A Double Negative-base Number representation system and Arithmetic Operations

Sakrapee Leelatham and Athasit Surarerks

Department of Computer Engineering, Chulalongkorn University,
254 Phyathai Road, Patumwan, Bangkok, Thailand
sakrapee.l@student.chula.ac.th and athasit@cp.eng.chula.ac.th

Abstract

Among several arithmetic operations required in the speed of computation, double base number system (DBNS) plays a significant role for fast arithmetic operations. The double base number representation used the orthogonal bases 2 and 3 is highly redundant and also provides much more sparse representations. DBNS arithmetic operations can be fast when performed in parallel manner because carry propagation chain is limited. This system has been proved to be effective for applications in cryptography and digital signal processing. Nevertheless, only positive integers can have a representation. In our work, we focus on the double negative-bases (DNBNS) using bases -2 and -3 which can describe all integers. We prove the completeness of this number system. We also introduce a normalization algorithm for reducing the consecutive number of 1's. Finally, we propose a conversion algorithm from binary into DNBNS and also demonstrate some useful algebra rules of addition and multiplication.

Keywords: Double base number system, computer arithmetic, arithmetic operations.

1. Introduction

A number representation and arithmetic operations play a significant role in computer arithmetic, the logical or mathematical theory which operates on numbers [2]. A binary number system is a basic number representation viewed as a specialized part of computational machines. Several different number systems, such as redundant number system, logarithm number system, flexible number system (*i.e.*, expressed for interval arithmetic) [11], and double base number system etc. have been introduced for some specific purposed systems. The concept of number system where a number can have more than one representation is introduced; the system is called *redundant* number system. Since redundancy provides carry free arithmetic operations which can speed up the computation, the classical redundant number system named *signed-digit number system* was first presented in 1960 by A. Avizienis [1]. Since three digits -1, 0, and 1 are allowed in the representation with base 2, the system becomes

redundant. Fundamental arithmetic operators are demonstrated to be parallelized in this system. The carry propagation chain is proved to be eliminated. Although parallel multiplication can be realized, they need at least a logarithmic time on the size of operands.

Later on, an interesting architecture for number representation named *double base number system* (DBNS) has been introduced in 1996 by V.S. Dimitrov and G.A. Julien [7]. The system is proposed to be applied in FIR filter implementation and cryptography. The DBNS is not only highly redundant, but it also provides much more sparseness. It is shown that the system becomes redundant using only two digits 1 and 0. However, the classical DBNS can represent only non-negative integers. In the DBNS, many researches studied about the minimal number of non-zero digits, so-called *canonical* form [3, 5-7]. Finding the canonic representation of a number in DBNS appears to be an NP-complete problem [6]. Greedy algorithms are thus proposed to represent number in DBNS containing a small number of non-zero digits, named *near canonical* representation [8]. The objective of our work is to establish the double negative-base number representation to extend this number scheme using the negative base -2 and -3.

This paper is organized as follows. Section 2 gives an overview on the DBNS and its arithmetic operations (addition and multiplication). In Section 3, we propose a double negative-base number system and also concentrate on the *completeness* of our system. Moreover, the *conversion* algorithm from binary number into the proposed number representation is characterized in Section 4. Some fundamental arithmetic operations such as addition, subtraction and multiplication are described by some algebra rules in Section 5. The conclusion is discussed in Section 6.

2. The double base number system

In this section, we review some notations and definitions on the double base number system (DBNS). We start by recall some notation and its redundant property. Addition and multiplication are also presented.

2.1 The DBNS representation

In 1996, V.S. Dimitrov and G.A. Julien had proposed the DBNS which is a power number system used bases 2 and 3 and allows only digit 0 or 1, see detail in [5]. The DBNS represents a given integer x into the form:

$$x = \sum_{alli,j} d_{i,j} 2^i 3^j, d_{i,j} \in \{0,1\} \qquad (\mathbf{1})$$

where i and j are independent non-negative integers. This number representation can be illustrated graphically as a 2-dimensional tabular form illustrated by Fig. 1. Each row (column) is indexed by the power of 2 (3). One cell represents a product of a row

index and a column index. Each non-zero digit is demonstrated as a gray cell, named an *active* cell, and each zero digit cell as a white cell. The DBNS representation is apparently very high redundant and sparseness. The representation containing a minimum number of non-zero digits will be referred to the *canonic* double base number representation (CDBNR). The canonic number representation is useful for fast carry-free arithmetic operations.

