정보학연구 제7권 제2호 2004년 6월(pp.19-28)
The Journal of Information Technology. Vol.7. No.2



# 소(少) 제수용 잉여수계 제산 연산기 설계에 관한 연구

김용성*

요 약

잉여수계를 사용한 연산기는 디지털 신호처리, 컴퓨터 그래픽 등에 있어서 여러가지 장점 을 갖으므로 전용 프로세서 설계에 사용되고 있다. 그러나 크기 비교와 일반적인 제산에 있어서 단점을 갖는다. 본 논문에서 제안된 연산기는 곱의 역을 사용한 제산의 결과가 나머지를 갖는다면, 현재 제수에 의해 산출된 몫의 최대 값보다 큰 값이 발생되는 조건을 반복연산의 종결조건으로 사용하였으며, 몫의 비교를 대응된 제수 값으로 대신 하였다. 그러므로, 설계된 연산기는 작은 크기의 제한된 제수를 사용하는 제한점은 갖지만, 컴퓨터 그래픽의 스케일링 등에 적용하는 경우 연산기의 크기 및 속도가 우수한 제산 연산기로 사용할 수 있다.

## Ⅰ. 서론

잉여수계(Residue Number System)는 모듈리(moduli) 간의 캐리 전달이 없고, 병렬처리의 특성을 가지며[1], 연산표(LUT: Look Up table)에 의한 연산을 수행하여 복합연산이 가능하다는 장점을 가지므로 디지털 신호처리, 신경망, 컴퓨터 그래픽 등과 같은 분야의 전용 프로세서 설계에 유용하게 사용된다[2][3][4][5]. 그러나 대소 비교, 일반적인 제산 등과 같은 경우, 혼합기수 변환(MRC: Mixed Radix Number System)등을 사용하여 가중치 수 체계로 변환을 하므로 연산처리의 지연이 수반되는 단점을 갖는다.

잉여수계에서의 제산은 곱의 역(Multiplicative Inverse), 스케일링(Scaling), 일반적인 제산(General Division)과 같은 3가지 방법으로 나누어진다[1]. 곱의 역에 의한 제산은 단순하지만 몫 이외에 나머지가 없어야 한다는 제한점을 갖으며, 이를 보완하는 스케일링은 제수를 모듈리의 곱으로만 사용하여야만 한다. 그러므로 제한된 제산 만이 가능하며, 컴퓨터 그래픽의 스케일링시 축소화 과정과 같은 경우에 적용하는 경우 만족된 축소를 하기 위해서는 승산과 겸용된 여러 번의 연산과정을 사용해야 된다는 단점을 갖는다. 또한, 일반적인 제산은 근사화된 제수를 사용하며, 연산결과에 따른 피제수와의 오차를 구하여 이를 보정하는 방법을 주로 사용한다. 근사화된 제수를 2의 승수의 최대치를 사용하는 방법은 초기의 반복적인 산출법과[1] 이를 개선하여 연산표를 사용하여 연산속도를 개선한 방법이 있으나[6], 종결 조건에 따른 부가회로가 추가되면 연산속도가 증가되는 단점을 갖는다[7]. 스케일링 연산을 이용한 일반 제산은 근사화된 제수를 혼합기수 변환에 의해 모듈리의 곱으로 선정하고, 스케일링과 기수확장을 수행하므로 연산속도가 저하되는 단점을 갖는다.

그러므로 본 논문에서는 잉여수계 제산 시 곱의 역의 제한점을 보정하고, 일반적인 제산법의 종결 조건 문제점을 개선하여, 컴퓨터 그래픽에

* 여주대학 컴퓨터 인터넷과 부교수

서 잉여수계를 사용한 프로세서 설계 시, 스케일링 과정에 필요한 작은 수의 제수 연산에 적합한 잉여수계 제산기를 설계하고자 한다.

## II. 잉여수 계 스케일링 및 그래픽의 적용

잉여수계에 사용되는 모듈리를 P라하고, 모듈러스(modulus)를 $m_i$라 할 때, 정수 X의 잉여수 표현은 $|X|_{m_i} = r_i$ 이다. $X$와 $Y$ 두개의 정수에 의해 표현되는 이항연산(Binary Operation)의 결과 $Z$는 식(1)과 같이 표현된다.[1]

$P=\{m_1, m_2, \ldots, m_n\}$( i= 1,..., n , n: 정수)

