International Journal of Electrical and Computer Engineering (IJECE)
Vol.4, No.2, April 2014, pp. 295~302
ISSN: 2088-8708

Analysis of Multiple-Bit Shift-Left Operations on Complex Numbers in (−1+j)-Base Binary Format

Tariq Jamil* and Usman Ali**
*Department of Electrical and Computer Engineering, Sultan Qaboos University, OMAN
**Laboratory of Signals and Systems, University of Paris, FRANCE

Article Info

Article history:

Received Dec 10, 2013
Revised Feb 7, 2014
Accepted Mar 2, 2014

Keyword:

Complex number
Complex binary
Binary number
Multiple shift
Shift left

ABSTRACT

Complex numbers find various applications in the field of engineering. To avoid excessive delays in production of results obtained by implementing divide-and-conquer technique in dealing with arithmetic operations involving this type of numbers in today's computer systems, Complex Binary Number System with base(−1+j) has been proposed in scientific literature. In this paper, we have investigated the effects of bit-wise shift left operations (from 1-8 bits) on the complex binary representation of complex numbers and analyzed these results using mathematical equations.

Corresponding Author:

Tariq Jamil,
Department of Electrical and Computer Engineering,
College of Engineering – P.O. Box 33,
Sultan Qaboos University, AlKhod 123, Muscat, OMAN
Email: tjamil@squ.edu.om

1. INTRODUCTION

Complex numbers find various applications in the field of engineering. In today's computer systems, addition of two complex numbers $(a + jb)$ and $(c + jd)$ involves two individual additions, one for the real parts $(a + c)$ and one for the imaginary parts $(b + d)$. Similarly, the subtraction of these two complex numbers involves two individual subtractions, one for the real parts $(a - c)$ and one for the imaginary parts $(b - d)$. Multiplication involves four individual multiplications ac, ad, bc, bd, one subtraction $j^2bd = -bd$, and one addition $ad + bc$. And finally, division involves six individual multiplications ac, ad, bc, bd, c^2, d^2, two additions $ad + bc$ and $c^2 + d^2$, one subtraction $bc - ad$, and then two individual divisions $\frac{ac+bd}{c^2+d^2}$ and $\frac{bc-ad}{c^2+d^2}$. These sub-operations within an operation dealing with complex numbers increase the delay in the calculation of overall result of the operation and hence degrades the performance of the computer system. The quest to find a better concise method for representing complex numbers has yielded the formulation of a unique number system referred to as Complex Binary Number System (CBNS) [1]. Extensive details of the arithmetic algorithms to be followed in CBNS for binary representations and operations of complex numbers can be found in [1,2,3] and the hardware designs and implementations of the arithmetic circuits based on this number system can be found in [4,5,6,7]. In this paper, we have presented the effects of multiple-bit (from 1 bit to 8 bits) shift-left operations on a complex number represented in CBNS.

This paper is organized as follows: In section 2, we'll present basic information about CBNS and how to represent an integer-only complex number into this new number system. Then we'll take the CBNS representation of complex numbers and, in section 3, give a comprehensive analysis of the effects of multiple-bit shift left operations on the complex numbers. Conclusion is presented in section 4, which is followed by acknowledgements and references.

2. $(-1+j)$-BASE COMPLEX BINARY NUMBER SYSTEM

Mathematically, the value of an n-bit binary number with base $(-1+j)$ can be written in the form of a power series as $a_{n-1}(-1+j)^{n-1} + a_{n-2}(-1+j)^{n-2} + a_{n-3}(-1+j)^{n-3} + \ldots + a_2(-1+j)^2 + a_1(-1+j)^1 + a_0\ (-1+j)^0$ where the co-effecients $a_{n-1}, a_{n-2}, a_{n-3}, \ldots, a_2, a_1, a_0$ are binary in nature (0 or 1) and belong to complex binary number system. Using the conversion algorithms given in [1], we are able to obtain a $base(-1+j)$ binary representation of any given complex number (whether it is made from integers, fractions, or floating point numbers) in Complex Binary Number System (CBNS).