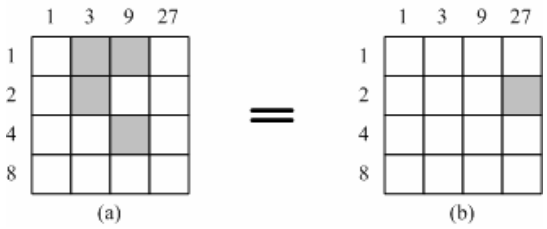

Figure 1. *Two different DBNS representations stand for an integer 54.*
(a) The representation contains four active cells.
(b) The canonic form of 54 contains only one active cell.

Although a number can have many canonic forms, the problem for generating a canonic form for a very large integer turns into a complex computational problem (NP-Complete). A recent algorithm for generating a canonic form is studied by G. Gilbert and J.M. Pierre Langlois in 2005 [8]. Nevertheless, a representation that does not contain any two consecutive active cells (in row or column) named *addition ready* (ARDBNR) can provide the sparseness in the number representation. Two algebra rules (column and row reductions) are produced to normalize any DBNR into ARDBNR as illustrated in Fig. 2.

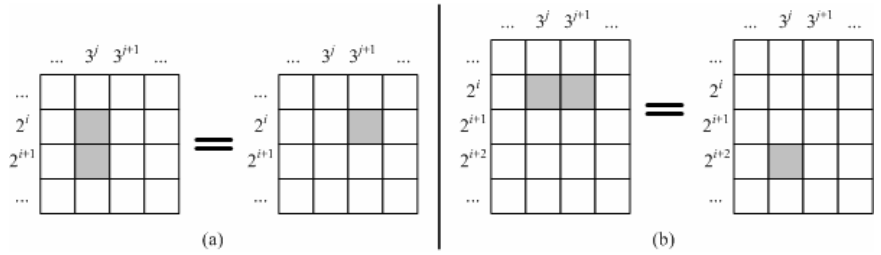

Figure 2. *Two algebra rules are proposed for reducing any consecutive active cells. The column rule(a): $2^i3^j + 2^{i+1}3^j = 2^i3^{j+1}$ reduces any two consecutive active cells in a column into one active cell. By the similar way, the row rule(b): $2^i3^j + 2^i3^{j+1} = 2^{i+2}3^j$ is used to decrease any two consecutive active cells in a row.*

Even though there are not appropriate approaches to find canonical representations, a simple greedy algorithm is introduced for finding a representation with less number of active cells. The recursive technique is to represent an integer x by the largest cell which is less than or equal to x combining with the representation of the difference. For example, the canonical and near canonical representation of 41 is shown in Fig. 3.

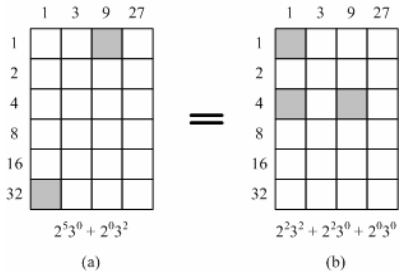

Figure 3. *(a) The canonical representation of 41 has only two active cells, 32 and 9. (b) The near canonical representation of 41 holds three active cells. Since 36 is the largest cell that is less than or equal to 41, the representation contains 36. The rest is performed by the same way.*

2.2 The DBNS addition and multiplication

In this section, we will discuss about some fundamental arithmetic operations. From (1), the system can represent only non-negative integers; we then discuss the operations specified addition and multiplication. Addition in DBNS can be computed by overlaying the matching DBNR maps. Then, the collision of two digits (active cells) can be occurred in the resulted map. The problem can be solved by applying the collision reduction rule: $2^i3^j + 2^i3^j = 2^{i+1}3^j$. For instance, addition of 7 and 37 is shown in Fig. 4. In order to simplify the addition, operands are usually represented in the ARDBNR form. The result is also produced in the ARDBNR form.