$Z=X \circ Y$ ( ' $\circ$ ' : 이항 연산자)

$$|Z|_{m_i} = | |X|_{m_i} \circ |Y|_{m_i}|_{m_i} = | r_1 \circ r_2 |_{m_i} \quad (1)$$

$$Z=(|Z|_{m_1}, |Z|_{m_2}, \ldots, |Z|_{m_n})$$

두 잉여수의 곱이 $| |d|_{m_i} \cdot |Y|_{m_i}| |_{m_i} = 1$ 이고, Y와 $m_i$가 서로 소인 경우, 곱의 역 (Multiplicative Inverse)은 식(2)와 같이 표현된다.

$$|1//Y|_{m_i} = |d|_{m_i} \quad (2)$$

피제수가 Y이고, 제수가 X 일 때, Z=X/Y의 모듈러스 $m_i$의 잉여수 연산결과 Z는 다음 식(3)과 같이 표시된다.

$$|Z|_{m_i} = | |X|_{m_i} \cdot |1//Y|_{m_i}|_{m_i} \quad (3)$$

그러나 식(3)의 연산결과가 Z값과 일치하려면 다음과 식에서 나머지가 $|X|_Y$=0인 경우이어야 한다.[1]

$$X=\left[\frac{X}{Y}\right]Y+|X|_Y, \quad X=[Z]Y+|X|_Y$$

(단, "[ ]"는 몫의 정수부) (4)

잉여수 스케일링은 제수를 모듈리로 제한하여 이러한 문제점을 보완하였으며, 모듈리가 $m_1, \ldots, m_4$일때, 정수 X의 잉여수 $r_1, \ldots, r_4$를 모듈리 $m_1$로 스케일링하는 경우,

$X=\left[\frac{X}{m_i}\right]m_i+|X|_{m_i}$ 이고,

$S_i = | (|X|_{m_i} - |X|_{m_1}) \cdot 1//m_1|_{m_1}$ 이므로

$S_2 = | (r_2 - r_1) \cdot 1//m_1|_{m_1}$, ...,

$S_4 = | (r_4 - r_1) \cdot 1//m_1|_{m_4}$ 되며,

$S_1$은 $||X|_{m_1} - |X|_{m_1}|_{m_1}$=0이고 $|1//m_1|_{m_1}$은 성립되지 않으므로, 산출된 $S_2, S_3, S_4$에의해 기수확장(Extension of Base)을 사용하여 구한다. 기수확장은 혼합기수 변환(Mixed Radix Conversion)을 사용한다. 모듈리가 $m_1, .., m_4$이고, 잉여수가 $r_1,.., r_4$ 인 경우 $m_1$에 의한 기수확장을 식 (4)에 나타내었다.

$$S = a_4 m_2 m_3 m_4 + a_3 m_2 m_3 + a_2 m_2 + a_1$$

$$S = (a_4 m_4 m_3 + a_3 m_3 + a_2) m_2 + S_2 ,$$

$$a_1 = |S|_{m_2} = S_2$$

$S^1_{j2} = |(S_j - a_1)//m_2|_{m_j}$ (j=1,3,4),

$$a_2 = |S^1_{32}|_{m_3}$$

$S^2_{j3} = |(S^1_{j2} - a_2)//m_3|_{m_j}$ (j=1,4),

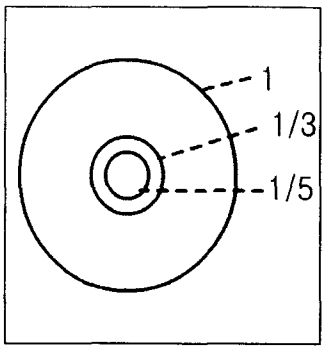
$$a_3 = |S^2_{43}|_{m_4}$$

$S^3_{14} = |(S^2_{j3} - a_3)/m_4|_{m_1}$ ,

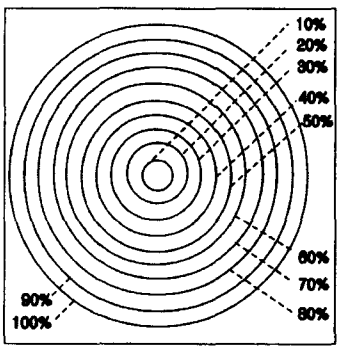
$$a_4 = |S^3_{14}|_{m_1} \quad (4)$$

$S_2, S_3, S_4$를 사용하여 혼합기수 변환 수행

시 계수 $a_4$=0가 되는 $S_1$을 구하며, N개의 모듈리에 대한 스케일링 전 수행 과정에 N단계가 소요된다.




