Digit Set Conversion from Redundant Number System into Complement Representation

Veerasit Charoensiri and **Athasit Surarerks**, Non-members

ABSTRACT

Redundant number system was proposed in order to solve the carry-propagation problem. Although it provides a carry-free parallel addition, this representation requires a lot of space to store itself. Many conversions from the redundant number system into another number representation have been introduced to decrease the storage usage. This paper proposes a generic algorithm in order to convert the redundant number representation into the complement number representation. The proposed algorithm can perform the conversion of a number in any integer radix and eliminates the carry chain of the traditional method. The proofs of the proposed algorithm in term of correctness are also included in this paper.

Keywords: Redundant Number System, Non-Redundant Number System, Twos complement Number, Digit Set Conversion

1. INTRODUCTION

In the computer system, arithmetic operations play a major role in many applications where speed is an essential issue. The speed of operations may depend on the implementation of arithmetic algorithms. The conventional operations, like addition, subtraction and multiplication, can produce the carry-propagation chains. In the late 1950, Avizienis [1] introduced the redundant number system to solve this problem. The important properties of this number system are to have more than one representation for its value and to represent negative number easily. In the other hand, the disadvantage is a problem of space usage for store itself. The conversion algorithms from one radix representation into another have been proposed as the solution to solve this problem such as the bit-serial conversion style [2], the digitparallel conversion style [3] or the conversion using finite automata [4]. Many researchers use redundant binary number system with the digit set $\{-1, 0, 1\}$ for computer computation because it can handle negative number without using twos complement method. However, the conventional number representation is more common used in computer computation. According to this reason, many conversion algorithms have been presented in order to convert the redundant binary number into another such as standard binary or twos complement number representation. The conversion of redundant number into twos complement can be done by using the classical method which first converts the digit set to decimal value by simple radix polynomial expression:

$$P = \sum_{i=0}^{n} d_i r^i,$$

where r is any positive integer and di is a digit in a redundant digit set.

Then convert the decimal value to standard binary by a repeated division algorithm. Later convert into twos complement representation in the case where the decimal value is negative. A novel method, for converting from the decimal value into binary number, was purposed by Kettani [5].

Moreover, the sequential conversion algorithms were proposed [6, 7] by using conversion table rules to convert the redundant binary number into twos complement number representation. In other words, those algorithms are limited to process only one digit at a time. Likewise, the algorithm introduced by Choo and Deshmukh [8], Wang and Tull [9] have the same processing style.

This paper presents the generic algorithm for converting the redundant number system into complement representation in any integer radix. The conversion can be divided into two parts. The first part is the carry generating part which generates the appropriated carry comparing to the input. The second part is the expected result calculation using the input and its carry. We have also shown that our algorithm is correct via mathematical proof. Finally, we have illustrated how our algorithm works through an example.

2. PRELIMINARIES

This section revisits some related works of this paper. We start by twos complement representation, signed digit representation, characteristics of the redundant number system, and characteristics of the digit set.

Manuscript received on December 15, 2006 ; revised on March 15, 2007.

The authors are with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University Pathumwan, Bangkok, 10330, Thailand; E-mail: G46vch@cp.eng.chula.ac.th, Athasit@cp.eng.chula.ac.th

2.1 Twos Complement Number Representation

Twos complement number representation was introduced to represent the negative number without using negative digit. A number system *(r, D)* is composed of a radix r such that $r \geq 2$ and a finite digit set D. Let X be a number representation for radix r on the digit set D. The numerical value of X denoted by $\|X\|$ is

$$\|X\| = x_{n+1}r^{n+1} + \sum_{i=0}^{n} x_i r^i,$$

where $X=(x_{n+1}x_n x_{n-1}...x_2 x_1 x_0)$.The term x_{n+1}in the representation of a number indicates the numeric sign, while the rest indicate the magnitude. For example, a number $(-6)_{10}$ can be represent as $(11111010)_2$

2.2 Redundant Number System

Signed digit representations and the redundant number system were first introduced by Avizienis [1] for eliminating carry-propagation chain. These number systems have been defined for any integer radix r>2 with a symmetric digit set $D = \{-a, ..., a\}$where $r/2 \leq a \leq r$-1. Moreover, the redundant number system with the number of digits $|D| = r$+1 is known as minimally redundant and with $|D| = 2r$-1 is known as maximally redundant.

