Minimum Delay Adder for Redundant Complex Binary Number System

JAWAD SALEEM, USMAN ALI, SADAF SAEED, SHAHID A. KHAN
Department of Electrical Engineering
COMSATS Institute of Information Technology, Abbottabad (Pakistan)

Abstract: We find the applications of complex numbers in almost all fields of modern science and engineering. Due to its much application a lot of research work is underway to reduce the computation requirements for the Complex binary numbers. Recent research indicates that complex numbers can be represented in single binary-unit format as in Complex Binary Number System (CBNS) with bases (−1+j) and (−1−j). One-unit binary representation of complex numbers in CBNS form reduces computational complexity in arithmetic operations involving such numbers, although the problem of carry appears to be more pronounced in this system compared to base-2 binary system. This has led to the definition of Redundant Complex Binary Number System (RCBNS), which permits carry-free addition of complex binary numbers. This paper proposes an hardware architecture for the carry-free addition of RCBNS. The presented adder is developed directly from truth table of 3-bit block RCBNS addition.

Key-Words: - Complex Binary Number System (CBNS), Redundant Complex Binary Number System (RCBNS), Complex Numbers, Arithmetic, Addition, Subtraction

1 Introduction

We find the application of complex numbers in diverse fields of science and engineering such as in digital signal processing and image processing. The complex nature of these numbers poses complexity in dealing with arithmetic operations. In this respect the focus of recent work is the search for efficient approaches to handle situations involving complex numbers. Presently the divide and conquer technique to deal with complex arithmetic operations is in practice. Recently researchers came up with the latest innovations for reducing the computational complexities involved in arithmetic operations one is obtained by defining binary numbers with bases other than 2, which facilitate one unit representation of complex numbers, which include work by knuth[1], penny[2] and stepaneko[3]. Jamil *et al*[4] have presented complete analysis of (-1+j)-base complex binary number system and elaborated how arithmetical, operations can be accomplished. Another interesting approach is by Zaini *et al*[5] which permits carry free arithmetic operator for complex numbers. These two techniques have largely reduced the complexities we encounter in arithmetic operations. The beauty of these techniques lies in the simplicity they offer in arithmetic operations and it is proved that the arithmetical operations involving complex numbers by the use of these techniques are no more complex. RCBNS is getting preference over CBNS due to carry-free addition, simple multiplication and direct conversion method between base-2 and base (-1+j), whereas CBNS does not provide these advantages.

2 (–1+j)-Base Complex Binary Number System (CBNS)

The value of an n-bit binary number with base (–1+j) can be written in the form of a power series as follows: $a_{n-1}(-1+j)^{n-1}+a_{n-2}(-1+j)^{n-2}+...+a_1(-1+j)^1+a_0(-1+j)^0$ where the coefficients $a_{n-1},a_{n-2},a_{n-3},\dots,a_2,a_1,a_0$ are binary (either 0 or 1). This is analogous to the ordinary binary number system power series of: $a_{n-1}(2)^{n-1}+a_{n-2}(2)^{n-2}+\dots+a_1(2)^1+a_0(2)^0$ except that the bases are different. Using the algorithms, given in Ref.[5], we are able to represent a given complex number as single complex binary number:

Let's first begin with the case of positive integers *N*. To represent *N* in the complex binary number system, we follow these steps:

1) Express *N* in terms of powers of 4 using division process.
2) Now convert the base 4 number $(..., q_5, q_4, q_3, q_2, q_1,q_0)$ to base –4 by replacing each digit in odd location $(q_1, q_3, q_5,...)$ with its negative to get $(..., -q_5, q_4,-q_3, q_2, -q_1, q_0)$.
3) Normalize the new number (i.e., get each digit in the range 0 to 3) by repeatedly adding four to the negative digits and adding a one to the digit on its left. If the digit is 4, replace it by a zero and subtract a one from the digit on its left. To represent the given number in the base –1+j, we replace each digit in base –4 representation with

the corresponding four bit sequence (0 →0000; 1→0001 ; 2 →1100 ; 3→1101).

Thus,
$2000_{base10} = 0001110000000001000100000000_{base\,-1+j}$

To convert a negative integer into (–1+j)-base representation, we simply multiply the representation of the corresponding positive integer with 11101 (equivalent to $-1_{base\,-1+j}$) according to the multiplication algorithm. Thus
-2000_{base10} = 1100 0000 0000 1101 0000 $0000_{base\,-1+j}$

To represent a positive or negative imaginary number in (–1+j)-base representation, we multiply the corresponding complex binary representation of the positive or negative integer with 11 (equivalent to j_{base10}) or 111 (equivalent to $-j_{base10}$), as appropriate.

