論文96-33B-1-4

고속 그래픽 처리를 위한 잉여수계 승산기 설계에 관한 연구

(A study on the design of RNS Multiplier to speed up the Graphic Process)

金 龍 成 *, 趙 源 敬 **

(Yong Sung Kim and Won Kyung Cho)

요 약

실시간 컴퓨터 그래픽 처리를 위하여는 고속 연산(승산 및 가산)회로가 필요하다. 잉여수 체계(RNS: Residue Number System)는 병렬성과 고속성을 갖는 정수연산체계이고, 또한 순환군(cyclic group)은 가산과 승산이 동형인 잉여수 연산을 수행하므로 고속의 승산기와 가산기의 설계가 가능하다. 그러므로, 본 논문에서는 DRNS(Double Residue Number System)를 제안하고, 순환부호(circula- tive code)를 이용한 고속의 잉여수 승산기를 설계하여, 이를 그래픽 프로세서의 연산기로 사용하고자 한다. 설계된 승산기는 TTL소자 74s09, 74s32를 사용한 경우 87MHz속도의 연산이 가능하다.

Abstract

To process computer graphics in real time, the high-speed operations(multiplier and adder)are needed to increase the speed of graphic process. RNS(Residue Number System) is integer number system that has the parallel and high-speed operation. Also, it is able to design both high-speed multiplier and adder, since a cyclic group has an isomorphic relation between multiplication and addition in RNS. So In this paper, DRNS(Double Residue Number System) is proposed, it is used for the multiplier and the adder, which are designed using a circulative code for the high-speed graphic processor in RNS. The designed multiplier would operate with the speed of 87Mzz two TTL using 74s09 and 74s32.

I. 서 론

근래에 이르러 컴퓨터 그래픽은 에니메이션 및 시뮬레이션 분야 등에 이용되고 있으며, 해상도 가 높아짐에 따라 보다 고속의 연산회로를 필요로 하고 있다. 행렬·벡타 연산은 그래픽 도형 생성식에 적합하기 때문에[1] 이를 시스톨릭 어레이 구조로 설계하려는 연구가 활발하다. 그래픽 도형의 생성식은 행렬·벡타 연산으로 표현되므로, 고속의 승산기 및 가산기가 요구된다.[2] [3] [4]

잉여수계(RNS : Residue Number System)는 선택된 모듈리(moduli)에 따라 큰 정수를 작은 잉여수의 집합으로 나누어 표시함으로써, 승산 및 가산등의 연산시 작은 크기의 연산기로 정수연산이 수행되며, 모듈리 사이에 캐리정보가 필요없고 병렬성과 고속성을

* 正會員, 麗州專門大學 電子計算科
(Dept of Coputer Science., Yeo Joo Tchnical College)
** 正會員, 慶熙大學校 電子工學科
(Dept. of Electronic Eng., Kyung Hee Univ.)
接受日字:1994年7月13日, 수정완료일:1995年11月25日

갖으므로 제한된 수범위 내에서 고속의 승산 및 가산이 요구되는 분야에 사용되므로,[5][6] 제한된 수범위를 갖는 그래픽 디스플레이 연산에 적합하다. 잉여수 승산기의 대표적인 설계방법은 ROM연산표를 사용한 방법과 순환군을 이용한 방법이다.[7] 순환군(cyclic group)은 mod p(p:소수)승산이 mod (p-1)의 가산과 동형을 형성하므로, 순환군을 2진 부호화한 순환부호(circulative code)의 부호정합 (code mapping)에 의하여 가산기를 사용한 승산기를 설계할 수 있는 장점을 갖는다.[8] 그러나 일반적인 순환부호 사용시 입출력부에 부호정합용 해독기가 필요하며, 연산기마다 순환부호를 생성하기 위한 기본 순환부호 저장용 ROM 및 부호 순환부가 필요하다.[8] 또한 순환부호를 사용한 잉여수 연산기는 모듈리가 전부 소수이어야 하므로 수범위가 증대하거나, 잉여수를 2진수로 변환시 문제점을 갖는다.

그러므로, 본 논문에서는 제한된 수범위내에서 보다 적은 칩(chip)면적에 고속의 연산회로를 실현하기 위하여, 순환부호 생성부 및 부호정합용 해독기를 필요로 하지 않는 DRNS를 이용한 고속의 잉여수 연산기를 설계하고, 그래픽 정보를 고속으로 디스플레이하기 위한 고속 연산프로세서의 연산회로로 사용하고자한다. 또한, 순환부호를 사용한 연산기 설계시 보다 작은 크기의 연산기로 연산을 수행하며, 잉여수의 2진수 변환을 고속화하기 위하여, 잉여수 체계를 재분할한 DRNS (Double Residue Number System)를 제안한다.

Ⅱ. 잉여수와 순환군

1. 잉여수계의 기본이론

현재 디지탈 연산회로에 많이 쓰이고 있는 2진수계는 가중치 수체계(Weighted Number System)에 속한다. 가중치 수체계는 크기의 비교와 부호 판단이 용이한 장점이 있다. 그러나 상위 비트의 연산시 하위 비트로 부터 전달되어 오는 캐리 정보를 반드시 필요로 하는 것이 단점이다.

잉여수계(RNS)는 비가중치 수체계(Unweighted Number System)로서 연산시 모듈리(moduli)간에 캐리정보가 필요없기 때문에 고속의 병렬연산에 적합하고 승산이 가산과 유사한 정도의 복잡도로 실현되는 장점이 있다.[9] 그러나, 크기 비교와 부호 판단에 부가회로를 필요로 하는것이 단점이다. 이러한 장,단점 때문에 잉여수계는 고속의 승산 및 가산이 필요한 특수한 분야에 이용되고 있다.[5][6]

1) 정수의 잉여수 표현

정수를 잉여수계로 표현하기 위하여는 먼저 서로소인 모듈리(moduli) P를 선택한다.

$P = \{ m_1, m_2, \ldots, m_n \}$ 일때, 표시가능한 정수 X의 범위는

$$0 \le X < M = \prod_{i=1}^{n} m_i$$ 이며, 잉여수계에서 정수 X는 식(1)과 같이 n개 터플(tuple)로 구성된다.

$$X \xrightarrow{RNS} (|x|_{m_1}, |x|_{m_2}, \ldots, |x|_{m_n})$$

(단, $|X|_{m_i} = X \bmod m_i$) (1)

2진수계의 이항연산(Binary operation) $Z = X \circ Y$ 는 잉여수계에서는 다음식과 같이 정의된다.

$$|Z|_{m_i} = \| |x|_{m_i} \circ |y|_{m_i} |_{m_i}$$

(단, '$\circ$'는 +, -, 또는 ×)

$$X \xrightarrow{RNS} (|x|_{m_1}, |x|_{m_2}, \ldots, |x|_{m_n}),$$

$$Y \xrightarrow{RNS} (|y|_{m_1}, |y|_{m_2}, \ldots, |y|_{m_n}) \quad (2\text{-}1)$$

$$Z \xrightarrow{RNS} (|x \circ y|_{m1}, |x \circ y|_{m2}, \ldots, |x \circ y|_{mn}) \quad (2\text{-}2)$$

$|Z|_{m_i} = \| |x|_{m_i} // |y|_{m_i} |_{m_i}$인 제산의 경우는 $||i|_{m_i} |y|_{m_i}|_{m_i}=1$이 성립하는 $|1//y|_{m_i}=|i|_{m_i}$(y,i: 서로소, x/y: 정수)을 사용한다. 위의 관계에 의해 잉여수계는 다음과 같은 특징을 같는다. 즉 모듈리간에 캐리정보가 없으며, 각 모듈러스(modulus)의 연산이 완전히 독립적이기 때문에 병렬구조의 연산회로 설계에 적합하다. 다음은 잉여수계 연산방법에 대한 예이다.

예 1) 정수 X의 잉여수 표현에 대한 예

P = {3, 5}라 할때, $0 \le X < M (= 3 \cdot 5 = 15)$ 이므로 정수 X는 잉여수계에서 표 1과 같이 단일한 2개 터플(tuple)로 표현된다. $x \times y = z$의 승산인 경우 x=3, y=4일 때 $|0|_3 \times |1|_3 = |0|_3$, $|3|_5 \times |4|_5 = |2|_5$ 이므로 결과 z는 (0,2)로 표현되며, 표 1에 의하여 12가 된다.

