論 文

실시간 디지탈 신호 처리용 고속 MULTIPLIER 단일칩화에 관한 연구

正會員 文 大 哲* 正會員 車 均 鉉**

A Study on the IC, Implementation of High Speed Multiplier for Real Time Digital Signal Processing

Dai Tchul MOON*, Kyun Hyon TCHAH** *Regular Members*

要 約 본 연구에서는 고속의 병렬 알고리즘을 이용하여 실시간 디지탈 신호를 처리할 수 있는 16×16 고속의 CMOS 승산기를 설계하였다. 설계된 병렬 승산기는 modified Booth's 알고리즘과 Ling's approach를 이용하여 4열의 가산기와 8개의 Booth 디코더로 구성하였으며, 2's complement의 데이타와 계수를 처리할 수 있도록 설계하였다.

또한 VLSI 구현에 적합하도록 modulrity하고 regularity하게 모든 회로를 설계하고 규칙적으로 내부 배열을하여 testability가 용이하도록 설계하였다.

ABSTRACT In this paper we present on architecture for a high sppeed CMOS multiplier which can be used for real-time digital signal processing. And a synthesis method for designing highly parallel algorithms in VLSI is presented. A parallel multiplier design based on the modified Booth's algorithm and Ling's algorithm. This paper addresses the design of multiplier capable of accepting data in 2's complement notation and coefficients in 2's complement notation. Multiplier consists| of an iterative array of sequential cells, and are well suited to VLSI implementation as a results of their modularity and regularity. Booth's decoders can be fully tested using a relatively small number af test vector.

I. 서 론

최근의 컴퓨터와 통신 시스템 분야가 급속도로 발전함에 따라서 DSP 또는 image processing 과 같은 실시간 처리 및 많은 양의 데이타 처리를 위하여 디지탈 시스템의 고속처리 능력이 필요하게 되었다.

Multiplier는 매트릭스 연산, 콘볼루션, eigenvalve 의 계산, 선형 방정식, 필터등 응용분야가 광범위하다. 이와 같이 대부분의 디지탈 시스템의 핵심적인 요소인 승산기의 연산속도가 전체 시스템에 미치는 영향으로 인하여 승산기의 연산속도 개선의 필요성이 대두되게 되었다. 디지탈 필터나 여러 signal 프로세서들의 하드웨어 구현에서 이와 같은 승산기는 입력 시퀀시와 variable 계수와의 승산 연산시 데이타의 처리 방법에

*湖西大學校 情報通信工學科
Dept. of Information Telecommunication Eng., Hoseo University
**高麗大學校 電子工學科
Dept. of Electronic Eng., Korea University
論文番號 : 90-63(接受1990. 7. 23)

따라 여러가지로 구분되어 진다[1].

고속연산에 적합한 병렬 승산기는 Wallace와 L. Dadda에 의하여 최초로 소개되었으나, 이는 막대한 양의 하드웨어를 필요로 했기 때문에 한번의 곱셈 연산을 위해서 거의 면적을 차지하지 않는 1-D 승산기의 개발로 발전하게 되었다. Jackson등은 한개의 전가산기와 몇개의 게이트를 포함한 N개의 셀을 필요로 하는 단 방향성 데이타 흐름 패턴을 갖는 1-D serial-parallel 시스토릭 승산기를 소개하였으며, Lyon은 위의 1-D serial-parallel systolic 승산기를 modified Booth's 알고리즘을 이용하여 더적은 클럭수를 갖고 더 적은 셀(n/2)로써 1-D 시스토릭 승산기로 직렬로 modify 했다. 또한 McCanny와 McWhirter는 파이프라인된 2-D 시스토릭 승산기를 소개하였다.[7]

그 이후 Chen과 Willoner에 의해 제안되고, Gnanasekaran에 의해 modify된 1-D serial 비시스토릭 승산기가 소개되었으며, 1987년에 I-Chen Wu에 의하여 소자의 수를 n/2로 줄이고 연산 속도를 증가시키며 2's complement 곱셈에 유용한 고속 I-D serial-parallel 시스토릭 승산기를 소개 하였다. 최근에는 처리속도를 더욱 고속화하기위한 고속연산 알고리즘과 최적의 아키텍쳐를 개발하기 위한 연구가 활발히 이루어 지고 있다.

따라서 본 논문은 이에 부응하기 위하여 연산시 캐리지연을 감소시키기 위해서 CLA 구성시 Ling's approach를 이용, pipeline와 병렬처리, wiring에 있어서 면적의 낭비를 최소화 하기위한 규칙적인 데이타의 흐름, 그리고 회로의 크기에 거의 상관없이 clock rate를 높게 유지할 수 있도록 고속의 실시간 디지탈 신호를 처리할 수 있는 병렬 승산기를 설계하였다.[5X6X7X9]

Ⅱ. Multiplier 실현

디지탈 시스템의 콘볼루션이나 여러 signal 프로세서들의 하드웨어 구현은 입력 시퀀시와 variable 계수와의 multiplier 연산시 데이타의 처리방법에 따라 bit-serial multiplier, serial-parallel multiplier, parallel multiplier등의 방법으로 구현할 수 있다.[1X2] serial한 경우는 전체적으로 space를 감소시킬 수 있으며 parallel multiplier의 경우 연산과정시 pipelining 방법을 이용 partial product에 대한 덧셈 연산을 신속히처리할 수 있음으로 고속연산에 적합한 장점을 갖는다.

