論文 83-20-5-7

RNS에 의한 고속 곱셈기 구성에 관한 연구
(A Study on the High-Speed Multiplier Architecture Using RNS)

金 善 榮*, 金 在 功**
(Sun Young Kim and Jae Kong Kim)

要 約

조합논리회로를 사용한 고속 RNS 곱셈기의 구성을 제안하였다. 연산시간과 하드웨어 절감을 결정하는 최적 moduli 선택조건에 대해서도 검토하였으며 RRNS에서는 magnitude index를 사용한 변형된 CRT로써 출력 변환하였다. 제안된 곱셈기의 추정시간은 파이프 라인없이 NRNS인 경우 31.7ns, RRNS인 경우 47.95ns이었다.

Abstract

In this paper, an architecture for high-speed RNS multiplier were proposed by using combinational logic circuit. The optimum conditions of moduli set which could be saved hardware and operation time were also considered. In the case of RRNS multiplier, output translation could be achieved effectively by means of the modified CRT with magnitude index. It is shown that the estimated multiplication time is about 31.7 ns in NRNS, whereas 47.95 ns in RRNS, respectively.

I. 序 論

신호처리에 있어서 응답의 신속성은 곱셈의 연산속도에 좌우된다.[1-3] 곱셈의 연산속도를 개선하기 위한 방법으로는 부분곱을 빨리 더하는 회로적인 방법과 부분 곱의 총수를 줄이는 알고리즘적인 방법이 혼용되어 오고 있다.[4] 회로적인 구성은 serial과 parallel 두 형태로 크게 나누어지고, 전자는 하드웨어 실현이 간단한 반면 shift-add 동작으로 인해 연산속도가 느리며[5] 후자는 adder array 및 반복 array 방법으로 다시 세분된다. 부분곱 bit 열을 줄이기 위해 카운터에 의한 덧셈으로써 곱셈을 행하는 adder array 방법은[6-8] serial한 경우보다 연산 속도는 빠르나 carry save adder의 게이트 수 감소, 보다 적은 연결, 그리고 지연의 최소화등이 문제가 된다.[9] 반복 array 방법은 carry를 대각선으로 더하는 방법, tree형태로 더하는 방법 및 이들을 혼용한 방법이 있다. 코딩 알고리즘은 이들 두 방법의 단점인 오퍼랜드 길이 증가에 따른 속도의 선형적 감소를 보완하는 한 방법이나 그룹 크기에 따라 추가 시간이 필요하다.[4]

Residue Number System(RNS) 연산은 같은 modulus 끼리의 연산이므로 carry가 없어 오퍼랜드를 짧게 나누어 병렬 처리하면 연산의 고속성 및 high precision 특성을 지닌 고속, 고정도 곱셈기 구성이 가능해 질 수 있다.

Svoboda-Valach[12] 및 Garner-Aiken[13-14]의 RNS 이론 컴퓨터 응용 시도이래 Szabo-Tanaka는 RNS 이론의 기틀을 수립했다.[14] 그러나 초기의 RNS기법은 core memory가 지닌 문제점과 각 digit가 독립적이어서 most significant bit가 없는 RNS로써는 나눗셈, 부호검출, 크기비교, 오우버플로우로 인해 실제 구성에 큰 관심이 되지 못했다. 그러나 최근 IC 기술

*準會員, **正會員, 東国大學校 工科大學 電子工學科
(Dept. of Electronic Eng., Dong Guk Univ.)

接受日字 : 1982年 7月 16日

의 향상은 ROM에 의한 residue변환 및 연산 장치의 구성을 가능케 하였으나[15-19] 오우버플로우를 없애기 위해 많은 메모리가 여전히 필요하다. Taylor[20]는 적절한 moduli 그리고 상용 곱셈기 (TRW-16HJ)와 ROM으로 구성된 룩업 테이블을 이용한 quarte square 곱셈방식으로 range를 48-72bit로 개선한 바 있고, 이때 연산속도는 약 400ns, pipeline하면 100ns이었다. 이처럼 이들 방법들은 residue 연산이 메모리 의존적이다.

본 논문에서는 ROM 대신 조합논리회로를 이용한 RNS 곱셈기의 이론적 구성을 제안하고 하드웨어 절약, 속도 향상의 관건인 moduli 선택 조건도 검토하였으며 입력 오퍼랜드는 sign-magnitude로 표현된 16bit 정수임을 전제로 한다.