For example, as shown in [1],

$+2012_{10}$
$= 1110000000001110000010000_{Base\ (-1+j)}$

-2012_{10}
$= 110000000000110111010000_{Base\ (-1+j)}$

$+j2012_{10}$
$= 1000000000010000110000_{Base\ (-1+j)}$

$-j2012_{10}$
$= 1110100000001110100011100000_{Base\ (-1+j)}$

$2012_{10}+j2012_{10} = 1110100000001110100011100000_{Base\ (-1+j)}$

3. MULTIPLE-BIT SHIFT-LEFT OPERATIONS IN CBNS

To analyze the effects of shift-left operations on a complex number represented in CBNS format, a computer program in C++ language was written which allowed for auto-variations in magnitude and sign of both real and imaginary components of a complex number in a linear fashion, and then decomposed the complex binary number after the shift-left operation into its real and imaginary components. The length of the original binary bit array was limited to 800 bits and 0s were padded on the left-side of the binary data when the given complex number required less than maximum allowable bits for representation in CBNS format.

To better understand these restrictions, let's consider the following complex number:
Original complex number represented in CBNS before padding:

$90_{10}+j90_{10} = 110100010001000_{Base\ (-1+j)}$

Padded complex binary array such that the total size of the array is 800 bits.

$90_{10}+j90_{10} = 0 \ldots 0110100010001000_{Base\ (-1+j)}$

Shifting this binary array by 1-bit to the left will yield $0 \ldots 01101000100010000_{Base\ (-1+j)}$ ensuring that total array-size remains 800 bits. This was done by removing one 0 from the left-side and inserting one 0 on the right-side of the number.

Similarly, shifting of the original binary array by 2,3,4,5,6,7,8-bits to the left will yield respectively:

$0 \ldots 01101000100000_{Base\ (-1+j)}$
$0 \ldots 011010001000000_{Base\ (-1+j)}$
$0 \ldots 0110100010000000_{Base\ (-1+j)}$
$0 \ldots 01101000100000000_{Base\ (-1+j)}$
$0 \ldots 011010001000000000_{Base\ (-1+j)}$
$0 \ldots 0110100010000000000_{Base\ (-1+j)}$
$0 \ldots 01101000100000000000_{Base\ (-1+j)}$

Table 1 presents an overall summary of the effect on the signs of the complex numbers, represented in CBNS format, because of multiple-bit shift-left operations (1 to 8 bits).

Shift-left operations on complex binary numbers affect not only the signs of the given complex numbers (as shown in Table 1) but also have impact on the magnitudes of the complex numbers according to various mathematical relationships. To find out the effects of shift-left operations on the magnitudes of the complex numbers, we varied the magnitude of real and imaginary components of the original complex numbers in a linear fashion (Figure 1). The complex numbers obtained after shift-left operations were analyzed by obtaining mathematical equations describing their behavior, as given in Figures. 2-9.

To fully understand the variations in the sign and magnitude of the complex numbers before and after the shift-left operation, we used Microsoft Excel to draw graphs as shown in the Figures. 10-17.

Table 1. Effect on signs of complex numbers in CBNS format after shift-left operations

Before Shift-Left		**After Shift-Left by 1-bit**		**After Shift-Left by 2-bits**		**After Shift-Left by 3-bits**		**After Shift-Left by 4-bits**	
Real	Imaginary	Real	Imaginary	Real	Imaginary	Real	Imaginary	Real	Imaginary
+	**0**	–	+	**0**	–	+	+	–	**0**
–	**0**	+	–	**0**	+	–	–	+	**0**
0	+	–	–	+	**0**	–	+	**0**	–
0	–	+	+	–	**0**	+	–	**0**	+
+	+	–	**0**	+	–	**0**	+	–	–
+	–	**0**	+	–	–	+	**0**	–	+
–	+	**0**	–	+	+	–	**0**	+	–
–	–	+	**0**	–	+	**0**	–	+	+
Before Shift-Left		After Shift-Left by 5-bits		After Shift-Left by 6-bits		**After Shift-Left by 7-bits**		**After Shift-Left by 8-bits**	
Real	Imaginary	Real	Imaginary	Real	Imaginary	Real	Imaginary	Real	Imaginary
+	**0**	+	–	+	+	–	–	+	–
–	**0**	–	+	–	–	+	+	–	+
0	+	+	+	–	+	+	–	+	+
0	–	–	–	+	–	–	+	–	–
+	+	+	–	–	+	–	–	+	+
+	–	–	–	+	+	–	+	+	–
–	+	+	+	–	–	+	–	–	+
–	–	–	+	+	–	+	+	–	–