고속 실시간 디지탈 신호를 처리하기 위해서 parallel multiplier로 설계하였으며, 또한 16×16 multiplication 연산시 발생되는 partial sum의 수를 줄이기위해서 modified Booth's 알고리즘을 사용하였다.[5X6X7] 또한 효율적인 partial sum 연산을 위해 Wallace tree 알고리즘을 이용한 adder 연산을 취하였고, CLA 설계에서는 Ling's approach를 이용하여 게이트 수를 감소시키고 속도를 증가시킴과 동시에 설계와 레이아웃 및 테스팅을 용이하게 할 수 있도록 modularity하고 regularity하게 회로들을 설계하였다.[2X4X11]

승산되어질 두개의 2진수 U,V를 다음과 같이 표현하자.

$$U=U_{n-1},\ U_{n-2},\cdots,\ U_1,\ U_0$$
$$V=V_{n-1},\ V_{n-2},\cdots,\ V_1,\ V_0$$

여기서 U_0, V_0는 LSB이다.

$P=U\cdot V$라하면

$$P=P_{2n-1},\ P_{2n-2},\ \cdots,\ P_1,\ P_0$$

가 된다. 따라서,

$$P=\sum_{i=0}^{n-1} 2^i \sum_{j=0}^{i} U_{i-j}\cdot V_j + \sum_{i=n}^{2n-n} 2^i \sum_{j=i-n+1}^{n-1} U_{i-j}\cdot V_j \qquad (1)$$

식 (1)에서 P의 $P_0,\cdots,P_{n-1}$의 n개 비트는 LSB를, $P_n,\cdots,P_{2n-1}$의 n개 비트는 MSB를 나타낸다. 식 (1)에서 U_iV_j를 다시 a_{ij}로 재표현하면

$$W=\sum_{i=0}^{n-1} 2^i \sum_{j=0}^{i} a_{i-j,j} + \sum_{i=n}^{2n-2} 2^i \sum_{j=i-n+1}^{n-1} a_{i-j,j}$$

가 된다.

승산기는 16 비트의 데이타와 16 비트의 계수를 갖는 병렬 승산기로 설계하였으며, 곱셈 연산시 발생되는 partial product의 수를 줄이기 위해서 modified Booth's 알고리즘을 사용하였다. 또한 효율적인 partial sum 연산과 속도 개선을 위해서 CSA 및 Wallace tree 알고리즘을 이용한 가산기 형태의 연산을 취하였다. 구현된 승산기 회로는 그림 1에서 처럼 Booth Decoder, Coefficient Latch, MUX Addition 블럭등으로 구성되었다.[6X8X9]

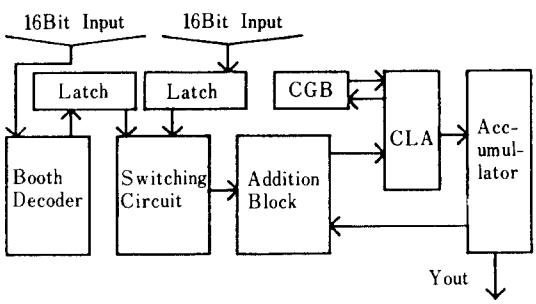


그림 1. multiplier 블럭도
Block diagram of multiplier

Ⅲ. 회로설계

Ⅲ-1. Modified Booth's 알고리즘 및 Booth 디코더

이 알고리즘은 multiplier의 세개의 bit 씩 묶어서 (±,0,±2)의 시퀀시로서 계수를 레코딩하는 방법으로, 회로의 복잡도는 증가하나 연산과정은 1/2로 줄어든다.[7]

Y를 계수, $Y_i'' \cdot 2^{i-1}$를 recoded number라고 할 때

$$Y=\sum_{\substack{i=0\\ \text{odd}}}^{N} Y_i'' \cdot 2^{i-1}$$

의 식이 성립된다.

여기서

$$Y_i''=Y_{i-1}+Y_i-2Y_{i+1} \in \{-2, -1, 0, 1, 2\}$$

관계가 얻어진다.

따라서 3개의 multiplier를 동시에 테스트함으로서 multiplcation 속도를 증가시킬 수 있다.

식 (1)로부터 Y_i''를 recoder로 발생함으로써 얻어지는 결과는 표 1와 같다.

