벡터 내적을 위한 효율적인 ROM 면적 감소 방법

정회원 최 정 필*, 성 경 진**, 유 경 주***, 정 진 균***

Efficient ROM Size Reduction for Distributed Arithmetic

Jung-Pil Choi*, Kyung-Jin Sung**, Kyung-Ju Yoo***, Jin-Gyun Chung*** *Regular Members*

요 약

본 논문에서는 벡터의 내적에 사용되는 Distributed Arithmetic의 ROM 면적을 줄일 수 있는 방법을 제시한다. 제안된 방법을 이용하여 만든 ROM table은 그 크기가 기존 방법에 비해 반으로 감소되며 새로 생성된 ROM table에 같은 방법을 반복 적용함으로써 그 크기를 계속 1/2씩 감소시킬 수 있다. 반면에 ROM table을 반으로 줄일 때마다 논리회로의 첨가로 인한 하드웨어의 증가와 critical path의 증가가 발생하는데, 이들 오버헤드를 최소화 할 수 있는 방법과 전체적인 면적 감소 효과를 극대화 할 수 있는 방법을 함께 제시한다. 제시한 방법을 적용함으로써 약 50%까지의 하드웨어를 감소시킬 수 있음을 예를 통하여 보인다.

ABSTRACT

In distributed arithmetic-based architecture for an inner product between two length-N vectors, the size of the ROM increases exponentially with N. Moreover, the ROMs are generally the bottleneck of speed, especially when their size is large. In this paper, a ROM size reduction technique for DA (Distributed Arithmetic) is proposed. The proposed method is based on modified OBC (Offset Binary Coding) and control circuit reduction technique. By simulations, it is shown that the use of the proposed technique can result in reduction in the number of gates up to 50%.

Ⅰ. 서 론

두 벡터의 곱셈은 콘볼루션이나 DCT의 구현에 필요한 연산이다[1-3]. 그러나 벡터 내적의 하드웨어 구현시 경우에 따라 큰 면적 및 처리 시간이 요구되며 이러한 문제점을 해결하기 위한 대표적인 방식이 Distributed Arithmetic (DA)이다[4].

두 벡터의 내적을 계산할 때 하나의 벡터 값이 알려져 있다면 내적에 관한 식을 재 조합함으로써 곱셈 없이 쉬프트와 가산기로써 동일한 결과 값을 얻을 수 있으며 하드웨어 면적을 상당히 감소시킬 수 있다. 그러나 입력 벡터길이가 커지면 ROM의 크기가 2^N으로 증가되므로 이를 해결하기 위해 Offset Binary Coding (OBC) 방식이 제안되었다. OBC는 DA의 ROM 면적을 반으로 줄일 수 있는 방식이다[4-5].

본 논문에서는 OBC에 의해 생성된 ROM 크기를 다시 반으로 줄일 수 있는 방법을 제시한다. 이 방법은 OBC의 ROM table을 기초로 이 값들을 antisymmetric하게 만듦으로써 ROM 면적을 반으로 줄이는 것이다. 새로운 ROM table에 이 방법을 반복 적용할 수 있으며 면적을 최대 50%까지 줄일 수 있다.

Ⅱ절에서는 DA 방식과 OBC 방식에 대해 간단히 설명하고 Ⅲ,Ⅳ절에서는 ROM 면적 감소를 위한 새로운 방법을 제시한다. Ⅴ절에서는 OBC와 새로 제안된 방법을 비교 분석하고 Ⅵ절에서 제안된 방법을 이용한 DCT 구현 예를 보인다.

* 현대전자(jpchoi@vlsidsp.chonbuk.ac.kr), ** LG전자(kjsung@lge.co.kr)
*** 전북대학교 전자·정보공학부(jgchung@moak.chonbuk.ac.kr)
논문번호 : 99241-0614, 접수일자 : 1999년 6월 14일
※ 본 연구는 정보통신부의 정보통신 우수 시범학교 지원사업에 의하여 수행되었습니다

Ⅱ. DA와 DA-OBC 구조

1. DA 구조

길이가 N인 두 벡터 C와 X에 대해 내적은

$$Y = \sum_{i=0}^{N-1} c_i x_i \qquad (1)$$

으로 정의된다. (1)에서 $\{c_i\}$는 M비트의 상수이고 $\{x_i\}$는 W비트의 2의 보수로 표현된 입력 데이터이다. (1)식을 다음과 같이 나타낼 수 있다.