(a)




(b)

(그림 1) 원 도형의 스케일링
(Fig. 1) Scailing of Circle Generation
(a) 잉여수 스케일링을 사용한 경우(1/3, 1/5 축소)
(b) 일반적인 제산을 사용한 경우(10%~100%)
(a) In case of using RNS Scailing
(b) In case of General Division(10%~100%)

잉여수계 스케일링을 컴퓨터 그래픽에 적용하는 경우, 원 도형의 그래픽 스케일링에 대한 결과를 그림 1에 표시하였다. 그림 1.(b)에서 일반적인 제산에 의한 스케일링은 그래픽 도형의 요구된 축소를 수행하기에 용이하다. 그러나, 모듈리 3, 5에 의한 잉여수계 스케일링으로 축소를 수행하는 경우 그림 1(a)와 같이 축소율이 높은 형태가 되며, 80%의 축소를 하려면 4배 확대 후에 1/5축소를 수행하여야 하는 이중 연산이 요구되는 단점을 갖는다. 잉여수의 일반적인 제산을 사용하는 경우 연산단계가 복잡하여 잉여수 스케일링에 의한 방법보다 처리속도가 늦어질 수 있다. 그러므로 다양한 스케일링을 수행하면서도 연산단계가 개선된 잉여수 제산기가 요구된다.

# Ⅲ. 근사화된 제수를 이용한 일반적인 잉여수계 제산기

## 3.1. 스케일링을 이용한 일반적인 잉여수계 제산기

스케일링을 사용한 일반적인 제산은 근사화된 제수를 사용하여 연산을 수행한다. 근사화된 제수는 제수 보다 크며, 스케일링을 이용하므로 모듈리의 곱으로 되어야 한다. 그러므로 혼합기수 변환을 사용하여 혼합기수 계의 계수를 구한 후, 최대 기수의 계수를 기수표현에 사용되지 않은 모듈리로 선택한다. 예를 들어 모듈리가 2, 3, 5, 7, 11, 13, 15 인 경우, 혼합기수로 표현된 최대기수가 11 ×13 ×15 이고 계수 $a_4$가 3이면 3대신 5를 선택하고, 근사화된 제수는 5 ×11 ×13 ×15를 선택한다. 연산기 설계 시에는 미리 저장된 LUT를 사용하여 근사화된 제수를 선택하여 사용한다. 스케일링을 사용한 일반적인 제산의 구체적인 과정은 다음과 같이 6단계를 수행하여 얻을 수 있다.[1]

단계 1. X는 피제수, Y는 제수이고 양의 정수

일 때, Y ≠0, 근사화된 제수 $\hat{Y}$는 다음과 같이 설정한다.

$$Y \le \hat{Y} < 2Y,\ \hat{Y} = Q\prod_{i=1}^{p-1} m_i \text{ (p : 정수)}$$

Y의 MRC 후, 최대 계수 $a_{max} = a_{p-1}$라 할 때, $Q \le \prod_{i=p}^{p+k} m_i$ (k=: 정수) ( $\prod_{i=1}^{p-1} m_i$에 사용되지 않은 모듈리의 곱).

단계 2. $X_0$=X, i=1, Z=0

단계 3. 근사화된 제수$\hat{Y}$에 의한 스케일링 및 기수확장 수행 $Z_i$산출

단계 4. i=i+1, $|X_i|_{m_i} = |X_{i-1} - \hat{Y}Z_i|_{m_i}$ (i=1,..n , n:정수)

단계 5. $Z = Z + Z_i$

단계 6. $Z_i$=0 또는 $X_i$=0때까지 단계 3을 반복.

( $r$회 반복시, if $X_r$=0 and $Z_r \neq 0$ then $Z_r = Z_i$

if $Z_r$=0 and $X_{r-1} > Y$, 단 $\hat{Y} \neq$Y then $Z_r$=1

else $Z_r$=0)

이 방법은 근사화된 계수를 산출하기 위하여 혼합기수 변환을 수행하고, 이를 사용한 반복된 스케일링 연산, 생성된 몫의 누적 및 종결 조건에서 비교 등이 수행되어야하므로, 연산속도가 저하되는 문제점을 갖는다.

## 3.2. 제수와 피제수의 최상위 비트차를 이용한 일반적인 잉여수 제산기

제수와 피제수의 2진 표현에 대한 최상위 비트차를 이용한 잉여수 제산기는 초기의 최대치를 구하는 함수를 사용하는 방법에서 LUT를 사용하여 연산과정을 단축시킨 개선된 방법을 주로 사용한다. X는 피제수, Y는 제수이고 양의 정수일 때 제산 과정은 다음과 같다[6][7].

단계 1. Q= $\lfloor X/Y \rfloor$ , Y ≠0 이고, 초기치 Q=0이다. 모듈리는 n개로 한다.

