2010

Optimization of new Chinese Remainder theorems using special moduli sets

Narendran Narayanaswamy
Louisiana State University and Agricultural and Mechanical College

OPTIMIZATION OF NEW CHINESE REMAINDER THEOREMS USING SPECIAL MODULI SETS

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
Requirements for the degree of
Master of Science in Electrical Engineering
in
The Department of Electrical and Computer Engineering

by
Narendran Narayanaswamy
B.Tech, Amrita School of Engineering, India 2008
December 2010

ACKNOWLEDGMENTS

I sincerely thank Dr. Alex Skavantzos for his constant support and guidance throughout this research work. He was very patient in listening to the problems occurred and provided invaluable help in the entire course of work. I specially thank him for the amount of work he has done for editing and commenting on an accepted IEEE conference paper and editing this report.

I am indebted to Dr. Ashok Srivastava and Dr. Juana Moreno for accepting my request to be on the advisory committee. I also thank Dr. Thanos Stouraitis for spending his most valuable time in presenting our accepted paper to an IEEE conference.

I would like to thank Hortensia T. Valdes and Dr. Thomas Kutter for providing finance for my entire Masters studies. I gained professional experience from both of them, which will be an excellent start for my forthcoming professional career. Hortensia cared about me like no one else would, and I have learned many valuable lessons from her through our interactions and conversations.

Finally, I thank my parents, relatives and all my friends who have helped me in making this thesis a successful one.

TABLE OF CONTENTS

Acknowledgements……..ii

List of Figures……..v

Abstract……...vi

1. Introduction To Residue Number Systems……....1
   1.1 Introduction…...……..……..……..……..
   1.2 Fundamentals of Residue Number System ……...1
      1.2.1. Weighted to RNS Conversion ……1
      1.2.2. Arithmetic Operations in RNS Domain ……..……2
      1.2.3. Dynamic Range of the RNS……...……..2
      1.2.4. RNS to Weighted Conversion ……..……..……..3
         1.2.4.1. Chinese Remainder Theorems (CRT) ……....3
         1.2.4.2. Mixed Radix Conversion (MRC) ……..…..4

2. New Chinese Remainder Theorems…….……..7
   2.1. New Chinese Remainder Theorem I (CRT I)……..……7
      2.1.1. Traditional Hardware Implementation for CRT I …….....10
   2.2. New Chinese Remainder Theorem II (CRT II)……..10
      2.2.1. Traditional Hardware Implementation for CRT II……...14
   2.3. New Chinese Remainder Theorem III (CRT III)……...……....15
      2.3.1. Requirements for the Efficient Implementation of CRT III ……...…….17
      2.3.2. Decoding Conjugate Pairs of Moduli ……...…....18
         2.3.2.1. Decoding Two Conjugate Pair Moduli…….……..…18
         2.3.2.2. Decoding Three Conjugate Pair Moduli…….……...19
      2.3.3. Traditional Hardware Implementation for CRT III……...……..…...….....20

3. Optimizing New Chinese Remainder Theorems…….....22
   3.1. Shortfalls of New Chinese Remainder Theorems……...22
   3.2. Basic Lemmas Used for the Optimization ……30
   3.3. Optimizing the Base Theorem for New Chinese Remainder Theorem I……....31
   3.4. Optimizing New Chinese Remainder Theorem II……...…...……33
   3.5. Optimizing New Chinese Remainder Theorem III…….35
      3.5.1. Extending the Optimized Four Moduli Set to A Five Moduli Set…….....38

4. Optimized Hardware Implementation of New Chinese Remainder Theorems ……..…41
   4.1. Introduction……41
   4.2. Optimized Hardware Implementation of the Base Theorem for New Chinese Remainder Theorem I…….……...……41
   4.3. Optimized Hardware Implementation of New Chinese Remainder Theorem II.....43
   4.4. Optimized Hardware Implementation of New Chinese Remainder Theorem III.....46
   4.5. Comparison between Optimized and Non-Optimized CRT II Implementations…....47
   4.6. Comparison between Optimized and Non-Optimized CRT III Implementations.....48

5. Conclusion and Future Work ……..……..……..50

References……..…51

Appendix A: Inverse Modulus Operation…….....53
A.1. Methods to Calculate Inverse Modulus Values…….....53
A.2.Code in C to Calculate Multiplicative Inverse……...…58

Appendix B: Proofs for the Sets to Be Pair Wise Relatively Prime……..60
B.1.The Set $\{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$, where n = 2k + 1, k = 1, 2, 3… Is Pair Wise Relatively Prime…….……..……..60
B.2.The Set $\{2^{n+2}+3, 2^{n+1}+1, 2^n+1, 2\}$, where n =1, 2, 3…Is Pair Wise Relatively Prime…….……60

Vita……...……..62

LIST OF FIGURES

Fig.2.1. Hardware assembly for calculating X using CRT I ……..…11

Fig.2.2. Implementation of *findno* procedure……12

Fig.2.3. Hardware assembly for calculating N_1 and N_2 using CRT II …….....15

Fig.2.4. Hardware assembly for calculating X using CRT II.……16

Fig.2.5. Traditional hardware implementation of CRT III..……..21

Fig.4.1.a. Hardware implementation of Q_1 by CRT I without using multipliers……...42

Fig.4.1.b. Hardware implementation of Q_2 by CRT I without using the multipliers…….42

Fig.4.1.c. Hardware implementation of Q_3 by CRT I without using the multipliers…….43

Fig.4.1.d. Hardware implementation of X by CRT I without using the multipliers…….…….43

Fig.4.2.a. Hardware implementation of N_1 by CRT II without using multipliers …...…….45

Fig.4.2.b. Hardware implementation of N_2 by CRT II without using multipliers …...…….45

Fig.4.3. Hardware implementation of X by CRT II without using multipliers……..….....46

Fig.4.4. Hardware implementation of X by CRT III without using multipliers…….….....48

ABSTRACT

The residue number system (RNS) is an integer number representation system, which is capable of supporting parallel and high-speed arithmetic. This system also offers some useful properties for error detection, error correction and fault tolerance. It has numerous applications in computation-intensive digital signal processing (DSP) operations, like digital filtering, convolution, correlation, Discrete Fourier Transform, Fast Fourier Transform, direct digital frequency synthesis, etc.

The residue to binary conversion is based on Chinese Remainder Theorem (CRT) and Mixed Radix Conversion (MRC). However, the CRT requires a slow large modulo operation while the MRC requires finding the mixed radix digits which is a slow process. The new Chinese Remainder Theorems (CRT I, CRT II and CRT III) make the computations faster and efficient without any extra overheads. But, New CRTs are hardware intensive as they require many inverse modulus operators, modulus operators, multipliers and dividers. Dividers and inverse modulus operators in turn needs many half and full adders and subtractors. So, some kind of optimization is necessary to implement these theorems practically.

In this research, for the optimization, new both co-prime and non co-prime multi modulus sets are proposed that simplify the new Chinese Remainder theorems by eliminating the huge summations, inverse modulo operators, and dividers. Furthermore, the proposed hardware optimization removes the multiplication terms in the theorems, which further simplifies the implementation.

1. INTRODUCTION TO RESIDUE NUMBER SYSTEMS

1.1. Introduction:

Residue number system is an integer system which is capable of supporting parallel, carry-free and high speed arithmetic. This system also offers some useful properties for error detection, error correction and fault tolerance in digital systems. It is very efficient in carrying out arithmetic operations like additions, subtractions and multiplications. The speed of the arithmetic operations relies on the size of the numbers involved; smaller numbers imply faster operations. Since, the numbers used in this system are smaller, it is known for faster implementation of arithmetic operations, and hence it is very attractive. This system is applied in the fields of Digital Signal Processing (DSP) intensive computations like digital filtering, convolutions, correlations, Discrete Fourier Transform (DFT) computations, Fast Fourier Transform (FFT) computations and direct digital frequency synthesis.

1.2. Fundamentals of Residue Number System:

A Residue Number System is defined by a set S = {m_1, m_2, m_3,…, m_L}and this set is called the moduli set, where m_1, m_2, m_3, …, m_L are positive integers. Earlier, the important criteria of moduli set is that the elements are pair wise relatively prime,

$$S = \{m_1, m_2, m_3 \ldots m_L\}, \ (m_i, m_j) = 1 \text{ for } i \neq j \qquad (1.1)$$

where (m_i , m_j) is the greatest common divisors of m_i and m_j. The works of [2] and [4] clearly show that the moduli set need not be pair wise relatively prime. This gives rise to wide choices of moduli sets and makes the RNS system efficient and attractive.

1.2.1. Weighted to RNS Conversion:

Let Z_M be {0, 1, 2… M-1), where M is the product of all moduli in the set S. Any integer X belonging to Z_M has a unique representation,

$$X \xrightarrow{RNS} (X_1, X_2, X_3, \ldots, X_L) \quad (1.2)$$

where,

$$X_i = X \bmod m_i \ , \ i = 1, 2, \ldots, L \quad (1.3)$$

The above equations (1.2) and (1.3) show the weighted to RNS conversion.

1.2.2. Arithmetic Operations in RNS Domain:

Let X and Y belong to $Z_{M,}$ where Z_M= {0, 1, 2… M-1), while M is the product of all moduli in the set S. Let X and Y have RNS representations as follows,

$$X \xrightarrow{RNS} (X_1, X_2, X_3 \ldots X_L) \quad (1.4)$$

$$Y \xrightarrow{RNS} (Y_1, Y_2, Y_3 \ldots Y_L) \quad (1.5)$$

Then Z = X ɵ Y is given in the RNS as follows:

$$Z = X \text{ ɵ } Y \xrightarrow{RNS} ((X_1 \text{ ɵ } Y_1) \bmod m_1 , (X_1 \text{ ɵ } Y_1) \bmod m_2 , \ldots , (X_1 \text{ ɵ } Y_1) \bmod m_L)) \quad (1.6)$$

where ɵ denotes addition, subtraction or multiplication. Equation (1.6) clearly shows that the nature of the implementations of RNS operations is carry free and parallel.

1.2.3. Dynamic Range of the RNS:

Let M be the product of all the moduli in the moduli set. Then if the system supports only unsigned numbers, the dynamic range is

$$DR = [0 \quad M-1] \quad (1.7)$$

If the system supports signed numbers then the dynamic range is

$$DR = [-M/2 \quad (M/2) - 1], \text{ if M is even} \quad (1.8)$$

$$DR = [- (M-1)/2 \quad (M-1)/2], \text{ if M is odd} \quad (1.9)$$

1.2.4. RNS to Weighted Conversion:

Traditionally, there are two basic techniques for converting the Residue Numbers System into the weighted system. The two techniques are:

1) Chinese Remainder Theorem (CRT)
2) Mixed Radix Conversion (MRC)

1.2.4.1. Chinese Remainder Theorem (CRT):

Consider the moduli set S = {m_1, m_2, m_3… m_L}, and let the RNS representation of an integer X be (X_1, X_2, X_3… X_L).

Then the Chinese Remainder Theorem reconstructs X from its residues as follows,

$$X = (X_1 M_1 N_1 + X_2 M_2 N_2 + \ldots + X_L M_L N_L) \bmod M \tag{1.10}$$

Alternately, we can have

$$X = (\sum_{i=1}^{L} X_i N_i M_i) \bmod M \tag{1.11}$$

where,

$$M = \prod_{i=1}^{L} m_i \tag{1.12}$$

$$M_i = M / m_i \quad \text{and} \quad N_i = (M_i^{-1}) \bmod m_i \tag{1.13}$$

Illustration 1:

Consider an RNS system defined by the set S= {m_1, m_2, m_3} = {15, 16, 17}. Here the value of M = ($m_1 * m_2 * m_3$) = 4080.

For an unsigned system, the dynamic range is [0 M-1], and hence here dynamic range is [0 4079].

For instance, let the two weighted numbers be X = 55 and Y = 58. Then the multiplication (Z = X * Y) is carried out as follows,

From (1.2) and (1.3), X = (X_1, X_2, X_3) = (10, 7, 4) and Y = (Y_1, Y_2, Y_3) = (13, 10, 7).

From (1.6), Z = X * Y = (($X_1 * Y_1$) mod m_1 , ($X_1 * Y_1$) mod m_2 , ($X_1 * Y_1$) mod m_3))

= (10, 6, 11)

Conversion of Z = (10, 6, 11) into the weighted system using the CRT is as follows:

From (1.12) and (1.13), M = 4080, M_1 = 272, N_1 = 8

M_2 = 255, N_2 = 15

and M_3 = 240, N_3 = 9

Using the CRT equation,

$$Z = (\sum_{i=1}^{3} (Z_i N_i M_i) \bmod M \tag{1.14}$$

Z = 3190

For verification, Z = X * Y = (55 * 58) = 3190.

1.2.4.2. Mixed Radix Conversion (MRC):

For the moduli set $S = \{m_1, m_2, m_3 \ldots m_L\}$, $(m_i, m_j) = 1$ *for* $i \neq j$, and the RNS representation of $X \xrightarrow{\text{RNS}} (X_1, X_2, X_3 \ldots X_L)$, X can be calculated from its residues using the Mixed Radix Conversion formula as shown below,

$$X = X_1' + m_1X_2' + m_1m_2X_3' + m_1m_2m_3X_4' + \ldots + m_1m_2m_3m_4 \ldots m_{L-1}X_L' \tag{1.15}$$

where X_1', X_2', X_3', …, X_L' are called the mixed radix digits and X_i' belongs to Z_{m_i} = 0, 1, 2, …, $m_i - 1$. The mixed radix digits X_1', X_2, $X_3' \ldots X_L'$ can be represented as functions of the residues $X_1, X_2, X_3, \ldots, X_L$ and the moduli $\{m_1, m_2, m_3, \ldots, m_L\}$. These mixed radix digits have respective weights associated with them. The weight associated with X_1' is 1,

X_2' is m_1, X_3' is m_1m_2, and similarly the weight associated with X_L' is $m_1m_2m_3m_4...m_{L-1}$. These mixed radix digits can be calculated as shown.

Consider a 4 – moduli system, which can be easily generalized to an arbitrary L – moduli system. Then the MRC formula is

$$X = X_1' + m_1X_2' + m_1m_2X_3' + m_1m_2m_3X_4' \quad (1.16)$$

Taking (mod m_1) on both sides of the above equation yields,

$$X \bmod m_1 = X_1' \bmod m_1 \quad (1.17)$$

We know, $X_1' \varepsilon Z_{m1,}$ therefore $X_1' \bmod m_1 = X_1'$ (1.18)

Hence $X_1' = X_1$ *(Since* $X \bmod m_1 = X_1$*)* (1.19)

For the calculation of X_2', consider (1.16) again, we get

$$X - X_1' = m_1X_2' + m_1m_2X_3' + m_1m_2m_3X_4'$$

Taking inverse modulus of m_1 on both sides, we get

$$m_1^{-1}(X - X_1') = X_2' + m_2X_3' + m_2m_3X_4'$$

Taking both sides mod m_2, we get,

$$[m_1^{-1}(X - X_1')] \bmod m_2 = X_2' \bmod m_2 + (m_2X_3' + m_2m_3X_4') \bmod m_2$$

Hence $X_2' = [m_1^{-1}(X_2 - X_1')] \bmod m_2$ (1.20)

For the calculation of X_3', consider (1.16) again. We get

$$X - X_1' - m_1X_2' = m_1m_2X_3' + m_1m_2m_3X_4'$$

Taking inverse modulus of m_1m_2 on both sides, and applying mod m_3, we get,

$$[(m_1m_2)^{-1}(X - X_1' - m_1X_2')] \bmod m_3 = [X_3' \bmod m_3 + (m_1m_2m_3X_4')] \bmod m_3 \quad (1.21)$$

Hence $X_3' = ((m_1m_2)^{-1}(X_3 - X_1' - m_1X_2') \bmod m_3$

Similarly X_4' can be calculated as $((m_1m_2m_3)^{-1}(X_4 - X_1' - m_1X_2' - m_1m_2X_3')) \bmod m_4$ (1.22)

Illustration 2:

Consider an RNS system defined by the moduli set S = {7, 9, 11, 13,}.Let the weighted number X be1478.