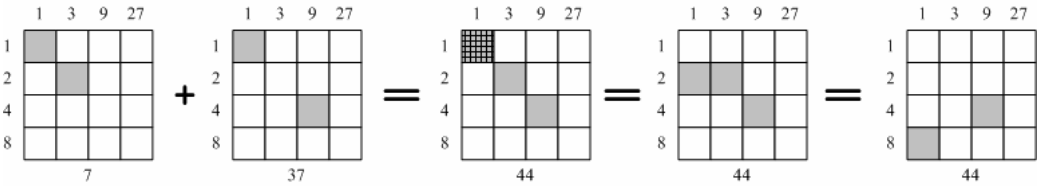

Figure 4*. Addition of 7 and 34 is realized by overlaying the matching operand maps. The collision of two cells at 2^03^0 is reduced into 2^13^0 by the collision rule. The row reduction rule is applied to remove two consecutive active cells, the result then contains two active cells. Grid cell states a cell where collision occurs.*

The product of two numbers can be achieved by two-dimensional shift of active cells combining with addition. Fig. 5 shows the product of 5 by 11. By the same concept, the operands and the result must be in the ARDBNR.

In addition, many researches are focused on implementation. For example, Moore machines (transducers) are selected to implement the adder circuit [12]. The concept of double base chains is shown to be useful for elliptic curve point multiplication algorithm; the complexity of scalar multiplication on elliptic curves can be reduced [4]. In 1998, the work of S. Sadeghi-Emamchaie et al, in [10] concentrated on

improving the performance in stability and robustness of arbitrary length arithmetic represented in DBNS by applying cellular neural networks (CNNs) for the implementation.

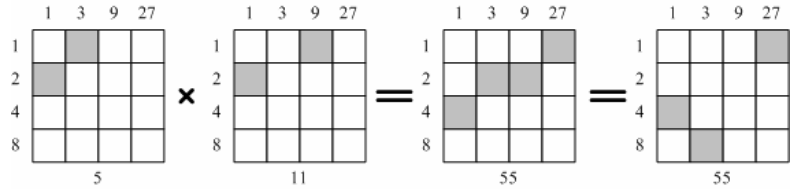

Figure 5. *Multiplication of 5 and 11 is performed by shifting and overlaying technique. The result should be represented in ARDBNR.*

3. The double negative-base number system

Although the classical DBNS has very high redundancy, sparseness properties and carry propagation chain limitation for speed up the computation, this number system can represent only positive integers. A signed-digit is an interesting candidate concept for proposing negative representations. Since the DBNS provides more high space, double the size by signed-digit is not a suitable technique. Hereafter, a representation with different digit sets has been described in order to extend this number system into negative representations [9].

In this work, we introduce the double negative-base number system (DNBNS). The system can represent all integers using the bases -2 and -3 together with the digit-set $\{0, 1\}$. The representation is given by the following definition.

Definition 1. *Let $\beta_1 = -2$ and $\beta_2 = -3$ be two bases in the system and let $d_{i,j}$ be a digit in a digit set {0, 1}. A representation in double negative-base number system (DNBNS) of an integer x is illustrated as follow:*

$$x = \sum_{all\, i,j} d_{i,j} (\beta_1)^i (\beta_2)^j, d_{i,j} \in \{0,1\} \qquad \textbf{(2)}$$

where i and j are non-negative integers.

Two negative bases can characterize the sign of each position in the representation. Fig.6 shows the feature of sign in the representation map. With this technique, the redundancy degree is then decreased; many redundant representations are replaced by some negative representations. It is obvious that one digit in DNBNR requires only one bit comparing with a signed-digit which uses two bits for the representation. Examples for the DNBNR of 126 and -126 are shown in Fig. 7.

-2/-3	$(-3)^0$	$(-3)^1$	$(-3)^2$	$(-3)^3$	...
$(-2)^0$	+	-	+	-	...
$(-2)^1$	-	+	-	+	...
$(-2)^2$	+	-	+	-	...
$(-2)^3$	-	+	-	+	...
:	:	:	:	:	...

