ENGINEERING JOURNAL

Article

On-line Digit Set Conversion for Rational Digit Number

Pipop Thienprapasith[a] **and Athasit Surarerks**[b,*]

ELITE Laboratory, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
E-mail: [a]pipopt@gmail.com, [b]athasit.s@chula.ac.th (Corresponding author)

Abstract. A number system that is well-designed can affect the computational time and the hardware implementation. An interesting number system called Round-to-Nearest coding (RN-coding) was proposed to reduce a time consuming in a rounding process. Rounding to the nearest in RN-coding can be done using only truncation at any positions in a sequence of digits (representation). This concept can save a lot of time in a parallel or pipeline computation manner. However, an RN-coding does not support an on-line arithmetic computation. In this paper, we propose a rational digit number system which is composed of rational signed-digits in the digit set. This new system preserves a round-to-nearest property and is suitable for an on-line arithmetic computation. Performing on-line elementary arithmetic operations in our system can be done by an on-line digit set conversion algorithm. We show that our new algorithm, which is an improvement of an on-line addition algorithm in our previous work, can be demonstrated by an on-line finite automaton with a finite on-line delay k.

Keywords: RN-coding, on-line computation, signed-digit, rational digit set.

ENGINEERING JOURNAL Volume 21 Issue 7
Received 20 February 2017
Accepted 31 May 2017
Published 29 December 2017
Online at http://www.engj.org/
DOI:10.4186/ej.2017.21.7.251

DOI:10.4186/ej.2017.21.7.251

1. Introduction

The characteristics of a number system mostly affect the performance of computation in a computer system. The *Signed-digit* number system proposed by Avizienis [1] is a well-known example. The system consists of positive and negative integer digits in the digit set that has an important characteristic called redundant property. It means that a redundant number in this system can have more than one representation. Since this property has an ability to limit carry propagation during a computation, a parallel computational manner can be performed in a constant time. However, a number representation in a computer system is finite and cannot represent all numbers. It needs an additional process to round an overflow number that needs more computational time.

An idea on number system called interval number system [2-4] was proposed to handle an inexact number. An interval consists of two numbers that are a lower bound and an upper bound. The idea of this system is to represent an inexact number within an interval. During a computational process, an interval representation does not need a rounding process often comparing with a conventional number representation. An improvement made on interval number system [5, 6] is a new system designed to reduce the number of digits in the digit set. The new system can save a space used to represent an interval. However, a computational time in an interval number system is slower than a conventional number system due to the comparison between a lower bound and an upper bound of an interval.

There is an interesting number system called *RN-coding* [7]. This system was proposed to reduce unnecessary time consumption in a rounding process. A rounding process in this system can be done by truncating at any position in a sequence of digits (representation). The obtained number representation always has the same numerical value as a rounding to the nearest. However, an RN-coding does not support an on-line computational manner that is suitable for an inexact number in the case where number is very large. An on-line computational manner [8, 9] was first introduced by Ercegovac and Trivedi. The idea of this computation is to perform addition, subtraction, multiplication and division in the same direction started from the most significant digit.

In this work, we propose a rational digit number system that supports an on-line computational manner. It also preserves a round-to-nearest property similar to an RN-coding. This number system consists of rational signed-digits in the digit set. Since an elementary arithmetic operation can be expressed by a digit set conversion, we then propose an on-line digit set conversion algorithm that is demonstrated by an on-line finite automaton with an on-line delay k in our system. This new algorithm is an improvement of an on-line addition algorithm in our previous work [10].

This paper is organized as follows: Section 2 recalls some background and notations used in this work. The rational digit number is proposed in Section 3. Section 4 shows a digit set conversion by an on-line finite automaton in our system. The conclusion is discussed in Section 5.

2. Background and Related Works

In this section, we start by reviewing some preliminaries on number systems including redundancy property which is required in an on-line computation. We then recall some definitions of a digit set conversion and an on-line finite automaton that are used in this work. Finally, we talk about a round-to-nearest number system.

2.1. Number Systems

A conventional number representation consists of a contiguous digit set ($\mathcal{D}$) and a positive integer base (β). A number is represented by a finite or infinite sequence of digits. In this system, each positive number can have only one representation. We start by introducing the definition of the conventional number representation in definition 1.

Definition 1 (Conventional number representation)
A conventional number representation consists of a base β which can be an integer number such that $\beta \geq 2$ and a finite digit set $\mathcal{D} = \{0, 1, \dots, \beta - 1\}$. A number representation X on $\mathcal{D}$ is a sequence of digits as $X = (x_n x_{n-1} \dots x_0 . x_{-1} x_{-2} \dots)_\beta$ where $x_i \in \mathcal{D}$ and $i \leq n$. A numerical value of X in a base β is denoted by $\|X\| = \sum_{i=-\infty}^{n} x_i \beta^i$.

Since a conventional number representation has only one representation in each number, it limits an ability to perform a parallel or pipelining arithmetic computation. An interesting number system called redundant number system was proposed handle those problems. Redundant number system has a characteristic that some numbers in this system can have more than one number representation. It can limit carry propagation during a computational process. To construct a redundant number system is to add more digits in the digit set. There are many number systems that have a redundant property such as a signed-digit number system proposed by Avizienis [1] in 1961. A generalized signed-digit number representation was proposed by Parhami [11] in definition 2.

Definition 2 (Generalized signed-digit number representation)
A generalized signed-digit number representation consists of a base $\beta \geq 2$ where β is an integer number and a finite digit set $\mathcal{D} = \{-a, -a+1, \ldots, 0, \ldots, b-1, b\}$ with $a$and b as integers that $a, b \geq 0$ and $a + b + 1 > \beta$. A numerical value of X can be defined as $\|X\| = \sum_{i=-\infty}^{n} x_i \beta^i$ where $x_i \in \mathcal{D}$ and $i \leq n$.

