Properties of Carry Value Transformation

Suryakanta Pal,1 Sudhakar Sahoo,2 and Birendra Kumar Nayak3

1 Department of Basic Science and Humanities, Silicon Institute of Technology, Silicon Hills, Patia, Bhubaneswar 751024, India
2 Department of Computer Science, Institute of Mathematics and Applications, Andharua, Bhubaneswar 751003, India
3 P.G. Department of Mathematics, Utkal University, Bhubaneswar 751004, India

Correspondence should be addressed to Sudhakar Sahoo, sudhakar.sahoo@gmail.com

Received 28 December 2011; Accepted 20 January 2012

1. Introduction

The notion of transformation is very important in mathematics. Accordingly, in the literature, one finds many kinds of transformations with interesting properties. Carry Value Transformations (CVTs) and Modified Carry Value Transformations (MCVTs) are two challenging transformations which currently have assumed much significance because of their applications in fractal formation [1], designing new hardware circuits for arithmetic operations [2], and so forth. Similar kind of transformations such as Extreme Value Transformation (EVT) [3], 2-Variable Boolean Operation (2-VBO) [4], Integral Value Transformation (IVT) [5] are also used to manipulate strings of bits and applicable in pattern formations [3, 4], solving Round Rabin Tournaments problem [6], Collatz-like functions [5], and so forth. All these applications in diversified domain motivated us to study the mathematical properties of these kinds of transformations.

The hardware circuit for arithmetic operations as designed in [2] is based on a result that after finite number of iterations, either CVT of the two nonnegative integers is equal
to 0 or their XOR value is equal to 0. But no mathematical proof regarding this result was discussed in [2]. This important result has been proved in this paper. Section 2 provides the basic concepts of CVT, MCVT, and XOR earlier defined in [1, 2]. In Section 3, it is proved that addition of any two nonnegative integers expressed as binary numbers is the same as addition of their CVT and their XOR values. This result is also shown to be true for any base of the number system. In Section 4, it is proved that in a successive addition of CVT and XOR of any two nonnegative integers, the maximum number of iterations required to get either CVT = 0 or XOR = 0 is equal to the length of the bigger integer expressed as a binary string. Further, in the same section, it is shown that MCVT of any two nonnegative integers = 0 requires a maximum of two iterations. In Section 5, an equivalence relation is defined using the concept of CVT, and the equivalence classes obtained due to it are presented.

2. Definitions of CVT and MCVT in Binary Number System

Let “a” and “b” be decimal representations of the binary strings \((a_n, a_{n-1}, \ldots, a_1)_2\) and \((b_n, b_{n-1}, \ldots, b_1)_2\), respectively, where each \(a_i, b_i \in B_2 = \{0, 1\}\) for all \(i = 1, 2, \ldots, n\) and \(B^n_2\) be the set of all possible binary strings of length \(n\) on the set \(B_2\). In binary number system, CVT as discussed in [1] is a mapping \(\text{CVT} : B^n_2 \times B^n_2 \rightarrow B^n_2 \times \{0\}\) defined by \(\text{CVT}(a, b) = (a_n \land b_n, a_{n-1} \land b_{n-1}, \ldots, a_1 \land b_1, 0)_2\), whereas MCVT in [1] is a mapping \(\text{MCVT} : B^n_2 \times B^n_2 \rightarrow B^n_2\) defined by \(\text{MCVT}(a, b) = (a_n \land b_n, a_{n-1} \land b_{n-1}, \ldots, a_1 \land b_1)_2\). That is, to find out CVT, we perform the bit wise XOR operation of the operands to get a string of sum-bits (ignoring the carry-in while performing the addition of \(a\) and \(b\)) and simultaneously the bit wise ANDing of the operands to get a string of carry-bits, the latter string is padded with a “0” on the right is called the CVT of these operands as shown in Figure 1, and MCVT is only the ANDing values except the bit “0” padded on the right, and thus the relation between these two operation is \(\text{CVT}(a, b) = 2 \times \text{MCVT}(a, b)\).

For example, suppose we want to find out the CVT of two numbers say 23 and 27. First of all, we have to find out the binary representation of these numbers, that is, \((23)_{10} \equiv (10111)_2\) and \((27)_{10} \equiv (11011)_2\).

The carry value is computed as in Figure 2.
Theorem 3.1.

Suppose \( a \) and \( b \) are any two nonnegative integers. Thus, CVT(23,27) = CVT(10111,11011) = (1 \land 1,0 \land 1,1 \land 0,1 \land 1,1 \land 0,0)\(_2\) (100110)\(_2\) = (38), and MCVT(23,27) = (19). It may be noted that in any number system, CVT and MCVT are mapping from \( Z \times Z \) to \( Z \), where \( Z \) is set of nonnegative integers.