The RNS representation of X is, X = (X_1, X_2, X_3, X_4) = (1, 2, 4, 9). Using the Mixed Radix Conversion technique, the above RNS can be converted into weighted number. From the equations (1.19), (1.20), (1.21) and (1.22), the values of X_1', X_2', X_3' and X_4' can be calculated as 1, 4, 1, 2 respectively. Finally, the MRC equation (1.16) gives the value of X as 1478.

2. NEW CHINESE REMAINDER THEOREMS

The speed of the arithmetic operations is based on the numbers involved in the operations. The size of the numbers is directly proportional to the delay of the operations, and therefore smaller numbers imply faster operations. However, the Chinese Remainder Theorem requires a slow large modulo operation while the Mixed Radix Conversion requires finding the mixed radix digits which is a slow process. New Chinese Remainder Theorems were designed to make the computations faster and efficient without any overheads.

2.1. New Chinese Remainder Theorem I (CRT I):

New Chinese Remainder theorem I (CRT I) is a modified version of the traditional Chinese Remainder Theorem [1]. In this conversion process, the weighted number can be retrieved faster because the operations are done in parallel, without depending on other results.

Some propositions proposed in [1] are necessary for this new conversion and are as follows,

Proposition 1:

$$a = 1 \bmod (m_1 m_2) \text{ implies } a = 1 \bmod m_1 \text{ and } 1 \bmod m_2 \tag{2.1}$$

The above proposition is obtained from the corollary,

$$a = 1 \bmod (m_1 m_2 \ldots m_k) \text{ implies } a = 1 \bmod m_1,\ a = 1 \bmod m_2 \ldots,\ a = 1 \bmod m_k \tag{2.2}$$

Proposition 2:

$$[a m_1] \bmod (m_1 m_2) = [a] \bmod m_2 * m_1 \tag{2.3}$$

Proposition 3:

For any y belonging to $[0, M-1]$, where $M = m_1 * m_2, \ldots, m_{n-1} * m_n$, there is unique mixed radix representation as follows, where y_i satisfies the condition $0 \leq y_i \leq m_{i+1}$.

$$y = y_0+y_1m_1+y_2m_1m_2+ \ldots + y_{n-1}m_1m_2 \ldots m_{n-1} \quad (2.4)$$

In the above equation, y_{n-1} is the most significant digit and y_0 is the least significant digit.

Given the residue numbers (X_1, X_2, $X_3 \ldots X_n$), the corresponding weighted number X can be computed using the following equation.

$$X = [X_1 + k_1m_1(X_2-X_1) + k_2m_1m_2(X_3 - X_2) + ... + k_{n-1}m_1m_2 \ldots m_{n-1}(X_n - X_{n-1})] \bmod m_1m_2 \ldots m_{n-1}m_n \quad (2.5)$$

where $k_1 = (m_1)^{-1} mod m_2m_3...m_n$, $k_2 = (m_1m_2)^{-1} \bmod m_3m_4 \ldots m_n$ and similarly

$k_{(n-1)} = (m_1m_2 \ldots m_{n-1})^{-1} \bmod m_n$

It is to be noted that, the above equation is different from the traditional CRT and MRC equations.

Proof for the above equation:

From the proposition 1, we know

$$k_1 = (m_1)^{-1} \bmod m_2$$

$$k_2 = (m_1m_2)^{-1} \bmod m_3$$

Extending the above to the larger dimensions, we get

$$k_1 = (m_1)^{-1} \bmod m_2 \text{ , } k_2 = (m_1m_2)^{-1} \bmod m_3 \text{ , } \ldots \text{ , } k_{(n-1)} = (m_1m_2m_3 \ldots m_{n-1})^{-1} \bmod m_n$$

Henceforth, $X = X_1 \bmod m_1$

$$X = \{X_1 \bmod m_2 + (k_1m_1) \bmod m_2 * (X_2-X_1)\} = X_2 \bmod m_2$$

$$X = \{X_1 \bmod m_3 + (k_1 m_1) \bmod m_2 * (X_2 - X_1) + (k_2 m_1 m_2) \bmod m_3 * (X_3 - X_2)\}$$

$$= X_3 \bmod m_2$$

Similarly, extending the above to the larger dimensions,

$$X = \{X_1 \bmod m_n + (k_1 m_1) \bmod m_n * (X_2 - X_1) + (k_2 m_1 m_2) \bmod m_n * (X_3 - X_2) + \ldots$$

$$+ (k_{n-1} m_1 m_2 \ldots m_{n-1}) \bmod m_n * (X_n - X_{n-1})\} \bmod m_n$$

$$= \{X_1 - (X_2 - X_1) + (X_3 - X_2) + \ldots + (X_n - X_{n-1})\} \bmod m_n$$

$$= X_n$$

This conversion process enables the parallel processing of the mixed radix digits because the digits are given based on the co-ordinates X_i as in the CRT. Hence this conversion overcomes the bottleneck of using traditional CRT and MRC techniques.

Illustration 1:

Consider a weighted number X = 2011 and the moduli set of the form $\{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$ where n=2k+1, k = 1,2,3,…Taking the value of n = 3, we get the moduli set m = (m_1, m_2, m_3, m_4) = (5, 7, 8, 9).

The proof for the set $\{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$ to be pair wise relatively prime is dealt in the appendix B (Section B.1).

RNS representation of X = (X_1, X_2, X_3, X_4)

= (X mod m_1, X mod m_2, X mod m_3, X mod m_4)

= (1, 2, 3, 4)

$k_1 = (m_1)^{-1} \bmod (m_2m_3m_4) = 5^{-1} \bmod (7 * 8 * 9) = 101$

$k_2 = (m_1m_2)^{-1} \bmod (m_3m_4) = (5 * 7)^{-1} \bmod (8 * 9) = 35$

$k_3 = (m_1m_2m_3)^{-1} \bmod m_4 = (5 * 7 * 8)^{-1} \bmod 9 = 1$

$X = [X_1 + k_1m_1 (X_2 – X_1) + k_2m_1m_2(X_3 – X_2) + k_3m_1m_2m_3 (X_4 – X_3)] \bmod (m_1m_2m_3m_4)$

$= [1 + 505 + 1225 + 280] \bmod 2520$

$X = 2011$

2.1.1. Traditional Hardware Implementation for CRT I:

In the traditional method, CRT I is implemented using adders, subtractors, multipliers, modulus operators and inverse modulus operators. Multipliers need several full and half adders, which increases the delay and cost involvement in the operations. This part is explained in Chapter 4 in detail. For 4–moduli set, hardware requirements are as follows:

1) Three Subtractors
2) Nine multipliers
3) Three adders
4) One modulus operator
5) Three Inverse Modulus operators

Figure 2.1 describes the hardware assembly for implementing CRT I using a 4-moduli set.

2.2. New Chinese Remainder Theorem II (CRT II):

The converter based on new Chinese remainder theorem II does not require any big size modulo adders for any moduli in the conversion process. This approach is applied only for co-prime moduli sets, and the converter algorithm is based on divide and conquer approach.

For instance, consider a 2-moduli set S = {m_1, m_2} where $m_1 < m_2$ and the RNS representation

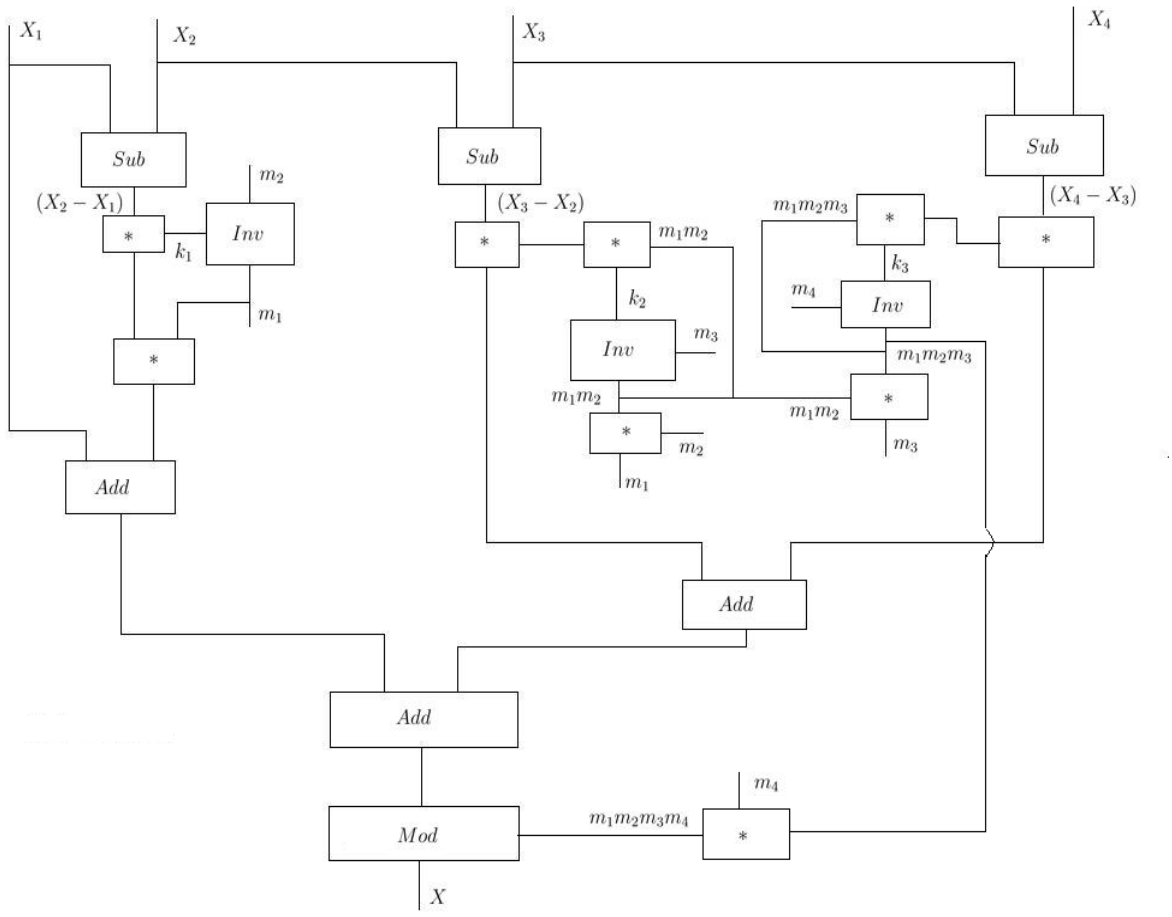


Fig.2.1. Hardware assembly for calculating X using CRT I

of a weighted number, X be (X_1, X_2). From the residue numbers X_1 and X_2, we want to calculate X, such that $X = X_1 \bmod m_1$ and $X = X_2 \bmod m_2$. The following derivation [1] shows how to retrieve the weighted number X from its residues.

Derivation:

Given that,

$$X = X_2 \bmod m_2 \text{ and } X = X_1 \bmod m_1$$

We know, $M_2 = \prod_{i=t+1}^{n} m_i$, $X = X_2 + k * m_2$, *for some integer k.*

$$X_2 + k * m_2 = X_1 \bmod m_1$$

$$k * m_2 = X_1 - X_2 \bmod m_1$$

Alternately, $k = (m_2)^{-1} \bmod m_1$

$$k * (X_1 - X_2) * m_2 = X_1 - X_2 \bmod m_1$$

Therefore we have, $X = [X_2 + k (X_1 - X_2) * m_2] \bmod (m_1 * m_2)$

$$X = X_2 + [\{k (X_1 - X_2) * m_2\} \bmod m_1] * m_2 \qquad (2.6)$$

The above derivation shows the decoding of two moduli sets in which the big sized modulo $M = m_1 * m_2$ is not required. The algorithm for finding the weighted number X from the residues $X_1$and X_2 is done using two procedures. They are *findno* and *translate*, and they are described as follows,

Procedure: findno(X_1, X_2, m_1, m_2, X):

1) Find the value of *k* , where $k = (m_2)^{-1} \bmod m_1$
2) $X = X_2 + [\{k (X_1 - X_2) * m_2\} \bmod m_1] * m_2$

The figures below show the diagrammatic representation of this procedure.

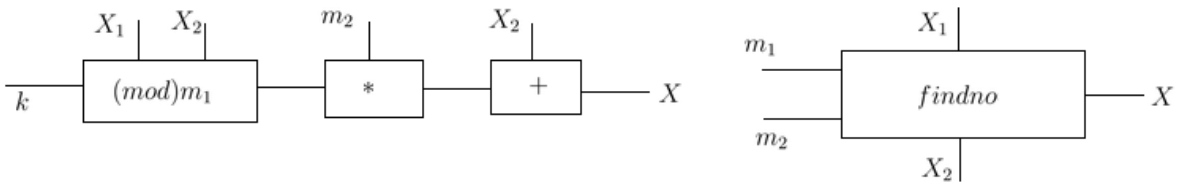


Fig.2.2. Implementation of *findno* procedure

Similarly, this derivation can be extended for bigger moduli sets. The procedure used is *translate* $((X_1, X_2, X_3 \ldots X_n), X)$ and is as follows,

Step 1) Consider an *n* moduli set and the residues of a number $X = (X_1, X_2, X_3 \ldots X_n)$. Consider

$$t = \left\lfloor \frac{n}{2} \right\rfloor$$

The procedure *translate* is defined as follows,

translate $((X_1, X_2, \ldots, X_t), N_1)$

$$M_1 = \prod_{i=1}^{t} m_i \tag{2.7}$$

translate $((X_{t+1}, X_{t+2}, \ldots, X_n), N_2)$

$$M_2 = \prod_{i=t+1}^{n} m_i \tag{2.8}$$

Step 2) Follow the procedure *findno* (N_1, N_2, M_1, M_2, X)

This retrieves the correct decimal value of X from $(X_1, X_2, X_3 \ldots X_n)$. The correctness of the above algorithm is proved in [1] by the induction method.

Illustration 2:

Consider a weighted number X = 2011 and a moduli set of the form $\{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$ where n = 2k+1, k = 1, 2, 3…

The proof for the set $\{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$ to be pair wise relatively prime is dealt in the appendix B (Section B.1). For this illustration let us take the value of n to be 3. So we get the moduli set as {5, 7, 8, 9}.The RNS representation of $X = (X_1, X_2, X_3, X_4)$

X_1=2011 mod 5 = 1; X_2=2011 mod 7 = 2; X_3= 2011 mod 8 = 3; X_4=2011 mod 9 = 4

Therefore X = (1, 2, 3, 4).