The digit set is not restricted with the symmetric style but it can also be the asymmetric as well [10]. The form of this digit set is $\{-b, ..., a\}$ where a $\neq$ b and the number of all possible digit values are $a+b+1$.

The important property of the redundant number system is that one number has more than one representation. Another property is the ability to handle the negative number. For example, given radix $r = 6$ with a digit set $\{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$, 52_{10}can be represented by

$$(1\ 3\text{-}2)_6 = 52_{10}$$
$$(1\ 2\ 4)_6 = 52_{10}$$
$$(2\text{-}4\ 4)_6 = 52_{10}.$$

In such example, one value may have several representations.

2.3 Digit-Set Conversion

Due to the problem of storage usage, the redundant number system with a digit set$\{-a, ..., a\}$ can be converted into another digit set. The conversion between those digit sets is an essential method of computer computation. Note that an addition can be considered as a digit set conversion from one to another. Ercegovac and Lang [2] presented an algorithm for such conversion but their algorithm was a purely sequential process. To improve the conversion algorithm, Kornerup [3] developed an algorithm which is to convert $P \in P[r, D]$ where $P = \sum d_i r^i$ and $d_i \in D$ into $Q \in P[r, E]$ that is $\|P\| = \|Q\|$.The digit set P and Q are the redundant and nonredundant digit set respectively where the digit set D and E are different.

The conversion mapping α between the digit set D and E with carry in C is

$$\alpha : C \times D \longrightarrow C \times E,$$

that can be derived to equation as

$$\acute{c} + d = c(r) + e.$$

From this conversion mapping, the function of carry is defined as a carry-transfer function;

$$\forall c \in C : \gamma_d(c) = \acute{c} \text{ where } \alpha(c, d) = (\acute{c}, e).$$

The function γ_d describes the mapping of an incoming carry value *(c)* into a digit d which produces outgoing carry value ($\acute{c}$).

The other set of function$\{\epsilon_d\}_d \in D, \epsilon_d : C \rightarrow E$,the digitmapping function, is defined as follow:

$$\forall c \in C : \epsilon_d(c) = e \text{ where } \alpha(c, d) = (\acute{c}, e).$$

The function ϵ_d describes the conversion of a digit value $_d$ into digit $_e$ when the incoming value is $_c$. Then, the function can be rewritten as

$$e_i = \epsilon d_i(c_{i-1}).$$

For example, given a number $X= 3\bar{2}112\bar{2}11$where digits are in digit set $D = \{-3, ..., 3\}$for radix $r = 3$, the conversion mappingα of $X$from digit set D into digit set $E= \{-1, 0, 1\}$with the carry set $C = \{-1, 0, 1\}$ can be shown below:
From the carry transfer function, we get

c_0	$\gamma_0(0)$	=	0
c_1	$\gamma_1\gamma_0(0)$	=	0
c_2	$\gamma_2\gamma_1\gamma_0(0)$	=	-1
.			
.			
.			
c_7	$\gamma_7\gamma_6\gamma_5\gamma_4\gamma_3\gamma_2\gamma_1\gamma_0(0)$	=	1

Thus, we get carry c$_i$ = $1\bar{1}000\bar{1}00$.