단계 2. 제수 Y에 있어서 최상위 비트의 위치를 $k$라한다. $k = h(|Y|_{m_1}, \ldots, |Y|_{m_n})$

단, $h(I)$= I+ $\lfloor \log_2 I \rfloor$ ( I가 양의 정수인 경우)

$h(I)$= 0 ( I=0인 경우)

$h(I)$= $\lfloor \log_2 I \rfloor$ ( $0 < I < 1$인 경우)

단계 3. 피제수 X에 있어서 최상위 비트의 위치 j라한다. $j = h(|X|_{m_1}, \ldots, |X|_{m_n})$

단계 4. If $j > k$ then (i=1,..,n)

$|Q'|_{m_i} = |\,|Q|_{m_i} + |2^{j-k-1}|_{m_i}\,|_{m_i}$,

$|X'|_{m_i} = |\,|X|_{m_i} - |2^{j-k-1}|_{m_i} * |Y\,|_{m_i}\,|_{m_i}$

$|Q\,|_{m_i} = |Q'|_{m_i}$, $|X\,|_{m_i} = |X'|_{m_i}$, 단계 3의 이동

단계 5. If j=k then (i=1,..,n)

$|X'|_{m_i} = |\,|X|_{m_i} - |Y\,|_{m_i}\,|_{m_i}$ ,

$j' = h(X')$ ,

$j' < j$이면 $|Q|_{m_i} = |Q+1|_{m_i}$

단계 2, 3의 과정은 LUT를 각각 1개씩 사용하여 그 결과를 얻을 수 있고, 단계4에서 각 모듈리 별로 LUT를 사용하여 $|2^{j-k-1}|_{m_i}$를 구한 다음, 누적연산과 승산 및 감산을 각각의 LUT 연산기에서 수행한다. j=k를 검출하기 위한 비교부는 참고문헌[7]의 구성도에 생략된 부분으로 해당 조건에 따른 감산의 수행 및 j'과 j의 비교부가 필요하며, 연산결과 Q의 추가 연산이 필요하므로, 설계된 구성도의 예상된 것보다 연산기

의 크기가 증대되고, 연산속도가 저하된다.

## Ⅳ. 소(少) 제수에 적합한 잉여수계 일반 제산기의 설계

본 논문에서는 기존의 일반적인 잉여수 제산기의 종결 조건에 의하여 연산기 크기가 증가 및 잉여수 스케일링을 이용한 제산기의 연산속도에 대한 문제점을 갖지 않도록 하기 위하여, 소수 제수에 적합한 잉영수계 제산기를 설계하고자한다. 곱의 역을 사용한 제산은 연산결과 나머지가 발생하지 않는 경우에 적합하다는 제한 조건을 갖는다. 피젯수를 정수 X, 제수를 정수 Y($Y \neq 0$), 피제수 Y의 정수 Q배가되는 수를 Xm으로 하는 경우, Xm+1,..., Xm+k ($1 \leq k < Y$, $k$:정수) 사이의 정수를 Y의 곱의 역으로 제산을 수행하면 다음 식(5-2)와 같이 표현된다.

$$Xm/y = Q+0,\ |\ |Xm|_{m_i} \cdot |1//Y|_{m_i}\ |_{m_i} = |Q|_{m_i}\ (i=1,...n) \quad (5\text{-}1)$$

$$|Xm+1|_{m_i} \cdot |1//Y|_{m_i} = |\ |Xm|_{m_i} \cdot |1//Y|_{m_i} + |1//Y|_{m_i}\ |_{m_i} = |\ |Q|_{m_i} + |1//Y|_{m_i}\ |_{m_i}$$

$$|Xm+2|_{m_i} \cdot |1//Y|_{m_i} = |\ |Xm+1|_{m_i} \cdot |1//Y|_{m_i} + |1//Y|_{m_i} = |\ |Q|_{m_i} + 2 \cdot |1//Y|_{m_i}\ |_{m_i}, \ldots,$$

$$|Xm+k|_{m_i} \cdot |1//Y|_{m_i} = |\ |Xm+k-1|_{m_i} \cdot |1//Y|_{m_i} + |1//Y|_{m_i}|_{m_i} = |\ |Q|_{m_i} + k \cdot |1//Y|_{m_i}\ |_{m_i}\ (i=1,...,n) \quad (5\text{-}2)$$

