정보학연구 제4권 제4호 2001년 12월(pp.51-63)
The Journal of Information Technology, Vol.4, No.4

분할 잉여수를 사용한 혼합기수변환기 설계에 관한 연구

김용성*

요 약

잉여수계(Residue Number System)는 각 모듈러스에 자리올림수의 전달이 필요 없고, 병렬 구조를 이루므로, 디지털 신호처리 및 신경망 처리기와 같은 전용 프로세서 설계에서 사용된다. 그러나, 크기 비교 및 부호 검출 시에 혼합기수변환(Mixed Radix Conversion)이 요구되며, 이는 전체 연산 속도를 저해하는 요인이 된다.

그러므로 본 논문에서는 혼합기수 변환의 속도를 향상시키기 위하여 잉여수 분할 방법을 개선한 혼합기수변환기를 설계하였다. 설계된 변환기는 기존의 변환기에 비하여 연산기의 크기는 증가하지만 연산시간은 최대 2배가 향상되었다.

Ⅰ. 서론

전용 프로세서 설계 시 기본 연산기의 속도는 전체 시스템처리 과정에서 중요한 요소이므로, 자리올림수 전달에 의한 연산속도의 저하는 시스템 속도 향상을 저해하는 요인이 된다. 비가중치 수 체계(Unweighted Number System)인 잉여수계(RNS : Residue Number System)는 모듈리(moduli) 간에 자리올림수 전달이 없고, 병렬 적으로 연산이 수행되며, 연산표(Look Up Table)에 의한 연산기를 사용하여 사칙 연산 및 복합연산 등이 동일한 속도로 수행되는 장점을 가지므로,[1] 디지털 신호처리, 신경망 처리, 컴퓨터 그래픽 처리 등의 전용 프로세서 설계에 주로 사용된다.[2]~[7] 그러나 대소 비교, 부호 검출 및 오버후로우(overflow)검출과 같은 연산의 경우는 가중치 수 체계(Weighted Number System)로 변환이 요구된다. 혼합기수변환기(MRC: Mixed Radix Converter)는 가중치 수 체계로 전환시키는 변환기이며, 잉여수 계 연산기의 속도를 저하시키는 주요 원인이 된다. Suzabo와 Tanak, Garcfa가 제안한 알고리즘은 2차원 병렬구조를 갖으며,[8] n 개의 모듈리인 경우, (n-1)의 처리 단계가 소요된다. Chakraborti의 잉여수 분할법은 직렬구조로 기수확장(Base Extension)을 사용하여 자리 올림수는 없으나,[9] 모듈리 수가 증가되면 부가 연산으로 인한 속도 저하가 발생된다. Hwang이 제안한 병렬구조는 구조가 간략하고 연산기의 크기가 감소되는 이점이 있지만[10], 자리올림수 전달에 의한 지연이 포함되어 모듈리 증가 시 연산속도가 감소되는 단점을 갖는다.

그러므로 본 논문에서는 분할된 잉여수에 기존의 기수확장을 사용한 혼합기수변환기를 개선하여, 분할된 잉여수의 상위 모듈리에 대한 혼합기수변환과정이 생략되어 자리 올림수의 전달이 필요 없으며, 고속의 연산을 수행 할 수 있는 혼합기수변환기(MRC: Mixed Radix Converter)

* 여주대학 컴퓨터 사이언스과 부교수

를 설계하고자 한다.

II. 혼합기수변환

서로 소인 모듈리(moduli) P를 선택한 경우, 임의의 정수 X에 대해서 사용할 수 있는 수의 범위와 잉여수 표현 R은 식 (1)과 같다.[1][8]

$$0 \le X < M,\ M = \prod_{i=1}^{n} m_i,\ P = \{m_1, m_2, \ldots, m_n\},\ n: \text{정수}$$
$$r_i = |x|_{m_i} = \text{mod}\ m_i \text{일 때},$$
$$X \xrightarrow{RNS} (|X|_{m_1}, |X|_{m_2}, \ldots, |X|_{m_n})$$
$$\text{또는, } R = (r_1, r_2, \ldots, r_n) \quad (1)$$

정수 X는 식(1)과 같이 같이 n개 터플(tuple)로 구성되고. 잉여수계에서 이항연산(Binary operation) $Z = X \circ Y$ 는 식 (2)와 같이 정의된다.

$$Z = (|Z|_{m_1}, |Z|_{m_2}, \ldots, |Z|_{m_n})$$
$$|Z|_{m_i} = |X|_{m_i} + |Y|_{m_i}, \quad (\text{단}, i = 1, \ldots, n,\ n: \text{정수}) \quad (2)$$

제산의 경우는 식 (3-1)과 같은 곱의 역(multiplicative Inverse)를 사용하며, 모듈러스 m_i이고, 젯수가 x 피젯수가 y일 때 몫 Z는 다음 식 (3-1)과 같이 표시된다.