표 1. Five level 승산기 레코딩
Five-level multiplier recoding

original multiplier			recoded multiplier	action
Y_{i-1}	Y_i	Y_{i+1}	$Y_i''=Y_{i+1}+Y_i-2Y_{i-1}$	
0	0	0	+0	add 0
0	0	1	+1	add X
0	1	0	+1	add X
0	1	1	+2	add 2X
1	0	0	−2	add 2X
1	0	1	−1	add −X
1	1	0	−1	add −X
1	1	1	−0	add −0

그리고 Y_i''를 FLR(five level recoder)로 구성하기 위하여 표 1로부터 modified Booth's 알고리즘의 동작에 대하여 sign-magnitude number Y_i^3, Y_i^2, Y_i^1 (Y_i^3−sign, Y_i^2−'2'magnitude, Y_i^1−'1'magnitude)로 표시할 수 있다. 따라서 코딩된 계수 (±2, ±1, 0)를 발생하기 위한 디코더 구성에 대한 디코딩 진리표를 작성하면 표 2와 같다.

표 2을 이용하여 Booth's 디코더를 설계하면 그림 2와 같다.

Ⅲ-2. Multiplication

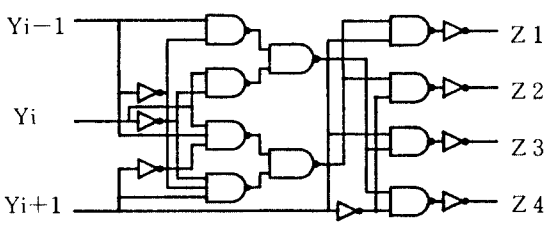
recoded number (±2, ±1,0)의 5개 항과 같은

표 2. 디코딩 진리표
Decoding truth table

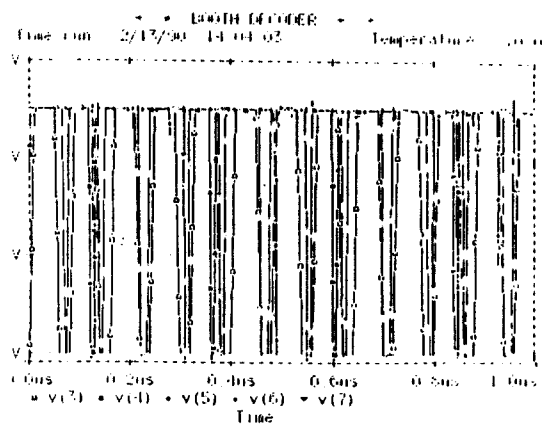
Y_{i+1}	Y_i	Y_{i-1}	Y_i''	Y_i^3	Y_i^2	Y_i^1	Z_1 (+1)	Z_2 (−1)	Z_3 (+2)	Z_4 (−2)	Z_5 (0)
0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	1	1	0	0	0	0
0	1	0	1	0	0	1	0	0	1	0	0
0	1	1	2	0	1	0	X	X	X	X	X
1	0	0	−2	1	1	0	0	1	0	0	0
1	0	1	−1	1	0	1	0	0	0	1	0
1	1	0	−1	1	0	1	X	X	X	X	X
1	1	1	−0	1	0	0	0	0	0	0	1

표 3. Recoded 계수에 의한 multiplicand 발생도표
Table of multiplcand generation with recoded coefficients

multiplie	multiplicand (외부 입력 데이타)										
(계 수)	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1
(+1) Z1	$\overline{D_{10}}$	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1
(−1) Z2	D_{10}	$\overline{D_{10}}$	$\overline{D_9}$	$\overline{D_8}$	$\overline{D_7}$	$\overline{D_6}$	$\overline{D_5}$	$\overline{D_4}$	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$
(+2) Z3	$\overline{D_{10}}$	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	0
(−2) Z4	D_{10}	$\overline{D_9}$	$\overline{D_8}$	$\overline{D_7}$	$\overline{D_6}$	$\overline{D_5}$	$\overline{D_4}$	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	1
(0) Z5	1	0	0	0	0	0	0	0	0	0	0



(a) 논리도



(b) 시뮬레이션

그림 2. Booth 디코더
Booth decoder

동작을 갖도록 16 bit multplicand에 대한 표를 작성하면 표 3와 같다.

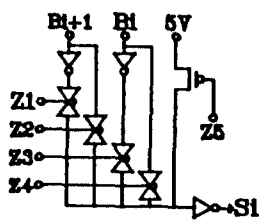
위의 항 중에서 −1과 −2는 2's complement 연산을 필요로 하므로 표 3의 항 중에서 LSB 부분에 +1을 해주는 과정이 필요하다. 또한 D_1 bit의 +2, −2의 경우는 +1, −1의 값을 MSB 방향으로 1 bit 시프트 해줌으로써 2배의 값을 얻도록 한다. 그리고 맨 마지막 D_{11} bit는 D_{10} bit를 sign-extention 한 bit이다.