$$\begin{aligned} Y &= \sum_{j=0}^{W-1} C_{w-1-j}\, 2^{-j}, \\ C_{W-1-j} &= \sum_{i=0}^{N-1} c_i x_{i,W-1-j}, \\ C_{W-1} &= -\sum_{i=0}^{N-1} c_i x_{i,W-1}. \end{aligned} \qquad (2)$$

식 (2)로부터 C_{W-1-j}는 $x_{i,W-1-j}$에 의해 결정되고 2^N개의 가능한 값만을 가지므로 이를 미리 계산해 ROM에 저장할 수 있다. 입력 벡터 $x_{i,W-1-j}$는 ROM값을 읽기 위한 주소로 사용되며 Y값은 단지 ROM에 저장된 C_{W-1-j}를 쉬프트하고 더함으로써 얻어진다. 표 1은 입력 벡터의 크기가 N=4인 경우에 ROM에 저장될 값을 나타낸 것이고, 그림 1은 크기가 N인 두 벡터의 내적을 계산하기 위한 구조이다. 그림 1에서 계산은 x_i의 최하위 비트 즉 j=0에서부터 시작하여 j=W-1 까지 진행되며 S는 j=W-1 일 때만 1이고 나머지는 0이다.

2. DA-OBC

DA 방식은 입력 벡터의 크기에 따라 ROM의 면적이 지수적으로 증가하게 되므로 입력 벡터의 크기(N)가 클 경우 ROM 면적이 매우 커지게 된다. .

OBC 방식을 사용하면 ROM 면적을 DA방식의 반으로 줄일 수 있다. 입력벡터 $x_{i,j}$는 다음과 같이 변형할 수 있다

$$x_i = \frac{1}{2}[x_i - (-x_i)]. \qquad (3)$$

다음과 같이 $d_{i,j}$를 정의 하면 $d_{i,j}$는 +1 또는 -1 값만을 갖는다.

$$d_{i,j} = \begin{cases} x_{i,j} - \overline{x_{i,j}}, & \text{for } j \neq W-1, \\ -(x_{i,W-1} - \overline{x_{i,W-1}}), & \text{for } j = W-1. \end{cases} \qquad (4)$$

표 1. DA의 ROM 값

x_{0j}	x_{1j}	x_{2j}	x_{3j}	Content of the ROM
0	0	0	0	0
0	0	0	1	c_3
0	0	1	0	c_2
0	0	1	1	c_2+c_3
0	1	0	0	c_1
0	1	0	1	c_1+c_3
0	1	1	0	c_1+c_2
0	1	1	1	$c_1+c_2+c_3$
1	0	0	0	c_0
1	0	0	1	c_0+c_3
1	0	1	0	c_0+c_2
1	0	1	1	$c_0+c_2+c_3$
1	1	0	0	c_0+c_1
1	1	0	1	$c_0+c_1+c_3$
1	1	1	0	$c_0+c_1+c_2$
1	1	1	1	$c_0+c_1+c_2+c_3$

표 2. DA-OBC의 ROM 값

x_{0j}	x_{1j}	x_{2j}	x_{3j}	Content of the ROM
0	0	0	0	$-(c_0+c_1+c_2+c_3)/2$
0	0	0	1	$-(c_0+c_1+c_2-c_3)/2$
0	0	1	0	$-(c_0+c_1-c_2+c_3)/2$
0	0	1	1	$-(c_0+c_1-c_2-c_3)/2$
0	1	0	0	$-(c_0-c_1+c_2+c_3)/2$
0	1	0	1	$-(c_0-c_1+c_2-c_3)/2$
0	1	1	0	$-(c_0-c_1-c_2+c_3)/2$
0	1	1	1	$-(c_0-c_1-c_2-c_3)/2$
1	0	0	0	$(c_0-c_1-c_2-c_3)/2$
1	0	0	1	$(c_0-c_1-c_2+c_3)/2$
1	0	1	0	$(c_0-c_1+c_2-c_3)/2$
1	0	1	1	$(c_0-c_1+c_2+c_3)/2$
1	1	0	0	$(c_0+c_1-c_2-c_3)/2$
1	1	0	1	$(c_0+c_1-c_2+c_3)/2$
1	1	1	0	$(c_0+c_1+c_2-c_3)/2$
1	1	1	1	$(c_0+c_1+c_2+c_3)/2$

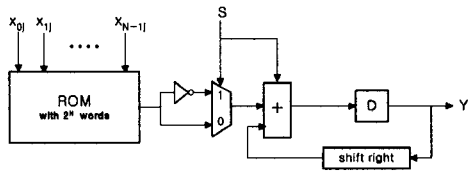


그림 1. DA에 의해 설계된 길이가 N인 벡터의 내적을 계산하기 위한 구조.