$|d \cdot y|_{m_i} = 1$이 성립되는 경우,

$$|1//y|_{m_i} = |d|_{m_i} \quad (3\text{-}1)$$

$$|Z|_{m_i} = |x//y|_{m_i} \quad (y, i\text{는서로소}, \ x/y = \text{정수}, \ '//': \text{곱의역}) \quad (3\text{-}2)$$

혼합기수변환은 잉여수를 가중치 수 체계로 전환하기 위하여 사용되며, 정수 X에 대한 혼합기수 표현이 (a_1, a_2, ... , a_n)인 경우, n번째 기수(Radix)를 $\prod_{i=1}^{n-1} m_i$로 하고, 계수를 a_n으로 표시하는 경우, 임의 정수 X의 혼합기수 표현은 식(4)와 같이 고정기수계(Fixed Radix Number System)로 표현된다.

$$X = a_n \prod_{i=1}^{n-1} m_i + \ \ldots\ + a_3 m_1 m_2 + a_2 m_1 + a_1 \quad (4)$$

식 (4)의 양변에 모듈리 m_1에 대한 잉여수를 취하면 $|X|_{m_1} = a_1$이므로, $a_1 = r_1$ 이다. 식(4)의 양변을 a_1으로 감산하고 m_1으로 제산을 수행하면 식(5)과 같이 표현되며, 식(5)에 m_2의 잉여수를 취하면 식(6)과 같이 a_2가 산출된다.

$$(X - a_1)/m_1 = (a_n \prod_{i=3}^{N-1} m_i + \ldots + a_3) m_2 + a_2 \quad (5)$$

$$|(X - a_1)/m_1|_{m_2} = |(a_n \prod_{i=3}^{N-1} m_i + \ldots + a_3) m_2|_{m_2} + |a_2|_{m_2}$$
$$|a_2|_{m_2} = |(X - a_1)/m_1|_{m_2} \quad (6)$$

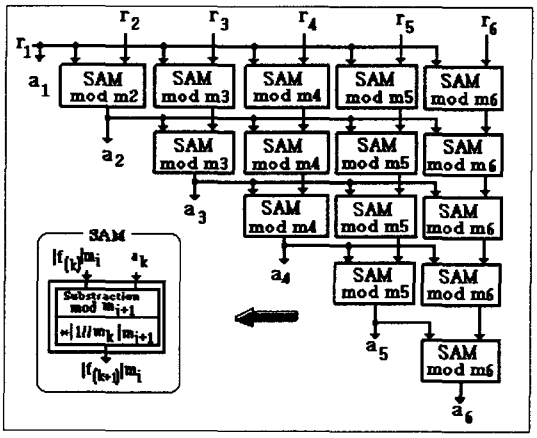
동일한 방법으로 (a_1, a_2, ... , a_n)을 산출하기 위하여 식(4)~(6)의 연산 과정을 식(7)과 같이 순환식으로 표현할 수 있다.[8]

$f_{(1)}=X$이라면 $|f_{(1)}|m_1=r_1=a_1$

$f_{(2)}=(f_{(1)}-a_1)/m_1,\ |f_{(2)}|m_2=a_2$

$f_{(k+1)}=(f_{(k)}-a_k)/m_k$

$a_{k+1}=|f_{(k+1)}|m_{k+1},$
(단, $k=1,\ldots,n-1$, n:정수) (7)



(그림 1) 병렬 혼합기수변환기의 구조(모듈리 수=6)

(Fig 1) Parallel Mixed Radix Converter (No. of moduli=6)

식(7)에 의하여 모듈리가 6개인 경우, 2차원 병렬 혼합기수 변환기를 그림 1에 나타내었다. 최 상단의 r_1의 광역 연결을 1단이라 하면 k단에서는 a_k의 연산결과가 출력되며, k단의 각 모듈리에서는 식(8)의 연산이 SAM(Substraction and Multiplication)에서 수행된다.

$|f_{(k)}-a_k|m_i,\ |1/m_k|m_i$
$(i=k,\ldots,n,\ k\le n,\ k,n$:정수) (8)

그림 1의 경우 15개의 승산기 및 감산기가 요구되는데, 이 연산기는 복합 연산이 가능한 연산표(LUT:Rom Look up Table)을 사용하면 15개의 연산표로 설계할 수 있다. 모듈리가 n개인 경우 (n-1)n/2개의 LUT가 소용되고, 1개 LUT의 연산 시간이 t_{LUT}인 경우 총 연산시간은 $(n-1)t_{LUT}$가 소요된다.

Ⅲ. 분할 잉여수를 사용한 혼합기수 변환

3.1 1개씩 분할된 잉여수를 사용한 혼합기수변환기

모듈리 수가 n개이고 임의의 정수 X에 대한 잉여수 표현이 $X=(r_1, r_2,\ldots, r_n)$와 같을 때, 1개씩 분할된 잉여수의 표현은 다음과 같다. i번째 잉여수 표현 X_i에 r_i가 1개만 포함되고 다른 잉여수는 "0"이 되도록 식(9)와 같이 상호 직교성을 갖도록 표현한다.[9]

$X_1=(r_1,0,\ldots\ldots,0).\ X_2=(0,r_2,0,\ldots,0),$
$\ldots\ldots\ Xn=(0,0,\ldots\ldots,r_n)$
$X=X_1+\ \ldots+X_n$
(단, $0\le X<M,\ 1\le i\le n,\ n$:정수) (9)

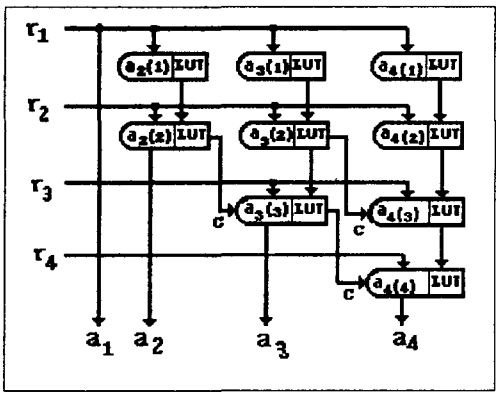
식 (9)에 대한 혼합기수변환은 X_i (i=1,..., n)는 식 (10)으로 표현되므로,

