Low Power Modulo Reduction Technique and Its Application in Residue-to-binary Converters

Shaoqiang Bi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Electrical Engineering) at
Concordia University
Montreal, Quebec, Canada

March 2004

Your file Votre référence
ISBN: 0-612-91002-4
Our file Notre référence
ISBN: 0-612-91002-4

ABSTRACT

Low Power Modulo Reduction Technique and
Its Application in Residue-to-binary Converters

Shaoqiang Bi

In this thesis, novel modulo reduction algorithms are proposed that considerably simplify a large modulo operation to the sum of a number of small modulo operations. By applying the proposed modulo reduction algorithms to the modified Chinese Remainder Theorem (CRT), the complexity of modulo operation in the modified CRT is reduced significantly. The modulo reduction technique and the modulo reduced modified CRT are applied to derive R/B algorithms for two existing three-moduli sets and four newly found three-moduli sets. A novel R/B converter for $N_1 = \{2^n, 2^n+1, 2^n-1\}$ with reduced modulo operations is proposed to show the efficiency of the proposed modulo reduction technique. Novel MUX-based designs of some components needed for constructing the new converter are developed. New unsigned and signed-2's complement incrementer/decrementer are designed to implement the operation of $Z \pm 1$. The new modulo incrementer and decrementer are developed to implement the operations of $|Z+1|_{2^n-1}$ and $|Z-1|_{2^n-1}$. Furthermore, a new modulo subtractor is proposed to conduct the modulo (2^n+1) subtraction. The complete architecture of the new converter is presented along with implementation. Based on the FPGA implementation results, a comparison study between the proposed R/B converter and the corresponding ones in the literature is carried out.

ACKNOWLEDGEMENT

It is my fortune that my professors provide me valuable suggestions and support for my graduate study in Concordia University. I am especially grateful to my supervisor Dr. Asim Al-Khalili, for his constant encouragement, enthusiastic help and generous support. It is his guidance that made me complete my graduate research. I also treasure very much the help from Dr. Wei Wang, who is with the Department of Electrical and Computer Engineering, the University of Western Ontario. Throughout my research, he has provided me numerous suggestions. I would like to express my appreciation to Yue Wang, who has been a great help to me during my research work. It is my pleasure to say thanks here to all my friends for their sincere help and kindness.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

ASIC	application specific integrated circuits
B/R	binary-to-residue
CPA	carry propagate adder
CRT	Chinese remainder theorem
CSA	carry save adder
DM	decision module
EAC	end-around-carry
FA	full adder
FPGA	field programmable gate arrays
GCD	greatest common divisor
HA	half adder
LSB	least significant bit
LSOB	least significant one bit
MRC	mixed radix algorithm
MSB	most significant bit
MUX	multiplixier
R/B	residue-to-binary
RNS	residue number system
VLSI	very large scale integrate circuit

$\lvert K\rvert_{P_i}$	modulo operation such as K mod P_i
$\lvert N_i^{-1}\rvert_{P_i}$	the multiplicative inverse of $\lvert N_i\rvert_{P_i}$
$\bar{x}$	complement operation
$<x><y>$	concatenate operation such as $<a><b>= a\times 2^n + b$
$\lfloor x \rfloor$	the largest integer less than or equal to x

CHAPTER 1

INTRODUCTION

1.1 General

The advance of very large-scale integration (VLSI) process technologies has made "system on a chip" feasible for very complex systems. This is partly responsible for the market demand for much shorter design cycles even though design complexity has continued to increase. VLSI finds many applications in the fields such as digital signal processing, computers, multimedia and wireless communication, video compression and so on [1], [2]. The layout of such VLSI digital systems can be performed with VLSI techniques such as Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuits (ASIC) [2], [3].

In the past, the major concerns of the VLSI designer were area, speed, and cost. Power consideration was typically of secondary importance. In recent years, however, this has begun to change and, increasingly, power is being given comparable weight to other design considerations. Several factors have contributed to this trend, including the remarkable success and growth of the class of battery-powered, personal computing devices and wireless communications systems that demand high-speed computation and

complex functionality with low power consumption. In these applications, extending the battery service life is a critical design concern. There also exists a significant pressure for producers of high-end products to reduce their power consumption. The main driving factors for lower power dissipation in these products are the cost associated with packaging and cooling as well as the circuit reliability.

System designers have started to respond to the requirement of power-constrained system designs by a combination of architectural improvements and advanced design automation methodologies and techniques for low power [4], [5]. In parallel, researchers and CAD tool developers have introduced a variety of models, algorithms, and techniques for reducing the power dissipation in VLSI circuits and systems in support of the low power optimization and synthesis techniques. A major low power technique at high-level abstraction is to introduce parallelism in the system and trade off the high speed to gain the low power consumption. The residue number system (RNS) is such a low power parallel system. It is known that the conventional weighted number system such as the binary number system and the decimal number system have a carry chain that often limits the speed. The RNS abates the carry chain problem by reducing the arithmetic operations from L-bit to log (L)-bit due to the reason that the digits in the RNS have no ordering significance, nor is there a requirement to manage carry information from digit to digit [6]-[8]. Thus, the RNS can result in a high-speed implementation. By trading off part of the speed gained by employing parallelism, power reduction can be achieved for VLSI digital design [9]-[12]. The RNS-based design has a promising future in the low-power FPGA and ASIC implementation of VLSI digital systems.

1.2 Low Power Scheme of RNS

The RNS system can reduce the power consumption by decreasing the speed but keeping the throughput [9]-[12]. This is to say that the intrinsic properties of RNS such as carry-free operations, parallelism and modularity are traded off for less power consumption.

What's more, the RNS structure reduces the switching activity of the system, which translates into a reduction in the power consumption [9]. Other benefits of using the RNS system are the significant reduction in the peak current and the increase in the design locality. The peak currents drawn in the RNS are sometimes five to six times lower than those in the conventional architectures [12]. Also, splitting a single data path into several residue subsystems increases the locality [9].

1.3 Objective of the Thesis

As shown in Fig.1.1, a general structure of an RNS consists of three parts: n binary-to-residue (B/R) converters, n parallel subsystems and a residue-to-binary converter. Given the moduli set $\{P_1, P_2, ..., P_n\}$, the binary system is partitioned into n subsystems, each corresponding to one modulus P_i. Each B/R converter consists of a modulo adder, while the R/B converter involves a lot of modulo operations. Thus, the residue-to-binary converter (R/B) is the most complicated part of a RNS system.

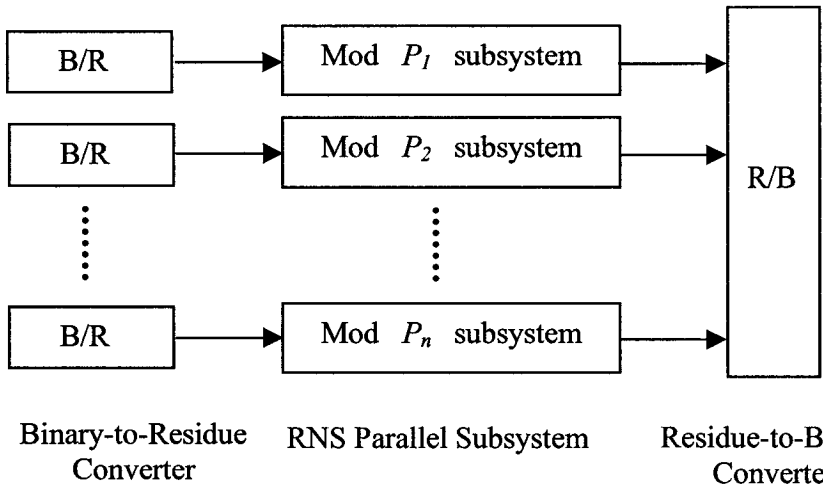


Figure 1.1. The Residue Number System

The R/B converter design is mainly based on the Chinese Remainder Theorem (CRT) that requires a large modulo inner product operation. In the literature, there exist extensive studies on the R/B conversion algorithms and R/B converter designs. Almost all of the R/B converters reported in the literature are based on the CRT directly [13]-[23] or indirectly [24]-[29]. In the critical path of the CRT-based R/B converter, the modulo part consumes a large portion of hardware and causes large delay and power consumption. Recently, new algorithms called modified Chinese Remainder Theorems (modified CRT) have been proposed to reduce the size of the modulo operation required by the CRT [24], [26]. For a moduli set $\{P_1, P_2, ..., P_n\}$, the CRT R/B conversion requires a modulo operation of $P_1, P_2, ..., P_n$, while the modified CRT requires a modulo operation of $P_2, ..., P_n$. The modified CRT is specifically useful for the design of the R/B converters for a number of three moduli sets, such as $\{2^n+1, 2^n, 2^n-1\}$ [25], [26], $\{2^n, 2^n-1, 2^{n-1}-1\}$[27], $\{2^{2n}+1, 2^n+1, 2^n-1\}$[28]. These three-moduli sets have received considerable

attention, due to the fact that the numbers in such systems can be efficiently scaled by any one of the moduli [29]. For these sets, based on the modified CRT, efficient R/B converter designs and implementations can be obtained [25]. However, the modified CRT only reduces the modulo operation by P_1. It is important to develop new algorithms and techniques to further reduce the modulo operation.

1.4 Contribution of the Thesis

In this thesis, novel modulo reduction algorithms are proposed to significantly simplify the large modulo operation to the sum of a number of small modulo operations. Based on the proposed algorithms, a modulo operation based on the product of $P_1P_2\cdots P_n$ can be divided to n individual modulo operations where each operation is based on one of the positive integers $P_1, P_2, \cdots, P_n$ respectively. By applying the proposed modulo reduction algorithms to the modified CRT, the complexity of modulo operation in the modified CRT is reduced to a great scale by partitioning the modulo operation with a large base to several individual modulo operations of small bases in parallel. The parallelism provides high concurrent operation and decreases delay. Then, the modulo reduction technique and the modulo reduced modified CRT are applied to derive R/B algorithms for the two existing three-moduli sets and four newly found three-moduli sets. The modulo operations required by these sets are efficiently reduced to one small modulo operation.

A novel multiplexer (MUX)-based algorithm for increment and decrement operations is proposed. The algorithm makes use of the mechanism of information transferring between the input Z and the output Y, resulting in a high-performance MUX-based binary

incrementer/decrementer. Based on this algorithm, new designs are introduced to improve both the unsigned and signed-2's complement incrementer/decrementer.

Based on the proposed MUX-based binary decrement technique, new modulo incrementer/decrementer that implements the operations of $|Z+1|_{2^n-1}$ and $|Z-1|_{2^n-1}$, are presented. Furthermore, the proposed MUX-based binary decrement technique is extended to derive modulo 2^n+1 subtractor. For the purpose of performance evaluation, the proposed binary and modulo MUX-based incrementer/decrementer are implemented using FPGA technology.

Based on the proposed modulo reduced R/B algorithm and the MUX-based modulo decrement technique, the design and FPGA implementation of the low-power R/B converter for the most popular three-moduli set N_1 is carried out. Based on the FPGA implementation results, a comparison study between the proposed R/B converter and the corresponding ones in the literature is carried out.

Most of the known solutions for modulo operations rely on end-around-carry (EAC) modulo addition. In this thesis, the unwanted race condition of the EAC modulo adders is also considered.

1.5 Organization of the Thesis

This thesis contains eight chapters. The rest of this thesis is organized as follows. In Chapter 2, we present the necessary background materials for completeness. The concept of RNS is given here. And the CRT and the modified CRT are also introduced.

In Chapter 3, we propose novel modulo reduction algorithms that reduce the base of modulo operation for two-moduli sets and general-moduli sets.

In Chapter 4, we apply the modulo reduction algorithms to improve the modified CRT and derive efficient modulo-reduced R/B algorithms for the two existing three-moduli sets and four newly found three-moduli sets. All these six new R/B algorithms reduce the modulo size compared to the modified CRT and provide new options to design high-speed R/B converters.

In Chapter 5, we present new algorithms and designs to implement novel MUX-based binary incrementer/decrementers for constructing the new R/B converter in Chapter 7. By extending the proposed MUX-based increment/decrement to FPGA design, we get an area-speed efficient FPGA implementation. Further, we carry out a comparison study between the proposed designs and the conventional adder-based incrementer/decrementer. Based on the study, we find that the proposed MUX-based designs are more efficient in terms of speed and hardware complexity compared to the adder-based ones for both unsigned and signed cases.

In Chapter 6, based on the proposed MUX-based binary decrement technique, new modulo incrementer/decrementer that implement the operations of $|Z+1|_{2^n-1}$ and $|Z-1|_{2^n-1}$, are presented. The modulo decrementer is used to construct the new R/B converter in Chapter 7. For the purpose of performance evaluation, the proposed modulo MUX-based incrementer/decrementer are implemented using FPGA technology.

In Chapter 7, the proposed MUX-based binary decrement technique is extended to derive modulo 2^n+1 subtractor that is a building block of the proposed converter. Based on the proposed modulo reduced R/B algorithm, the design and FPGA implementation of the low-power R/B converter for the most popular three-moduli set $N_1 = \{2^n, 2^n+1, 2^n-1\}$ is carried out.

Finally, in Chapter 8, we conclude with the highlights of the contributions of this thesis and suggest some possible further work.

CHAPTER 2

RESIDUE NUMBER SYSTEM

In this chapter, we present the necessary background materials for completeness. The concept of RNS is given here. The CRT and the modified CRT are also introduced.

2.1 Residue Number System

For any two numbers X and P_i, $x_i = X \bmod P_i$ is defined as $X = x_i + bP_i$ for some integer b such that $0 \le x_i < P_i$. For convenience, we denote $X \bmod P_i$ by $|X|_{P_i}$.

Residue Number System: Assuming n>1, a residue number system is defined in terms of a relatively prime moduli set $\{P_1, P_2, ..., P_n\}$, that is, $\mathrm{GCD}(P_i, P_j) = 1$ for $i \neq j$. A binary number X can be represented as $X = (x_1, x_2, ..., x_n)$, where $x_i = |X|_{P_i}$, $0 \le x_i < P_i$. Such a representation is unique for any integer $X \in [0, M-1]$, where $M = P_1 P_2 ... P_n$ is the dynamic range of the moduli set $\{P_1, P_2, ..., P_n\}$ [25].

The RNS is a carry-free system for addition, subtraction, and multiplication operations. We use the following example to illustrate the carry-free characteristic of the RNS.

Example 2.1: Given the moduli set {3, 7, 8, 11}, the residue format of the integer 18 is (0, 4, 2, 7) and that of the integer 20 is (2, 6, 4, 9). When the sum of (18 + 20) is

calculated in RNS, the corresponding residues are added in parallel. Without long carry chain, we have (2, 3, 6, 5), which is the RNS representation of the result 38.

	0	4	2	7
+)	2	6	4	9
	2	10	6	16
	2	3	6	5

The RNS provides, in a sense, carry-free arithmetic and thus the possibility of faster implementation. Hence, a large dynamic range binary system can be partitioned into several small wordlength channels in parallel. Thus, the RNS can result in a parallel and high-speed operation.

2.2 Chinese Remainder Theorem

In order to convert from binary to residue numbers and vice-versa, a B/R converter is required at the front-end of the system and a R/B converter at the back-end. The B/R converter consists of several modulo adders, while the R/B converter involves a lot of modulo operations. Thus, the R/B converter is a crucial part of the RNS system. To perform the R/B conversion, that is, to convert the residue number $(x_1, x_2, ..., x_n)$ into the binary number X, the CRT and mixed radix conversion (MRC) are widely used.

Chinese Remainder Theorem: The binary number X is computed by

$$X = \left| \sum_{i=1}^{n} N_i \left| N_i^{-1} \right|_{P_i} x_i \right|_M \tag{2.1}$$

where $n > 1$, $N_i = \frac{M}{P_i}$, and $\left|N_i^{-1}\right|_{P_i}$ is the multiplicative inverse of $\left|N_i\right|_{P_i}$, and is defined by the relationship[9]

$$\left| N_i^{-1} \right|_{P_i} N_i \Big|_{P_i} = 1 \tag{2.2}$$

Mixed Radix Conversion: The number X can be computed by the formula

$$X = \sum_{i=1}^{n} v_i a_i \tag{2.3}$$

where $n > 1$, $v_i = \prod_{j=1}^{i-1} P_j$ for $2 \le i \le n$, $v_1 = 1$, and a_i, called the *mixed radix digits*, are computed by the formulas: $Y_1 = X$, $Y_i = (Y_{i-1} - a_{i-1}) \left| P_{i-1}^{-1} \right|_{P_i}$, $a_i = |Y_i|_{P_i}$. The MRC approach is a sequential algorithm and is not as "parallel" as the CRT method. Thus, in order to solve the R/B conversion problem, the CRT schemes are preferred for efficient VLSI implementations. However, the CRT requires a binary inner product operation followed by a modulo M operation that is not very efficient. So we need to use a new formulation of the CRT that reduces the size of the modulo operation [26].

2.3 Modified Chinese Remainder Theorem

Modified Chinese Remainder Theorem [26]**:** Given the moduli set $\{P_1, P_2, \cdots, P_m\}$, the residue number $(x_1, x_2, \cdots, x_m)$ is converted into the binary number X by

$$X = x_1 + P_1 \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2 \cdots P_n} \tag{2.4}$$

where m>1,

$$w_1 = \frac{N_1 \left| N_1^{-1} \right|_{P_1} - 1}{P_1},$$

$$w_i = \frac{N_i}{P_1}, \text{ for i=2,3,...,m,}$$

$$x_1' = x_1,$$

$$x_i' = \left| N_i^{-1} x_i \right|_{P_i}, \text{ for i=2,3,...,m}$$

Comparing (2.1) and (2.4), we can see the modified CRT reduces the modulo base from $M = P_1 P_2 ... P_n$ to $P_2 ... P_n$. It is possible then to implement a converter with less hardware. However, if the modulo base can be reduced further, for example, from $P_2 ... P_n$ to $P_3 ... P_n$, the converter will be simplified further. In the case of $N_1 = \{2^n, 2^n+1, 2^n-1\}$, the modulo base is $2^n \times (2^n+1) \times (2^n-1)$ for CRT, while $(2^n+1) \times (2^n-1)$ for the modified CRT. It would be even better if the base can be reduced to 2^n+1 or 2^n-1. Being in the critical path of the converters, the modulo operation consumes a large portion of hardware and results in large delay. Thus, such a study is of importance.

In Chapter 3, we will propose a new theorem that reduces the base of the modulo operation. We need the following properties:

Lemma 1 $\left| A+B \right|_P = \left| \left| A \right|_P + \left| B \right|_P \right|_P$ for all integers A, B and P

Lemma 2 $\left| KP_1 \right|_{P_1 P_2} = P_1 \left| K \right|_{P_2}$ for all integers K, P_1 and P_2

Lemma 3 $\left| \left| K \right|_{P_1} \right|_{P_1 P_2} = \left| K \right|_{P_1}$ for all integers K, P_1 and P_2

Lemma 4 $2^n - 1 - X = \overline{x_{n-1}} \cdots \overline{x_0}$ for any non-zero n-bit binary number X

Lemma 5 $\left| -X \right|_{2^n-1} = \overline{x_{n-1}} \cdots \overline{x_0}$ for any non-zero n-bit binary number X

Lemma 6 $\left| X \times 2^{n_0} \right|_{2^n-1} = x_{n-n_0-1} \cdots x_0 x_{n-1} \cdots x_{n-n_0}$ for n-bit binary number X

CHAPTER 3

PROPOSED MODULO REDUCTION ALGORITHMS

3.1 Introduction

In this chapter, we propose novel modulo reduction algorithms to design small area, fast and low-power modulo circuits for the R/B converter design, B/R converters and RNS subsystems. Also, other modulo algorithms such as modulo division, modulo comparison and modulo encryption algorithm might benefit too. It is expected that the proposed modulo reduction algorithms have many applications in RNS study.

3.2 Proposed Modulo Reduction Algorithm for Two-Moduli

Proposition 3.1 Given any positive integers K, P_1 and P_2, we have

$$|K|_{P_1P_2} = P_1\left\|\left\lfloor\frac{K}{P_1}\right\rfloor\right\|_{P_2} + |K|_{P_1} \tag{3.1}$$

where $\left\lfloor\frac{K}{P_1}\right\rfloor$ is the floor of $\frac{K}{P_1}$.

Proof:

$$|K|_{P_1P_2} = \left|P_1\left\lfloor\frac{K}{P_1}\right\rfloor + |K|_{P_1}\right|_{P_1P_2}$$

$$= \left| P_1 \left\| \left\lfloor \frac{K}{P_1} \right\rfloor \right\|_{P_1P_2} + \left\| |K|_{P_1} \right|_{P_1P_2} \right|_{P_1P_2} \quad \text{by Lemma 1}$$

$$= \left| P_1 \left\| \left\lfloor \frac{K}{P_1} \right\rfloor \right\|_{P_2} + |K|_{P_1} \right|_{P_1P_2} \quad \text{by Lemma 2 and Lemma 3}$$

Since $P_1 \left\| \left\lfloor \frac{K}{P_1} \right\rfloor \right\|_{P_2} \le P_1(P_2 - 1)$ and $|K|_{P_1} < P_1$, we have $\left[P_1 \left\| \left\lfloor \frac{K}{P_1} \right\rfloor \right\|_{P_2} + |K|_{P_1} \right] < P_1P_2$. Thus

$$|K|_{P_1P_2} = P_1 \left\| \left\lfloor \frac{K}{P_1} \right\rfloor \right\|_{P_2} + |K|_{P_1}$$

■

The interesting part of Proposition 3.1 is that the modulo operation based on P_1P_2 is divided to two modulo operations. One is based on P_1. The other is based on P_2. What's more, the modulo operator is also decreased from K to $\left\lfloor \frac{K}{P_1} \right\rfloor$. When $P_1 = 2^n$, $|K|_{P_1} = |K|_{2^n}$ is just a truncation operation, namely $|K|_{2^n}$ is the n-bit least significant bits (LSBs) of K, whereas the floor operation $\left\lfloor \frac{K}{P_1} \right\rfloor = \left\lfloor \frac{K}{2^n} \right\rfloor$ is the remaining part after the truncation. Then, these two operations will not require any hardware resource in the VLSI implementation. Thus, in Proposition 3.1, the modulo base is reduced from P_1P_2 to P_2 without introducing any extra hardware.