Then we can obtain e$_i$ from the digit mapping function.

e_0	$\epsilon_d 0$	=	1
e_1	$\epsilon_d 1$	=	1
e_2	$\epsilon_d 2$	=	1
.			
.			
.			
e_8	$\epsilon_d 8$	=	1

Thus, the expected output is e$_i$ = $(1\bar{1}1111111)_3$

2.4 The Conversion of Redundant Number

Since the redundant number system requires more space, several solutions have been proposed to solve the problem. One of them uses the redundant binary number system instead because it requires less space. However, it is still not widely used for computer unlike twos complement representation. Hence, the methods of the conversion from the redundant binary number system into twos complement representation were introduced [6-8]. More Recently, Wong and Tull [9] proposed an improved method for the conversion by using Encoded Conversion Truth Table for RB-NB where RB and NB stand for a redundant binary number and a normal binary number respectively.

Table 1: *Encoded Conversion Truth Table for RB-NB*

Input				Output	
Redundancy bit			Carry in	Binary	Carry Out
x_j	s_j	d_i	c_j	b_j	c_{j+1}
0	0	1	1	1	1
0	0	1	0	0	0
1	1	1	1	0	0
1	1	1	0	1	0
–1	0	0	1	0	1
1	0	0	0	1	1

For example, given a redundant binary number X= $(\bar{1}00100\bar{1}00\bar{1}01)_2$, the output bi produced using Table 1 is (1100011011101) .

3. EQUATIONS

In this section, we present an algorithm which can convert the redundant number with a symmetric digit set into the complement representation in the same radix. The algorithm is divided into two parts. The first part is an adaptation of generating carry c_i called a *variable carry* function. The second part is the mapping of an input with a carry to the expected output.

3.1 Variable carry function

To convert the redundant number, we have to consider the carry and its restriction for the conversion. We use the variable carry c_{i-1} to indicate the incoming carry and c_i to indicate the outgoing carry. We now define a set of carry function F_{xk} describing the outgoing carry c_i when incoming carry c_{i-1} passing through an input x_i as:

$$c_i = F(x_i, F(x_{i-1}, F(x_{i-2}, ...F(x_j, c_{j-1})))).$$

Let $X = (x_n x_{n-1} x_{n2} ... x_2 x_1 x_0)$ be a number represented in radix $r \geq 2$ where x_i is in digit set D = $\{-a, ..., a\}$ and $r/2 \leq a \leq r-1$. From the carry function F_{xk} , the outgoing carry c_i can be defined by

$$c_i = \lceil (c_{i-1} - rx_i)/(r^2 + c_{i-1}) \rceil \times (r-1) \qquad (1)$$

where the initial carry $c_1 = 0$.

Lemma 3.1
Let$X$ = $(x_n x_{n-1} x_{n-2} ... x_2 x_1 x_0)$ *be a number represented in radix* $r \geq 2, x_i \in \{-a, ..., a\}$*where* $r/2 \leq a \leq r-1$.*Given a carry* $c_1 = 0$, *the variable carry function ci defined as*

$$c_i = \lceil (c_{i-1} - rx_i)/(r^2 + c_{i-1}) \rceil \times (r-1),$$

must be either 0 *or* (r-1).

Proof
From the variable carry function,

$$c_i = \lceil (c_{i-1} - rx_i)/(r^2 + c_{i-1}) \rceil \times (r-1),$$

where $x_i \in \{0, \pm1, ..., \pm(r-1)\}$ and i is a nonnegative integer.
We have that

$$x_i < \mathrm{r}$$

Then multiply by r

$$rx_i < r^2$$

Since $x_i \in \{0, \pm1, ..., \pm(r-1)\}$,the term -($rx_i$) or rx_i is always less than r^2.Then add c_{i-1}, we get

$$c_{i-1} + (-rx_i) < c_{i1} + r^2.$$

From above, we have that

$$-1 < (c_{i1} - rx_i)/(c_{i-1} + r^2) < 1,$$

where $x_i \in Z^+ \cup \{0\}$. Then, the term

$\lceil (c_{i-1} - rx_i/r^2 + c_{i-1} \rceil$is in $\{0, 1\}$. It is clear that

$\lceil (c_{i-1} - rx_i/r^2 + c_{i-1} \rceil \times (r-1)$is in $\{0, r-1\}$

3.2 Digit mapping function

Let $Y = (y_{n+1} y_n y_{n-1} ... y_2 y_1 y_0)$ be a number represented in radix r≥2. Let Y be the result of the conversion of X. Then, we present the other function H_{xk} describing the output y_i when the incoming carry value is c_{i-1} and the input is x_i as follow:

$$y_i = H(x_i, H(x_{i1}, H(x_{i2}, ..H(x_j, c_{j1})))).$$

Thus, we can derive H_{xk} into an equation below

$$y_i = (x_i + c_{i-1}) - r \times \lfloor (x_i + c_{i-1}/r \rfloor) \qquad (2)$$

The equation (2) is used to compute the output yi from the least significant digit to n^{th} position. For $n+1^{th}$ position, consider the most significant digit of X or x_n, if it is negative, the digit $y_{n+1} = 1$ indicated the signed digit otherwise the digit $y_n + 1 = 0$.

Before we show that the input and the output have the same numerical value, we now have to show that the output is represented with digit in the digit set E

$= \{0, ..., r1\}$.

Lemma 3.2

Let $X = (x_n x_{n-1} x_{n-2} ... x_2 x_1 x_0)$ *be a number represented in radix* $r \geq 2$ *with a symmetric digit set* $D = \{-a, ..., a\}$ *where* $r/2 \leq a \leq r-1$. *Let* $c_{-1} = 0$ *and* c_i *follows the variable carry function. For every outgoing digit* y_i *defined as*

$$y_i = (x_i + c_{i-1}) - (r \times \lfloor (x_i + c_{i-1}/r \rfloor D$$

must be in the digit set $E = \{0, ..., r-1\}$.

Proof

From the digit mapping function y_i, we have to show that the result y_i is always in $\{0, ..., r-1\}$ and the carry c_i must be 0 or r-1. The proof consists of two cases. The first case is the carry $c_{i-1} = 0$. The latter case is the carry $c_{i-1} = r$-1

1. In the case where $c_{i1} = 0$, we have that
   $y_i = x_i - (r \times \lfloor x_i / r \rfloor)$

   1)If 0$\leq x_i \leq$ r-1, then $y_i = x_i$. Since $\{0, ..., r-1\}$, we have that $y_i \in \{0, ..., r-1\}$.

   2)If -(r-1)$\leq x_i \leq -1$, then $y_i = x_i$+r. Since $x_i \in \{-(r-1), ..., -1\}$, we also have that $y_i \in \{1, ..., r-1\}$.

   In two cases, it is true that y_i is in $\{0, , r-1\}$.
2. In the case where c_{i-1} = r-1, we have that
   $y_i = (x_i + c_{i-1}) - (r \times \lfloor (x_i + c_{i-1} \rfloor)$

   1)If x_i=0,then y_i = r-1. Thus, we have that y_i =r-1.

   2)If 1$\leq x_i \leq r-1, then y_i = x_i - 1$. Since $x_i \in \{1, ..., r-1\}$, we have that y_i is in $\{0, ..., r-2\}$.

   3)If-(r-1)$\leq x_i \leq -1$, then $y_i = x_i$+(r-1). Since $x_i \in \{-(r-1), ..., -1\}$,we also have that y_i is in $\{0, , r-2\}$.

In all cases, yi is always in $\{0, ..., r-1\}$.

Since we have proved the output y_i is in digit set $E = \{0, , r-1\}$, we now show that the numerical value of input X must equal to the output Y or $\|X\| = \|Y\|$ by the following theorem.

Theorem 3.1

Let X be a number representation for integer radix r=$\geq$*2 ,* $x_i \in \{a, ..., a\}$ *where* $r/2 \leq a \leq r-1$. *Let Y be a number representation for integer radix r*$\geq$*2 where* y_i *is in digit set* $E = \{0, 1, ..., r-1\}$. *The conversion from X to Y can be done by variable carry* c_i *and digit mapping function* y_i *and*$\|X\| = \|Y\|$.