$X_1=(a_1,\ a_{2(1)},\ a_{3(1)},\ \ldots,\ a_{n(1)}),$
$X_2=(0,\ a_{2(2)},\ a_{3(2)},\ \ldots,\ a_{n(2)}),$
$\ldots\ldots\ Xn=(0,0,\ldots\ldots,a_{n(n)})$ (10)

a_i는 식(11)과 같이 X_i의 잉여수에 대한 혼합기수변환의 각 분할계수의 합으로 표현된다.

$$a_1 = |a_1|_{m_1},\ a_2 = |a_{2(1)} + a_{2(2)}|_{m_2},$$
$$a_3 = |a_{3(1)} + a_{3(2)} + a_{3(3)}|_{m_3},$$
$$\ldots, a_n = \left|\sum_{k=1}^{n} a_{n(k)}\right|_{m_n} \quad (11)$$

그림 2에 모듈 리가 4개인 경우 식(11)에 의한 1개씩 분할된 잉여수를 사용한 혼합기수변환기를 나타내었으며, 각 계수 및 합은 복합 연산이 가능한 연산표(LUT)를 사용하여 구성하였다. 계수 연산용으로 n(n+1)/2-1개의 연산표가 소요되며, n단계 이후 출력이 생성된다.



(그림 2) 1개씩 분할된 잉여수를 사용한 혼합기수변환기(모듈리수:4)

(Fig 2) MRC using partitioned residue one by one(No. of Moduli:4)

3.2 2개로 분할된 잉여수에 기수확장을 사용한 혼합기수변환기

잉여수를 2개씩 분할하여 혼합기수변환기를 설계하는 경우, 모듈 리가 4개일 때, 임의 정수 X에 대한 잉여수 표현 $X=(r_1, r_2, r_3, r_4,)$을 식(12)와 같이 두 개의 잉여수 표현으로 분리한다.[9]

$$RNS:\ X_1=(r_1,\ r_2,\ 0,\ 0),$$
$$MRC:\ X_1=(a_1, a_2,\ a'_3,\ a'_4) \quad (12\text{-}1)$$

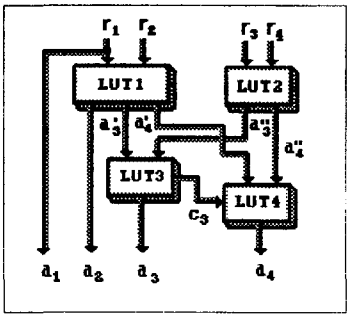
$$RNS:\ X_2=(0,\ 0,\ r_3,\ r_4),$$
$$MRC:\ X_2=(0,\ 0,\ a''_3,\ a''_4) \quad (12\text{-}2)$$

$$X = X_1 + X_2 \quad (\text{단},\ 0 \le X < M)$$

$$a_3 = |a'_3 + a''_3|_{m_3},\quad a_4 = |a'_4 + a''_4|_{m_4},$$

$$\text{MRC}:\ X=(a_1,\ a_2,\ a_3,\ a_4) \quad (12\text{-}3)$$

식(12-1)에서 혼합기수 변환된 분할계수 a_1, a_2, a'_3, a'_4가 생성되고, 식(12-2)의 r_3와 r_4에서 생성된 a'_3, a'_4는 식 (12-3)과 같이 최종적으로 합산된다. $a'_3 + a''_3$ 잉여수 연산 시 자리올림수는 $a'_4 + a''_4$ 에 합산되며, 이 때 발생된 자리올림수는 무시된다. 모듈 리가 4개인 경우, 2개씩 분할된 잉여수를 사용한 혼합기수변환기를 그림 3에 나타내었다.



(그림 3) 2개로 분할한 잉여수를 사용한 혼합기수변환기(모듈리수: 4)

(Fig 3) MRC using partitioned residue number by two(No. of Moduli:4)

그림 3에서 LUT1은 a_2, a'_3, a'_4를 생성하며, LUT3의 합산결과는 LUT4로 전달되므로 3단계 후 혼합기수의 계수가 생성된다.

모듈 리가 6개인 경우 전체 모듈리를 2개로 분할하여 적용하는 경우, 연결 구조는 단순해질 수 있으나, 연산기의 크기가 광대해지는 문제점을 갖게된다. 그러므로 그림3의 설계를 확장하여 적용하며, 임의 정수 X에 대한 잉여수 표현이 $X=(r_1, r_2, r_3, r_4, r_5, r_6$일 때, 2개씩 분할된 잉여수 및 혼합기수 표현은 식(13)과 같다.