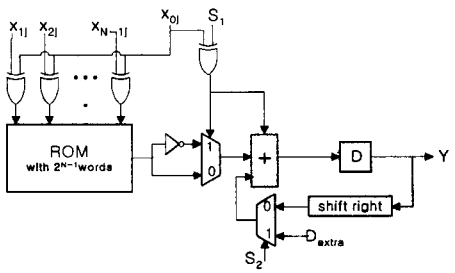


그림 2. OBC를 사용해서 그림 1을 재구성한 구조.

식(1)은 (2)~(4)를 이용하여 아래와 같이 다시 쓸 수 있다.

$$Y = \sum_{j=0}^{W-1} D_{W-1-j} 2^{-j} + D_{extra} 2^{-(W-1)},$$

$$\textit{where}, \quad D_j = \sum_{i=0}^{N-1} \frac{1}{2} c_i d_{i,W-1-j}, \quad \text{for } 0 \le j \le W-1,$$

$$D_{extra} = -\frac{1}{2}\sum_{i=0}^{N-1} c_i . \qquad (5)$$

위 식을 이용하여 작성한 벡터 크기 N=5인 경우 ROM table은 표 2와 같다. 표 2에서 상위 2^{N-1}개의 ROM 값과 그 나머지의 값들이 서로 antisymmetric하므로 상위 2^{N-1} 개의 ROM table만 있으면 나머지 값도 표현할 수 있다. 그림 2는 크기가 2^{N-1}으로 줄어든 ROM table을 사용해서 그림 1을 재구성한 구조이다. 그림 2에서 S_1은 $j=W-1$일 때만 1이고 나머지는 0이며 S_2는 $j=0$일 때만 1이고 나머지는 0이다.

Ⅲ. DA-OBC의 수정된 구조

표 2의 처음 2^{N-1} 개의 ROM 값을 표 3과 같이 c_0항을 제외한 나머지항을 따로 묶어서 표현해보면 처음 2^{N-2}개의 값과 그 나머지 값들이 서로 antisymmetric함을 알 수 있다. 따라서, 첫 번째에서 4번째까지의 값만 있으면 그 값들의 보수를 취한 후 $-c_0/2$를 더하여 5번째에서 8번째까지의 값들을 얻을 수 있다. 이 경우 $x_{0,j}=x_{1,j}=0$에 대응하는 ROM table (2^{N-2})만 있으면 되므로 DA-OBC구조에서 사용한 ROM 면적을 반으로 줄일 수 있다. 더불어 디코더 면적도 반 이상으로 줄어들게 된다. 새로운 ROM table은 표 4와 같으며

표 4에서 c_1항을 제외한 나머지를 묶으면 역시 ROM 값들이 antisymmetric하게 되어 그 ROM크기를 반으로 줄일 수 있다. 같은 방법을 계속 반복하면 ROM 면적을 최대 2워드까지 줄일 수 있다.그러나 위에 기술한 것처럼 ROM 크기를 줄일 때 2가지 문제점이 발생한다.

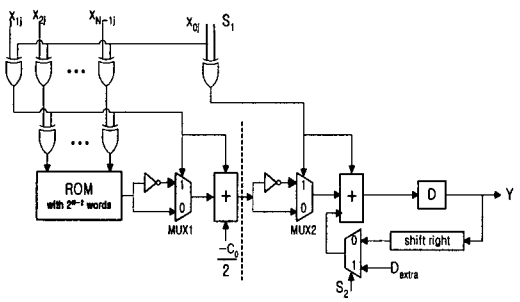


그림 3. 그림 2의 ROM의 크기를 절반으로 줄인 구조

표 3. C_0를 제외한 나머지를 묶은 경우의 ROM Table

x_{1j} x_{2j} x_{3j}	Content of the ROM
0 0 0	$-c_0/2-(c_1+c_2+c_3)/2$
0 0 1	$-c_0/2-(c_1+c_2-c_3)/2$
0 1 0	$-c_0/2-(c_1-c_2+c_3)/2$
0 1 1	$-c_0/2-(c_1-c_2-c_3)/2$
1 0 0	$-c_0/2+(c_1-c_2-c_3)/2$
1 0 1	$-c_0/2+(c_1-c_2+c_3)/2$
1 1 0	$-c_0/2+(c_1+c_2-c_3)/2$
1 1 1	$-c_0/2+(c_1+c_2+c_3)/2$