식(5-2)에서 Xm+k(1< $k$<Y, k: 정수)를 Y의 곱의 역으로 제산하는 경우, 몫이 $|Q|_{m_i}$보다 $|k \cdot |1//Y|_{m_i}|_{m_i}$ 만큼 증가하게 된다. 잉여수 연산은 정수 연산이므로, $1 \leq k < Y$ 조건에서 몫이 Q와 다른 값으로 산출되지 않아야 된다. 그러므로, Xm+k를 k만큼 감소시켜 곱의 역을 사용하여 제산의 결과를 산출한다. 모듈리가 $m_1$, $m_2$인 경우, 잉여수의 표현 범위는 0~ $m_1m_2$ -1이다. 제수 Y에 의해 수행된 제산의 나머지 없는 몫의 최대 값을 $Q_{max}$라하고, 피제수를 $m_1m_2$-p라 하는 경우( $0 \leq p < y$, $p$: 정수) $Q_{max}$는 다음과 같이 표현된다.

$$|Q_{max}|_{m_1} = |(m_1m_2 - p)//Y|_{m_1},\ |Q_{max}|_{m_2} = |(m_1m_2 - p)//Y|_{m_2}$$

위 식의 몫을 혼합기수변환 하면 다음 식과 같이 표현되며,

$$Q_{max} = |\ |(m_1m_2 - p)//Y|_{m_2} - |(m_1m_2 - p)//Y|_{m_1}|_{m_2} \cdot |1//m_1|_{m_2} + |(m_1m_2 - p)//Y|_{m_1}$$

$|(m_1m_2)|_{m_2}$=0 이므로 식(6)과 같이 표현된다.

$$Q_{max} = |\ |-p//Y|_{m_2} - |-p//Y|_{m_1}|_{m_2} \cdot |1//m_1|_{m_2} + |-p//Y|_{m_1} \quad (6)$$

식(5-2)에서 모듈리가 $m_1, m_2$이고 초기치 Xm=0인 경우 식(6)과 동일한 조건을 적용하여 혼합기수변환을 수행하면 식(7)과 같이 표현할 수 있다.

$$r_1 = |k//Y|_{m_1},\ r_2 = |k//Y|_{m_2}$$

$$|\ |k//Y|_{m_2} - |k//Y|_{m_1}|_{m_2} \cdot |1//m_1|_{m_2} + |k//Y|_{m_1}$$

$$= | \,|([\,(k+p)-p]//Y\,|_{m_2}-$$
$$|[\,(k+p)-p]//Y\,|_{m_1}|_{m_2}$$
$$\cdot\,|1//m_1|_{m_2}+|[\,(k+p)-p]//Y\,|_{m_1}$$
$$= | \,|((k+p)//Y\,|_{m_2}-$$
$$|[\,(k+p)//Y\,|_{m_1}|_{m_2}\cdot|1//m_1|_{m_2}+$$
$$|(k+p)//Y\,|_{m_1}$$
$$+|\,|-p//Y\,|_{m_2}-|-p//Y\,|_{m_1}|_{m_2}$$
$$\cdot\,|1//m_1|_{m_2}+|-p//Y\,|_{m_1}$$
$$= | \,|((k+p)//Y\,|_{m_2}-$$
$$|[\,(k+p)//Y\,|_{m_1}|_{m_2}\cdot|1//m_1|_{m_2}+$$
$$|(k+p)//Y\,|_{m_1}+Q_{\max} \quad (7)$$

식(7)의 결과에서 식(5-2)의 $|k\cdot|1//Y\,|_{m_i}|_{m_i}$ (i=1,...,n)의 가중치 표현 결과는 $1 \le k$<Y에서 $Q_{\max}$보다 큰 값임을 알 수 있으므로, 정수 X에 대한 곱의 역을 사용한 제산 시 몫 Q를 산출하는 반복연산에 대한 종결조건으로 사용한다. 소수(少數)를 제수로 사용한 잉여수계 일반 제산기의 기본 연산과정은 다음의 단계로 수행한다.

단계 1. 모듈러 $m_1,...,m_n$( n: 정수), $m_i$는 소수(Prime)이고, 피제 X, 제수 Y (X,Y :정수), Y< $m_i$ , 1<X ≤M-1 (M= $\prod_{i=1}^{n} m_i$)로 선정한다.

단계 2. $|Z\,|_{m_i}=|X//Y\,|_{m_i}$ ( $i$=1,..., $n$)