$$RNS: X_1=(r_1, r_2, 0, 0, 0, 0),\quad MRC: X_1=(a_1, a_2, a'_3, a'_4, a'_5, a'_6) \tag{13-1}$$

$$RNS: X_2=(0, 0, r_3, r_4, 0, 0),\quad MRC: X_2=(0, 0, a''_3, a''_4, a''_5, a''_6) \tag{13-2}$$

$$RNS: X_2=(0, 0, 0, 0, r_5, r_6),\quad MRC: X_2=(0, 0, a'''_3, a'''_4) \tag{13-3}$$

$$X= X_1+ X_2+ X_3 \quad (\text{단, } 0\le X< M)$$

$$a_3= | a'_3+a''_3 | m_3,$$

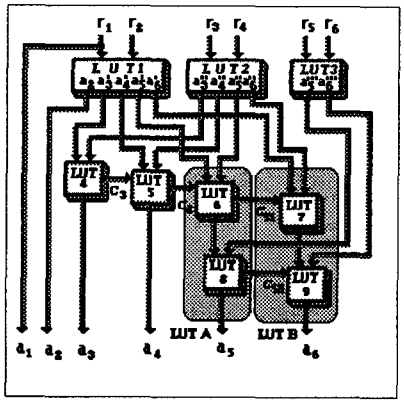
$$a_4= | a'_4+a''_4 | m_4,$$

$$a_5= | a'_5+a''_5+a'''_5 | m_5,$$

$$a_6= | a'_6+a''_6+a''_6 | m_6 ,$$

$$MRC: X=(a_1, a_2, a_3, a_4, a_5, a_6) \tag{13-4}$$

모듈 리가 6개일 때, 식 (13)에 의한 혼합기수 변환기를 그림 4에 나타내었다. LUT1은 a_1을 제외한 식(13-1)의 혼합기수의 계수가 생성되며, LUT2와 LUT3는 각각 식(13-2)와 식(13-3)의 계수가 생성된다. LUT4에서 LUT9까지는 식(13-4)의 각 분할된 계수의 합을 연산하며, 연산시 발생된 자리올림 수는 다음 LUT에 전달된다. 자리올림수 전달에 의한 지연으로 6단계 후 혼합기수 변환된 계수가 출력되고, 3입력 연산표인 LUTA, LUTB를 사용하는 경우, 연산표 수가 7개로 줄고 5단계 이후 출력할 수 있으나, 연산표의 크기는 증가한다.



(그림 4) 2개로 분할한 잉여수를 사용한 혼합기수변환기(모듈리수: 6)

(Fig 4) MRC using partitioned residue number by two(No. of Moduli:6)

3.3 분할된 잉여수에 기수확장을 사용한 혼합기수변환기

2절과 동일하게 잉여수를 2개씩 분할하여 기본적으로 4개의 모듈리에 대한 설계를 하고, 이를 확장 적용한다. 기수 확장(BE:Base Extension)은 모듈러스 P가 ($m_1, m_2, ..., m_n$) 일 때, 새로운 모듈러스 m_k에 대한 잉여수를 산출하는 방법이다. 모듈리 수가 4개인 경우, 임의 정수 X에 대한 잉여수 표현은 $X=r_1, r_2, r_3, r_4$)이며,

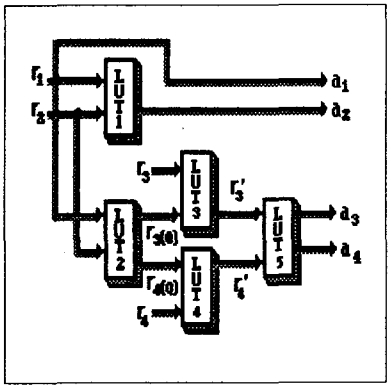
기수 확장된 모듈리를 포함하여 식(14)와 같이 두 개의 잉여수 표현으로 분리한다.[10]

$$X_1 = a_1 + a_2 m_1,\ X_1 = (r_1,\ r_2,\ r_{3(0)},\ r_{4(0)}) \quad (14\text{-}1)$$

$$X_2 = a_3 m_1 m_2 + a_4 m_1 m_2 m_3,\quad X_2 = (0,\ 0,\ r'_3,\ r'_4)$$
$$r'_3 = |\ r_3 - r_{3(0)}\ |\ m_3,\ \ r'_4 = |\ r_4 - r_{4(0)}\ |\ m_4 \quad (14\text{-}2)$$

$$X = X_1 + X_2 \quad (\text{단},\ 0 \le X < M) \quad (14\text{-}3)$$

식(14-1)에서 r_1과 r_2를 혼합기수 변환하면 계수 a_1과 a_2가 생성되고, a'_3과 a'_4를 '0'으로 하기 위하여 $r_{3(0)}$와 $r_{4(0)}$를 기수확장에 의해 표현한다. 식 (14-2)에서 r'_3와 r'_4는 기수확장에 대한 보상된 값으로 이에 따라 혼합기수 변환하면 계수 a_3과 a_4가 생성된다.



(그림 5) 분할된 잉여수에 기수확장을 사용한 혼합기수변환기(모듈리수: 4)

(Fig 5) MRC using base extension in partitioned residue number(No. of Moduli:4)

그림 5에 모듈리 수가 4개인 경우, 4개의 LUT로 구성된 분할 잉여수에 기수확장을 사용한 혼합기수변환기를 나타내었다. LUT3는 $|\ r_3 - r_{3(0)}\ |\ m_3$의 연산을 수행하고, LUT4는 $|\ r_4 - r_{4(0)}\ |\ m_4$의 연산을 수행한다. 또한, 식 (14-1)에 의해 계수를 구하는 경우 a'_3=0, a'_4=0이므로, 이에 대한 연산표는 구성할 필요가 없다. 식(14-2)의 a_3와 a_4의 연산에서도 자리올림수의 연산이 필요 없다는 장점을 갖으며, 3단계 이후 출력이 생성된다.