표 4. 크기가 절반으로 줄어든 ROM Table

x_{1j} x_{2j} x_{3j}	Content of the ROM
0 0 0	$-(c_1+c_2+c_3)/2$
0 0 1	$-(c_1+c_2-c_3)/2$
0 1 0	$-(c_1-c_2+c_3)/2$
0 1 1	$-(c_1-c_2-c_3)/2$

첫째, 그림 3에서 보듯이 부가회로가 증가한다. XOR 개수는 입력 벡터 크기에 비례하고 나머지의 면적은 계수에 비례하여 커진다. ROM 면적은 입력 벡터크기, 계수워드길이에 비례하므로 입력 벡터 크기가 커질수록 면적 감소의 효과가 커진다. 전체적인 면적감소의 효과를 얻기 위해서는 XOR, MUX (multiplexer), INV(inverter), FA의 면적증가를 ROM과 디코더 면적의 감소 폭 이하로 최소화 해야 한다.

둘째, MUX, XOR, INV, FA의 첨가로 인해 critical path가 증가한다. ROM, inverter, multiplexer, adder의 지연 시간이 각각 T_{ROM}, T_I, T_M, T_A라 할 때 DA-OBC 구조에서 critical path는 $T_{ROM}+T_I+T_M+T_A$ 이고 수정된 DA-OBC 구조의 경우 $T_{ROM}+2T_I+2T_M+2T_A$로써 약 2배 정도의 critical path가 증가한다. 만일 critical path를 반으로 감소시키고자 한다면 그림 3의 점선부분에 파이프라인을 사용해야한다. 추가적인 critical path 감소를 원할 경우 파이프라인의 반복 적용이 필요하다.

Ⅳ. 수정된 DA-OBC의 면적 최소화

이 절에서는 DA-OBC 구조를 수정할 때 발생하는 ROM, Decoder, XOR, MUX, INV, FA에 대한 면적 변화를 TR단위로 분석하고 XOR, FA의 면적 증가를 최소화하기 위한 방법을 기술한다. 본 논문에서 사용하는 회로는 가장 일반적인 회로를 기준으로 삼았으며 디코더는 AND 게이트로만 구성되고 Adder는 ripple carry adder로 가정하였다. N_c는 입

력 벡터의 크기, N_v는 입력 벡터의 워드길이, N_c는 계수의 워드길이를 나타낸다. 또한, AND, XOR, MUX, INV는 각각 4, 8, 6, 2개의 트랜지스터로 구성되는 것으로 가정한다.

1. Decoder, ROM

디코더 면적은 N_v에만 관계되고 N_i 및 N_c와는 무관하다. OBC와 수정된 OBC구조에서 디코더에 들어가는 AND 게이트의 개수를 각각 D_{OBC}, D_{mod}라고 하면 AND 게이트에 TR 4개가 사용될 경우 줄어든 TR 개수는 다음과 같다.

$$D_r = (D_{OBC} - D_{\mathrm{mod}}) \times [AND\,gatesize(TR's)] = [\,2(N_v-2) + 2^{(N_v-1)}\,] \times 4\ (TR's). \quad (6)$$

DA-OBC와 수정된 DA-OBC의 ROM 면적이 각각 R_{OBC}, R_{mod}일 때 ROM 면적 감소량은 다음과 같다.

$$R_r = R_{OBC} - R_{\mathrm{mod}} = 2^{N_v-2} \times N_c(TR's). \quad (7)$$

2. MUX(multiplexer), INV(inverter)

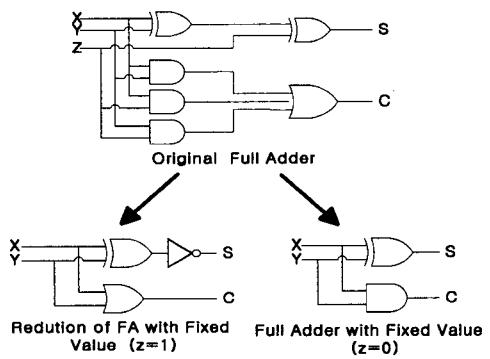


그림 4. 하나의 고정된 입력값을 갖는 FA

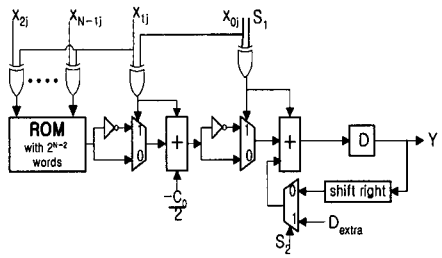


그림 5. 그림 3의 control block을 간소화한 구조

MUX와 INV에 의한 면적 증가는 단지 계수의 워드길이에 좌우된다. OBC를 수정할 때 MUX와 INV가 하나씩 추가되므로 면적 증가는 다음과 같다.