In a signed-digit number representation, we usually denotes a signed-digit $-\alpha$ by $\overline{a}$. We show an example of the number system that has a redundant property in example 1.

Example 1 A representation of $X = (312)_{10}$ in a base $\beta = 3$ with a digit set $\mathcal{D} = \{\overline{1}, 0, 1, 2\}$.
Since the number of digits in the digit set $\mathcal{D}$ is more than the base β, it means that this system has a redundant property by Definition 2. Thus, the number $X = (312)_{10}$ can have more than one representation in a base $\beta = 3$ such as $(11\overline{1}2\overline{1}0)_3$ and $(110\overline{1}\overline{1}0)_3$. □

2.2. Digit Set Conversion

Digit set conversion [12, 13] can be used to describe arithmetic operations. Digit set conversion is a conversion from a digit set E to a digit set $\mathcal{D}$ with the same base β where $E \neq \mathcal{D}$. It means that given any number representation in a digit set E, find the number representation in a digit set $\mathcal{D}$ that has the same numerical value with the same base β. A conversion function λ can be defined as $\lambda: E^{\mathbb{N}} \rightarrow \mathcal{D}^{\mathbb{N}}$ where $E^{\mathbb{N}}$ and $\mathcal{D}^{\mathcal{N}}$ are finite or infinite words and for each $X \in E^{\mathbb{N}}$, $\|\lambda(X)\| = \|X\|$.

A digit set conversion can be applied to an addition and a multiplication with a fixed integer. An addition by a digit set conversion is a conversion from $E = \{e \in \mathbb{Z} | 2a \leq e \leq 2b\}$ to $\mathcal{D} = \{d \in \mathbb{Z} | a \leq d \leq b\}$. A multiplication with an integer m is a conversion from $E = \{e \in \mathbb{Z} | ma \leq e \leq mb\}$ to $\mathcal{D} = \{d \in \mathbb{Z} | a \leq d \leq b\}$ with the same base β. For example, an addition of two numbers in base $\beta = 2$ with a digit set $\mathcal{D} = \{\overline{1}, 0, 1\}$ can be described as a digit set conversion from a digit set $E = \{\overline{2}, \overline{1}, 0, 1, 2\}$ to a digit set $\mathcal{D} = \{\overline{1}, 0, 1\}$ with the same numerical value.

Example 2 An addition of $X = 1\overline{1}001\overline{1}$ and $Y = 101\overline{1}1\overline{1}$ in a base $\beta = 2$ by a digit set conversion.
An addition of any two numbers can be done by a digit set conversion from a digit set $E = \{\overline{2}, \overline{1}, 0, 1, 2\}$ to a digit set $\mathcal{D} = \{\overline{1}, 0, 1\}$ with the same base $\beta = 2$. A process starts by making a sum in all positions which gives a temporary result as $2\overline{1}1\overline{1}2\overline{2}$. This corresponds to the conversion from $2\overline{1}1\overline{1}2\overline{2}$ to $1\overline{1}0\overline{1}10\overline{1}0$ in base $\beta = 2$.

$$\begin{array}{rl} 1\,\overline{1}\,0\,0\,1\,\overline{1} & X \\ \underline{1\,0\,1\,\overline{1}\,1\,\overline{1}}+ & Y \\ 2\,\overline{1}\,1\,\overline{1}\,2\,\overline{2} & X+Y \\ \underline{1\,\overline{1}\,0\,\overline{1}\,1\,0\,\overline{1}\,0} & \end{array}$$

2.3. On-line Finite Automaton

In order to reduce an arithmetic computational time, an on-line finite automaton is applied with a digit set conversion to perform a pipelining arithmetic computation. An on-line finite automaton [14-17] is known as transducer or sequential automaton. This automaton can convert a digit set E to a digit set $\mathcal{D}$, flowing serially digit by digit, starting from the most significant digit.

Definition 3 (On-line finite automaton)
An on-line finite automaton $\mathcal{A} = (Q, E \times (\mathcal{D} \cup \varepsilon), q_0, F)$ with an on-line delay k consists of a finite set of states Q, a finite set of input E and output $\mathcal{D}$, an empty word ε, an initial state q_0, and a finite set of transition F which can be illustrated by $p \xrightarrow{x/y} q$ with $x \in E$ and $y \in \mathcal{D}$.

An on-line finite automaton has a special characteristic called an on-line delay k. The process of an on-line finite automaton with an on-line delay k starts from reading the first k input digits (from the most significant digit) without producing any output digits. From the $(k+1)^{th}$-position of the input, it produces one output digit after reading one input digit consequently. The transition of an on-line finite automaton can be defined as $p \xrightarrow{x/y} q$ where $x \in E$ and $y \in \mathcal{D}$. This automaton can be described with a directed labeled graph as follow.

$$q_0 \xrightarrow{x_1/\varepsilon} q_1 \xrightarrow{x_2/\varepsilon} q_2 \ldots \xrightarrow{x_k/\varepsilon} q_k$$

Before it produces an output result after reading all input digits and then stopping at state q_j, it needs an additional function to calculate the rest of the output result from a last state q_j. A function called *terminal function* $\omega: Q \to \mathcal{D}^*$ is defined to produce k output digits appended to the end of an output result. An example of an on-line finite automaton in base $\beta = 2$ that converts a digit set $E = \{\bar{2}, \bar{1}, 0, 1, 2\}$ to a digit set $\mathcal{D} = \{\bar{1}, 0, 1\}$ with an on-line delay $k = 2$ is shown in Fig. 1.