단계 3. $|Z\,|_{m_i}$ ( $i$=1,..., $n$) $\xrightarrow{\text{가중치변환}}$ $Z$

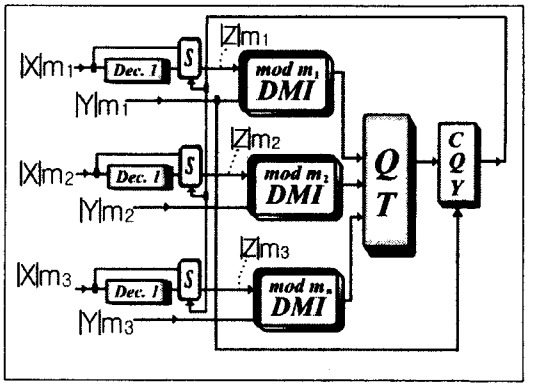
단계 4. if $Z \ge Q_{\max}$ then

$|X\,|_{m_i}=|\,|X\,|_m-1|_{m_i}$( $i$=1,..., $n$),

단계2로 이동

else (Z < $Q_{\max}$) 연산 종결

제안된 잉여수 제산 연산기의 설계를 그림 2에 표시하였다. 각 모듈리에 대한 DMI(Division using Multiplicative Inverse)부는 단계 2의 연산을 수행하는 LUT로 구성하고, 몫의 가중치 수체계 변환은 QT(Quotient)에서 수행한다. 수행 단계 4.에서 조건 비교를 위하여, 제수 Y에 따른 $Q_{\max}$와 연산결과의 비교는 CQY(Comparison of Quotient and Y) LUT 연산기에서 수행하고, 판정결과는 S 선택기에 전달되며, 감산의 수행은 각각 모듈리의 Dec.1(Decrement 1) LUT에서 수행한다.




(그림 2) 제안된 잉여수계 일반 제산 연산기
(Fig. 2) The proposed General Division Operator in RNS

제수가 $Y_1,....,Y_j$ (j<M-1, j: 정수 , $Y_{j-1}$< $Y_j$)라고 하고, 각각의 대응된 몫은 $Q^1_{\max},...,$ $Q^j_{\max}$이라 하면, $Q^{j-1}_{\max} > Q^j_{\max}$이 되고 $Q^1_{\max}$이 최대 크기의 몫이 된다. 제안된 연산기에 사용되는 제수는 단계 1과 같이 Y< $m_i$인 작은 수이므로, QT는 $|Z\,|_{m_i}$ ( $i$=1,..., $n$)의 가중치로

변환된 값 대신에 $Q^1_{max},..., Q^j_{max}$과 비교하여 $Q^j_{max}$보다 큰 경우는 대응된 $Y_j$값을 저장하여야 사용한다. QT의 결과 $Y_j$를 CQY LUT의 비교대상 입력으로 사용하고 현재 Y값과 비교를 수행할 수 있으므로 작은 크기의 LUT로 CQY와 QT를 구성할 수 있다. 그러므로 위의 연산 수행 단계 3과 단계 4는 다음과 같이 수정하여 수행한다.

단계 3 수정. $|Z|_{m_i}$ ( $i$=1,..., $n$)

$\xrightarrow{\text{비교값변환}}$ if $Q^j_{max} \le Z < Q^{j-1}_{max}$

then $Y_j$

단계 4 수정. if $Y_j$ of CQY $\ge$ Y then

$|X|_{m_i} = |\,|X|_{m_i} - 1|_{m_i}$( $i$=1,..., $n$),

단계2로 이동

else ( $Y_j$ of CQY < Y ) 연산 종결

# V. 실험 및 고찰

본 논문에서 제안된 소수(少數) 제수에 적합한 잉여수계 일반 제산 연산기에 대한 논리적 검증은 HLL과 VHDL을 사용하여 수행하였다. 모듈리가 11, 13, 17이면 정수 X의 범위가 $0 \le X \le 2430$가 되고, Y는 $Y < m_i$이므로 $1 \le Y \le 10$로 한다. 이 경우 표 1. (a)에 제수 Y에 의한 제산 결과 몫의 최대값 $Q_{max}$를 표시하였고, 표 1. (b)에 연산 결과 $|Z|_{m_i}$(i=1~3)가 {2,3,1}~{2,3,9}인 경우, 가중치 수체계(10진수) 변환 값Z와 Ⅳ절의 단계3 수정 시 조건 $Q^j_{max} \le Z < Q^{j-1}_{max}$에 의한 QT LUT의 저장 값을 표시하였다( "11"은 $Q_{max}$ 최소치보다 적은 수).

〈표 1〉 $Q_{max}$에 의한 QT LUT의 구성 예.