모듈리가 6개인 경우는 식 (15)와 같으며 식 (14)를 확장 적용한다.

$$X_1 = a_1 + a_2 m_1,\ X_1 = (r_1,\ r_2,\ r_{3(0)},\ r_{4(0)},\ r_{5(0)},\ r_{6(0)}) \quad (15\text{-}1)$$

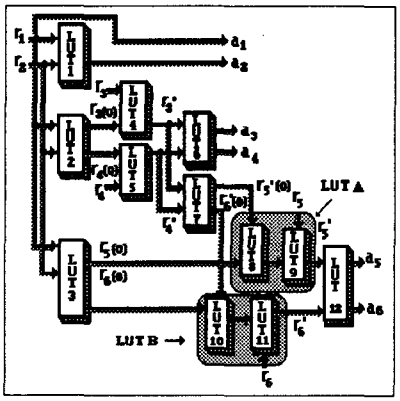
$$X_2 = a_3 m_1 m_2 + a_4 m_1 m_2 m_3,$$
$$X_2 = (0,\ 0,\ r'_3,\ r'_4, r'_{5(0)}, r'_{6(0)})$$
$$r'_3 = |\ r_3 - r_{3(0)}\ |\ m_3,\ \ r'_4 = |\ r_4 - r_{4(0)}\ |\ m_4 \quad (15\text{-}2)$$

$$X_3 = a_5 m_1 m_2 m_3 m_4 + a_6 m_1 m_2 m_3 m_4 m_5,$$
$$X_3 = (0,\ 0,\ 0,\ 0, r'_5, r'_6)$$
$$r'_5 = |\ r_5 - (r_{5(0)} + r'_{5(0)})\ |\ m_5,$$
$$r'_6 = |\ r_6 - (r_{6(0)} + r'_{6(0)})\ |\ m_6 \quad (15\text{-}2)$$

$$X = X_1 + X_2 + X_3 \quad (\text{단},\ 0 \le X < M) \quad (15\text{-}3)$$

모듈 리가 6개인 경우, 혼합기수변환기를 그림 6에 나타내었다. LUT2와 LUT3에서 식 (15-1)의 기수 확장된 잉여수 $r_{3(0)}$, $r_{4(0)}$, $r_{5(0)}$, $r_{6(0)}$를 생성하며, $r'_{5(0)}$와 $r'_{6(0)}$는 LUT7에서 연산된다. 식(15-2)에서 $r_{5(0)} + r'_{5(0)}$는 LUT8에서 연산되고, 이 에따라 LUT9에서 r'_5가 연산

된다. 모듈 리가 6개인 경우 총 12개의 LUT가 소요되며, 6단계 후 출력이 생성된다. LUT8 ,9 와 LUT10, 11에 3 입력 LUTA, B를 사용하는 경우 연산표 수 는 2개 감소하고 5단계 이후 출력이 생성되므로 연산시간은 줄일 수 있으나, 연산기의 크기는 증가한다.



(그림 6) 분할된 잉여수에 기수확장을 사용한 혼합기수변환기(모듈리수: 6)

(Fig 6) MRC using base extension in partitioned residue number(No. of Moduli:6)

Ⅳ. 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기

Ⅲ의 1,2에서 1개씩 또는 2개씩 분할된 잉여수를 사용하여 혼합기수변환기를 설계하는 경우 단순한 구조를 갖지만 자리올림수에 의한 지연으로 모듈 리가 6개인 경우 전체 속도가 는 저하된다. 또한, 분할 잉여수에 기수확장을 사용한 혼합기수변환기는 자리올림수에 대한 문제점은 없지만, 기수확장에 의한 분할 계수의 연산으로 연산속도가 지연된다.

그러므로 본 논문에서는 자리올림수에 의한 지연이 없고 처리 속도가 향상되도록 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기를 설계하고자 한다.

Ⅲ의 3의 식(14-1)에서 기수 확장은 $X_1=(r_1, r_2, r_{3(0)}, r_{4(0)})$에 의해서 생성된 a'_3과 a'_4를 '0'으로 하여, 분할된 잉여수 X_2의 분할계수 생성과 독립적으로 연산을 수행하려는 목적이지만 중복적인 설계가 발생되게 되므로, 본 논문에서는 다음과 같이 제안하고자 한다. 기수 확장된 $r_{3(0)}, r_{4(0)}$ 를 식(16)과 같이 표현한다.

$$X_1=(r_1, r_2, r_{3(0)}, r_{4(0)}),$$
$$MRC: (a_1, a_2, a'_3, a'_4)$$
$$a_1=r_1,\ a_2=|r_2-a_1|m_2\cdot|1//m_1|m_2$$
$a'_3=0$이고, $a'_4=0$이려면
$$a'_3=||r_{3(0)}-a_1|m_3\cdot|1//m_1|m_3-a_2|m_3=0$$
$$a'_4=||r_{4(0)}-a_1|m_4\cdot|1//m_1|m_4-a_2|m_4=0$$
$$r_{3(0)}=|a_2m_1+a_1|m_3$$
$$r_{4(0)}=|a_2m_1+a_1|m_4 \qquad (16)$$

식(14-2)는 식(16)에 의해 식(17-1), 식(17-3)과 같이 a_1 및, a_2와 관련된 r'_3와 r'_4로 표현할 수 있다.