$k_1= (m_2)^{-1} \text{ mod } m_1 = (7)^{-1} \text{ mod } 5 = 3$

$k_2= (m_4)^{-1} \text{ mod } m_3 = (9)^{-1} \text{ mod } 8 = 1$

$k_3= (m_3m_4)^{-1} \text{ mod } (m_1m_2) = (72)^{-1} \text{ mod } 35 = 18$

$N_1 = X_2 + [k_1*(X_1 – X_2) \text{ mod } m_1] * m_2$

$= 2 + \{[3*(-1)] \text{ mod } 5\}*7$

$= 16$

$N_2 = X_4 + [k_2*(X_3 – X_4) \text{ mod } m_3] * m_4$

$= 4 + \{[1*(-1) \text{ mod } 8\} * 9$

$= 67$

$X = N_2 + [k_3*(N_1-N_2) \text{ mod } (m_1m_2)] * (m_3m_4)$

$= 67 + \{[18 * (-51)] \text{ mod } 35\} * 72$

$= 2011$

2.2.1. Traditional Hardware Implementation for CRT II:

In the traditional method, CRT II is implemented using subtractors, multipliers, dividers, modulus operators and inverse modulus operators. Multipliers and dividers need several full and half adders, which increases the delay and cost involvement of the operations. For 4–moduli set,

hardware requirements are as follows:

1) Three Subtractors
2) Four multipliers
3) Three adders
4) Three modulus operator
5) Three Inverse Modulus operator

Figures 2.3 and 2.4 describe the hardware assembly for implementing CRT II using 4-moduli set.

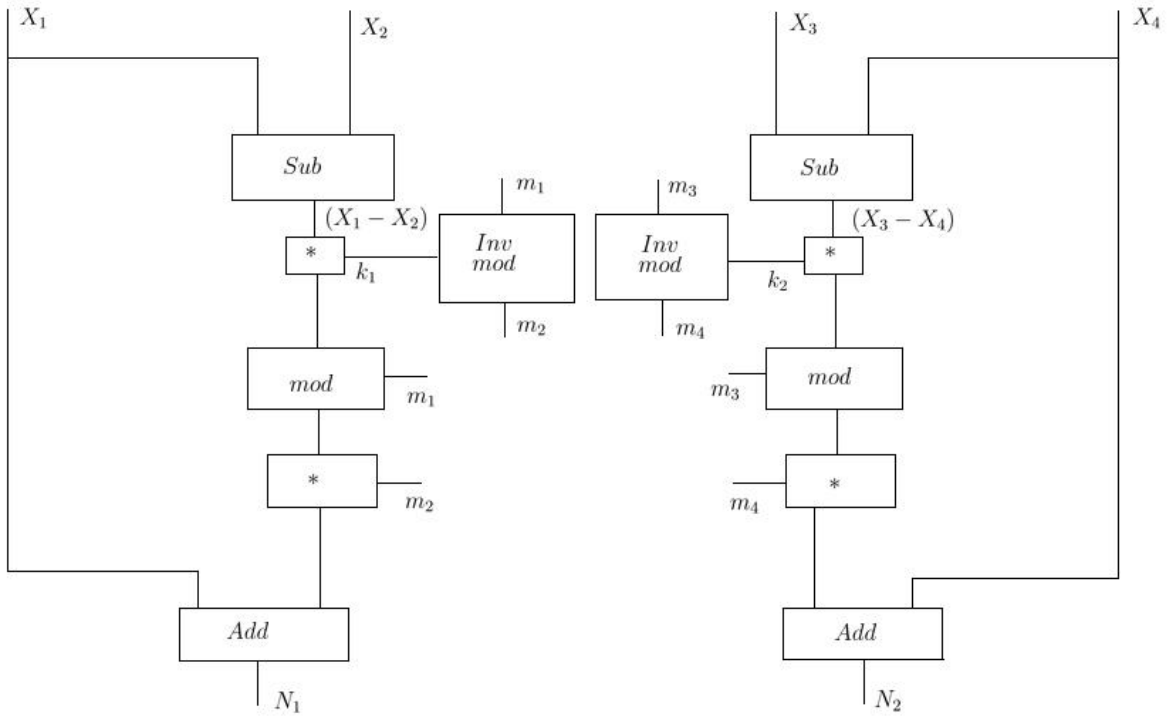


Fig.2.3. Hardware assembly for calculating N_1 and N_2 using CRT II

2.3. New Chinese Remainder Theorem III (CRT III):

CRT I and CRT II are designed for co-prime moduli sets, whereas CRT III is designed for non co-prime moduli sets. That is, for a moduli set S = {m_1, m_2, m_3... m_k} the non co-prime

criteria is (m_i, m_j) ≠ 1, i ≠ j and k ≥ 2. The dynamic range for this non co-prime moduli set is the least common multiple of the moduli in the set S, that is

$$M = lcm\ (m_1, m_2, m_3 \ldots m_k) \tag{2.9}$$

Let *t* be the floor of (k/2).

$$t = \left\lfloor \frac{k}{2} \right\rfloor \tag{2.10}$$

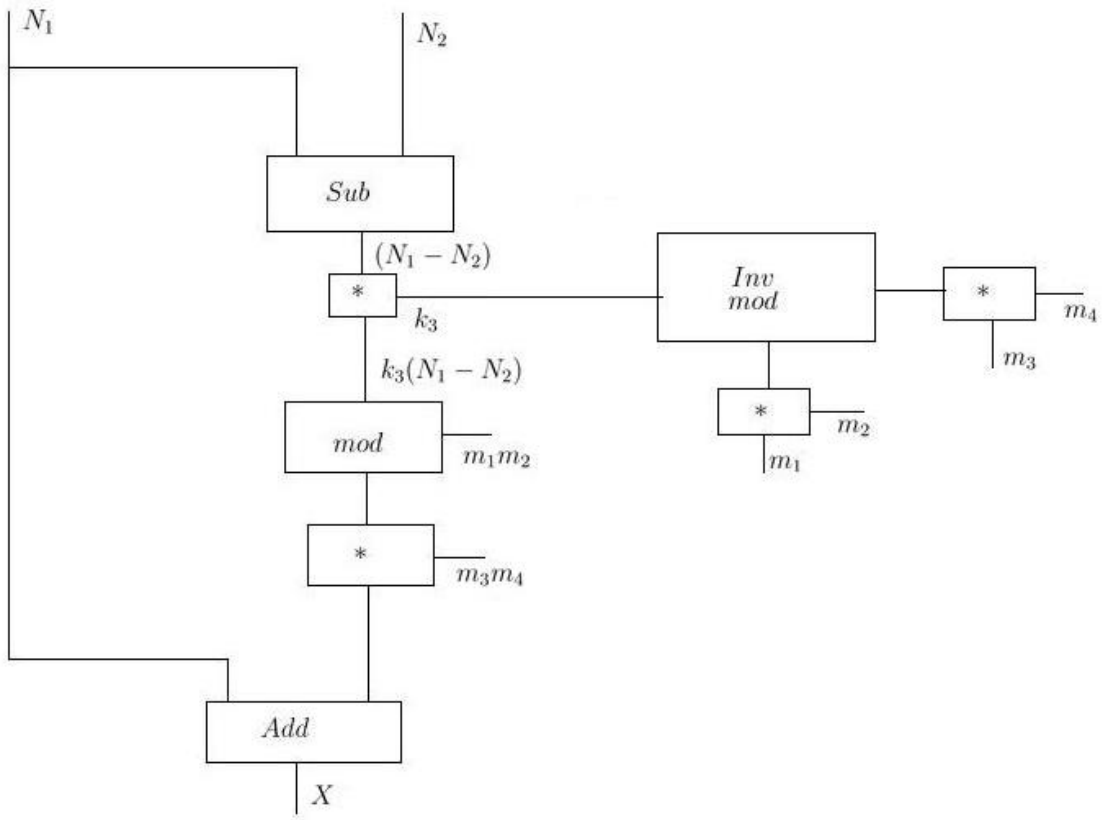


Fig.2.4. Hardware assembly for calculating X using CRT II

Splitting the moduli set S into two subsets as {m_1, m_2, m_3... m_t} and {m_{t+1}, m_{t+2}, m_{t+3}... m_k}. M_1 and M_2 is defined as follows,

$$M_1 = lcm\ (m_1, m_2, m_3 \ldots m_t) \tag{2.11}$$

$$M_2 = lcm\ (m_{t+1}, m_{t+2}, m_{t+3} \ldots m_k) \tag{2.12}$$

The weighted number X has the following RNS representation,

$$X \longrightarrow (X_1, X_2) \tag{2.13}$$

where $X_1 = X \bmod M_1$ and $X_2 = X \bmod M_2$.

The reconstruction of X from X_1 and X_2 is shown in equation (2.14).

$$X = X_1 + M_1 \left[\left(\frac{M_1}{d} \right)^{-1} \left(\frac{X_2 - X_1}{d} \right) \right] \bmod \left(\frac{M_2}{d} \right) \tag{2.14}$$

where $d = (M_1, M_2)$. In the above formula, it is to be noticed that if d = 1, then it yields the traditional Mixed Radix Conversion formula.

2.3.1. Requirements for the Efficient Implementation of CRT III:

Consider the equation (2.14), in which the constants M_1 and M_2 are involved. The moduli set should be chosen in such a way that $M_1 > M_2$. If this condition is satisfied, then the hardware requirements will be less because mod (M_2/d) hardware is required instead of mod (M_1/d) hardware. Also, moduli sets should be chosen such that the least common multiple of the moduli involved is as large as possible, so that bigger dynamic range is obtained. As the division operation can slow down the process, it should be avoided if possible. In equation (2.14), we have the term (X_2-X_1)/d. This division can be eliminated if (d^{-1}) mod (M_2/d) exists. In this case, the formula simplifies to

$$X = X_1 + M_1 \left[(\frac{M_1}{d})^{-1} [X_1 - X_2](d)^{-1} \right] \bmod \left(\frac{M_2}{d} \right) \tag{2.15}$$

All these issues are addressed, if the chosen moduli sets have conjugate pairs of moduli. Refer to [2] for the details of the conjugate pair of moduli. In the later section, special moduli sets are used to handle all these issues and some other special issues for the efficient and simplified implementation of CRT III.

2.3.2. Decoding Conjugate Pairs of Moduli:

As conjugate pairs of moduli offer larger dynamic ranges, hardware efficient implementation and fast and balanced RNS arithmetic, conjugate pair moduli is one of the best choices.

2.3.2.1. Decoding Two Conjugate Pair Moduli:

Consider the 4 –moduli set S consisting of two pairs of conjugate moduli [2],

$$S = m_1^*, m_1, m_2^*, m_2 = 2^{n_1}+1,\ 2^{n_1}-1,\ 2^{n_2}+1,\ 2^{n_2}-1 \qquad (2.16)$$

where $2^{n_1}+1$ and $2^{n_1}-1$ is a conjugate pair and $2^{n_2}+1$ and $2^{n_2}-1$ is another conjugate pair. The dynamic range of the above set is the least common multiple of $\{m_1^*, m_1, m_2^*, m_2\}$. Let the weighted number be X and its RNS representation be

$$X \xrightarrow{\text{RNS}} (X_1^*, X_1, X_2^*, X_2) \qquad (2.17)$$

$$X_i^* = X \bmod m_i^* \quad \text{and} \quad X_i = X \bmod m_i \quad \text{where i = 1, 2.} \qquad (2.18)$$

For better understanding, the decoding is done in two stages. In the first stage, moduli $\{m_1^*, m_1\}$ and $\{m_2^*, m_2\}$ are used separately in the conversion process. As m_1^* and m_1 are co-prime, CRT III becomes MRC and let the variable Z_1 be described by MRC formula as follows,

$$Z_1 = X_1^* + m_1^*[(m_1^*)^{-1}(X_1 - X_1^*)] \bmod m_1 \qquad (2.19)$$

Using the moduli from (2.16) and substituting in (2.19), we get,

$$Z_1 = X_1^{*} + (2^{n_1} + 1)\left[(2^{n_1} - 1)^{-1}(X_1 - X_1^{*})\right] \; mod \; (2^{n_1} - 1) \tag{2.20}$$

Similarly using m_2* and m_2 and, let the variable Z_2 be described by MRC formula as follows,

$$Z_2 = X_2^{*} + m_2^{*}\,[(m_2^{*})^{-1}\,(X_2 - X_2^{*})] \; mod \; m_2 \tag{2.21}$$

Using the moduli from (2.16) and substituting in (2.21), we get

$$Z_2 = X_2^{*} + (2^{n_2} + 1)\left[[2^{n_2} - 1]^{-1}[X_2 - X_2^{*}]\right] \; mod \; (2^{n_2} - 1) \tag{2.22}$$

In the second stage, using Z_1, Z_2, $M_1 = lcm\ (m_1^*, m_1)$ and $M_2 = lcm\ (m_2^*, m_2)$ in the equation (2.14), we get

$$X = Z_1 + (2^{2n_1} - 1)\left[\left(\frac{2^{2n_1} - 1}{d}\right)^{-1}\left(\frac{Z_2 - Z_1}{d}\right)\right] \bmod \left(\frac{2^{2n_2} - 1}{d}\right) \tag{2.23}$$

where $d = \gcd[2^{2n_1} - 1,\ 2^{2n_2} - 1]$

Depending on the value of n_1 and n_2, the value of d differs, and since d differs, the calculation of X from the equation (2.15) also differs. If $(d^{-1}) \bmod (2^{n_2} - 1)/d$ exists, then the calculation of X varies and the equation (2.23) becomes of the form (2.15).

Hence, the weighted number is retrieved back from its residues using CRT III.

2.3.2.2. Decoding Three Conjugate Pair Moduli:

The sets with three pairs of conjugate moduli achieve large dynamic ranges and become more balanced. According to [2], the largest possible dynamic range and the most balanced RNS arithmetic is achieved by the following moduli set,

$$S = \{m_1^*, m_1, m_2^*, m_2, m_3^*, m_3\} = \{2^{n+2}+1, 2^{n+2}-1, 2^{n+1}+1, 2^{n+1}-1, 2^n+1, 2^n-1\} \quad (2.24)$$

where n = odd.

The dynamic range for the above set is given by [2] as,

$$M = [(2^{2n} - 1)(2^{2n+2} - 1)(2^{2n+4} - 1) / 3^2] \quad (2.25)$$

The decoding technique for this is same as the technique used in the previous section for decoding two pairs of conjugate moduli [2], but the difference is that decoding three pairs is done in three stages. In the first stage, Z_1 is calculated using m_1* and m_1, Z_2 is calculated using m_2* and m_2, and Z_3 is calculated using m_3* and m_3. In the second stage, Z_4 is calculated using equation (2.14) replacing Z_1, Z_2 in the place of X_1 and X_2 respectively. In the third stage, X is calculated using (2.14), replacing Z_4 and Z_3 in the place of X_1 and X_2 respectively and replacing M_4 and M_3 in place of M_1 and M_2, where M_4 = lcm (M_1, M_2) and M_3 = lcm { m_3* , m_3 }.Hence the weighted number X is retrieved back from its residues.

2.3.3. Traditional Hardware Implementation for CRT III:

In the traditional method, CRT III is implemented using subtractors, multipliers, dividers, modulus operators and inverse modulus operators. Multipliers and dividers need several full and half adders, which increases the delay and cost involvement in the operations. For 4–moduli set, hardware requirements are as follows:

1) One Subtractor
2) Four multipliers
3) Three dividers
4) One Modulus operator

5) One Inverse Modulus operator

Figure 2.5 describes the hardware assembly for implementing CRT III using 4-moduli set.