Ⅱ. RNS 개괄

RNS는 fixed radix system과는 달리 서로소인 modulus 집합 $m=(m_1, m_2, \cdots, m_N)$으로 구성된다. x가 range $0 \le x < M = \prod_{i=1}^{N} m_i$ 안에 있는 임의 정수이면 euclidean 알고리즘에 의해 같은 구간내 임의의 정수 q_i, r_i는

$$x = q_i m_i + r_i \qquad 0 \le r_i < m_i$$

$$= m_i \left[\frac{x}{m_i}\right] + |x|_{m_i} \qquad (1)$$

을 만족해야 된다. 여기서 r_i는 x를 m_i로 나눈 least positive remainder이며 이를 residue라 하고

$$|x|_{m_i} = x \bmod m_i \qquad (2)$$

로 표기한다. 따라서 x와 M+x는 residue 값이 같으며 $x \in [0, M-1]$에서 N개 원소 $(r_1, r_2, \cdots\cdots, r_N)$ 에 의해 유일하게 결정될 때 Non-redundant RNS (NRNS), 시스템 구성에 유용하도록 여분의 R개 moduli를 도입 전체 range가 $M_T = \prod_{i=1}^{N+R} m_i$일때 redundant RNS (RRNS)라 한다. 두 수 x, y의 RNS 연산 특성은 다음과 같다.

1) 덧셈과 뺄셈

$$|x \pm y|_{m_i} = |\,|x|_{m_i} \pm |y|_{m_i}\,|_{m_i} \qquad (3)$$

2) 곱셈

$$|xy|_{m_i} = |\,|x|_{m_i} |y|_{m_i}\,|_{m_i} \qquad (4)$$

즉 두 수 합, 차, 곱의 rusidue는 각각 두 수 residue 합, 차, 곱의 residue이다.

Ⅲ. Range와 Moduli

1. Range

RNS를 2진 보수로 표현하면 전체 range M이 $\frac{1}{2}$로 축소되므로[14] 부호 비트를 제외한 나머지를 residue로 표현한다. 입력 오퍼랜드를 16bit로 가정하였으므로 필요한 range는 두 오퍼랜드의 곱 $M = 2^{30}$과 같거나 커야 한다.

2. Moduli 선택

입력 변환과 출력 변환을 쉽게 하고 하드웨어의 절감을 위한 조건은 다음과 같다.

1) Moduli는 $n2^k \pm 1$, 2^k 형태의 서로 소이어야 한다. 이런 moduli set만이 hardwired scaling 혹은 ORing이 가능하고 연산시간 단축과 구조가 간단해진다.

2) Scaling이 간단해지려면 multiplicative inverse는 2^k 또는 1에 가깝게 선택되어야 한다.

3) 필요한 range에 맞도록 여러개의 작은 moduli가 선택되어야 한다.

위 조건과 최적 column compression (C.C.)을[6-7][23] 위해 range에 적합한 moduli를 RRNS에서는 {65, 63, 31, 19, 17, 11}, NRNS에서는 {33, 32, 31, 29, 23, 19, 17}로 선택했다. 하드웨어 매트릭스 구성상 m_i 최대값을 각각 65, 40이하로 제한했다.

Ⅳ. RNS 곱셈기

그림 1은 제안된 RNS 곱셈기 구성을 나타낸다. 두 수의 곱 $xy=z$에서 2진 입력 x, y는 선택된 m_i에 의해 각각 residue $|x|_{m_i}$, $|y|_{m_i}$로 변환된 후 같은 modulus끼리 서로 곱해져 $|z|_{m_i}$가 되고 RRNS의 경우 magnitude index[21-22] P_z를 이용한 변형된 Chinese Remainder Theorem (CRT)에 의해 z가 산출되고 NRNS 경우는 P_z과정이 생략되며 CRT에 의해 직접 z가 산출된다.

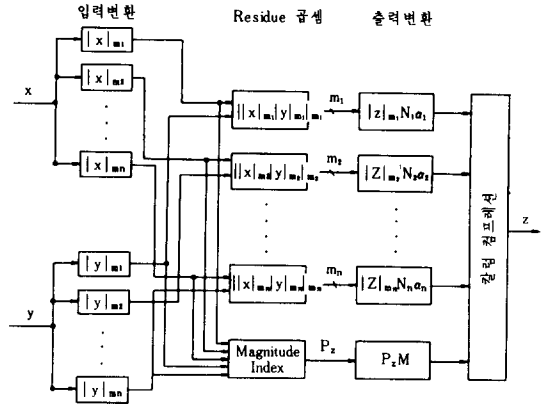

그림 1. RRNS 곱셈기 구성도

Fig. 1. RRNS multiplier block diagram.