We can see this point more clearly with the following collorary.

Collorary 3.1 For any positive integers n, K and P, we have

$$|K|_{2^nP} = 2^n \left\| \left\lfloor \frac{K}{2^n} \right\rfloor \right\|_P + |K|_{2^n} \tag{3.2}$$

where $\left\lfloor \frac{K}{2^n} \right\rfloor$ is the floor of $\frac{K}{2^n}$.

Proof:

Based on Proposition 3.1, we assume $P_1 = 2^n$ and $P_2 = P$ without loss of generality. We have

$$|K|_{2^n P} = 2^n \left\| \frac{K}{2^n} \right\|_P + |K|_{2^n}$$

■

In Collorary 3.1, with a m-bit integer K, $\left\lfloor \frac{K}{2^n} \right\rfloor$ is the $(m-n)$-bit most signinficant bits (MSBs) of K, while $|K|_{2^n}$ is the n-bit LSBs of K. It can be seen from Collorary 3.1 that the size of the modulo operation is reduced by 2^n, that is, from $2^n P$ to P. Also, the size of the modulo operator is reduced from m-bit K to (m–n)-bit $\left\lfloor \frac{K}{2^n} \right\rfloor$. What's more, the multiplication by 2^n and the subsequent binary addition of $|K|_{2^n}$ is a simple concatenation, that is, the modulo operation of base P produces the result, which forms the MSBs of $|K|_{2^n P}$, whereas $|K|_{2^n}$ forms the LSBs of $|K|_{2^n P}$. The reduction in the size of the modulo operation and that of the modulo operator will result in a saving of the hardware resource for a VLSI implementation. It is noticeable that the concatenation and the calculation of $|K|_{2^n}$ do not consume any hardware resources. Thus, Collorary 3.1 is useful for the VLSI implementation of R/B converters to reduce the size of the modulo operation.

3.3 Proposed Modulo Reduction Algorithm for General-Moduli

Proposition 3.2 Given integers $K, P_1, P_2, \cdots, P_n$, where n >1, we have

$$|K|_{P_1P_2\cdots P_n} = \sum_{m=1}^{n-1}\left(\left\lfloor \frac{K}{\prod_{i=1}^{m} P_i} \right\rfloor_{P_{m+1}} \prod_{i=1}^{m} P_i\right) + |K|_{P_1} \tag{3.3}$$

Proof: (Proved by mathematical induction)

(1) Base step:

Since n >1, let n = 2. According to Proposition 3.1, we have

$$|K|_{P_1P_2} = P_1\left\lfloor \frac{K}{P_1} \right\rfloor_{P_2} + |K|_{P_1}$$

When n = 3, we have

$$|K|_{P_1P_2P_3} = P_1P_2\left\lfloor \frac{K}{P_1P_2} \right\rfloor_{P_3} + |K|_{P_1P_2}$$

$$= P_1P_2\left\lfloor \frac{K}{P_1P_2} \right\rfloor_{P_3} + P_1\left\lfloor \frac{K}{P_1} \right\rfloor_{P_2} + |K|_{P_1}$$

$$= \sum_{m=1}^{2}\left(\left\lfloor \frac{K}{\prod_{i=1}^{m} P_i} \right\rfloor_{P_{m+1}} \prod_{i=1}^{m} P_i\right) + |K|_{P_1}$$

Thus, Proposition 3.2 holds for n =2 and n =3.

(2) Induction step:

Assumption: Proposition 3.2 is true when n = W, where W is a positive integer and W > 1. That is,

$$|K|_{P_1P_2\cdots P_W} = \sum_{m=1}^{W-1}\left(\left\lfloor \frac{K}{\prod_{i=1}^{m} P_i} \right\rfloor_{P_{m+1}} \prod_{i=1}^{m} P_i\right) + |K|_{P_1}$$

We need to show that Proposition 3.2 holds for n = W+1. That is,

$$|K|_{P_1P_2\cdots P_WP_{W+1}} = \sum_{m=1}^{W}\left(\left\lfloor \frac{K}{\prod_{i=1}^{m} P_i} \right\rfloor_{P_{m+1}} \prod_{i=1}^{m} P_i\right) + |K|_{P_1}$$

Proof for induction step:

$$|K|_{P_1P_2\cdots P_WP_{W+1}} = P_1P_2\cdots P_W\left\lfloor \frac{K}{P_1P_2\cdots P_W} \right\rfloor_{P_{W+1}} + |K|_{P_1P_2\cdots P_W}$$

$$= \left\lfloor \frac{K}{\prod_{i=1}^{W} P_i} \right\rfloor_{P_{W+1}} \prod_{i=1}^{W} P_i + \sum_{m=1}^{W-1}\left(\left\lfloor \frac{K}{\prod_{i=1}^{m} P_i} \right\rfloor_{P_{m+1}} \prod_{i=1}^{m} P_i\right) + |K|_{P_1}$$

$$= \sum_{m=1}^{W}\left(\left\lfloor \frac{K}{\prod_{i=1}^{m} P_i} \right\rfloor_{P_{m+1}} \prod_{i=1}^{m} P_i\right)_{P_{m+1}} + |K|_{P_1}$$

Thus, we have shown that Proposition 3.2 holds for n = W + 1 under the assumption that Proposition 3.2 holds for n = W.

In conclusion, from the base step and induction step, Proposition 3.2 holds for any positive integer that is greater than 1.

■

Proposition 3.2 is the general case of Proposition 3.1. The modulo base in Proposition 3.2 is a product of n positive integers instead of a product of only two positive integers as in Proposition 3.1.With Proposition 3.2, it is seen that a modulo operation based on the product of n positive integers $P_1, P_2, \cdots, P_n$ can be divided to n individual modulo operations where each operation is based on one of the positive integers $P_1, P_2, \cdots, P_n$. The method considerably simplifies the modulo operation based on the product of n positive integers $P_1, P_2, \cdots, P_n$. What's more, using Proposition 3.2, a modulo operation with large base can be partitioned into several small wordlength channels in parallel. Thus, Proposition 3.2 can result in a parallel and high-speed operation.

We give the following example to illustrate how Proposition 3.2 works.

Example 3.1: For a modulo operation $|1099|_{120}$ and four small integers $2\times3\times4\times5=120$, we have

$$|1099|_{120}=19$$

Using Proposition 3.2, we have

$$|1099|_{2\times3\times4\times5}=2\times3\times4\times\left\|\frac{1099}{2\times3\times4}\right\|_5+2\times3\times\left\|\frac{1099}{2\times3}\right\|_4+2\times\left\|\frac{1099}{2}\right\|_3+|1099|_2$$

$$=0+18+0+1=19$$

With Proposition 3.2, we use four parallel small size modulo operations to replace one big size modulo operation. Proposition 3.2 provides us with a very high concurrent operation, thus resulting in very high-speed and low-power VLSI implementation.

The proposed modulo reduction algorithms, namely, Propositions 3.1 and 3.2, provide a novel way to design size-reduced, fast and low-power modulo circuits for the R/B converter design, B/R converters and RNS subsystems. Also, other modulo algorithms such as modulo division, modulo comparison and modulo encryption algorithm might benefit too. It is expected that the proposed modulo reduction algorithms have many applications in RNS study.

3.4 Summary

In this chapter, we have proposed novel modulo reduction algorithms that reduce the base of modulo operation for two-moduli sets and general-moduli sets. The proposed algorithms significantly simplify the large modulo operation to the sum of a number of small modulo operations. Based on the proposed algorithms, a modulo operation based on the product of $P_1 P_2 \cdots P_n$ can be divided to n individual modulo operations where each operation is based on one of the positive integers $P_1, P_2, \cdots, P_n$ respectively.

CHAPTER 4

MODULO REDUCTION IN R/B CONVERSION

4.1 Introduction

In this chapter, we first apply the proposed modulo reduction technique of Chapter 3 to simplify the modulo operation with large base in the modified CRT. Then, based on the proposed modulo reduced modified CRT, we derive new and efficient R/B conversion algorithms for six three-moduli sets in form of $\{P_1, 2^n, P_3\}$.

4.2 Application of Modulo Reduction to Modified CRT

Theorem 4.1 Given the moduli set $\{P_1, P_2, \cdots, P_n\}$, the residue number $(x_1, x_2, \cdots, x_n)$ is converted into the binary number X by

$$X = x_1 + \sum_{m=1}^{n-2} \left(\left\lfloor \frac{\sum_{i=1}^{n} w_i x_i'}{\prod_{i=2}^{m+1} P_i} \right\rfloor_{P_{m+2}} \prod_{i=1}^{m+1} P_i \right) + P_1 \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2} \qquad (4.1)$$

where n>1,

$$w_1 = \frac{N_1 \left| N_1^{-1} \right|_{P_1} - 1}{P_1},$$

$$w_i = \frac{N_i}{P_1}, \text{ for i } = 2,3,\ldots,n$$

$$x_1' = x_1,$$

$$x_i' = \left| N_i^{-1} x_i \right|_{P_i}, \text{ for i} = 2,3,\ldots,n$$

Proof:

Based on the modified CRT [26], we have

$$X = x_1 + P_1 \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2 P_3 \cdots P_n}$$

where n >1

$$w_1 = \frac{N_1 \left| N_1^{-1} \right|_{P_1} - 1}{P_1},$$

$$w_i = \frac{N_i}{P_1}, \text{ for i=2,3,}\ldots,n$$

$$x_1' = x_1,$$

$$x_1' = \left| N_i^{-1} x_i \right|_{P_i}, \text{ for i=2,3,}\ldots,n$$

Using Proposition 3.2, we have

$$X = x_1 + P_1 \left[\sum_{m=1}^{n-2} \left(\left\| \frac{\sum_{i=1}^{n} w_i x_i'}{\prod_{i=2}^{m+1} P_i} \right\|_{P_{m+2}} \prod_{i=2}^{m+1} P_i \right) + \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2} \right]$$

$$= x_1 + \sum_{m=1}^{n-2} \left(\left\| \frac{\sum_{i=1}^{n} w_i x_i'}{\prod_{i=2}^{m+1} P_i} \right\|_{P_{m+2}} \prod_{i=1}^{m+1} P_i \right) + P_1 \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2}$$

■

Collorary 4.1: Given the moduli set $\{P_1, P_2, P_3 \cdots, P_n\}$, where $P_2 = 2^k$, the residue number $(x_1, x_2, \cdots, x_n)$ is converted into the binary number X by

$$X = x_1 + \sum_{m=2}^{n-2} \left(\left\| \frac{\sum_{i=1}^{n} w_i x_i'}{\prod_{i=2}^{m+1} P_i} \right\|_{P_{m+2}} \prod_{i=1}^{m+1} P_i \right) + P_1 \left[2^k \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^k} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^k} \right] \tag{4.2}$$

where n>1,

$$w_1 = \frac{N_1 \left| N_1^{-1} \right|_{P_1} - 1}{P_1},$$

$$w_i = \frac{N_i}{P_1}, \text{ for i } = 2,3,\ldots,n$$

$$x_i' = x_1,$$

$$x_i' = \left| N_i^{-1} x_i \right|_{P_i}, \text{ for i} = 2,3,\ldots,n$$

Proof:

Based on Theorem 4.1, let $P_2 = 2^k$, we have

$$X = x_1 + \sum_{m=2}^{n-2} \prod_{i=1}^{m+1} P_i \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{\prod_{i=2}^{m+1} P_i} \right\|_{P_{m+2}} + P_1 P_2 \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{P_2} \right\|_{P_3} + P_1 \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2}$$

$$= x_1 + \sum_{m=2}^{n-2} \prod_{i=1}^{m+1} P_i \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{\prod_{i=2}^{m+1} P_i} \right\|_{P_{m+2}} + P_1 \left[2^k \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^k} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^k} \right]$$

■

Theorem 4.1 and Collorary 4.1 can be considered as the further development of the modified CRT by using the proposed modulo reduction algorithms. The complexity of modulo operation in CRT and the modified CRT is reduced significantly by partitioning the modulo operation with a large base to several individual modulo operations of small bases in parallel. Based on Theorem 4.1, R/B converters can be obtained with higher performance in terms of area and delay than CRT and the modified CRT due to its parallelism and modularity. The parallelism provides high concurrent operation and decreases delay. And by choosing the bases of several individual modulo operations with similar magnitude, we can increase the modularity and further reduce area.

We use the following example to compare the CRT, Modified CRT and Theorem 4.1 and Collorary 4.1.

Example 4.1: Given the moduli set {9, 8, 7, 5}, the residue number (7, 4, 3, 2) is converted to the binary number format *X*.

Using the CRT, we get $N_1 = 280$, $N_2 = 315$, $N_3 = 360$, $N_4 = 504$, $\left|N_1^{-1}\right|_9 = 10$, $\left|N_2^{-1}\right|_8 = 3$, $\left|N_3^{-1}\right|_7 = 5$, and $\left|N_4^{-1}\right|_5 = 4$. The modulo base is $9 \times 8 \times 7 \times 5 = 2520$.

$$\begin{aligned} X &= \left|280 \times 10 \times x_1 + 315 \times 3 \times x_2 + 360 \times 5 \times x_3 + 504 \times 4 \times x_4\right|_{2520} \\ &= \left|280 \times 10 \times 7 + 315 \times 3 \times 4 + 360 \times 5 \times 3 + 504 \times 4 \times 2\right|_{2520} \\ &= \left|32812\right|_{2520} = 52 \end{aligned}$$

Using the modified CRT, we get $w_1 = 311$, $w_2 = 35$, $w_3 = 40$, $w_4 = 56$, $\left|N_1^{-1}\right|_9 = 10$, $\left|N_2^{-1}\right|_8 = 3$, $\left|N_3^{-1}\right|_7 = 5$, and $\left|N_4^{-1}\right|_5 = 4$. The modulo base is $8 \times 7 \times 5 = 280$.

$$\begin{aligned} X &= x_1 + 9 \times \left|311 \times x_1 + 35 \times 3 \times x_2 + 40 \times 5 \times x_3 + 56 \times 4 \times x_4\right|_{280} \\ &= 7 + 9 \times \left|311 \times 7 + 35 \times 3 \times 4 + 40 \times 5 \times 3 + 56 \times 4 \times 2\right|_{280} \end{aligned}$$

$$= 7 + 9 \times |3645|_{280} = 7 + 9 \times 5 = 52$$

Using Theorem 4.1 and Collorary 4.1, we get $w_1 = 311$, $w_2 = 35$, $w_3 = 40$, $w_4 = 56$, $|N_1^{-1}|_9 = 10$, $|N_2^{-1}|_8 = 3$, $|N_3^{-1}|_7 = 5$, and $|N_4^{-1}|_5 = 4$. Thus,

$$\sum_{i=1}^{n} w_i x_i' = 311 \times 7 + 35 \times 3 \times 4 + 40 \times 5 \times 3 + 56 \times 4 \times 2 = 3645$$

$$X = x_1 + P_1 P_2 P_3 \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{P_2 P_3} \right\|_{P_4} + P_1 \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

$$= 7 + 9 \times 8 \times 7 \times \left\| \frac{3645}{8 \times 7} \right\|_5 + 9 \times 8 \times \left\| \frac{3645}{8} \right\|_7 + |3645|_8 = 52$$

The above R/B conversion requires one modulo-2520 operation by using the CRT and one modulo-280 operation if using the modified CRT. By using Theorem 4.1 and Collorary 4.1, the same R/B conversion requires only three small size operations: modulo-5, modulo-7 and modulo-8. Compared to the CRT, the proposed modulo-reduced modified CRT decreases the size of modulo operation from 12-bit to 4-bit. Compared to the modified CRT, the modulo reduced modified CRT decreases the modulo size from 9-bit to 4-bit.

4.3 Application of Modulo Reduction to R/B Conversion for Three-Moduli Sets

We now apply the modulo reduction technique and the modulo reduced modified CRT to derive R/B algorithms for the two existing three-moduli sets $\{2^n+1, 2^n, 2^n-1\}$, $\{2^{n-1}-1, 2^n, 2^n-1\}$ and four newly found three-moduli sets: $\{2^{2n}+1, 2^n, 2^n+1\}$,

$\{2^{2n}+1,\ 2^n,\ 2^n-1\}$, $\{2^{2n}+1,\ 2^n,\ 2^{2n}-1\}$ and $\{2^{n+1}+1,\ 2^n,\ 2^{n+1}-1\}$. All these three-moduli sets are chosen in form of $\{P_1,\ 2^n,\ P_3\}$ due to two reasons. First, according to the proposed modulo reduction technique, we know that the R/B conversion in form of $\{P_1,\ 2^n,\ P_3\}$ can be reduced to one simple modulo operation based on P_3. Secondly, in the above three-moduli sets, P_3 has the form of 2^n-1 or 2^n+1. The moduli of the forms of 2^n-1 or 2^n+1 are referred to as the low-cost moduli [26]. The modulo operations based on these moduli can be simply implemented with EAC adders.

Before deriving the R/B algorithms for the six cases, we now present Corallary 4.2 for the three-moduli set $\{P_1,\ 2^n,\ P_3\}$, which is the three-moduli case of Corallary 4.1.

Collorary 4.2: Given the moduli set $\{P_1,P_2,P_3\}$, where $P_2 = 2^k$, the residue number (x_1,x_2,x_3) is converted into the binary number X by

$$X = x_1 + P_1\left[\left\|2^k\left\|\frac{\sum_{i=1}^{3} w_i x_i'}{2^k}\right\|_{P_3} + \left|\sum_{i=1}^{3} w_i x_i'\right|_{2^k}\right] \tag{4.3}$$

where n>1,

$$w_1 = \frac{N_1\left|N_1^{-1}\right|_{P_1} - 1}{P_1},$$

$$w_i = \frac{N_i}{P_1}, \text{ for i } = 2,3,\ldots,n$$

$$x_i' = x_1,$$

$$x_i' = \left|N_i^{-1}x_i\right|_{P_i}, \text{ for i} = 2,3,\ldots,n$$

Proof:

Based on Collarary 4.1, let $n=3$, we have

$$X = x_1 + \sum_{m=2}^{n-2}\prod_{i=1}^{m+1} P_i \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{\prod_{i=2}^{m+1} P_i} \right\|_{P_{m+2}} + P_1 \left[2^k \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^k} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^k} \right]$$

$$= x_1 + P_1 \left[2^k \left\| \frac{\sum_{i=1}^{3} w_i x_i'}{2^k} \right\|_{P_3} + \left| \sum_{i=1}^{3} w_i x_i' \right|_{2^k} \right]$$

■

The following R/B algorithms for six three-moduli sets are derived based on Collorary 4.2 in a unified manner. The modulo operation of the R/B algorithms are simplified to modulo P_3.

Proposition 4.1: For $N_1=\{2^n+1, 2^n, 2^n-1\}$, we have

$$X = x_1 + (2^n+1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^n-1} + |K|_{2^n}$$

$$K = (2^{2n-1}-1)x_1 + (2^n-1)^2 x_2 + 2^{2n-1} x_3$$

Proof:

Based on Collorary 4.2, we have

$$X = x_1 + P_1 \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

$$= x_1 + (2^n + 1)\left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{2^n-1} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

It is easy to see that $K = \sum_{i=1}^{n} w_i x_i' = (2^{2n-1} - 1)x_1 + (2^n - 1)^2 x_2 + 2^{2n-1} x_3$ [26]. Then, we have

$$X = x_1 + (2^n + 1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^n-1} + |K|_{2^n}$$

■

We use the following example to illustrate the improvement that the proposed method provides over the modified CRT for this three-moduli set.

Example 4.2: For a R/B converter with 8-bit dynamic range based on the moduli set $N_1 = \{2^n + 1, 2^n, 2^n - 1\}$, the specific moduli set {9,8,7} is chosen when n=3, since $9\times8\times7=504>2^8=256$. Randomly choose a number from 0 to 255, for example, X=169. Its RNS representation $X=(x_1, x_2, x_3)$ is (7,1,1).

Based on the modified CRT, we have

$$\begin{aligned} Y &= \left| (2^{2n-1} - 1)x_1 + (2^n - 1)^2 x_2 + 2^{2n-1} x_3 \right|_{2^n(2^n-1)} \\ &= \left| (2^{2\times3-1} - 1)\times 7 + (2^3 - 1)^2 \times 1 + 2^{2\times3-1} \times 1 \right|_{2^3(2^3-1)} \\ &= |298|_{56} = 18 \end{aligned}$$

$$X = x_1 + (2^n + 1)\times Y = 7 + 9\times18 = 169$$

Based on Proposition 4.1, we have

$$K = (2^{2n-1}-1)x_1 + (2^n-1)^2 x_2 + 2^{2n-1}x_3$$
$$= (2^{2\times3-1}-1)\times7 + (2^3-1)^2\times1 + 2^{2\times3-1}\times1 = 298$$
$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^n-1} + |K|_{2^n}$$
$$= 2^3 \times \left\| \frac{298}{2^3} \right\|_7 + |298|_8$$
$$= 2^3 \times |37|_7 + |298|_8$$

The binary representation of 298 is $(100101010)_2$, thus $298\times2^{-3} = (100101.010)_2$. And $(100101)_2$ is equal to 37, while $(010)_2$ is the binary representation of $|298|_8$. Then, we have

$$Y = 2^3 \times |(100101)_2|_7 + (010)_2$$
$$= 2^3 \times (010)_2 + (010)_2$$
$$= (010010)_2 = 18$$
$$X = x_1 + (2^n+1)\times Y = 7 + 9\times18 = 169$$

When using the modified CRT, the above R/B conversion needs a modulo 56 operation. By using Proposition 4.1 which is derived from Theorem 4.1, the modulo size is reduced from 56 to 7. That is to say, the length of modulo operation is reduced from 6-bit to 3-bit. Also, the modulo operator is decreased from 298 to 37, reduced by 3-bit. The 3-bit LSBs of Y are just the same 3-bit LSBs of 298. And the operations of multiplication by 2^3 and addition to $(010)_2$ correspond to a simple concatenation operation.