2.1. Extensions of CVT, MCVT, and XOR Operations for Arbitrary Number System

For any number system in base \( \beta \), CVT of any two nonnegative integers \( a = (a_n, a_{n-1}, \ldots, a_1)_\beta \) and \( b = (b_n, b_{n-1}, \ldots, b_1)_\beta \) is defined by an integer \( c = (c_n c_{n-1} \cdots c_1 0)_\beta \) where \( c_i = \begin{cases} 1, & \text{if } a_i + b_i \geq \beta \\ 0, & \text{if } a_i + b_i < \beta \end{cases} \) for \( i = 1, 2, 3, \ldots, n \). Similarly, MCVT of \( a \) and \( b \) in base \( \beta \) is the CVT value \( c = (c_1 c_2 \cdots c_n)_\beta \) except the padding bit 0 in the least significant bit position. That is CVT\((a,b) = \beta \times \text{MCVT}(a,b)\) and the definition of XOR operation in binary number system can be extended for any number system in base \( \beta \) as \( a \oplus b = ((a_n + b_n) \mod \beta, (a_{n-1} + b_{n-1}) \mod \beta, \ldots, (a_1 + b_1) \mod \beta) \), where \( + \) is the usual addition in decimal number system.

For example, in ternary number system, CVT(466,458) = CVT(122021,121222) = (110110)\(_3\) = 336, MCVT(466,458) = MCVT(122021,121222) = (11011)\(_3\) = 112, XOR(466,458) = XOR(122021,121222) = (210210)\(_3\) = 588.

3. Properties of CVT and XOR

We have observed in the last example that CVT(23,27) = 38 and XOR(23,27) = 12. Now 23 + 27 = 38 + 12, that is, 23 + 27 = CVT(23,27) + (23 \oplus 27). In general, we prove the following.

**Theorem 3.1.** \( a + b = \text{CVT}(a,b) + (a \oplus b) \), where \( a \) and \( b \) are any two nonnegative integers.

**Proof.** Suppose \( a = a_n a_{n-1} \cdots a_{k-1} a_k a_{k+1} \cdots a_2 a_1 \) and \( b = b_n b_{n-1} \cdots b_{k-1} b_k b_{k+1} \cdots b_2 b_1 \) are the binary representations of \( a \) and \( b \) both expressed using \( n \) bits. Then, \( \text{CVT}(a,b) = c_n c_{n-1} c_{n-2} \cdots c_1 0 \) for \( i = 1, 2, \ldots, n \). We will prove that sum of the contribution of \( a_k \) and \( b_k \) in \( a + b \) is the same as the sum of the contribution of \( c_k \) and \( a_k \oplus b_k \) in \( \text{CVT}(a,b) + (a \oplus b) \), where \( k = 1, 2, 3, \ldots, n \). The place values of \( a_k \) and \( b_k \) in \( a \) and \( b \) are \( a_k \times 2^{k-1} \) and \( b_k \times 2^{k-1} \), respectively. So the total contributions of both \( a_k \) and \( b_k \) in \( a + b \) is \( (a_k + b_k)2^{k-1} \). The binary variable \( a_k \) and \( b_k \) can have four choices, and their place values are shown in Table 1.

From third column and seventh column, it can be verified that the total contribution of \( a_k \) and \( b_k \) in \( a + b \) is the same as the sum of the contribution of \( c_k \) and \( a_k \oplus b_k \) in \( \text{CVT}(a,b) + (a \oplus b) \) for \( k = 1, 2, \ldots, n \). Therefore, \( a + b = \text{CVT}(a,b) + (a \oplus b) \).
4. Convergence Behavior of CVT and MCVT

4.1. Convergence of CVT

Let \( f : Z \times Z \to Z \times Z \) be defined as \( f(a, b) = (\text{CVT}(a, b), (a \oplus b)) \) for all \((a, b) \in Z \times Z\). Consider the iterative scheme \((x_{n+1}, y_{n+1}) = f(x_n, y_n), n = 0, 1, 2, 3, \ldots\). In this section, we will prove an important theorem which states that the sequence generated by the iterative scheme \((x_{n+1}, y_{n+1}) = f(x_n, y_n), n = 0, 1, 2, 3, \ldots\) converges to \((0, x_0 + y_0)\). The convergence behavior of CVT and XOR values of different order pairs are shown in Table 3.