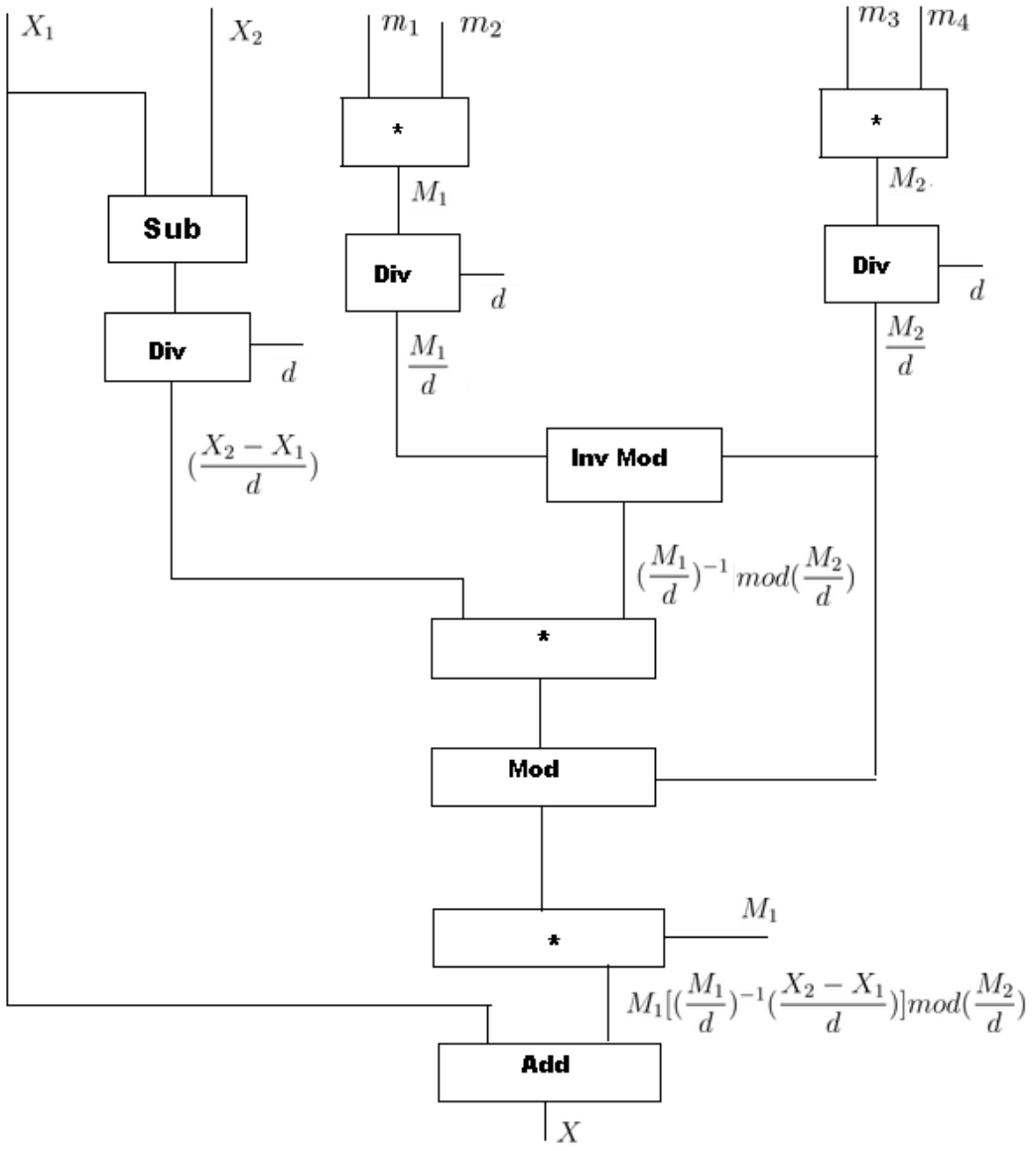


Fig.2.5. Traditional hardware implementation of CRT III

3. OPTIMIZING NEW CHINESE REMAINDER THEOREMS

3.1. Shortfalls of New Chinese Remainder Theorems:

New Chinese Remainder theorems overcome the shortfalls of traditional CRT and MRC techniques by improving parallelization. However, New CRTs are hardware intensive as the equations involve many inverse modulus operators, modulus operators, multipliers and dividers. Dividers and inverse modulus operators in turn needs many half and full adders and subtractors. This can be explained by the following illustrations.

Illustration 1:

For instance, consider the generalized application of CRT II using a moduli set from [4], $M = \{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$ *where* $n = 2k+1$, $k = 1, 2, 3, 4 \ldots$

The proof for the set $\{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$ to be pair wise relatively prime is dealt in the appendix B (Section B.1).

Therefore $n = 3, 5, 7, 9 \ldots$

Let $X = \{x_1, x_2, x_3, x_4\}$

$k_1 = (m_2)^{-1} \bmod m_1$

$= (2^n-1)^{-1} \bmod (2^{n-1}+1)$

Let us calculate the values of k_1 for the different values of n

For $n = 3$, $k_1 = (7)^{-1} \bmod 5 = 3$

$= 0\ 1\ 1$ *(in binary)*

$= 2^1+1$

$= 2^{n-2} +1$

For $n = 5$, $k_1 = (31)^{-1} \bmod (17) = 11$

$= 1\ 0\ 1\ 1$ (in binary)

$= 2^3+ 2^1+1$

$= 2^{n-2} + 2^{n-4} +1$

For $n =7$, $k_1 = (127)^{-1} \bmod (65) = 43$

$= 1\ 0\ 1\ 0\ 1\ 1$ (in binary)

$=2^5+ 2^3+ 2^1+1$

$= 2^{n-2} + 2^{n-4} +2^{n-6}+1$

For $n =9$, $k_1 = (511)^{-1} \bmod (257) =171$

$=1\ 0\ 1\ 0\ 1\ 0\ 1\ 1$ (in binary)

$=2^7+2^5+ 2^3+ 2^1+1$

$= 2^{n-2} + 2^{n-4} +2^{n-6}+2^{n-8}+1$

Thus the above values of k_1 can be generalized as follows,

$$k_1 = \sum_{l=1}^{n-2} 2^l + 1$$ where l is odd.

Let us calculate the values of k_2 for the different values of n

$$k_2 = (m_4)^{-1}\, m_3$$

$$= (2^n+1)^{-1} \bmod 2^n$$

$$(m_4)^{-1}\, m_3 = (m_4 \bmod m_3)^{-1} \bmod m_3$$

$$= [(2^n+1) \bmod 2^n]^{-1} \bmod 2^n$$

Since $(2^n+1) \bmod 2^n = 1$

$$k_2 = 1^{-1} \bmod 2^n$$

$$k_2 = 1$$

For example,

For $n = 3$, $k_2 = (9)^{-1} \bmod 8 = 1$

For $n = 5$, $k_2 = (33)^{-1} \bmod 32 = 1$

For $n = 7$, $k_2 = (129)^{-1} \bmod 128 = 1$

For $n = 9$, $k_2 = (513)^{-1} \bmod 512 = 1$

Similarly, let us calculate the values for k_3 by substituting different values of n.

$$k_3 = [m_4 m_3]^{-1} \bmod (m_1 m_2)$$

$$k_3 = [(2^n)\,(2^n + 1)]^{-1} \bmod [(2^{n-1} + 1)\,(2^n - 1)]$$

$$k_3 = [(2^n)\,(2^n + 1)]^{-1} \bmod [(2^{n-1} + 1)\,(2^n - 1)]$$

Using the property, $a^{-1} \bmod b = x$, *where* $[a * x] \bmod b = 1$

$$[(2^n)\,(2^n + 1) * x] \bmod [(2^{n-1} + 1)\,(2^n - 1)] = 1$$

Therefore $x = 2^{n-2} + 2^{2(n-1)}$

$$k_3 = 2^{n-2} + 2^{2(n-1)}$$

For example,

For $n = 3$, $72^{-1} \bmod 35 = 18$

$= 1\ 0\ 0\ 1\ 0$ (in binary)

$= 2^1 + 2^4$

For $n = 5$, $1056^{-1} \bmod 527 = 264$

$= 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0$ (in binary)

$= 2^3 + 2^8$

For $n = 7$, $16512^{-1} \bmod 8255 = 4128$

$= 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0$ (in binary)

$= 2^5 + 2^{12}$

For $n = 9$, $262656^{-1} \bmod 131327 = 65664$

$= 2^7 + 2^{16}$

Hence N_1 and N_2 are calculated as follows,

$$N_1 = x_2 + [\sum_{l=1}^{n-2} (2^l + 1)(x_1 - x_2) \bmod m_1] * m_2 \quad (3.1)$$

$$N_2 = x_4 + [(x_3 - x_4) \bmod m_3] * m_4 \quad (3.2)$$

Hence X can be calculated as,

$$X = N_2 + [(2^{n-2} + 2^{2(n-1)})(N_1 - N_2) \bmod m_1 m_2] * m_3 m_4 \quad (3.3)$$

Illustration 2:

Consider a generalized application of CRT III using a non co-prime moduli set from [4], $M = \{2^{n-2} + 1, 2^{n-2} - 1, 2^{n} + 1, 2^{n} - 1\}$ where $n = 2k + 1$, $k = 3, 4, 5$.

Therefore $n = 7, 9, 11, 13, 15 \ldots$

$$X = \{x_1^*, x_1, x_2^*, x_2\}$$

$$Z_1 = x_1^* + m_1^* [(m_1^*)^{-1} (x_1 - x_1^*)] \bmod m_1$$

$$Z_1 = x_1^* + 2^{n-2} + 1 [(2^{n-2} + 1)^{-1} (x_1 - x_1^*)] \bmod (2^{n-2} - 1) \quad (3.4)$$

Let us calculate the value of $[(2^{n-2} + 1)^{-1} \bmod (2^{n-2} - 1)]$.

$$[(2^{n-2} + 1)^{-1} \bmod (2^{n-2} - 1)] = [(2^{n-2} + 1) \bmod (2^{n-2} - 1)]^{-1} \bmod (2^{n-2} - 1)$$

$$= 2^{-1} \bmod (2^{n-2} - 1)$$

Using the property, $a^{-1} \bmod b = x$, where $[a * x] \bmod b = 1$

$$[2 * x] \bmod (2^{n-2} - 1) = 1$$

$$x = 2^{n-3}$$

Therefore, $[(2^{n-2} + 1)^{-1} \bmod (2^{n-2} - 1)] = 2^{n-3}$

For example,

For $n = 7$, $(33)^{-1} \bmod 31 = 16 = 2^4$

For $n = 9$, $(129)^{-1} \bmod 127 = 64 = 2^6$

For $n = 11$, $(513)^{-1} \bmod 511 = 256 = 2^8$

For $n = 13$, $(2049)^{-1} \bmod 2047 = 1024 = 2^{10}$

For $n = 15$, $(8193)^{-1} \bmod 8191 = 4096 = 2^{12}$

$$Z_2 = x_2^* + (2^n + 1)\,[(2^n + 1)^{-1}\,(x_2 - x_2^*)] \bmod (2^n - 1) \quad (3.5)$$

Let us calculate the value of $[(2^n + 1)^{-1} \bmod (2^n - 1)]$.

$$(2^n + 1)^{-1} \bmod (2^n - 1) = [(2^n + 1) \bmod (2^n - 1)]^{-1} \bmod (2^n - 1)$$

$$= 2^{-1} \bmod (2^n - 1)$$

Using the property, $a^{-1} \bmod b = x$, where $[a * x] \bmod b = 1$

$$[2 * x] \bmod (2^n - 1) = 1$$

$$x = 2^{n-1}$$

Therefore, $[(2^n + 1)^{-1} \bmod (2^n - 1)] = 2^{n-1}$

For example,

For n = 7, $(129)^{-1} \bmod 127 = 64 = 2^6$

For n = 9, $(513)^{-1} \bmod 511 = 256 = 2^8$

For n = 11, $(2049)^{-1} \bmod 2047 = 1024 = 2^{10}$

For n = 13, $(8193)^{-1} \bmod 8191 = 4096 = 2^{12}$

For n = 15, $(32769)^{-1} \bmod 32767 = 16384 = 2^{14}$

$$M_1 = lcm\,(m_1^*, m_1) = lcm\,(2^{n-2} + 1, 2^{n-2} - 1)$$

$$= 2^{2(n-2)} - 1$$

$$M_2 = lcm\,(m_2^*, m_2) = lcm\,(2^n + 1, 2^n - 1)$$

$$= 2^{2n} - 1$$

$$d = (2^{2(n-2)} - 1,\ 2^{2n} - 1) = 3$$

$$X = Z_1 + (2^{2(n-2)} - 1)[(\frac{2^{2(n-2)} - 1}{3})^{-1}(\frac{Z_2 - Z_1}{3})] \bmod (\frac{2^{2n} - 1}{3}) \quad (3.6)$$

The value of $(\frac{2^{2(n-2)} - 1}{3})^{-1} \bmod (\frac{2^{2n} - 1}{3})$ can be calculated for different values of n as follows,

For $n = 7$, $(341)^{-1} \bmod 5461 = 1089$

$= 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1$ *(in binary)*

$= 2^{10} + 2^6 + 1$

$= 2^{n+3} + 2^{n-1} + 1$

For $n = 9$, $(5461)^{-1} \bmod 87381 = 17473$

$= 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1$ *(in binary)*

$= 2^{14} + 2^{10} + 2^6 + 1$

$= 2^{n+5} + 2^{n+1} + 2^{n-3} + 1$

For $n = 11$, $(87381)^{-1} \bmod 1398101 = 279617$

$= 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1$ *(in binary)*

$= 2^{18} + 2^{14} + 2^{10} + 2^6 + 1$

$= 2^{n+7} + 2^{n+3} + 2^{n-1} + 2^{n-5} + 1$

For $n = 13$, $(87381)^{-1} \bmod 1398101 = 4473921$

$= 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1$ *(in binary)*

$= 2^{22} + 2^{18} + 2^{14} + 2^{10} + 2^6 + 1$

$$= 2^{n+9} + 2^{n+5} + 2^{n+1} + 2^{n-3} + 2^{n-7} + 1$$

This can be generalized as follows,

$$(\frac{2^{2(n-2)}-1}{3})^{-1} \bmod (\frac{2^{2n}-1}{3}) = \sum_{k=(6-n)}^{n-4} 2^{n+k} + 1 \tag{3.7}$$

which can be used in (3.6).

Therefore, (3.6) becomes as follows,

$$X = Z_1 + (2^{2(n-2)}-1)[\sum_{k=(6-n)}^{n-4} 2^{n+k} + 1(\frac{Z_2 - Z_1}{3})] \bmod (\frac{2^{2n}-1}{3}) \tag{3.8}$$

Now let us observe the hardware implementation of the above two illustrations for calculating *X* from its residues.

First, let us implement CRT II to calculate *X* using the equations (3.1), (3.2) and (3.3) from illustration 1. In equation (3.1), to calculate N_1, *(n-2)* adders are required for evaluating the summation term in the equation, where *n = 3, 5, 7, 9 . . .* Apart from this, one full adder, one subtractor, one multiplier and one modulus operator are also needed. In equation (3.2), to calculate N_2, one full adder, one subtractor, one multiplier and one modulus operator are required. Similarly, to calculate *X* from equation (3.3), two adders, one subtractor, two multipliers and a modulus operator are needed.

Now, let us implement CRT III to calculate *X* using the equations (3.4), (3.5) and (3.6). To calculate Z_1, two subtractors, three adders, one modulus operator and one multiplier are needed. Similarly for Z_2, three adders, two subtractors, one modulus operator and one multiplier are required. To calculate *X* from equation (3.6), we need *(2n-9)* adders for implementing the

summation term in the equation, where *n = 7, 9, 11, 13, 15 . . .* Apart from this, we also need three subtractors, two adders, two dividers, one multiplier and one modulus operator.

These two illustrations clearly show that the New Chinese Remainder Theorems are hardware intensive to implement them practically. So, these theorems have to be optimized for the pragmatic implementations in the field of Digital Signal Processing intensive computations such as digital filtering, convolutions, correlations, discrete Fourier transforms and fast Fourier transform computations and direct digital frequency synthesis.

3.2. Basic Lemmas Used for the Optimization:

The main drawback in using New Chinese Remainder theorems is hardware implementation, because the inverse modulo operation requires large number of full and half adders. So, it was observed that, if the inverse modulo operations are eliminated from those theorems, then the new CRTs are practically feasible to implement. In order to eliminate inverse modulo operations, optimized moduli sets have to be chosen based on the following lemmas.