$$r'_3=|r_3-r_{3(0)}|m_3,\ r'_3=|r_3-|a_2m_1+a_1|m_3|m_3$$
$$a_{31}=|r_3-a_1|m_3\text{이라면},\quad r'_3=|a_{31}-a_2m_1|m_3 \qquad (17\text{-}1)$$

$$a_3 = | \ r'_3 \cdot | 1//m_1m_2 | m_3 | m_3 \quad (17\text{-}2)$$

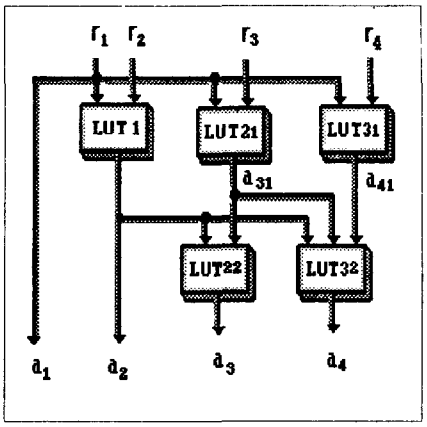
$$r'_4 = | r_4 - r_{4(0)} | m_4, \ r'_4 = | r_4 - | a_2m_1 + a_1 | m_4 | m_4$$

$$a_{41} = | r_4 - a_1 | m_4 \text{이라면}, \quad r'_4 = | a_{41} - a_2m_1 | m_4 \quad (17\text{-}3)$$

$$a_4 = | | r'_4 \cdot | 1//m_1m_2 | m_4 - a_3 | m_4 \cdot | 1//m_3 | m_4 | m_3 \quad (17\text{-}4)$$

그림 7에 모듈리 수가 4개인 경우 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기를 나타내었다. LUT1에서는 a_2가 연산되며, 식(17-1)의 a_{31}은 LUT21에서 연산된 후 a_2와 같이 LUT22에 입력되어 식(17-2)의 연산이 연산표에 의해 LUT22에서 수행된다. 식(17-1)의 a_{31}은 LUT21에서 연산된 후 a_2와 같이 LUT22에 입력되어 식(17-2)의 연산이 연산표에 의해 LUT22에서 수행된다.

a_{31}과 같은 방법으로 식 (17-3)에 의하여 LUT31에서 a_{41}이 연산되며, 식 (17-4)의 연산은 LUT32의 수행된다. 식의 순서대로 식(17-4)를 연산하는 경우, a_3 연산 후 a_4가 생성되므로 연산표의 크기는 줄일 수 있으나 연산속도가 느려지게 된다. 그러나, 식(17-4)는 식(17-1),식(17-2), 식(17-3)에 의해 표현할 수 있으며, 연산표에 의한 연산은 복합 연산이 가능하다. 따라서 a_3의 입력 없이 a_{41}, a_2 및 a_{31}의 입력에 따라 a_4를 LUT32에서 출력함으로써 출력 속도를 향상시킬 수 있다. 모듈리가 6개인 경우는 식(16), 식(17)을 확장 적용한다.



(그림 7) 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기(모듈리 수:4)

(Fig 7) Modified MRC using BE in partitioned residue number (No. of Moduli:4)

분할된 모듈리는 식(15-1)과 동일하며, a_1, ~a_4까지는 식(16),식(17)과 동일하다.

$X_1 = (r_1, r_2, r_{3(0)}, r_{4(0)}, r_{5(0)}, r_{6(0)})$에서 $r_{5(0)}$와 $r_{6(0)}$는 다음 식과 같이 표현된다.

$$a'_5 = | \ | (r_{5(0)} - a_1) \cdot | 1//m_1 | m_5 - a_2 | \ m_5 | m_5 = 0$$

$$a'_6 = | \ | (r_{6(0)} - a_1) \cdot | 1//m_1 | m_6 - a_2 | \ m_6 | m_6 = 0$$

$a'_5 = 0, a'_6 = 0$이려면,

$$r_{5(0)} = | \ a_2m_1 + a_1 | m_5$$

$$r_{6(0)} = | \ a_2m_1 + a_1 | m_6$$

또한, $X_2 = (0, 0, r'_3, r'_4, r'_{5(0)}, r'_{6(0)})$에서 $r'_{5(0)}$와 $r'_{6(0)}$는 다음 식과 같이 표현된다.

$$a''_5 = | \ | (r'_{5(0)} \cdot | 1//m_1m_2 | m_5 - a_3) \cdot | 1//m_3 | m_5 - a_4 | m_5 = 0$$

$$a''_6 = | \ | (r'_{6(0)} \cdot | 1//m_1m_2 | m_6 - a_3) \cdot | 1//m_3 | m_6 - a_4 | m_6 = 0$$

$a''_5 = 0, a''_6 = 0$이려면,

$r'_{5(0)} = | a_4 m_3 m_2 m_1 + a_3 m_2 m_1 | m_5$

$r'_{6(0)} = | a_4 m_3 m_2 m_1 + a_3 m_2 m_1 | m_6$

또한, $X_3 = (0, 0, 0, 0, r'_5, r'_6)$에서 r'_5, a_5는 다음 식과 같이 표현한다.

$r'_5 = | r_5 - r_{5(0)} - r'_{5(0)} | m_5$

$r'_5 = | r_5 - | a_4 m_3 m_2 m_1 + a_3 m_2 m_1 | m_5 - | a_2 m_1 + a_1 | m_5 | m_5$

$a_{51} = | r_5 - r_1 | m_5$, $a_{52} = | a_{51} - a_2 m_1 | m_5$라 하면,

(18-1)

$r'_5 = | a_{52} - | (a_4 m_3 + a_3) m_2 m_1 | m_5 | m_5$ (18-2)

$a_5 = | r'_5 \cdot | 1//m_1 m_2 m_3 m_4 | m_5 | m_5$ (18-3)

r'_5와 a_6는 다음 식과 같이 표현한다.