표 1. 정수 X의 잉여수표현(moduli {3,5} 인 경우)

Table 1. Residue expression of integer X (in case of moduli {3,5}).

X	$\|X\|_3$	$\|X\|_5$	X	$\|X\|_3$	$\|X\|_5$
0	0	0	10	1	0
1	1	1	11	2	1
2	2	2	12	0	2
3	0	3	13	1	3
4	1	4	14	2	4
5	2	0	15	0	0
6	0	1	16	1	1
7	1	2	17	2	2
8	2	3	:	:	:
9	0	4			

2) 잉여수계를 이용한 연산기의 기본 구조.

그래픽 도형의 생성식은 행렬 연산으로 표현되며, 이와 같은 연산은 식(3)과 같이 승산과 가산이 혼합된 누산(MAC:Multiplier with Accumulator)연산을 기본구조로 한다.

$$Z_i = Z_{i-1} + (X_i \times Y_i) \qquad (3)$$

두 정수 X,Y의 누산 연산을 하는 잉여수 연산기의 기본 구조를 그림 1에 나타내었다. 잉여수계에 사용되는 모듈리를 m_1, m_2,...,m_n이라 할 때, 정수 입력 X_i, Y_i는 그림 1의 2진수-잉여수 변환기를 통하여 식(2-1)과 같이 잉여수로 변환된다. 변환된 수는 각 모듈리에 따라 승산 및 가산이 누산 연산부에서 이루어지며, 각 모듈리에서 연산된 결과는 잉여수-2진수 변환기를 통하여 2진수 출력이 발생된다.

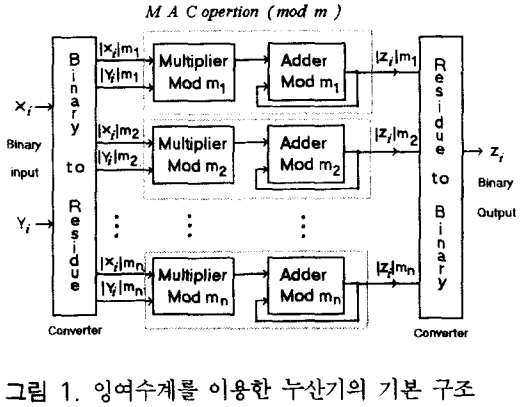


그림 1. 잉여수계를 이용한 누산기의 기본 구조

Fig. 1. Basic configuration of MAC(multiplier with accumulator) using RNS.

Ⅲ. 순환군을 이용한 모듈러 연산

1. 순환군

집합 G가 공집합이 아니고 정수 a,b에 대하여 a,b ∈ G이고 , 연산 '∘'가 a∘b ∈ G일 때, 이 연산이 결합법칙, 교환법칙이 성립하고 항등원 및 역원이 존재하면, 연산 '∘'에 관하여 집합 G는 군을 이루며, 〈G,∘〉로 표기한다. 또한 군 〈G,∘〉에 대하여 g ∈ G라 할 때 생성원(generator)g에 의해서 생성된 순환 부분군 〈g〉는 다음과 같으며, G=〈g〉 인 군을 순환군이라 한다.[10]

$$\{g^n \mid n \in Z\} = \{..., g^2, g^1, g^0, g^{-1}, g^{-2}, ...\}$$

(g:생성원, g^0 : 항등원, Z,g,n :정수)

순환군 G가 위수 n의 유한군이면 $G=\{g^0, g^1, g^2, ..., g^{n-1}\}$이고 $g^n = g^0$이다. 모듈러 연산은 순환군(Cy-clic Group)을 형성하므로 모듈러 연산기는 군론을 기초로 하여 설계될 수 있다. 연산 '×'에 대한 n차 순환군 {G,×}의 연산은 표 2와 같다[8][9]

표 2. n차 순환군

Table 2. Oder n cyclic group.

×	g^0	g^1	g^2	.	.	.	.	g^{n-1}
g^0	g^0	g^1	g^2	.	.	.	.	g^{n-1}
g^1	g^1	g^2	g^3	.	.	.	.	g^0
g^2	g^2	g^3	g^4	.	.	.	.	g^1
:	:							:
:	:							:
g^{n-1}	g^{n-1}	g^0	g^1	.	.	.	.	g^{n-2}

g^0 = 항등원

g^1 = g = 생성원

순환군 {G;×}의 정의로 부터 mod m가산은 m차 순환군이고, 임의의 mod p승산은 (p는 소수, 0 디지트는 제외) (p-1)차 순환군이다. 일반적으로 mod p승산에 대하여, 생성원은 Fermat의 이론으로 부터 다음과 같이 구할 수 있다. $|g^{p-1}|_p$가 가장 작은 g^{p-1} mod p의 잉여수일 때, g가 조건 $|g^{p-1}|_p = 1$을 만족한다면 정수 g 〈 p는 mod p승산의 생성원이다. 예를들어, 발생기가 3인 경우 mod 7승산은 6차 순환군이고, 표3 (a)와 같이 순환군을 형성하기 때문에 모듈러 연산에 이용할 수 있다. 임의의 순환군에 대해서 p가 소수인 경우 mod p 승산 (0 digit 제외)에 대한 순환군은 mod (p-1) 가산의 순환군과 1대 1 대응하여 사상(mapping)될 수 있으므로 동형(isomorphic)이다.

표 3. mod 7 승산과 mod 6가산 동형관계
(a) mod 7 승산(g=3)
(b) mod 6 가산
Table 3. Isomorphic relation of mod 7 multiplication and mod 6 addition.
(a) mod 7 multiplication(g=3)
(b) mod 6 addition

(a)

×	1	3	2	6	4	5
1	1	3	2	6	4	5
3	3	2	6	4	5	1
2	2	6	4	5	1	3
6	6	4	5	1	3	2
4	4	5	1	3	2	6
5	5	1	3	2	6	4

(b)

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

표 3에 g=3인 경우 mod 7승산의 순환군과 mod 6가산과의 동형관계를 나타내었으며, 표 4에 이에 대한 사상(mapping)을 나타내었다. 표 4에서 mod 7 승산은 mod 2 및 mod 3의 두개의 가산과 1대 1 대응되게 표현되므로, mod 2가산과 mod 3가산은 mod 7승산의 준동형(homomorphism)이며, $H_1\{1,2,4\}=0$, $H_1\{3,6,5\}=1$, $H_2\{1,6\}=0$, $H_2\{3,4\}=1$, $H_2\{2,5\}=2$와 같은 대응관계가 성립된다. 그러므로 mod p 승산(P:소수)은 동형관계에 의해 mod (p-1)의 가산으로 수행될 수 있으며, 준동형 사상에 의해 mod j 가산 및 mod k 가산(j,k:정수, (p-1)=j × k)에 의해서 수행될 수 있다. 이와같은 특성을 사용하면 잉여수 승산기를 설계할 수 있다.

표 4. mod 7 승산과 mod 6가산, mod 2가산, mod 3가산의 사상관계
Table 4. Mapping relation of mod 7 multiplication and mod 6 addition.

mod 7승산 $x=\|3^y\|_7$	mod 6가산 $\|y\|_6$	mod 2 가산 $\|y\|_2$	mod 3 가산 $\|y\|_3$
1	0	0	0
3	1	1	1
2	2	0	2
6	3	1	0
4	4	0	1
5	5	1	2

2. 순환 부호를 이용한 연산

표 3에 나타난 순환군을 사용하여 연산기의 논리회로를 설계하는 경우 순환군을 2진 부호화하여야 한다. 순환군을 2진 부호화하는 기본적인 방법은 순환군과 1대 1 대응된 2진 부호화를 수행하여 논리회로로 설계하는 것이다. 표 2는 mod n 승산의 순환군인 경우 기본요소의 수가 n개 이고, mod p의 승산을 수행하므로 p = n 이며, g^m은 $0 < g^m \le (p-1)$, $(0 < m \le n-1)$ 이므로 각 기본요소당 소요되는 비트수 b는 $b = \lceil \log_2 (n-1) \rceil$ 비트이다. 그러므로 순환군의 각 요소를 1 대 1 사상하여 2진 부호화하는 경우 1개 연산기에 $n^2 \times \lceil \log_2 (n-1) \rceil$ 이상의 논리연산이 필요하므로, 연산기의 크기를 감소시키기 위하여 표 5에 순환부호(Circulative code)를 사용한 n차 순환군의 2진 부호화를 나타내었으며, 순환부호는 다음의 특성을 갖는다. [8]

1) 순환군내에 2개의 연속된 요소에 대한 2진 부호를 m비트로 구성하는 경우, 앞 부호의 우측 m-1비트는 다음 부호의 좌측 m-1비트와 동일하다.
2) 각각의 2진 부호는 서로 구분이 가능하며, n개 2진 부호는 순환 루프(loop)를 형성한다. 또한, n개 순환부호의 n번째 순환부호의 다음은 1번째 순환 부호와 동일하다.