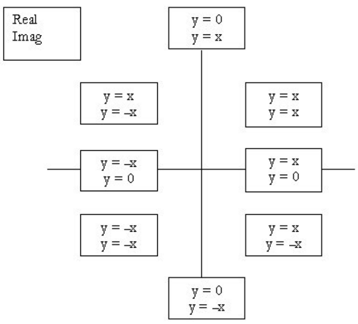

Figure 1. Before shift-left

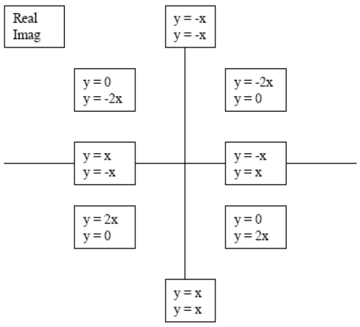

Figure 2. After shift-left by 1-bit

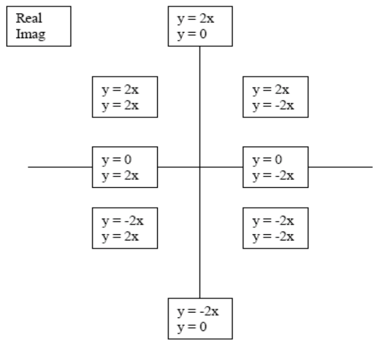

Figure3. After shift-left by 2-bits

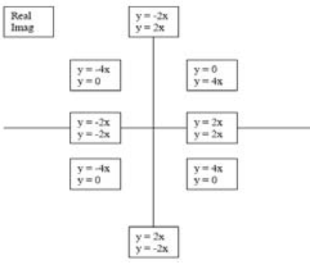

Figure 4. After shift-left by 3-bits

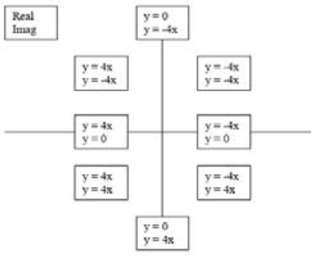

Figure 5. After shift-left by 4-bits

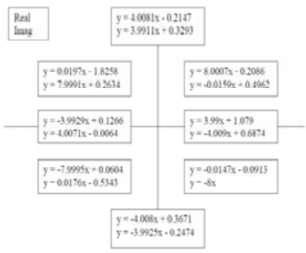

Figure 6. After shift-left by 5-bits)