Conversion algorithms for fractions and floating-point numbers are described in detail in Ref. [6], which leads to the conclusion that any type of number can be represented in CBNS.

Having known the conversion algorithms, the binary representation for any given complex number can be easily obtained, as shown by the following example:

2004 + j2004
= $1110100000001110111001100000_{base(-1+j)}$
This can be verified by computing the power series $(-1+j)^{27} + (-1+j)^{26} + (-1+j)^{25} + (-1+j)^{23} + (-1+j)^{15} + (-1+j)^{14} + (-1+j)^{13} + (-1+j)^{11} + (-1+j)^{10} + (-1+j)^{9} + (-1+j)^{6} + (-1+j)^{5}$ = 2004 + j2004

2.1 CBNS Addition

The binary addition of two complex binary numbers follows these rules: 0 + 0 = 0;0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 1100. If two numbers with 1s in position n are added, this will result in 1s in positions n+3 and n+2 and 0s in positions n+1 and n in the sum. Furthermore, 11 + 111 = 0 [Zero Rule]. Designs and implementations of nibble-size minimum-delay and ripple-carry adders have been presented in Ref. [7] while a size-free adder design has been presented in Ref.[8].

3 Redundant Complex Binary Number System (RCBNS)

RCBNS is a positional number system that has a complex radix and uses a digit set {–α to +α} that allows for carry-free additions[9]. α is restricted to [(r – 1)/2] ≤ α ≤ (r – 1) where r = 4. Using radix of (–1+j) and a digit set {–3, –2, –1, 0, +1, +2, +3}⇒ α = 3, yields the value of X = $(x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$ given by the expression:

$$X = \sum_{i=0}^{n-1} x_i(-1+j)^i \quad \ldots (1)$$

Where x_i is in the range {–3 to +3}.

Using Equation (1), the RCBNS representations of some complex numbers are given in Table 1.

Table 1: RCBNS representations for some complex numbers

No	$(-1+j)^3$	$(-1+j)^2$	$(-1+j)$	$(-1+j)^0$
	2 + j2	- j2	-1 + j	1
-3 –j3	-1	1	1	0
-2 –j3	0	1	-1	-3
-1-j3	0	1	-1	-2
0-j3	0	1	-1	-1
1-j3	0	1	-1	0
2-j3	0	1	-1	1
3-j3	0	1	-1	2
-3 +j3	0	-1	1	-2
-2 +j3	0	-1	1	-1
-1+j3	0	-1	1	0
0+j3	0	-1	1	1
1+j3	0	-1	1	2
2+j3	1	-1	-1	-1
3+j3	1	-1	-1	0
-3 –j2	0	1	0	-3
-2 –j2	0	1	0	-2
-1-j2	0	1	0	-1
0-j2	0	1	0	0
1-j2	0	1	0	1
2-j2	0	1	0	2
3-j2	0	1	0	3

Conversion of a complex number to the RCBNS form is achieved by converting the complex number into the binary form for the real and imaginary parts and then by using the following steps [9]:

1) Check the sign of both the real and imaginary parts. If each is positive or negative, then for the positive numbers place a 0 in front of each of 2-bit digit (e.g. for 9 =10 01, then **0**10 and **0**01), and similarly for the negative numbers place 1 in front of each of 2-bit digit (e.g. for –9 = -10 01, then **1**10 and **1**01).
2) Combine the right three bits of the imaginary part to the right three bits of the real part. Repeat the same for the next three bits to form rows named as q_i(where i=0,1,2,3….).(see example below)
3) Find the equivalent value for combined group of three bits from real and imaginary part in each row

q_i (where i=0,1,2,3….) from Table 2. There are four tables for the combinations of the sign of the real part and the sign of the imaginary part (+/+, +/–, –/+, –/–). Table 2 shows only the positive sign for both the imaginary and real parts.

4) Label a group of 4 bits in the first row as q_0. To generate the successive rows, multiply the current row by –4 and so on. This results in the change of sign and magnitude from one row to the next. Therefore, even rows have positive and odd rows have negative signs (... $+q_5, -q_4, +q_3, -q_2, +q_1, -q_0$).