$$M_i = 6 \times N_c\ (TR's), \quad (8)$$

$$I_i = 2 \times N_c\ (TR's). \quad (9)$$

3. FA(Full Adder)

FA는 다른 회로에 비해 면적이 매우 크므로 그 면적을 효과적으로 줄이는 것이 무엇보다도 중요하다. 그림 3에서 OBC를 수정할 때 첨가되는 FA는 입력 하나가 $-C_0/2$로 고정되어 있다. 그러므로 고정된 입력 값이 0인 경우와 1인 경우로 나누어서 원래의 FA를 재구성하면 그림 4와 같다. 예를 들어 계수의 워드길이가 8인 경우 총 8 개의 FA가 필요하게된다. 만일 원래의 전가산기를 쓰게되면 8×30=240 (TR's)이나 그림 4의 FA를 쓰면 계수의각 비트가 모두 1인 최악의 경우 8×14=112로써 FA의 면적을 절반 이상 감소시킬 수 있다. 평균적으로 계

표 5. Control Block의 입출력 비교

	ROM input	MUX1 control input	MUX2 control input
x_{oj}=0, x_{1j}=0	x_{2j} x_{3j}	non inverting	non inverting
x_{oj}=0, x_{1j}=1	$\overline{x_{2j}}$ $\overline{x_{3j}}$	inverting	non inverting
x_{oj}=1, x_{1j}=0	x_{2j} x_{3j}	inverting	inverting
x_{oj}=1, x_{1j}=1	$\overline{x_{2j}}$ $\overline{x_{3j}}$	non inverting	inverting

표 6. Nv=Nc=8 인 경우의 면적 비교

Logic	Architecture using OBC	Modified Architecture	Difference
Decoder	170×4 = 680	94×4 = 376	-284
ROM	128×8 = 1024	64×8 = 512	-512
XOR	7×8=56	=56	
MUX	8×2×6=96	96+8×6 = 144	+48
INV	8×2=16	16+8×2 = 32	+16
Regiter	8×16=128	= 128	
Shift Acc.	8×16=128	= 128	
FA	8×30=240	240+(14×4+12×4) = 344	+104
Total	2368 TR's	1740 TR's	-628

수의 각 비트의 절반이 1의 값을 갖는다고 할 때를 가정해 계산해보면 4×4 + 4×12 = 104(TR's) 이다. 이를 식으로 표현하면 다음과 같다.

$$F_i = \frac{N_c}{2} \times 14 + \frac{N_c}{2} \times 12 = 13N_c \ (TR's). \quad (10)$$

4. XOR

그림 3의 control block을 보면 OBC를 수정할 때 (N_v-2)만큼의 XOR 게이트가 증가하게 된다. 그런데, control block의 입력과 출력을 분석해 보면 수정된 회로의 XOR 게이트 개수를 원래의 OBC와 같은 개수로 줄일 수 있다는 것을 알 수 있다. 즉, 그림 3의 control block의 동작은 표 5와 같이 ROM 입력 및 MUX1, MUX2의 control 입력에 관한 테이블로 정리 할 수 있다. 표 5를 참조하여 그림 2의 control block을 재구성하면 그림 5와 같이 단순화할 수 있다.

V. OBC와 수정된 OBC의 비교

OBC를 수정할 때 감소 및 증가한 TR 개수는 D_r+R_r이고 총면적증가량은 $M_i+I_i+F_i$ 이므로

$$D_r+R_r \rangle M_i+I_i+F_i \quad (11)$$

를 만족하는 N_t와 N_c를 가질 때 본 논문에서 제시한 구조를 사용하면 면적 감소의 효과를 얻을 수 있다.