Proof

Before proving the theorem, we consider (1) and (2), we can conclude that

1) In the case where incoming carry $c_{i-1} = 0$ pass through x_i where $x_i \in Z^+ \cup \{0\}$, we get $x_i = y_i$ and $c_i = 0$.

2) In the case where incoming carry $c_{i_1} = 0$ pass through x_i where $x_i \in Z^-$, we get $x_i = y_i - r$ and c_i = r-1.

3) In the case where incoming carry $c_{i_1} = r$-1 pass through x_i where $x_i \in Z^- \cup \{0\}$, we get $x_i = y_i - r + 1$ and c_i = r-1.

4) In the case where incoming carry $c_{i_1} = r$-1 pass through x_i where $x_i \in Z^-$, we get $x_i = y_i + 1$ and c_i = 0.

We now prove the theorem $\|X\| = \|Y\|$ by using mathematical induction. This is to show

$$\sum_{i=0}^{n} x_i r^i = -S_{n+1} r^{n+1} + \sum_{i=0}^{n} y_i r^i \qquad (3)$$

where the initial carry $c_1 = 0$ and

$$S_{n+1} = \begin{cases} 0 & if\ c_n = 0 \\ 1 & if\ c_n = r-1 \end{cases}$$

Let *P(n)* be the proposition that this formula (3) is correct for the integer n $\geq$ 0.

BASIS STEP: We show that *P(0)* is true.

From (3) with n = 0, we get

$$\begin{aligned} \sum_{i=0}^{0} x_0 r^0 &= -S_{0+1} r^{0+1} + \sum_{i=0}^{0} y_0 r^0 \\ x_0 r^0 &= -S_1 r^1 + y_0 r^0 \\ x_0 r^0 &= -S_1 r^1 + y_0 \end{aligned}$$

Case 1: Since $x_0 \in Z^+ \cup \{0\}$ and incoming carry $c_{-1} = 0$, we have $x_0 = y_0$ and $c_0 = 0$.

Then we get

$$x_0 = -S_1 r^1 + x_0$$

Since $c_0 = 0$, $S_1 = 0$.

$$x_0 = x_0$$

Case 2: Since $x_0 \in Z^-$and incoming carry $c_{-1} = 0$, we have $x_0 = y_0 - r$ and $c_0 = r - 1$.

Then we get

$$x_0 = -S_1 r^1 + x_0 + r$$

Since $c_0 = 0$, $S_1 = 0$.

$$\begin{aligned} x_0 &= -r + x_0 + r \\ &= x_0 \end{aligned}$$

In two cases, We have that *P(0)* is true.

*INDUCTIVE STEP:*Assume that *P(k)* is true. That is, assume $n = k$ and $k \geq 0$. We have to show that *P(k+1)* is also true.

$$\sum_{i=0}^{k+1} x_i r^i = x_{k+1} r^{k+1} + \sum_{i=0}^{k} x_i r^i \quad (4)$$

Then substitute (3) in (4), we get

$$\sum_{i=0}^{k+1} x_i r^i = x_{k+1} r^{k+1} - S_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i \quad (5)$$

To carry on the inductive step using this assumption, the proof composes of two cases. The first case is $S_k + 1 = 0$ and the other is $S_{k+1} = 1$.

Case 1: With $S_{k+1} = 0$ and the incoming carry $c_k = 0$, we can divide this case into two sub cases. The first case is $x_{k+1} \in Z^+ \cup \{0\}$and the other is $x_{k+1} \in Z^-$.

Case 1.1: With $x_{k+1} \in Z^+ \cup \{0\}$,we have$x_{k+1} = y_{k+1}$and c_{k+1}= 0. Since$c_{k+1} = 0$we get S_{k+2}=0.