계수의 자연 발생을 위하여 MUX 회로로써 구현하여 보면, MUX 회로는 표 4와 같은 동작을 갖으면 되며, Z5 제어는 Z1~Z4를 OR 게이트로 묶어서 처리하였다.

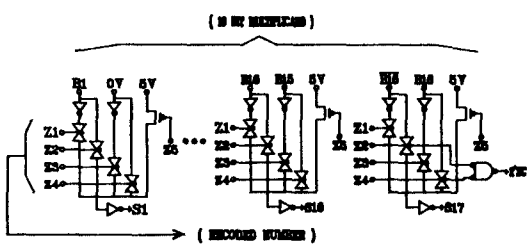
표 4. 입력 계수에 대한 멀티플렉서의 동작
Multiplexer operation for input coefficients

Z1	Z2	Z3	Z4	5	S
1	0	0	0	0	b_{i+1}
0	1	0	0	0	b_{i+1}
0	0	1	0	0	b_i
0	0	0	1	0	b_i
0	0	0	0	1	0

실제적인 자연 계수발생 회로에서 MSB의 MUX 회로는 부호 확장(sign-extention)을 위한 회로 구성을 위하여 그림 3에서의 Z1과 Z3 제어에 의해 D_{10}가 그리고 Z2와 Z4 제어에 의해서는 D_{10}이 발생되는 회로를, LSB에서는 B_{I+1}에 D_I이 B_I에 '0' volt 값을 주어야만 실제 적용되는 multiplicand 가 생성된다.



(a) 기본 회로



(b) 전체 회로

그림 3. 멀티플렉서 회로도
Circuit diagram of multiplexer

Ⅲ-3. 부호 확장을 위한 회로 설계

Ⅲ-2에서 설명한 자연계수 발생회로에서는 MSB가 반전된 형태로 발생되도록한 회로이므로 이를 MSB 값으로 변환하여 부호를 확장시키는 회로가 필요하다.

부호확장을 고려한 연산결과를 얻기위해서는 $\overline{MSB}$ 값을 MSB 값으로 변환시켜 주어야하며, $\overline{MSB}$의 값이(MSB가 n bit 일때) n+1 bit이후에 partial sum 계산에 더해져야 하므로 이를 위한 회로가 필요하다.

부호 확장 회로설계에 있어서의 HA+1 회로는 additon 블럭에서 Wallace tree 알고리즘을 이용하여 전파되는 partial sum과 partial carry를 묶는 과정에서 부호 확장을 필요로 하는 부분 중 +1 연산을 취해야할 부분에 사용함으로 연산 속도 및 사용되는 소자 감축면에서 잇점을 가져온다.

실제적인 자연 계수발생 회로에서 MSB의 MUX 회로는 부호 확장을 위한 회로 구성을 위하여 그림 3에서의 Z1과 Z3 제어에 의해 D_{10}가 그리고 Z2와 Z4 제어에 의해서는 D_{10}이 발생되는 회로를, LSB에서는 B_{I+1}에 D_I이 B_I에 '0' volt 값을 주어야만 실제 적용되는 multiplicand 가 생성된다. 그러므로 앞에서 설명한 자연계수 발생회로에서는 MSB가 반전된 형태로 발생되도록한 회로이므로 이를 MSB 값으로 변환하여 부호를 확장시키는 회로가 필요하다. 즉 부호확장을 고려한 연산결과를 얻기위해서는 $\overline{MSB}$ 값을 MSB 값으로 변환시켜 주어야 하며, MSB의 값이(MSB가 n bit 일때) n+1 bit 이후에 partial sum 계산에 더해져야 하므로 이를 위한 회로가 필요하다.

부호 확장을 위한 HA+1 회로를 표 5로 부터 구현하면 그림 4와 같다.

표 5. HA+1 진리표
Truth table of HA+1

a	b	S1	C1	S2	C2
0	0	0	0	1	0
0	1	1	0	0	1
1	0	1	0	0	1
1	1	0	1	1	1

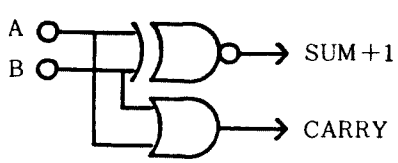
여기서 S1, C1은 HA 회로에서의 sum과 carry 값이고 S2, C2은 HA+1 회로에서의 sum과 carry 값이다. 따라서 HA+1에 대한 부울리안 함수를 표현하면 다음과 같다.

$$SUM+1=\bar{a}b+a\bar{b}=a\bigcirc b=\overline{a\oplus b}$$
$$CARRY=\bar{a}b+a\bar{b}+ab=(\bar{a}+a)b+ab=b+a\bar{b}$$
$$=b+ab+a\bar{b}=a+b$$