표 5. n차 순환군을 구성하는 잉여수의 순환부호($2^{m-1}<n\le 2^m$)
Table 5. Circulative code of RNS in order-n cyclic group ($2^{m-1}<n\le 2^m$).

잉여수 $\|X\|_n$ \ 순환부호		X_1 X_2 . . . . . X_m
0	x_1	b_1 b_2 . . . . . b_m
1	x_2	b_2 b_3 . . . . . b_{m+1}
.	.	. .
.	.	. .
.	.	. .
n-1	x_{n-1}	b_n b_1 . . . . . b_{m-1}

(단 $0\le\|X\|_n\le n-1$, if $\|X\|_n=i$ then $X_i=1$ else 0, n:정수)

표 5의 n개의 요소에 의해 R = X + Y의가산을 수행하는 경우, 식 (4-1)에 의해서 입력 X대한 BX를 P에 정합(Mapping)하고 PY를 R에 정합하여 최종적인 X + Y의 결과를 얻게 된다.

$P_i = X_i, \quad 1 \le i \le m$

$$\begin{bmatrix} p_{m+1} \\ p_{m+2} \\ \cdot \\ \cdot \\ \cdot \\ p_n \end{bmatrix} = \begin{bmatrix} b_{m+1} & b_{m+2} & b_{m+3} & \cdots & b_m \\ b_{m-2} & b_{m+3} & b_{m+4} & \cdots & b_{m-1} \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ b_n & b_1 & b_2 & \cdots & b_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \quad (4\text{-}1)$$

$$\begin{bmatrix} R_1 \\ R_2 \\ \cdot \\ \cdot \\ \cdot \\ R_m \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & \cdots & P_n \\ P_2 & P_3 & P_4 & \cdots & P_1 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ p_m & p_{m+1} & p_{m+2} & \cdots & p_{m-1} \end{bmatrix} \begin{bmatrix} x_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} \quad (4\text{-}2)$$

표 6. mod 5 연산일 때의 순환 부호
Table 6. circulative code of mod 5 operation.

잉여수	X	X_1 X_2 X_3	X_4 X_5
0	x_1	0 0 0	1 1
1	x_2	0 0 1	1 0
2	x_3	0 1 1	0 0
3	x_4	1 1 0	0 0
4	x_5	1 0 0	0 1

표 6에서 mod 5인 경우의 순환 부호를 나타내었으며, 이 경우 R= X + Y의 가산과정의 예는 다음과 같다. $x_1 = \overline{X}_1\ \overline{X}_3$, $x_2 = \overline{X}_2 X_3$, $x_3 = X_2 X_3$, $x_4 = X_1 X_2$, $x_5 = X_1\ \overline{X}_2$, $y_1 = \overline{Y}_1\ \overline{Y}_3$, $y_2 = \overline{Y}_2 Y_3$, $y_3 = Y_2 Y_3$, $y_4 = Y_1 Y_2$, $y_5 = Y_1\ \overline{Y}_2$으로 X 및 Y가 입력된다.

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \quad (4\text{-}3)$$

$$\begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 & P_5 \\ P_2 & P_3 & P_4 & P_5 & P_1 \\ P_3 & P_4 & P_5 & P_1 & P_2 \\ P_4 & P_5 & P_1 & P_2 & P_3 \\ P_5 & P_1 & P_2 & P_3 & P_4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} \quad (4\text{-}4)$$

R = X + Y의 연산의 경우, X = 1, Y = 2이면, 식 (4-3)에서 '00110'이 선택되며,

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Y에 의해 식 (4-4)에서 순환된 부호 P_3 ... P_5 P_1 선택되어 결과는 3이 된다.

$$\begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 & P_5 \\ P_2 & P_3 & P_4 & P_5 & P_1 \\ P_3 & P_4 & P_5 & P_1 & P_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} \quad (4\text{-}5)$$

또한, 표 6에서 순환부호는 X_1, X_2, Xc_3만으로 부호의 구분이 가능하므로 식 (4-5)와 같이 3비트만으로 출력할 수 있다.

순환 부호를 이용한 승산은 mod p 승산(p:소수)일 때 표 4의 사상관계를 이용하여 설계하며, 동형관계에 따라 mod (p-1)가산을 사용하는 방법과 준동형관계에 따라 mod j, mod k용 가산기를 사용하는 두가지 방법이 있다. 두경우 모두 사상관계에 대한 입출력 해독기가 요구되는데, 준동형관계에 따라 가산기를 사용하는 경우는 작은 크기의 가산기로 승산이 가능하지만 모듈리가 커질수록 입출력용 해독기의 크기가 가산부보다 커지는 단점이 있으므로, 본 논문에서는 mod (p-1)가산기를 이용하고, 입출력 해독기가 생략되는 순환부호를 사용하여 고속의 잉여수 승산기를 설계하고자한다.

Ⅳ. 고속 잉여수 연산회로의 설계

잉여수계 연산은 입력된 수를 선택된 모듈리에 따라 분할하고 각 모듈리 별로 연산하므로, 연산기의 크기가 감소되며, 모듈리간에 캐리정보가 필요없으므로 고속의 연산을 수행할 수 있다. Ⅲ장의 순환부호를 이용하여 고속의 잉여수 연산기를 설계하는 경우, 각 모듈리는 소수를 사용하여 한다는 제약점을 갖는다. 모듈리가 서로소가 아닌 소수만을 사용할 경우, 연산기의 크기가 증가하며, 잉여수를 2진수로 변환시 변환기의 연산속도가 저하되는 문제점을 갖는다. 그러므로, 본논문에서는 DRNS를 제안하여 연산기의 크기를 감소하고, 순환부호를 사용한 승산기를 개선하여, 그래픽 정보의 고속 디스플레이에 필요한 고속 연산프로세서의 연산회로로 사용하고자 한다.

1. DRNS(Double Residue Number System)

DRNS는 1차 선택된 모듈리에 따라 표현된 잉여수

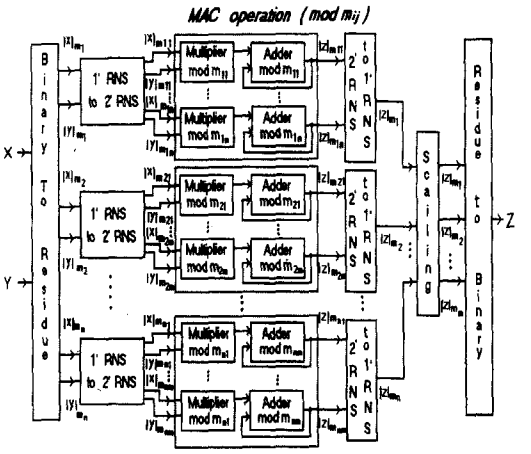


그림 2. DRNS를 이용한 누산기의 기본 구조
Fig. 2. Basic configuration of MAC(multiplier and accumulator) using DRNS.