Proposition 4.2: For $N_2=\{2^{2n}+1, 2^n, 2^n+1\}$, we have

$$X = x_1 + (2^{2n}+1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^n+1} + |K|_{2^n}$$

$$K = (2^{2n-1} - 1)x_1 + (2^n + 1)x_2 + 2^{2n-1}x_3$$

Proof:

Based on Collorary 4.2, we have

$$X = x_1 + P_1 \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

$$= x_1 + (2^{2n} + 1) \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{2^n+1} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

It is easy to see that $K = \sum_{i=1}^{n} w_i x_i' = (2^{2n-1} - 1)x_1 + (2^n + 1)x_2 + 2^{2n-1}x_3$ [26]. Then, we have

$$X = x_1 + (2^{2n} + 1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^n+1} + |K|_{2^n}$$

■

We use the following example to illustrate the improvement that the proposed method provides over the modified CRT for this three-moduli set.

Example 4.3: For a R/B converter with 8-bit dynamic range based on the moduli set $N_2=\{2^{2n}+1, 2^n, 2^n+1\}$, the specific moduli set {17,4,5} is chosen when n=2, since

$17\times5\times4=340>2^8=256$. Randomly choose a number from 0 to 255, for example, X=38. Its RNS representation $X=(x_1, x_2, x_3)$ is (4,2,3).

Based on the modified CRT, we have

$$\begin{aligned} Y &= \left|(2^{2n-1}-1)x_1 + (2^n+1)x_2 + 2^{2n-1}x_3\right|_{2^n(2^n+1)} \\ &= \left|(2^{2\times2-1}-1)\times4 + (2^2+1)\times2 + 2^{2\times2-1}\times3\right|_{2^2(2^2+1)} \\ &= |62|_{20} = 2 \\ X &= x_1 + (2^{2n}+1)\times Y = 4+17\times2 = 38 \end{aligned}$$

Based on Proposition 4.2, we have

$$\begin{aligned} K &= (2^{2n-1}-1)x_1 + (2^n+1)x_2 + 2^{2n-1}x_3 \\ &= (2^{2\times2-1}-1)\times4 + (2^2+1)\times2 + 2^{2\times2-1}\times3 = 62 \\ Y &= 2^n\left\| \frac{K}{2^n} \right\|_{2^n-1} + |K|_{2^n} \\ &= 2^2\times\left\| \frac{62}{2^2} \right\|_5 + |62|_4 \\ &= 2^2\times|15|_5 + |62|_4 \end{aligned}$$

The binary representation of 62 is $(111110)_2$, thus $62\times2^{-2}=(1111.10)_2$. And $(1111)_2$ is equal to 15, while $(10)_2$ is the binary representation of $|62|_4$. Then, we have

$$\begin{aligned} Y &= 2^2\times|(1111)_2|_5 + (10)_2 \\ &= 2^2\times(000)_2 + (10)_2 \\ &= (00010)_2 = 2 \\ X &= x_1 + (2^{2n}+1)\times Y = 4+17\times2 = 38 \end{aligned}$$

When using the modified CRT, the above R/B conversion needs a modulo 20 operation. By using Proposition 4.2 which is derived from Theorem 4.1, the modulo size

is reduced from 20 to 5. That is, the length of modulo operation is reduced from 5-bit to 3-bit. Also, the modulo operator is decreased from 62 to 15, reduced by 2-bit. The 2-bit LSBs of Y are just the same 2-bit LSBs of 62. Thus, the operations of multiplication by 2^2 and addition to $(10)_2$ correspond to a simple concatenation operation.

Proposition 4.3: For $N_3=\{2^n-1,\ 2^n, 2^{n-1}-1\}$, we have

$$X = x_1 + (2^n - 1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^{n-1}-1} + |K|_{2^n}$$

$$K = (2^{2n-1} - 2^{n+1} + 1)x_1 + (2^{2n-2} - 1)x_2 + 2^{2n-2}x_3$$

Proof:

Based on Collorary 4.2, we have

$$X = x_1 + P_1 \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

$$= x_1 + (2^n - 1) \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{2^{n-1}-1} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

It is easy to see that $K = \sum_{i=1}^{n} w_i x_i' = (2^{2n-1} - 2^{n+1} + 1)x_1 + (2^{2n-2} - 1)x_2 + 2^{2n-2}x_3$ [26].

Then, we have

$$X = x_1 + (2^n - 1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^{n-1}-1} + |K|_{2^n}$$

■

We use the following example to illustrate the improvement that the proposed method provides over the modified CRT for this three-moduli set.

Example 4.4: For a R/B converter with 8-bit dynamic range based on the moduli set $N_3=\{2^n-1,\ 2^n,\ 2^{n-1}-1\}$, the specific moduli set {15,16,7} is chosen when n=4, since $7\times16\times15=1680>2^8=256$. Randomly choose a number from 0 to 255, for example, X=38. Its RNS representation $X=(x_1, x_2, x_3)$ is (8,6,3).

Based on the modified CRT, we have

$$\begin{aligned} Y &= \left|\left(2^{2n-1}-2^{n+1}+1\right)x_1+\left(2^{2n-2}-1\right)x_2+2^{2n-2}x_3\right|_{2^n(2^{n-1}-1)} \\ &= \left|(2^{2\times4-1}-2^{4+1}+1)\times8+(2^{2\times4-2}-1)\times6+2^{2\times4-2}\times3\right|_{2^4(2^{4-1}-1)} \\ &= |1346|_{112}=2 \\ X &= x_1+(2^n-1)\times Y=8+15\times2=38 \end{aligned}$$

Based on Proposition 4.3, we have

$$\begin{aligned} K &= \left(2^{2n-1}-2^{n+1}+1\right)x_1+\left(2^{2n-2}-1\right)x_2+2^{2n-2}x_3 \\ &= (2^{2\times4-1}-2^{4+1}+1)\times8+(2^{2\times4-2}-1)\times6+2^{2\times4-2}\times3=1346 \\ Y &= 2^n\left\lfloor\frac{K}{2^n}\right\rfloor_{2^{n-1}-1}+|K|_{2^n} \\ &= 2^4\times\left\lfloor\frac{1346}{2^4}\right\rfloor_7+|1346|_{16} \\ &= 2^4\times|84|_7+|1346|_{16} \end{aligned}$$

The binary representation of 1346 is $(10101000010)_2$, thus $1346\times2^{-4}=(1010100.0010)_2$. And $(1010100)_2$ is equal to 84, while $(0010)_2$ is the binary representation of $|1346|_{16}$. Then, we have

$$Y = 2^4 \times \left|(1010100)_2\right|_7 + (0010)_2$$
$$= 2^4 \times (000)_2 + (0010)_2$$
$$= (00000010)_2 = 2$$
$$X = x_1 + (2^n - 1) \times Y = 8 + 15 \times 2 = 38$$

When using the modified CRT, the above R/B conversion needs a modulo 112 operation. By using Proposition 4.3 which is derived from Theorem 4.1, the modulo size is reduced from 112 to 7. That is, the length of modulo operation is reduced from 7-bit to 3-bit. Also, the modulo operator is decreased from 1346 to 84, reduced by 4-bit. The 4-bit LSBs of Y are just the same 4-bit LSBs of 1346. Thus, the operations of multiplication by 2^4 and addition to $(0010)_2$ correspond to a simple concatenation operation.

Proposition 4.4: For $N_4=\{2^{2n}+1,\ 2^n, 2^n-1\}$, we have

$$X = x_1 + \left(2^{2n} + 1\right)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^n-1} + \left|K\right|_{2^n}$$
$$K = \left(2^{2n-1} - 1\right)x_1 + \left(2^n - 1\right)^2 x_2 + 2^{2n-1} x_3$$

Proof:

Based on Collorary 4.2, we have

$$X = x_1 + P_1 \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

$$= x_1 + \left(2^{2n}+1\right)\left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{2^n-1} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n}\right]$$

It is easy to see that $K = \sum_{i=1}^{n} w_i x_i' = \left(2^{2n-1}-1\right)x_1 + \left(2^n-1\right)^2 x_2 + 2^{2n-1} x_3$ [26]. Then, we have

$$X = x_1 + \left(2^{2n}+1\right)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^n-1} + |K|_{2^n}$$

■

We use the following example to illustrate the improvement that the proposed method provides over the modified CRT for this three-moduli set.

Example 4.5: For a R/B converter with 8-bit dynamic range based on the moduli set $N_4=\{2^{2n}+1, 2^n, 2^n-1\}$, the specific moduli set {65,8,7} is chosen when n=3, since $65\times8\times7=3640>2^8=256$. Randomly choose a number from 0 to 255, for example, X=38. Its RNS representation $X=(x_1, x_2, x_3)$ is (38,6,3).

Based on the modified CRT, we have

$$Y = \left|\left(2^{2n-1}-1\right)x_1 + \left(2^n-1\right)^2 x_2 + 2^{2n-1} x_3\right|_{2^n(2^n-1)}$$

$$= \left|(2^{2\times3-1}-1)\times38 + (2^3-1)^2\times6 + 2^{2\times3-1}\times3\right|_{2^3(2^3-1)}$$

$$= |1568|_{56} = 0$$

$$X = x_1 + (2^{2n}+1)\times Y = 38 + 65\times0 = 38$$

Based on Proposition 4.4, we have

$$K = (2^{2n-1}-1)x_1 + (2^n-1)^2 x_2 + 2^{2n-1}x_3$$
$$= (2^{2\times3-1}-1)\times 38 + (2^3-1)^2 \times 6 + 2^{2\times3-1}\times 3 = 1568$$
$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^n-1} + |K|_{2^n}$$
$$= 2^3 \times \left\| \frac{1568}{2^3} \right\|_7 + |1568|_8$$
$$= 2^3 \times |196|_7 + |1568|_8$$

The binary representation of 1568 is $(11000100000)_2$, thus $1568\times 2^{-3} = (11000100.000)_2$. And $(11000100)_2$ is equal to 196, while $(000)_2$ is the binary representation of $|1568|_8$. Then, we have

$$Y = 2^3 \times |(11000100)_2|_7 + (000)_2$$
$$= 2^3 \times (000)_2 + (000)_2$$
$$= (000000)_2 = 0$$
$$X = x_1 + (2^{2n}+1)\times Y = 38 + 65\times 0 = 38$$

When using the modified CRT, the above R/B conversion needs a modulo 56 operation. By using Proposition 4.4 which is derived from Theorem 4.1, the modulo size is reduced from 56 to 7. That is, the length of modulo operation is reduced from 6-bit to 3-bit. Also, the modulo operator is decreased from 1568 to 196, reduced by 3-bit. The 3-bit LSBs of Y are just the same 3-bit LSBs of 1568. Thus, the operations of multiplication by 2^3 and addition to $(000)_2$ correspond to a simple concatenation operation.

Proposition 4.5: For $N_5=\{2^{2n}+1, 2^n, 2^{2n}-1\}$, we have

$$X = x_1 + (2^{2n}+1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^{2n}-1} + |K|_{2^n}$$

$$K = (2^{2n-1}-1)x_1 + (2^n-1)(2^{2n}-1)x_2 + 2^{2n-1}x_3$$

Proof:

Based on Collorary 4.2, we have

$$X = x_1 + P_1 \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

$$= x_1 + (2^{2n}+1) \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{2^{2n}-1} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

It is easy to see that $K = \sum_{i=1}^{n} w_i x_i' = (2^{2n-1}-1)x_1 + (2^n-1)(2^{2n}-1)x_2 + 2^{2n-1}x_3$ [26].

Then, we have

$$X = x_1 + (2^{2n}+1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^{2n}-1} + |K|_{2^n}$$

■

We use the following example to illustrate the improvement that the proposed method provides over the modified CRT for this three-moduli set.

Example 4.6: For a R/B converter with 8-bit dynamic range based on the moduli set $N_5=\{2^{2n}+1,\ 2^n, 2^{2n}-1\}$, the specific moduli set {17,4,15} is chosen when n=2, since $17\times4\times15=1020>2^8=256$.

Randomly choose a number from 0 to 255, for example, X=38. Its RNS representation $X=(x_1, x_2, x_3)$ is (4,2,8).

Based on the modified CRT, we have

$$\begin{aligned} Y &= \left|(2^{2n-1}-1)x_1 + (2^n-1)(2^{2n}-1)x_2 + 2^{2n-1}x_3\right|_{2^n(2^{2n}-1)} \\ &= \left|(2^{2\times2-1}-1)\times4 + (2^2-1)(2^{2\times2}-1)\times2 + 2^{2\times2-1}\times8\right|_{2^2(2^{2\times2}-1)} \\ &= |182|_{60} = 2 \\ X &= x_1 + (2^{2n}+1)\times Y = 4+17\times2 = 38 \end{aligned}$$

Based on Proposition 4.5, we have

$$\begin{aligned} K &= (2^{2n-1}-1)x_1 + (2^n-1)(2^{2n}-1)x_2 + 2^{2n-1}x_3 \\ &= (2^{2\times2-1}-1)\times4 + (2^2-1)(2^{2\times2}-1)\times2 + 2^{2\times2-1}\times8 = 182 \\ Y &= 2^n\left\lfloor\frac{K}{2^n}\right\rfloor_{2^{2n}-1} + |K|_{2^n} \\ &= 2^2\times\left\lfloor\frac{182}{2^2}\right\rfloor_{15} + |182|_4 \\ &= 2^2\times|45|_{15} + |182|_4 \end{aligned}$$

The binary representation of 182 is $(10110110)_2$, thus $182\times2^{-2} = (101101.10)_2$. And $(101101)_2$ is equal to 45, while $(10)_2$ is the binary representation of $|182|_4$. Then, we have

$$\begin{aligned} Y &= 2^2\times|(10110110)_2|_{15} + (10)_2 \\ &= 2^2\times(0000)_2 + (10)_2 \\ &= (000010)_2 = 2 \\ X &= x_1 + (2^{2n}+1)\times Y = 4+17\times2 = 38 \end{aligned}$$

When using the modified CRT, the above R/B conversion needs a modulo 60 operation. By using Proposition 4.5 which is derived from Theorem 4.1, the modulo size is reduced from 60 to 15. That is, the length of modulo operation is reduced from 6-bit to

4-bit. Also, the modulo operator is decreased from 182 to 45, reduced by 2-bit. The 2-bit LSBs of Y are just the same 2-bit LSBs of 182. Thus, the operations of multiplication by 2^2 and addition to $(10)_2$ correspond to a simple concatenation operation.

Proposition 4.6: For $N_6=\{2^{n+1}+1, 2^n, 2^{n+1}-1\}$, we have

$$X = x_1 + (2^{n+1}+1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^{n+1}-1} + |K|_{2^n}$$

$$K = (2^n-1)x_1 + (2^n-1)(2^{n+1}-1)x_2 + 2^n x_3$$

Proof:

Based on Collorary 4.2, we have

$$X = x_1 + P_1 \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{P_3} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

$$= x_1 + (2^{n+1}+1) \left[2^n \left\| \frac{\sum_{i=1}^{n} w_i x_i'}{2^n} \right\|_{2^{n+1}-1} + \left| \sum_{i=1}^{n} w_i x_i' \right|_{2^n} \right]$$

It is easy to see that $K = \sum_{i=1}^{n} w_i x_i' = (2^n-1)x_1 + (2^n-1)(2^{n+1}-1)x_2 + 2^n x_3$ [26]. Then, we have

$$X = x_1 + (2^{n+1}+1)Y$$

where n>1, and

$$Y = 2^n \left\| \frac{K}{2^n} \right\|_{2^{n+1}-1} + |K|_{2^n}$$

■

We use the following example to illustrate the improvement that the proposed method provides over the modified CRT for this three-moduli set.

Example 4.7: For a R/B converter with 8-bit dynamic range based on the moduli set $N_6=\{2^{n+1}+1,\ 2^n\ ,2^{n+1}-1\}$, the specific moduli set {17,8,15} is chosen when n=3, since $17\times8\times15=2040>2^8=256$.

Randomly choose a number from 0 to 255, for example, X=38. Its RNS representation $X=(x_1, x_2, x_3)$ is (4,6,8).

Based on the modified CRT, we have

$$\begin{aligned} Y &= \left|(2^n-1)x_1 + (2^n-1)(2^{n+1}-1)x_2 + 2^n x_3\right|_{2^n(2^{n+1}-1)} \\ &= \left|(2^3-1)\times 4 + (2^3-1)(2^{3+1}-1)\times 6 + 2^3\times 8\right|_{2^3(2^{3+1}-1)} \\ &= |722|_{120} = 2 \\ X &= x_1 + (2^{2n}+1)\times Y = 4+17\times 2 = 38 \end{aligned}$$

Based on Proposition 4.6, we have

$$\begin{aligned} K &= (2^n-1)x_1 + (2^n-1)(2^{n+1}-1)x_2 + 2^n x_3 \\ &= (2^3-1)\times 4 + (2^3-1)(2^{3+1}-1)\times 6 + 2^3\times 8 = 722 \\ Y &= 2^n \left\| \frac{K}{2^n} \right\|_{2^{n+1}-1} + |K|_{2^n} \\ &= 2^3 \times \left\| \frac{722}{2^3} \right\|_{15} + |722|_8 \\ &= 2^3 \times |90|_{15} + |722|_8 \end{aligned}$$

The binary representation of 722 is $(1111010010)_2$, thus $722 \times 2^{-3} = (1111010.010)_2$. And $(1111010)_2$ is equal to 90, while $(010)_2$ is the binary representation of $|722|_8$. Then, we have

$$\begin{aligned} Y &= 2^3 \times |(1111010)_2|_{15} + (010)_2 \\ &= 2^3 \times (0000)_2 + (010)_2 \\ &= (0000010)_2 = 2 \\ X &= x_1 + (2^{n+1} + 1) \times Y = 4 + 17 \times 2 = 38 \end{aligned}$$

When using the modified CRT, the above R/B conversion needs a modulo 120 operation. By using Proposition 4.6 which is derived from Theorem 4.1, the modulo size is reduced from 120 to 15. That is, the length of modulo operation is reduced from 7-bit to 4-bit. Also, the modulo operator is decreased from 722 to 90, reduced by 3-bit. The 3-bit LSBs of Y are just the same 3-bit LSBs of 722. Thus, the operations of multiplication by 2^3 and addition to $(010)_2$ correspond to a simple concatenation operation.

4.4 Comprehensive Study on the Proposed R/B Algorithms for Three-Moduli Sets

We summarize the comparison of the modulo operations required by the six three-moduli sets in Table 4.1. It is seen in Table 4.1 that all these six new R/B algorithms reduce the modulo size compared to the modified CRT and provide new options to design high-speed R/B converters. The modulo part is the critical path of the R/B converter. The proposed method significantly reduces the critical path. Thus, the speed of the R/B converter will be very much improved.

In [26], a comprehensive study of R/B converters for three-moduli sets has been carried out based on the modified CRT. It has been shown that the moduli set N_1 can offer the fastest R/B converter while requiring the least hardware resources for a given dynamic range. However, based on the proposed modulo reduction method, the R/B converter designs are different from those based on the modified CRT. Thus, the conclusion of [26] might not hold for these new designs. For example, it is seen in Table 4.1 that the modulo operation required by N_2 is the smallest compared to other three-moduli sets. Hence, the N_2-based converter might be the fastest.

Table 4.1 Modulo comparison of six three-moduli sets

Three-Moduli Sets	8-bit Dynamic range	Modulo Size of the Modified CRT	Modulo Size of the Proposed CRT
$N_1=\{2^n+1, 2^n, 2^n-1\}$	{9,8,7}	6-bit	3-bit
$N_2=\{2^{2n}+1, 2^n, 2^n+1\}$	{17,4,5}	5-bit	3-bit
$N_3=\{2^n-1, 2^n, 2^{n-1}-1\}$	{15,16,7}	7-bit	3-bit
$N_4=\{2^{2n}+1, 2^n, 2^n-1\}$	{65,8,7}	6-bit	3-bit
$N_5=\{2^{2n}+1, 2^n, 2^{2n}-1\}$	{17,4,15}	6-bit	4-bit
$N_6=\{2^{n+1}+1, 2^n, 2^{n+1}-1\}$	{17,8,15}	7-bit	4-bit

4.5 Summary

In this chapter, we have applied the modulo reduction algorithms to improve the modified CRT and derive efficient modulo-reduced R/B algorithms for six three-moduli sets. By applying the proposed modulo reduction algorithms to the modified CRT, the complexity of modulo operation in the modified CRT is reduced considerably by partitioning the modulo operation with a large base to several individual modulo operations of small bases in parallel. The parallelism provides high concurrent operation and decreases delay. We then apply the modulo reduction technique and the modulo

reduced modified CRT to derive R/B algorithms for the two existing three-moduli sets and four newly found three-moduli sets, all in form of $\{P_1, 2^n, P_3\}$.

CHAPTER 5

NEW DESIGNS OF BINARY INCREMENTER/DECREMENTER

5.1 Introduction

An incrementer/decrementer is a common building block in many digital systems like microcontrollers, microprocessors and frequency divider [30], [31]. It is also used as part of the proposed R/B converter in this thesis. The binary incrementer/decrementer implements the function of $Y=Z\pm1$ where Z is the input integer number. The current implementations of incrementer/decrementers are mainly adder-based or counter-based [32], [33]. For the adder-based incrementer/decrementers, the operating speed limitation comes from the inherent carry propagation of adders. The counter-based incrementer/decrementers usually implemented as Finite Machines have a similar speed limitation problem. This is because the up/down counter is also designed based on the concept of addition, although it uses half-adder rather than full-adder [34].

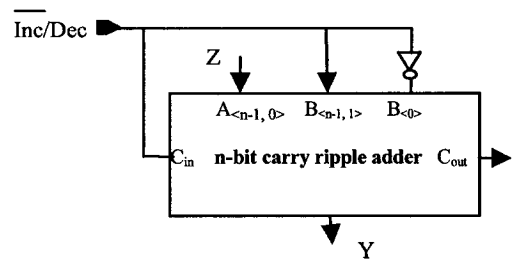


Fig. 5.1. The CPA-based incrementer/decrementer

A carry propagate adder (CPA)-based n-bit signed-2's complement incrementer/decrementer [33] is shown in Fig. 5.1. When the operand A is to be incremented by 1, $\overline{Inc}/Dec = 0$, thus B receives 000...01 and the carry input is set to 0. If the operand A is to be decremented by 1, $\overline{Inc}/Dec = 1$, thus B receives 111...10 and the carry input is set to 1 to achieve signed-2's complement operation. To implement increment and decrement by the same circuit, the operand B and the carry input can be tied directly to the mode-selection signal $\overline{Inc}/Dec$ or its complement, as shown in Fig. 5.1. With this configuration, there is a carry propagation effect from C_{in} to C_{out} through a series of full adders that makes this circuit slow.