Table 2: Table of conversion from Binary form to RCBNS form in +/+ case

(R + J) $base_{10}$						RCBNS (-1+j),3											
J			R			2+j2			-j2			-1+j			1		
						S	0-3		S	0-3		S	0-3		S	0-3	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0
0	0	1	0	1	0	0	0	1	0	0	0	1	0	1	1	0	1
0	0	1	0	1	1	0	0	1	0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1
0	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0
0	1	0	0	1	1	0	0	0	1	0	1	0	0	0	0	1	1
0	1	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1
0	1	1	0	0	1	0	0	0	1	0	1	0	0	1	0	1	0
0	1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	0	1
0	1	1	0	1	1	0	0	1	1	0	1	1	0	1	0	0	0

The example below illustrates the conversion of a complex binary number to the RCBNS form.

Example: Convert complex number 6+7j to RCBNS form

Step 1: Express numbers in the binary form for imaginary and real parts.

Part 1 (J) $\Rightarrow 7 = 01\ 11 \Rightarrow 7 = 001\ 011$

Part 2 (R) $\Rightarrow 6 = 01\ 10 \Rightarrow 6 = 001\ 010$

Step 2: Get 3 bits of each part and combine them together starting with LSB as 3 bits of J and put them together with 3 bits of R to form rows.

q0 $\Rightarrow$ 011 010

q1 $\Rightarrow$ 001 001

Step 3: Find the equivalent values for each group of R and J from Table 2.

q0 $\Rightarrow$ 011 010 -> 001 101 101 101 $\Rightarrow$ 1 –1–1–1

q1 $\Rightarrow$ 001 001 -> 000 000 001 010 $\Rightarrow$ 0 0 1 2

Step 4: Multiply the positive sign to even rows and the negative sign to the odd rows.

q0 $\Rightarrow$ 1–1–1–1

q1 $\Rightarrow$ 0 0 –1 -2

Finally combine them again in the following order.

(…q5 , q 4, q 3, q 2, q 1, q 0).

So complex number 6+7j is represented in RCBNS as (0 0 -1 –2 1 -1 -1 -1).

3.1 RCBNS Addition

Complex numbers expressed in the RCBNS form can be used as operands to perform arithmetic operations. The results of the operation will be in the RCBNS form. An example of the addition of two complex numbers, (4+j9) and (18+j25) expressed in the RCBNS form is illustrated below.

First, the two complex numbers (4+j9) and (18+j25) are converted into the RCBNS form. Then the addition operation is performed. The complex number (4+j9) is equivalent to (0 1 0 –1 0 0 1 1) in the RCBNS form, and similarly the complex number (18+j25) is equivalent to (0 0 1 2 0 1 0 0 1 0 –1 –1).

```
X       0  0  1  2  0  1  0  0  1  0  -1  -1
Y       0  0  0  0  0  1  0 -1  0  0   1   1
===============================
S1      0  0  1  2  0  2  0 -1 1  0   0   0
```

The addition above is carry-free.

In some cases the result may have numbers ranging from –6 to 6; in such cases normalization is necessary. The process of normalization is as follows:

• Normalize the intermediate result to be in the set of {–3, –2, –1, 0, 1, 2, 3}.

• Get the final result S = Normalization + Carry, or $S=S_1$, if S_1 in the range of –3 to 3.

4 Minimum Delay Adder

Let us suppose that we have to add A and B RCBNs. Let A is represented by as

$$A=A_0\ A_1\ A_2\ A_3 \ldots A_n$$

And B be represented as

$$B=B_0\ B_1\ B_2\ B_3 \ldots . B_n$$

Each An and Bn is 3-bit number. For example in the example presented in the previous section

X=1 0 –1 –1 0 1 0 0 1 0 –1 0

Here X is arranged in the following order

$X=X_0\ X_1\ X_2\ X_3\ X_4\ X_5\ X_6\ X_7\ X_8\ X_9\ X_{10}\ X_{11}$

Here each X_i (where i=0,1,2….12) is each 3-bit.

Our design would add each B_n and A_n separately using the minimum delay adder and would utilize serial architecture to add all other A_n and B_n.

The design of a 3-bit minimum-delay CBNS adder involves the following steps:

(i) Generation of a truth table with two 3-bit operands --- operand A with $a_0a_1a_2$ bits and Operand B with $b_0b_1b_2$ bits --- addition of these two operands produces three outputs which are labeled as $c_0c_1c_2$.

The truth table has a total of $2^6 = 64$ minterms as shown in Table 3.

(ii) We'll use a 6x64 decoder to implement this truth table. For this purpose, we express each output in sum-of-minterms form as shown in Table 4.