The sequences generated from the ordered pair \((127, 65)\) in Table 3 may be interpreted as \(127 + 65 = 130 + 62 = 4 + 188 = 8 + 184 = 16 + 176 = 32 + 160 = 64 + 128 = 0 + 192\). These
Figure 3: Showing the state transition diagrams (STDs) of three order pairs $(1, 23), (1, 15)$, and $(17, 11)$ as shown in Table 3.

generated sequences are named as the orbit of the order pair $(127, 65)$. Figure 3 shows the state transition diagrams (STDs) of some of the points and their orbits.

Observations:

1. In any number system, CVT $= 0$ in any iteration $\iff$ in its previous iteration the sum of the corresponding bits of CVT and XOR is always less than the base of that number system;
2. If two numbers expressed in binary are complement to each other, then their CVT $= 0$. But the converse is not true;
3. If XOR value is 0 in any iteration, then CVT $= 0$ in the next iteration;
4. The points in a single orbit are collinear as shown in Figure 3.

According to the definition of CVT for any two $n$-bit numbers, CVT will be of at most $(n + 1)$ bits. It seems that the recursive procedure of the CVT + XOR of two nonnegative integers always increases the length of the CVT by 1 in each iteration but it is not true, which is clear from the next proof.

Lemma 4.1. If the maximum length of two nonnegative integers in binary representation is $n$ then the CVT and XOR values in each iteration expressed in binary strings must be of length at most $(n + 1)$. 
Proof. Let \( a \) and \( b \) be two nonnegative integers with length at most \( n \) in their binary representations. Let \( c \) and \( d \) be two numbers to be added in \( k \)th iteration while performing the repeated sum of CVT and XOR. Suppose the number of (valid) bits in \( \text{CVT}(c,d) \geq n + 2 \) (rejecting the zeros in the left of the first nonzero bit) in an iteration. The smallest number with valid \( (n + 2) \) bits is \( 100 \cdots 0 = 1 \times 2^{n+1} = 2^{n+1} \). So, \( \text{CVT}(c,d) \geq 2^{n+1} \Rightarrow \text{CVT}(c,d) + (c \oplus d) \geq 2^{n+1} \).

Since \( \text{CVT}(c,d) + (c \oplus d) = c + d \) (from Theorem 3.1), \( c + d \geq 2^{n+1} \). Since \( c + d = a + b \), so

\[
a + b \geq 2^{n+1}. \tag{4.1}
\]

The maximum number with \( n+2 \) bits is \( 111 \cdots 11 = 1 \times 2^{n-1} + 1 \times 2^{n-2} + \cdots + 1 \times 2^1 + 1 \times 2^0 = 1 + 2 + 4 + 8 + \cdots + 2^{n-2} + 2^{n-1}. \)

Maximum value of \( a + b \) is \( 2(1 + 2 + 4 + \cdots + 2^{n-2} + 2^{n-1}) = 2(2^n - 1)/(2 - 1) = 2^{n+1} - 2: \)

\[
\Rightarrow a + b \leq 2^{n+1} - 2. \tag{4.2}
\]

From (4.1) and (4.2), we get \( 2^{n+1} \leq a + b \leq 2^{n+1} - 2 \) which is absurd. Thus, our assumption was wrong, and hence all CVTs will be of at most \( (n + 1) \) bits in every iteration.

Same logic can be applied to XOR operation also, that is, if we write CVT in place of XOR in above proof, we also get an absurd result for XOR. Therefore, all XOR operations are of at most \( (n + 1) \) bits in every iteration. \( \square \)

**Lemma 4.2.** In any iteration if there is a “0” in CVT at \( k \)th position (counted from right), then there must be a “0” in \((k + 1)\)th position in the next iteration while forming the subsequent CVTs. The number of zeros in the CVT increases by at least one in each iteration.

**Proof.** Suppose a CVT contains 0 at \( k \)th position in any iteration. In the next iteration, this 0 will be added to either 0 or 1 of XOR value obtained in the previous iteration. When we form CVT, \((k + 1)\)th position of CVT will be either 0 \( \land 1 = 0 \) or 0 \( \land 0 = 0 \). Thus, we get a 0 in the \((k + 1)\)th position of the newly formed CVT. Thus, once a “0” appears in a CVT in any iteration, then “0” appears in all subsequent CVT’s in all subsequent iterations, but the position will be shifted by one in each iteration. By definition of CVT, one additional zero is added to the rightmost position in each iteration. So number of zero increases by at least one in a CVT in each iteration. \( \square \)

**Lemma 4.3.** If \( a \) and \( b \) are of maximum \( n \) binary bits, then the number of iterations required to get \( \text{CVT} = 0 \) is at most \( (n + 1) \).