In this chapter, a novel MUX-based algorithm for increment and decrement operations is proposed. The algorithm makes use of the mechanism of information transferring between the input Z and the output Y, resulting in a high-performance MUX-based binary incrementer/decrementer. Based on this algorithm, new designs are introduced to improve both the unsigned and signed-2's complement incrementer/decrementer. Furthermore, for the purpose of performance evaluation, the proposed unsigned MUX-based incrementer/decrementer is implemented using FPGA technology.

5.2 Proposed Unsigned Binary Incrementer/Decrementer

In this section, we propose a new unsigned MUX-based incrementer/decrementer avoiding the delay introduced by the inherent ripple carry of the adder-based design. The decrementer of Z-1 is the most important part of the critical path of the proposed incrementer/decrementer. Thus, we first present a new decrement algorithm, and then, apply it to design a MUX-based incrementer/decrementer.

5.2.1 New MUX-Based Unsigned Decrement Algorithm

The truth table of the unsigned decrement operation is shown in Table 5.1. Let the input be $Z = Z_{n-1}....Z_1\ Z_0$. To find the output, start from the LSB Z_0 and search to the MSB Z_{n-1} for the first occurrence of bit '1'. Let the first bit '1' to be Z_J. Then the output is obtained as follows:

1) Complement all data bits Z_I for I = J....0.
2) Leave all other bits of Z_I 's for I > J as they are.
3) Complement all bits if Z = 0.

We denote Z_J as the Least Significant One Bit (LSOB). Thus, for n-bit input Z, we derive the algorithm shown in Fig. 5.2 to get the n-bit output Y. Proposition 5.1 can also describe the proposed algorithm.

Proposition 5.1 Given any n-bit binary input Z, we get its decrement result Y = Z-1 as follows.

$$Y = Z - 1 \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (Z_0 + Z_1 + \cdots + Z_{j-1})}, & 1 \le j \le n-1 \\ Y_j = \overline{Z_0}, & j = 0 \end{cases} \qquad (5.1)$$

Proof:

According to the proposed algorithm in Fig. 5.2, we have

$$Y_0 = \overline{Z_0}$$

$$Y_1 = Z_1 Z_0 + \overline{Z_1 Z_0} = \overline{Z_1 \oplus Z_0}$$

$$Y_2 = Z_2(Z_0 + Z_1) + \overline{Z_2 Z_0 + Z_1} = \overline{Z_2 \oplus (Z_0 + Z_1)}$$

$$\vdots$$

$$Y_{n-1} = \overline{Z_{n-1} \oplus (Z_0 + Z_1 + \cdots + Z_{n-2})}$$

■

The following example is used to demonstrate the operation of the proposed algorithm.

Example 5.1: Find the decrement result of the 8-bit number $(10110100)_2$.

We carry out the operation by searching from the LSB to the MSB to find the first '1' bit. This LSOB is the 3rd bit from the LSB. Thus, the first 3 LSBs 100 will be inverted to 011 in the output and the rest of the bits will keep their values as 10110 in the output. Then, the final output will be 10110011, which is exactly the desired result.

Table 5.1. Unsigned decrement truth table

Z	*Y = Z-1*
0···000	1···111
0···001	0···000
0···010	0···001
0···011	0···010
⋮	⋮
0···010···0	0···001···1
⋮	⋮
1···111	1···110

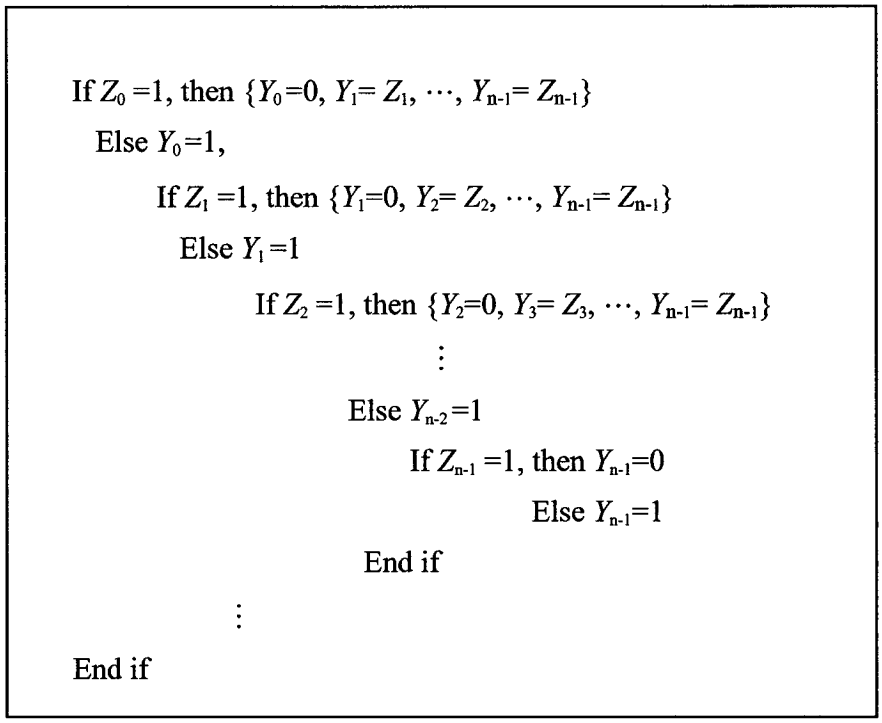


Fig. 5.2. The proposed decrement algorithm

5.2.2 New MUX-based Unsigned Decrementer

Based on the proposed algorithm, a new *n*-bit MUX-based decrementer is designed as shown in Fig. 5.3. It is composed of a data-out MUX array and a decision module (DM) used to find the LSOB. The output of DM is D_{n-1} ... D_1 D_0. When D_J = 0 ($J = 0,...,n-1$), the input bits from Z_J to Z_0 are 0. From Proposition 5.1, it can be noted that each bit of the decrement result Y except Y_0 can be derived by a MUX operation. For example, by equation (5.1), we have $Y_1 = Z_1 Z_0 + \overline{Z_1 Z_0} = \overline{Z_1 \oplus Z_0}$ that can be implemented by a MUX whose inputs are Z_1 and $\overline{Z_1}$ with a select signal connected to Z_0.

In the case of Z = 0···000, the decrement output should be Y = 1···111, and there is a carry-out (Cout = 0). In other cases, there is no carry-out (Cout = 1). Thus, we have

$$C_{out} = 0, \quad when \quad D_{n-1} = Z_0 + Z_1 + \cdots + Z_{n-1} = 0$$

$$C_{out} = 1, \quad when \quad D_{n-1} = Z_0 + Z_1 + \cdots + Z_{n-1} = 1$$

Combining the above two equations, we can then obtain

$$C_{out} = D_{n-1} = Z_0 + Z_1 + \cdots + Z_{n-1} \tag{5.2}$$

Based on (5.2), the implementation of the carry-out is just a direct connection to D_{n-1} as shown in Fig. 5.3.

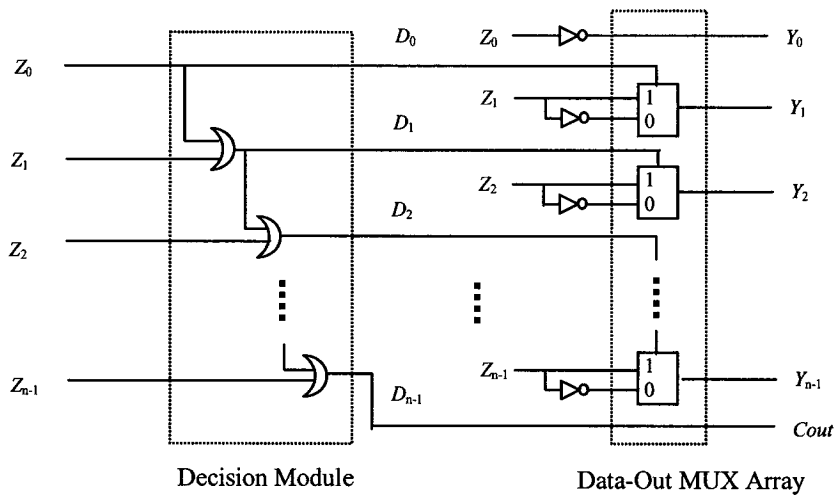


Fig. 5.3. The proposed MUX-based unsigned decrementer

5.2.3 New MUX-based Unsigned Incrementer/Decrementer

The proposed MUX-based decrementer can be used to build an unsigned incrementer/decrementer. The increment function will require that all the input bits be inverted in advance and then sent to the proposed decrement circuit. Then, the inverted output of the decrementer will give the desired increment result. This feature can be summarized as Proposition 5.2. We rewrite Lemma 4 which is needed for the proof of Proposition 5.2 as follows.

Lemma 4 $2^n - 1 - X = \bar{x}_{n-1}\bar{x}_{n-2}\cdots\bar{x}_0$ (5.2)

for any non-zero n-bit binary number $X = x_{n-1}x_{n-2}\cdots x_0$

Proposition 5.2 Given any *n*-bit binary number Z, we have

$$Z+1 = \overline{\overline{Z}-1} \tag{5.3}$$

Proof:

According to Lemma 4, we have $\overline{Z} = 2^n - 1 - Z$. Thus

$$\begin{aligned}\overline{\overline{Z}-1} &= 2^n - 1 - (\overline{Z} - 1) \\ &= 2^n - \overline{Z} \\ &= 2^n - (2^n - 1 - Z) \\ &= Z + 1\end{aligned}$$

■

The following example is used to demonstrate the operation of increment based on Proposition 5.2.

Example 5.2: Find the increment result of the 8-bit number $(10110100)_2$.

The inverted input is 01001011. Then, after the proposed decrementer, the output will be 01001010. By inverting this output, we will have the correct increment result as 10110101.

By applying Proposition 5.2 to Proposition 5.1, we can derive a new binary increment algorithm as follows.

Proposition 5.3 Given any n-bit binary input Z, we get its binary increment result $Y = Z+1$ as follows.

$$Y = Z+1 \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}, & 1 \le j \le n-1 \\ Y_j = \overline{Z_0}, & j = 0 \end{cases} \quad (5.4)$$

Proof:

1) In the case of $1 \le j \le n-1$, according to Proposition 5.1 and 5.2, we have

$$\begin{aligned} Y_j &= \overline{Z_j} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}}) \\ &= Z_j(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}}) + \overline{Z_j}\,\overline{Z_0 + Z_1 + \cdots + Z_{j-1}} \\ &= \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})} \end{aligned}$$

2) In the case of $j = 0$, according to Proposition 5.1 and 5.2, we have

$$Y_j = \overline{\overline{Z_j}} = \overline{Z_j}$$

■

Comparing equation (5.1) and (5.4), it can be observed that they are both $Y_j = \overline{Z_0}$ in the case of $j = 0$. In the case of $1 \le j \le n-1$, equation (5.1) is $Y_j = \overline{Z_j \oplus (Z_0 + Z_1 + \cdots + Z_{j-1})}$ and equation (5.2) is $Y_j = \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}$. Since the logic OR of all input bits: $Z_0 + Z_1 + \cdots + Z_{j-1}$ is implemented by the DM in Fig.

5.3, we can implement $\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}}$ by inverting the input of the DM in Fig. 5.3. The proposed MUX-based binary incrementer is shown in Fig. 5.4.

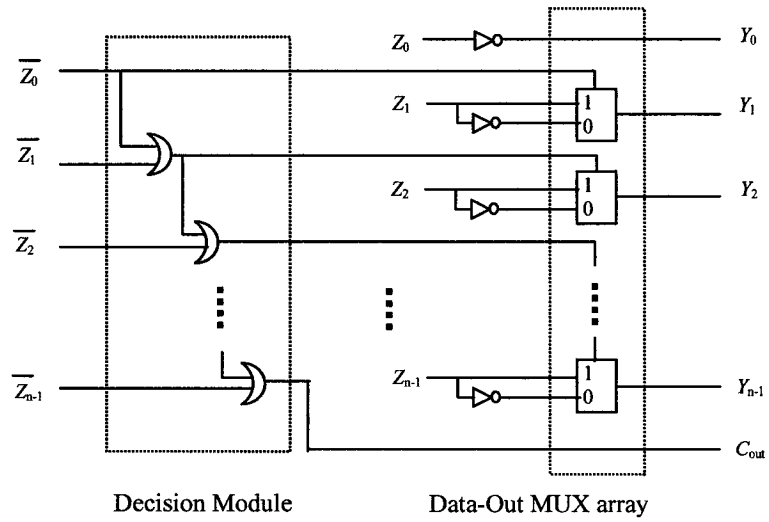


Fig. 5.4. The proposed MUX-based unsigned incrementer

In the case of $Z = 1\cdots111$, the increment output should be $Y = 0\cdots000$, and there is a carry-out (Cout = 0). In other cases, there is no carry-out (Cout = 1). Thus, we have

$$C_{out} = 0, \quad when \quad D_{n-1} = \overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}} = 0$$

$$C_{out} = 1, \quad when \quad D_{n-1} = \overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}} = 1$$

Combining the above two equations, we can then obtain

$$C_{out} = D_{n-1} = Z_0 + Z_1 + \cdots + Z_{n-1} \tag{5.5}$$

Based on (5.5), the implementation of the carry-out is just a direct connection to D_{n-1} as shown in Fig. 5.4.

The proposed n-bit MUX-based incrementer/decrementer is shown in Fig. 5.5. It is composed of a data-in MUX array, a data-out MUX array and a DM used to find the LSOB. The data-in MUX array is used to select between the input Z and its complement.

The output of DM is $D_{n-1} \ldots D_1 D_0$. When $D_J = 0$ $(J = 0,\ldots,n-1)$, the input bits from Z_J to Z_0 are 0 for the case of decrement and 1 for the case of increment. To implement the function of increment and decrement by the same circuit, we use the mode-selection signal $\overline{Inc}/Dec$. When the operand Z is to be incremented by 1, the signal $\overline{Inc}/Dec$ is set to 0. If the operand Z is to be decremented by 1, the signal $\overline{Inc}/Dec$ is set to 1.

In both cases of increment and decrement, when the signal *Cout* is 0, it means that there is a carry-out from the MSB. This $Cout = 0$ happens in two cases. One is $Z = 0\cdots000$ and the function to be implemented is decrement. The other is $Z = 1\cdots111$ and the function to be implemented is increment. In both cases, D_{n-1} is zero. In other cases, $D_{n-1} = 1$ and there is no carry-out ($Cout = 1$). Thus, the carry-out circuit here is the same as Fig. 5.4.

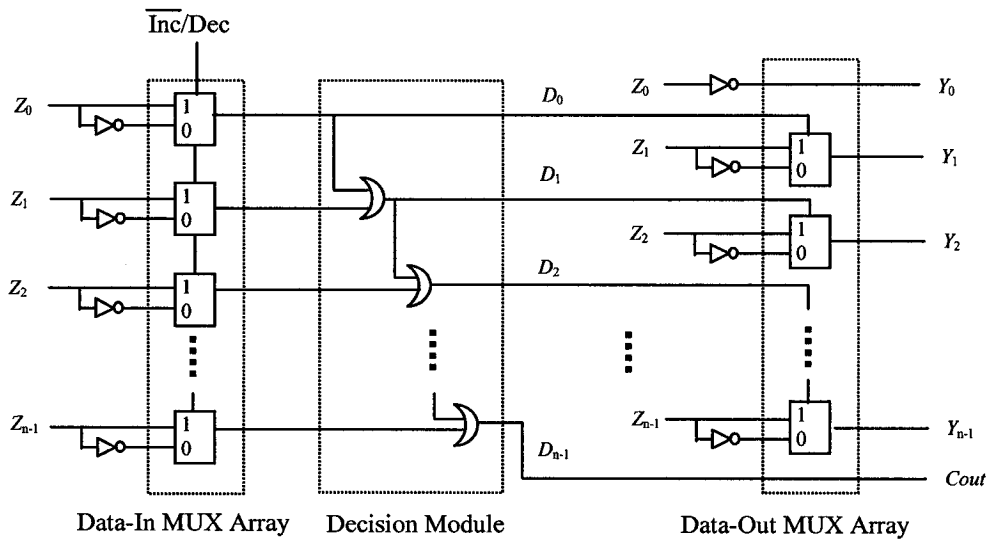


Fig. 5.5. The proposed MUX-based unsigned incrementer/decrementer

5.2.4 A Comparison among the Incrementer/Decrementers

Compared to the existing adder-based incrementer/decrementer in Fig. 5.1, the proposed incrementer/decrementer has higher speed and requires less hardware resources as shown in Table 5.2. The reason is that the proposed design here uses MUX instead of full adder as basic arithmetic unit. This reduces the hardware complexity since a MUX is much simpler than a FA. Furthermore, the information of lower significance bits is transferred to higher significance bits by a series of OR gates. That is, the critical path of the output *Y* of the proposed decrementer consists of one inverter, (*n-2*) OR gates and two MUXs. This structure reduces the delay introduced by the inherent ripple carry of CPA in the CPA-based incrementer/decrementer and reduces the complexity of the CPA-based incrementer/decrementer to a great scale.

Table 5.2. Complexity and delay of the unsigned incrementer/decrementer based on CPA and MUX

Incrementer/ Decrementer	***Complexity***	***Delay***
CPA-based	$nFAs + nInvs$	$nt_{FA} + t_{Inv}$
Proposed MUX-based	$(2n-1)MUXs + nInvs + (n-1)ORs$	$t_{inv} + (n-2)t_{OR} + 2t_{MUX}$

5.3 Proposed Signed-2's Complement Incrementer/Decrementer

In this section, we extend the proposed unsigned increment/decrement method to the design of signed incrementer/decrementers. The 2's complement number representation is used for this study.

Table 5.3. Truth table of signed and unsigned increment/decrement

Decimal	*Signed-2's complement*			*Decimal*	*Unsigned*		
	Z	*Y=Z-1*	*Y=Z+1*		*Z*	*Y=Z-1*	*Y=Z+1*
+7	0111	0110	1000	7	0111	0110	1000
+6	0110	0101	0111	6	0110	0101	0111
+5	0101	0100	0110	5	0101	0100	0110
+4	0100	0011	0101	4	0100	0011	0101
+3	0011	0010	0100	3	0011	0010	0100
+2	0010	0001	0011	2	0010	0001	0011
+1	0001	0000	0010	1	0001	0000	0010
+0	0000	1111	0001	0	0000	1111	0001
–0	—	—	—	—	—	—	—
–1	1111	1110	0000	15	1111	1110	0000
–2	1110	1101	1111	14	1110	1101	1111
–3	1101	1100	1110	13	1101	1100	1110
–4	1100	1011	1101	12	1100	1011	1101
–5	1011	1010	1100	11	1011	1010	1100
–6	1010	1001	1011	10	1010	1001	1011
–7	1001	1000	1010	9	1001	1000	1010
–8	1000	0111	1001	8	1000	0111	1001

Table 5.3 lists all possible 4-bit signed binary numbers in signed-2's complement representation as well as their increment/decrement results. Note that the positive numbers in signed-2's complement representation have a 0 in the leftmost position. All negative numbers have a 1 in the leftmost bit position to be distinguished from positive numbers.

The corresponding unsigned number, are also shown in Table 5.3 for the purpose of comparison. It is seen from Table 5.3 that the increment and decrement of the signed-2's complement representation of the 4-bit numbers from 0 to 7 are the same as the increment and decrement of the unsigned representations. And the increment and decrement of the signed-2's complement representation of the 4-bit numbers from -1 to –8 are the same as the increment and decrement of the unsigned representation of the 4-bit

numbers from 15 to 8. For example, the signed-2's complement decrement result of 1001 (the signed-2's complement of –7) is 1000 (the signed-2's complement of –8), while the unsigned decrement result of 1001 (the unsigned representation of 9) is also 1000 (the unsigned representation of 8). Thus, we can use the *n*-bit unsigned incrementer/decrementer to implement the *n*-bit signed-2's complement incrementer/decrementer as shown in Fig. 5.6.