그림 6(a)은 N_c가 4~16, N_t가 4~16 일 때 각각의 경우에 대한 면적감소량을 백분율로 도시한 것이다. 그림 6을 보면 벡터크기가 5이하인 경우에는 오히려 면적이 증가하는 역효과가 발생함을 알 수 있다. 그러나 모든 N_c에 대해 N_t가 6인 시점부터 면적 감소의 효과가 발생하여서 N_t가 점차적으로 커짐에 따라 최대 약 50%까지 면적을 감소시킬 수 있음을 알 수 있다. 즉, ROM 면적의 감소에 결정적 영향을 미치는 요소는 입력 벡터 수이며 그 이유는 N_c가 커지면 ROM 면적의 감소율이 커지는 반면 기타 FA나 MUX의 면적도 그만큼 증가하므로 전체 면적의 감소율은 별로 커지지 않으나 N_t가 커지면 FA나 MUX의 면적은 증가하지 않고 ROM의 면적만 줄어들게 되기 때문이다.

그림 6(b)는 N_c= 8, N_t= 4~16 에 대해 면적감소량을 백분율로 도시하였다. 표 6은 계수 워드 길이 8, 입력 벡터 크기 8인 경우에 대해 OBC와 수정된 구조의 면적을 비교 분석해 놓았다.

표 6은 벡터크기와 워드 길이가 8인 경우, DA-OBC에 비해 수정된 DA-OBC의 면적이 약 26.5% 감소됨을 보여준다. 제안한 방법은 CDMA 등 이동통신의 웨이브 쉐이핑 필터[6], DCT 등 벡터의 내적이 응용되는 분야에 사용될 수 있으며 다음 절에서는 DCT 응용예를 보인다.

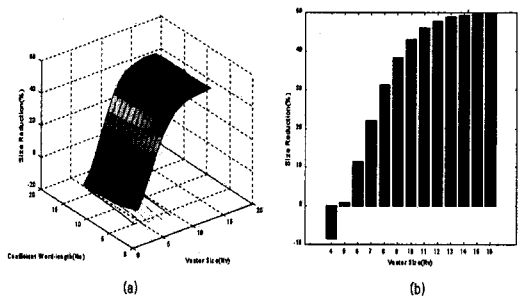


그림 6. (a) Nc=8, Nv=4-6 일 때 면적감소율
(b) Nc와 Nv에 대한 면적감소율

VI. DCT 응용

DCT(type Ⅱ)에 대한 일반식이 식(12)에 주어졌다. N-point DCT는 N개의 독립적인 내적 연산과 동일하다. 그러므로 각각의 내적을 수정된 DA-OBC를 이용하면 전체 연산량을 크게 줄일 수 있다.

$$X(k) = \frac{2}{N} e(k) \sum_{i=0}^{N-1} x(n) \cos \frac{(2n+1)k\pi}{2N},$$
$$k = 0, 1, \cdots, N-1,$$
$$where, \ e(k) = \begin{cases} \frac{1}{\sqrt{2}}, & k=0, \\ 1, & otherwise. \end{cases} \quad (12)$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_1 & C_3 & C_5 & C_7 & C_9 & C_{11} & C_{13} & C_{15} \\ C_2 & C_6 & C_{10} & C_{14} & C_{18} & C_{22} & C_{26} & C_{30} \\ C_3 & C_9 & C_{15} & C_{21} & C_{27} & C_1 & C_7 & C_{13} \\ C_4 & C_{12} & C_{20} & C_{28} & C_4 & C_{12} & C_{20} & C_{28} \\ C_5 & C_{15} & C_{25} & C_3 & C_{13} & C_{23} & C_1 & C_{11} \\ C_6 & C_{18} & C_{30} & C_{10} & C_{22} & C_2 & C_{14} & C_{26} \\ C_7 & C_{21} & C_3 & C_{17} & C_{31} & C_{13} & C_{27} & C_9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}$$

$$where, \quad c_i = \cos \frac{i}{16} \pi. \quad (13)$$

식(12)는 type Ⅱ DCT에 관한 일반적인 식을 나타내고 식(13)은 8-point DCT에 대해 행렬로 전개한 것이다.

식(13)은 코사인 함수의 주기성을 이용하여 식(14)와 같이 2개의 4×4 행렬식으로 나타낼 수 있다.

$$\begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 \\ C_2 & C_6 & -C_6 & -C_2 \\ C_4 & -C_4 & -C_4 & C_4 \\ C_6 & -C_2 & C_2 & -C_6 \end{bmatrix} \begin{bmatrix} x(0)+x(7) \\ x(1)+x(6) \\ x(2)+x(5) \\ x(3)+x(4) \end{bmatrix},$$

$$\begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix} = \begin{bmatrix} C_1 & C_3 & C_5 & C_7 \\ C_3 & -C_7 & -C_1 & -C_5 \\ C_5 & -C_1 & -C_7 & C_3 \\ C_7 & -C_5 & C_3 & -C_1 \end{bmatrix} \begin{bmatrix} x(0)-x(7) \\ x(1)-x(6) \\ x(2)-x(5) \\ x(3)-x(4) \end{bmatrix},$$