**Proof.** By Lemma 4.1, all CVTs will be of at most \( (n + 1) \) bits in all iterations.

By Lemma 4.2, once a “0” appears in a CVT in any iteration, then this zero will appear in all the subsequent CVT’s in all subsequent iterations, but the position will be shifted by one in each iteration.

Also the number of zero in CVT increases by at least one in each iteration, the \((n+1)\) bits in CVT will be converted to \((n + 1)\) zeros in at most \((n + 1)\)-iterations. \( \square \)

**Note.** If \( a \) and \( b \) are of maximum \( n \) binary bits and Hamming distance between \( a \) and \( b \) is \( n \), then CVT = 0 in one iteration. Otherwise, if Hamming distance between two selected numbers is \( k \) for \( k < n \), then number of iterations required to get CVT = 0 is at most \((k + 2)\).
Lemma 4.4. If \( a \) and \( b \) are of maximum \( n \) binary digits and CVT = 0 in \((n + 1)\)th iteration, then XOR = 0 in the \( n \)th iteration.

Proof. Let us assume that CVT = 0 in the \((n + 1)\)th iteration and suppose XOR\( \neq 0 \) in the \( n \)th iteration. Then at least, one bit of the XOR in \( n \)th iteration must be “1”. It is sure that in the \( k \)th iteration (where \( k = 1, 2, 3, \ldots \) or \((n - 1)\)) of successive addition, XOR bit must be 1, and the corresponding carry bit must be 0 which is impossible. So our assumption was wrong. Thus, XOR = 0 in the \( n \)th iteration. Hence proved.

Combining Lemmas 4.3 and 4.4, we have proved the following theorem.

Theorem 4.5. Let \( f : Z \times Z \to Z \times Z \) be defined as \( f(a, b) = (\text{CVT}(a, b), (a \oplus b)) \). Then, the iterative scheme \((x_{n+1}, y_{n+1}) = f(x_n, y_n)\), \( n = 0, 1, 2, 3, \ldots \) converges to \((0, x_0 \oplus y_0)\) for any initial choice \((x_0, y_0) \in Z \times Z\). Further, for any nonnegative integers “\( x_0 \)” and “\( y_0 \)” (where \( x_0 \geq y_0 \)), the number of iterations required to get either CVT = 0 or XOR = 0 is at most the length of “\( x_0 \)” when expressed as a binary string.

### 4.2. Convergence of MCVT

The following theorem gives the number of iterations required for MCVT = 0.

Theorem 4.6. The procedure of calculating the MCVT and XOR values of any two nonnegative integers requires a maximum of two iterations to get their MCVT = 0.

Proof. Let \( a = a_n a_{n-1} \cdots a_1 \) and \( b = b_n b_{n-1} \cdots b_1 \) be two \( n \)-bits number. In the first iteration, we get MCVT\((a, b)\) and \( a \oplus b \).

Let \( x = \text{MCVT}(a, b) = (a_n \land b_n, a_{n-1} \land b_{n-1}, \ldots, a_1 \land b_1) \) and \( y = a \oplus b = (a_n \oplus b_n, a_{n-1} \oplus b_{n-1}, \ldots, a_1 \oplus b_1) \). Then, in the second iteration, we get MCVT\((x, y)\) and \((x \oplus y)\). We will show that MCVT\((x, y)\) = 0.

From Table 4, it can be verified that:

\[
\text{MCVT}(x, y) = ((a_n \land b_n) \land (a_{n-1} \land b_{n-1}), (a_{n-1} \land b_{n-1}) \land (a_{n-2} \land b_{n-2}), \ldots, (a_1 \land b_1) \land (a_1 \land b_1)) = (0, 0, 0, 0, \ldots, 0) = 0.
\]  
\[(4.3)\]

If \( a_i \land b_i \neq 1 \) for all \( i \), then MCVT\((a, b)\) = 0 in one iteration. Hence proved.
5. An Equivalence Relation is Defined Using the Notion of CVT

Let $A = \{0, 1, 2, 3, \ldots, 2^n - 1\}$ be a finite subset of $\mathbb{Z}$ for some nonnegative integer $n$, and let $R$ be a relation on $A \times A$ defined as $(a, b)R(c, d) \iff (a, b)$ and $(c, d)$ requiring equal number of iterations for their CVT = 0 or XOR = 0.

It can be easily verified that the relation $R$ is reflexive, symmetric, and transitive on the set $A \times A$. Therefore, $R$ is an equivalence relation on $A \times A$.