Theorem: Let the co-prime moduli set be M = $\{m_1, m_2, m_3, m_4\}$. For co-prime numbers, if (m_1 mod m_2) is equal to 1, then $(m_1)^{-1}$ mod m_2 is also equal to 1.

Proof: By the property of inverse modulo,

$$(m_1)^{-1} \bmod m_2 = [(m_1 \bmod m_2)^{-1}] \bmod m_2$$

Substituting, $m_1 \bmod m_2 = 1$ in the above,

$$(m_1)^{-1} \bmod m_2 = 1 \bmod m_2$$

$$= 1$$

This theorem gives rise to 3 lemmas. They are as follows,

Lemma 1: If m_3 mod $m_4 = 1$, then $(m_3)^{-1}$ mod $m_4 = 1$

Lemma 2: If $(m_1*m_2) \bmod (m_3*m_4) = 1$, then $(m_1*m_2)^{-1} \bmod (m_3*m_4) = 1$

Lemma 3: If $(m_1*m_2*m_3) \bmod m_4 = 1$ then, and $(m_1*m_2*m_3)^{-1} \bmod m_4 = 1$ respectively.

3.3. Optimizing the Base Theorem for New Chinese Remainder Theorem I:

Consider the theorem which is a base for constructing the new Chinese Remainder Theorem I. For a four moduli set, the equation is

$$X = X_1 + k_1m_1(X_2-X_1) + k_2m_1m_2(X_3-X_2) + k_3m_1m_2m_3(X_4-X_3)] \bmod (m_1m_2m_3m_4) \qquad (3.9)$$

where, $k_1 = (m_1)^{-1} \bmod (m_2m_3m_4)$

$k_2 = (m_1m_2)^{-1} \bmod (m_3m_4)$

$k_3 = (m_1m_2m_3)^{-1} \bmod m_4$

To implement the above equation, the requirement is 3 inverse modulo operations which may have huge summation terms, 9 multipliers, 3 adders, 3 subtractors and 1 modulo operator. Hence, it is necessary to optimize this theorem so that it can be implemented in hardware. Let us optimize the theorem using 4-modui set.

Consider a moduli set $M = \{2^{n+2}+3, 2^{n+1}+1, 2^n+1, 2\}$ where $n = 1, 2, 3. . .$ Plugging the values for n, we get sample moduli sets as follows:

For n = 1: M = {11, 5, 3, 2}

For n = 2: M = {19, 9, 5, 2}

For n = 3: M = {35, 17, 9, 2} and so on.

The proof for the set $M= \{2^{n+2}+3, 2^{n+1}+1, 2^n+1, 2\}$ where n = 1, 2, 3. . ., to consist of pair wise relatively prime moduli is dealt in Appendix B(Section B.2).

Consider, $k_1 = (m_1)^{-1} \bmod (m_2m_3m_4)$

$$= (2^{n+2}+3)^{-1} \bmod [(2^{n+1}+1) * (2^n+1) * (2)]$$
$$= [(2^{n+2}+3) \bmod [(2^{n+1}+1) * (2^n+1) * (2)]]^{-1} \bmod [(2^{n+1}+1) * (2^n+1) * (2)]$$

Using the property, $a^{-1} \bmod b = x$, *where* $[a * x] \bmod b = 1$

$$[(2^{n+2}+3) * x] \bmod [(2^{n+1}+1) * (2^n+1) * (2)] = 1$$

$x = (2^{n+2}+3)$, which is same as m_1

The value of $(m_1)^{-1} \bmod (m_2m_3m_4) = m_1$ and this can be verified by plugging different values for *n*.

Consider $k_2 = (m_1m_2)^{-1} \bmod (m_3m_4)$

$$= [(2^{n+2}+3)* (2^{n+1}+1)]^{-1} \bmod [(2^n+1) * (2)]$$

$$= [(2^{n+2}+3)* (2^{n+1}+1)] \bmod [(2^n+1) * (2)]^{-1} \bmod [(2^n+1) * (2)]$$

$$= 1^{-1} \bmod [(2^n+1) * (2)]$$

$$=1$$

This can be verified by plugging the different values for *n*.

Consider $k_3 = (m_1m_2m_3)^{-1} \bmod m_4$

$$= [(2^{n+2}+3)*(2^{n+1}+1)*(2^n+1)]^{-1} \bmod 2$$

$$= [[(2^{n+2}+3)*(2^{n+1}+1)*(2^n+1)] \bmod 2]^{-1} \bmod 2$$

$$= 1^{-1} \bmod 2$$

$$= 1$$

This can be verified by plugging the different values for *n*.

Thus this moduli set reduces the equation (3.9) to,

$$X=[X_1+m_1^2(X_2-X_1) + m_1m_2(X_3-X_2) +m_1m_2m_3(X_4-X_3)] \bmod (m_1m_2m_3m_4) \qquad (3.10)$$

The above does not require any inverse modulo operators, which means it does not require any adders to implement huge summation terms, and the number of multipliers and adders also gets reduced. The hardware optimization is separately dealt in the next chapter.

Illustration 3:

Consider a weighted number X = 75 and moduli set M = {11, 5, 3, 2}.

Residue numbers for X are, X= $\{X_1, X_2, X_3, X_4\}$ = {9, 0, 0, 1}

Using equation (3.10), we get,

$$X = [9 + 121(-9) + 55(0) + 165(1)] \bmod 330$$

$$= [-915] \bmod 330$$

$$= 75$$

3.4. Optimizing New Chinese Remainder Theorem II:

Earlier, illustration 1 in this chapter proved that new CRT II is hardware intensive and it requires eliminating some of the terms that it has, so that it is feasible to implement in real time. Let us focus our attention to a four moduli set to optimize the theorem.

Consider a 4- moduli set $M = \{2, 2^n+1, 2^{n+1}+1, 2^{n+2}+3\}$ where $n = 1, 2, 3. . .$ Sample moduli sets are as follows,

$n= 1: M = \{2, 3, 5, 11\}$, $n=2: M= \{2, 5, 9, 19\}$, $n = 3: M= \{2, 9, 17, 35\}$ and so on.

The proof for the set $M= \{2, 2^n+1, 2^{n+1}+1, 2^{n+2}+3\}$ where $n = 1, 2, 3. . .$ to consist of pair wise relatively prime moduli is dealt in Appendix B (Section B.2).

Consider $k_1 = (m_2)^{-1} \bmod m_1$

$$= (2^n+1)^{-1} \bmod 2$$

$$= [(2^n+1) \bmod 2]^{-1} \bmod 2$$

$= 1^{-1} \bmod 2$

$= 1$

This can be verified by plugging different values for n in $(2^n+1)^{-1} \bmod 2$

Consider $k_2 = (m_4)^{-1} \bmod m_3$

$= (2^{n+2}+3)^{-1} \bmod 2^{n+1}+1$

$= [(2^{n+2}+3) \bmod 2^{n+1}+1]^{-1} \bmod 2^{n+1}+1$

$= 1^{-1} \bmod 2^{n+1}+1$

$= 1$

This can be verified by plugging different values for n in $(2^{n+2}+3)^{-1} \bmod 2^{n+1}+1$

Consider, $k_3 = (m_4 m_3)^{-1} \bmod (m_1 m_2)$

$= [(2^{n+2}+3)*(2^{n+1}+1)]^{-1} \bmod [(2^n+1)^* 2]$

$= [(2^{n+2}+3)*(2^{n+1}+1)] \bmod [(2^n+1)^* 2]^{-1} \bmod [(2^n+1)^* 2]$

$= 1^{-1} \bmod [(2^n+1)^* 2]$

$=1$

This can be verified by plugging different values for n in $[(2^{n+2}+3)*(2^{n+1}+1)]^{-1} \bmod [(2^n+1)^* 2]$

Hence the CRT II equations become as follows,

$$N_1 = x_2 + ([x_1 - x_2] \bmod m_1) * m_2$$

$$N_2 = x_4 + ([x_3 - x_4] \bmod m_3) * m_4$$

$$X = N_2 + [N_1 - N_2] \bmod (m_1 m_2)*(m_3 m_4) \qquad (3.11)$$

Equation (3.11) is a fully optimized CRT II equation, where it does not have any big size terms and inverse modulo terms. Hence, this equation is easy to implement in hardware for practical purposes and the hardware implementation is shown in the next section.

Illustration 4:

Consider a weighted number $X = 200$ and a moduli set $M = \{2, 3, 5, 11\}$.

Residue numbers for X are, $X = \{0, 2, 0, 2\}$

$$N_1 = x_2 + ([x_1 - x_2] \bmod m_1) * m_2$$

$$= 2 + (-2 \bmod 2)*3$$

$$= 2$$

$$N_2 = x_4 + ([x_3 - x_4] \bmod m_3) * m_4$$

$$= 2 + ([-2] \bmod 5) *11$$

$$= 35$$

$$X = N_2 + [N_1 - N_2] \bmod (m_1 m_2)*(m_3 m_4)$$

$$= 35 + [-33 \bmod 6] * 55$$

$$= 200$$

3.5. Optimizing New Chinese Remainder Theorem III:

Consider a moduli set,

$$M = \{2^{n+2}+1, 2^{n+2}, 2^{n+1}+1, 2\}, \text{ where } n = 1, 2, 3 \ldots \quad (3.12)$$

When $n = 1, M = \{9, 8, 5, 2\}$, $n = 2, M = \{17, 16, 9, 2\}$ $n = 3, M = \{33, 32, 17, 2\}$ and so on.

For the above moduli set in (3.12),

$$M_1 = (2^{n+2}+1)(2^{n+2}) \quad (3.13)$$

$$M_2 = (2^{n+1}+1)\ (2) \tag{3.14}$$

$$d = gcd\ (M_1, M_2)$$

$$= 2$$

$$\left(\frac{M_1}{d}\right)^{-1} \bmod \left(\frac{M_2}{d}\right) = \left[\frac{\left[(2^{n+2}+1)(2^{n+2})\right]}{2}\right]^{-1} \bmod \left[\frac{\left[(2^{n+1}+1)(2)\right]}{2}\right] \tag{3.15}$$

$$= [(2^{n+2}+1)*(2^{n+1})]^{-1} mod\ (2^{n+1}+1)$$

$$= [[(2^{n+2}+1)*(2^{n+1})]\ mod\ (2^{n+1}+1)]^{-1} mod\ (2^{n+1}+1)$$

$$= 1\ mod\ (2^{n+1}+1)$$

$$= 1$$

For example,

For n = 1, $(36)^{-1}$ mod 5 = 1.

For n = 2, $(136)^{-1}$ mod 9 = 1.

Similarly, *for n = 3, $(528)^{-1}$ mod 17 = 1.*

This series goes on and thus,

$$\left(\frac{M_1}{d}\right)^{-1} \bmod \left(\frac{M_2}{d}\right) = 1 \tag{3.16}$$

Substituting the above in the original CRT III equation (2.14), we get

$$X = X_1 + M_1\left[\left(\frac{X_2 - X_1}{d}\right)\right] \bmod \left(\frac{M_2}{d}\right) \tag{3.17}$$

Also,

$$\frac{M_2}{d} = \frac{\left[(2^{n+1}+1)(2)\right]}{2} = (2^{n+1}+1) = m_3 \tag{3.18}$$

Substituting (3.18) in (3.17), we get

$$X = X_1 + M_1\left[\left(\frac{X_2 - X_1}{d}\right)\right] \bmod m_3 \tag{3.19}$$

The division term in the above equation can be eliminated as follows,

$$X = X_1 + M_1\,[(d)^{-1}\,(X_2 - X_1)]\ mod\ m_3 \tag{3.20}$$

$$(d)^{-1}\ mod\ m_3 = (2)^{-1}\ mod\ (2^{n+1}+1)$$

$$= 2^n + 1 \tag{3.21}$$

Substituting (3.21) in (3.20), we get

$$X = X_1 + M_1\,[(2^n+1)\,(X_2 - X_1)]\ mod\ m_3 \tag{3.22}$$

Equation (3.22) is a completely optimized new Chinese Remainder Theorem III, which has no big size summation terms and the most important is that it does not have any division terms and inverse modulo terms. To implement this equation in hardware is not a resource intensive and further the hardware optimization is done in the next chapter.

Illustration 5:

Let us consider a weighted number $X = 75$, and the moduli set be $M = \{9, 8, 5, 2\}$.

$M_1 = lcm\ (m_1, m_2) = (9, 8) = 72$

$M_2 = lcm\ (m_3, m_4) = (5, 2) = 10$

$d = gcd\ (M_1, M_2) = 2$

$X_1 = X \bmod M_1$

$= 75 \bmod 72 = 3$

$X_2 = X \bmod M_2$

$= 75 \bmod 10 = 5$

Using the equation (3.22), we calculate the value of X as follows,

$X = X_1 + M_1\ [(2^n + 1)\ (X_2 - X_1)] \bmod m_3$

$= 3 + 72\ [(3)\ (5-3)] \bmod 5$

$= 3 + 72\ [6] \bmod 5$

$= 75$

3.5.1. Extending the Optimized Four Moduli Set to A Five Moduli Set:

Consider the following moduli set,

$$M = \{2^{n+3}+2,\ 2^{n+2}+1,\ 2^{n+2},\ 2^{n+1}+1,\ 2\},\ where\ n = 1, 2, 3 \ldots \quad (3.23)$$

$M_1 = (2^{n+3}+2)*(2^{n+2}+1)$

$M_2 = (2^{n+2})*(2^{n+1}+1)*(2)$

$d = gcd\ (M_1, M_2) = 2$

$$(\frac{M_1}{d})^{-1} \bmod (\frac{M_2}{d}) = \frac{[(2^{n+3}+2)(2^{n+2}+1)]}{2}]^{-1} \bmod [\frac{[(2^{n+2})(2^{n+1}+1)(2)]}{2}$$

$= [(2^{n+2}+1)*(2^{n+2}+1)]^{-1} \bmod (2^{n+2})*(2^{n+1}+1)$

$= [(2^{n+2}+1)^2 \bmod (2^{n+2})*(2^{n+1}+1)]^{-1} \bmod (2^{n+2})*(2^{n+1}+1)$

$= 1 \bmod (2^{n+2})*(2^{n+1}+1)$

$$\left(\frac{M_1}{d}\right)^{-1} \bmod \left(\frac{M_2}{d}\right) = 1$$

Thus the CRT III equation becomes as follows:

$$X = X_1 + M_1\left[\left(\frac{X_2 - X_1}{d}\right)\right] \bmod \left(\frac{M_2}{d}\right)$$

Also,

$$\frac{M_2}{d} = \frac{\left[(2^{n+2})(2^{n+1}+1)(2)\right]}{2} = m_3 * m_4$$

Here (d^{-1}) mod (m_3*m_4) does not exist because both d and (m_3*m_4) are even.

Hence the optimized equation is,

$$X = X_1 + M_1\left[\left(\frac{X_2 - X_1}{d}\right)\right] \bmod (m_3 * m_4) \qquad (3.24)$$

Illustration 6:

Consider a weighted number $X = 221$ and the moduli set $M = \{18, 9, 8, 5, 2\}$

$M_1 = lcm\ (m_1, m_2) = (18, 9) = 162$

$M_2 = lcm\ (m_3, m_4) = (8, 5, 2) = 80$

$d = gcd\ (M_1, M_2) = 2$

$X_1 = X \bmod M_1$

$= 221 \bmod 162 = 59$

$X_2 = X \bmod M_2$

$= 221 \bmod 80 = 61$

Using the equation (3.24), we calculate the value of X as follows:

$$X = X_1 + M_1 \left[\left(\frac{X_2 - X_1}{d} \right) \right] \bmod (m_3 * m_4)$$

$= 59 + 162\ [1] \bmod (40)$

$= 221$

4. OPTIMIZED HARDWARE IMPLEMENTATION OF NEW CHINESE REMAINDER THEOREMS

4.1. Introduction:

In the previous chapter, the new Chinese Remainder theorems were optimized by removing inverse modulo and division operations which reduced the overhead of using big size summation terms in the equations. But, those equations carried multiplication terms which requires additional full and half adders, and this is a barrier for hardware implementation. This chapter deals with the removal of those multiplication terms which means the usage of additional full and half adders can be eliminated. The optimization is carried out by replacing multipliers by left shift registers, which not only avoids the usage of adders but also avoids the time delay in carrying out the multiplication operations.