The proposed signed design implements overflow mechanism as follows. In the signed design, overflow (*Cout* = 0) occurs in the following two cases. If the function to be implemented is decrement, in the case of Z = 1000 (-8), the output is Y = 0111 (7). This is not a legal operation. Thus, there is overflow (*Cout* = 0). If the function to be implemented is increment, in the case of Z = 0111 (7), the output is Y = 1000 (-8). This is not a legal operation. Thus, there is overflow (*Cout* = 0). In these two cases, we have

$$C_{out} = 0, \ when \begin{cases} \overline{Inc}/Dec = 1 \ \ AND \ \ D_{n-2} = 0 \ \ AND \ \ Z_{n-1} = 1 \\ \overline{Inc}/Dec = 0 \ \ AND \ \ D_{n-2} = 0 \ \ AND \ \ Z_{n-1} = 0 \end{cases}$$

$$C_{out} = 1, \ when \ \ others$$

Combining the above two equations, we can then obtain

$$\begin{aligned} C_{out} &= \overline{\overline{Inc}/Dec \cdot \overline{D_{n-2}} \cdot Z_{n-1}} \cdot \overline{\overline{\overline{Inc}/Dec} \cdot \overline{D_{n-2}} \cdot \overline{Z_{n-1}}} \\ &= \left(\overline{\overline{Inc}/Dec} + D_{n-2} + \overline{Z_{n-1}}\right) \cdot \left(\overline{Inc}/Dec + D_{n-2} + Z_{n-1}\right) \\ &= \overline{\overline{Inc}/Dec} \cdot D_{n-2} + \overline{\overline{Inc}/Dec} \cdot Z_{n-1} + D_{n-2} + D_{n-2} \cdot Z_{n-1} + D_{n-2} \cdot \overline{Inc}/Dec \\ &\quad + \overline{Inc}/Dec \cdot \overline{Z_{n-1}} + D_{n-2} \cdot \overline{Z_{n-1}} \\ &= D_{n-2} \cdot \left(\overline{\overline{Inc}/Dec} + 1 + Z_{n-1} + \overline{Inc}/Dec + \overline{Z_{n-1}}\right) + \overline{\overline{Inc}/Dec} \cdot Z_{n-1} + \overline{Inc}/Dec \cdot \overline{Z_{n-1}} \end{aligned}$$

$$= D_{n-2} + \overline{Inc} / Dec \oplus Z \tag{5.6}$$

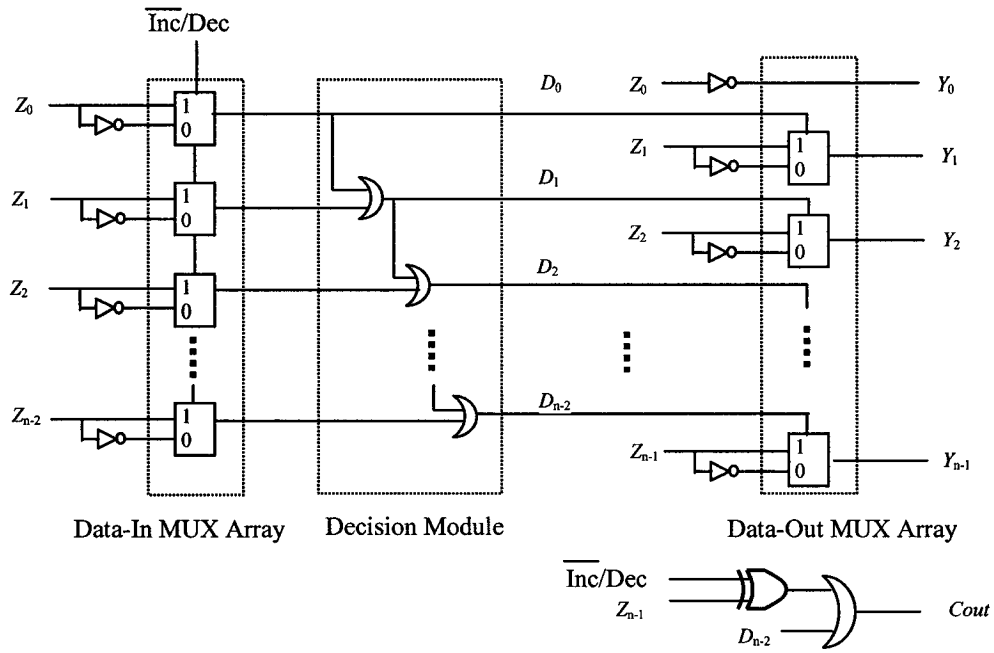


Fig. 5.6. The proposed signed-2's complement incrementer/decrementer

Based on (5.6), the implementation of the overflow flag C_{out} consists of one XOR gate and one OR gate as shown in Fig. 5.6. The mode-selection signal $\overline{Inc} / Dec$ is used to select the function to be implemented. D_{n-2} is the MSB of the output of the DM. When $D_J = 0$ $(J = 0,...,n-2)$, the input bits from Z_J to Z_0 are 0 for the case of decrement and 1 for the case of increment.

The performance of the proposed 2's complement n-bit incrementer/decrementer is similar to that of the unsigned design in Section 5.2, with a minor modification of the carry-out circuit. Thus, the comparison between the proposed design and the CPA-based design is similar to that shown in Table 5.2. The proposed MUX-based design has higher speed and requires less hardware resources in the signed-2's complement case.

5.4 FPGA Implementation Results

To get a practical performance measure, both the traditional and the proposed unsigned MUX-based incrementer/decrementers are implemented using Xilinx FPGA technology for the 32-bit and 64-bit cases. The synthesis and implementation tools are Synopsys's Design Compiler Version 3.4b and Xillinx Alliance M1.3 software. The target technology is a Xillinx 4010e-3 FPGA. The performance evaluation in terms of power, area and delay is carried out. The results are compared in Table 5.4. The results of the FPGA implementation show that the proposed design consumes 40% less hardware than those based on CPA. The delay is also reduced close to 50%. The power consumed by the MUX-based design is almost 30% less than that of the CPA-based design. The reason for this improvement is that the arithmetic operation of the CPA-based incrementer/decrementer is based on addition with ripple carry, while the proposed incrementer/decrementer takes MUX as its arithmetic base. To reduce delay, the traditional incrementer/decrementers based on high-speed adders such as carry select adders or carry look ahead adders could have been used. However, these come at higher cost of power and area.

Table 5.4. Implementation results of the unsigned incrementer/decrementers

	32-bit			*64-bit*		
Incrementer/Decrementer	***Power (mw)***	***Cell Area***	***Delay (ns)***	***Power (mw)***	***Cell Area***	***Delay (ns)***
C_1 : CPA-based	13.4495	161	225.39	25.5791	321	443.95
C_2 : MUX-based	9.8598	95	120.38	18.1628	191	238.14
$\frac{C_1 - C_2}{C_1} \times 100\%$	26.69	40.99	46.59	28.99	40.50	46.36

5.5 Summary

In this chapter, new algorithms and designs have been proposed for the signed and unsigned incrementer/decrementers. The proposed MUX-based incrementer/decrementer is more efficient in terms of speed and hardware complexity compared to the adder-based incrementer/decrementer for both signed and unsigned cases. An FPGA based comparison of the proposed incrementer/decrementer with the traditional one shows that the proposed design requires 40% less hardware while the delay is reduced close to 50%. The power consumed by the MUX-based design is almost 30% less than that of the CPA-based design.

CHAPTER 6

NEW DESIGNS OF MODULO 2^n-1 INCREMENTER/DECREMENTER

6.1 Introduction

The modulo increment/decrement operations of $|Z-1|_{2^n-1}$ and $|Z+1|_{2^n-1}$ where Z is a n-bit integer, are useful building blocks. They find applications in different circuits such as frequency dividers, memory management units and microcontrollers (circulating buffer pointer increment/decrement). They are also very important for implementation of an area-time efficient and power-saving R/B converter which is the most complicated part of a RNS system. For example, in [27], the author suggests to use a modulo 2^n–1 adder to implement the variable C which is defined by equation (20) in [27] as $|r_2+1|_{2^{k-1}-1}$. If we assume $Z=r_2$ and n = k-1, then we have the modulo increment operation as $|Z+1|_{2^n-1}$. In [35], the implementation of the variable Y which is given by equation (7c) in [6] employs the modulo 2^n–1 decrement operation. By replacing the variables in equation (7c) with equations (5a) – (5f) in [35], we can have the equation of Y as follows.

$$Y = \begin{cases} \left|2^{k-2}x_1 - x_2 + 2^{k-2}x_3\right|_{2^{k-1}-1} & for \quad x_2 \geq x_1 \\ \left|2^{k-2}x_1 - x_2 + 2^{k-2}x_3 - 1\right|_{2^{k-1}-1} & for \quad x_2 < x_1 \end{cases}$$

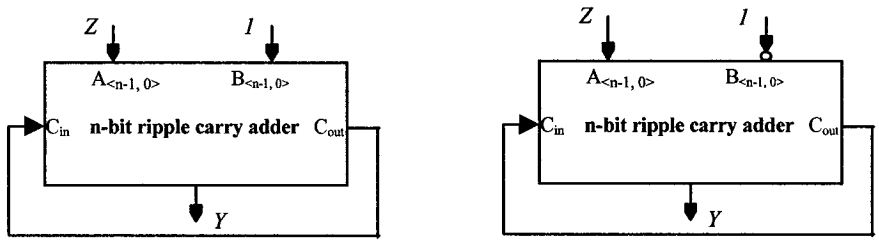
If we assume $Z = 2^{k-2}x_1 - x_2 + 2^{k-2}x_3$ and $n = k$-1, then we have the modulo decrement operation as follows.

$$Y = \begin{cases} \left|Z\right|_{2^n-1} & for \quad x_2 \geq x_1 \\ \left|Z-1\right|_{2^n-1} & for \quad x_2 < x_1 \end{cases}$$

To further reduce the delay compared to [27], the author of [35] uses two modulo adders working in parallel to implement the modulo operations $\left|Z\right|_{2^n-1}$ and $\left|Z-1\right|_{2^n-1}$ of the variable *Y*.

The standard implementation of a modulo 2^n-1 adder uses a conventional binary adder with the carry output connected to the carry input to achieve the EAC. The CPA-based modulo incrementer/decrementer is shown in Fig. 6.1 (a) and (b). Fig. 6.1 (a) shows a modulo incrementer that conducts a 1's complement addition of $\left|Z+1\right|_{2^n-1}$. Fig. 6.1 (b) is a modulo decrementer that conducts a 1's complement subtraction of $\left|Z-1\right|_{2^n-1}$. In Fig. 6.1(b), the subtrahend "000...01" is reverted to "111...10", and is added with EAC. This conventional design introduces delay and consumes hardware and power to a degree that is not acceptable for a constant modulo plus/minus one operation in RNS system. If we use this standard design, the delay, area and power dissipation introduced by the modulo incrementer/decrementer will trade off the merit of RNS. The savings on delay, area and power gained by RNS due to its intrinsic properties such as carry-free operations, parallelism and modularity will be counteracted to some extent. This is also one reason why the author of [35] uses the parallel adder structure in Module *Y* to

implement $|Z|_{2^n-1}$ and $|Z-1|_{2^n-1}$. The parallel adder structure in [35] can reduce the delay compared to the serial adder structure where $|Z|_{2^n-1}$ is implemented by a CSA tree and a modulo 2^n-1 adder that are followed by another modulo 2^n-1 adder to implement $|Z-1|_{2^n-1}$. However, as compensation, the circuit becomes more complicated. Thus, it is important to develop new techniques to improve the performance of the modulo incrementer/decrementers that implement the operations of $|Z-1|_{2^n-1}$ and $|Z+1|_{2^n-1}$.



(a) $|Z+1|_{2^n-1}$ (b) $|Z-1|_{2^n-1}$

Fig. 6.1. The CPA-based modulo incrementer/decrementer

In this chapter, we use the new MUX-based algorithm proposed in Chapter 5 to design new modulo incrementer/decrementer that implement the operations of $|Z-1|_{2^n-1}$ and $|Z+1|_{2^n-1}$. The design and FPGA implementation of the MUX-based modulo incrementer/decrementer is carried out. The implementation results show that the proposed design has better performance than those based on adders in terms of area, delay and power dissipation.

6.2 Proposed Modulo Decrementer

Table 6.1. Truth table of binary/modulo decrement

Z	$Y = Z\text{-}1$	$Y = \lvert Z-1 \rvert_{2^n-1}$
0···000	**1···111**	**1···110**
0···001	0···000	0···000
0···010	0···001	0···001
0···011	0···010	0···010
⋮	⋮	⋮
1···111	1···110	1···110

The comparison of the binary and modulo decrement operations is given in Table 6.1. It is seen that the only difference of their truth table is in the case of Z=0···000. When Z=0···000, the result of binary decrement is 1···111, whereas it is 1···110 for modulo decrement. These two decrement results differ from each other at their LSB. The LSB of the binary decrement is '1', whereas it is '0' for the modulo decrement. Based on this feature, we can have the modulo decrement algorithm as Proposition 6.1.

Proposition 6.1 Given any n-bit binary input Z, we get its modulo decrement result Y $= \lvert Z-1 \rvert_{2^n-1}$ as follows.

$$Y = \lvert Z-1 \rvert_{2^n-1} \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (Z_0 + Z_1 + \cdots + Z_{j-1})}, & 1 \le j \le n-1 \\ Y_j = Z_0 \oplus (Z_0 + Z_1 + \cdots + Z_{n-1}), & j = 0 \end{cases} \quad (6.1)$$

Proof:

1) In Table 6.1, for all inputs Z≠0, namely, $Z_0 + Z_1 + \cdots + Z_{n-1} \neq 0$, we have $Y = Z - 1 = \lvert Z-1 \rvert_{2^n-1}$. Thus, according to Proposition 5.1, we have

$$Y = \lvert Z-1 \rvert_{2^n-1} \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (Z_0 + Z_1 + \cdots + Z_{j-1})}, & 1 \le j \le n-1 \\ Y_j = \overline{Z_0}, & j = 0 \end{cases}$$

2) In the case of Z=0, namely, $Z_0 + Z_1 + \cdots + Z_{n-1} = 0$, the binary decrement result and the modulo decrement result differ with each other at their LSB. The LSB of $Y = |Z-1|_{2^n-1}$ is Y_0 that is the complement of K_0, the LSB of $K = Z-1$. Thus, according to Proposition 5.1, we have

$$Y = |Z-1|_{2^n-1} \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (Z_0 + Z_1 + \cdots + Z_{j-1})}, & 1 \le j \le n-1 \\ Y_j = Z_0, & j = 0 \end{cases}$$

Combining the above two cases, we have

$$Y = |Z-1|_{2^n-1} \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (Z_0 + Z_1 + \cdots + Z_{j-1})}, & 1 \le j \le n-1 \\ Y_j = Z_0 \overline{Z_0 + Z_1 + \cdots + Z_{n-1}} + \overline{Z_0}(Z_0 + Z_1 + \cdots + Z_{n-1}) \\ \quad = Z_0 \oplus (Z_0 + Z_1 + \cdots + Z_{n-1}), & j = 0 \end{cases}$$

■

Based on Proposition 6.1, we can design a new modulo decrementer using the proposed binary decrementer in Fig 5.3. By comparing equation (5.1) and (6.1), we can see that the modulo decrement result Y_j is equal to the binary decrement result in the case of $1 \le j \le n-1$. The only difference is Y_0. It is $Y_0 = \overline{Z_0}$ in equation (5.1) and $Y_0 = Z_0 \oplus (Z_0 + Z_1 + \cdots + Z_{n-1})$ in equation (6.1). Based on this analysis, we use a MUX to take the place of the inverter corresponding to Y_0 in Fig. 5.3. The inputs of this MUX are Z_0, the LSB of the input Z, and its complement. From equation (5.2), we have $C_{out} = D_{n-1} = Z_0 + Z_1 + \cdots + Z_{n-1}$, then $Y_0 = Z_0 \oplus (Z_0 + Z_1 + \cdots + Z_{n-1}) = Z_0 \oplus C_{out}$. Thus, we can use the carry-out signal C_{out} of the binary decrementer as the select signal of the MUX used to derive Y_0. Then, we get a new design of the modulo decrementer as shown in Fig. 6.2. When the carry-out signal is one, it means that the input Z is non-zero

(Z≠0···000). Then the output of the proposed modulo decrementer is exactly the output of the binary decrementer. When the carry-out signal is zero, it means that the input Z is zero (Z=0···000). Then the LSB of the modulo decrementer is the complement of the LSB of the binary decrementer, while other bits of the modulo decrementer are the same as the output of the binary decrementer.

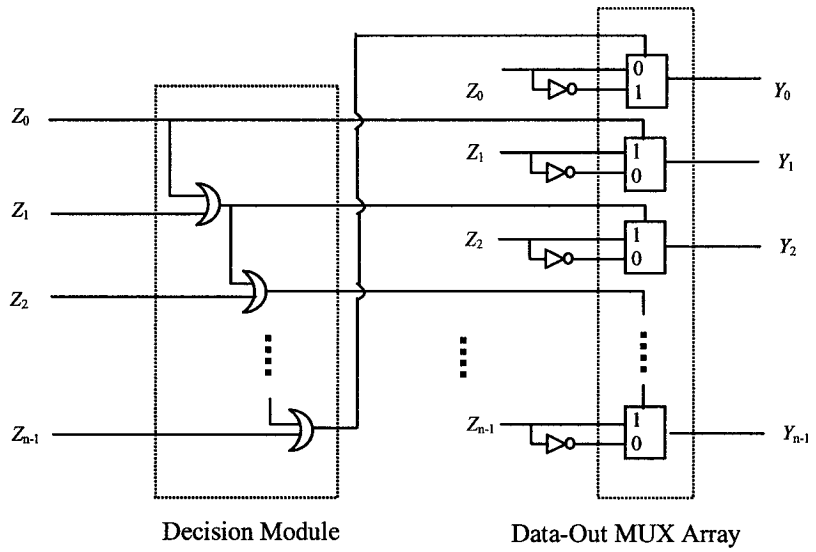


Fig. 6.2. The proposed MUX-based modulo decrementer

6.3 Proposed Modulo Incrementer

We now propose a new algorithm to implement a modulo incrementer based on the proposed MUX-based modulo decrementer. We rewrite Lemma 4 which is needed for the proof of Proposition 6.2 as follows.

Lemma 4 $\left|2^n - 1 - X\right|_{2^n-1} = \overline{X} = \overline{x}_{n-1}\overline{x}_{n-2}\cdots\overline{x}_0$ (6.2)

for any non-zero n-bit binary number $X = x_{n-1}x_{n-2}\cdots x_0$.

Proposition 6.2 Given any n-bit binary number Z, we have

$$|Z+1|_{2^n-1} = \begin{cases} \overline{|\overline{Z}-1|_{2^n-1}}, & Z \neq 2^n-2 \\ |\overline{Z}-1|_{2^n-1}, & Z = 2^n-2 \end{cases} \tag{6.3}$$

Proof:

According to Lemma 4, for $Z \neq 0$, we have $\overline{Z} = |2^n - 1 - Z|_{2^n-1}$.

If $Z \neq 2^n - 2$, $|Z+1|_{2^n-1} \neq 0$, then we have

$$|Z+1|_{2^n-1} = \overline{\overline{|Z+1|_{2^n-1}}}$$

$$= \overline{|2^n - 1 - |Z+1|_{2^n-1}|_{2^n-1}} \text{ by Lemma 4}$$

$$= \overline{|2^n - 1 - Z - 1|_{2^n-1}}$$

$$= \overline{|\overline{Z}-1|_{2^n-1}} \text{ by Lemma 4}$$

If $Z = 2^n - 2$, $|Z+1|_{2^n-1} = 0$, then we have

$$|\overline{Z}-1|_{2^n-1} = |2^n - 1 - Z - 1|_{2^n-1} \text{ by Lemma 4}$$

$$= |2^n - 2 - (2^n - 2)|_{2^n-1} = 0$$

Thus, we have $|Z+1|_{2^n-1} = |\overline{Z}-1|_{2^n-1}$.

■

Proposition 6.2 eliminates the all ones representation of zero. The typical modulo incrementer design in Fig 6.1. (a) has the double representation of zero. Proposition 6.2 will be simplified if the double representation of zero is adopted.

Collorary 6.1 Given any *n*-bit binary number Z, if the double representation of zero is adopted, we have

$$|Z+1|_{2^n-1} = \overline{|\overline{Z}-1|_{2^n-1}} \tag{6.4}$$

Proof:

According to Proposition 6.2, for $Z = 2^n - 2$, we have

$$1)\ |Z+1|_{2^n-1} = \overline{|\overline{Z}-1|_{2^n-1}}$$

$$= \overline{|2^n - 1 - Z - 1|_{2^n-1}} \text{ by Lemma 4}$$

$$= \overline{|2^n - 2 - (2^n - 2)|_{2^n-1}}$$

$$=<11\cdots1>_n$$

$$2)\ |Z+1|_{2^n-1} = |\overline{Z}-1|_{2^n-1}$$

$$= |2^n - 1 - Z - 1|_{2^n-1} \text{ by Lemma 4}$$

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

$$=<00\cdots0>_n$$

Since the double representation of zero, all ones and all zeroes, is allowed, both $<11\cdots1>_n$ and $<00\cdots0>_n$ represent zero. Then equation $|Z+1|_{2^n-1} = |\overline{Z}-1|_{2^n-1}$ is covered by equation $|Z+1|_{2^n-1} = \overline{|\overline{Z}-1|_{2^n-1}}$.

■

By applying Proposition 6.2 to Proposition 6.1, we can derive a new modulo increment algorithm as follows.

Proposition 6.3 Given any *n*-bit binary input *Z*, we get its modulo increment result *Y* $= |Z+1|_{2^n-1}$ as follows.

$$Y = |Z+1|_{2^n-1} \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})} \oplus (\overline{Z_0} Z_1 \cdots Z_{n-1}), 1 \le j \le n-1 \\ Y_j = Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}) \oplus (\overline{Z_0} Z_1 \cdots Z_{n-1}), \quad j = 0 \end{cases} \quad (6.5)$$

Proof:

1) In the case of $Z \neq 2^n - 2$, namely, $\overline{Z_0} Z_1 \cdots Z_{n-1} = 0$, according to Proposition 6.1 and 6.2, we have

$$|Z+1|_{2^n-1} = \overline{|\overline{Z}-1|_{2^n-1}} = \begin{cases} Y_j = \overline{Z_j} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}}), 1 \le j \le n-1 \\ Y_j = \overline{Z_0} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}), \quad j = 0 \end{cases}$$

Since $Y_j = \overline{Z_j} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}}) \qquad 1 \le j \le n-1$

$$= \overline{Z_j}\,\overline{(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})} + Z_j(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})$$

$$= \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}$$

$$Y_j = \overline{\overline{Z_0} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})} \qquad j = 0$$

$$= Z_0\,\overline{(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})} + \overline{Z_0}(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})$$

$$= Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})$$

Thus, in the case of $Z \neq 2^n - 2$, we have

$$|Z+1|_{2^n-1} = \overline{|\overline{Z}-1|_{2^n-1}} = \begin{cases} Y_j = \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}, 1 \le j \le n-1 \\ Y_j = Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}), \quad j = 0 \end{cases} \tag{6.6}$$

2) In the case of $Z = 2^n - 2$, namely, $\overline{Z_0} Z_1 \cdots Z_{n-1} = 1$, according to Proposition 6.1 and 6.2, we have

$$|Z+1|_{2^n-1} = |\overline{Z}-1|_{2^n-1} = \begin{cases} Y_j = \overline{\overline{Z_j} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}, 1 \le j \le n-1 \\ Y_j = \overline{Z_0} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}), \quad j = 0 \end{cases}$$

Since $Y_j = \overline{\overline{Z_j} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})} \qquad 1 \le j \le n-1$

$$= Z_j\,\overline{(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})} + \overline{Z_j}(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})$$

$$= Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})$$

$$Y_j = \overline{Z_0} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}) \qquad j = 0$$

$$= \overline{Z_0}\,\overline{(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})} + Z_0(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})$$

$$= \overline{Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})}$$

Thus, in the case of $Z = 2^n - 2$, we have

$$|Z+1|_{2^n-1} = |\overline{Z}-1|_{2^n-1} = \begin{cases} Y_j = Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}}), 1 \le j \le n-1 \\ Y_j = \overline{Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})}, \quad j = 0 \end{cases} \tag{6.7}$$

3) Combining equations (6.6) and (6.7), we have

For $1 \le j \le n-1$,

$$Y_j = Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})(\overline{Z_0} Z_1 \cdots Z_{n-1}) + \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})(\overline{Z_0} Z_1 \cdots Z_{n-1})}$$

$$= \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})} \oplus (\overline{Z_0} Z_1 \cdots Z_{n-1})$$

For $j = 0$,

$$Y_j = Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})\overline{(\overline{Z_0} Z_1 \cdots Z_{n-1})} + \overline{Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})}(\overline{Z_0} Z_1 \cdots Z_{n-1})$$

$$= Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}) \oplus (\overline{Z_0} Z_1 \cdots Z_{n-1})$$

Thus, we have

$$Y = |Z+1|_{2^n-1} \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})} \oplus (\overline{Z_0} Z_1 \cdots Z_{n-1}), 1 \le j \le n-1 \\ Y_j = Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}) \oplus (\overline{Z_0} Z_1 \cdots Z_{n-1}), \quad j = 0 \end{cases}$$

■

If the double representation of zero is allowed, Proposition 6.3 can be simplified as follows.