1. 입력변환

입력 x 가 n bit 크기의 2진수라면 (1)식은

$$x = \sum_{j=0}^{n-1} B_j 2^j \qquad (5)$$

여기서 B_j는 binary digit이다. 양변에 mod m_i 취하면

$$|x|_{m_i} = \left|\sum_{j=0}^{n-1} B_j 2^j\right|_{m_i}$$

$$= \left|\sum_{j=0}^{n-1} B_j |2^j|_{m_i}\right|_{m_i} \qquad (6)$$

y에 대해서도 같다.

Modulus가 $n2^k-1$ 형태이면 residue weight $|2^j|_{m_i}$ 는 반복 형태로 나타난다. 그림 2는 입력 변환회로의 한 예이다. 전술한 최적 moduli는 아니지만 회로 구성의 예를 간단히 표현할 수 있으므로 이하 모든 구성은 Mod 7 으로 한다.

그림 3은 그림 2의 decoder를 표시한다. Decoder는 weight가 같은 것끼리 받아들여 십진하중 residue 를 산출한다.

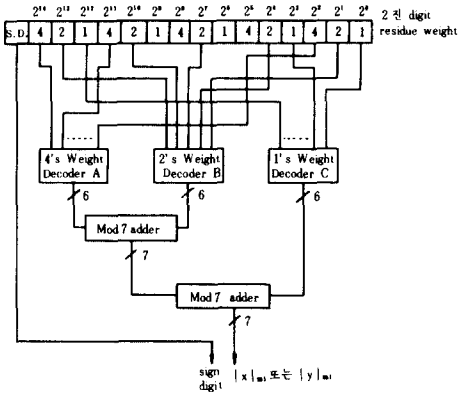

그림 2. 입력변환(Mod 7)

Fig. 2. Input translation (Mod 7).

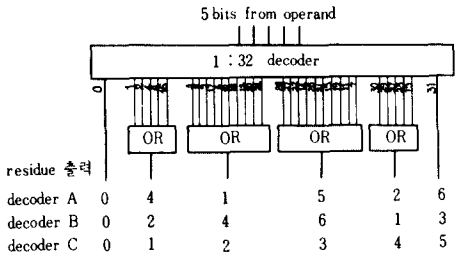

그림 3. 디코더 구성

Fig. 3. Decoder scheme.

각각의 하중에 대응하는 decoder의 출력은 표 1과 같은 기능을 지닌 그림 4의 Modulus m_i adder로 더한다. 표 1에서 decoder 출력 A와 B의 residue 합을

표 1. 모듈러스 m_i 덧셈(Mod 7)

Table 1. Modulus m_i addition (Mod 7).

A \ B	0	1	2	3	4	6
0	0	1	2	3	4	6
1	1	2	3	4	5	0
2	2	3	4	5	6	1
4	4	5	6	0	1	3
5	5	6	0	1	2	4
6	6	0	1	2	3	5

C \ A + B	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	6
2	2	3	4	5	6	0
3	3	4	5	6	0	1
4	4	5	6	0	1	2
5	5	6	0	1	2	3
6	6	0	1	2	3	4

$|x|_{m_i}$ 또는 $|y|_{m_i}$

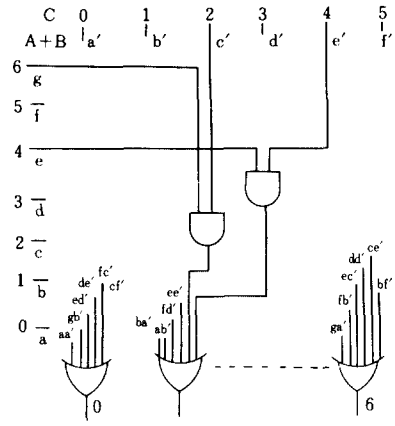

그림 4. 모듈러스 7 adder

Fig. 4. Modulus 7 adder.