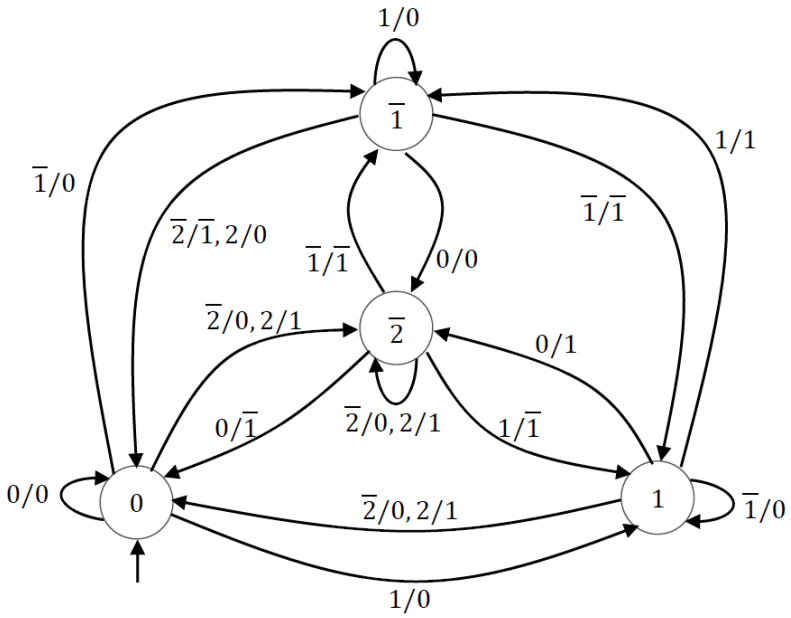

Fig. 1. An on-line finite automaton in base $\beta = 2$ that converts a digit set $E = \{\bar{2}, \bar{1}, 0, 1, 2\}$ to a digit set $\mathcal{D} = \{\bar{1}, 0, 1\}$ with an on-line delay $k = 2$ [16].

2.4. Round-to-Nearest Number System

Round-to-Nearest (RN) number system has an important property that truncating at any position of a sequence of digits (representation) is equivalent to rounding to the nearest. *Balanced ternary system* [18, 19] consists of a base $\beta = 3$ and a digit set $\mathcal{D} = \{\bar{1}, 0, 1\}$ is an example of the system that has this property. In 2005, Kornerup and Muller [7] introduced a generalization of round-to-nearest number system called *RN-coding* which was defined in definition 4.

Definition 4 (RN-coding)
A number system consists of a base $\beta \geq 2$ with a digit set $\mathcal{D} = \{-\beta + 1, -\beta + 2, \ldots, 0, \ldots, \beta - 2, \beta - 1\}$. Any number representation $X = (x_n x_{n-1} \ldots x_0 . x_{-1} x_{-2} \ldots)_\beta$ is an RN-coding if and only if

$$\forall i \le n, \left| \sum_{i=-\infty}^{j-1} x_i \beta^i \right| \le \frac{1}{2}\beta^j$$

Some important characteristics of an RN-coding are

- For an odd base β, a number representation X is an RN-coding if and only if $\forall i, \frac{-\beta+1}{2} \le x_i \le \frac{\beta-1}{2}$
- For an even base β, a number representation X is an RN-coding if and only if $\forall i, \frac{-\beta}{2} \le x_i \le \frac{\beta}{2}$ and if $|x_i| = \frac{\beta}{2}$ then the next first nonzero digit on the right must have an opposite sign compare to x_i.

Remark There are two representations of an RN-coding in an even base but there is only one representation in an odd base.

Example 3 RN-coding of $X = (151)_{10}$ in base $\beta = 2$ and $\beta = 3$.
RN-coding of $X = (151)_{10}$ in base $\beta = 2$ and $\beta = 3$ both use the same digit set $\mathcal{D} = \{\bar{1}, 0, 1\}$. For an odd base $\beta = 3$, a unique representation X is $(1\bar{1}0\bar{1}\bar{1})_3$. For an even base $\beta = 2$, there are two representations X that are $(1\bar{1}01\bar{1}100\bar{1})_2$ and $(1\bar{1}01\bar{1}10\bar{1}1)_2$. □

There are many researches in RN-coding [20-24] that propose about arithmetic algorithms together with their implementation. However, they do not support an on-line computational manner.

3. Rational Digit Number

We propose a rational digit number system that is suitable for an on-line digit set conversion. The definition of a rational digit number is described in definition 5.

Definition 5 (Rational digit number)
A number $X = (x_n x_{n-1} \dots x_0 . x_{-1} x_{-2} \dots)_\beta$ with $x_i \in \mathcal{D}$ and $i \le n$ is said to be a representation in a rational digit number if and only if a base $\beta \ge 2$ and a digit set $\mathcal{D} = \left\{-\alpha, -\alpha + \frac{1}{2}, \dots, 0, \dots, \alpha - \frac{1}{2}, \alpha\right\}$ where $\alpha = \frac{\beta + (\beta \bmod 2)}{4}$.

A rounding process of a rational digit number $x_n x_{n-1} \dots x_0 . x_{-1} x_{-2} \dots$ at position j $(j \le n)$ can be performed using only a truncation at the position j and

$$\left| \sum_{i=-\infty}^{j-1} x_i \times \beta^i \right| < \frac{1}{2}\beta^j$$

The obtained representation $X' = x_n x_{n-1} \dots x_j$ is always closest to X and also valid in a rational digit number representation.

To show the completeness of this number system, we introduce a theorem that can convert every conventional number into a rational digit number.

Theorem 1
Any conventional number can have a representation in a rational digit number.
Proof: To proof this theorem, we introduce two conversion algorithms for an even base and odd base. Both algorithms convert a conventional number into a rational digit number.