$r'_6 = | r_6 - r_{6(0)} - r'_{6(0)} | m_6$

$r'_6 = | r_6 - | a_4 m_3 m_2 m_1 + a_3 m_2 m_1 | m_6 - | a_2 m_1 + a_1 | m_6 | m_6$

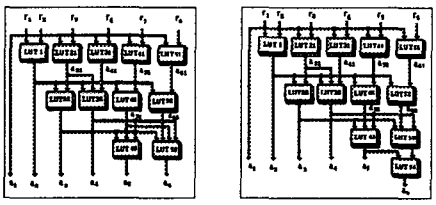
$a_{61} = | r_6 - r_1 | m_6$, $a_{62} = | a_{61} - a_2 m_1 | m_6$라 하면,

(19-1)

$r'_6 = | a_{62} - | (a_4 m_3 + a_3) m_2 m_1 | m_6 | m_6$ (19-2)

$a_6 = | (r'_6 - a_5) \cdot | 1//m_1 m_2 m_3 m_4 | m_5 | m_5$ (19-3)

모듈 리가 6개인 경우, 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기를 그림 8 (a)에 나타내었다. LUT1에서 LUT32까지는 모듈 리가 4개인 경우와 동일하다, 식 (18-1)의 a_{51}은 LUT41에서 연산되며, a_{52}은 LUT42에서 연산된다. 식(18-2)과 식(18-3)에 따라 입력 a_{52}, a_3, a_4에 의한 출력 a_5는 LUT43에서 수행한다. 식 (19-1)의 a_{51}와 a_{52}는 LUT51과 LUT52에서 연산되고, 식(19-3)의 a_6는 식 (18-2), 식(18-3), 식(19-2)에 의해 LUT53에서 입력 a_{52}, a_{62}, a_3, a_4 에 따라 생성되므로, a_5 입력과 관계없이 연산이 되어 출력시간이 절감된다. 또한, 설계된 변환기의 각 연산표 출력에 레지스터를 부가하면 파이프라인(Pipeline)된 혼합기수변환기로 설계할 수 있다. 그림 8(b)는 LUT53의 입력되는 a_{52}를 분리하고, LUT53과 a_5에 의해 LUT54에서 계수 a_6를 출력하면 연산표 크기를 줄일 수 있다. 그림 8 (a)에서 총 11개의 LUT가 소요되며, LUT53의 연산기의 크기를 감소시키기 위하여 사용되는 모듈리는 $m_1 > m_2 > ... > m_6$가 되도록 한다. 3단계 후 출력이 생성되므로 그림 6과 비교할 때, 출력 속도가 2배 향상되었음을 알 수 있다.



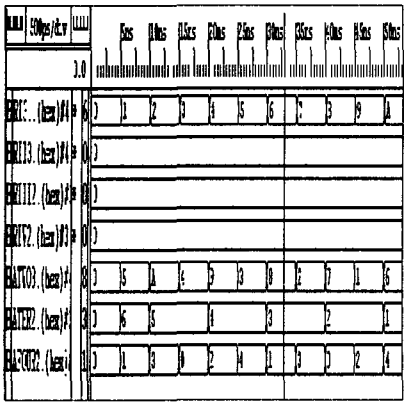
(그림 8) 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기(모듈리 수:6)
(a) 연산시간을 고려한 경우
(b) 연산표 크기를 고려한 경우

(Fig 8) Modified MRC using base extension in partitioned residue number(No. of Moduli:6)
(a) In case of considering operation time
(b) In case of considering size of LUT

V. 실험 및 고찰

본 논문에서 설계한 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기에 대한 논리

적 검증은 High Level Language와 VHDL을 사용하여 수행하였으며, 모듈리가 P=(13, 11, 7, 5)이고 정수 X의 범위가 $0 \leq X < 5004$인 경우 혼합기수 변환기의 입출력 신호를 그림 9에 16진수 형태로 표시하였다. RI3, RII3, RIII2, RIV2는 정수 X의 영여수 입력 r_1, r_2, r_3, r_4이고, 혼합기수 변환된 출력 a_1은 r_1과 동일하며, a_2, a_3, , a_4는 ATWO3, ATHR2, AFOUR2로 표시하였다. 그림 9에서 $X=1540_{10}$의 잉여수 $r_1=6_{16}$, $r_2=0_{16}$, $r_3=0_{16}$, $r_4=0_{16}$일 때, 혼합기수 변환된 출력 $a_1=r_1=6_{16}$, $a_2=8_{16}$, $a_3=3_{16}$, $a_4=1_{16}$에 대한 신호를 수직선으로 표시하였다.