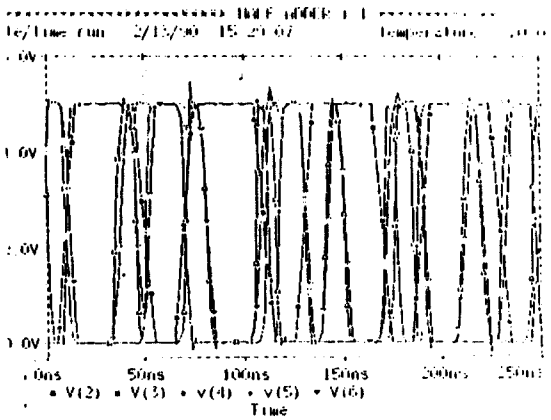
Ⅲ-4. CLA 단의 설계

1) CLA 단계 설계

Addition은 carry save 기법으로써 좀더 효율



(a) HA+1 회로



(b) 시뮬레이션 결과

그림 4. HA+1 회로 및 시뮬레이션 결과
HA+1 circuit and result of simulation

적인 partial sum 연산과 속도개선을 위하여 Wallace tree를 이용한 가산기 연산을 취하였다. 입력 워드의 크기를 갖는 가산기의 캐리 지연은 그 캐리들을 CSA 어레이의 마지막 단에서 병렬로 각 단에서 계산해 줌으로써 개선이 가능하다. 여기서 CLA 블럭은 sum 발생이 좀더 복잡하지만 리플캐리없이 캐리 게이트의 한 단계를 줄여서 효과적으로 속도를 증가시키고 소자를 감축할 수 있도록 Ling's approach를 사용하였다.[9X10]

기존의 CLA에서 두 수를 가산할 경우에 i차 단의 캐리의 generate 항과 propagate 항은 다음과 같다.

$g_i=A_i \cdot B_i$ (carry 발생신호)
$P_i=A_i \oplus B_i$ (propagate 신호)

상기식으로부터 carry와 최종 sum에 관한 순환 관계식은 다음과 같이 표현된다.

$$G_i=g_i+P_i \cdot G_{i-1} \quad (2)$$
$$S_i=A_i \oplus B_i \oplus C_{i-1}=P_i \oplus G_{i-1} \quad (3)$$

Ling's 가산기는 기존의 CLA의 G_i항 대신에 carry in이나 carry out가 발생했다는 의미를 갖는 H_i를 다음과 같이 정의한다.

$$H_i=G_i+G_{i-1} \quad (4)$$

식 (2)에서 G_i항은 $\{g_i : g_i \supset G_i\}$이므로, 식 (4)에서 $G_i \supset H_i$가 된다.
그러므로
$$g_i=g_i \cdot H_i \quad (5)$$

가 성립한다. 그리고 식 (4)에서 $P_i \cdot G_{i-1}$항은

$$\begin{aligned} P_iG_{i-1} &\equiv P_i \cdot G_{i-1}+P_i \cdot g_i+P_i \cdot G_{i-1} \\ &\equiv P_i \cdot G_{i-1}+P_i \cdot G_i \\ &\equiv P_i(G_{i-1}+G_i) \\ &\equiv P_i \cdot H_i \end{aligned} \quad (6)$$

가 된다. 식 (5)와 식 (6)를 식 (4)에 대입하면

$$\begin{aligned} G_i &= g_i+P_i \cdot G_{i-1} \\ &= g_i \cdot H_i+P_i \cdot H_i \\ &= (g_i \cdot P_i)H_i \\ &= t_i \cdot H_i \end{aligned} \quad (7)$$

가 된다. 여기서 $t_i=A_i+B_i$이다. 그러므로 식 (4)은 다음과 같이 유도된다.

$$\begin{aligned} H_i &= G_i+G_{i-1} \\ &= g_i+P_i \cdot G_{i-1}+G_{i-1} \\ &= g_i+G_{i-1}(P_i+1) \\ &\quad (P_i+1=1) \\ &= g_i+G_{i-1} \end{aligned}$$

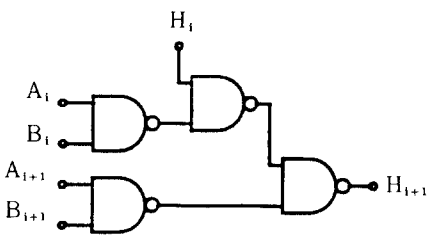
따라서 상기식에 식 (7)을 대입하면 Ling's recurence 관계식은 다음과 같다.