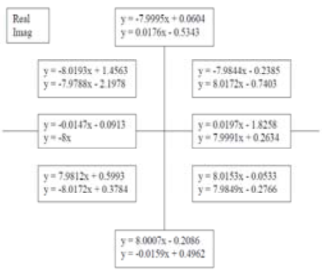

Figure 7. After shift-left by 6-bits

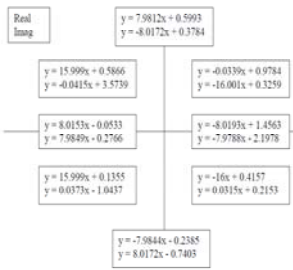

Figure 8. After shift-left by 7-bits

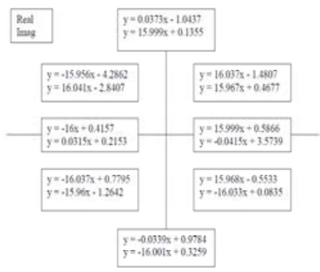

Figure 9. After shift-left by 8-bits

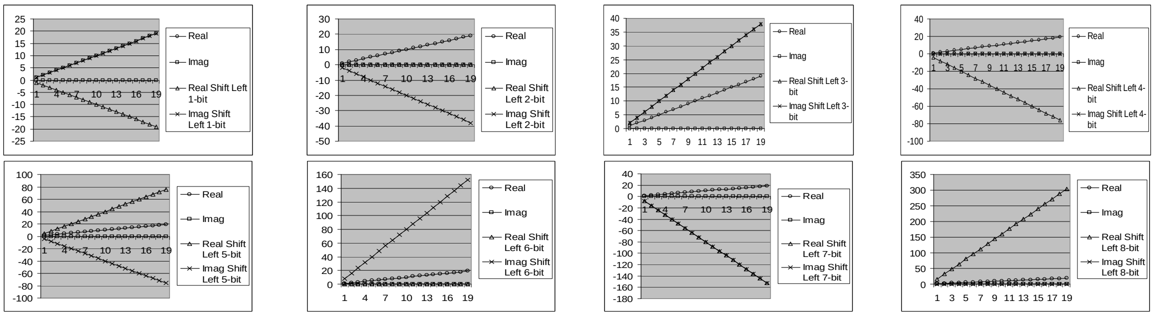

Figure 10. Effects of shift-left operation on sign and magnitude of a positive real-only complex number (1-8 bits)

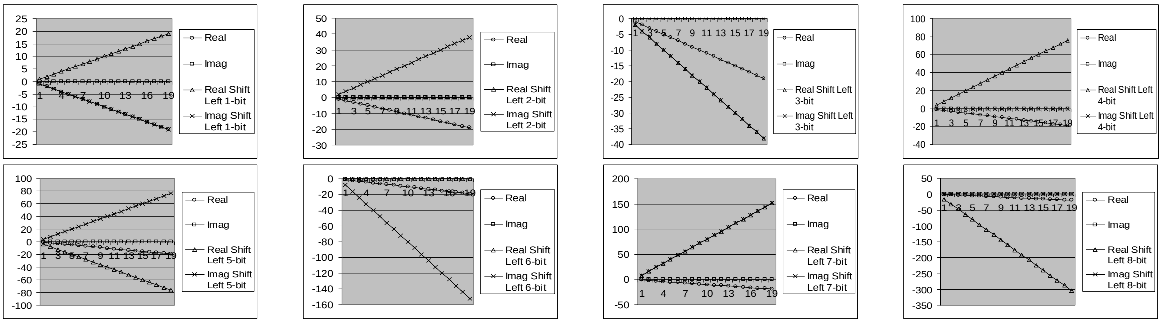

Figure 11. Effects of shift-left operation on sign and magnitude of a negative real-only complex number (1-8 bits)

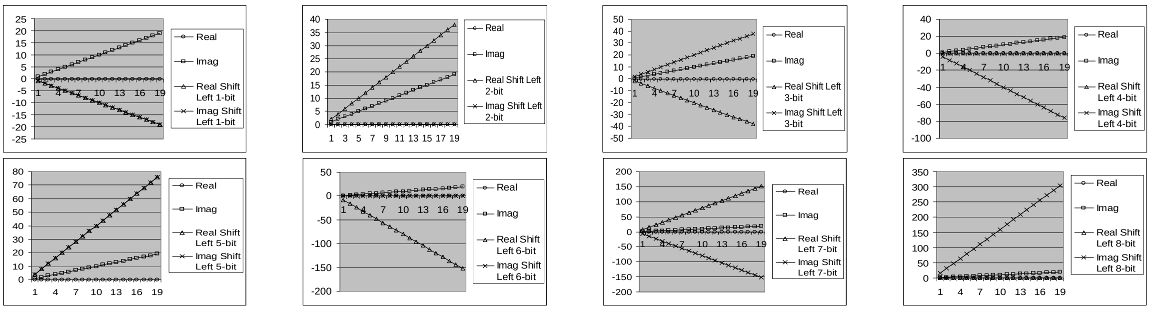

Figure 12. Effects of shift-left operation on sign and magnitude of a positive imaginary-only complex number (1-8 bits)