Algorithm 1 (Rational digit number conversion for an even base)
input $X = x_n x_{n-1} \dots x_0 . x_{-1} x_{-2} \dots$ where $0 \le x_i \le \beta - 1$
output $Z = z_{n+2} z_{n+1} \dots z_0 . z_{-1} z_{-2} \dots$ where $-\alpha \le z_i \le \alpha$
begin
$c_{initial} = 0$
for all $(i \le n)$

if $0 \le x_i \le \lceil\alpha\rceil - 1$ **then** $c_{i+1} = 0$ **endif**
if $\lceil\alpha\rceil \le x_i \le \frac{\beta}{2} + \lceil\alpha\rceil - 1$ **then** $c_{i+1} = \frac{1}{2}$ **endif**
if $\frac{\beta}{2} + \lceil\alpha\rceil \le x_i \le \beta - 1$ **then** $c_{i+1} = 1$ **endif**
$y_i = x_i - (c_{i+1} \times \beta) + c_i$
$y_{n+1} = c_{n+1}$

for all $(i \le n+1)$
if $\lceil-\alpha\rceil \le y_i \le \lfloor\alpha\rfloor$ **then** $c_{i+1} = 0$ **endif**
if $y_i = \alpha$ and $y_{i-1} < \alpha$ **then** $c_{i+1} = 0$ **endif**
if $y_i = \alpha$ and $y_{i-1} \ge \alpha$ **then** $c_{i+1} = \frac{1}{2}$ **endif**
if $y_i = \lceil\alpha\rceil$ **then** $c_{i+1} = \frac{1}{2}$ **endif**
$z_i = y_i - (c_{i+1} \times \beta) + c_i$
$z_{n+2} = c_{n+2}$

end

Proof of the algorithm: We separate the proof of this algorithm into two parts;

Part 1: Given $\alpha = \frac{\beta}{4}$. All input digits x_i will be converted to temporary output digits y_i where $0 \le x_i \le \beta - 1$ and $\lceil-\alpha\rceil \le y_i \le \lceil\alpha\rceil$. Since $y_i = x_i - (c_{i+1} \times \beta) + c_i$ and $c_i \in \{0, \frac{1}{2}, 1\}$, it is found that $x_i = y_i + c_{i+1}\beta - c_i$.

Case: $0 \le x_i \le \lceil\alpha\rceil - 1$

From the algorithm, $c_{i+1} = 0$ and $x_i = y_i - c_i$. Hence $c_i \le y_i \le \lceil\alpha\rceil - 1 + c_i$. Since $c_i \in \{0, \frac{1}{2}, 1\}$ then all possible values of y_i satisfy $0 \le y_i \le \lceil\alpha\rceil$.

Case: $\lceil\alpha\rceil \le x_i \le \frac{\beta}{2} + \lceil\alpha\rceil - 1$

From the algorithm, $c_{i+1} = \frac{1}{2}$ and $x_i = y_i + \frac{\beta}{2} - c_i$. Hence $\lceil-\alpha\rceil + c_i \le y_i \le \lceil\alpha\rceil - 1 + c_i$. Since $c_i \in \{0, \frac{1}{2}, 1\}$ then all possible values of y_i satisfy $\lceil-\alpha\rceil \le y_i \le \lceil\alpha\rceil$.

Case: $\frac{\beta}{2} + \lceil\alpha\rceil \le x_i \le \beta - 1$

From the algorithm, $c_{i+1} = 1$ and $x_i = y_i + \beta - c_i$. Hence $\lceil-\alpha\rceil + c_i \le y_i \le -1 + c_i$. Since $c_i \in \{0, \frac{1}{2}, 1\}$ then all possible values of y_i satisfy $\lceil-\alpha\rceil \le y_i \le 0$.

From three cases, we can conclude that $\lceil-\alpha\rceil \le y_i \le \lceil\alpha\rceil$.

Part 2: Some temporary output digits y_i may not be valid in our system. In that case, the algorithm proceeds to convert all temporary output digits y_i to final output digits z_i where $-\alpha \le z_i \le \alpha$. Since $z_i = y_i - (c_{i+1} \times \beta) + c_i$ and $c_i \in \{0, \frac{1}{2}\}$, it is found that $y_i = z_i + c_{i+1}\beta - c_i$.

Case: $\lceil-\alpha\rceil \le y_i \le \lfloor\alpha\rfloor$

From the algorithm,$c_{i+1} = 0$ and $y_i = z_i - c_i$. Hence $\lceil-\alpha\rceil + c_i \le z_i \le \lfloor\alpha\rfloor + c_i$. Since $c_i \in \{0, \frac{1}{2}\}$ then all possible values of z_i satisfy $\lceil-\alpha\rceil \le z_i \le \alpha$.

Case: $y_i = \alpha$ and $y_{i-1} < \alpha$

From the algorithm, $c_{i+1} = 0$ and $y_i = z_i - c_i$. Hence $z_i = \alpha + c_i$. Since $c_i = 0$ then $z_i = \alpha$.

Case: $y_i = \alpha$ and $y_{i-1} \ge \alpha$

From the algorithm, $c_{i+1} = \frac{1}{2}$ and $y_i = z_i + \frac{\beta}{2} - c_i$. Hence $z_i = -\alpha + c_i$. Since $c_i \in \{0, \frac{1}{2}\}$ then all possible values of z_i satisfy $-\alpha \le z_i \le -\alpha + \frac{1}{2}$.

Case: $y_i = \lceil\alpha\rceil$

From the algorithm, $c_{i+1} = \frac{1}{2}$ and $y_i = z_i + \frac{\beta}{2} - c_i$. Hence $z_i = \lceil-\alpha\rceil + c_i$. Since $c_i \in \{0, \frac{1}{2}\}$ then all possible values of z_i satisfy $\lceil-\alpha\rceil \le z_i \le \lceil-\alpha\rceil + \frac{1}{2}$.

From four cases, we can conclude that $-\alpha \le z_i \le \alpha$ which are valid in our system. ■

For better understanding about Algorithm 1, an example of a conversion from any conventional number into a rational digit number with the same even base is shown in example 4.