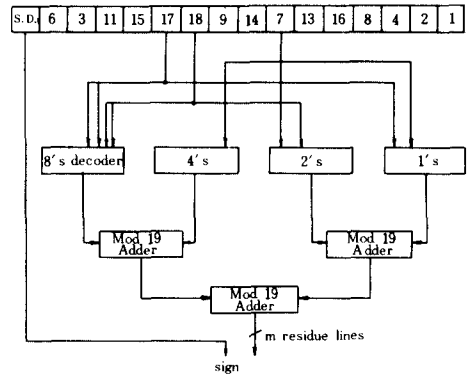

그림 5. 입력변환(Mod 19)

Fig. 5. Input Translation (Mod 19).

먼저 구한 다음 C의 residue를 다시 합해 입력 변환이 수행된다.

만일 modulus 값이 $n2^k \pm 1$ 형태가 아니거나 weight값이 큰 경우에는 그림5와 같이 분리한다.

2. Residue 곱셈

(4)식을 다시 쓰면,

$$|Z|_{m_i} = |x|_{m_i} |y|_{m_i} = ||x|_{m_i} |y|_{m_i}|_{m_i} \quad (7)$$

표2는 Mod 7 일때 (7)식의 곱셈 결과를 나타낸다.

표 2. 곱셈표(Mod 7)

Table 2. Multiplication table (Mod 7).

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

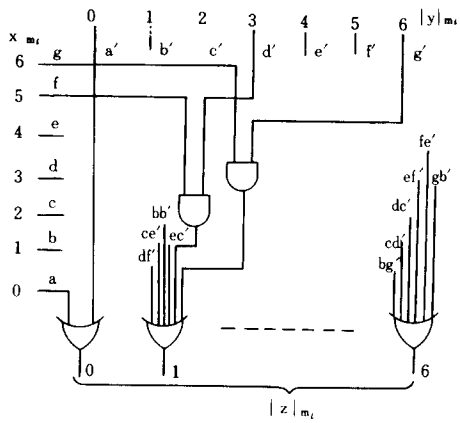

그림 6. Residue 곱셈 구성도(Mod 7)

Fig. 6. Residue multiplication scheme (Mod 7).

표2의 기능을 회로화하면 그림6과 같다.

그림5로부터 곱셈기는 $(m_i - 1)^2$ 개의 AND 게이트와 m_i 개의 OR 게이트로 일반화할 수 있다. 그러나 m_i 가 커지면 전체 element수와 특히 OR게이트 fan-in 수가 많아짐은 자명하다.

$|Z|_{m_i}$의 부호 결정은 XOR로 실현 가능하다.

3. 출력 변환

$|Z|_{m_i}$의 역변환 Z는 mixed radix conversion (MRC)[19]과 Chinese Remainder Theorem(CRT) 에 의해 구할 수 있다. MRC 는 moduli 값이 작고 갯수가 많은 경우에 유용하나 residue의 mixed radix 변환에는 특별 하드웨어가 필요하고 반복 형태때문에 속도가 느리다.[14][24] CRT는 2진 출력에 유용하다.

출력 Z는 $Z = |xy|_M = |x|_M |y|_M$ 따라서

$$Z_{RRNS} = \sum_{i=1}^{n} |x|_{m_i} |y|_{m_i} N_i \alpha_i - P_z M \quad (8-1)$$

$$Z_{NRNS} = |\sum_{i=1}^{n} |x|_{m_i} |y|_{m_i} N_i \alpha_i|_M \quad (8-2)$$

와 같다(부록 I). (8-1)식은 다음 과정에 의한다.

1) $\sum_{i=1}^{n} |x|_{m_i} |y|_{m_i} N_i \alpha_i$ 의 출력

6개 modulus에 대해 각각 그림7처럼 $|x|_{m_i} |y|_{m_i} N_i \alpha_i$를 30bit 2진 형태로 access한다.

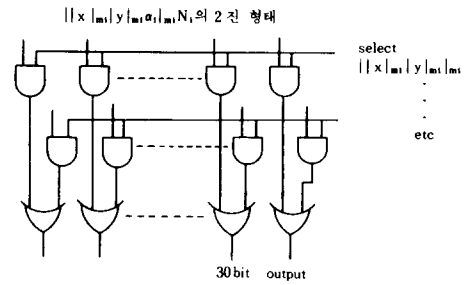

그림 7. $|x|_{m_i} |y|_{m_i} N_i \alpha_i$ access

Fig. 7. $|x|_{m_i} |y|_{m_i} N_i \alpha_i$ access.