를 2차 선택된 모듈리에 따라 재분할함으로써 보다 작은 크기의 연산기로 고속의 연산이 가능하다. 정수를 잉여수계로 표현하기 위하여 1차 선택된 모듈리를 P라 하고, 2차 선택된 모듈리를 P_i라 할 때 다음과 같이 표현되며,

$$P=\{m_1, m_2, \cdots, m_n\},\ P_i=\{ m_{i1}, m_{i2}, \cdots, m_{ik} \}$$
$$(i=1,\cdots,n;\ n,k:\text{정수})$$

표시 가능한 수 X의 범위는 $0 \le X < M = \prod_{i=1}^{n} m_i$ 이며, $0 \le m_i < M_K = \prod_{j=1}^{k} m_{ij}$이다.
정수 X는 DRNS에서 다음과 같이 n×k개 터플(tuple)로 구성된다.

$$X \xrightarrow{1'RNS} (|x|m_1, |x|m_2, \cdots, |x|m_n),$$

$$|x|m_i \xrightarrow{2'RNS} (||x|m_i|m_{i1}, ||x|m_i|m_{i2},\ldots,||x|m_i|m_{ik})$$

2진수계의 이항연산 $Z = X \circ Y$ 는 DRNS에서는 식(5)와 같이 정의된다.

$$|Z|m_{ij} = |||x|m_i|m_{ij} \circ ||y|m_i|m_{ij}|m_{ij}$$
$$(\text{단, } '\circ' \text{는 } +, -, \text{또는} \times) \qquad (5\text{-}1)$$

$$(||Z|m_i|m_{i1}, ||Z|m_i|m_{i2},\ldots,||Z|m_i|m_{ik}) \xrightarrow{2'\ RNS\ to\ 1'\ RNS} |Z|m_i \qquad (5\text{-}2)$$

$$(|Z|m_1, |Z|m_2,\ldots,|Z|m_n) \xrightarrow{RNS\ to\ Binary} Z \qquad (5\text{-}3)$$

Ⅱ장의 식(3)과 같이 승산과 가산이 혼합된 누적연산을 기본구조로 하는 DRNS연산기의 기본 구조를 그림 2에 나타내었다. 그림 1과의 차이점은 두번의 잉여수 변환을 위하여 두개의 변환기를 사용하는 점과 각 모듈리별로 연산된 결과가 2번의 변환을 통하여 정수로 변환하는 부분이다. 그래픽 도형 생성시에 선분 및 원도형을 발생하는 경우 정수 × 실수 + 정수연산을 수행하며, 피승수는 실수(삼각함수,기울기,...등)이므로, 정수연산을 수행하는 잉여수 연산기 사용시 실수를 k(k:정수)배한 정수로 입력하고 최종 연산결과를 1/k로 축소하여야 한다. 그러므로 그림 2에 최종 연산결과를 축소하는 스케일링(scaling) 연산부를 추가하였다.

위의 관계에 의해 DRNS는 1차 잉여수계와 비교하여 다음과 같은 특징을 갖는다. 즉 DRNS는 1차 잉여수 변환시 선택된 모듈리를 2차 변환시 보다 작은 모듈리로 분할하여 연산하므로 연산기의 크기가 감소한다.

또한, 1차 잉여수만를 사용하는 경우 순환부호를 사용한 연산시 모듈리를 전부 소수로 선택해야하므로, 순환부호를 사용하지 않는 경우와 비교할 때 동일한 수 범위의 연산시 선택하여야할 모듈리가 커지므로 연산기 크기가 증가한다. 그러나 DRNS의 경우 1차 잉여수계의 모듈리는 서로소로 하고, 2차 잉여수계는 소수를 사용하므로 연산기의 크기가 증대하는 단점을 감소시켰다. 그러므로, Ⅲ장의 순환부호를 사용한 연산기를 사용하여도 연산기의 크기가 증대하지 않으며, 고속의 연산기 설계가 가능하다.

2. 순환 부호를 이용한 연산회로의 구성

Ⅲ장 3절의 순환부호를 사용하여 잉여수 연산기를 설계하는 경우, 가산기는 표 6과 같은 순환부호를 사용하여 설계할 수 있다. 그러나 모듈러 p의 승산기는 모듈러 (p-1)의 가산기를 이용하여 설계하므로, 승산기의 입력 및 출력시 표 4의 동형관계에 따라 부호 정합(mapping)용 해독기(decoder)가 필요하며, 승산기의 입력 '0'(m_0=0)인 경우는 출력이 '0'이 되도록 한다. 모듈러 7 승산과 모듈러 6 가산의 부호 정합관계에 따른 순환부호를 표 7에 나타내었다. 순환부호 1은 표 6과 같이 참고문헌 [8]에서 사용한 순환부호이다. 연산결과 전달시 순환부호1을 사용하는 경우, 하위 4비트의 정보는 전송할 필요없다.

표 7. mod 6 가산과 mod 7승산의 순환부호 및 부호 정합 관계
Table 7. Circulative code and mapping relation of mod 6 addtion and mod 7 multiplication.

mod_6 가산			mod_7 승산(g=3)		
잉여수	순환부호1 $a_5a_4a_3a_2a_1a_0$	순환부호2 $a_5a_4a_3a_2a_1a_0$	잉여수	순환부호1 $m_6m_5m_4\ m_3m_2m_1m_0$	순환부호2 $m_6\,m_5\,m_4\,m_3\,m_2\,m_1\,m_0$
0	0 0 0 1 1 1	1 1 1 1 1 0	1	0010110	1111101
1	0 0 1 1 1 0	1 1 1 1 0 1	3	1011000	1110111
2	0 1 1 1 0 0	1 1 1 0 1 1	2	0101100	1111011
3	1 1 1 0 0 0	1 1 0 1 1 1	6	10 00101	0111111
4	1 1 0 0 0 1	1 0 1 1 1 1	4	0110001	1101111
5	1 0 0 0 1 1	0 1 1 1 1 1	5	1100010	1011111
			0	000 1011	1111110

그러나 연산시 7비트를 모두 사용하므로 부호생성을 위한 부가회로가 필요하며, 식 (6-1)과 같이 부호정합용 해독기(d_i= mod 6 가산기 입력 'i', i=0,…,5)가 필요하다.

$$d_0 = \overline{m_6}\ \overline{m_5}\,m_4,\ d_1 = m_6m_4,\ d_2 = \overline{m_6}\ \overline{m_5}\,m_4,\ d_3 = m_6\,\overline{m_5}\ \overline{m_4},$$

$$d_4 = m_5m_4,\ d_5 = m_6m_5,\ s = m_6m_5m_4$$

(s: '0'검출용, d_i:해독기 출력,i=0,…,5) (6-1)

본 논문에서는 부호 정합용 해독기를 사용하지 않기 위하여 표 7의 순환부호 2를 사용하고자 한다. 순환부호 2는 부호 정합시 각 비트가 1 대 1 대응되므로 식(6-2) 같이 입출력선의 재배열로 부호정합이 가능하며, 부호를 재생성하기 위한 부가회로가 필요하지 않다.

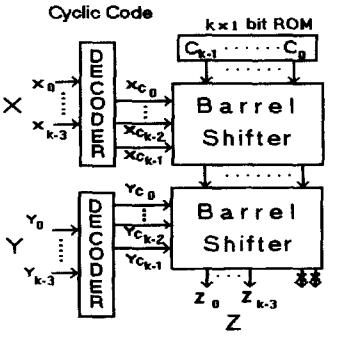
$$a_5 = m_5,\ a_4 = m_4,\ a_3 = m_6,\ a_2 = m_2,\ a_1 = m_3,\ a_0 = m_1.$$

$$s = m_0 \text{ (s:'0'검출용)} \quad (6\text{-}2)$$

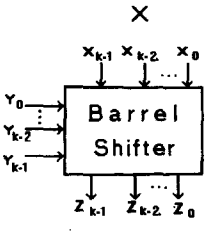
순환 부호를 이용한 연산기를 다음과 같이 설계하고자 한다.