Collorary 6.2 Given any *n*-bit binary input *Z*, if the double representation of zero is allowed, we get its modulo increment result $Y = |Z+1|_{2^n-1}$ as follows.

$$Y = |Z+1|_{2^n-1} \Leftrightarrow \begin{cases} Y_j = \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}, 1 \le j \le n-1 \\ Y_j = Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}), \quad j = 0 \end{cases} \tag{6.8}$$

Proof:

According to Proposition 6.1 and Collorary 6.1, we have

$$|Z+1|_{2^n-1} = \overline{|\overline{Z}-1|_{2^n-1}} = \begin{cases} Y_j = \overline{Z_j} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}}), 1 \le j \le n-1 \\ Y_j = \overline{\overline{Z_0} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})}, \quad j = 0 \end{cases}$$

Since $Y_j = \overline{Z_j} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}}) \qquad 1 \le j \le n-1$

$$= \overline{Z_j}\,\overline{(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})} + Z_j(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})$$

$$= \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}$$

$$Y_j = \overline{\overline{Z_0} \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})} \qquad j = 0$$

$$= Z_0\,\overline{(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})} + \overline{Z_0}(\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})$$

$$= Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}})$$

Thus, we have

$$|Z+1|_{2^n-1} = \overline{|\overline{Z}-1|_{2^n-1}} = \begin{cases} Y_j = \overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}, 1 \le j \le n-1 \\ Y_j = Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}), \quad j = 0 \end{cases}$$

■

The proposed MUX-based modulo incrementer (single representation of zero) is presented in Fig. 6.3. This modulo incrementer is to implement equation (6.5). It is composed of a decision module, a data-out MUX array and a data-select MUX array controlled by a n-input AND gate. The data-select MUX is used to select the correct output between the two cases in equation (6.3) respectively.

Corresponding to the case of $Z \neq 2^n - 2$ ($\overline{Z_0}Z_1 \cdots Z_{n-1} = 0$), equation (6.5) equals to equation (6.6). By comparing equation (6.1) and (6.6), we can observe that in equation (6.6), $\overline{Z_j \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{j-1}})}, 1 \le j \le n-1$ and $Z_0 \oplus (\overline{Z_0} + \overline{Z_1} + \cdots + \overline{Z_{n-1}}), j = 0$ can be

implemented based on the proposed modulo decrementer in Fig. 6.2 which is to implement $\overline{Z_j \oplus (Z_0 + Z_1 + \cdots + Z_{j-1})}, 1 \le j \le n-1$ and $Z_0 \oplus (Z_0 + Z_1 + \cdots + Z_{n-1}), j = 0$. In contrast with the modulo decrementer in Fig. 6.2, the input of the DM in Fig. 6.3 is the complement value of Z_i instead of Z_i itself. The output of the data-out MUX array is the implementation of equation (6.6).

In the case of $Z = 2^n - 2$ ($\overline{Z_0} Z_1 \cdots Z_{n-1} = 1$), equation (6.5) equals to equation (6.7). By comparing equation (6.6) and (6.7), we can see that the value of equation (6.7) can be derived by inverting the result of equation (6.6), namely, by inverting the output of the data-out MUX array.

To obtain the desired modulo increment result, we use a n-input AND gate of $\overline{Z_0} Z_1 \cdots Z_{n-1}$ to select either the output of the data-out MUX array or its complement corresponding to the two cases in equation (6.3) of $Z \neq 2^n - 2$ or $Z = 2^n - 2$. This 2-select-1 operation, corresponding to the item of XOR $\overline{Z_0} Z_1 \cdots Z_{n-1}$ in equation (6.5), is implemented by the data-select MUX array as shown in Fig. 6.3.

The proposed MUX-based modulo incrementer (double representation of zero) is depicted in Fig. 6.4. This modulo incrementer is to implement equation (6.8). It is noted that equation (6.8) is the same as equation (6.6). From the above discuss, we know that the output of the data-out MUX array is the result of equation (6.6). Thus, the proposed MUX-based modulo incrementer (double representation of zero) can be obtained by eliminating the data-select MUX array from the circuit in Fig. 6.3.

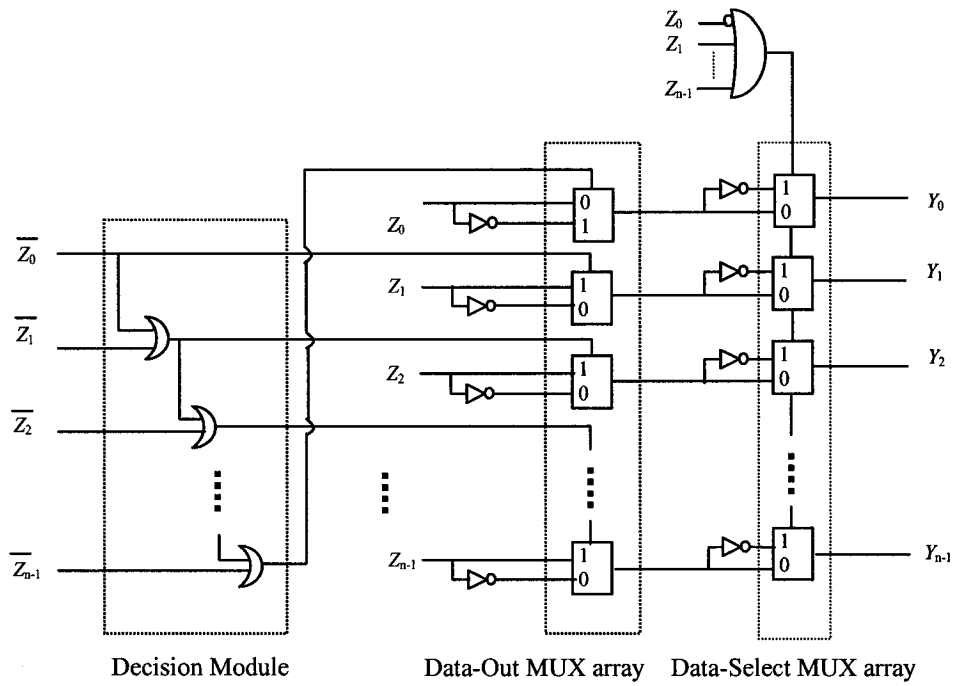


Fig. 6.3. The proposed modulo incrementer (simple representation of zero)

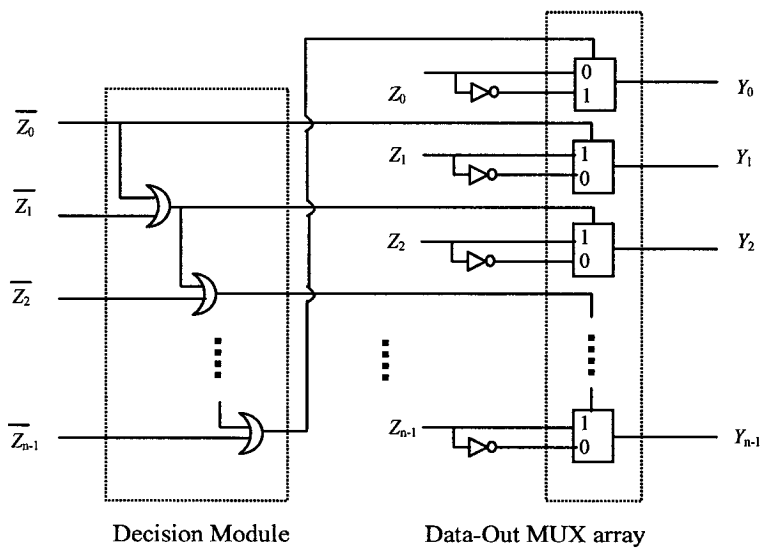


Fig. 6.4. The proposed modulo incrementer (double representation of zero)

6.4 Comparison Study of the Implementation Results

To get a practical performance measure of the modulo incrementer and decrementer, both the proposed MUX-based design and the standard CPA-based design are implemented using Xilinx FPGA technology for the 32-bit and 64-bit cases. Since the standard CPA-based modulo incrementer uses double representation of zero, we correspondingly implement the proposed circuit in Fig. 6.4 to facilitate the comparison of modulo incrementers.

The synthesis and implementation tools are Synopsys's Design Compiler Version 3.4b and Xillinx Alliance M1.3 software. The target technology is a Xillinx 4010e-3 FPGA. The performance evaluation is carried out in terms of power, area and delay. The results of modulo 2^n-1 decrementers are compared in Table 6.2. The results of the FPGA implementation show that the proposed design consumes 50% less hardware than those based on CPA. The delay is also reduced close to 50%. The power consumed by the MUX-based design is almost 20% less than that of the CPA-based design. The results of modulo 2^n-1 incrementers are compared in Table 6.3. The results show that the proposed modulo 2^n-1 incrementer reduces both power and delay to the degree of about 50% than those based on CPA. The area consumed by the proposed design is 40% less than the area of CPA-based design. The reason for this improvement is that the arithmetic operation of the CPA-based modulo incrementer/decrementer is based on addition with ripple carry, while the proposed modulo incrementer/decrementer takes MUX as its arithmetic base. To reduce delay, the traditional incrementer/decrementers based on high-speed adders such as carry select adders or carry look ahead adders could have been used. However, these come at higher cost of power and area.

Table 6.2. Implementation results of the modulo decrementers

	32-bit			*64-bit*		
Decrementers	***Power (mw)***	***Cell Area***	***Delay (ns)***	***Power (mw)***	***Cell Area***	***Delay (ns)***
C_1 : CPA-based	20.4	192	229	40.9	384	447.6
C_2 : MUX-based	16	95	116.7	31.9	191	234.5
$\frac{C_1 - C_2}{C_1} \times 100\%$	21.6	50.1	49	22	50.3	47.6

Table 6.3. Implementation results of the modulo incrementers

	32-bit			*64-bit*		
Incrementers	***Power (mw)***	***Cell Area***	***Delay (ns)***	***Power (mw)***	***Cell Area***	***Delay (ns)***
C_3: CPA-based	24.3	160	225.4	48.5	320	444
C_4 : MUX-based	12.1	95	120.9	24.3	191	238.7
$\frac{C_3 - C_4}{C_3} \times 100\%$	50.2	40.6	46.4	49.9	40.3	46.2

6.5 Summary

In this chapter, new algorithms and designs have been proposed for the modulo 2^n-1 incrementer and decrementer. The proposed MUX-based incrementer and decrementer are more efficient in terms of power, speed and hardware complexity compared to the adder-based designs. An FPGA based comparison of the proposed decrementer with the traditional ones shows that the proposed design requires 50% less hardware while the delay is reduced close to 50%. The power consumed by the MUX-based design is almost 20% less than that of the CPA-based design. The comparison also shows that the proposed incrementer reduces both power and delay to the degree of about 50% than

those based on CPA. The area consumed by the proposed design is 40% less than the area of CPA-based design.

CHAPTER 7

DESIGN AND IMPLEMENTATION OF HIGH-SPEED R/B CONVERTER FOR N_1

7.1 Introduction

The CRT requires a binary inner product operation followed by a modulo-*M* (large-valued) operation that is not very efficient. There are many converters for the three-moduli set $\{2^n, 2^n+1, 2^n-1\}$ proposed in [36]-[45] that use special algorithms derived from the CRT to reduce the size of the modulo operation or avoid the modulo-*M* operation. These converters mostly require big modulo $(2^n+1)(2^n-1)$ operation. In the early design of [38], it is proposed to use four $2n$-bit binary adders, two of them in parallel, to evaluate this summation. The critical delay path includes three consecutive additions of $2n$-bit numbers. The converter proposed in [39] is the reconsideration of hardware realization of the converter from [38]. In [39], the first stage of the converter contains two levels of $2n$-bit carry-save adders (CSA) each comprising of $2n$ full adders since four inputs need to be added. The second stage is a $2n$-bit CPA with EAC used to obtain the modulo 2^{2n}-1 summation. The designs in [42], [43] and [44] reduce the four addends to three, thus only needing one level of CSA. Different methods are presented in these papers. In the literature, there also exist R/B converter designs that avoid modulo

$(2^n+1)(2^n-1)$ operation by using a redundant representation of X [40]. Better performance in terms of area has been reported with compared to the existing nonredundant designs. However, as compensation, certain dynamic range of X is unusable. To further reduce the area and delay of a R/B converter, it is important to develop efficient techniques to reduce the modulo operations.

Recently, several new formulations of the CRT [the Modified Chinese Remainder Theorem and the New Chinese Remainder Theorems (New CRT-I, II, and III)] that reduces the size of the modulo operation have been proposed [24], [26]. For the moduli set $N_1 = \{2^n, 2^n+1, 2^n-1\}$, the direct implementation of the CRT requires a modulo base of $2^n \times (2^n+1) \times (2^n-1)$. By using these techniques and other special algorithms derived from the CRT, a modulo base of $(2^n+1) \times (2^n-1)$ are used [36]-[45].

Proposition 7.1: For the moduli set $N_1 = \{2^n, 2^n+1, 2^n-1\}$, a binary number X can be calculated as

$$X = x_1 + 2^n Y$$

where n >1 and

$$Y = \left| (x_1 - x_2) + 2^{n-1}(2^n+1)(x_3 - 2x_1 + x_2) \right|_{2^{2n}-1}$$

Being in the critical path of the converters, the modulo operation consumes a large portion of hardware and results in large delay. As shown in Proposition 7.1, the modulo 2^{2n}-1 operation is required for the R/B conversion of N_1. In next section, we will propose new algorithms that reduce this modulo operation to two parallel small modulo operations based on 2^n–1 and 2^n+1 respectively.

7.2 Proposed R/B Algorithm for N_1

In this section, we first apply the proposed modulo reduction technique to reduce the modulo size of the R/B algorithm of Proposition 7.1. Then, we further simplify the modulo reduced R/B algorithm for efficient implementation.

We have proposed a R/B algorithm for another permutation of N_1 in Proposition 4.1 to show that the proposed modulo reduction technique can further reduce the modulo size of the Modified CRT. Since in the literature, most of the existing R/B converters based on $N_1 = \{2^n, 2^n+1, 2^n-1\}$ rather than $\{2^n+1, 2^n, 2^n-1\}$, for the convenience of comparison, we now apply the modulo reduction technique to reduce the modulo size of the R/B algorithm of Proposition 7.1.

Proposition 7.2: For three-moduli set $N_1 = \{2^n, 2^n+1, 2^n-1\}$, we have

$$X = x_1 + 2^n Y$$

where n>1, and

$$Y = (2^n+1)\left\lfloor 2^{n-1}x_2 + 2^{n-1}x_3 - x_1 + \frac{x_1 - x_2}{2^n+1} \right\rfloor_{2^n-1} + |x_1 - x_2|_{2^n+1}$$

Proof:

Based on Proposition 7.1, we have

$$X = x_1 + 2^n Y$$

$$Y = \left|(x_1 - x_2) + 2^{n-1}(2^n+1)(x_3 - 2x_1 + x_2)\right|_{2^{2n}-1}$$

$$= \left|(2^{2n} - 2^n - 1)x_1 + (2^{2n-1} + 2^{n-1} - 1)x_2 + 2^{n-1}(2^n+1)x_3\right|_{2^{2n}-1}$$

$$= \left|-2^n x_1 + 2^{n-1}(2^n+1)x_2 - x_2 + 2^{n-1}(2^n+1)x_3\right|_{2^{2n}-1}$$

Using the modulo reduction technique of Proposition 3.1, we have

$$Y=(2^n+1)\left\lfloor 2^{n-1}x_2+2^{n-1}x_3-\frac{2^n x_1+x_2}{2^n+1}\right\rfloor_{2^n-1}+\left|-2^n x_1-x_2\right|_{2^n+1}$$

$$=(2^n+1)\left\lfloor 2^{n-1}x_2+2^{n-1}x_3-x_1+\frac{x_1-x_2}{2^n+1}\right\rfloor_{2^n-1}+\left|x_1-x_2\right|_{2^n+1}$$

■

The R/B algorithm proposed in Proposition 7.2 decomposes the big modulo base of Proposition 7.1 from $(2^n+1)(2^n-1)$ to two parallel small modulo operations based on 2^n+1 and 2^n-1 respectively. This improvement results in a high-speed and low-power VLSI implementation. We use the following example to illustrate the improvement that the proposed method provides over Proposition 7.1 for the R/B conversion of N_1.

Example 7.1: For a R/B converter with 8-bit dynamic range based on the moduli set $N_1=\{2^n,\ 2^n+1, 2^n-1\}$, the specific moduli set {8, 9, 7} is chosen when n=3, since $9\times8\times7=504>2^8=256$. Randomly choose a number from 0 to 255, for example, X=169. Its RNS representation $X=(x_1,x_2,x_3)$ is (1,7,1).

Based on Proposition 7.1, we have

$$Y=\left|(x_1-x_2)+2^{n-1}(2^n+1)(x_3-2x_1+x_2)\right|_{2^{2n}-1}$$

$$=\left|(1-7)\times1+2^2\times(2^3+1)\times(1-2+7)\right|_{(2^3+1)(2^3-1)}$$

$$=\left|210\right|_{63}=21$$

$$X=x_1+2^n\times Y=1+8\times21=169$$

Based on Proposition 7.2, we have

$$Y=(2^n+1)\left\lfloor 2^{n-1}x_2+2^{n-1}x_3-x_1+\frac{x_1-x_2}{2^n+1}\right\rfloor_{2^n-1}+\left|x_1-x_2\right|_{2^n+1}$$

$$= (2^3+1)\left\| 2^2\times 7+2^2\times 1-1+\frac{1-7}{2^3+1}\right\|_{2^3-1}+|1-7|_{2^3+1}$$

$$= 9\times\left\| 28+2^2\times 1-1-\frac{6}{9}\right\|_7+|-6|_9$$

$$= 9\times|30|_7+|-6|_9 = 21$$

$$X = x_1+2^n\times Y = 1+8\times 21 = 169$$

When using Proposition 7.1, the above R/B conversion needs a modulo 63 operation. By using Proposition 7.2, the modulo size is reduced from 63 to 9. That is to say, the length of modulo operation is reduced from 6-bit to 4-bit.

The R/B algorithm proposed in Proposition 7.2 considerably reduces the size of modulo operation. However, it can be further simplified for efficient implementation.

Proposition 7.3: Given the moduli set $N_1 = \{2^n, 2^n+1, 2^n-1\}$, the residue number (x_1, x_2, x_3) is converted into the binary number X by

$$X = x_1+2^n Y$$

where n>1, and

$$Y=\begin{cases}(2^n+1)|T_1+T_2+T_3|_{2^n-1}+|x_1-x_2|_{2^n+1}, & for \quad x_2\in[0,2^n-1], x_1\geq x_2\\ (2^n+1)|T_1+T_2+T_3-1|_{2^n-1}+|x_1-x_2|_{2^n+1}, & for \quad x_2\in[0,2^n-1], x_1<x_2\\ (2^n+1)|T_1+(2^{n-1}-1)+T_3|_{2^n-1}+(x_1+1), & for \quad x_2=2^n\end{cases}$$

where

$$T_1 = |-x_1|_{2^n-1} = \bar{x}_{1,n-1}\cdots\bar{x}_{1,0}$$

$$T_2 = |2^{n-1}x_2|_{2^n-1} = x_{2,0}x_{2,n-1}\cdots x_{2,1}$$

$$T_3 = |2^{n-1}x_3|_{2^n-1} = x_{3,0}x_{3,n-1}\cdots x_{3,1}$$

Proof:

For $N_1 = \{2^n, 2^n+1, 2^n-1\}$, referred to Proposition 7.2, we have

$$X = x_1 + 2^n Y$$

where n>1, and

$$Y = (2^n+1)\left\| 2^{n-1}x_2 + 2^{n-1}x_3 - x_1 + \frac{x_1 - x_2}{2^n+1} \right\|_{2^n-1} + |x_1 - x_2|_{2^n+1}$$

Thus

(1) If $x_2 \in [0, 2^n-1]$, namely $x_{2n} = 0$, since $x_1 \in [0, 2^n-1]$, we have $x_1 - x_2 \in [-(2^n-1), 2^n-1]$.

I. For $x_1 \geq x_2$, $x_1 - x_2 \in [0, 2^n-1]$, then $\frac{x_1 - x_2}{2^n+1} \in [0,1)$. Thus $\left\lfloor \frac{x_1 - x_2}{2^n+1} \right\rfloor = 0$.

We calculate Y as:

$$Y = (2^n+1)\left|2^{n-1}x_2 + 2^{n-1}x_3 - x_1\right|_{2^n-1} + |x_1 - x_2|_{2^n+1}$$

$$= (2^n+1)\left|T_1 + T_2 + T_3\right|_{2^n-1} + |x_1 - x_2|_{2^n+1} \qquad (7.1)$$

where

$T_1 = |-x_1|_{2^n-1} = \bar{x}_{1,n-1} \cdots \bar{x}_{1,0}$ by Lemma 5

$T_2 = \left|2^{n-1}x_2\right|_{2^n-1} = x_{2,0}x_{2,n-1} \cdots x_{2,1}$ by Lemma 6

$T_3 = \left|2^{n-1}x_3\right|_{2^n-1} = x_{3,0}x_{3,n-1} \cdots x_{3,1}$ by Lemma 6

II. For $x_1 < x_2$, $x_1 - x_2 \in [-(2^n-1), 0)$, then $\frac{x_1 - x_2}{2^n+1} \in (-1, 0)$.

We calculate Y as:

$$Y = (2^n+1)\left|2^{n-1}x_2 + 2^{n-1}x_3 - x_1 - 1\right|_{2^n-1} + |x_1 - x_2|_{2^n+1}$$

$$= (2^n+1)\left|T_1 + T_2 + T_3 - 1\right|_{2^n-1} + |x_1 - x_2|_{2^n+1} \qquad (7.2)$$

(2) If $x_2 = 2^n$, namely $x_{2n} = 1$, $x_{2,n-1} = \cdots = x_{2,0} = 0$, since $x_1 \in [0, 2^n - 1]$, we have

$x_1 - x_2 \in [-2^n, -1]$, thus $\frac{x_1 - x_2}{2^n + 1} \in (-1, 0)$.