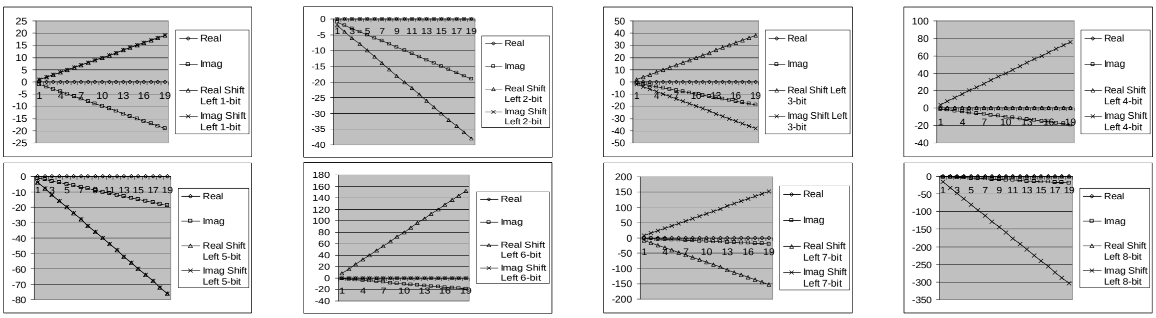

Figure 13. Effects of shift-left operation on sign and magnitude of a negative imaginary-only complex number (1-8 bits)

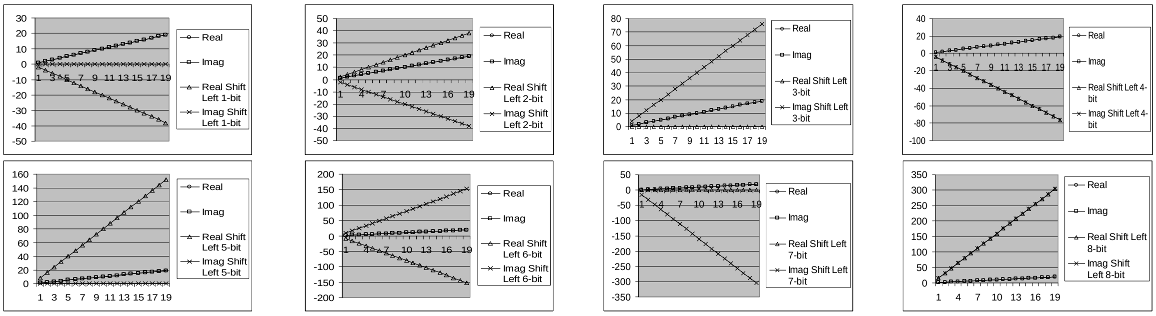

Figure 14. Effects of shift-left operation on sign and magnitude of a +Real+Imaginary complex number (1-8 bits)

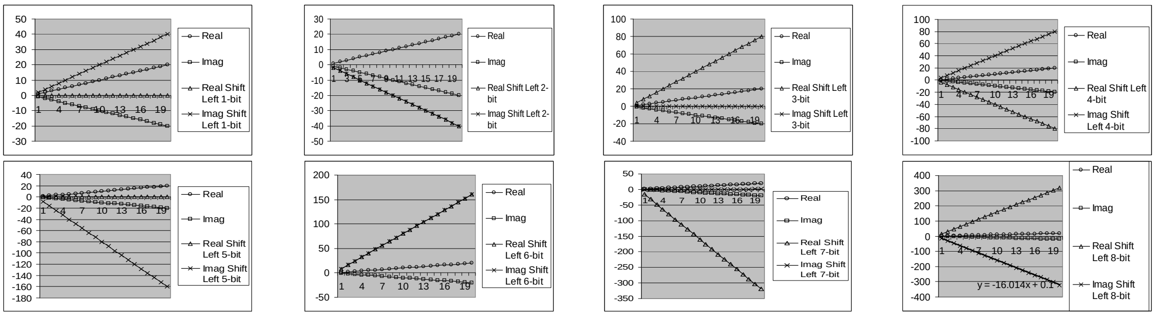

Figure 15. Effects of shift-left operation on sign and magnitude of a +Real–Imaginary complex number (1-8 bits)