We have calculated the number of iterations required for the set of ordered pair in $A \times A$, where $A = \{0, 1, 2, \ldots, 31\}$ and constructed Figure 4 using a two-step procedure as follows.

**Step 1.** Write all the integers 0,1,2,3,…,31 in ascending order in both, uppermost row and leftmost column of Figure 4.
Step 2. Compute number of iterations required for any ordered pair \((a, b)\) to get either CVT = 0 or XOR = 0 and store it in the position \((a, b)\).

From Figure 4, we have observed that:

1. the matrix is symmetric;
2. if we consider Figure 4 as 4 quadrants, each quadrant is a symmetric matrix. Again if each quadrant is divided further into 4 smaller quadrants, then also the 1st quadrant is the same as the 3rd quadrant. Hence a self-similar fractal behaviour is noticed in Figure 4;
3. in a block of size \((2^n - 1) \times (2^n - 1)\), there are no ordered pairs in the 2nd quadrant which transform into CVT = 0 or XOR = 0 in \(n\)-iterations.

In Figure 4 \(R\) divides the set \(\{0, 1, 2, 3, \ldots, 2^n - 1\} \times \{0, 1, 2, 3, \ldots, 2^n - 1\}\) into \(n\) disjoint equivalence classes.

For \(n = 1\), there is one equivalence class \([(0,0)] = \{(0,0),(0,1),(1,0),(1,1)\}\) and \(|[0,0]| = 4\).

For \(n = 2\), there are two equivalence classes \([(0,0)], [(1,3)]\), where

\[
[(0,0)] = \{(0,0),(0,1),(0,2),(0,3),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2),(3,0),(3,3)\}, \tag{5.1}
\]

\[
[(1,3)] = \{(1,3),(2,3),(3,1),(3,2)\}. \tag{5.2}
\]

Here, \(|[0,0]| = 12, |[1,3]| = 4\).

For \(n = 3\), there are three equivalence classes \([(0,0)], [(1,3)], [1,7]\):

\[
|[0,0]| = 34, |[1,3]| = 18, |[1,7]| = 12. \tag{5.3}
\]

For \(n = 4\), there are four equivalence classes \([(0,0)], [(1,3)], [1,7], [1,15]\):

\[
|[0,0]| = 96, |[1,3]| = 78, |[1,7]| = 58, |[1,15]| = 24. \tag{5.4}
\]

For \(n = 5\), there are five equivalence classes \([(0,0)], [(1,3)], [1,7], [1,15], [1,31]\):

\[
|[0,0]| = 274, |[1,3]| = 306, |[1,7]| = 263, |[1,15]| = 133, |[1,31]| = 48. \tag{5.5}
\]

From above, we conclude that if we take a block of size \((2^n - 1) \times (2^n - 1)\), then

1. number of ordered pairs for which CVT = 0 or XOR = 0 in one iterations is \(3^n + (2^n - 1)\) for \(n = 1, 2, 3, 4, \ldots\);
2. number of ordered pairs for which CVT = 0 or XOR = 0 in \(n\) iterations is \(3 \times 2^{n-1}\) for \(n = 3, 4, 5, \ldots\).
6. Conclusion and Future Research Work

In the present paper, we have proved some important results on Carry Value Transformation (CVT) and Modified Carry Value Transformation (MCVT). Firstly, it has been proved that for any base of the number system, the sum of any two nonnegative integers is the same as the sum of their CVT and XOR values. This result is actually the correctness proof of the algorithm based on which the adder circuit is designed in [2]. Our second result, that is, “the number of iterations leading to either CVT = 0 or XOR = 0 does not exceed the maximum of the lengths of the two addenda expressed as binary strings” is about the efficiency at which the hardware circuit designed in [2] will produce the addition result. The state transition diagrams (STDs) and certain observations on CVT and MCVT are found out. Our third result such as addition of Modified Carry Value Transformation (MCVT) and XOR requires a maximum of two iterations for MCVT to be zero, is an interesting result for MCVT. A new equivalence relation is obtained on the set $\mathbb{Z} \times \mathbb{Z}$ which divides the CV Figure 4 into disjoint equivalence classes.

In future we propose to study the following aspects:

1. investigating into the state transition diagrams (STDs) of different IVTs;
2. extending the domain of CVT from nonnegative integers to real numbers and complex numbers;
3. exploring the behaviour of hybrid IVTs and their applications;
4. explaining the relationship of IVTs with cellular automata.

Acknowledgments

The authors would like to acknowledge Professor P. Pal Choudhury and Sk. Sarif Hassan, Indian Statistical Institute, Kolkata for motivating us to work further in the domain of CVT and IVT.

References