$$H_i=g_i+t_{i-1}\cdot H_{i-1} \quad (8)$$

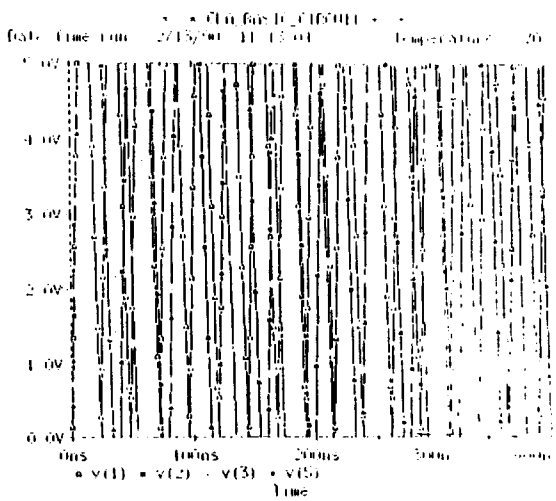
그리고, sum S_i를 구하기 위하여 정의식 (3)에서 식 (7)을 대입하면

$$S_i=t_i\oplus H_i+g_i\cdot t_{i-1}\cdot H_{i-1} \quad (9)$$

가 된다. 식 (8)에 대하여 CLA의 기본회로를 설계하면 그림 5와 같다.



(a) 기본회로



(b) 시뮬레이션

그림 5. CLA의 기본 회로
Basic citcuit of CLA

결과적으로 Ling's 가산기는 $(a_i+b_i,\ a_i\oplus b_i)$로 간단하게 시작되나, 복잡한 결과를 가져온다. 그러나 한단계의 t는 순서가 감소됨으로써 표 6비교표에서 처럼 캐리 발생 항이 상당히 간소하게 된다는 잇점이 있다.

식 (8)과 (9)을 이용하여 4-stage로의 확장을 고려해 볼때, 기존 CLA에서의 캐리 발생함수 C_3를 Ling's 알고리즘을 사용한 식으로 표현했을 경우에 H_3와 비교해보면 C_3는 4개의 게이트에 대하여 10개의 입력을 갖는 반면에, H_3는 3개의 게이트에 대하여 7개의 입력을 갖으므로, 궁극적으로 캐리 전파시간의 감소와 사용되어지는 게이트가 감소하므로 전체적인 칩 면적이 감소한다.

표 6. 기존 CLA와 Ling's 가산기를 적용한 CLA와의 비교
Contrast of conventional CLA and Ling's CLA

기존 CLA의 carry 발생 수식
$C_0=C_0$ $C_1=g_1+P_1C_0$ $C_2=g_2+P_2g_1+P_2P_1C_0$ $C_3=g_3+P_3g_2+P_3P_2g_1+P_3P_2P_1C_0$ $C_4=g_4+P_4g_3+P_4P_3g_2+P_4P_3P_2g_1+P_4P_3P_2C_0$
Ling's CLA의 carry 발생 수식
$H_0=g_0$ $H_1=g_1+g_0$ $H_2=g_2+g_1+t_1g_0$ $H_3=g_3+g_2+t_2g_1+t_2t_1g_0$ $H_4=g_4+g_3+t_3g_2+t_3t_2g_1+t_3t_2g_0$

또한, 식 (8)과 (9)의 recurence 관계로 부터 회로를 설계할때 기존의 CLA처럼 사이즈의 문제가 발생한다. 그래서 보통 carry look-ahead의 stage 수는 4개로 제한되며, 4단을 기본으로 하여 recursive하게 다음단에 이용될 수 있도록 modularity하게 설계하였다. 그리고 기본 4단의 개선 CLA에서 발생한 마지막 캐리를 다음단의 초기 캐리로 만들어 주기위한 종속 캐리 발생 블럭(CGB)을 설계하였다.[1X2]

CGB의 구성은 CLA에서 4-stage씩 나누어 계산될때 각 stage에서 최종단의 캐리 값만을 따로 계산하여 동시에 다음단에 캐리로 전달함으로써 좀더 효과적으로 계산하기 위하여 만들었으며 그 회로는 그림 6와 같다.