(iii) Finally, these expressions are implemented using the decoder and OR gates as shown in Figure 2.

Table 3: Truth Table e Minimum-Delay RCBN Adder

	a_0	a_1	a_2	b_0	b_1	b_2	c_0	c_1	c_2
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	1
2	0	0	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0	1	1
4	0	0	0	1	0	0	0	0	0
5	0	0	0	1	0	1	1	0	1
6	0	0	0	1	1	0	1	1	0
7	0	0	0	1	1	1	1	1	1
8	0	0	1	0	0	0	0	0	1
9	0	0	1	0	0	1	0	1	0
10	0	0	1	0	1	0	0	1	1
11	0	0	1	0	1	1	N	O	R
12	0	0	1	1	0	0	0	0	1
13	0	0	1	1	0	1	0	0	0
14	0	0	1	1	1	0	1	0	1
15	0	0	1	1	1	1	1	1	0
16	0	1	0	0	0	0	0	1	0
17	0	1	0	0	0	1	0	1	1
18	0	1	0	0	1	0	N	O	R
19	0	1	0	0	1	1	N	O	R
20	0	1	0	1	0	0	0	1	0
21	0	1	0	1	0	1	0	0	1
22	0	1	0	1	1	0	0	0	0
23	0	1	0	1	1	1	1	0	1
24	0	1	1	0	0	0	0	1	1
25	0	1	1	0	0	1	N	O	R
26	0	1	1	0	1	0	N	O	R
27	0	1	1	0	1	1	N	O	R
28	0	1	1	1	0	0	0	1	1
29	0	1	1	1	0	1	0	1	0
30	0	1	1	1	1	0	0	0	1
31	0	1	1	1	1	1	0	0	0
32	1	0	0	0	0	0	0	0	0
33	1	0	0	0	0	1	0	0	1
34	1	0	0	0	1	0	0	1	0
35	1	0	0	0	1	1	0	1	1
36	1	0	0	1	0	0	0	0	0
37	1	0	0	1	0	1	1	0	1
38	1	0	0	1	1	0	1	1	0
39	1	0	0	1	1	1	1	1	1
40	1	0	1	0	0	0	1	0	1
41	1	0	1	0	0	1	0	0	0
42	1	0	1	0	1	0	0	0	1
43	1	0	1	0	1	1	0	1	0
44	1	0	1	1	0	0	1	0	1
45	1	0	1	1	0	1	1	1	0
46	1	0	1	1	1	0	1	1	1
47	1	0	1	1	1	1	N	O	R
48	1	1	0	0	0	0	1	1	0
49	1	1	0	0	0	1	1	0	1
50	1	1	0	0	1	0	0	0	0
51	1	1	0	0	1	1	0	0	1
52	1	1	0	1	0	0	1	1	0
53	1	1	0	1	0	1	1	1	1
54	1	1	0	1	1	0	N	O	R
55	1	1	0	1	1	1	N	O	R
56	1	1	1	0	0	0	1	1	1
57	1	1	1	0	0	1	1	1	0
58	1	1	1	0	1	0	1	0	1
59	1	1	1	0	1	1	0	0	0
60	1	1	1	1	0	0	1	1	1
61	1	1	1	1	0	1	N	O	R
62	1	1	1	1	1	0	N	O	R
63	1	1	1	1	1	1	N	O	R

5 Functional design of Adder

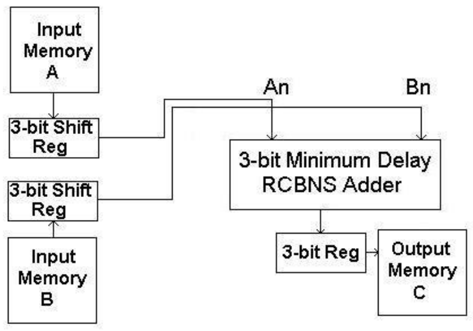

Figure 1: Function Diagram of RCBNS Adder

Functional diagram for RCBNS adder is shown in figure 1. Our design would take input 3 bit numbers A_n and B_n from memory A and memory B respectively into 3-bit shift register and pass these numbers to 3-bit minimum delay RCBNS adder. The final results after addition are passed on to the 3-bit register and then finally stored in memory C.