2) $P_z M$ 출력

p_z는 모든항 $|x|_{m_i} N_i \alpha_i$, $|y|_{m_i} N_i \alpha_i$와 P_x, P_y의 함수이므로 $P_z \neq P_x P_y$이다. 따라서 P_z를 구하기 위해 (8-1)식을 Base Extension (B. E)[25]하면

$$P_z = |\sum_{i=1}^{n} |x|_{m_i} |y|_{m_i} \alpha_i \mu_i + z_{n+1} \mu_S|_{m_{n+1}} \quad (9)$$

여기서 m_{n+1} : B. E. modulus

$\mu_i = |\frac{1}{m_i}|_{m_{n+1}}$: multiplicative inverse

$\mu_S = |\frac{-1}{M}|_{m_{n+1}}$: super modulus

$z_{n+1} = |z|_{m_{n+1}}$: B. E. residue

(9)식을 구하기 위해서

① $|\mu_i| |x|_{m_i} |y|_{m_i} \alpha_i|_{m_i}|_{m_{n+1}}$을 구성한다. (부록 II)

② $|z_{n+1}\mu_s|_{m_{n+1}}$의 table을 구성한다.

③위 과정이 동시 수행된 후 CC에 의해 ①, ②항을 더한 결과에 Mod m_{n+1}을 취한다. 이때 $m_{n+1}=2^k$ 형태이면 residue변환에 추가 logic과 시간이 필요치 않다. M은 기지의 값이므로 P_zM은 hardwired access로 구한다(그림7).

3) 최종 출력변환

(i)과 (ii)에서 도합 7개의 2진 형태 30bit 라인이 동시 얻어지면 z는 c. c[23]로 구해진다(그림8).

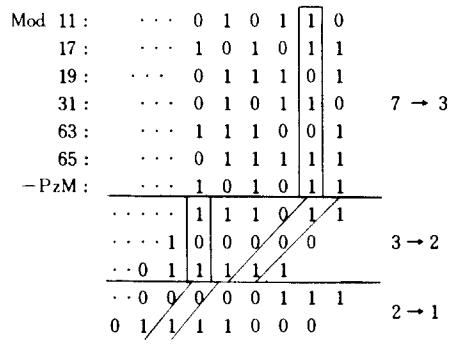

그림 8. 칼럼 컴프레션

Fig. 8. Column compression.

Ⅳ. 연산속도 추정

표3은 ECL의[28] 최대 게이트 delay를 기준으로 한 RRNS 및 NRNS의 연산 추정시간이다. 선택한 moduli에 대해 각 연산과정의 레벨수는 예로 든 mod 7인 경우와 같고 또 각 modulus마다 연산이 병렬로 동시 수행되므로 입력 변환에는 그림3의 decoder에 3.6, ORing에 0.95, modulo adder는 2레벨이며, 그림4에서 2.05므로 4.1, 도합 8.65ns로 추정된다. Residue 곱셈은 그림6에서 2.05ns이고 p_z과정에는 부록2의 테이블 구현은 곱셈과정과 동일하므로 2.05, 7→3,[23] 3→2 카운터에 5.5, 2→1 카운터는 $m_{n+1}=64$므로 6bit binary adder로 가능하므로 4.15, P_zM select에 1.1, 총 18.3ns로 추정된다. RRNS의 경우 P_z과정에 총 시간의 약 36%가 소요된다. 또한 출력변환에는 7→3, 3→2 카운터에 5.5, 2→1 카운터는 30bit carry lookahead adder므로 10, 총 21ns로 추정된다.

표 3. 추정시간

Table 3. Estimated timing.

곱셈과정		RRNS	NRNS
입력변환	디코더	3.60	3.60
	ORing	0.95	0.95
	adder	4.1	4.1
	시간소계	8.65	8.65
Residue 곱셈		2.05	2.05
P_z		18.3	
	시간소계	18.3	2.05
출력변환	7→3 카운터	5.5	5.5
	3→2	5.5	5.5
	2→1	10	10
	시간소계	21	21
총 시 간		47.95ns	31.7ns

Ⅴ. 結 論

연산속도 개선을 목적한 RNS 고속 곱셈기의 이론적 구성을 시도했다. 오퍼랜드 wordlength가 증가하여도 carry없이 고속 곱셈이 가능하였고, RRNS는 magnitude index를 이용한 CRT로써 출력 변환하였다. 곱셈에 소요되는 추정 연산시간은 파이프라인없이 RRNS, NRNS 각각 약 47.95(ns) 및 31.7(ns)로 taylor방법보다 352ns, 368ns 개선되었다. 조합논리회로의 사용으로 게이트 수는 증가하나 RRNS 경우는 $|z|_{m_i}$를 출력 변환전에 다른 연산에 이용할 수 있으므로 limit cycle oscillation이 적은 디지탈 필터링, beamforming 등에 유용한 특수 복소수 곱셈기에 확장 가능하다. 이 점에 대해선 다음 과제로 남긴다.