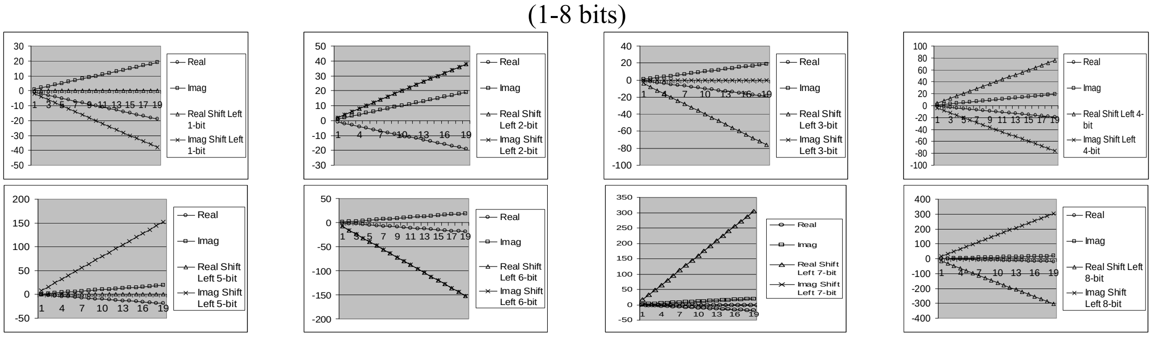

Figure 16. Effects of shift-left operation on sign and magnitude of a –Real+Imaginary complex number (1-8 bits)

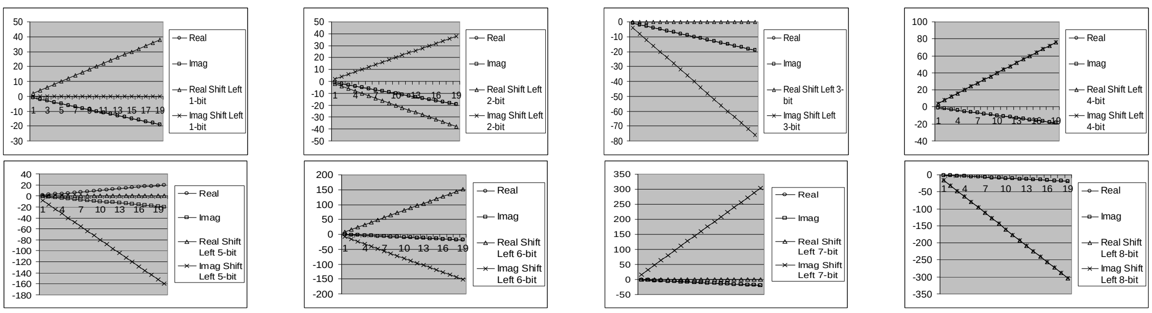

Figure 17. Effects of shift-left operation on sign and magnitude of a –Real–Imaginary complex number (1-8 bits)

After analyzing Figures. 1-17, we are able to obtain the characteristic equations describing complex numbers in CBNS format after shift-left operations. These equations are given in Table 2.

Table 2. Characteristic equations describing complex numbers in CBNS format after shift-left operations