Example 4 Let β=4, find a representation of $X = (10231)_4$ in a rational digit number.
From Algorithm 1, a conventional number representation $X = (10231)_4$ in an even base with a digit set $\mathcal{D} = \{0, 1, 2, 3\}$ can be converted to a rational digit number representation $Z = (\frac{1}{2}\,\overline{1}\,\frac{1}{2}\,1\,\frac{\overline{1}}{2}\,\overline{1})_4$ with a digit set $\mathcal{D}' = \{\overline{1}, \frac{\overline{1}}{2}, 0, \frac{1}{2}, 1\}$ in the same base. □

Algorithm 2 (Rational digit number conversion for an odd base)
input $X = x_n x_{n-1} \ldots x_0 . x_{-1} x_{-2} \ldots$ where $0 \le x_i \le \beta - 1$
output $Z = z_{n+1} z_n \ldots z_0 . z_{-1} z_{-2} \ldots$ where $-\alpha \le z_i \le \alpha$
begin
$c_{initial} = 0$
for all $(i \le n)$
if $0 \le x_i \le \alpha - 1$ **then** $c_{i+1} = 0$ **endif**
if $\alpha - \frac{1}{2} \le x_i \le \frac{\beta}{2} + \alpha - 1$ **then** $c_{i+1} = \frac{1}{2}$ **endif**
if $\frac{\beta}{2} + \alpha - \frac{1}{2} \le x_i \le \beta - 1$ **then** $c_{i+1} = 1$ **endif**
$z_i = x_i - (c_{i+1} \times \beta) + c_i$
$z_{n+1} = c_{n+1}$
end

Proof of the algorithm: Given $\alpha = \frac{\beta+1}{4}$. Since $z_i = x_i - (c_{i+1} \times \beta) + c_i$ and $c_i \in \{0, \frac{1}{2}, 1\}$, it is found that $x_i = z_i + c_{i+1}\beta - c_i$.
Case: $0 \le x_i \le \alpha - 1$
From the algorithm, $c_{i+1} = 0$ and $x_i = z_i - c_i$. Hence $c_i \le z_i \le \alpha - 1 + c_i$. Since $c_i \in \{0, \frac{1}{2}, 1\}$, it is found that $0 \le z_i \le \alpha$.
Case: $\alpha - \frac{1}{2} \le x_i \le \frac{\beta}{2} + \alpha - 1$
From the algorithm, $c_{i+1} = \frac{1}{2}$ and $x_i = z_i + \frac{\beta}{2} - c_i$. Hence $-\alpha + c_i \le z_i \le \alpha - 1 + c_i$. Since $c_i \in \{0, \frac{1}{2}, 1\}$, it is found that $-\alpha \le z_i \le \alpha$.
Case: $\frac{\beta}{2} + \alpha - \frac{1}{2} \le x_i \le \beta - 1$
From the algorithm, $c_{i+1} = 1$ and $x_i = z_i + \beta - c_i$. Hence $-\alpha + c_i \le z_i \le -1 + c_i$. Since $c_i \in \{0, \frac{1}{2}, 1\}$, it is found that $-\alpha \le z_i \le 0$.
From three cases, we can conclude that $-\alpha \le z_i \le \alpha$ which are valid in our system. ■

The following example demonstrates a conversion from a conventional number into a rational digit number with the same odd base.

Example 5 Let β=5, find a representation of $X = (1240)_5$ in a rational digit number.
A conventional number representation $X = (1240)_5$ in an odd base with a digit set $\mathcal{D} = \{0, 1, 2, 3, 4\}$ can be converted to a rational digit number representation $Z = (\frac{1}{2}\,\overline{1}\,\frac{1}{2}\,\overline{1}\,0)_5$ in the same base with a digit set $\mathcal{D}' = \{\frac{\overline{3}}{2}, \overline{1}, \frac{\overline{1}}{2}, 0, \frac{1}{2}, 1, \frac{3}{2}\}$. □

4. On-line Digit Set Conversion

In this work, we propose an on-line digit set conversion algorithm for a rational digit number. This algorithm can be performed by an on-line finite automaton with an on-line delay k and β^k states.

Algorithm 3 (On-line digit set conversion for rational digit number)
input $X = x_n x_{n-1} \ldots$ where $-\gamma \le x_i \le \gamma$
output $Z = z_{n+k} z_{n+k-1} \ldots$ where $-\alpha \le z_i \le \alpha$
begin
$m = \frac{\gamma}{\alpha}$
$q_{n+k} = 0$ (initial state)

$g \geq \frac{(-\alpha)(\beta^k - m)}{\beta - 1}$
$j = n$
while $j \leq n$ **do**
$z_{j+k} \leq \frac{-g + (q_{j+k}\beta) + x_j}{\beta^k}$
$q_{j+k-1} = \left(q_{j+k}\beta + x_j\right) - (z_{j+k}\beta^k)$
$j = j - 1$
enddo
end

Before proving this algorithm, we start by introducing how to construct an on-line finite automaton for this on-line digit set conversion. After that we introduce two lemmas that are required in the proof of this algorithm. Lemma 1 describes how to construct an on-line finite automaton and show that a set of states Q is finite. Lemma 2 describes how to derive an inequality of an on-line delay k.

We can construct an on-line finite automaton $\mathcal{A} = (Q, E \times \mathcal{D}, 0, F)$ where Q is a finite set of states, $E = \{-\gamma, \ldots, \gamma\}$ and $\mathcal{D} = \{-\alpha, \ldots, \alpha\}$ are finite sets of input and output digits, 0 is an initial state and F is a set of transitions. Every transition can be defined as

$$q_{j+k} \xrightarrow{x_j / z_{j+k}} q_{j+k-1}$$

with q_{j+k} and q_{j+k-1} are the member of Q, x_j is an input digit, z_{j+k} is an output digit.

Lemma 1

A set of states Q in an on-line finite automaton is finite and valid in $\frac{(-\alpha)(\beta^k - m)}{\beta - 1} \leq q \leq \frac{\alpha(\beta^k - m)}{\beta - 1}$.