We calculate Y as:

$$Y = (2^n + 1)\left|2^{n-1}2^n + 2^{n-1}x_3 - x_1 - 1\right|_{2^n-1} + \left|x_1 - x_2\right|_{2^n+1}$$

$$= (2^n + 1)\left|(2^{n-1} - 1) + 2^{n-1}x_3 - x_1\right|_{2^n-1} + \left|x_1 - 2^n\right|_{2^n+1}$$

$$= (2^n + 1)\left|T_1 + (2^{n-1} - 1) + T_3\right|_{2^n-1} + \left|x_1 - 2^n\right|_{2^n+1}$$

Since $x_1 \in [0, 2^n - 1]$, $x_1 + 1 < 2^n + 1$, then $\left|x_1 - 2^n\right|_{2^n+1} = \left|x_1 + 1\right|_{2^n+1} = x_1 + 1$.

Then, we have

$$Y = (2^n + 1)\left|T_1 + (2^{n-1} - 1) + T_3\right|_{2^n-1} + (x_1 + 1) \tag{7.3}$$

In conclusion, from equations (7.1), (7.2) and (7.3), Proposition 7.3 holds for any residue number (x_1, x_2, x_3).

■

We use the following example to illustrate how Proposition 7.3 works.

Example 7.2: For a R/B converter with 8-bit dynamic range based on the moduli set $N_1 = \{2^n, 2^n + 1, 2^n - 1\}$, the specific moduli set {8, 9, 7} is chosen when n=3, since $9 \times 8 \times 7 = 504 > 2^8 = 256$. Randomly choose a number from 0 to 255, for example, X=169. Its RNS representation $X = (x_1, x_2, x_3)$ is (1, 7, 1) =(001, 0111, 001).

Based on Proposition 7.3, the calculation for finding the binary number is as follows.

$$T_1 = \bar{x}_{1,2}\bar{x}_{1,1}\bar{x}_{1,0} = 110$$

$$T_2 = x_{2,0}x_{2,2}x_{2,1} = 111$$

$$T_3 = x_{3,0}x_{3,2}x_{3,1} = 100$$

Since $x_1 < x_2$, $x_2 < 2^3$, we have

$$Y = (2^n + 1)\left|T_1 + T_2 + T_3 - 1\right|_{2^n-1} + \left|x_1 - x_2\right|_{2^n+1}$$

$$= (2^3 + 1) \times \left|110 + 111 + 100 - 1\right|_7 + \left|001 - 111\right|_9$$

$$= 9 \times 010 + 011 = 21$$

$$X = x_1 + 2^n Y$$

$$= 1 + 2^3 \times 21 = 169$$

With the modulo reduced R/B algorithm presented in Proposition 7.3, we can implement a fast design of the R/B converter based on N_1. It is noted that there are some arithmetic components such as the binary increment $x_1 + 1$, the modulo decrement $\left|T_1 + T_2 + T_3 - 1\right|_{2^n-1} = \left|Z - 1\right|_{2^n-1}$ and the modulo subtraction $\left|x_1 - x_2\right|_{2^n+1}$ in Proposition 7.3 that serve as building blocks in the proposed R/B converter. The new MUX-based designs of the binary incrementer and the modulo decrementer have been given in Chapter 5 and Chapter 6 respectively. In the following section, we propose a novel design for the modulo 2^n+1 subtractor for constructing a complete R/B converter.

7.3 Novel Designs of Modulo 2^n+1 Subtractor

In the literature, there exist many papers on the design of modulo *m* adders [26], [47]. Some papers particularly propose design for modulo 2^n+1 adders. Although, as what these papers declare, modulo 2^n+1 subtraction can be implemented using the modulo 2^n+1 adders with some modification, very few papers give directly the design of the modulo 2^n+1 subtraction. In this section, we propose a new fast design for the determination of $\left|x - y\right|_{2^n+1}$ using the MUX-based decrement technique.

The residue difference of two n-bit residue digits, $(x - y)$ mod 2^n+1, is the residue of the difference $x - y$ with respect to the modulus 2^n+1. The operation may be defined as follows.

$$D = |x - y|_{2^n+1} = \begin{cases} x - y, & x - y \in [0,\ 2^n) \\ x - y + 2^n + 1, & x - y \in (-2^n,\ 0) \end{cases} \tag{7.4}$$

First, we use Proposition 7.4 to simplify the calculation of $|x - y|_{2^n+1}$ in equation (7.4).

Proposition 7.4 Given any two n-bit binary input x and y, we have

$$D = |x - y|_{2^n+1} = \begin{cases} S, & x - y \in [0,\ 2^n) \\ S + 1, & x - y \in (-2^n,\ 0) \end{cases} \tag{7.5}$$

where $S = x + \bar{y} + 1$.

Proof:

If $x - y \in [0,\ 2^n)$, from equation (7.4), by using 2's complement subtraction, we have

$$D = |x - y|_{2^n+1} = x - y = x + \bar{y} + 1 = S$$

If $x - y \in (-2^n,\ 0)$, we have

$$\begin{aligned} D = |x - y|_{2^n+1} &= x - y + 2^n + 1 \\ &= x + (2^n - 1) - y + 2 \\ &= x + (\bar{y} + 1) + 1 \quad \text{by Lemma 4} \\ &= S + 1 \end{aligned}$$

■

Let the representation of x as a binary number be $x_{n-1}, \cdots, x_0$. Similarly let y be represented by $y_{n-1}, \cdots, y_0$. And the difference S, $S = x + \bar{y} + 1$, be represented by

$s_{n-1},\cdots,s_0$. c_n is the carry-out signal generated by the subtraction. When $x \geq y$, which implies $c_n = 1$, we have $D = S$. When $x - y \in (-2^n,\ 0)$, which implies $c_n = 0$, we have $D = S + 1$.

Equation (7.5) provides us a way to implement area-time efficient modulo 2^n+1 subtractors. The proposed modulo subtractor is depicted in Fig. 7.1 (a). Fig. 7.1 (b) shows the block diagram of the unit. First, the circuit adds the 2's complement of y to x. Then the proposed MUX-based binary incrementer is used to form $S+1$. C_{out} is the carry-out signal of the incrementer. When C_{out} is 0, it means that there is carry-out. If c_n is 0, which implies $x - y \in (-2^n,\ 0)$, the output d_n is $\overline{C_{out}}$ and $d_{n-1},\cdots,d_0$ is the output of the incrementer. If c_n is 1, which implies $x \geq y$, then d_n is 0 and $d_{n-1},\cdots,d_0$ is $s_{n-1},\cdots,s_0$.

The hardware required in the new modulo 2^n+1 subtractor shown in Fig. 7.1 (a) is as following: n FAs, $2n$ MUXs, $2n+1$ inverters and n-1 OR gates. The delay of the proposed modulo subtractor t_{sub} is the sum of the delay of a MUX t_{MUX}, the delay of three Inverters t_{INV}, the delay of a OR gate t_{OR} and the delay of a CPA $t_{CPA(n)}$, i.e., $t_{sub} = t_{MUX} + 3\,t_{INV} + t_{OR} + t_{CPA(n)}$.

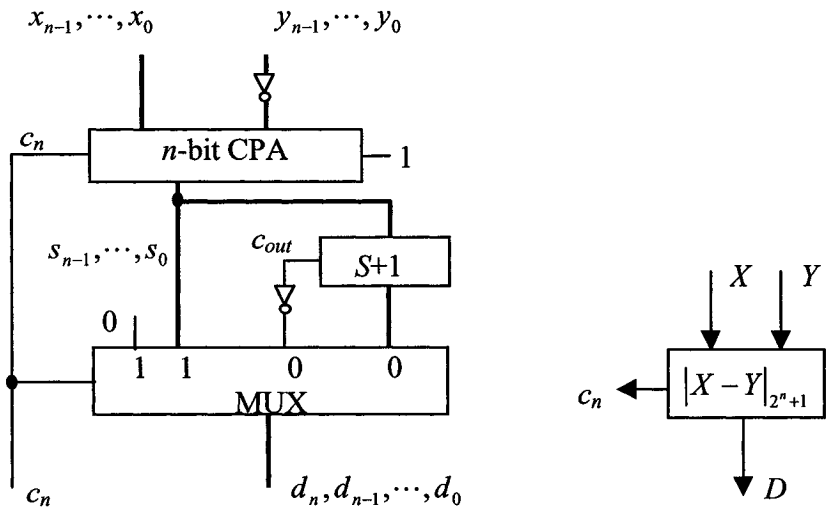


Fig. 7.1. The proposed modulo 2^n+1 subtractor

7.4 Proposed High-speed R/B Converter for N_1

In this section, based on the proposed modulo reduced R/B algorithms and the novel MUX-based designs of the binary incrementer, the modulo decrementer and subtractor, we present an efficient design of R/B converters for the three-moduli set N_1.

Based on Proposition 7.3, we propose a new R/B converter for N_1 using n-bit adders. The formulas of Y in Proposition 7.3 can be rewritten as follows.

$$Y=\begin{cases}(2^n+1)|Z|_{2^n-1}+|x_1-x_2|_{2^n+1}, & for \quad x_2\in[0,2^n-1], x_1\geq x_2\\ (2^n+1)|Z-1|_{2^n-1}+|x_1-x_2|_{2^n+1}, & for \quad x_2\in[0,2^n-1], x_1<x_2\\ (2^n+1)|Z'|_{2^n-1}+(x_1+1), & for \quad x_2=2^n\end{cases} \tag{7.6}$$

where $Z=T_1+T_2+T_3$ and $Z'=T_1+(2^{n-1}-1)+T_3$.

Calculation of $|Z-1|_{2^n-1}$, $|Z|_{2^n-1}$ or $|Z'|_{2^n-1}$ We use the following two steps to present the n-bit 1's complement adder unit, which is used to generate $|Z-1|_{2^n-1}$, $|Z|_{2^n-1}$ or $|Z'|_{2^n-1}$ in equation (7.6).

1) Z and Z' is calculated as follows. For the first two cases of (7.6), we need to calculate Z for $x_2 \in [0,2^n-1], x_1 \geq x_2$ or for $x_2 \in [0,2^n-1], x_1 < x_2$ respectively. For the third case $x_2 = 2^n$, we need to compute the number Z'. Since the only difference between Z and Z' is that the second item of Z is T_2, and the second item of Z' is $2^{n-1}-1$, we can integrate the calculation of these two numbers Z and Z' by using MUX and CSA. If $x_2 \in [0,2^n-1]$, then $x_{2,n}=0$, we have $T_2 = x_{2,0}x_{2,n-1}\cdots x_{2,1}$. If $x_2 = 2^n$, then $x_{2,n}=1, x_{2,n-1}=\cdots=x_{2,0}=0$, we have $2^{n-1}-1=0\langle 1\cdots 1\rangle_{n-1}$. We can use $x_{2,n}$ to select T_2 or $2^{n-1}-1$ as input to CSA. The addition is shown in Fig. 7.2 (a) and (b). Fig. 7.2 (b) shows the block diagram of the unit. It consists of n FAs, n MUXs, and n inverters. The delay of this unit is the delay of a FA plus the delay of a MUX. This circuit produces two numbers $C_{n-1}\ C_{n-2}\ldots\ C_1\ C_0$ and $S_{n-1}\ S_{n-2}\ldots\ S_1\ S_0$. We denote $C = C_{n-1}\ C_{n-2}\ldots\ C_1\ C_0$ and $S = S_{n-1}\ S_{n-2}\ldots\ S_1\ S_0$, and then

$$C+S=\begin{cases} T_1+T_2+T_3, & for \quad x_{2,n}=0 \\ T_1+(2^{n-1}-1)+T_3, & for \quad x_{2,n}=1 \end{cases}$$

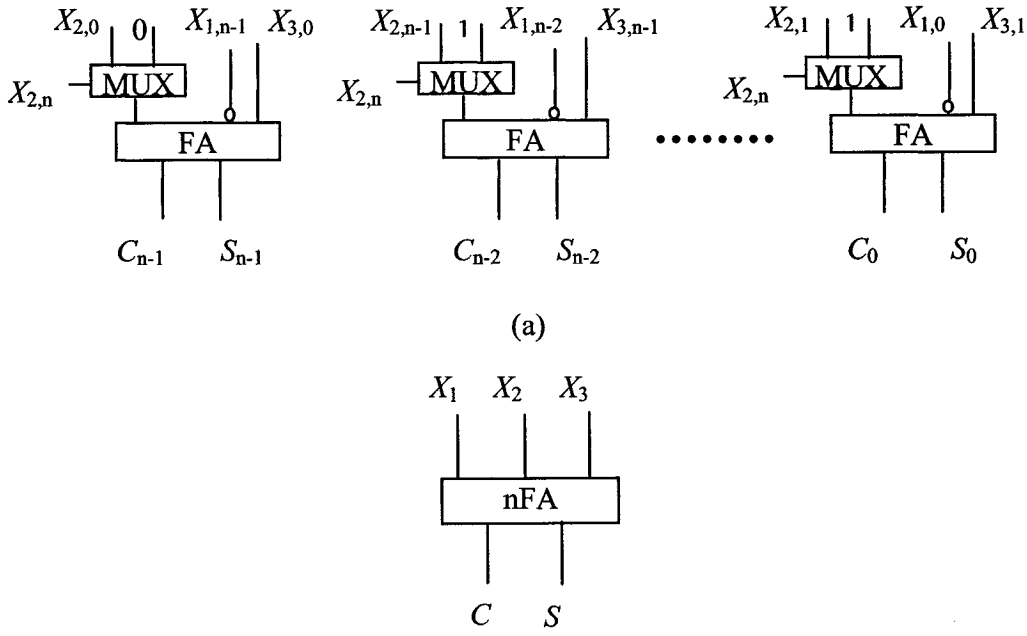


Fig. 7.2. The addition unit using adders and MUXs

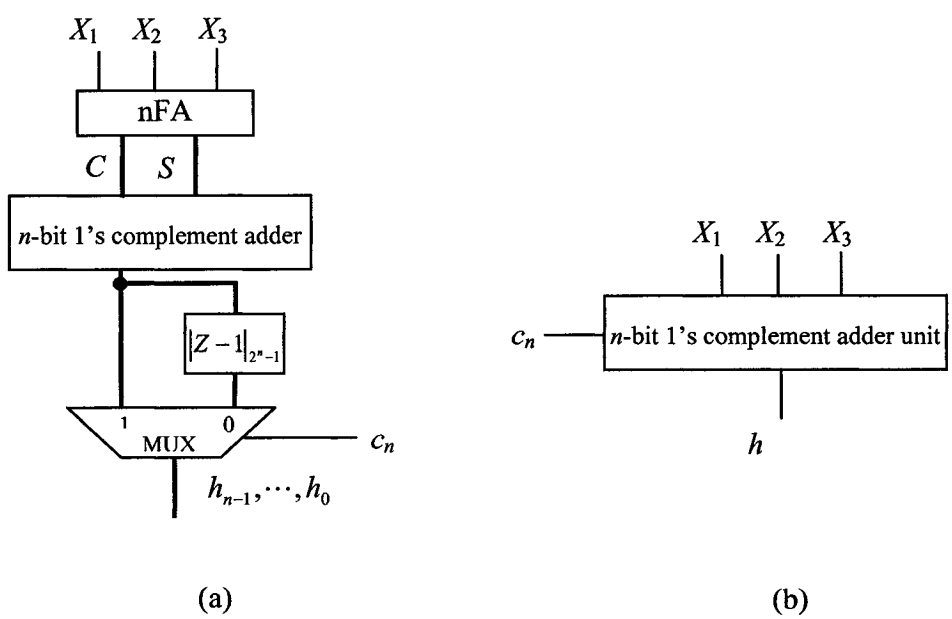


Fig. 7.3. The *n*-bit 1's complement adder unit

2) In Fig. 7.3, we present the n-bit 1's complement adder unit, which is used to generate $|Z-1|_{2^n-1}$, $|Z|_{2^n-1}$ or $|Z'|_{2^n-1}$ in equation (7.6). The unit nFA, which is used to produce C and S, is connected to a n-bit 1's complement adder. The output of the 1's complement adder is connected to a n-bit MUX-based modulo decrementer. The following MUX array using the carry-out signal c_n of the modulo 2^n+1 subtractor as its selecting signal, produces the value $h_{n-1},\cdots,h_0$. If $x_1 < x_2$, the carry-out c_n is '0', then the value $h_{n-1},\cdots,h_0$ is the output of the modulo decrementer $|Z-1|_{2^n-1}$. If $x_1 \geq x_2$ or $x_2 = 2^n$, the carry-out c_n is '1', then the value $h_{n-1},\cdots,h_0$ is formed by the output of the 1's complement adder.

Calculation of $|x_1 - x_2|_{2^n+1}$ or x_1+1: As shown in Fig. 7.4, we perform the modulo subtraction $|x_1 - x_2|_{2^n+1}$ using the proposed n-bit modulo 2^n+1 subtractor in Fig. 7.1. In equation (7.6), it can be noted that, for $x_2 \in [0,2^n-1], x_1 \geq x_2$ or for $x_2 \in [0,2^n-1], x_1 < x_2$, we have $x_{2,n} = 0$. Thus, we feed the modulo subtractor with $x_{2,n-1}\cdots x_{2,0}$ as the subtrahend. The n-bit modulo subtractor performs the operation of $|x_1 - x_2|_{2^n+1} = |x_{1,n-1}\cdots x_{1,0} - x_{2,n-1}\cdots x_{2,0}|_{2^n+1}$. If the carry-out signal c_n is 0, it implies the case of $x_2 \in [0,2^n-1], x_1 < x_2$. If c_n is 1, it means $x_2 \in [0,2^n-1], x_1 \geq x_2$ or $x_2 = 2^n$. In the case of $x_2 = 2^n$, namely $x_{2,n} = 1$, we have $x_1 + 1$. We apply the proposed binary incrementer in Chapter 5 to get the result of $x_1 + 1$. Finally, We can use $x_{2,n}$ to select the correct value as the output $k_n, k_{n-1},\cdots,k_0$.

The hardware required in the modulo 2^n+1 subtraction unit in Fig. 7.4 (a) is as following: n FAs, $4n$ MUXs, $4n$ +1 inverters and $2n$-2 OR gates. The delay of the

proposed modulo subtraction unit t_{mod} is the sum of the delay of two MUXs t_{MUX}, the delay of three Inverters t_{INV}, the delay of a OR gate t_{OR} and the delay of a CPA $t_{CPA(n)}$, i.e., $t_{mod} = 2\,t_{MUX} + 3\,t_{INV} + t_{OR} + t_{CPA(n)}$.

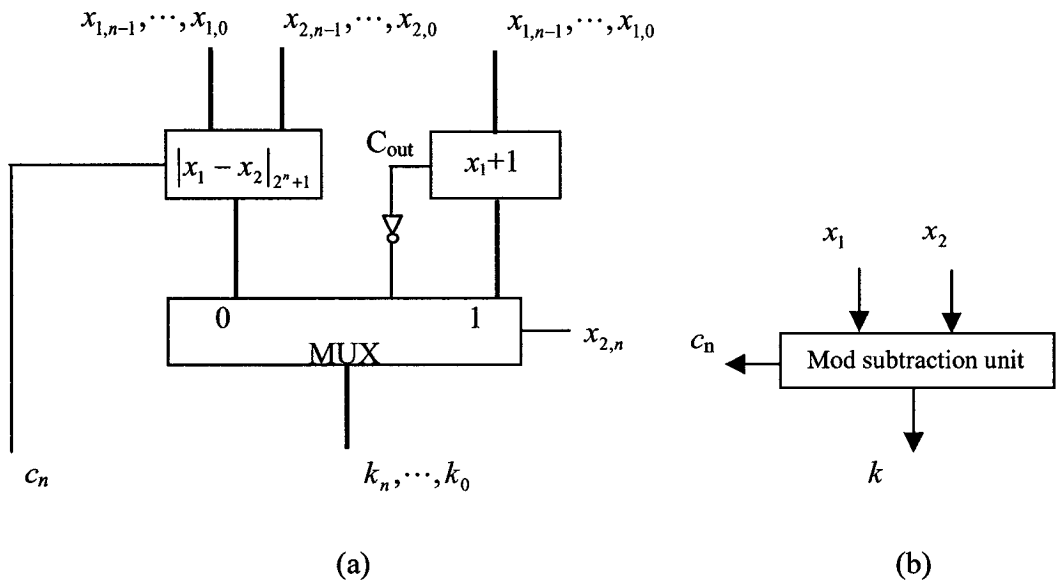


Fig. 7.4. The modulo subtraction unit

Final Addition in Calculation of (7.6) With the output $h_{n-1},\cdots,h_0$ formed by the n-bit 1's complement adder unit and the output $k_n,k_{n-1}\cdots,k_0 \in [0,2^n]$ derived from the modulo subtraction unit, now equation (7.6) can be simply rewritten as follows.

$$Y = (2^n+1) < h_{n-1},\cdots,h_0 > + < k_n,k_{n-1},\cdots,k_0 >$$

$$= 2^n < h_{n-1},\cdots,h_0 > + < h_{n-1},\cdots,h_0 > + < k_n,k_{n-1},\cdots,k_0 > \qquad (7.7)$$

In Fig. 7.5, we present a simple circuit to implement equation (7.7). A n-bit CPA is used to add $h_{n-1},\cdots,h_0$ and $k_{n-1}\cdots,k_0$ to get $Y_{n-1}\cdots Y_0$ of the final binary number X. In the case of $k_n,k_{n-1}\cdots,k_0 \in [0,2^n-1]$, namely $k_n = 0$, if the carry-out signal C_{out} is 1,

$Y_{2n-1}\cdots Y_n$ of X is formed by incrementing $h_{n-1},\cdots,h_0$ by one instead of $h_{n-1},\cdots,h_0$. In the case of $k_n,k_{n-1}\cdots,k_0 = 2^n$, namely $k_n = 1$ and $k_{n-1}\cdots,k_0 = 0$, the carry-out signal C_{out} is 0 and the value $h_{n-1},\cdots,h_0$ is incremented by one to form $Y_{2n-1}\cdots Y_n$ of X. In other cases, namely, k_n *OR* $C_{out} = 0$, $Y_{2n-1}\cdots Y_n$ is formed by $h_{n-1},\cdots,h_0$.