From (5), we get

$$\sum_{i=0}^{k+1} x_i r^i = x_{k+1} r^{k+1} - S_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i$$

Since $S_{k+1} = 0$, we have

$$\begin{aligned} \sum_{i=0}^{k+1} x_i r^i &= x_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= y_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= \sum_{i=0}^{k} y_i r^i \end{aligned}$$

Thus, we have that $\sum_{i=0}^{k+1} x_i r^i - S_{k+2} r^{k+2} + \sum_{i=0}^{k+1} y_i r^i$

Case 1.2: With $x_{k+1} \in Z^-$,we have$x_{k+1} = y_{k+1} - r$and c_{k+1}= r-1. Since$c_{k+1} = r - 1$we get S_{k+2}=1.

From (5), we get

$$\sum_{i=0}^{k+1} x_i r^i = x_{k+1} r^{k+1} - S_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i$$

Since $S_{k+1} = 0$, we have

$$\begin{aligned} \sum_{i=0}^{k+1} x_i r^i &= x_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= (y_{k+1} - r) r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= -r^{k+2} + \sum_{i=0}^{k} y_i r^i \end{aligned}$$

Thus, we have that $\sum_{i=0}^{k+1} x_i r^i - S_{k+2} r^{k+2} + \sum_{i=0}^{k+1} y_i r^i$

Case 2: With $S_{k+1} = 0$ and the incoming carry c_k=r-1, we can divide this case into two sub cases. The first case is $x_{k+1} \in Z^+$and the other is $x_{k+1} \in Z^- \cup \{0\}$.

Case 2.1: With $x_{k+1} \in Z^+$,we have$x_{k+1} = y_{k+1}$and c_{k+1}= 0. Since$c_{k+1} = 0$we get S_{k+2}=0.

From (5), we get

$$\sum_{i=0}^{k+1} x_i r^i = x_{k+1} r^{k+1} - S_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i$$

Since $S_{k+1} = 1$, we have

$$\begin{aligned} \sum_{i=0}^{k+1} x_i r^i &= x_{k+1} - r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= y_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= \sum_{i=0}^{k} y_i r^i \end{aligned}$$

Thus, we have that $\sum_{i=0}^{k+1} x_i r^i - S_{k+2} r^{k+2} + \sum_{i=0}^{k+1} y_i r^i$

Case 2.2: With $x_{k+1} \in Z^- \cup \{0\}$,we have$x_{k+1} = y_{k+1} - r + 1$and c_{k+1}= r-1. Since$c_{k+1} = r - 1$we get S_{k+2}=1.

From (5), we get

$$\sum_{i=0}^{k+1} x_i r^i = x_{k+1} r^{k+1} - S_{k+1} r^{k+1} + \sum_{i=0}^{k} y_i r^i$$

Since $S_{k+1} = 1$, we have

$$\begin{aligned} \sum_{i=0}^{k+1} x_i r^i &= x_{k+1} - r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= (x_{k+1} - 1) r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= (x_{k+1} - 1 + r - r) r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= -r^{k+2} + y_{k-1} r^{k+1} + \sum_{i=0}^{k} y_i r^i \\ &= -r^{k+2} + \sum_{i=0}^{k} y_i r^i \end{aligned}$$

Thus, we have that $\sum_{i=0}^{k+1} x_i r^i - S_{k+2} r^{k+2} + \sum_{i=0}^{k+1} y_i r^i$

In all cases, it is true that $\|X\| = \|Y\|$.

4. EXAMPLE

Given $X = (21\bar{4}23043\bar{3})_5$ where radix $r = 5$ where x_i is in digit set $D = \{0, \pm 1, ..., \pm 4\}$. The conversion, from X to $Y$where y_i is in digit set E = $\{0, 1, ..., r - 1\}$, can be shown below.