〈Table 1〉 Ex. of QT LUT composision according to $Q_{max}$

(a) 제수 Y에 대한 $Q_{max}$, (b) $Q_{max}$에 의한 QT의 구성

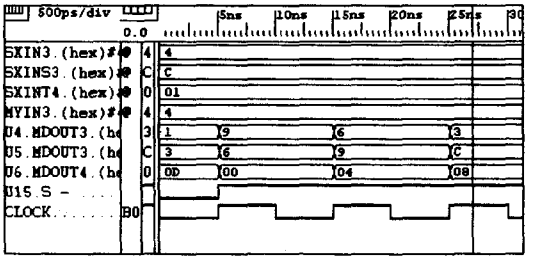
(a) $Q_{max}$ of divisior Y, (b) QT composision according to $Q_{max}$

(a)

| Y | Qmax |
|---|---|
| 2 | 1215 |
| 3 | 810 |
| 4 | 608 |
| 5 | 486 |
| 6 | 405 |
| 7 | 348 |
| 8 | 304 |
| 9 | 270 |
| 10 | 243 |

(b)

| $\|Z\|_1$ | $\|Z\|_2$ | $\|Z\|_3$ | $Z$ | QT |
|---|---|---|---|---|
| 2 | 3 | 1 | 783 | 4 |
| 2 | 3 | 2 | 1498 | 2 |
| 2 | 3 | 3 | 2213 | 2 |
| 2 | 3 | 4 | 497 | 5 |
| 2 | 3 | 5 | 1212 | 3 |
| 2 | 3 | 6 | 1927 | 2 |
| 2 | 3 | 7 | 211 | 11 |
| 2 | 3 | 8 | 926 | 3 |
| 2 | 3 | 9 | 1641 | 2 |




(그림 3) 제안된 잉여수계 일반 제산 연산기의 입출력 신호 (모듈리 :11, 13, 17)

(Fig. 3) Input-output signal of the proposed General Division Operator in RNS (moduli : 11, 13, 17)

모듈리가 11, 13, 17이고, 피제수가 108, 제수가 4인 경우에 대한 제안된 잉여수계 제산 연산기의 입출력 신호(16진수)를 그림 3에 나타내었다. 피제수는 각 모듈리 순서에 따라 SXIN3, SXINS3, SXINT4로 표시되었고, 제수는 MYIN3

로 표시되었으며, 결과는 U4.MDOUT3, U5M-DOUT3, U6MDOUT4으로 표시하였다. 초기 연산 결과는 3번의 클럭 신호 후 결과 $3_{HEX}$, $C_{HEX}$, $8_{HEX}(25_{10})$가 출력됨(수직선부분)을 알 수 있다.

모듈리 수가 n개이고 스케일링을 이용한 일반적인 잉여수계 제산기의 경우, 혼합기수 변환 과정에서 (n-1) $t_{LUT}$의 시간이 소요되고, 스케일링에서 r번의 반복 연산수행 시 nr $t_{LUT}$의 시간이 소요되며, 종결 조건에서 (2r+1) $t_{LUT}$의 시간이 소요된다. 또한, 몇 개의 모듈리를 사용하여 스케일링을 수행하는가에 따라 제산 연산기의 구조가 변형되어야 한다는 문제점을 갖는다. 제수와 피제수의 최상위 비트 차를 이용한 일반적인 잉여수 제산기는 r번의 반복 연산수행 시 4r $t_{LUT}$의 시간이 소요되며, 종결 조건에서 5개의 연산부 중, 2개 연산부는 전단과 동시에 수행될 수 있으므로, 3 $t_{LUT}$의 시간이 소용되어, (4r +3) $t_{LUT}$ 의 시간이 소요된다. 제안된 잉여수 일반 제산 연산기의 경우 4r $t_{LUT}$의 시간만 소요된다. 모듈리가 11, 13, 17인 경우 연산시간 및 연산기 크기의 비교를 표2에 나타내었다. 소요된 연산기의 크기는 그림 1의 제안된 연산기의 크기가 가장 작으며, 연산시간은 3.1절의 최상위 비트 차를 이용한 연기와 본 논문의 연산기가 같은 반복 회수 r인 경우, 유사함을 알 수 있다. 본 논문에서 제안된 제산 연산기는 연산기 크기와 속도에 있어서 우수하지만 제수가 모듈리 보다 적어야 한다는 제한점을 갖는다.

## Ⅵ. 결론

잉여수 계는 연산기 설계 시 모듈리 간의 연산에 자리올림수가 필요 없고, LUT를 사용하여 가산과 승산을 동일한 속도로 수행할 수 있으며, 병렬구조로 설계가 가능하므로, 신경망 처리, 디지털 신호처리와 같은 특수목적의 전용 프로세서 설계 시 이점을 갖는다. 그러나 일반적인 제산에 있어서 처리속도가 저하되고, 연산기 크