Proof: We assume that g and h is a lower bound and an upper bound of a state $q \in Q$.

$$g \leq q \leq h \tag{1}$$

We define that $Q = \{g, g + \frac{1}{2}, \ldots, 0, \ldots, h - \frac{1}{2}, h\}$. Given a base $\beta \geq 2$, a state q_{j+k-1} is defined as

$$q_{j+k-1} = \left(q_{j+k}\beta + x_j\right) - (z_{j+k}\beta^k) \tag{2}$$

From Eq. (1) and Eq. (2),

$g \leq \left(q_{j+k}\beta + x_j\right) - (z_{j+k}\beta^k) \leq h.$

An output digit z_{j+k} can be calculated as

$$\frac{-h + (q_{j+k}\beta + x_j)}{\beta^k} \leq z_{j+k} \leq \frac{-g + (q_{j+k}\beta + x_j)}{\beta^k} \tag{3}$$

From Eq. (3), we will calculate g from this inequality

$$z_{j+k} \leq \frac{-g + (q_{j+k}\beta + x_j)}{\beta^k}$$

Given $= \frac{\beta + (\beta mod 2)}{4}$, $z_{j+k} = -\alpha$, $q_{j+k} = g$, $x_j = m(-\alpha)$ and $m = \frac{\gamma}{\alpha}$. Hence,

$$g \geq \frac{(-\alpha)(\beta^k - m)}{\beta - 1} \tag{4}$$

From Eq. (3), h can be calculated from this inequality

$$z_{j+k} \geq \frac{-h + (q_{j+k}\beta + x_j)}{\beta^k}$$

Given $z_{j+k} = \alpha, q_{j+k} = h, x_j = m(\alpha)$ and $m = \frac{\gamma}{\alpha}$. Hence,

$$h \leq \frac{\alpha(\beta^k - m)}{\beta - 1} \tag{5}$$

Thus, from Eq. (4) and Eq. (5), a set of states Q is finite and valid in

$$\frac{(-\alpha)(\beta^k - m)}{\beta - 1} \leq q \leq \frac{\alpha(\beta^k - m)}{\beta - 1}$$

Lemma 2

An on-line delay k of an on-line finite automaton is $k \geq \log_\beta(\frac{1 + 4\alpha m - \beta}{1 + 4\alpha - \beta})$.

Proof: Given a base $\beta \geq 2$. Since the number of states q is β^k then

$$2|g| + 2|h| + 1 \geq \beta^k$$

From Lemma 1, $g \geq \frac{(-\alpha)(\beta^k - m)}{\beta - 1}$ and $h \leq \frac{\alpha(\beta^k - m)}{\beta - 1}$ then

$$2\left|\frac{(-\alpha)(\beta^k-m)}{\beta-1}\right| + 2\left|\frac{\alpha(\beta^k-m)}{\beta-1}\right| + 1 \geq \beta^k$$

Thus, an on-line delay k can be calculated from this inequality

$$k \geq \log_\beta(\tfrac{1+4\alpha m-\beta}{1+4\alpha-\beta})$$

Proof of the algorithm 3: From Lemma 1, states q are valid in $l \leq g \leq q \leq h \leq u$ where $l = \frac{-\alpha(\beta^k-m)}{\beta-1}$ and $u = \frac{\alpha(\beta^k-m)}{\beta-1}$. From Lemma 2, we will show that an on-line delay k is enough to produce the last k digits of the result. We divide the proof into two parts

Part 1: we will show that $(-\alpha)\beta^k + (-\alpha)\beta^{k-1} + \cdots + (-\alpha) \leq l$.

$$\begin{aligned}
\beta^{k+2} &\geq \beta^k \\
\beta^{k+2} - \beta^{k+1} &\geq \beta^k \\
\beta^{k+2} - \beta^{k+1} - \beta + 1 &\geq \beta^k - m \\
(\beta^{k+1} - 1)(\beta - 1) &\geq \beta^k - m \\
\beta^{k+1} - 1 &\geq \tfrac{\beta^k - m}{\beta-1} \\
(-\alpha)(\beta^{k+1} - 1) &\leq (-\alpha)(\tfrac{\beta^k - m}{\beta-1}) \\
(-\alpha)\beta^{k+1} - (-\alpha) &\leq \tfrac{-\alpha(\beta^k-m)}{\beta-1} \\
(-\alpha)\beta^k + (-\alpha)\beta^{k-1} + \cdots + (-\alpha) &\leq \tfrac{-\alpha(\beta^k-m)}{\beta-1}
\end{aligned} \tag{6}$$

Part 2: we will show that $\alpha\beta^k + \alpha\beta^{k-1} + \cdots + \alpha \geq u$.

$$\begin{aligned}
\beta^{k+2} &\geq \beta^k \\
\beta^{k+2} - \beta^{k+1} &\geq \beta^k \\
\beta^{k+2} - \beta^{k+1} - \beta + 1 &\geq \beta^k - m \\
(\beta^{k+1} - 1)(\beta - 1) &\geq \beta^k - m \\
\beta^{k+1} - 1 &\geq \frac{\beta^k - m}{\beta-1} \\
\alpha(\beta^{k+1} - 1) &\geq \alpha(\frac{\beta^k - m}{\beta-1}) \\
\alpha\beta^{k+1} - \alpha &\geq \tfrac{\alpha(\beta^k-m)}{\beta-1} \\
\alpha\beta^k + \alpha\beta^{k-1} + \cdots + \alpha &\geq \tfrac{\alpha(\beta^k-m)}{\beta-1}
\end{aligned} \tag{7}$$

From Eq. (6) and Eq. (7), we can conclude that an on-line delay k is enough to produce the last k digits of the result.