$$where, \quad c_i = \cos\frac{i}{16}\pi. \tag{14}$$

식 (14)에 의해 *N*-point DCT는 벡터 크기가 *N*/2인 *N*개의 벡터 내적과 동일하므로 그림 7과 같이 구현할 수 있다. 예로써 다음과 같은 시스템 입력이 주어졌을 때

$$\begin{aligned} x_0 &= 00110010,\ x_1 = 10101010, \\ x_2 &= 01010000,\ x_3 = 00101010, \\ x_4 &= 11001010,\ x_5 = 00110110, \\ x_6 &= 10100100,\ x_7 = 00011000, \end{aligned} \tag{15}$$

X(5)를 구하기 위한 ROM의 값은 다음과 같다.

$$\begin{aligned} c_5 &= 01000111,\ -c_1 = 10000010, \\ c_7 &= 00011000,\quad c_3 = 01101010. \end{aligned} \tag{16}$$

이 때, *X*(5)의 값은 다음과 같이 구해진다.

$$\begin{aligned} X(5) &= (c_5\ -c_1\ c_7\ c_3)\cdot(x_0-x_7\ x_1-x_6\ x_2-x_5\ x_3-x_4) \\ &= c_5(x_0-x_7)+(-c_1)(x_1-x_6) \\ &\quad + c_7(x_2-x_5)+c_3(x_3-x_4) \quad = 0.3651. \end{aligned} \tag{17}$$

위에서 구한 값을 2진수로 표현하면 00101110이 된다. 제안한 수정된 DA-OBC 방법을 이용하여 ROM을 구성한 후 VHDL 시뮬레이션한 결과는 그림 8과 같으며 그림 8의 시뮬레이션 결과(원안의 값)를 보면 *X*(5)=00010111로 표현되는데 이는 ROM 값을 1/2로 스케일링 한 값이므로 결과 값에 2를 곱해주면(shift left) 계산에서 구해진 값과 같은 결과를 얻는다

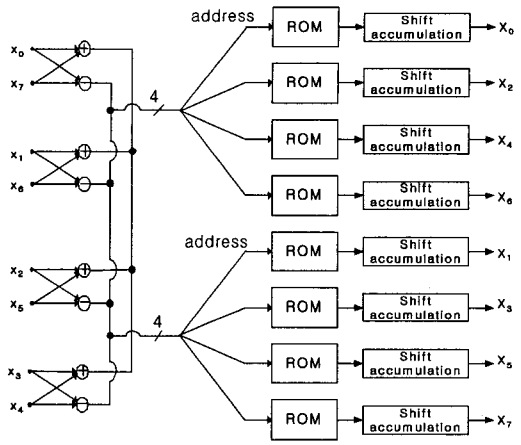


그림 7. 수정된 DA-OBC를 사용한 8-point DCT 구조

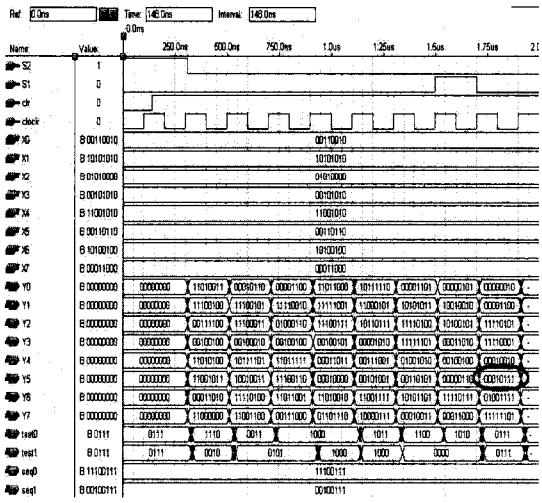


그림 8. 8-point DCT에 대한 VHDL 시뮬레이션

16-point DCT는 입력 벡터 크기가 8인 내적 연산 16개로 구성되므로 워드 길이를 8비트로 양자화할 경우 표 6에 의해 DA-OBC를 사용하면 2,368×16=37,888의 트랜지스터가 필요하고 수정된 DA-OBC를 사용하면 1,740×16=27,840개의 트랜지스터로 DCT 구현이 가능하여 면적이 약 27% 감소된다. IDEC의 셀 라이브러리(IDEC-C631 0.6㎛, 3.3V cell based library, release 9804)를 이용하여 SYNOPSYS로 시뮬레이션 한 결과 입력에서 출력까지의 총 지연시간이 18.42ns로써 NTSC와 PAL에서 표준으로 정한 MPEG-2 이미지 포맷처리 속도 54㎒를 만족한다[7]. 속도가 만족되지 못하는 응용에서는 그림 7의 구조에 파이프라인을 사용하여 critical path를 더욱 감소시켜 사용할 수 있다.