Figure 6. *This table illustrates the signed characteristic of the double negative-base number representation. These can be realized because two bases are negative integers.*

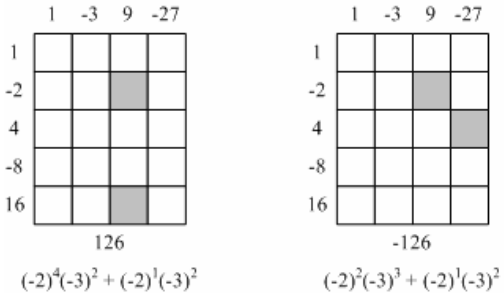

Figure 7. *Positive and negative integers can have a representation in the DNBNS. For instance, 126 and -126 contain only two digits.*

3.1 Completeness

In studying a number representation, one of the key issues is the completeness of the system (*i.e.*, every number can be represented in the system). The following theorem demonstrates that all integers can have at least one representation in the DNBNS.

Theorem 1. *All integers can have a representation in the double negative-base number system.*

Proof: In order to show the completeness of the system, the proof will be divided into two cases: non-negative and negative integers.

Case 1: the representation for non-negative integers. These can be established by introducing an algorithm for converting a non-negative integer N from the classical DBNR into the proposed representation. Let N be a non-negative integer. N can be expressed in DBNR as follows:

$$N = \sum_{i=0}^{m} d_{i,0}(2)^i(3)^0 \quad , \tag{3}$$

where i is a non-negative integer and $d_{i,0}$ is a digit in $\{0, 1\}$. The integer N can be written as the sum of two DBNRs as $N = X + Y$ where

$$X = \sum_{i=0}^{\left\lfloor \frac{m}{2} \right\rfloor} d'_{2i,0}(-2)^{2i}(3)^0 \;,\; Y = \sum_{i=1}^{\left\lceil \frac{m}{2} \right\rceil} d''_{2i-1,0}(2)^{2i-1}(3)^0 \;, \quad \textbf{(4)}$$

where

$$d'_{k,0} = \begin{cases} d_{k,0} : k = even \\ 0 : k = odd \end{cases} \qquad d''_{k,0} = \begin{cases} 0 : k = even \\ d_{k,0} : k = odd \end{cases}.$$

It is obvious that $d_{2i,0}$ and $d''_{2i-1,0}$ are either 0 or 1. The objective is to represent N in the DNBNR, that is

$$N = \sum_{i=0}^{m} \sum_{j=0}^{n} e_{i,j}(-2)^i(-3)^j \;, \quad \textbf{(5)}$$

where $e_{i,j}$ is a digit in the digit-set $\{0, 1\}$. Consider Y in (4), it is obtained that

$$Y = \sum_{i=1}^{\left\lceil \frac{m}{2} \right\rceil} d''_{2i-1,0}(2)^{2i-1}(3)^0$$

$$Y = -(-Y)$$

$$Y = ((\bar{1})(2)^1 + (1)(2)^0)(-Y)$$

$$Y = \sum_{i=1}^{\left\lceil \frac{m}{2} \right\rceil} d''_{2i-1,0}(2)^{2i}(3)^0 + \sum_{i=1}^{\left\lceil \frac{m}{2} \right\rceil} -d''_{2i-1,0}(2)^{2i-1}(3)^0 \;. \quad \textbf{(6)}$$

From (4), (6) and $N = X - (-Y)$, N can be illustrated as:

$$N = \sum_{i=0}^{\left\lfloor \frac{m}{2} \right\rfloor} d'_{2i,0}(2)^{2i}(3)^0 + \sum_{i=1}^{\left\lceil \frac{m}{2} \right\rceil} d''_{2i-1,0}(2)^{2i}(3)^0 + \sum_{i=1}^{\left\lceil \frac{m}{2} \right\rceil} -d''_{2i-1,0}(2)^{2i-1}(3)^0$$

$$N = \sum_{i=0}^{\left\lceil \frac{m}{2} \right\rceil} (d'_{2i,0} + d''_{2i-1,0})(2)^{2i}(3)^0 + \sum_{i=1}^{\left\lceil \frac{m}{2} \right\rceil} -d''_{2i-1,0}(2)^{2i-1}(3)^0$$

$$N = \sum_{i=0}^{\left\lceil \frac{m}{2} \right\rceil} (d'_{2i,0} + d''_{2i-1,0})(-2)^{2i}(3)^0 + \sum_{i=1}^{\left\lceil \frac{m}{2} \right\rceil} d''_{2i-1,0}(-2)^{2i-1}(3)^0$$

where $d'_{2\left\lceil \frac{m}{2} \right\rceil,0} = 0$ iff $\left\lceil \frac{m}{2} \right\rceil = \left\lfloor \frac{m}{2} \right\rfloor$, and $d''_{-1,0} = 0$. Since $d_{2i,0}$ and $d''_{2i-1,0}$ must be either 0 or 1, $(d_{2i,0} + d_{2i-1,0})$ must be in $\{0, 1, 2\}$. It is obvious that if the sum $(d_{2i,0} + d_{2i-1,0})$ is 0 or 1, by setting $e_{2i,0} = (d_{2i,0} + d_{2i-1,0})$ and $e_{2i-1,0} = d_{2i-1,0}$, it obtains (5). Then the representation of N is in the DNBNS.