Before Shift-Left		**After Shift-Left by 1-bit**		**After Shift-Left by 2-bits**	
$Real_{old}$	$Imaginary_{old}$	$Real_{new}$	$Imaginary_{new}$	$Real_{new}$	$Imaginary_{new}$
+	**0**	$-Real_{old}$	$+Real_{old}$	0	$-2Real_{old}$
–	**0**	$-Real_{old}$	$+Real_{old}$	0	$-2Real_{old}$
0	+	$-Imag_{old}$	$-Imag_{old}$	$+2Imag_{old}$	0
0	–	$-Imag_{old}$	$-Imag_{old}$	$+2Imag_{old}$	0
+	+	$-2Real_{old}$	0	$+2Real_{old}$	$-2Imag_{old}$
+	–	0	$-2Imag_{old}$	$-2Real_{old}$	$+2Imag_{old}$
–	+	0	$-2Imag_{old}$	$-2Real_{old}$	$+2Imag_{old}$
–	–	$-2Real_{old}$	0	$+2Real_{old}$	$-2Imag_{old}$
Before Shift-Left		**After Shift-Left by 3-bits**		**After Shift-Left by 4-bits**	
$Real_{old}$	$Imaginary_{old}$	$Real_{new}$	$Imaginary_{new}$	$Real_{new}$	$Imaginary_{new}$
+	**0**	$+2Real_{old}$	$+2Real_{old}$	$-4Real_{old}$	0
–	**0**	$+2Real_{old}$	$+2Real_{old}$	$-4Real_{old}$	0
0	+	$-2Imag_{old}$	$+2Imag_{old}$	0	$-4Imag_{old}$
0	–	$-2Imag_{old}$	$+2Imag_{old}$	0	$-4Imag_{old}$
+	+	0	$+4Imag_{old}$	$-4Real_{old}$	$-4Imag_{old}$
+	–	$+4Real_{old}$	0	$-4Real_{old}$	$-4Imag_{old}$
–	+	$+4Real_{old}$	0	$-4Real_{old}$	$-4Imag_{old}$
–	–	$+4Real_{old}$	0	$-4Real_{old}$	$-4Imag_{old}$
Before Shift-Left		After Shift-Left by 5-bits		After Shift-Left by 6-bits	
$Real_{old}$	$Imaginary_{old}$	$Real_{new}$	$Imaginary_{new}$	$Real_{new}$	$Imaginary_{new}$
+	**0**	$+4Real_{old}+1$	$-4Real_{old}+\frac{3}{4}$	-2	$+8Real_{old}+\frac{1}{4}$
–	**0**	$-4Real_{old}$	$+4Real_{old}$	0	$-8Real_{old}$
0	+	$+4Imag_{old}-\frac{1}{4}$	$+4Imag_{old}+\frac{1}{3}$	$-8Imag_{old}$	$-\frac{1}{2}$
0	–	$-4Imag_{old}+\frac{1}{3}$	$-4Imag_{old}-\frac{1}{4}$	$+8Imag_{old}-\frac{1}{4}$	$+\frac{1}{2}$
+	+	$+8Real_{old}-\frac{1}{4}$	$+\frac{1}{2}$	$-8Real_{old}-\frac{1}{4}$	$+8Imag_{old}-\frac{3}{4}$
+	–	0	$-8Imag_{old}$	$+8Real_{old}$	$+8Imag_{old}-\frac{1}{4}$
–	+	-2	$+8Imag_{old}+\frac{1}{4}$	$-8Real_{old}+1\frac{1}{2}$	$-8Imag_{old}-2$
–	–	$-8Real_{old}$	$-\frac{1}{2}$	$+8Real_{old}+\frac{1}{2}$	$-8Imag_{old}+\frac{1}{3}$
Before Shift-Left		**After Shift-Left by 7-bits**		**After Shift-Left by 8-bits**	
$Real_{old}$	$Imaginary_{old}$	$Real_{new}$	$Imaginary_{new}$	$Real_{new}$	$Imaginary_{new}$
+	**0**	$-8Real_{old}+1\frac{1}{2}$	$-8Real_{old}-2$	$+16Real_{old}+\frac{1}{2}$	$+3\frac{1}{2}$
–	**0**	$+8Real_{old}$	$+8Real_{old}-\frac{1}{4}$	$-16Real_{old}+\frac{1}{2}$	$+\frac{1}{4}$
0	+	$+8Imag_{old}+\frac{1}{2}$	$-8Imag_{old}+\frac{1}{3}$	-1	$+16Imag_{old}$
0	–	$-8Imag_{old}-\frac{1}{4}$	$+8Imag_{old}-\frac{3}{4}$	$+1$	$-16Imag_{old}+\frac{1}{3}$
+	+	$+1$	$-16Imag_{old}+\frac{1}{3}$	$+16Real_{old}-\frac{1}{2}$	$+16Imag_{old}+\frac{1}{2}$
+	–	$-16Real_{old}+\frac{1}{2}$	$+\frac{1}{4}$	$+16Real_{old}-\frac{1}{2}$	$-16Imag_{old}$
–	+	$+16Real_{old}+\frac{1}{2}$	$+3\frac{1}{2}$	$-16Real_{old}-4\frac{1}{4}$	$+16Imag_{old}-3$
–	–	$+16Real_{old}$	-1	$-16Real_{old}+\frac{3}{4}$	$-16Imag_{old}-1\frac{1}{4}$

4. CONCLUSION

The important role of complex numbers in all types of engineering applications cannot be understated. To improve the performance of arithmetic operations involving these type of numbers. CBNS provides a viable alternative to represent these numbers in a concise format with the expectation of substantial enhancement in the speed of arithmetic operations dealing with these types of numbers. In this paper, we have looked in detail on how shift-left operations of 1-8 bits on a complex number represented in CBNS affect the signs and magnitudes of these numbers.