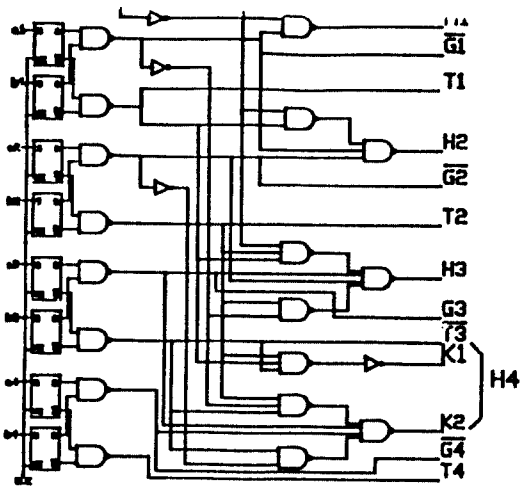


그림 6. 4단 CLA 회로
Circuit of 4-stage CLA

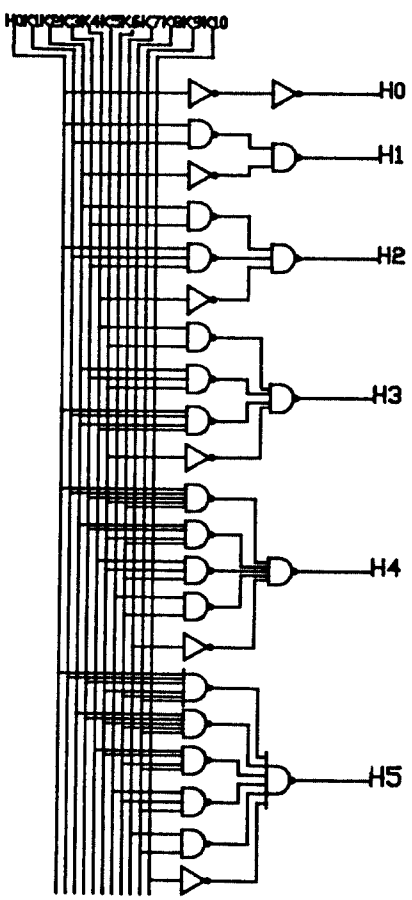


그림 7. 캐리 발생 블럭
Carry generation block

2) Addition 블럭.

Booth 디코더로부터 발생된 multiplier와 외부 데이타에 의하여 발생된 partial product를 CSA 기법 및 Wallace-tree를 이용하여 addition 함으로서 효율적인 연산을 취하였으며 CSA를 이용한 Wallace-tree 감소단계를 그림 8에 나타내었다.

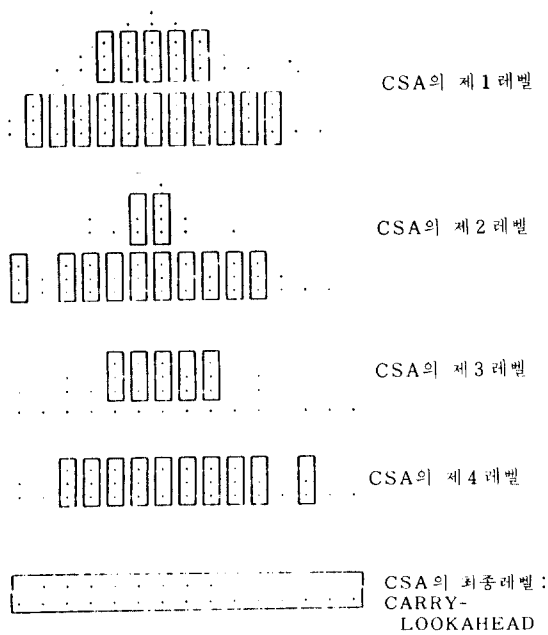


그림 8. CSA를 이용한 wallace-tree 감소단계
Wallace-tree reduction using carry save adder

그리고 최종 sum을 만드는 단에서 ACC에서는 외부의 조작 신호로서 시프트할 수 있는 기능과 출력을 제어할 수 있도록 3상태 MUX를 사용하여 설계 하였다.

최종 sum을 얻기위한 누산과정은 다음과 같다.

$$A=S_a+C_a,\ U=V+A,B$$
$$(S_u,C_u)=(S_v,C_v)+A,B$$

여기서 A,B 그리고 C는 모두 2진수이다. 위 식에서 A,B는 modified booth 알고리즘을 사용

하여 발생된 (16×16)비트 multiplicand 수이며 그림 9은 전체적인 CSA 비트 슬라이스와 CLA 관계를 나타낸 그림이다.[1X5]

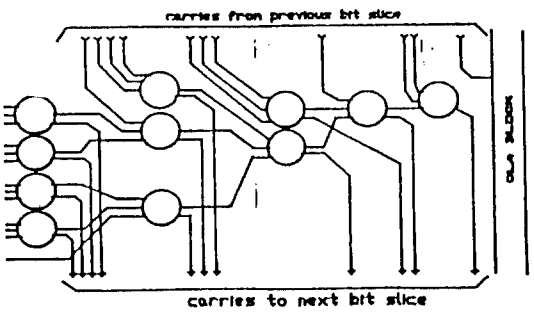


그림 9. CSA 비트 슬라이스와 CLA 관계
Relation of CLA and CSA bit slice

Ⅳ. 결 론

본 논문에서는 고속의 병렬 알고리즘을 이용하여 실시간 디지탈 신호를 처리할 수 있는 16×16 고속의 CMOS 승산기를 설계 하였다.

설계된 승산기는 연산시에 계수를 recoding 함으로써 연산 소자의 수를 1/2로 줄일 수 있는 modified Booth's 알고리즘을 사용하였고, 가산기 어레이는 효율적인 partial sum 연산과 속도 개선을 위해서 CSA 기법 및 Wallace tree 알고리즘을 이용하였으며, CLA 블럭에서는 소자수의 감축과 속도를 증진 시킬수 있는 Ling's approach 에 기초를 두고 회로를 실현하였다. 이렇게 설계되어진 승산기는 순차적인 셀의 내부 연결과 modularity하고 규칙적으로 구성하였음으로 VLSI 회로를 구현하기에 아주 적합하게 하고, 또한 Booth 디코더는 적은 수의 테스트 벡터를 사용하여 전체적인 테스트가 가능하게 하도록 설계하였다.