In the case where $(d_{2i,0} + d_{2i-1,0}) = 2$, we have that $d_{2i,0} = 1$ and $d_{2i-1,0} = 1$. Using the following algebra:

$$1 \times (2)^{2i}(3)^0 + 1 \times (2)^{2i-1}(3)^0 = 1 \times (2)^{2i-1}(3)^1,$$

the representation of *N* in DNBNR is obtained by setting $e_{2i,0} = 0$, $e_{2i-1,0} = d_{2i-1,0}$ and $e_{2i-1,1} = 1$. Then it obtains (5). This completes the proof for non-negative integers.

Case 2: the representation for negative integers. Let *N* be a negative integer where *N* can be expressed as:

$$N = \sum_{i=0}^{m} - d_{i,0}(2)^i(3)^0 \quad , \tag{7}$$

where $d_{i,0}$ is a digit in $\{0, 1\}$. The rest of the proof is similar to the first case. ■

For instance, let 24 be an input *N*. In the classical DBNS, *N* can be written as $X + Y = d'_{4,0}(-2)^4(3)^0 + d''_{3,0}(2)^3(3)^0$ where $d'_{4,0} = 1$ and $d''_{3,0} = 1$.In the DNBNS, *N* can be written as $\sum_{i=0}^{m}\sum_{j=0}^{n} e_{i,j}(-2)^i(-3)^j$. In this case, *Y* is considered as:

$$Y = d''_{3,0}(2)^3(3)^0$$
$$Y = ((-2)^1(3)^0 + (-2)^0(3)^0)(-d''_{3,0}(2)^3(3)^0)$$
$$Y = d''_{3,0}(2)^4(3)^0 + (-d''_{3,0}(2)^3(3)^0)$$

Thus,

$$N = d'_{4,0}(-2)^4(3)^0 + d''_{3,0}(2)^4(3)^0 + (-d''_{3,0}(2)^3(3)^0)$$
$$N = (d'_{4,0} + d''_{3,0})((-2)^4(3)^0) + (d''_{3,0}(-2)^3(3)^0)$$

Since $d'_{4,0} + d''_{3,0} = 2$, we can use the following algebra: $(2)^4(3)^0 + (2)^3(3)^0 = (2)^3(3)^1$.

Then *N* can be represented in the DNBNS where $e_{4,0} = 0$, $e_{3,0} = d''_{3,0}$ and $e_{3,1} = 1$.

3.2 Normalization

It is obvious that a number can have more than one representation in DNBNS. The system is thus redundant. For instance, a negative integer -12 can have at least two different DNBNR as $(-2)^1 (-3)^1 + (-2)^1 (-3)^2$ and $(-2)^2 (-3)^1$. It should be note that zero can have more than one representation in our system. This can be explained by the following relation:

$$(-2)^i(-3)^j + (-2)^i(-3)^{j+1} + (-2)^{i+1}(-3)^j + (-2)^{i+2}(-3)^j = 0 \, . \tag{8}$$

In order to avoid the zero-redundant problem, we now characterize a representation called *normal* form by the following definition:

Definition 2. *A DNBNR that has no consecutive non-zero digits along with the power of three is said to be a* normal *representation.*

It is clear that any integer can have at least one normal representation. Now, we concentrate on the problem how a number can be transformed into a normal form. We propose an algebra rule called row reduction for the normalization. By the definition of normal form, the representation resulted from the row reduction is a normal form.