6 Conclusion & Future Work

Both CBNS and RCBNS hold promising future because of their ease in representing complex numbers as one unit and thus simplifying arithmetic operations involving such numbers. We have reviewed RCBNS and their addition and proposed hardware architecture for RCBNs adder. We have designed a minimum delay adder for RCBNs, which is implemented directly from the truth table and use only arrays of OR and AND gates. Future of RCBN adder is very prominent, efficient adder if developed would make RCBNS very important in the field of computations

References:

[1] D. Knuth: "An Imaginary Number System", Communications of the ACM, pp. 245-247,1960.

[2] W. Penney:, "A Binary System for Complex Numbers", Journal of the ACM, pp. 247-248, April 1965.

[3] V. Stepanenko: "Computer Arithmetic of Complex Numbers", Cybernetics and System Analysis, Vol. 32, No. 4, pp. 585-591, 1996.

[4] T. Jamil, N. Holmes, and D. Blest: "Towards Implementation of a Binary Number System for Complex Numbers", Proceedings of the IEEE SoutheastCon 2000, Nashville, Tennessee (USA), pp. 268-274, April 2000.

[5] H. Zaini and R. G. Deshmukh "Complex Number Representation in RCBNS Form for Arithmetic Operations and Conversion of the Result into Standard Binary Form" Journal of Systemics, Cybernetics and Informatics, Vol.2, No.1, 2003.

[6] T. Jamil: "The Complex Binary Number System – Basic Arithmetic Made Simple", IEEE Potentials, Vol. 20, No. 5, pp. 39-41, December/January 2002.

[7] T. Jamil, B. Arafeh, and A. Al Habsi: "Hardware Implementation and Performance Evaluation of Complex Binary Adder Designs", Proceedings of the 7th World Multiconference on Systemics Cybernetics, and Informatics (SCI 2003), Orlando, Florida (USA), Vol. II, pp. 68-73, July 2003.

[8] J. Goode, T. Jamil, and D. Callahan: "A Simple Circuit for Adding Complex Numbers," WSEAS Transactions on Information Science and Applications, Vol. 1, No. 1, pp. 61-66, July 2004.

[9] T. Jamil , A. A. Abdulghani, and A. Al-Maashari "Design of a Nibble-Size Subtractor for (-1+j)-Base Complex Binary Numbers", WSEAS Transactions on Circuits and Systems, Vol. 3, No. 5, pp. 1067-1072, July 2004.

Table 4: Sum-of-Minterms expressions for Outputs of a 3-bit Minimum-Delay Adder

$c_0=\Sigma$ (5,6,7,14,15,23,37,38,39,40,44,45,46,48,49,52,53,56,57,58,60)

$c_1=\Sigma$ (2,3,6,7,9,10,15,16,17,20,24,28,29,34,35,38,39,43,45,46,48,52,53,56,57,60)

$c_2=\Sigma$ (1,3,5,7,8,10,12,14,17,21,23,24,28,30,33,35,37,39,40,42,44,46,49,51,53,56,58,60)

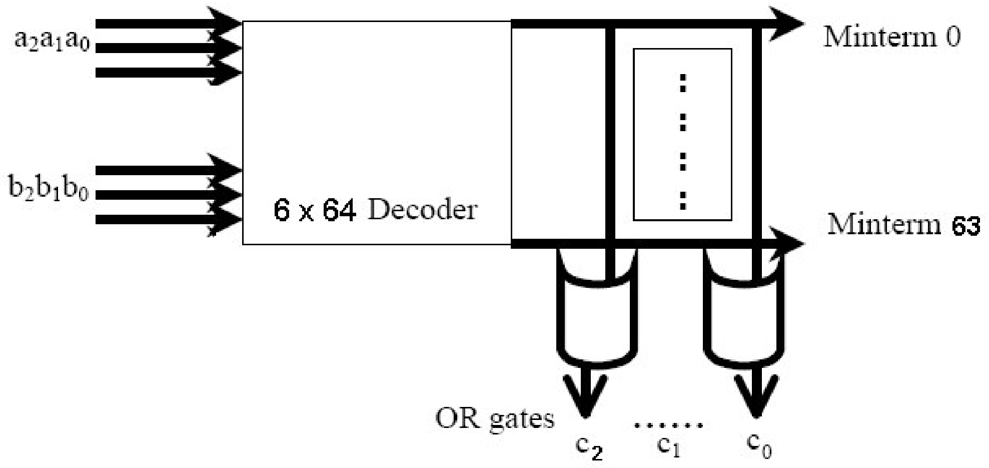

Figure 2: Block Diagram of a 3-bit Minimum-Delay RCBN Adder using Decoder