1) 배럴 쉬프터(Barrel Shifter)를 사용한 순환 부호용 연산기의 설계

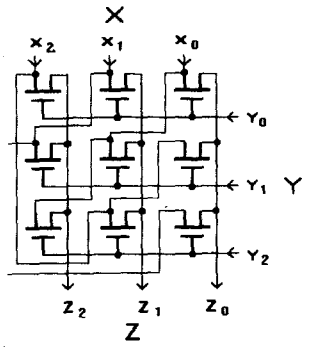
입력 x_i, 선택 y_i, 출력 z_i 인 배럴 쉬프터($0 \le i \le 2$,크기:3×3)의 기본구조를 그림 3(c)에 나타내었다.[11] 배럴 쉬프터는 선택선 y_i를 선택함에 따라, 입력된 데이타의 i비트 순환된 결과를 출력하며, i번 순환된 속도가 동일하므로, 배럴 쉬프터를 이용하여 순환부호용 연산기를 설계하고자 한다. 배럴 쉬프터를 이용한 모듈러 p용 순환부호 가산기를 그림 3에 나타내었다. 그림 3(a)의 순환부호1을 사용하는 경우 입력 및 출력은 P 비트에서 $k=\lceil \log_2 p \rceil$비트만 사용한다. 기본 순환부호(모듈러 6가산의 경우: 000011)p 비트를 저장하고, 피가수 x를 해독한 출력이 배럴 쉬프터(p×p)의 선택선에 입력되어 식(4-3)과 같은 정합이 이루어진다. 생성된 순환부호1은 다음단의 배럴 쉬프터 (p×k)로 입력되고, 가수 y의 해독된 출력에 의해 순환부호의 가산이 이루어져 출력 z가 생성된다. 순환부호2를 사용하는 경우는 p비트로입력된 순환부호2를 그대로 사용하므로 해독기 및 식(4-3)과정이 생략되므로, 그림 3(b)에서 식(4-4)와 같이 정합용 배럴 쉬프터(pxp) 1개에 의하여 가산기가 구성된다.



(a)



(b)



(c)

그림 3. 배럴 쉬프터를 사용한 순환부호용 가산기 (k=⌈log2p⌉)
(a) 순환부호1을 사용하는 경우
(b) 순환부호2를 사용하는 경우
(c) 배럴 쉬프터 기본형태 (3 × 3)

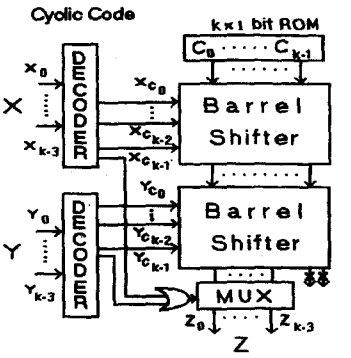
Fig. 3. Adder for circulative code using barrel shifter (k=⌈log2p⌉).
(a) In case of using circulative code 1 (b) In case of using circulative code 2
(c) Basic configuration of barrel shifter (3 × 3)

배럴 쉬프터를 이용한 순환 부호용 승산기를 그림 4에 나타내었다. 순환부호용 승산기는 모듈러 p(p:소수)승산의 경우 모듈러 (p-1)의 가산기를 사용하므로 그림 3의 순환부호용 가산기와 거의 동일한 구조를 갖는다. 순환부호1을 사용하는 경우는 입출력 부분에 식(6-1)의 부호정합용 해독기를 사용하며, 순환부호2의 경우는 입출력에 식(6-2)에 따라 부호정합을 위한 배열선의 재배열을 수행하면 되므로 부가회로가 필요하지 않다. 또한 승산기에 입력된 두수중에서 하나 이상이 '0'이면 출력은 '0'이 되여야 한다. 그러므로 순환부호1의 경우는 $m_{p-1}m_{p-2}...m_{p-k-1}=0$을 검출되면 출력을 '0'으로 하며, 순환부호2의 경우는 $m_0=0$이 검출되면 출력을 '0'으로 한다.

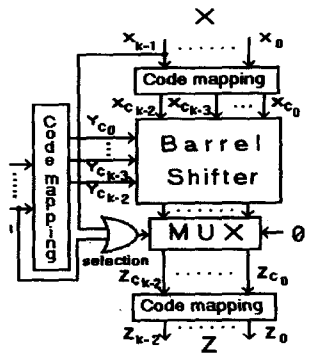
배럴 쉬프터를 사용한 순환부호2용 연산회로는 회로가 보다 간결하며, i(i≥0)번 순환된 속도가 동일하므로 고속의 처리가 가능하다. 그러므로 가산 및 승산을 요구하는 그래픽처리에 유용할뿐만 아니라 대량의 데이타를 처리하기 위한 여러 알고리즘에 적용시 설계가 단순해지며 고속의 처리를 할 수 있는 장점을 갖는다.

3. 2진수의 잉여수계 변환기, 잉여수의 2진수 변환기 및 스케일링 연산기의 설계

Ⅳ장 1절의 잉여수 연산을 하기 위하여서는 2진수계를 잉여수계로 변환하여야 한다. 또한 연산된 결과를 2진수계로 변환하여야 한다. 이 경우 변환기의 속도가 느리면 전체적인 연산속도가 저하되므로 변환기의 속도를 향상시키기 위하여, Ⅳ장 2절에서 설계한 배럴 쉬프터를 이용한 연산기를 사용하여 변환기를 설계하고자 한다.



(a)



(b)

그림 4. 배럴 쉬프터를 사용한 순환부호용 승산기
(a) 순환부호1을 사용하는 경우
(b) 순환부호2를 사용하는 경우

Fig. 4. Multiplier for circulative code using barrel shifter.
(a) In case of using circulative code 1
(b) In case of using circulative code 2

1) 2진수의 잉여수 변환기

, n비트 의 2진수 A를 잉여수로 변환하는 기본식은 식 (7-1)과 같다.

$$|x|m_i = \left|\sum_{k=0}^{n} a_k \cdot 2^k\right|m_i \qquad (7\text{-}1)$$

x의 입력 범위가 0~63(n=5)이고, m_i=5인 경우의 예를 들면, 입력된 2진수를 식(7-2)와 같이 3비트씩 분리하여 잉여수계로 변환한 후 가산을 수행하면 그림

5와 같이 작은 크기의 연산기로 잉여수 변환을 고속으로 수행할 수 있다.

$$|x|_{m_i} = \left|\sum_{k=0}^{2} a_k \cdot 2^k\right|_{m_i} + \left|\sum_{k=0}^{2}(a_{k+3}\cdot 2^k)2^3\right|_{m_i}\Big|_{m_i} \quad (7\text{-}2)$$

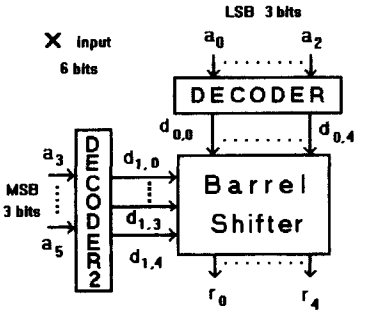


그림 5. 2진수를 잉여수계로 변환하는 회로
Fig. 5. Conversion circuit for binary number to RNS.

그림 5에서 $\left|\sum_{k=0}^{2} a_k \cdot 2^k\right|_{m_i}$연산은 해독기1에서 수행되며, $\left|\sum_{k=0}^{2}(a_{k+3}\cdot 2^k)2^3\right|_{m_i}$연산은 해독기2에서 실행된다. 그림 5의 상단 기본 순환부호와 해독기의 출력에 의해 순환부호가 발생되고, 그림 3(b)의 배럴 쉬프터를 이용한 순환부호2용 가산기와 동일한 방법으로 식 (7-2)의 모듈러 가산이 수행된다. 그러므로 입력된 2진수(a_0~a_2)는 잉여수계로 변환되어 순환부호(r_0~r_4)로 출력된다.