Algebra 1 (Row reduction): *Two consecutive non-zero digits lying in one row can be reduced as following algebra:*

$$(-2)^i(-3)^j + (-2)^i(-3)^{j+1} = (-2)^{i+1}(-3)^j \ . \qquad \textbf{(9)}$$

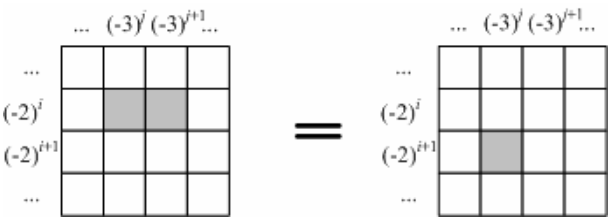

Figure 8. *The row reduction algebra is used to reduce two non-zero consecutive digits in the same column.*

From the relation (8), zero-redundant representation (*i.e.*, the representation of zero that includes some non-zero digits) must contain two consecutive non-zero digits in the same row. This can conclude that zero-redundant forms are all removed from the system.

3.3 Conversion

In the signed characteristic of DNBNS (Fig. 6.), redundancy degrees for both positive and negative integers are similarly established in the proposed system. Theorem 1 shows that the system is complete. The technique for converting DBNR to DNBNR defined in Theorem 1 can also be used to convert a binary representation to DNBNR. Remark that an obtainable positive integer outputted from the conversion technique may holds more non-zero digits comparing with the classical DBNS.

At this point, we will describe a greedy algorithm for the conversion. Given an integer x, the representation of x in DNBNR is produced using a concept of the nearest digit. The algorithm is expressed as follows:

Algorithm: Conversion

Input an integer x

Output S, the series of multiplication of power of -2 and -3 in the negative double base number representation of x

```
begin
  while (x ≠ 0) do
        Find a, the nearest of (-2)^i(-3)^j to x
        S ← S ∪ {i, j}
        x ← x – a
  enddo
end
```

Proof of the algorithm. It is evident that the output is in the DNBNR and has the same numerical value as x.

Time complexity. Technique used in the algorithm is to reduce the number by introducing the nearest digit. Since digit in DNBNR is of the form 2^i3^j, the maximum number of digits must be delimited by $\log_2 x$. The nearest digit can also be solved in a logarithmic-time technique. This concludes that the time complexity of the algorithm is $O(\log x)$.

This algorithm is clearly that does not guarantee to provide a canonical form. For instance, let $x = 15$ be the input, the greedy algorithm for DNBNS returns 15 = 16 – 2 + 1, but the canonical form is 15 = 9 + 6. Another example, let x be -15, the above method responds -15 = -12 + - 3 which is the canonical form. However, this algorithm is very easily implemented and it provides much more sparse representation.

4. Arithmetic operations for DNBNS

In researching a number system area, one of the principal considerations is some fundamental arithmetic operations, such as addition and multiplication. In this paper, we first introduce an addition and then present a multiplication for the double negative-base number system.

4.1 Addition

The addition of two numbers represented in DNBNR can be performed by overlaying two operand maps. A collision of two cells in the representation maps can be occurred. We also identify two essential algebras for solving the problem.

Algebra 2 (Collision transformation): *A collision of two non-zero digits can be transformed into two non-zero digits as follow:*

$$(-2)^i(-3)^j + (-2)^i(-3)^j = (-2)^{i+1}(-3)^j + (-2)^{i+2}(-3)^j . \qquad \textbf{(10)}$$

Algebra 3 (Collision reduction): *A collision of two non-zero digits can be reduced to zero digits as the following algebra:*

$$(-2)^i(-3)^j + (-2)^i(-3)^j + (-2)^{i+1}(-3)^j = 0 \,. \tag{11}$$

For instance, addition of two numbers in the DNBNS is illustrated by Fig. 9. The row reduction (Algebra 1) and the collision transformation (Algebra 2) are applied in order to achieve the result expressed in the normal form. It is clear that subtraction can also be processed by addition with the negative value of the adder.