실현된 승산기의 시뮬레이션 결과 지연시간은 약 80ns, 속도-전력곱은 2.7로 측정 되었다.

참 고 문 헌

1. Weste Eshraghian, "Prionciples of CMOS VLSI Design", 1988
2. Carver Mead & Lynn Conway, "Introduction to VLSI System", 1980.
3. Douglas A. Pucknell & Kamran Eshrghian, "Basic VLSI Dsign Systems and Circuit", 1988.
4. Richard J.Higgins, "Digital Signal Processing in VLSI", Prentice-Hall, Inc., 1990.
5. N. Zafar, M.N. Konar, T.Oseth, "A 24×24 Bit Parallel Multiplier based a futher Modified Booth's Algorithm", IEEE, 1985.
6. Louis P. Rubinfield, "A Proof of the Modified Booth's Algorithm for Multiplication", IEEE, 1975.
7. I-Chen Wu, "A Fast 1-D Serial-parallel Systolic Multiplier", IEEE, 1987.
8. Rondney T. Masumto, "The Design of a 16×16 Multiplier", LAMBDA First Quarter, 1980.
9. R.W. Doran, "Variants of an Improved Carry Look-Ahead Adder", IEEE Trans. On Comp. Vol. 37, no. 9, Sep., 1988.
10. H.Ling, "High Speed Binary Adder", IBM J. Res. Develop., Vol. 25, pp.156, May, 1981.
11. R. P.Brent and H. T. Kung, "A Regular Layout for Parallel Adders", IEEE Trans. Compt., Vol. C-31, pp.260-264, 1982.
12. L. Daada, "On Parallel Digital Multipliers", ALTA FREQUENZA, Vol. XLV-N. pp. 574-580, 10-OTT-OBRE 1976.
13. WILLIAM J. STENZEL, WILLIAM J. KUBITZ, "A Compact High-Speed Parallel Multiplication Scheme", IEEE Trans. on Comp., Vol. C-26, no.10, pp.948-057 Oct.1977.
14. CHARLES R. BAUGH, "A Two's Complement Parallel Array Multiplication Algorithm", IEEE Trans. on Comp., Vol. C-22, No. 12, pp.1045-1047, Dec. 1973.
15. SHINJI NAKAMURA, "Algorithm for Iteractive Array Multiplication" IEEE Trans. on Comp., Vol. c-35, No. 8, pp.713-719, August, 1986.
16. 박건표, 문대철, "실시간 디지탈신호 처리용 고속 MULTIPLIER 설계및 제작에 관한 연구", 통신학회, 춘계종합 학술 발표회 논문집. pp.203-207. 1990.
17. 노희돈, 문대철, "Ling's Approach를 이용한 Carry Look-ahead adder 설계 및 제작에 관한연구.", 춘계종합 학술 발표회 논문집, pp.21-214, 1990.

부 록

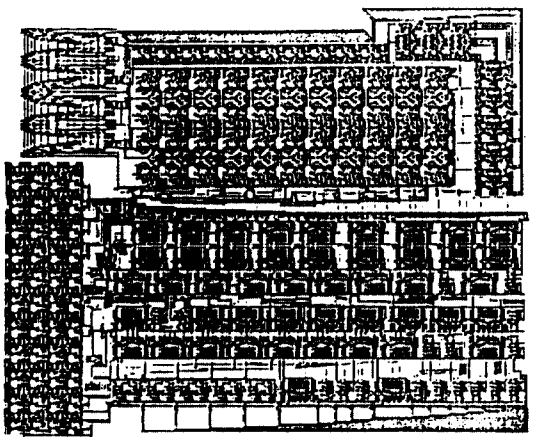
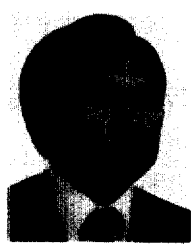


그림 A1. Multiplier 전체 Layout

文 大 哲(Dai Chul MOON) 正會員
1955年 6月 7日生
1979年 2月 : 숭전大 電子科 卒業
1981年 8月 : 高麗大 大學院 電子科(碩士)
1988年 2月 : 高麗大 大學院 電子科 (工學博士)
1984年 3月~現在 : 湖西大情報通信工學科 助教授



車 均 鉉(Kyun Hyon TCHAH) 正會員
1939年 3月26日生
1965年 : 서울大學校 工學士
1967年 : 美國일리노이大學校 工學碩士 學位 取得
1976年 : 서울大學校 工學博士學位 取得
1987年~現在 : 高麗大學校 電子電算工學科 教授
※主關心分野는 CAD 및 通信시스템等.