〈표 2〉 잉여수 일반 제산기의 연산시간 및 연산표 크기 비교(모듈리 :11, 13, 17)
〈Table 1〉 The comparison of operation time and LUT size of General Division operator 11, 13, 17)

| 연산기 종류 | 연산 시간 ( $t_{LUT}$) | 연산표 전체크기(bits) | |
|---|---|---|---|
| | | 모듈리 수=n | moduli 17, 13, 11 |
| Ⅲ-1절 | (n+2)r+n | $\sum_{i=1}^{s}\sum_{j=1}^{n-i} m_j^2 \log_2 m_j + \sum_{i=1}^{n-s}\sum_{j=1}^{i} m_j^2 \log_2 m_j + \sum_{i=1}^{n-1}\sum_{j=1}^{i} m_j^2 \log_2 m_j + \prod_{i=1}^{n} m_i + 4\sum_{i=1}^{n} m_i^2 \log_2 m_i$ | (s=2) 18260 |
| Ⅲ-2절 | 4r+3 | $3\log_2[\log_2(M-1)]\prod_{i=1}^{n} m_i + [\log_2(\log_2(M-1))]^2 \times (1+\sum_{i=1}^{n}\log_2 m_i) + 4\sum_{i=1}^{n} m_i^2 \log_2 m_i$ | 39816 |
| 그림1 | 4r | $\log_2 m_{i_{max}} \prod_{i=1}^{n} m_i + (\log_2 m_{i_{max}})^2 + \sum_{i=1}^{n} m_i m_{i_{max}} \log_2 m_i + \sum_{i=1}^{n}(m_i+1)\log_2 m_i$ | 12000 |
| s: 스케일링에 사용된 모듈리 수, r :반복 연산 회수 | | | |

기가 증대하는 문제점을 갖는다.

그러므로 본 논문에서는 이러한 문제점을 해결하기 위해서, 나머지가 없는 몫이 발생되는 경우에 사용되는 곱의 역을 이용한 제산을 수행하고, 나머지가 발생되는 경우에 산출된 몫은 제수에 따라 정해진 모듈리에 의해서 산출할 수 있는 몫의 최대 값보다 크게되는 조건에 따라, 연산의 종결 조건으로 사용한 연산기를 설계하였다. 그러므로, 연산결과 종결 조건을 몫의 최대치와 대응된 제수 값을 사용함으로써 제산 연산기의 크기를 감소시킬 수 있었으며, 기존의 연산기와 비교하여 연산기의 크기와 연산 시간에 있어서 우수한 특성을 가지므로 작은 수의 제산에 적합한 컴퓨터 그래픽의 스케일링에 사용하기에는 적합하다. 그러나, 제수가 모듈리 보다 작아야 한다는 제한점을 가지므로, 앞으로 제수의 범위가 광범위 한 경우에도 적용할 수 있는 향상된 잉여수 제산 연산기에 대한 연구가 필요할 것으로 사료된다.

## 참고문헌


[1] Nicholas S. Szab, Richard I. Tanakas (1967), *Residue arithmetic and its applications to computer technology*, McGrow-Hill,

[2] K.D. Weinmann, M.A. Soderstrand and S. Shebani(1982), Evaluation of new hardware for a high-speed, digital, adaptive filter using the residue number system, *16th Asilomar Conference on Circuits, Systems, and Computers*, 187-191.

[3] 金龍成 外 1(1996), "高速 그래픽 處理를 위한 剩餘數係 乘算器 設計에 關한 硏究", 「電子工學會論文誌」, 33(1), 25-37.

[4] 尹賢植, 趙源敬(1993), "剩餘數係를 利用한 디지탈 神經網回路의 實現", 「電子工學會論文誌」, 30(2), 44-50.

[5] Michael A. Soderstrand, Bhaskar Sinha (1984), A pipelined recursive residue number system digital filter, *IEEE Transactions on Circuits and Systems*, CAS-31(4), 415-417.

[6] A. Hiasat, and H. Abdel-Aty-Zohdy(1995), High-speed division algorithm for residue number system, in the Processing of *1995 IEEE International Symposium on Circuits and Systems*, 3, 1996-1999.

[7] A. Hiasat, and H. Abdel-Aty-Zohdy(1997), Design and implementation of an RNS division algorithm, in the Processing of *1997 IEEE International Symposium on Computer*, March, 240-249.

# A study on the design of general division operator for the divisor with a small number in RNS

Yong-Sung Kim*

## Abstract

Many kind of operators using Residue Number System are used to design the special purpose processor for many merits in Digital Signal Processing, Computer Graphics, etc. But It get demerits for general division and the magnitude comparison.

In this paper, general division operator for divisor with a small number in RNS is proposed. If the result of division using the multiplicative inverse has remainder, the quotient of this is larger than maximum quotient of division that has the same divisor to dividend of the maximum size. This condition is used for the ending condition of the recursive operation. And, the divisor is substitute for the compared value of quotients. So, the proposed division operator has a small size and fine operation speed, but with the limitation of divisor.

* Dept of Computer and Internet, Yeojoo Institute of Technology