Some examples about an on-line digit set conversion in a rational digit number are shown in example 6 and example 7. An on-line digit set conversion in a base $\beta = 3$ with $m = 2$ and an on-line delay $k = 1$ is described in example 6. Example 7 illustrates an on-line digit set conversion in the same base $\beta = 3$ but change $m = 4$ and an on-line delay $k = 2$.

Example 6 Given a number $X = (1\bar{\frac{1}{2}}0\bar{2}1\frac{1}{2})_3$ where $X \in E^*$. Find a result of a digit set conversion of X from a digit set $E = \{-2, \ldots, 2\}$ to a digit set $\mathcal{D} = \{-1, \ldots, 1\}$ in a rational digit number with a base $\beta = 3$ by an on-line finite automaton.

From Algorithm 3, a digit set conversion of a number $X = (1\bar{\frac{3}{2}}0\bar{2}1\frac{1}{2})_3$ can be performed using an on-line finite automaton with $m = 2$. An on-line delay k can be calculated from Lemma 2 which is $k = 1$. A digit set conversion from a digit $E = \{-2, \ldots, 2\}$ to a digit set $\mathcal{D} = \{-1, \ldots, 1\}$ in a rational digit number in a base $\beta = 3$ by an on-line finite automaton with an on-line delay $k = 1$ in Fig. 2 can be illustrated by this directed labelled graph,

$$0 \xrightarrow{1\,/\,\frac{1}{2}} \frac{-1}{2} \xrightarrow{\frac{\bar{3}}{2}\,/\,\bar{1}} 0 \xrightarrow{0\,/\,0} 0 \xrightarrow{\bar{2}\,/\,\frac{\bar{1}}{2}} \frac{-1}{2} \xrightarrow{1\,/\,0} \frac{-1}{2} \xrightarrow{\frac{1}{2}\,/\,\frac{\bar{1}}{2}} \frac{1}{2}$$

After reading all input digits, a temporary result is $\frac{1}{2}\bar{1}0\frac{\bar{1}}{2}0\frac{\bar{1}}{2}$. There is one more step needed for completing this conversion. Since a conversion process stops at state $q = \frac{1}{2}$ and an on-line delay $k = 1$ then we define a terminal function $\omega: Q \to \mathcal{D}^*$ such that $\omega\left(\frac{1}{2}\right) = \frac{1}{2}$. We append a digit $\frac{1}{2}$ at the end of a temporary result. Finally, a final result is $(\frac{1}{2}\bar{1}0\frac{\bar{1}}{2}0\frac{\bar{1}1}{22})_3$ is equal to 107. □

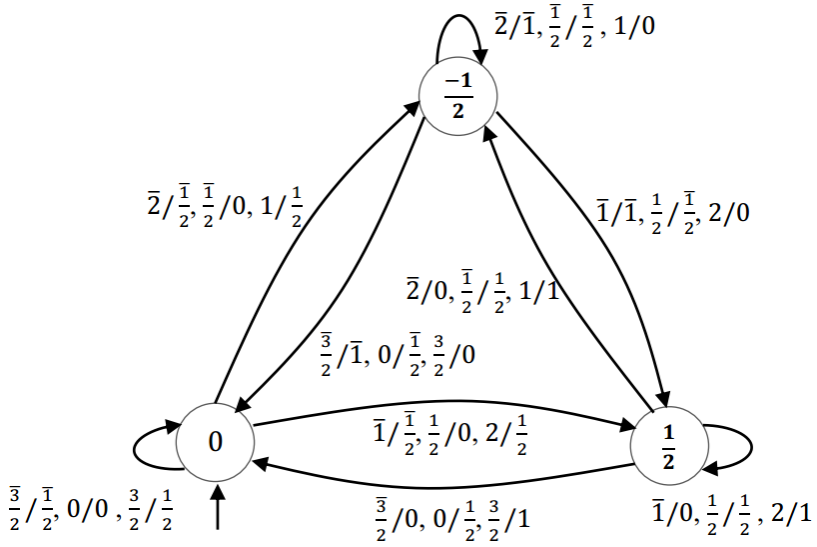

Fig. 2. An on-line finite automaton in base $\beta = 3$ with $m = 2$ and an on-line delay $k = 1$.

Example 7 Given a number $X = (3\frac{7}{2}\frac{\bar{3}}{2}\bar{4}2\frac{1}{2})_3$ where $X \in E^*$. Find a result of a digit set conversion of X from a digit set $E = \{-4, \dots, 4\}$ to a digit set $\mathcal{D} = \{-1, \dots, 1\}$ in a rational digit number with a base $\beta = 3$ by an on-line finite automaton.

A number $X = (3\frac{7}{2}\frac{\bar{3}}{2}\bar{4}2\frac{1}{2})_3$ has a numerical value which is 942.5. A digit set conversion from a digit set $E = \{-4, \dots, 4\}$ to a digit set $\mathcal{D} = \{-1, \dots, 1\}$ in a rational digit number where $m = 4$ in a base $\beta = 3$ can be performed by an on-line finite automaton. An on-line delay k can be calculated from Lemma 2 that is equal to $k = 2$. We show briefly the way to construct an on-line finite automaton by Algorithm 3. An output digit z_{j+k} can be calculated from an incoming input digit x_j with a current state q_{j+k} which can be demonstrated by a following inequality

$$z_{j+k} \leq \frac{\frac{-5}{2} + (q_{j+k} \times 3) + x_j}{9}$$

The next state q_{j+k-1} can be derived from

$$q_{j+k-1} = \left((q_{j+k} \times 3) + x_j\right) - (z_{j+k} \times 9)$$

Hence, this digit set conversion by an on-line finite automaton with an on-line delay $k = 2$ can be illustrated by a following directed labelled graph,

$$0 \xrightarrow{3/\frac{1}{2}} \frac{-3}{2} \xrightarrow{\frac{7}{2}/0} -1 \xrightarrow{\frac{\bar{3}}{2}/\frac{\bar{1}}{2}} 0 \xrightarrow{\bar{4}/\frac{\bar{1}}{2}} \frac{1}{2} \xrightarrow{2/\frac{1}{2}} -1 \xrightarrow{\frac{1}{2}/0} \frac{-5}{2}$$