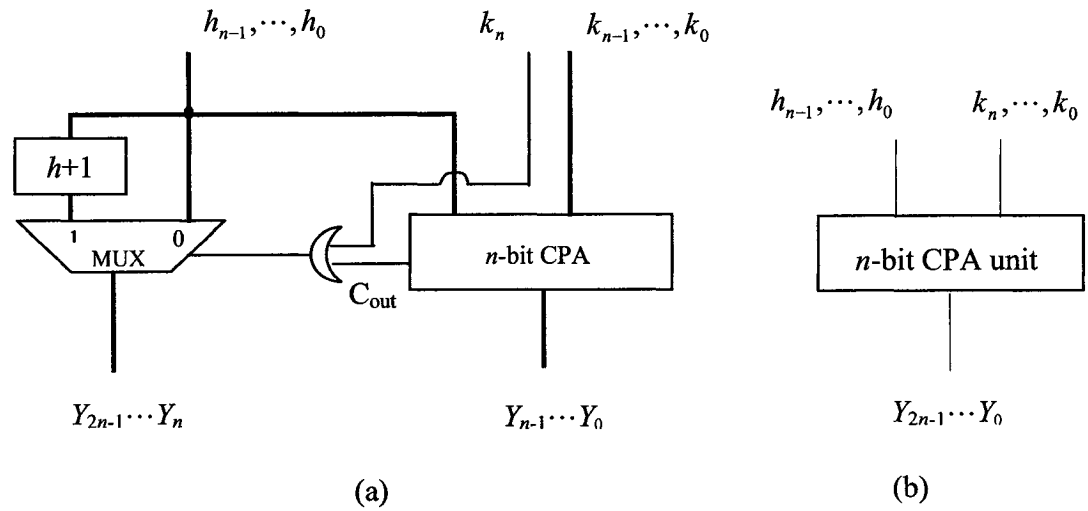
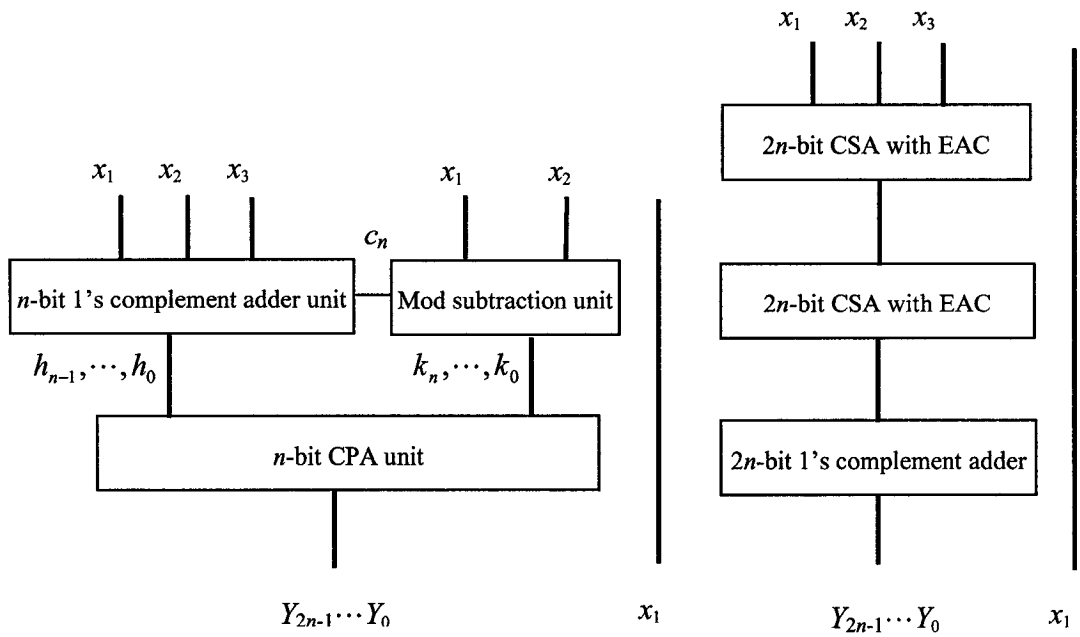


Fig. 7.5. The *n*-bit CPA unit

In Fig. 7.6 (a), we depict the entire architecture for the proposed converter. $Y_{2n-1}\cdots Y_0$ forms the $2n$ MSBs of X, while x_1 forms the n LSBs of X. The hardware required in the new converter shown in Fig. 7.6 (a) is as follows: $4n$ FAs, $9n$-2 MUXs, $7n$+1 inverters and $4n$-3 OR gates. The delay of the proposed converter t_{conv} is the sum of the delay of four MUXs t_{MUX}, the delay of a FA t_{FA}, the delay of two OR gates t_{OR}, the delay of n-bit CPA $t_{CPA(n)}$ and the delay of n-bit 1's complement adder $t_{1CA(n)} = 2t_{CPA(n)}$ [19], i.e., $t_{conv} = 4\,t_{MUX} + t_{FA} + 2\,t_{OR} + 3t_{CPA(n)}$.



(a) Proposed converter (b) Converter [39]

Fig. 7.6. The n-bit adder-based modulo reduced R/B converter

We tabulate the detailed comparison of the adder-based converters and the proposed design in Table 7.1, where the data for [36], [37] and [38] is from [39]. The converter proposed in [39] is the reconsideration of hardware realization of the converter from [38] whose original version has the critical delay path that includes three consecutive additions of $2n$-bit numbers. Both of the two converters are based on $2n$-bit CPA with EAC. It can be noted that one of the best nonredundant converters based on $2n$-bit adders is presented in [39]. For the convenience of comparing the performance, we show the main components used in the converter proposed in [39] as Fig. 7.6 (b). The delay in [39] is $2t_{FA} + t_{Inv} + 2t_{CPA(2n)}$.

In the literature, there exist R/B converter designs that avoid modulo-M operation by using a redundant representation of X [40]. In [40], the final (CPA) stage used to generate

the value X is implemented with a carry look-ahead (CLA) adder. To make a fair comparison with the converters in [39], the authors of [40] implement the $2n$-bit CPA with EAC using CLA. The design of [40] is slightly faster and requires less area than the design of [39] when n becomes large. As compensation, certain dynamic range of X is unusable.

Table 7.1. Performance comparison of adder-based converters

Converter	Hardware							Delay
	FA	AND /OR	XOR/ XNOR	CPA		CLA	Other	
				$2n$	n	n		
I. $2n$-bit adder-based converters								
[36] [37]	$6n$	–	n	2	–	–	–	$2t_{CPA(n)}+2t_{CPA(2n)}+2t_{XOR}$
[38]	$6n$	$4n$	$2n$	1	–	–	–	$3t_{CPA(2n)}+t_{XOR}+\lceil \log(2n) \rceil t_{AND}$
[39]	$4n$	$2n$	$2n$	1	–	–	$2n$ inverter	$2t_{FA}+t_{Inv}+2t_{CPA(2n)}$
[42]	$2n$	n	–	1	–	–	$2n$ inverter	$t_{FA}+t_{OR}+2t_{CPA(2n)}$
[44]	$2n$	2	–	1	–	–	$2n$ inverter	$t_{FA}+t_{OR}+t_{Inv}+2t_{CPA(2n)}$
[45]	$2n$	–	1	1	–	–	$2n$ inverter 2MUX, 1HA	$t_{FA}+t_{Inv}+t_{MUX}+2t_{CPA(2n)}$
Proposed	$3n$	$4n$	–	–	1	–	$9n$-2 MUX, $7n$+1 inverter	$4t_{MUX}+t_{FA}+2t_{OR}+3t_{CPA(n)}$
II. n-bit adder-based converters								
[40]	$2n$	–	–	–	–	4	$2n$ MUX, $2n$ inverter	$2t_{OR}+t_{Inv}+2t_{MUX}+t_{FA}+t_{CPA(n)}$
[43]	$6n$	n	n	–	–	4	$2n$ MUX,	$2t_{FA}+2t_{NAND}+t_{MUX}+t_{CPA(n)}$

7.5 Implementation of R/B Converter for N_1

The proposed converter and the existing converter in [39] are implemented using Xilinx FPGA technology for the 32-bit and 64-bit cases. The synthesis and implementation tools are Synopsys's Design Compiler Version 3.4b and Xillinx Alliance

M1.3 software. The target technology is a Xillinx 4010e-3 FPGA. The performance evaluation in terms of power, area and delay is carried out.

The FPGA implementation results of the proposed converter and that of [39] are compared in Table 7.2. The saving provided by the proposed converter over the designs in [39] is also shown in Table 7.2. The result of the FPGA implementation shows that, for the dynamic range of 64-bit, the proposed design reduces the delay around 20% than those in [39] while using a similar amount of hardware resources. The power consumption is also improved. The reason for this improvement is that the modulo operation of the converter in [39] is based on $2^{2n}-1$, while the proposed converter takes 2^n-1 and 2^n+1 as its modulo bases. Since the proposed R/B converter makes use of the proposed modulo reduction technique and is simplified by using the MUX-based binary incrementer, modulo decrementer and subtractor, high-speed implementation is obtained compared to the previous work of [39].

Table 7.2. Implementation results of the R/B converters

	32-bit				
Converters	Power (mw)	Cell Area	Time (ns)	AT	AT^2
C_1: Designs in [39]	18.9886	508	333.1	169164.8	56365449.9
C_2: Proposed design	18.8735	523	277.3	145027.9	40216236.7
$\frac{C_1-C_2}{C_1}\times 100\%$	1	- 3	16.8	11.7	28.7
	64-bit				
C_1: Designs in [39]	37.9501	1014	633.59	642460.26	407056396.1
C_2: Proposed design	36.5445	1040	502.71	522818.4	262826037.9
$\frac{C_1-C_2}{C_1}\times 100\%$	4	- 3	20.7	18.6	35.4

7.6 Summary

In this chapter, we have initially applied the proposed modulo reduction technique to derive modulo reduced R/B algorithm for the most popular three-moduli set $N_1 = \{2^n, 2^n +1, 2^n -1\}$. Then, we further simplify this R/B algorithm for efficient implementation of a new R/B converter. Novel design of a modulo $2^n +1$ subtractor needed for constructing the new R/B converter has been developed. Finally, the complete architecture of the new R/B converter has been presented along with implementation. Based on the FPGA implementation results, a comparison study between the proposed R/B converter and the corresponding ones in the literature has been carried out.

CHAPTER 8

CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

8.1 Concluding Remarks

In this thesis, several contributions have been presented. They are given below.

1. Novel modulo reduction algorithms have been proposed to significantly simplify a large modulo operation to the sum of a number of small modulo operations. By applying the proposed modulo reduction algorithms to the modified CRT, the complexity of modulo operation in the modified CRT is reduced to a great scale by partitioning the modulo operation with a large base to several individual modulo operations of small bases in parallel. Then, the modulo reduction technique and the modulo reduced modified CRT have been applied to derive R/B algorithms for the two existing three-moduli sets N_1, N_2 and four newly found three-moduli sets.
2. A novel R/B converter for $N_1 = \{2^n, 2^n+1, 2^n-1\}$ with reduced modulo and binary operations has been proposed. First, the proposed modulo reduction algorithm has been applied to reduce the required modulo operation from one big modulo $(2^n+1)(2^n-1)$ operation to two small modulo operations based on 2^n-1

and 2^n+1 respectively. Then, an algorithm has been proposed to simplify the binary part of the proposed modulo reduced R/B algorithm.

3. Novel MUX-based designs of some components needed for constructing the new converter have been developed. Both of the new unsigned and signed-2's complement incrementer/decrementer have been designed to implement the operation of $Z \pm 1$. New modulo incrementer and decrementer have been developed to implement the operations of $|Z+1|_{2^n-1}$ and $|Z-1|_{2^n-1}$.
4. Furthermore, a new modulo subtractor has been proposed to conduct the modulo (2^n+1) subtraction.
5. Finally, the complete architecture of the new converter has presented along with implementation. Based on the FPGA implementation results, a comparison study between the proposed R/B converter and the corresponding ones in the literature has been carried out.

8.2 Suggestions for Further Work

In Chapter 4, we have proposed modulo reduced R/B algorithms for six three-moudli sets. In [26], a comprehensive study of R/B converters for three-moduli sets has been carried out based on the modified CRT. It has been shown that the moduli set N_1 can offer the fastest R/B converter while requiring the least hardware resources for a given dynamic range. However, based on the proposed modulo reduction method, the R/B converter designs are different from those based on the modified CRT. Thus, the conclusion of [26] might not hold for these new designs. For example, it is seen in Table 4.1 that the modulo operation required by N_2 is the smallest compared to other three-

moduli sets. Hence, the N_2-based converter might be the fastest and is worth further study.

The proposed modulo reduction algorithm provides a novel way to design small, fast and low-power modulo circuits for R/B converters, B/R converters and RNS subsystems. Besides these, other modulo algorithms such as modulo division and modulo comparison might benefit too. What's more, the use of the proposed techniques in the design and VLSI implementation of different cryptographic schemes and protocols based on the residue arithmetic can be investigated. It is expected that the proposed modulo reduction algorithms have many applications in RNS study.

REFERENCES

1 N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, MA: Addison-Wesley Company, 1994.

2 M. M. Mano, *Digital Deign*, New York: Prentice Hall, 1991.

3 M. J. S. Smith, *Application-Specific Integrated Circuits*, MA: Addison-Wesley, 1997.

4 F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," IEEE Transactions on VLSI Systems, Vol. 2, No. 4, pp. 446-455, 1994.

5 M. Pedram, "Power Minimization in IC Design: Principles and Applications," ACM Transactions on Design Automation of Electronic Systems, Vol. 1, No. 1, pp. 3-56, 1996.

6 H. L. Garner, " The residue number system," *IRE Trans. Electronic Computers.* vol. 8, pp. 140-147, June 1959.

7 M. A. Soderstrand, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing.* New York: IEEE Press, 1986.

8 N. Szabo and R. Tanaka, *Residue Arithmetic and its Applications to Computer Technology.* New York: McGraw-Hill, 1967.

9 M. Bhardwaj and A. Blaram, "Low power signal processing architectures using residue arithmetic," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 3017-3120, 1998.

10 W.L. Freking and K.K. Parhi, "Low-power FIR digital filters using residue arithmetic", *Proc. IEEE International Symposium on Circuits and Systems*, pp. 739-743, 1998.

11 W.A. Chren, "One-hot residue coding for low delay-power product CMOS design," *IEEE Transactions on Circuits and Systems II*, vol. 45, pp. 303-313, Mar. 1998.

12 M.N. Mahesh and M. Mehendale, "Low power realization of residue number system based FIR filters," *Proc. 13th International conference on Circuits and Systems*, vol. 2, pp. 1090-1093, 2000.

13 F. Barsi and M. Pinotti, "A fully parallel algorithm for residue to binary conversion," *IPL*, pp. 1-8, 1994.

14 C. N. Zhang, B. Shirazi, and D. Yun, "An efficient algorithm and parallel implementations for binary and residue number systems," *Journal of Symbolic Computation*, vol. 15, pp. 451-462, 1993.

15 J. Kim, K. Park, and H. Lee, "Efficient residue-to-binary conversion technique with rounding error compensation," *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 315-317, Mar. 1991.

16 G. Alia and E. Martinelli, "VLSI binary-to-residue converters for pipelined processing," *Journal of Computing*, vol. 33, no. 5, pp. 473-475, 1990.

17 C. N. Zhang, B. Shirazi, and D. Yun, "Parallel designs for Chinese remainder conversion," in *Proceedings 16th Int. Conf. Parallel Processing*, Aug. 1989.

18 R. M. Capocelli and R. Gian Carlo, "Efficient VLSI networks for converting an integer from binary system to residue number system and vice versa," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 11, pp. 1425-1430, Nov. 1988.

19 G. Alia, F. Barsi, and E. Martinelli, "A fast VLSI conversion between binary and residue systems," *IPL*, vol.18, pp. 141-145, 1984.

20 B. Vinnakota and V. V. Rao, "Fast conversion techniques for binary-residue number systems," *IEEE Transactions on Circuits and Systems I*, vol. 41, no. 12, pp. 927-929, Dec. 1994.

21 J. Mathew, D. Radhakrishnan, and T. Srikanthan, "Fast residue-to-binary architectures," *42nd Midwest Symposium on Circuits and Systems*, vol. 2, pp. 1090-1093, 2000.

22 G. C. Cardarilli, M. Er and R. Lojacona, "RNS-to-binary conversion for efficient VLSI implementation," *IEEE Transactions on Circuits and Systems II*, vol. 45, no. 6, pp. 667-669, June 1998.

23 Y. Wang, M. N. S. Swamy and M. Omair Ahmad, "Three number moduli sets based residue number systems," *IEEE Transactions on Circuits and Systems II*, vol. 46, no. 2, pp. 180-183, Feb. 1999.

24 Y. Wang, "Residue-to-binary converters based on new Chinese remainder theorems," *IEEE Trans. Circuits and Systems-II*, pp.197-206, Mar. 2000.

25 W. K. Jenkins and B. J. Leon, "The use of residue number systems in the design of finite impulse response digital filter," *IEEE Trans. Circuits and Systems*, vol. CAS-24, pp. 191-201, Apri. 1977.

26 Wei Wang, M. N. S. Swamy, M. O. Ahmad and Yuke Wang, "A study of residue-to-binary converters for three-moduli sets," *IEEE Trans. Circuits and Systems-I*, vol. 50, No. 2, Feb. 2003.

27 A.A. Hiasat and H. S. Abdel-Aty-Zohdy, "Residue-to-binary arithmetic converter for the moduli set (2^k, 2^k -1, 2^{k-1} -1)," *IEEE Trans. Circuits and Systems*, vol. 45, pp. 204-208, Feb. 1998.

28 F.Pourbigharaz and H. M. Yassine, "A signed-digit architecture for residue to binary transformation," *IEEE Trans. Computers.*, vol. 46, no. 10, pp. 1146-1150, Oct. 1997.

29 G.C. Cardarilli, M. Er, and R. Lojacona, "RNS-to-binary conversion for efficient VLSI implementation," *IEEE Trans. Circuits and Systems-II*, vol. 45, pp. 667-669, June 1998.

30 S. Furber, *ARM System Architecture*. Reading, MA: Addison-Wesley, 1997.

31 D.R. Lutz and D.N.Jayasimha, "Programmable modulo-K counters," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 43, pp. 939-941, Nov.1996.

32 M.W. Evans, "Minimal logic synchronous up/down counter implementations for CMOS," *U.S Patent* no. 4,611,337, Sept. 1986.

33 N. West, and K. Eshraghian, *Principles of CMOS VLSI Design*, Reading, MA: Addison-Wesley, 1985.

34 C. Huang, J. Wang, and Y. Huang, "A high-speed CMOS incrementer/decrementer circuits and systems," *Proc. IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 88 -91, 2001.

35 Wei Wang, M. N. S. Swamy, M. O. Ahmad, and Yuke Wang, "A high-speed residue-to-binary converter for three-moduli (2^k, 2^k-1, 2^{k-1}-1) RNS and a scheme of its VLSI implementation," *IEEE Transactions on Circuits and Systems II*, vol. 47, pp. 1576-1581, Dec. 2000.

36 K. Ibrahim and S. Saloum, "An efficient residue to binary converter design," *IEEE Trans. Circuits and Systems*, vol. 35, pp. 1156-1158, Sept. 1988.

37 A. Dhurkadas, "An efficient residue to binary converter design," *IEEE Trans. Circuits and Systems*, vol. 37, pp. 849-850, June 1990.

38 S. Andraos and H. Ahmad, "A new efficient memoryless residue to binary converter," *IEEE Trans. Circuits and Systems*, vol. 35, pp. 1441-1444, Nov. 1988.

39 S. Piestrak, "A high-speed realization of a residue to binary number system converter," *IEEE Trans. Circuits and Systems-II*, vol. 42, Oct. 1995.

40 R. Conway and J. Nelson, "Fast converter for 3 moduli RNS using new property of CRT," *IEEE Trans. Computers*, vol. 48, pp. 852-860, Aug. 1999.

41 Y. Wang, X. Song, and M. Aboulhamid, "Near-optimal residue to binary converter for the moduli" in *Proc. 8th Great Lakes Symp. VLSI*, Feb. 1998.

42 Z. Wang, G. A. Jullien and W. C. Miller "An improved residue-to-binary converter," *IEEE Trans. Circuits and Systems-I*, vol. 47, pp. 1437-1440, Sept. 2000.

43 M. Bhardwaj, A. B. Premkumar, and T. Srikanthan, "Breaking the 2n-bit carry propagation barrier in residue to binary conversion for the $(2^n-1, 2^n, 2^n+1)$ module set," *IEEE Trans. Circuits and Systems-II*, vol. 45, pp. 998-1002, Sept. 1998.

44 A. Dhurkadas, "Comments on "A high-speed realization of a residue to binary number system converter"," *IEEE Trans. Circuits and Systems-II*, vol. 45, pp. 446-447, Mar. 1998.

45 Y. Wang, X. Song, and M. Aboulhamid and H. Shen, "Adder based residue to binary number converter for $(2^n-1, 2^n, 2^n+1)$," *IEEE Trans. Signal Processing*, vol. 50, pp. 1772-1779, July. 2002.

46 M. Dugdale, "VLSI implementation of residue adders based on binary adder," *IEEE Trans. Circuits and Systems-II*, vol. 39, May 1992.

47 B. Parhami, "RNS representations with redundant residue," *35th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1651-1655, Nov. 2001.

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Shaoqiang Bi

Entitled: Low Power Modulo Reduction Technique and Its Application in Residue-to-binary Converters

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical Engineering)

Complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

_____ Examiner

_____ Examiner

_____ Co-supervisor

_____ Supervisor

Approved by _____
Chair of Department or Graduate Program Director

_____ 2004 _____
Dean of Faculty