Ⅶ. 결 론

두 벡터의 내적을 계산할 때 ROM을 사용하여 곱셈기의 직접적인 구현 없이 하드웨어 소모를 효과적으로 감소시킬 수 있다. 또한 OBC 방법에 의해 DA에 사용된 ROM 면적을 반으로 줄일 수 있다. 본 논문에서는 OBC의 ROM table을 분석함으로써 OBC에 사용된 ROM 면적을 반으로 줄일 수 있음을 보였다.

그러나 ROM 면적이 감소되는 반면에 덧셈기, XOR, 멀티플렉서 등과 같은 제어 회로가 첨가된다. 덧셈기는 하나의 고정된 입력을 가지므로 일반적인 목적의 덧셈기보다 50%이상 적은 면적을 갖도록 설계할 수 있었고, XOR은 control block의 입출력 관계를 분석함으로써 원래 DA-OBC와 같은 개수로 감소시켰다.

제안된 방법을 반복해서 적용하면 ROM 크기를 최대 2 워드까지 줄일 수 있으나 기타 논리 회로와 critical path가 증가하므로 설계자가 상황에 따라 주어진 속도와 면적을 고려하여 적절히 선택해야 한다.

참 고 문 헌

[1] M. T. Sun, T. C. Chen, and A. M. Gotlieb, "VLSI implementation of a 16× 16 discrete cosine transform," *IEEE Trans. on Circuits and Systems,* vol. CAS-36, no. 4, pp. 610-617, Apr. 1989.

[2] M.T. Sun, L. Wu, and M. L. Liou, "A concurrent architecture for VLSI imple- mentation of discrete cosine transform," *IEEE Trans. on Circuits and Systems,* vol. CAS-34, no. 8, pp. 992-617994, Aug. 1987.

[3] N. Demassieux and F. Jutand, "Orthogonal transforms,"in *VLSI Implementations for Image Communications* (P. Pirsch, ed.), pp. 217-250, Elsevier, 1993.

[4] C. S. Burrus, "Digital filter structures described by distributed arithmetic," *IEEE Trans. on Circuits and Systems,* Dec. 1977.

[5] K. K. Parhi, *VLSI Digital Signal Processing Systems.* John Wiley & Sons, Inc. 1999.

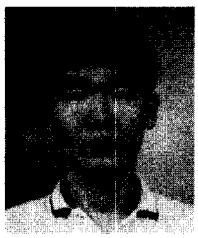
[6] I. Kang, K. I. Yeon, H. C. Jo, J. W. Chong and K. Kim, "Multiple 1:N interpolation FIR filter design based on a single architecture", *ISCAS* 98, Monterey(CA), May, 1998

[7] *SMPTE-standards for advanced tele- vision and high definition production with supplemental standards including ancillary audio,* The Society of Motion Picture and Television Enginners, May 1996.

최 정 필(Jung-Pil Choi) 정회원

1998년 2월 : 전북대학교 제어계측공학과 졸업
2000년 2월 : 전북대학교 정보통신공학과 석사
2000년 2월~현재 : 현대전자
<주관심 분야> IMT2000

성 경 진(Kyung-Jin Sung) 정회원



1998년 2월 : 전북대학교 물리학과 졸업
2000년 2월 : 전북대학교 정보통신공학과 석사
2000년 2월~현재 : LG전자
<주관심 분야> DTV

유 경 주(Kyung-Ju Yoo) 정회원

1998년 8월 : 전북대학교 정보통신공학과 졸업
1999년 2월~현재 : 전북대학교 정보통신공학과 석사과정
<주관심 분야> VLSI 신호처리

정 진 균(Jin-Gyun Chung) 정회원

1985년 2월 : 전북대학교 전자공학과 졸업
1991년 12월 : University of Minesota 전기공학과 석사
1994년 12월 : University of Minesota 전기공학과 박사

1995년 3월~현재 : 전북대학교 전자 · 정보공학부 조교수
1996년 9월~현재 : 전북대학교 정보통신연구소 연구원
<주관심 분야> VLSI 신호처리, 고속 DSL모뎀