附 錄

1. CRT와 오퍼랜드

선형합동(linear congruence) 연립방정식

$$x \equiv |x|_{m_i} (\text{mod } m_i) \quad i=1,2,\cdots\cdots,n \qquad (A-1)$$

해는 다음과 같은 CRT이다.[26]

$$x=\sum_{i=1}^{n} |x|_{m_i} N_i \alpha_i \qquad (A-2)$$

y의 경우도 동일하다.

여기서 $N_i=\dfrac{\text{range}}{\text{modulus}}=M/m_i$

$\alpha_i=\left|\dfrac{1}{N_i}\right|_{m_i}$: multiplicative inverse

$0<x<M$ (non-redundant system) 면 A식은

$$|x|_M=\left|\sum_{i=1}^{n} |x|_{m_i} N_i \alpha_i\right|_M$$

$$x=\left|\sum_{i=1}^{n} |x|_{m_i} N_i \alpha_i\right|_M \qquad (A-3)$$

y의 경우도 동일하다.

x>M (redundant system) 면 A식은

$$|x|_M = \left|\sum_{i=1}^{n} |x|_{m_i} N_i \alpha_i\right|_M$$

$$= \sum_{i=1}^{n} |x|_{m_i} N_i \alpha_i - PxM \qquad (A-4a)$$

$$|y|_M = \sum_{i=1}^{n} |x|_{m_i} N_i \alpha_i - P_y M \qquad (A-4b)$$

여기서 P_x, P_y는 magnitude index이다.

2. $|\mu_i||x|_{m_i}|y|_{m_i}\alpha_i|_{m_i}|_{m_{n+1}}$의 구성

$|x|_{m_i}|y|_{m_i}$를 이용 $||x|_{m_i}|y|_{m_i}\alpha_i|_{m_i}$ 구한 다음 다시 $|\mu_i||x|_{m_i}|y|_{m_i}\alpha_i|_{m_i}|_{m_{n+1}}$을 구한다. 하드웨어 구성은 곱셈과정과 동일하다.

Mod 7, $\alpha_i = 5$, $\mu_i = 3$, $m_{n+1} = 5$인 예

0	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	5	3	1	6	4	2
2	0	3	6	2	5	1	4
3	0	1	2	3	4	5	6
4	0	6	5	4	3	2	1
5	0	4	1	5	2	6	3
6	0	2	4	6	1	3	5

⟹

0	0	0	0	0	0	0
0	0	4	1	3	2	1
0	4	3	1	0	1	2
0	1	1	4	2	0	3
0	3	0	2	4	1	1
0	2	1	0	1	3	4
0	1	2	3	1	4	0

$||x|_{m_i}|y|_{m_i}\alpha_i|_{m_i}$ $\quad$ $|\mu_i||x|_{m_i}|y|_{m_i}\alpha_i|_{m_i}|_{m_{n+1}}$

3. Moduli Set

구해진 최적 moduli set에 대한 α_i, N_i, $N_i\alpha_i$를 각각 나타낸다.

RRNS

modulus	$N=M/m_i$	α_i	$N_i\alpha_i$
11	41003235	5	205016575
17	26531505	4	106126020
19	23738715	2	23738715
31	14549535	6	87297210
63	7159295	5	35796475
65	6939009	64	444096576

NRNS

modulus	$N_i=M/m_i$	L_i	N_iL_i
17	414863328	6	2489179968
19	371193504	5	1855967520
23	306638112	17	5212847904
29	243195744	22	5350306368
31	227505696	12	2730068352
32	220391644	15	3305874645
33	213717472	1	213717472

參 考 文 獻

[1] Y.S. Wu, "Architectural considerations of signal processor under microprogram control," *AFIPS Spring Joint Comput. Conf. Proc.*, vol. 40, pp. 675-683, May 1972.