4.2. Optimized Hardware Implementation of the Base Theorem for New Chinese Remainder Theorem I:

Using the moduli set from section 3.3, M = $\{2^{n+2}+3, 2^{n+1}+1, 2^n+1, 2\}$ where n = 1, 2, 3... and the optimized equation from the previous chapter, we get

$$X=[X_1+m_1^2(X_2-X_1) + m_1m_2(X_3-X_2) +m_1m_2m_3(X_4-X_3)] \mod (m_1m_2m_3m_4)$$

Substituting the above moduli set values in the equation, we get

$$X = [X_1 +Q_1+ Q_2 + Q_3] \mod (m_1m_2m_3m_4) \quad (4.1)$$

Where $Q_1 = (2^{2n+4} + 2^{n+4} + 2^{n+3} +2^3 +1)*(X_2 – X_1)$

$Q_2 = (2^{2n+3} + 2^{n+3} + 2^{n+1} + 2^1+1)*(X_3 – X_2)$

$Q_3= [(2^{3n+3} + 2^{2n+3} + 2^{2n+1} + 2^{n+1}+2^n) + (2^{2n+3} + 2^{n+3} + 2^{n+1} + 2^1+1)] * (X_4 – X_3)$

The hardware implementation of equation (4.1) without using multipliers is shown in the figures Fig. 4.1.a, Fig. 4.1.b, Fig. 4.1.c and Fig. 4.1.d.

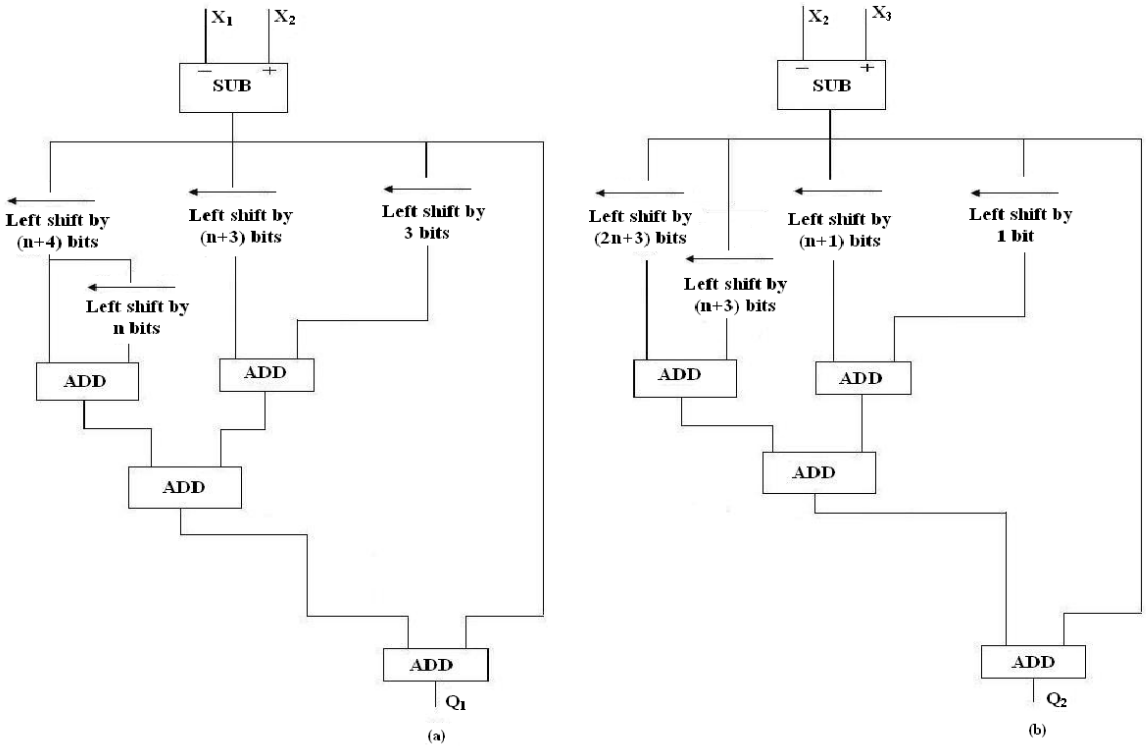


Fig. 4.1.a Hardware implementation of Q_1 by CRT I without using multipliers.

Fig. 4.1.b Hardware implementation of Q_2 by CRT I without using multipliers.

The total hardware requirements for the above implementations are

1) 3 subtractors
2) 1 modulus operators
3) 16 adders
4) 2 (n+4)-bit, 3 (n+3)-bit, 2 (2n+3)-bit, 2 (n+1)-bit, 2 1-bit, 1 (n+3)-bit, and 1 3-bit left shift registers.

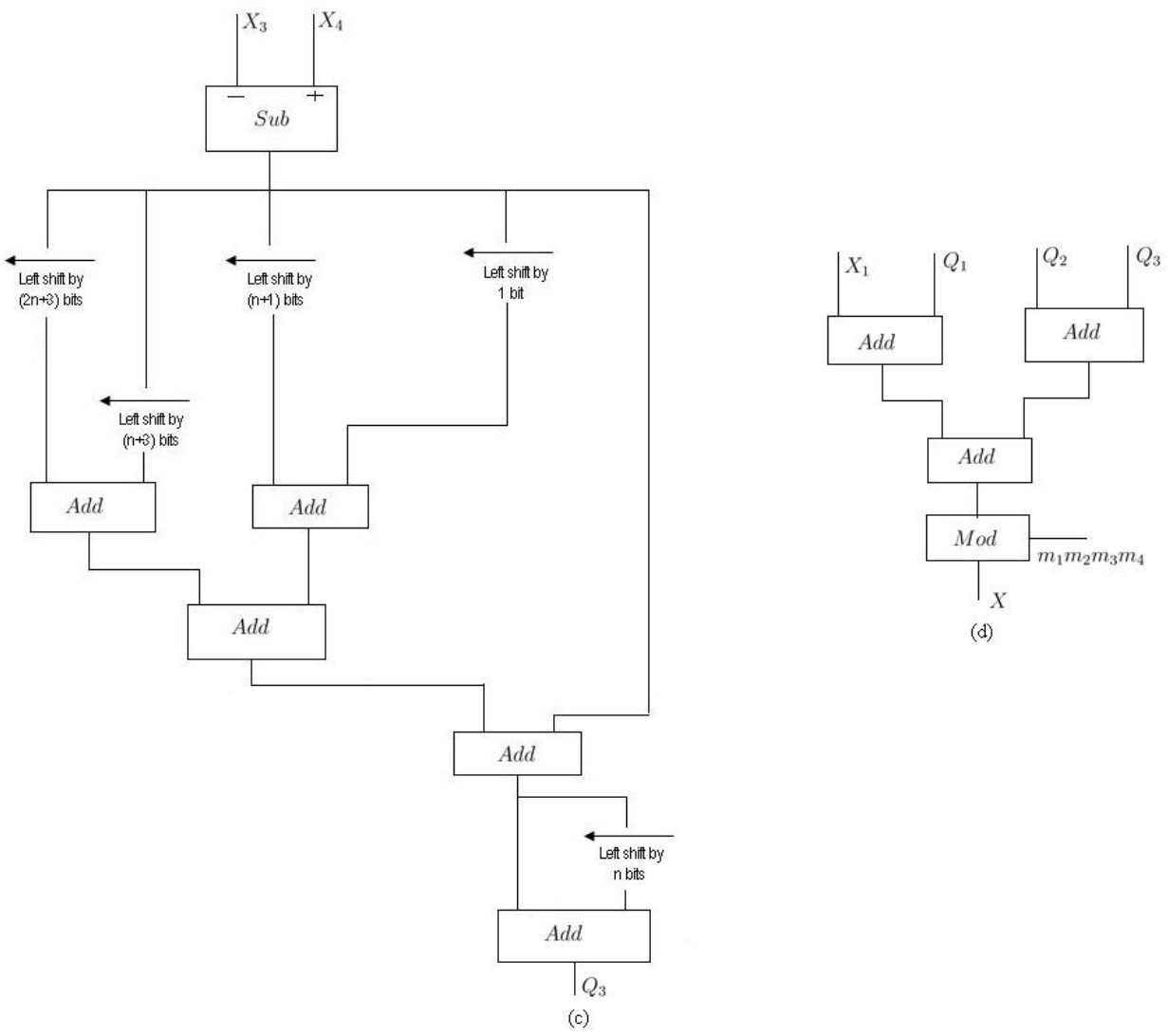


Fig. 4.1.c Hardware implementation of Q_3 by CRT I without using multipliers.

Fig. 4.1.d Hardware implementation of X by CRT I without using multipliers.

4.3. Optimized Hardware Implementation Of New Chinese Remainder Theorem II:

Using the moduli set from the section 3.4, $M = \{2, 2^n+1, 2^{n+1}+1, 2^{n+2}+3\}$ where $n = 1, 2, 3. . .$, and considering the optimized CRT II equation from chapter 3,

$$N_1 = x_2 + ([x_1 - x_2] \bmod m_1) * m_2$$

$$= x_2 + ([x_1 - x_2] \bmod 2) * (2^n + 1)$$

$$N_1 = x_2 + ([x_1 - x_2] \bmod m_1) * 2^n + ([x_1 - x_2] \bmod m_1) \quad (4.2)$$

$$N_2 = x_4 + ([x_3 - x_4] \bmod m_3) * m_4$$

$$= x_4 + ([x_3 - x_4] \bmod m_3) * (2^{n+2}+3)$$

$$= x_4 + ([x_3 - x_4] \bmod m_3) * (2^{n+2}+2^1+1)$$

$$N_2 = x_4 + ([x_3 - x_4] \bmod m_3) * (2^{n+2}) + ([x_3 - x_4] \bmod m_3) *(2^1) + ([x_3 - x_4] \bmod m_3) \quad (4.3)$$

$$X = N_2 + [N_1 - N_2] \bmod (m_1 m_2) * (m_3 m_4)$$

$$= N_2 + [N_1 - N_2] \bmod (m_1 m_2) * [(2^{n+1}+1).(2^{n+2}+3)]$$

$$X = N_2 + [N_1 - N_2] \bmod (m_1 m_2) * [2^{2n+3} + 2^{n+3} + 2^{n+1} + 2^2] \quad (4.4)$$

Equations (4.2), (4.3) and (4.4) are implemented in hardware without using multipliers as shown in the figures 4.2.a, 4.2.b and 4.3.

The total hardware requirements for the above implementations are

1) 3 subtractors
2) 3 modulus operators
3) 9 adders
4) 1 n-bit, 1 (n+1)-bit, 1 1-bit, 1 (2n+3)-bit, 1 (n+3)-bit, 1 (n+1)-bit and 1 2-bit left shift registers.

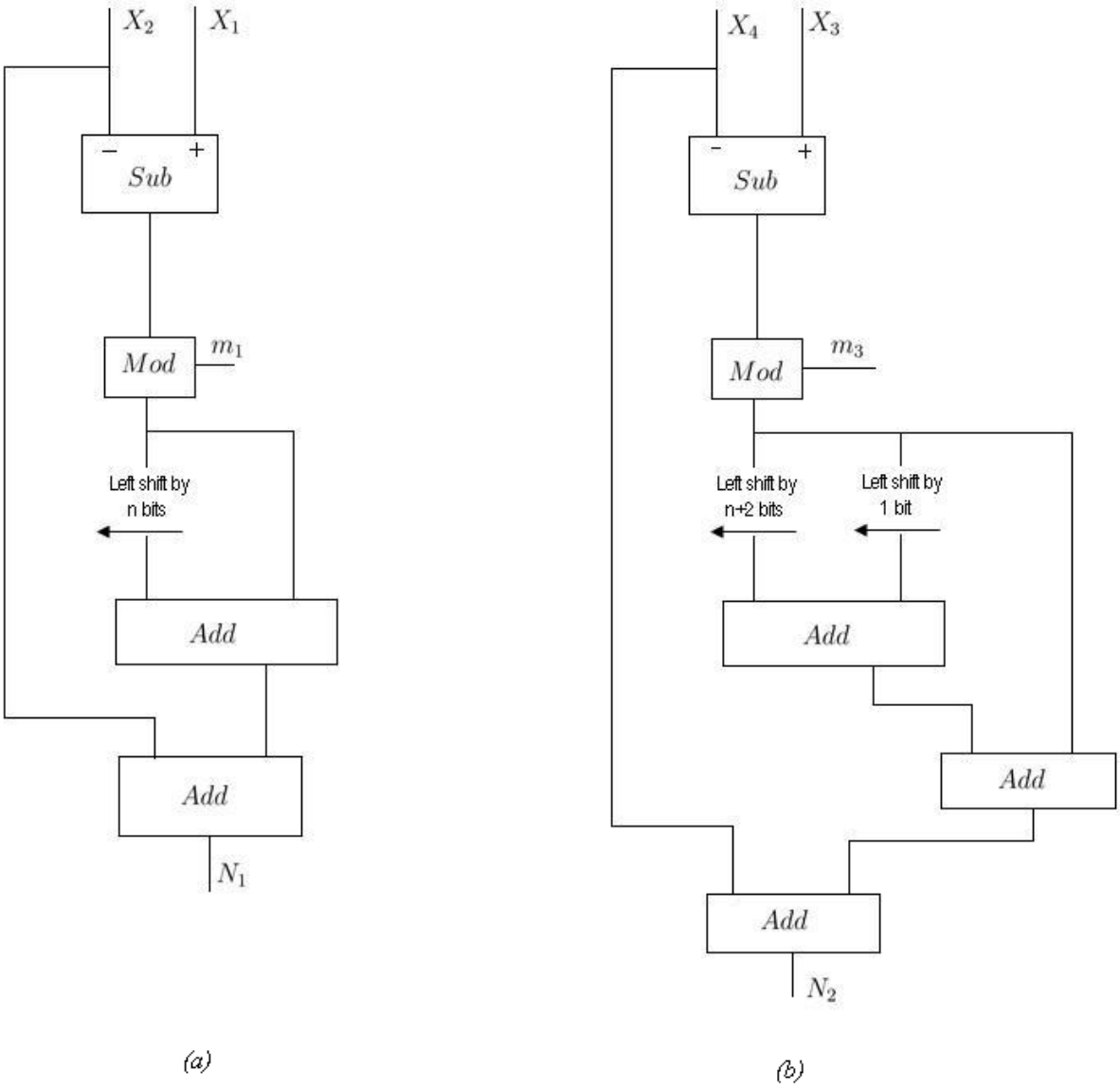


Fig.4.2.a Hardware implementation of N_1 by CRT II without using multipliers.

Fig.4.2.b Hardware implementation of N_2 by CRT II without using multipliers.