(그림 9) 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기의 입출력 신호 (모듈리 :13, 11, 7, 5)

(Fig 9) Input-output signal of Modified MRC using base extension in partitioned residue number(moduli : 13, 11, 7, 5)

기존의 혼합기수 변환기와 본 논문에서 설계된 혼합기수변환기의 최소 연산표 크기 및 연산속도를 표 1에 나타내었다. 그림 4, 6에서 2입력 연산표로 구성한 경우는 "소"로 3입력 연산표를 사용한 경우는 "대"로 "연산표 전체 크기"에 표시하였고, 그림 8(b)의 경우도 "소"로 표시하였다. 연산시간은 1개 LUT의 연산 시간이 t_{LUT}인 경우로 표시하였으며 그림 2의 1개씩 분리된 잉여수의 혼합기수변환기가 가장 느리고, 그림 7,8의 경우가 가장 빠름을 알 수 있다. 연산표의의 개수는 그림 3,4의 경우가 가장 작으며, 연산표의 전체 크기는 그림 7, 8을 제외한 경우 다른 변환기의 경우에 유사한 크기이다. 그러므로 본 논문에서 설계한 혼합기수변환기의 경우 연산표의 크기는 증가 하지만 연산 시간은 다른 변환기 보다 최대 1/2 단축되었음을 알 수 있다.

〈표 1〉 분할 잉여수를 사용한 혼합기수변환기의 연산시간 및 연산표 크기 비교

〈Table 1〉 The comparison of operation time and LUT size of MRC using partitioned residue [P1=(13, 11, 7, 5), P2=(13, 11, 9, 7, 5, 4)]

MRC 종류	연산표 개수			연산시간 (tLUT)				연산표 전체크기 (bits)		
	모듈리수			모듈리 수						
	n	4	6	n	4	6		P1	P2	
그림 2	n(n+1)/2-1	9	20	n	4	6		1664	4130	
그림 3,4	3n/2-2	4	7	n-1	3	소	6	1862	소	4115
						대	5		대	4768
그림 5,6	(3n2-2n)/8	5	12	n-1	3	소	6	1787	소	3829
						대	5		대	4118
그림 7,8	n(n+2)/4	5	11	n/2	2	소	4	2426	소	6291
						대	3		대	8267

Ⅵ. 결론

잉여수계는 비가중치 수체계로 연산 시 모듈리 간에 자리올림 수의 전달이 필요 없고 병렬적인 구조로 수행 될 수 있으므로, 디지털 신호처리 등의 특수한 목적의 연산기 설계에 적합하다. 그러나, 연산 중 대소 비교 및 부호 판별 등에는 혼합기수 변환을 사용하여 가중치 수 체계로 변환을 하며, 변환 시 연산 속도가 저하되는 문제점을 갖는다.

그러므로, 본 논문에서는 혼합기수변환기의 속도를 향상시키기 위하여 Suzabo와 Tanaka에 의해서 제안된 병렬구조의 혼합기수 연산기와 분할 잉여수를 사용한 혼합기수 연산기를 고찰하고, 분할 잉여수의 변환 시 분할된 계수의 누적에 자리올림수에 의한 동작시간의 지연과, 기수확장법의 적용 시 속도저하 문제를 해결하기 위하여 분할 잉여수에 기수확장을 사용한 개선된 혼합기수변환기를 설계하였다. 설계된 혼합기수변환기는 비교표에서 다른 변환기 보다 연산기의 크기는 증가되었으나 연산 속도는 최대 2배까지 향상되었음을 알 수 있었다.

앞으로 설계된 연산기의 크기를 감소시키는 연구와 잉여수계로 설계된 전체 시스템에 적용할 때 예측된 혼합기수 변환에 대한 연구 및 효율적인 잉여수의 2진수 변환에 대한 연구가 계속되어야 할 것으로 사료된다.

참고문헌

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

[2] 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.

[3] 金龍成 外 1, "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, 第 33卷, B編, 第 1號, pp. 25-37, 1996, 1月.

[4] 尹賢植, 趙源敬, "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, 第 30卷, B編, 第 2號, pp. 44-50, 1993, 2月.

[5] 金龍成, "MAC演算用 多重 剩餘數係에 대한 硏究", 驪州大學 産業技術 硏究所 論文集, 第1卷, pp.317-332, 1995, 11月

[6] Michael A. Soderstrand, Bhaskar Sinha, "A Pipelined Recursive Residue Number System Digital Filter", *IEEE Transactions on Circuits and Systems*, Vol. CAS-31, No. 4, April, 1984.

[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] Antonio Garcfa and Graham A. Jullien, Comments on "An Arithmetic Free Parallel Mixed Radix Conversion Algorithm", *IEEE Trans. on Circuits and Systems-II: Analog and Digital*

Siganal Processing, Vol. 46, No 9, pp. 1259-1260, September, 1999.

[9] N. B. Chakraborti, John S. Soundararajan, and A. L. N. Reddy, "An Implementation of Mixed Radix Conversion for Residue Number Applications", *IEEE Trans. on Computer*, Vol. C-35, No 8, pp. 762-764, August, 1986.

[10] C. H. Huang, "A fully parallel mixed radix conversion algorithm for RN applications", *IEEE Trans. on Computer*, Vol. C-31, pp. 321-325, April, 1982.

A Study On the Design of Mixed Radix Converter using Partitioned Residues.

Yong-Sung, Kim*

Abstract

Residue Number System has carry free operation and parallelism each modulus, So it is used for special purpose processor such as Digital Signal Processing and Neuron Processor. Magnitude comparison and sign detection are in need of Mixed Radix Conversion, and these operations are impediment to improve the operation speed.

So in this Paper, MRC(Mixed Radix Converter) is designed using modified partitioned residue to speed up the operation of MRC, so it has progressed maximum twice operation time but increased the size of converter comparison to other converter.

* Dep. of computer science, yeojoo college