Step 1: generate the carry c_i by the variable carry function with $c_{-1} = 0$. Given the variable carry function $c_i = \lceil (c_{i-1} - r_{xi})/r^2 + c_{i-1}) \rceil \times$(r-1). Then

$$\begin{aligned} c_0 &= \lceil (0 - 5(\bar{3}))/(4^2 + 0) \rceil \times (4) \\ &= 4 \\ c_1 &= \lceil (0 - 5(4))/(4^2 + 4) \rceil \times (4) \end{aligned}$$

$$\begin{aligned}
&= 0\\
c_2 &= \lceil (0-5(0))/(4^2+0) \rceil \times (4)\\
&= 0\\
&\quad .\\
&\quad .\\
&\quad .\\
c_7 &= \lceil (0-5(2)/(4^2+0) \rceil \times (4)\\
&= 0
\end{aligned}$$

Thus, we get carries 0,0,0,4,0,0,0,0 and 4.

Step 2: get the output y_i by the digit mapping function by computing c_{i1} and x_i. Given the digit mapping function $y_i = (x_i + c_{i-1}) - (r \times \lfloor (x_i + c_{i-1})/r \rfloor) with c_1 = 0$. Then

$$\begin{aligned}
y_0 &= (-3+0) - (5 \times \lfloor (-3+0)/5 \rfloor)\\
&= 2\\
y_1 &= (4+4) - (5 \times \lfloor (4+4)/5 \rfloor)\\
&= 3\\
y_2 &= (0+0) - (5 \times \lfloor (0+0)/5 \rfloor)\\
&= 0\\
&\quad .\\
&\quad .\\
&\quad .\\
y_7 &= (2+0) - (5 \times \lfloor (2+0)/5 \rfloor)\\
&= 2
\end{aligned}$$

Thus, the output Y is $(20123032)_5$.

Step 3: Since $c_7 = 0$, we obtain that $S_8 = 0$. Therefore the final result is $(020123032)_5$.

5. CONCLUSION

In this paper, we propose an algorithm for converting the redundant number system with a symmetric digit set into the complement representation in the same radix. The algorithm consists of two parts. The first part is the carry generating part. The other part is the digit mapping part, which the output is obtained by combining the input with the computed carry. The proposed method is a generic algorithm that can convert the redundant number in any radix to the complement representation. Using the mathematical induction, it shows that the proposed algorithm is correct

References

[1] A. Avizienis, Signed-Digit Number Representations for Fast Parallel Arithmetic, *IRE Transaction on Electronic Computers*, Vol. 10, pp.389-400, 1961.

[2] M. Ercegovac and T. Lang, On-the-Fly Conversion of Redundant into Conventional Representation, *IEEE Transactions on Computers*, Vol. C-36, No.7, pp.895-897, 1987.

[3] P. Kornerup, Digit-Set Conversion: Generalization and Application, *IEEE Transactions on Computers*, Vol. 43, pp.622-629, 1994.

[4] A. Surarerks, Digit Set Conversion by On-Line Finite Automata, *Bull. Belg. Math. Society*, Vol. 8, pp.337-358, 2001.

[5] H. Kettani, On the Conversion Between Number System, *Proceedings of the 2004 Internatrional Conference on Algorithmic Mathematics and Computer Science (AMCS04)*, pp.317-320, 2004.

[6] S.Yen, C. Laih, C. Chen, and J. Lee, An Efficient Redundant-Binary Number to Binary Number Converter, *IEEE Journal of Solid-State Circuits*, Vol. 27, pp.109-112, 1992.

[7] A. Herrfeld and S. Hentschke, Conversion of Redundant Binary into Two's Complement Representations, *IEEE Electronics Letters*, Vol. 31, pp.1132-1133, 1995.

[8] I. Choo and R.G. Deshmukh, A Novel Conversion Scheme from a Redundant Binary Number to 2s Complement Binary Number for Parallel Architectures, *Proceedings of the IEEE SouthernCon*, pp.196-201, 2001.

[9] Number to twos complement Number Converter, *Region 5 Conference: Annual Technical and Leadership Workshop, pp.141-143, 2004.*

[10] B. Parhami, Computer Arithmetic Algorithm and Hardware Design, Oxford University Press, New York, ch. 3, 1999.

Photograph is not available at time of printing

Veerasit Charoensiri

Photograph is not available at time of printing

Athasit Surarerks