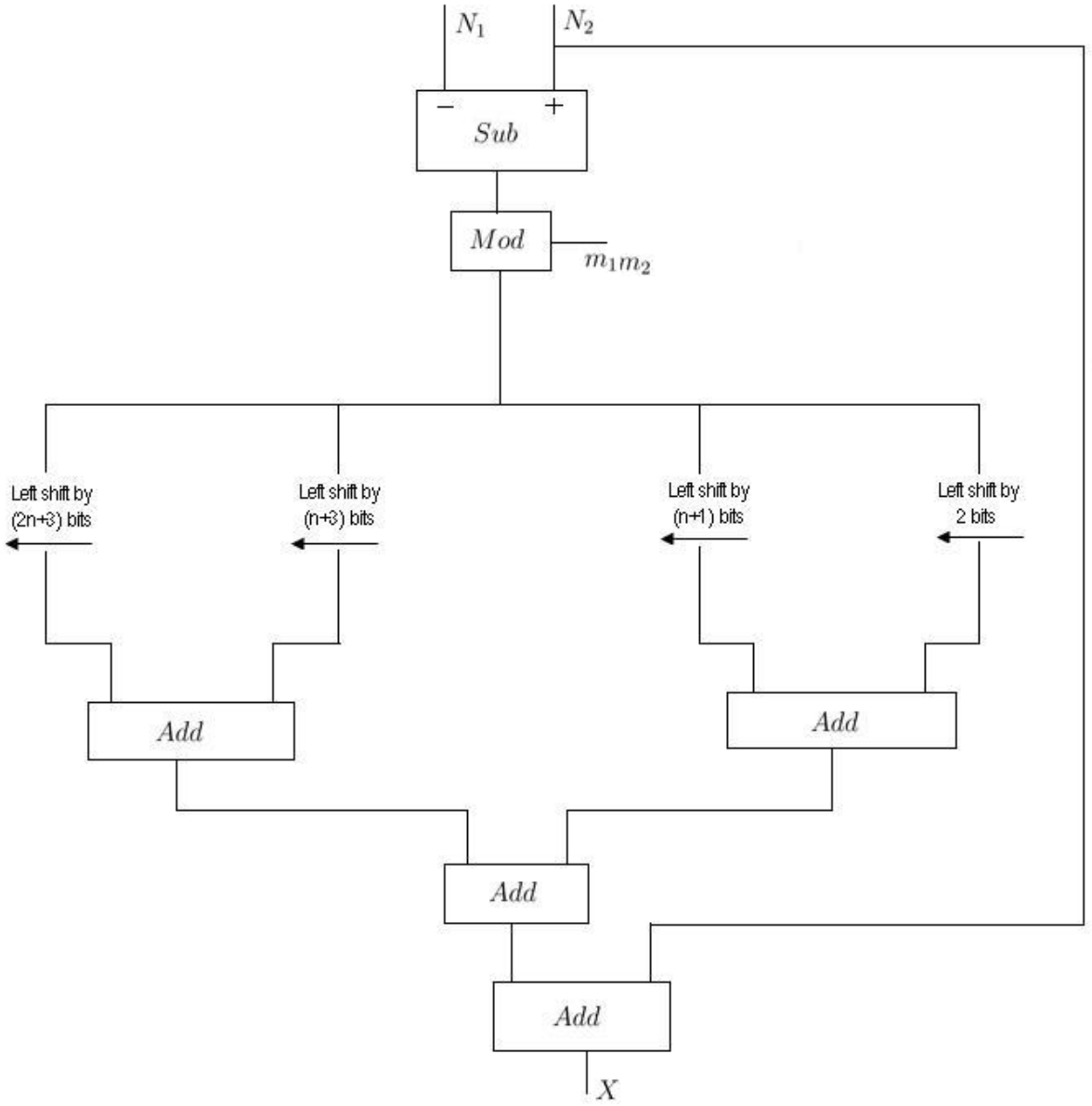


Fig.4.3. Hardware implementation of X by CRT II without using multipliers

4.4. Optimized Hardware Implementation of New Chinese Remainder Theorem III:

Using the moduli set from the section 3.5, $M = \{2^{n+2}+1, 2^{n+2}, 2^{n+1}+1, 2\}$, where $n = 1, 2, 3. . .$, and considering the optimized CRT III equation from the chapter 3,

$$X = X_1 + M_1 [(2^n +1) (X_2 – X_1)] \bmod m_3$$

where $M_1 = (2^{n+2}+1)*(2^{n+2}) = (2^{2(n+2)} + 2^{n+2})$

$$X = X_1 + (2^{2(n+2)} + 2^{n+2}) \ [(2^n+1)(X_2 - X_1)] \bmod m_3$$

$$X = X_1 + (2^{2(n+2)})*[(2^n+1)(X_2 - X_1)] \bmod m_3 + (2^{n+2})*[(2^n+1)(X_2 - X_1)] \bmod m_3 \quad (4.5)$$

The hardware implementation of the above equation without using multipliers is shown in the figure (4.4).

The total hardware requirements for the above implementation are as follows:

1) 1 subtractor
2) 1 modulus operator
3) 3 adders
4) 1 n-bit, 2(n+2)-bit left shift registers.

4.5. Comparison between Optimized and Non-Optimized CRT II Implementation:

Consider a moduli set $C = \{r, r^{n-3}+1, r^{n-1}+1, r^n+1\}$ where $r = 2$ and $n = 2k+1, k=2, 3, 4,\ldots$ from [5] and the moduli $M = \{2, 2^n+1, 2^{n+1}+1, 2^{n+2}+3\}$ where $n = 1, 2, 3$ from this paper for comparison, as the dynamic ranges of both sets is almost same. To implement CRT II using the moduli set C requires 3 subtractor units, 4 multiplier units, 3 adder units, 3 modulus operators and 3 inverse modulus operators. Similarly to implement CRT II using the optimized moduli set M, 3 subtractor units, 3 modulus operators, 9 adder units and 1 n-bit, 1 (n+1)-bit, 1 1-bit, 1 (2n+3)-bit, 1 (n+3)-bit, 1 (n+1)-bit and 1 2-bit left shift registers. The shift registers do not offer any time delay, and hence the optimized implementation of CRT II is faster and utilizes less hardware.

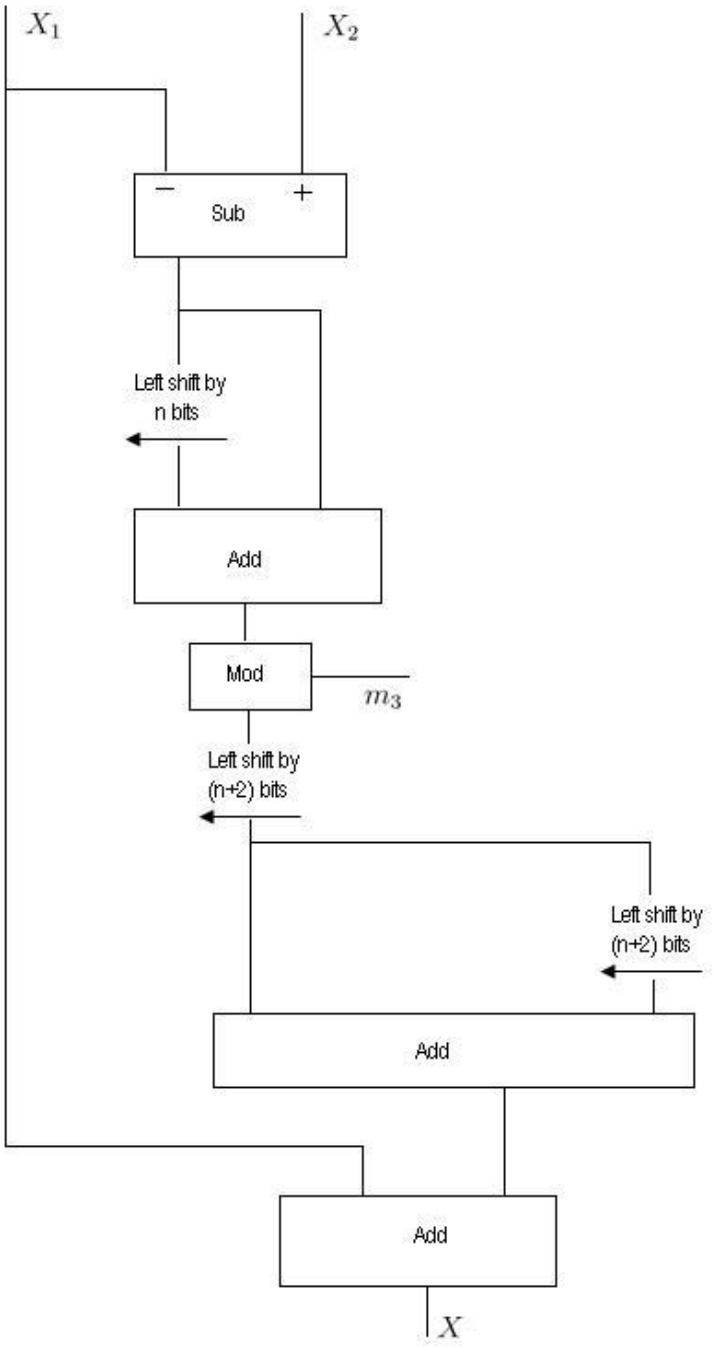


Fig.4.4. Hardware implementation of X by CRT III without using multipliers

4.6. Comparison between Optimized and Non-Optimized CRT III Implementation:

Consider a non co-prime moduli set G = $\{2^{n+2}+1, 2^{n+2}-1, 2^{n+1}+1, 2^{n+1}-1, 2^{n}+1, 2^{n}-1\}$ where n = odd, from [2] and the moduli set $N = \{2^{n+2}+1, 2^{n+2}, 2^{n+1}+1, 2\}$, where $n = 1, 2, 3$...for

comparison. To implement CRT III using the set G requires 1 subtractor, 4 multipliers, 3 dividers, 1 modulus operator and 1 inverse modulus operator. But the optimized implementation of CRT III using set N requires only 1 subtractor, 1 modulus operator, 3 adders and 1 n-bit, 2(n+2)-bit left shift registers. Hence, this comparison shows the optimized implementation of CRT III using set N offers faster computations than the non-optimized implementation.

5. CONCLUSION AND FUTURE WORK

The main goal of this research is to simplify the New Chinese Remainder Theorems, for their practical implementations in the real world. In this research, new 4 and 5 moduli sets are proposed that eliminate huge summation terms, inverse modulo terms and division terms. Furthermore, the hardware optimization using shift registers removes the usage of multipliers which in turn needs many half and full adders. The comparisons with other modulus sets in earlier papers clearly prove that the proposed modulus sets reduce area and time delay in the implementations. Therefore, the introduced CRT optimizations facilitate the efficient and cost effective hardware implementations of the New CRTs for various applications.

The future work that can come out of this research is that the proposed modulus sets can be extended to 8, 12 and 16 moduli sets with both co-prime and non co-prime moduli, so that they will simplify the new CRTs. Further, there can be radical improvements in the hardware implementations as well, by using CSAs, CPAs, and more VLSI designs.

REFERENCES

[1] Yuke Wang, "Residue-to-Binary Converters Based On New Chinese Remainder Theorems," *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, vol. 47, No. 3, pp.197–205, March 2000.

[2] Alexander Skavantzos and Yuke Wang, "New Efficient RNS-to-Weighted Decoders for Conjugate-Pair- Moduli Residue Number Systems," *in proceedings of the Thirty-Third Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1345 -1350, Nov. 1999.

[3] E. Al-Radadi and P.Siy, "Four-Moduli Set (2, 2^n-1, 2^n+2^{n-1}-1, $2^{n+1}+2^n$-1) Simplifies the Residue To Binary Converters Based on CRT II," *PERGAMON Computers and Mathematics with Applications*, vol. 44, no. 12, pp. 1581-1587, Dec. 2002.

[4] Y. Wang, "New Chinese Remainder Theorems," in Proceedings of the Thirty Second Asilomar Conference on Signals, Systems and Computers, vol. 1, pp. 165-171, Nov. 1998.

[5] Mohammad Abdallah and Alexander Skavantzos, "On MultiModuli Residue Number Systems with Moduli of Forms r^a, r^b-1, r^c+1," *IEEE Transactions On Circuits and Systems – I*: Regular Papers, vol. 52, No. 7, July 2005.

[6] Narendran Narayanaswamy, Alex Skavantzos, Thanos Stouraitis, "Optimal Modulus Sets for Efficient Residue-to-Binary Conversion Using the New Chinese Remainder Theorems" *Accepted in 17th IEEE International Conference on Electronics, Circuits, and Systems*, Athens, December 2010.

[7] Mohan, P.V.A. and Premkumar, A.B., "RNS-to-Binary Converters for Two Four-Moduli Sets $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$," *IEEE Transactions on Circuits and Systems I:* Regular Papers, vol.54, no.6, pp.1245-1254, June 2007.

[8] Skavantzos, A and Abdallah, M, "Implementation issues of the two-level residue number system with pairs of conjugate moduli," *IEEE Transactions on Signal Processing*, vol.47, no.3, pp.826-838, Mar 1999.

[9] Skavantzos. A., "An efficient residue to weighted converter for a new residue number system," VLSI, 1998. Proceedings of the 8th Great Lakes Symposium on , vol., no., pp.185-191, 19-21 Feb 1998

[10] Piestrak, S.J, "Design of residue generators and multioperand modular adders using carry-save adders," *IEEE Transactions on Computers*, vol.43, no.1, pp.68-77, Jan 1994.

[11] Wei Wang, Swamy, M.N.S., Ahmad, M.O. Yuke Wang , "A study of the residue-to-binary converters for the three-moduli sets," Circuits and Systems I: *IEEE Transactions on Fundamental Theory and Applications*, vol.50, no.2, pp. 235- 243, Feb 2003.

[12] Wei Wang, Swamy, M.N.S., Ahmad, M.O., Yuke Wang , "A note on "A high-speed residue-to-binary converter for three-moduli (2k 2k - 1, 2k-1 - 1) RNS and a scheme for its VLSI implementation," Circuits and Systems II: , *IEEE Transactions on Analog and Digital Signal Processing*, vol.49, no.3, pp.230-230, Mar 2002.

[13] Zhongde Wang, Jullien, G.A., Miller, W.C., "An improved residue-to-binary converter," Circuits and Systems I: *IEEE Transactions on Fundamental Theory and Applications*, vol.47, no.9, pp. 1437- 1440, Sep 2000.

[14] Wang, Y., Song, X., Aboulhamid, M., Shen, H., "Adder based residue to binary number converters for (2n-1, 2n, 2n+1)," *IEEE Transactions on Signal Processing*, vol.50, no.7, pp.1772-1779, Jul 2002.

[15] Cao, B., Chang, C.-H., Srikanthan, T., , "A Residue-to-Binary Converter for a New Five-Moduli Set," Circuits and Systems I: *IEEE Transactions on Regular Paper*, vol.54, no.5, pp.1041-1049, May 2007.

[16] Hiasat, A., "Efficient residue to binary converter," Computers and Digital Techniques, *IEE Proceedings* - , vol.150, no.1, pp. 11- 16, 20 Jan. 2003.

[17] Cao, B., Chang, C.-H., Srikanthan, T., "Adder based residue to binary converters for a new balanced 4-moduli set," Image and Signal Processing and Analysis, 2003. ISPA 2003. Proceedings of the 3rd International Symposium, vol.2, no., pp. 820- 825 Vol.2, 18-20 Sept. 2003.

[18] Das. A., "A novel efficient parallel algorithm for RNS to binary conversion for arbitrary moduli set," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1994. ICASSP-94., 1994, vol.ii, no., pp.II/485-II/488 vol.2, 19-22 Apr 1994.