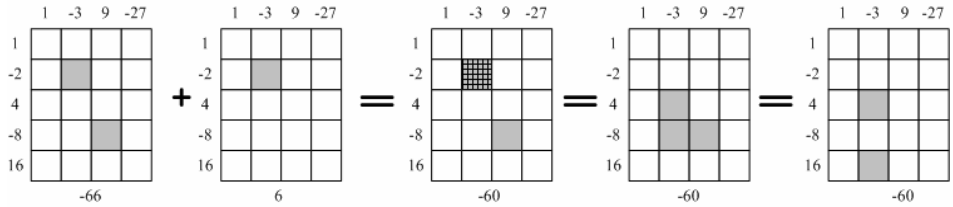

Figure 9. *Addition of -66 and 6 is performed by overlaying two representation maps. The collision transformation rule together with the row reduction are applied, the obtained result is thus in a normal form. Grid cell is a cell where collision occurs.*

4.2 Multiplication

Similar to the multiplication for DBNS, two numbers represented in DNBNR can be basically performed by 2-dimensional shift and overlaying (addition) techniques. An example of multiplication is shown in Fig. 10. Both operands and the result are in the normal form.

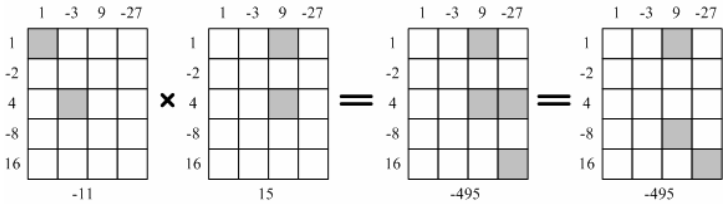

Figure 10. *Product of -11 and 15 is processed by 2-dimensional shift of one operand by the other. The row reduction rule is executed for obtaining the normal form.*

5. Conclusion

In our work, we introduce the double negative-base number representation system using bases -2 and -3. The great advantage of employing this proposed number system is that an integer can have at least one representation. We concentrate on the completeness together with the proof and we also introduce a normalization technique in order to eliminate some zero-redundant representations. Moreover, we focus on a greedy algorithm for converting binary number into DNBNR. Finally, we also

demonstrate some fundamental arithmetic operations; addition (subtraction) and multiplication along with some useful algebra rules.

References

Avizienis, A.: *Signed-digit number representations for fast parallel arithmetic.* IRE Trans. Electronic Computers, Vol. 10. (1961) 389-400

Benini, M., Nowotka, D., Pulley, C.: *Computer Arithmetic: Logic, Calculation and Rewriting.* Intl. Conf. FroCos '98 -Frontiers of Combining Systems (1998)

Berth, V., Imbert, L., Jullien, G.A.: *More on Converting Numbers to the Double-Base Number System.* Research Report LIRMM-04031 October (2004)

Dimitrov, V.S., Imbert, L., Mishra, P.K.: *Fast Elliptic Curve Point Multiplication using Double-Base Chains.* Laboratoire d'Informatique, Robotique et Microélectronique de Montpellier (2005)

Dimitrov, V.S., Jullien, G.A., Miller, W.C.: *Theory and Applications of the Double-Base Number System.* IEEE Transactions on Computers, Vol. 48. October (1999) no.10 1098-1106

Dimitrov, V.S., Jullien, G.A..: *Loading the Bases: A New Number Representation with Applications.* IEEE Circuits and Systems Magazine, Vol. 3. (2003) 6-23

Dimitrov, V.S., Sadeghi-Emamchaie, S., Jullien, G.A., Miller, W.C.: *A Near Canonic Double-Based Number System with Applications in DSP.* SPIE Conference on Signal Processing Algorithms, Vol. 2846. (1996) 14-25

Gilbert, G., Langlois, J.M.P.: *Multipath Greedy Algorithm for Canonical Representation of Numbers in the Double Base Number System.* The 3rd International IEEE-NEWCAS Conference June (2005) 39-42

Leelatham, S., Surarerks, A.: *An extended double base number system using different digit sets.* Proceedings of 10th National Computer Science and Engineering Conference (NCSEC), October (2006) 140-145

Sadeghi-Emamchaie, S., Jullien, G.A., Dimitrov V.S., Miller, W.C.: *Digital Arithmetic Using Analog Arrays.* Proc. Eighth Great Lakes Symp. VLSI, Lafayette, La., Febuary (1998) 202-207

Thienprapasith, P., Surarerks, A.: *Flexible interval representation system.* Proceeding of the 10th Annual National Symposium on Computational Science and Engineering (ANSCSE), March (2006) 327-332

Wangjitman, K., Surarerks, A.: *Addition transducer for double base number system.* The 6th IEEE International Symposium on Communications and Information Technologies 2006 (ISCIT), October (2006)