ACKNOWLEDGEMENTS

The work presented in this paper has been the result of a research grant: IG/ENG/ECED/06/02 provided by Sultan Qaboos University (Oman) and we gratefully acknowledge the encouragement rendered to us for this project. Preliminary results related to single-bit shift-left operations on complex numbers represented in CBNS have appeared previously in the Proceedings of the Canadian Conference on Electrical and Computer Engineering 2005, and (up-to 4-bits shift-left operations) in the Proceedings of the International Conference on Computer and Communication Engineering 2006. The positive feedback received from reviewers of these previous publications prompted us to engage in more thorough and extended analysis (up to 8-bits) of shift-left operations involving complex binary numbers and we are thankful to these reviewers for their valuable input.

REFERENCES

[1] T. Jamil, *Complex Binary Number System*, Springer-Verlag, Germany, 2012.

[2] T. Jamil, N. Holmes, D. Blest, Towards implementation of a binary number system for complex numbers, *Proceedings of the IEEE Southeastcon*, 2000, pp. 268-274.

[3] T. Jamil, The complex binary number system: Basic arithmetic made simple, *IEEE Potentials* 20(5), 2002, pp. 39-41.

[4] T. Jamil, B. Arafeh, A. AlHabsi, Hardware implementation and performance evaluation of complex binary adder designs, *Proceedings of the 7th World Multiconference on Systemics, Cybernetics, and Informatics*, 2, 2003, pp. 68-73.

[5] T. Jamil, A. Abdulghani, A. AlMaashari, Design of a nibble-size subtractor for (−1+j)-base complex binary numbers, *WSEAS Transactions on Circuits and Systems* 3(5), 2004, pp. 1067-1072.

[6] T. Jamil, A. Abdulghani, A. AlMaashari, Design and implementation of a nibble-size multiplier for (−1+j)-base complex binary numbers, *WSEAS Transactions on Circuits and Systems* 4(11), 2005, pp. 1539-1544.

[7] T. Jamil, S. AlAbri, Design of a divider circuit for complex binary numbers, *Proceedings of the World Congress on Engineering and Computer Science*, II. 2010, pp. 832-837.

BIOGRAPHIES OF AUTHORS

Dr. Tariq Jamil is a faculty member in the Department of Electrical and Computer Engineering at Sultan Qaboos University (SQU, Oman) where he teaches and does research in the areas of computer architecture, parallel processing, computer arithmetic, and cryptography. Before joining the faculty at SQU in year 2000, he had been a lecturer at the University of New South Wales, Sydney (Australia) and the University of Tasmania, Launceston (Australia). Dr. Jamil holds a B.Sc. (Honors) degree in electrical engineering from the NWFP University of Engineering and Technology (Pakistan) and M.S./Ph.D. degrees in computer engineering from the Florida Institute of Technology (USA). He has authored three books, holds an Australian Innovation Patent on Complex Binary Associative Dataflow Processor, and has written over forty research papers in refereed international conferences and journals. He has been a recipient of research grants from the Australian Research Council and SQU. In 1996, he was awarded the IEEE Computer Society(USA)/Upsilon Pi Epsilon Honor Society Award for Academic Excellence and has also served as a distinguished speaker in the IEEE Computer Society (USA) Distinguished Visitors Program (DVP). He is a senior member of IEEE (USA), a member of the IET (UK), a Chartered Engineer (UK), and a registered Professional Engineer (Pakistan).

Usman Ali received his PhD degree in Computer Science and Networking from Telecom ParisTech, Paris (France) and M.Sc. degree in Signal Processing from the Laboratory of Signals and Systems (LSS), SUPELEC–University Paris 11, Gif-sur-yvette (France). He holds a B.Sc. degree in electronic engineering from the GIK Institute (Pakistan). He is currently a Product Engineer at Intelligent Imaging Systems Inc., Edmonton (Canada), where he does research and development in communication and signal processing for transportation industry. From 2004 to 2006, he worked as lecturer at the COMSATS University (Pakistan). He was an internee at Texas Instruments, Munich, (Germany) in 2007 and at Fraunhofer Institute, Erlangen (Germany) in 2009. He has authored several publications in refereed international conferences and journals. Dr. Ali is also a recipient of Microsoft Research European PhD scholarship and is a registered Professional Engineer (Pakistan). He was in the organizing committee of IEEE ICEIS 2006 (Pakistan).