표 8 (a)는 2진수-잉여수 변환기의 해독기 출력을 나타내었으며, 입출력 사이에 1 대 1 정합이 이루어진다. 그러므로 입력을 a_k, 해독기1의 출력을 $d_{0,i}$, 해독기 2의 출력을 $d_{1,i}(0\le i\le 4)$라고 하면,

식 (8-1)과 식(8-2)으로 표현할 수 있다.

$$d_{00}= \bar{a}_0\bar{a}_1\bar{a}_2 + a_0\bar{a}_1a_2,\ d_{01}= a_0\bar{a}_1\bar{a}_2 + \bar{a}_0a_1a_2,$$

$$d_{02}= \bar{a}_0a_1\bar{a}_2 + a_0a_1a_2,\ d_{03}= a_0a_1\bar{a}_2,\ d_{04}= \bar{a}_0\bar{a}_1a_2 \quad (8\text{-}1)$$

$$d_{10}= a_3\bar{a}_4a_5,\ d_{11}= \bar{a}_3a_4a_5 + a_3a_4\bar{a}_5,\ d_{12}= \bar{a}_3\bar{a}_4a_5,$$

$$d_{13}= a_3\bar{a}_4\bar{a}_5 + \bar{a}_3a_4a_5,\ d_{14}= a_3a_4\bar{a}_5 \quad (8\text{-}2)$$

그림 2의 DRNS회로를 사용하는 경우 잉여수로 변환하는 과정이 2번 필요하다. x의 입력 범위가 0~63(n=5)이고, 첫번째 모듈리가 11이고 두번째 모듈리가 5인 경우 두번의 변환과정을 표 8(b)에 나타내었다. 입력 2진수과 잉여수로 두번 변환된 결과가 1 대 1 대응되므로 해독기 1과 해독기 2가 식 (8-3)과 식 (8-4) 로 표현된다. 그러므로 그림 6과 동일한 1개 변환기를 사용하여 두번의 잉여수 변환을 수행할 수 있다.

표 8. 2진수-잉여수 변환기의 해독기 출력.
(a)2진수-잉여수 변환 1번 수행하는 경우
(b)2진수-잉여수 변환 2번 수행하는 경우

Table 8. Decoder output of binary number to RNS converter.
(a)In case of once conversion binary number to RNS
(b) In case of twice conversion binary number to RNS

(a)

2진수 $a_2a_1a_0$	mod 5 해독기1 $d_{04}d_{03}d_{02}d_{01}d_{00}$	2진수 $a_5a_4a_3$	mod 5 해독기2 $d_{14}d_{13}d_{12}d_{11}d_{10}$
000	0 0 0 0 1	000	0 0 0 0 0
001	0 0 0 1 0	001	0 1 0 0 0
010	0 0 1 0 0	010	0 0 0 1 0
011	0 1 0 0 0	011	1 0 0 0 0
100	1 0 0 0 0	100	0 0 1 0 0
101	0 0 0 0 1	101	0 0 0 0 1
110	0 0 0 1 0	110	0 1 0 0 0
111	0 0 1 0 0	111	0 0 0 1 0

(b)

2진수 $a_2a_1a_0$	mod11	mod 5 해독기1 $d_{04}d_{03}d_{02}d_{01}d_{00}$	2진수 $a_5a_4a_3$	mod 11	mod 5 해독기2 $d_{14}d_{13}d_{12}d_{11}d_{10}$
000	000	0 0 0 0 1	000	0000	0 0 0 0 1
001	001	0 0 0 1 0	001	0100	0 1 0 0 0
010	010	0 0 1 0 0	010	0101	0 0 0 0 1
011	011	0 1 0 0 0	011	0010	0 0 1 0 0
100	100	1 0 0 0 0	100	1010	0 0 0 0 1
101	101	0 0 0 0 1	101	0111	0 0 1 0 0
110	110	0 0 0 1 0	110	0100	1 0 0 0 0
111	111	0 0 1 0 0	111	0001	0 0 0 1 0

$$d_{00}= \bar{a}_0\bar{a}_1\bar{a}_2 + a_0a_1\bar{a}_2,\ d_{01}= a_0\bar{a}_1\bar{a}_2 + \bar{a}_0a_1a_2,$$

$$d_{02}= \bar{a}_0a_1\bar{a}_2 + a_0a_1a_2,\ d_{03}= a_0a_1\bar{a}_2,\ d_{04}= a_0\bar{a}_1\bar{a}_2 \quad (8\text{-}3)$$

$$d_{10}= \bar{a}_3\bar{a}_5 + \bar{a}_3\bar{a}_4,\ d_{11}= a_3a_4\bar{a}_5,\ d_{12}= a_3\bar{a}_4a_5 + a_3a_4a_5,$$

$$d_{13}= a_3\bar{a}_4\bar{a}_5,\ d_{14}= \bar{a}_3a_4a_5 \quad (8\text{-}4)$$

2) 잉여수계의 2진수 변환기

잉여수계를 2진수로 변환하는 알고리즘은 중국 잉여수 정리(CRT:Chinese Remainder Theorem)와 혼합기수변환(MRC: Mixed Radix Conversion)이 있으며, 회로구성에 대하여는 이미 많은 연구가 실행되었다. 그러나 이들 알고리즘들은 고속의 승산기 또는 가산기를 필요로 하므로 설계된 승산기의 효율성을 저하시킨다. 그러므로 본 논문에서는 그래픽 처리에 적합한 수범위에서 혼합 기수변환을 적용하며, 설계된 승산기 및 가산기를 이용하여 2진수변환기를 설계하고자 한다. 2진수로 변환된 수는 X, 모듈리가 m_0, m_1, m_2 3개이고 최종 잉여수 연산결과가 r_0, r_1, r_2인 경우 혼합기수변환의 기본식은 식 (9-1)이며, 계수 $a_i(0 \le i \le 2)$는 식(9-2)로 생성된다.[9]

$$X = |a_2 \cdot m_0 \cdot m_1 + a_1 \cdot m_0 + a_0|_M,\ 0 \le X \le M = \prod_{i=0}^{2} m_i \quad (9\text{-}1)$$

$$a_0 = r_0,\ a_1 = |(r_1 - |r_0|m_1)|1//m_0|m_1|m_1,\ a_2 = |\{(r_2 - |r_0|m_1)|1//m_0|m_2 - a_1\}|1//m_1|m_2|m_2 \quad (9\text{-}2)$$

순환부호를 사용한 승산기의 경우 모듈리를 소수만 사용하지만, DRNS를 사용하므로 상위 모듈러가 서로 소이어도 된다. 그러므로 모듈러 $m_0=32$, $m_1=31$, $m_2=5$로 선택하였을 때, 사용할 수 있는 수의 범위는 0~4959이므로 그래픽 도면의 위치점을 지정할 수 있다.

계수의 범위는 $0 \le a_0 \le 31$, $0 \le a_1 \le 30$, $0 \le a_2 \le 4$이다. r_2-a_0등의 연산은 우방향으로 데이타가 이동되는 배럴 쉬프터를 사용한 감산기를 사용하며, 그림 3(b)의 가산기와 동일한 구조를 갖는다. 또한, $|1//m_0|m_1 = |1//32|_{31} = |1|_{31}$, $|1//32|_5 = |3|_5$, $|1//31|_5 = |1|_5$ 이므로, $|3|_5$만 상수값을 갖는다. 그러므로 $|3|_5$을 승산하는 경우는 표 7과 동일한 방식을 사용하여, 모듈러 4 가산과 모듈러 5 승산의 정합관계를 사용하면 출력선의 재배열만으로 연산이 가능하다. 식 (9-1)을 식 (9-3)과 같이 표현하면 2^4의 배수가 되므로 계수의 연산결과를 4비트씩 구분이 가능하므로 고속의 변환을 할 수 있다.

$$X = (31a_2 + a_1)32 + a_0 = \{32a_2 + (a_1 - a_2)\}32 + a_0 = a_2 2^8 + (a_1 - a_2)2^4 + a_0 \quad (9\text{-}3)$$

2진수로 변환시 a_0는 하위 4비트가 되며, (a_1-a_2)는 중위 4비트이고, a_2는 상위 3비트이다. 단, $a_1-a_2<0$인 경우 a_2를 1감소한다.