[19]Skavantzos, A., *Dr. Alex Skavantzos:Home Page*. Retrieved December 20, 2009, from LSU Electrical and Computer Engineering: http://www.ece.lsu.edu/alex/index.html

APPENDIX A: INVERSE MODULUS OPERATION

This appendix deals with the description and calculation of inverse modulus values, which are used in most of the RNS to Binary conversions. This is an intensive operation because the hardware requirements to implement this are high and expensive. But, it is used in all traditional and new Chinese Remainder Theorems and Mixed Radix Conversions. There are many theorems to calculate multiplicative inverse modulus values, and some familiar ones are Euclidean algorithm, Euler Totient function, Euler and Fermat's theorems. Now let us deal with the basics and calculation of inverse modulus functions.

The multiplicative inverse of a number a mod b is denoted as a^{-1} *mod* b. The value of a^{-1} *mod* b exists if and only if a and b are co-prime to each other, that is, greatest common divisor of a and b is 1.

A.1. Methods to Calculate Inverse Modulus Values:

There are many ways to calculate multiplicative inverse values, out of which some are trial and error methods and some converge closely to the trial and error value. The following theorem mentioned in the third chapter 3 (repeated here for convenience) is very useful in many cases.

Theorem:

For co-prime numbers m_i, if $(m_1 \bmod m_2) = 1$, then $(m_1)^{-1} \bmod m_2 = 1$.

Refer to chapter 3 for the proof. The above theorem has the following 3 *lemmas* that refer to the moduli of a co-prime four-modulus set $M = \{m_1, m_2, m_3, m_4\}$:

Lemma 1: If $(m_3 \bmod m_4) = 1$, then $(m_3)^{-1} \bmod m_4 = 1$.

Lemma 2: If $(m_1 * m_2) \bmod (m_3 * m_4) = 1$, then $(m_1 * m_2)^{-1} \bmod (m_3 * m_4) = 1$.

Lemma 3: If $(m_1 * m_2 * m_3) \bmod (m_4) = 1$, then $(m_1 * m_2 * m_3)^{-1} \bmod (m_4) = 1$.

This theorem eliminates the chances of calculating inverse modulo values between two numbers, if their modulus value is 1.

The following property is also useful in some cases, where the numbers are huge. It helps in breaking the big numbers to smaller numbers.

Property: $a^{-1} \bmod b = [a \bmod b]^{-1} \bmod b$

Example 1: Find the value of $17^{-1} \bmod 11$.

$$17^{-1} \bmod 11 = [17 \bmod 11]^{-1} \bmod 11$$

$$= 6^{-1} \bmod 11$$

$$= 2$$

Method 1:

Let x be the value of the $a^{-1} \bmod b$. The value of x is such that,

$$[a * x] \bmod b = 1 \qquad \text{(A.1)}$$

The smallest value of x is the inverse modulus value.

Example 2: Find the inverse modulus value of 15 mod 11.

Using the equation (1), $[15 * x] \bmod 11 = 1$

$$x = 3$$

Example 3: Find the inverse modulus value of 13 mod 7.

Using the equation (A.1), *[13 * x] mod 7 = 1*

$$x = 6$$

Method 2:

This method is based on the Extended Euclidean algorithm. Extended Euclidean algorithm is the extension of Euclidean algorithm, which is mainly used when two numbers *a* and *b* are co-prime to each other. It states that,

$$ax + by = gcd\ (a, b) \qquad \text{(A.2)}$$

where a, b are integers.

By the definition of inverse modulus, *ax is congruent to (1 mod b)*

Hence, *b* is the divisor of *ax – 1*, and it gives the following,

$$ax - by = 1 \qquad \text{(A.3)}$$

In the above equation, *a* and *b* are the numbers whose inverse modulo value is to be found out, *x* is the inverse modulus value and *y* is an integer and it can be ignored.

Example 4: Find the value of 23^{-1} mod 7.

Using the equation A.3, and approaching trial and error method, it is found out that x =4 and y = 13. Ignoring the value of y, the value of 23^{-1} mod 7 is 4.

Example 5: Find the value of 14^{-1} mod 3.

Using the equation A.3, and approaching trial and error method, it is found out that x =2 and y = 9. Ignoring the value of y, the value of 14^{-1} mod 3 is 2.

Euler's Totient Function (Phi function):

Consider a positive integer *m*, and let *φ (m)* be the number of positive integers that are smaller than *m* and also co-prime to *m. φ(m)* is called Euler's totient function or phi function. Then,

- If *P* is a positive prime then *φ (P) =P-1*.

- If m = P^e, where m>0 and *P* = prime then $\varphi(P^e) = P^{e-1}(P-1)$.
- If $(a, b) = 1$, then $\varphi(a\,b) = \varphi(a)\,\varphi(b)$.
- If *m* is composite and $m = P_1^{e_1} \cdot P_2^{e_2} \ldots P_L^{e_L}$

  where $P_1, P_2 \ldots P_L$ are primes and $e_1, e_2 \ldots e_L$ are integers then,

$$\varphi(m) = \varphi\left(P_1^{e_1} \cdot P_2^{e_2} \ldots P_L^{e_L}\right) = P_1^{e_1-1}(P_1-1) \cdot P_2^{e_2-1}(P_2-1) \ldots P_L^{e_L-1}(P_L-1)$$

Euler's Theorem:

If $(a, m) = 1$ (*a* and *m* are relatively prime), $m > 0$, then,

$$[a^{\varphi(m)}] \bmod m = 1$$

Fermat's Theorem:

This is a special case of Euler's theorem. It states that if P is prime and (a, P) = 1, then

$$[a^{P-1}] \bmod P = 1$$

Usefulness of Euler's and Fermat's Theorems:

Euler's and Fermat's theorems are useful since they provide closed solution for computing the multiplicative inverse. Recalling Euler's theorem, $(a, m) = 1$, then $[a^{\varphi(m)}] \bmod m = 1$.

Thus, $a^{-1} \bmod m = [a^{\varphi(m)-1}] \bmod m$

Fermat's theorem states that if P is prime and (a, P) = 1, then $[a^{P-1}] \bmod P = 1$, thus,

$$a^{-1} \bmod P = a^{P-2} \bmod P$$

Example 6: Find the value of $6^{-1} \bmod 11$ using the above theorems.

Here $\gcd(6, 11) = 1$ and thus $6^{-1} \bmod 11$ exists. The number $P=11$ is prime and therefore using Fermat's theorem

$$6^{-1} \bmod 11 = 6^{(11-2)} \bmod 11 = 6^9 \bmod 11$$

$= ((6^2)^4.6) \bmod 11$

$= (3^4 * 6) \bmod 11$

$= (81 * 6) \bmod 11$

$= 2$

Example 7: Find the value of $5^{-1} \bmod 36$ using the above theorems.

Here $gcd(5, 36) = 1$ and thus $5^{-1} \bmod 36$ exists. The number 36 is composite and hence using Euler's theorem,

$$5^{-1} \bmod 36 = (5^{\varphi(36)-1}) \bmod 36$$

The value of $\varphi(m)$ where $m = 36$ can be found as follows,

$m = 36 = 2^2 * 3^2$; (*2* and *3* are primes)

Therefore $\varphi(36) = \varphi(2^2 * 3^2) = \varphi(2^2) * \varphi(3^2)$

$= 2^{2-1} * (2-1) * 3^{2-1} * (3-1)$

$= 12$

Thus there are 12 numbers smaller than 36, which are relatively prime to 36. They are 1, 4, 7, 11, 13, 17, 19, 23, 25, 29, 31, and 35. Therefore,

$(5^{\varphi(36)-1}) \bmod 36 = (5^{12-1}) \bmod 36$

$= ((5^3)^3 * 5^2) \bmod 36$

$= (125 \bmod 36)^3 * 5^2) \bmod 36$

$= (17^3 * 5^2) \bmod 36$

$= (17^2 * 17 * 5^2) \bmod 36$

$= (289 * 17 * 5^2) \bmod 36$

$= (17 * 5 * 5) \bmod 36 = 29$

A.2.Code in C to Calculate Multiplicative Inverse:

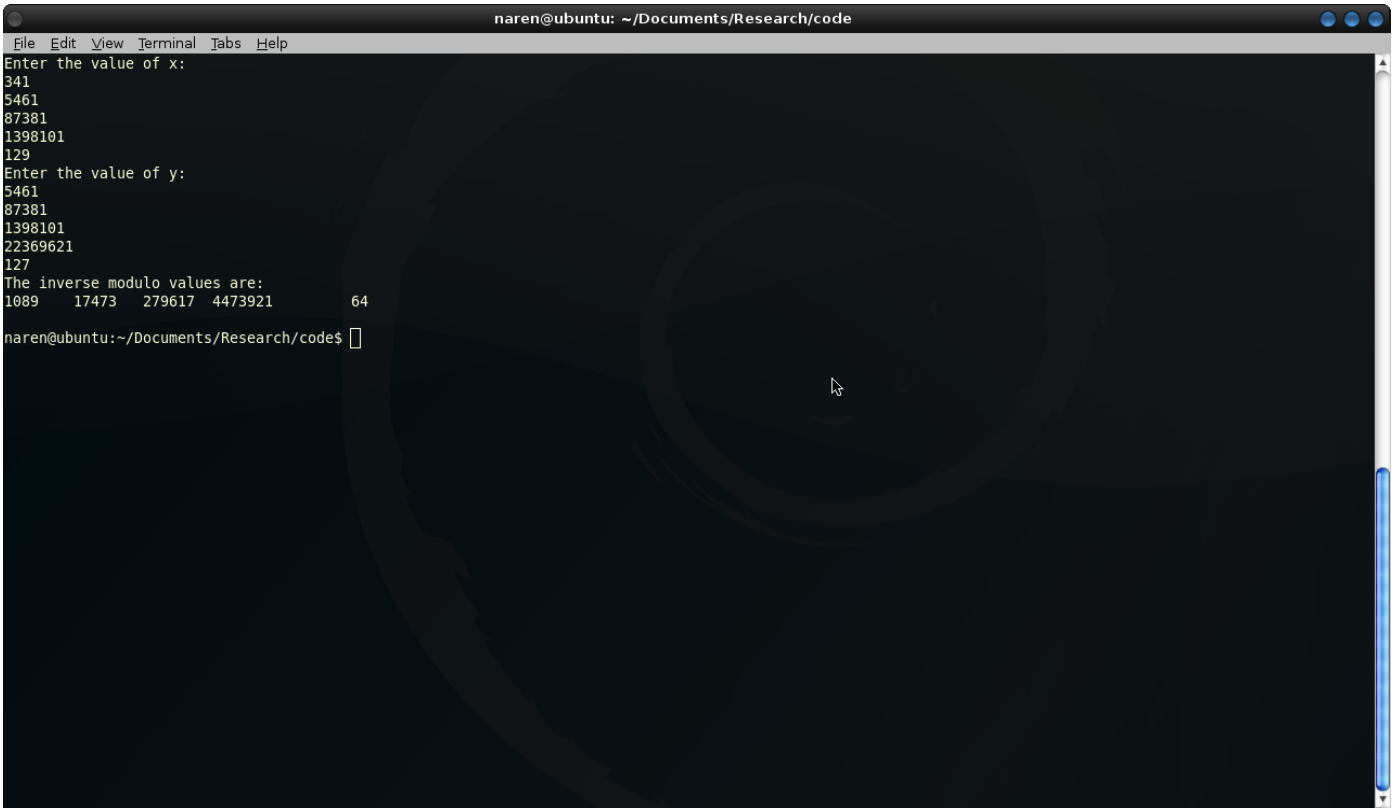
The following code in C is used to simulate the series of multiplicative inverse values, and is based on Extended Euclidean algorithm.

Code:

```
/* To calculate multiplicative inverse series */
#include<stdio.h>
main()
{
system("clear");
        int Inverse(int a[], int n[])
        {
        int k, i[5], v[5], d[5], t[5], x[5];
        for(k=0;k<5;k++)
                {
        i[k] = n[k], v[k] = 0, d[k] = 1;
                        while (a[k]>0)
                                {
                                t[k] = i[k]/a[k], x[k] = a[k];
                                a[k] = i[k] % x[k];
                                i[k] = x[k];
                                 x[k] = d[k];
                                d[k] = v[k] - t[k]*x[k];
                                v[k] = x[k];
                                }
                        v[k] %= n[k];
                        if (v[k]<0) v[k] = (v[k]+n[k])%n[k];
                        printf("%d \t",v[k]);
                        }
        printf("\n\n");
        }
int x[10], y[10],j;
```

```
printf("Enter the value of x:\n");
for(j=0;j<5;j++)
{
scanf("%d",&x[j]);
}
printf("Enter the value of y:\n");
for(j=0;j<5;j++)
{
scanf("%d",&y[j]);
}
printf("The inverse modulo values are:\n");
Inverse(x,y);
}
```

Result:



APPENDIX B: PROOFS FOR THE SETS TO BE PAIR WISE RELATIVELY PRIME

B.1. The Set $\{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$, where n= 2k + 1, k = 1, 2, 3… is Pair Wise Relatively Prime:

Proof:

It is known that 2^n-1, 2^n, 2^n+1 are relatively prime to each other.

Since n is odd, $2^n+1 \in S_1$ of [7]. And, since n-1 is even, $2^{n-1}+1 \notin S_1$ of [7]. Therefore, according to theorem 1 of [7], $(2^{n-1}+1, 2^n+1) = 1$.

Since n is odd, using theorem 3 of [7], we get $(2^{n-1}+1, 2^n-1) = 1$.

Therefore the set $\{2^{n-1}+1, 2^n-1, 2^n, 2^n+1\}$ is pair wise co-prime.

B.2 The Set $\{2^{n+2}+3, 2^{n+1}+1, 2^n+1, 2\}$, where n =1, 2, 3…is Pair Wise Relatively Prime:

Proof:

If n = odd, $2^n+1 \in S_1$, and n+1 is even, $2^{n+1}+1 \notin S_1$ [7].

If n = even, $2^n+1 \notin S_1$ and n+1 is odd, $2^{n+1}+1 \in S_1$ [7].

Therefore according to theorem 1 of [7], 2^n+1 and $2^{n+1}+1$are relatively prime.

Let *a/b* mean "*a* divides *b*."

It is true that, if *a/b* and *a/c*, then $a/[k_1b \pm k_2c]$. Let d be a common divisor of $2^{n+2}+3$ and $2^{n+1}+1$. Then $d/2^{n+2}+3$ and $d/2^{n+1}+1$implies $d/ [(2^{n+2}+3) – 2(2^{n+1}+1)$ and in turn implies d/1. Therefore d=1.

Thus the only common divisor of $2^{n+2}+3$ and $2^{n+1}+1$is 1 and therefore $2^{n+2}+3$ and $2^{n+1}+1$ are relatively prime.

At last, let d' be a common divisor of $2^{n+2}+3$ and 2^n+1. Then $d'/2^{n+2}+3$ and $d'/2^n+1$ implies $d'/ [2^2(2^n+1) – (2^{n+2}+3)]$ and in turn implies $d'/1$. Therefore $d' = 1$.

Thus the only common divisor of $2^{n+2}+3$ and $2^n+1$is 1 and therefore $2^{n+2}+3$ and 2^n+1 are relatively prime.

Therefore the set $\{2^{n+2}+3, 2^{n+1}+1, 2^n+1, 2\}$ is a co-prime set.

VITA

Narendran Narayanaswamy was born in Uduamlpet, a textile town in the state of Tamil Nadu, India. He finished his higher secondary schooling in 2004 with distinction. He received a Bachelor of Technology in Electronics and Communication Engineering degree with first class from Amrita School of Engineering, Coimbatore, India. Upon graduation, he enrolled in Louisiana State University, to pursue a master's degree in electrical and computer engineering. His areas of specialization are computer arithmetic and database management. He is also employed as graduate assistant as UNIX/system administrator in the Department of Physics and Astronomy, Louisiana State University.