 팬아웃 문제를 해결하기 위한 8비트 TYPE 1 덧셈기와 고속 RB2TC를 위한 64비트 GCG회로를 혼합하여 구성하였다. 64비트 TYPE 1 덧셈기의 경우 CLA와 트랜지스터 개수는 비슷하나 게이트 지연은 기존의 덧셈기보다 23.5% 감소하며, CSA와 비교시 트랜지스터 개수는 58.2% 적으며 게이트 지연은 29.7% 감소한다. 64비트 TYPE 2 덧셈기의 경우 트랜지스터 개수는 CSA에 비해 37.7%감소하며 게이트 지연은 45.9% 감소하나 CLA와 비교시 게이트 지연의 감소(37.9%)는 있으나 트랜지스터 개수는 53.8% 증가한다. 따라서 TYPE 2 덧셈기는 면적과 게이트 지연을 고려한 설계를 하여야 한다. 24비트 TYPE 1과 TYPE 2 덧셈기의 포스트 시뮬레이션시 전달지연 시간은 각각 1.4ns와 1.2ns로 나왔다.

제안한 덧셈기는 레이아웃이 매우 규칙적이기 때문에 빠른 시간에 회로 설계 및 레이아웃이 가능하며 마이크로프로세서나 DSP 등과 같이 고속연산을 필요로 하는 경우에 적합하다.

## 참 고 문 헌

[1] H.R.Srinivas and K.K.Parhi, "A Fast VLSI Adder Architecture", *IEEE Jour.* Solid-State Circuits, Vol. 27, No. 5, pp. 761-767, May 1992.

[2] N.E. Weste and K. Eshraghian, "principles of CMOS VLSI design : A systems perspective", *Addison-Wesley*, NY, 1993.

[3] J.M Dobson and G.M. Blair, "Fast two's complement VLSI adder design", Electronics Letters, Vol. 31, No. 20, pp. 773-783, 28th Sept. 1995.

[4] A. Vandemeulebroecke, E. Vanzieleghem. T. Denayer, and P.G. Jespers, "A new carry-free division algorithm and its application to a single-chip 1024-b RSA processor", IEEE Jour. of Solid-state Circuits, Vol. 25, No. 3, pp. 748-756, June 1990.

[5] H. Makino et al. "An 8.8-ns 54×54-bit multiplier with high speed redundant binary architecture", IEEE Jour. of Solid-State Circuits, Vol. 31, No. 6, pp. 773-783, June, 1996.

[6] R.P. Brent and H. T. Kung, "A regular layout for parallel adders", IEEE Trans. Comput., vol. C-31, no. 3, pp. 260-264, Mar, 1982.

[7] M. Uya, K Kaneko, and J. Yasui, "A CMOS

floating point multiplier", IEEE J. Solid-State Circuits, vol. SC-19, no. 5, pp. 697-702, Oct. 1984.

[8] A. Bellaouar and H. Touzene, "3.3-V BiCMOS current mode logic circuits for high speed adders", IEEE Jour. of Solid-State Circuits, vol. 31, pp. 1165-1169, 1996.

[9] P.K.Chan, M.D.F.Schlag, C.D, and V.G. Oklobdzija, "Delay optimization of carry-skip adders and block-carry-lookahead adders using multidimensional dynamic programming". IEEE Transactions on Computers, vol 41, pp. 920-930, 1992.

[10] C.-I.H.Chen and A.Kumar, "Comments on Area-time optimal adder design", IEEE Transactions on Computers, von 43, pp. 507-512, 1994.

[11] W.Heimsch et al. "Merged CMOS/bipolar current switch logic (MCSL)", IEEE Jour. of Solid-State Circuits, vol 24, pp. 13007-1311, 1989.

[12] I.S.Hwang and A.L.Fisher, "Ultrafast compact 32-bit CMOS adders in multiple-output domino logic", IEEE Jour. of Solid-State Circuits, vol. 24, pp. 358-369, 1989.

[13] K.J.Janik and S.-L.Lu, "VLSI implementation of a 32-bit Kozen formulation Ladner-Fischer parallel prefix adder", proceedings of the IEEE ASIC Conference, pp. 57-59, 1995.

[14] J.B.Kuo, H.J.Liao and H.P.Chen, " A BiCMOS dynamic Manchester carry look ahead circuit for VLSI implementation of high speed arithmetic unit" Proceedings of the Custom Integrated Circuits Conference(IEEE), paper 25.4., 1992.

[15] M.Suziki et al., "A 1.5ns 32-b CMOS ALU in double pass-transistor logic", IEEE Jour. of Solid-State Circuits, vol. 28, pp. 1145-1151, 1993.

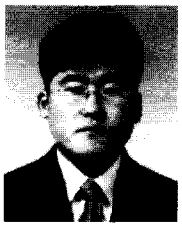
[16] A. Tyagi, "A reduced-area scheme for carry-select adders", IEEE Transactions on Computers, vol. 42, pp. 1163-1170, 1993.

[17] J. Wang et al., "Area-time analysis of carry look ahead adders using enhanced multiple output domino logic", Proceedings of the International Symposium on Circuits and Systems(IEEE), vol. 4(VLSI), pp. 59-62, 1994.

## 저 자 소 개

趙 相 福(終身會員)

1955년 6월 10일생. 1979년 2월 한양대학교 전자공학과 졸업(공학사). 1981년 2월 동 대학원 전자공학과 졸업(공학석사). 1985년 2월 동 대학원 전자공학과 졸업(공학박사). 1994년 8월~1995년 8월 Univ. of Texas, Austin 초빙학자. 1986년 3월~현재 울산대학교 교수. 자동차전자 연구센터 소장. 주관심 분야는 ASIC 설계, 자동차 전자회로 설계, 비전 시스템 개발, 테스트 및 테스트 용이한 설계 등임



李 泰 旭(學生會員)

1973년 5월 28일생. 1998년 2월 울산대학교 전자공학과 졸업(공학사). 2000년 2월 동 대학원 전자공학과 졸업(공학석사). 2000년 3월~현재 동 대학원 전자공학과 박사과정. 주관심 분야는 VLSI 설계, 저전력 설계 등임