3) 스케일링 연산기

Ⅳ장 1절에서 그래픽 도형 생성시에 입력이 소수인 경우 일정비율로 확대하여 정수로 입력하고, 최종 연산결과를 동일 비율로 축소하는 스케일링 연산기를 사용한다. 스케일링은 식 (10-1)과 같이 정수 X를 정수 Y로 제산하는 경우 몫을 구하는 것이다.[9] 잉여수가 r_0, r_1, r_2 이고, 모듈리 3개가 m_0, m_1,m_2인 경우 사용 가능한 수의 범위는 $(0 \sim m_0m_1m_2-1)$이므로, m_i로 스케일링하면 연산결과는 $0 \sim (\prod_{j=0}^{2} m_j - 1)(i \neq j)$ 이다. m_0로 스케일링한 결과는 식 (10-2)와 같이 두개의 모듈리로 표시되며, m_0m_1으로 스케일링하는 경우 식 (10-3)와 같이 표시된다. 식(9-2)에서 혼합기수변환의 계수 a_1, a_2연산과 동일한 연산과정을 수행하므로 혼합기수변환의 계수 생성부와 동일한 방식으로 스케일링 연산기를 설계할 수 있다.

$$X = [X/Y]Y + |X|_Y,\ [X/Y] = (X - |X|_Y) / Y\ (\text{'}[\ ]\text{'}: \text{정수부},\ \text{'-'}: \text{널 (null)}) \quad (10\text{-}1)$$

$$\{-,\ s_1 = |(r_1 - |r_0|m_1)|1//m_0|m_1|m_1,\ s_2 = |(r_2 - |r_0|m_2)|1//m_0|m_2|m_2\} \quad (10\text{-}2)$$

$$\{-,\ -,\ |[|s_2 - s_1|m_2]|1//m_1|m_2|m_2\} \quad (10\text{-}3)$$

Ⅴ. 결과 및 고찰

본 논문에서 설계한 고속 디스플레이를 위한 잉여수 승산기는 C 언어 및 논리회로 시뮬레이션을 사용하여 논리적인 검증을 수행하였다. 그림 4(b)의 순환부호2를 사용한 승산기의 논리적인 수행을 그림 6에 TTL소자를 사용하여 설계하였다. 그림 7은 모듈러가 7인 경우 $|z|_7 = |x|_7 \cdot |y|_7$연산을 수행하며, 부호정합은 표 7의 순환부호 2를 사용하여 입력선의 재배열로 수행되므로 논리소자가 소요되지 않는다. 배럴 쉬프터는 AND 어레이를 대신하여 삼상버퍼를 사용하였으며, 입력을 피승수로(x)하고 선택선을 승수 입력(y)으로 사용하였다. 그림 6의 순환부호2용 승산기의 입출력 신호를 그림 7에 나타내었다. $|3|_7 \cdot |4|_7 = |5|_7$ 의 연산인 경우, x_3와 y_4이 '0', 클럭신호가 '1'일 때 소자의 지연시간 후 z_5만 '0'으로 출력함을 알 수 있다.

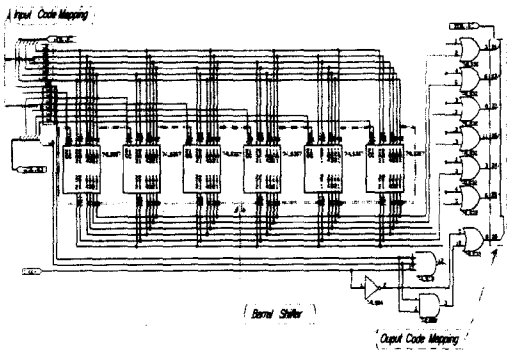


그림 6. TTL소자를 사용한 mod 7 순환부호2용 승산기의 논리회로

Fig. 6. Logic circuit of mod 7 multiplier for circulative code 2 using TTL.

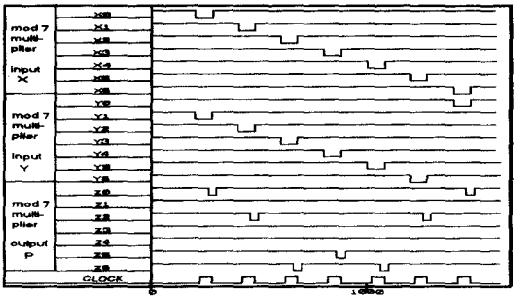


그림 7. TTL소자를 사용한 mod 7 순환부호2용 승산기(그림 6)의 입출력 신호

Fig. 7. Input-output signal of mod 7 multiplier for circulative code 2 using TTl.

모듈러 m(소수)의 연산시 ROM을 이용한 연산기, 순환부호1, 순환부호2를 이용한 연산기의 크기 및 연산속도의 비교를 표 9에 나타내었다. ROM 연산표를 이용한 잉여수계 연산기는 잉여수 연산기의 대표적인 설계방법이다.[7] $|5|_7=|3|_7\cdot|4|_7$ 연산의 경우, 4입력 NAND $2^3\times2^3=64$개 와 인버터 12개로 구성된 해독기, 192(=64×3) 크기 OR 어레이 및 버퍼 3개로 구성된다.

순환부호1을 사용한 승산기의 경우 $k=\lceil\log_2 m\rceil$비트로 입력 구분이 가능하므로, 입력 해독기에는 NAND, 출력 해독기는 AND로 구성 하였다. '0'의 검출은 k 비트 입력 NOR를 사용하며, 출력은 2입력 NOR를 k 개 사용한다. 순환부호2를 사용한 승산기는 그림 7과 같이 '0'의 검출은 2입력 AND를 사용하며, 출력은 NAND 및 OR를 사용하였다. 모듈러 m의 경우 순환부호2를 사용한 승산기의 크기가 가장 적음을 알 수 있다.

표 9. 잉여수 승산기의 연산시간 및 연산기 크기 비교

Table 9. The comparison of oeration time and size of oerators.
(I: input, i: Inverter, na: NAND, no: NOR, oa: OR array, aa: AND array, b: Buffer, o: OR, a: AND)

연산기 종류		연산기 크기 ('[]':mod 30 경우)	연산시간
RNS (modulusm)	ROM	2k I NAND=2^{2k}[1024], Inverter=2k [10], Buffer=k [5], OR array= 2^{2k}*k [5120] ($k=\lceil\log_2 m\rceil$ [5])	$t_{ROM}=t_i+t_{na}+t_{oa}+t_b$ ≒ 4 t_{ROMmax}
	배럴 쉬프터 (순환부호1)	kI NAND= 2(m-1) [60], 3I OR= 1 kI NOR= 2, Inverter=k+1 [6], AND array= $(m-1)^2$+k(m-1)+m-1 [1080], 2I AND= k [5], 2I NOR= k [5], Buffer=k [5] ($k=\lceil\log_2 m\rceil$ [5])	$t_{c1}=t_{na}+2t_{oa}+t_a+t_{oo}$ ≒ 5 t_{c1max}
	배럴 쉬프터 (순환부호2)	AND array= $(m-1)^2$ [900], 2I OR= m-1 [29], 2I AND= 1 2I NAND= 1, Inverter= 1	$t_{c2}=t_{aa}+t_o$ ≒ 2 t_{c2max}
DRNS (first modulus 31)	ROM (second moduli 3,5,7,11)	4I NAND=16, 6 I NAND=128, 8I NAND=256, Inverter=24, Buffer=12, OR array=1440	$t_{ROM}=t_i+t_{na}+t_{oa}+t_b$ ≒ 4 t_{ROMmax}
	배럴 쉬프터 (순환부호1 second moduli 3,5,7,11)	4I NAND=20, 3I NAND=20, 2I NAND=4, 4I NOR =2, 3I NOR = 4, 2I NOR=,14 Inverter= 14, 3I OR=4, 2I AND=12 AND array= 252, Buffer=12 2I AND=5, 2I NOR= 5	$t_{c1}=t_{na}+2t_{oa}+t_a+t_{oo}$ ≒ 5 t_{c1max}
	배럴 쉬프터 (순환부호2 second moduli 3,5,7,11)	AND array= 156, 2I OR= 22, 2I AND= 4 2I NAND= 4, Inverter= 4	$t_{c2}=t_{aa}+t_o$ ≒ 2 t_{c2max}
RNS (modulus31)	참고문헌[8] (순환부호1 준동형 사용 moduli 5,6 가산기사용)	9I NAND = 2, 8I NAND = 3, 7I NAND = 4, 6I NAND = 3, 5I NAND = 14, 4I NAND = 90, 3I NAND = 19, 2I NAND = 18	t_{c1tot} = 6 t_{na}