After reading all input digits, a temporary result is $\frac{1}{2}0\frac{\bar{1}}{2}\frac{\bar{1}}{2}\frac{1}{2}0$. Since a final state $q = \frac{-5}{2}$ and an on-line delay $k = 2$ then we will define a terminal function $\omega: Q \to \mathcal{D}^*$ such that $\omega\left(\frac{-5}{2}\right) = \bar{1}\frac{1}{2}$. Thus, the final result $(\frac{1}{2}0\frac{\bar{1}}{2}\frac{\bar{1}}{2}\frac{1}{2}0\bar{1}\frac{1}{2})_3$ gives a correct result. □

5. Conclusion

In this work, we propose an on-line digit set conversion algorithm for a rational digit number system. This algorithm converts a digit set $E = \{-\alpha \times m, \dots, \alpha \times m\}$ to a digit set $\mathcal{D} = \{-\alpha, \dots, \alpha\}$ with $\alpha = \frac{\beta + (\beta mod 2)}{4}$ in the same base $\beta \geq 2$. We show that an on-line digit set conversion can be illustrated by an on-line finite

automaton with an on-line delay k where $k \geq \log_\beta(\frac{1+4\alpha m-\beta}{1+4\alpha-\beta})$. In our future work, we will concentrate on non-classical bases such as a negative base and a complex base to work with our system.

References

[1] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," IRE Transactions on Electronic Computers, vol. EC-10, no. 3, pp. 389-400, 1961.

[2] R. E. Moore, *Interval Analysis*, vol. 4. Englewood Cliffs: Prentice-Hall, 1966.

[3] R. E. Moore and R. Moore, *Methods and Applications of Interval Analysis*, vol. 2. SIAM, 1979.

[4] R. B. Kearfott, "Interval computations: Introduction, uses, and resources," *Euromath Bulletin*, vol. 2, pp. 95-112, 1996.

[5] P. Thienprapasith and A. Surarerks, "A flexible interval representation system and its fundamental arithmetic operations," in *Proceeding of the 5th International Conference on Information Technology and Applications (ICITA 2008)*, Cairns, 2008.

[6] P. Thienprapasith and A. Surarerks, "Flexible interval representation system in negative binary base," in *2016 8th International Conference on Knowledge and Smart Technology (KST)*, 2016, pp. 190-195.

[7] P. Kornerup and J.-M. Muller, "RN-coding of numbers: Definition and some properties," in *Proceedings of IMACS'2005*, Paris, 2005.

[8] M. D. Ercegovac, "On-line arithmetic: An overview," in *28th Annual Technical Symposium*, 1984, pp. 86-93.

[9] K. S. Trivedi and M. D. Ercegovac, "On-line algorithms for division and multiplication," *in Computer Arithmetic (ARITH), 1975 IEEE 3rd Symposium on*, 1975, pp. 161-167.

[10] P. Thienprapasith and A. Surarerks, "Rational digit set for on-line addition," in *The 13th International Joint Conference on Computer Science and Software Engineering (JCSSE2016)*, Khon Kaen, Thailand, 2016.

[11] B. Parhami, "Generalized signed-digit number systems: A unifying framework for redundant number representations," *IEEE Transactions on Computers*, vol. 39, pp. 89-98, 1990.

[12] P. Kornerup, "Digit-set conversions: generalizations and applications," *IEEE Transactions on Computers*, vol. 43, pp. 622-629, 1994.

[13] B. Phillips and N. Burgess, "Minimal weight digit set conversions," *IEEE Transactions on Computers*, vol. 53, pp. 666-677, 2004.

[14] C. Frougny, "On-line digit set conversion in real base," *Theoretical Computer Science*, vol. 292, pp. 221-235, 2003.

[15] J.-M. Muller, "Some characterizations of functions computable in on-line arithmetic," *IEEE Transactions on Computers*, vol. 43, pp. 752-755, 1994.

[16] A. Surarerks, "Digit set conversion by on-line finite automata," *Bulletin of the Belgian Mathematical Society Simon Stevin*, vol. 8, pp. 337-358, 2001.

[17] N. Tanatechawong, "On-line addition on Penney complex number system," Master's thesis, Computer Engineering, Chualolongkorn University, 2007.

[18] B. Parhami and M. McKeown, "Arithmetic with binary-encoded balanced ternary numbers," in *Signals, Systems and Computers, 2013 Asilomar Conference on*, 2013, pp. 1130-1133.

[19] B. Parhami, "Truncated ternary multipliers," *IET Computers & Digital Techniques*, vol. 9, pp. 101-105, 2014.

[20] J.-L. Beuchat and J.-M. Muller, "Multiplication algorithms for radix-2 RN-Codings and Two's Complement Numbers," in *Proc. Application-Specific Systems, Architecture Processors, 2005. ASAP 2005. 16th IEEE International Conference on*, 2005, pp. 303-308.

[21] P. Kornerup and J.-M. Muller, "RN-codings: New insights and some applications," in *Proceedings of the 7th Conference on Real Numbers and Computers (RNC'7)*, 2006, p. 117.

[22] A. Panhaleux, "Contributions to floating-point arithmetic: Coding and correct rounding of algebraic functions," Doctoral dissertation, Ecole normale supérieure de lyon-ENS LYON, 2012.

[23] P. Kornerup, J.-M. Muller, and A. Panhaleux, "Performing arithmetic operations on round-to-nearest representations," *IEEE Transactions on Computers*, vol. 60, pp. 282-291, 2011.

[24] P. Kornerup, J.-M. Muller, and A. Panhaleux, "Floating-point arithmetic on round-to-nearest representations," Internal Report, Oct. 7, 2011.