[2] J. Allen, "Computer architecture for signal processing, *"Proc. IEEE,* vol. 63, pp. 624-633, Apr. 1975.

[3] L.R. Rabiner, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, N.J., Prentice Hall, 1975.

[4] K. Hwang, *Computer Arithmetic.* John Wiley & Sons, New York, 1979.

[5] I. Flores, *The Logic of Computer Arithmetic.* Englewood Cliffs, N.J., Prentice Hall, 1963.

[6] C.S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 14-17, Feb. 1964.

[7] L. Dadda, "Some schemes for parallel multipliers, *"Alta Freq.*, vol. 34, pp. 349-356, May 1965.

[8] W.J. Stenzel, "A compact high-speed parallel multiplication scheme," *IEEE Trans. Comput.*, vol. C-26, no. 10, Oct. 1977.

[9] H.C. Lai, "Logic networks of carry save adders, *"IEEE Trans. Comput.*, vol. C-31, no. 9, Sept. 1982.

[10] S.D. Pezaris, "A 40 ns 17-bit array multiplier," *IEEE Trans. Comput.*, vol. C-20, pp. 442-447, Apr. 1971.

[11] D.P. Agrawal, "High-speed arithmetic array," *IEEE Trans. Comput.*, vol. C-28, no. 3, pp. 215-224, March 1979.

[12] A. Svoboda, "The numerical of residual classes in mathematical machines," *Information Processing, Proc. UNESCO Conf.*, pp. 419-422, 1960.

[13] H. Aiken, W. Semon, *Advanced Digital Logic.* Wright Air Development Center, Tech. Report WADC TR-59-472, July 1959.

[14] N.S. Szabo, R.I. Tanaka, *Residue Arithmetic and its Applications to Computer Technology.* New York, McGraw-Hill, 1967

[15] G.A. Jullien, "Residue number scaling and other operations using ROM arrays," *IEEE Trans. Comput.*, vol. C-27, no. 4, pp. 325-337, Apr. 1978.

[16] C.H. Huang, F.T. Taylor, "Memory compression scheme for modular arithmetic," *IEEE Accoustics, Speech and Signal Processing,* vol. ASSP-27, Dec. 1979.

[17] M.A. Soderstrand, "A high-speed low-cost recursive filter using residue arithmetic," *Proc. IEEE,* vol. 65, pp. 1065-1067, July 1977.

[18] W.K. Jenkins, B.J. Leon, "The use of residue number systems in the design of finite impulse response digital filters," *IEEE Trans. Circuits Syst.,* vol. CAS-24, no. 4, pp. 191-201, Apr. 1977.

[19] W.K. Jenkins, "A highly efficient residue-combinatorial architecture for digial filters," *Proc. IEEE,* vol. 66, no. 6, pp. 700-702, June 1978.

[20] F.J. Taylor, "A VLSI residue arithmetic multiplier," *IEEE Trans. Comput.,* vol. C-31, no. 6, June 1982.

[21] A. Sasaki, "Addition and subtraction in the residue system," *IEEE Trans. Comput.,* vol. EC-16, no 2, Apr. 1967.

[22] R.N. Rao, "Binary logic for residue arithmetic using magnitude index," *IEEE Trans. Comput.,* vol. C-19, no. 8, Aug. 1970.

[23] I.T. Ho, T.C. Chen, "Multiple addition by residue threshold functions and their representation by array logic," *IEEE Trans. Comput.,* vol. C-22 no. 8, pp. 762-767, Aug. 1973.

[24] R.D. Merrill, "Improving digital computer performance using residue number theory," *IEEE Trans. Comput.,* vol. EC-13, pp. 93-101, Apr. 1964.

[25] K.H. O'Keefe, "Remarks on base extention for modular arithmetic," *IEEE Trans. Comput.,* vol. C-22, no. 9, pp. 833-835, Sept. 1973.

[26] D.M. Burton, *Elementary Number Theory.* Allyn & Bacon, Boston, Mass., 1976.

[27] D.K. Banerji, J.A. Brzozowski, "On the translation algorithms in residue number system," *IEEE Trans. Comput.,* vol. C-21, no. 12, pp. 1281-1285, Dec. 1972.

[28] Fairchild, *F100k ECL Data Book,* 1982.

[29] Jae K. Kim, *Multiplier for Digital Signal Processing.* Lough-borough Univ., England, 1981.