그래픽 도면의 해상도가 1024×1024이며, 그림 2와 같이 정수 × 실수 + 정수인 그래픽 연산시, 실수를 1000배 하여 연산하면 수범위는 0~(1023×1000+1023=)1024023이다. 이 때 모듈리는 2, 29, 31, 32, 33이며, 대표적으로 모듈러 31연산을 수행하는 경우를 표 9 상단부에 '[ ]'에 나타내었다. 첫번째 모듈리가 31일 때 DRNS를 사용하면 수범위는 0~(30×30+30=)930이므로 두번째 모듈리는 3, 5, 7, 11이다. DRNS를 사용할 경우의 연산기의 크기를 표 9의 하단부에 나타내었다. 단일 잉여수 처리를 하였을 때보다 DRNS를 사용한 경우 연산기의 크기가 감소한다. 참고문헌 [8]의 순환부호1을 사용한 승산기는 준동형관계에 따라 NAND를 사용하여 설계하였으며, mod 31 승산의 경우 mod 6과 mod 5가산으로 분리하여 수행한다. 그러나 입력 및 출력시 사용된 해독기가 연산부 보다 2배 이상 커지므로, 전체 연산기가 커지는 단점을 갖는다. 단일 잉여수에서 ROM을 사용한 승산

기와 DRNS에서 순환부호2용 승산기를 비교한 경우 DRNS사용하는 경우 승산기의 크기가 1/30배 이상 축소됨을 알 수 있다.

그래픽 도형 생성시 연산은 승산 및 가산을 사용하는데 배럴 쉬프터를 사용한 순환부호2용 가산기는 AND어레이만 사용하므로 승산기와 비슷한 크기로 설계된다. 참고문헌 [8]의 순환부호1을 사용한 가산기의 경우 mod 31가산시 입력은 5비트이다. 그러나 내부 연산에서 26비트의 순환부호 생성용 부가회로가 사용되므로, mod 31 가산시 10입력 NAND=140개, 9입력 NAND=760개, 8입력 NAND=270개, 5입력 NAND=25개가 소모되므로 승산기보다 6배이상 커지는 단점을 갖는다.

동일한 소자를 사용한 경우 연산기의 속도는 논리소자의 최대시간에 의해 결정된다. 논리소자 $l_i(i=0, \cdots,n)$를 사용한 P 승산기의 연산시간 t_P라 하는 경우 $t_{pmax}=\{ t_0,t_1,\ldots,t_n\}_{max}$, $t_P=k\cdot t_{pmax}$ (k:정수)로 표시하는 경우 표 9와 같이 표시된다. $t_{ROMmax} \approx t_{cp1Max} \approx t_{cp2Max}$라 할 때 순환부호2를 사용한 승산기가 가장 빠름을 알 수 있으며, TTL소자 74s09, 74s32를 사용한 경우 87MHz속도로 연산될 수 있다. 참고문헌 [8]의 순환부호1을 사용한 승산기는 승산기의 입출력 및 연산부가 2 단(level)의 NAND로 구성되므로 배럴 쉬프터를 사용한 순환부호2용 승산기보다 느림을 알 수 있다.

1024×1024 크기의 그팩도면인 경우 각 위치점은 정수이므로, 최장 선분의 길이는 1024이다. 본 논문에서 설계한 순환부호2를 사용한 승산기를 사용하는 경우 초당 84918개의 직선을 발생할 수 있다.

VI. 결 론

그래픽 응용분야에서 고해상도의 그래픽 도면에 다양한 도형 생성이 요구되므로, 고속의 그래픽 처리가 필요하다. 그래픽의 기본적인 도형 생성시 고속의 승산 및 가산이 필수적이며, 특히 도형발생을 고속화하기 위하여 고속의 승산기가 요구된다.

잉여수 연산은 정수연산을 수행하며, 수를 각각의 모듈리로 분리하여 연산하므로 모듈리간에 캐리정보가 필요하지 않다. 그러므로 연산기 크기가 감소하며, 고속의 연산기 설계가 가능하다. 순환군은 mod p(p:소수)승산이 mod (p-1)의 가산과 동형을 형성하므로 부호정합에 의하여 승산기 설계가 가능하다. 그러므로 본 논문에서는 두번의 잉여수 변환을 수행하는 DRNS를 제안하여 잉여수 연산기의 크기를 감소시켰으며, 순환부호를 이용한 잉여수 연산시 소수만을 모듈리로 사용함으로써 수범위가 확장되는 단점을 감소시켰다. 표 7의 순환부호2를 사용하여 부호정합을 입출력선의 재배열만으로 수행함으로써 연산기의 크기를 감소 시켰으며, 연산속도를 향상시킬 수 있었다.

잉여수 연산기의 대표적 설계방법인 ROM연산표를 이용한 승산기와 본 논문에서 설계한 승산기와 비교할 때, DRNS에서 배럴 쉬프터를 사용한 순환부호2용 승산기가 1/30이상 연산기 크기가 축소하며, 연산속도가 2배이상 향상됨을 알 수 있었다. TTL소자 74s09, 74s32를 사용한 경우 87MHz속도로 연산이 가능하므로 실시간 그래픽 처리를 위항 연산기로 사용할 수 있다. 앞으로의 연구결과를 실용화하기 위하여, 본 연구에서 설계한 연산기를 이용한 그래픽 프로세서에 관한 연구가 계속되어야할 것이다.

참 고 문 헌

[1] Foley, Van Dam, Feiner, and Hugles, *Computer Graphic Princilpe and Practice*, Addison-Wesley, Second Edition, 1990.

[2] 金龍成, 趙源敬, "행렬·벡터 연산용 1-차원 시스톨릭 어레이 프로세서를 이용한 그래픽 가속기의 설계", 電子工學會論文誌 ,第 30卷, B編, 第 1號, pp. 1-9, 1月, 1993

[3] 金龍成, 趙源敬, "1차원 시스톨릭 어레이 프로세서를 이용한 고속 곡선 발생기에 관한 연구", 電子工學會論文誌 ,第 31卷, B編, 第 5號, pp. 533-543, 5月, 1994

[4] Chi-Hsu Wang, Hore-Yuan, and Rong-Sheng Wen, "Pipelined Computations of B-spline Curve", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol 22, No. 2, pp. 327-331, March/April, 1992.

[5] K.D.Weinmann, M.A.Soderstrand and S.Shebani, "Evaluation of New Hardware for a High-speed, Digital, Adaptive Filter Using the Residue Number

System", *16th Asilomar Conference on Circuits, Systems, and Computers*, pp. 187-191, 1982.

[6] 尹賢植, 趙源敬, "잉여수계를 이용한 디지탈 신경회로의 실현", 電子工學會論文誌 ,第 30卷, B編, 第 2號, pp. 44-50, 2月, 1993

[7] M.A. Soderstrand, C. Vernia and Jui_Hua Chang, "An Improved Residue Number System Digital to Analog Converter", *IEEE Trans. on Circuits and Systems*, Vol CAS-30, pp. 903-909, December, 1983.

[8] Stepphen S. Yau, and Jackson Chung, "On the Design of Modulo Arithmetic Units Based on Cyclic Group", *IEEE Trans. on Computers*, Vol C25, No. 11, pp. 1057-1067, November, 1976.

[9] Nicholas S. Szab, Richard I. Tanakas, *Residue Arithmetic and its Applications to Computer Technology*, McGraw-Hill, 1967.

[10] 朴在傑, 朱鎭求, 金京壽, 現代 代數學, 喜重堂, 3版,1990

[11] Neil H.E.Weste, Karman Eshraghian, *Principle of CMOS VLSI Design*, Addison-Wesley, Second Edition, 1988.

저 자 소 개

金 龍 成(正會員)

1959년 9월 10일생. 1983년 경희대학교 공과대학 전자공학과 졸업(공학학사). 1985년 경희대학교 대학원 전자공학과 졸업(공학석사). 1995년 경희대학교 대학원 전자공학과 졸업(공학박사). 1995년 현재 여주전문대학 전자계산과 전임강사. 주관심분야는 고속 그래픽처리를 위한 VLSI설계, 고속연산기 설계 및 고속 디지탈 신호처리용 VLSI설계, CAD등임

趙 源 敬(正會員) 第 28卷 B編 第 10號 參照
현재 경희대학교 전자공학과 교수