# Bit-wise Byte-wide Analysis of Shift-Right Operations on Binary Representation of Complex Numbers

*AUTHORS AND AFFILIATIONS*

**Usman Ali [1] and Tariq Jamil [2]**

**[1]Laboratory of Signals and Systems L2S, CNRS-Supélec, University of Paris 11, Plateau de Moulon 91192, Gif-sur-Yvette, FRANCE**

**[2]Department of Electrical and Computer Engineering, Sultan Qaboos University, AlKhod 123, Muscat, SULTANATE OF OMAN**



*CORRESPONDING AUTHOR*

**Tariq Jamil**

**P.O.Box 33, Dept. ECE**

**Sultan Qaboos University**

**AlKhod 123 – Muscat**

**OMAN**

## ABSTRACT

Complex numbers play very important role in electrical and computer engineering. Arithmetic operations involving complex numbers represented in binary number system are handled in today's computers by the application of *divide-and-conquer* technique wherein complex numbers are partitioned into real parts and imaginary parts and then operations between each pair (real with real, imaginary with imaginary) are carried out independent of one another. This results in multiple sub-operations within an operation and delay in the production of the final result. Complex Binary Number System provides a one-unit binary representation of a complex number and allows for arithmetic operations on complex numbers to be carried out in exactly the same way as the real numbers. In this paper, we have investigated the effect of bit-wise byte-wide shift right operations on the complex binary representation of complex numbers and analyzed these results using mathematical equations.

## KEYWORDS

Complex number, complex binary number, multiple-shift, shift-right

## 1. Introduction

Complex numbers play very important role in electrical and computer engineering. These days, arithmetic operations involving complex numbers represented in binary number system are handled by the application of *divide-and-conquer* technique wherein complex numbers are partitioned into real parts and imaginary parts and then operations between each pair (real with real, imaginary with imaginary) are carried out independent of one another. This results in multiple sub-operations within an operation and delay in the production of the final result. For example, let's consider arithmetic operations involving two complex numbers $(a + jb)$ and $(c + jd)$. Their addition involves two individual additions, one for the real parts$(a + c)$and one for the imaginary parts$(b + d)$. Their subtraction 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 $ad + bc$. 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 $\frac{bc-ad}{c^2+d^2}$. If we assume that each individual addition/subtraction, involving complex numbers, takes $p$ nsec to complete and each individual multiplication/division takes $q$ nsec to execute, such that $p \ll q$ (multiplication can be assumed to be *repeated-addition* and division can be assumed to be *repeated subtraction*), then each complex addition/subtraction will take $2p$ nsec, each complex multiplication will take $4q + p + p = (2p + 4q)$ nsec, and each complex division will take $6q + 2p + p + 2q = (3p + 8q)$ nsec. Now let's imagine a number system in which complex arithmetic does not involve any *combination of individual arithmetic sub-operations* as described previously. That is, addition, subtraction, multiplication, or division of complex numbers is just one *pure* addition, one *pure*subtraction, one *pure*multiplication, or one *pure*division operation respectively and not a combination of various individual sub-operations within a given arithmetic operation. This will effectively reduce the complex addition/subtraction time to $p$ nsec and complex multiplication/division time to $q$ nsec. Mathematically, such a complex number system will yield reduction in execution time of addition/subtraction operation roughly by a factor of $\frac{p}{2p} \times 100 = 50\%$, for multiplication $\frac{(2p+4q)}{q} \times 100 = \frac{4q}{q} \times 100 = 400\%$ (since $p$ is very small compared to $q$), and for division $\frac{(3p+8q)}{q} \times 100 = \frac{8q}{q} \times 100 = 800\%$. With the reduction in execution times of complex arithmetic operations roughly by factors of 50% to 800% in digital signal and image processing applications of engineering where complex numbers are most frequently utilized, it is possible to achieve tremendous improvement in the overall performance of systems, provided that a technique exists which treats a complex number as a *single entity* (rather than *two entities* comprising of real and imaginary parts) and facilitates a single-unit representation of complex numbers in binary format within a microprocessor environment (rather than *two individual representations* for real and imaginary parts respectively, as in

today's computers). Such a unique number system is referred to as *Complex Binary Number System* (CBNS) [1].

There have been several efforts in the past to define a binary number system (0 or 1) with bases other than 2 which would facilitate a one-unit representation of complex numbers. In 1960, Donald Knuth described a number system with base $2j$ and analyzed the arithmetic operations of numbers based on this imaginary base [2]. He was unsuccessful in providing a division algorithm for binary numbers based on this imaginary base and considered it as a main obstacle towards hardware implementation of this number system.Four years later, Walter Penney defined a binary number system, first by using a negative base of $-4$ [3], and then by using a complex number $(-1+j)$ as the base [4]. Like Donald Knuth before him, Walter Penney was unable to formulate an efficient division process using these bases and, consequently, these number systems remained dormant for more than thirty years. V. Stepanenko, in 1996, defined a number system with the base $j\sqrt{2}$ and generated real and imaginary parts of complex numbers by taking even and odd powers of this base respectively [5]. He succeeded somewhat in resolving the division problem as an "all-in-one" operation but, in his algorithm, "…everything…reduces to good choice of an initial approximation" in a Newton-Raphson iteration which may or may not converge.Jamil *et al*, in 2000 [6], revisited Penney's number system with base $(-1+j)$ and presented a detailed analysis of this number system, now famously called Complex Binary Number System (CBNS) [7]. Extensive details of the algorithms to be followed in CBNS for binary representation of a complex number can be found in [1] and the implementation statistics of the arithmetic circuits designed based on this number system can be found in [8,9,10,11]. In this paper, we haverestricted ourselves to investigating the effects of multiple-bit (from 1 bit to 8 bits) shift-right operations only on a complex number represented in single-unit binary notation of complex binary number system.



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 effect of multiple-bit shift right operations on the complex numbers. Conclusions are presented in section 4, which are followed by acknowledgments and references.

## 2. $(-1+j)$-base Complex Binary Number System

Mathematically, the value of any n-bit binary number $(b_{n-1}b_{n-2}b_{n-3}\ldots b_2b_1b_0)$in base 2 can be represented in the form of a power series as $b_{n-1}(2)^{n-1} + b_{n-2}(2)^{n-2} + b_{n-3}(2)^{n-3} + \ldots + b_2(2)^2 + b_1(2)^1 + b_0(2)^0$. Following the same procedure, 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 one-unit binary representation of any given complex number, whether it is made from integers, fractions, or floating point numbers, in Complex Binary Number System (CBNS). In this paper only the conversion algorithm for integershas been excerpted from [1].

The reader is referred to [7] for a thorough coverage of the conversion algorithms for fractions and floating point numbers.

A positive integer *N* can be converted into CBNS representation by following these steps:

(i) Express *N* in terms of powers of 4 by repeatedly dividing by 4 and keeping track of the remainders. This will result in a base 4 number $(\ldots n_{5,}n_{4,}n_{3,}n_{2,}n_{1,}n_{0,})$ where $n_i \in \{0,1,2,3\}$

(ii) Convert Base 4 number $(\ldots n_{5,}n_{4,}n_{3,}n_{2,}n_{1,}n_{0,})$to Base $-4$ by replacing each digit in the odd location $(n_{1,}n_{3,}n_{5\,,\ldots})$ with its negative to get $(\ldots -n_{5,}n_{4,}-n_{3,}n_{2,}-n_{1,}n_{0,})$. Next, we normalize the new number, i.e., get each digit in the range 0 to 3, by repeatedly adding 4 to the negative digits and adding a 1 to the digit on its left. This operation will get rid of negative numbers but may create some digits with a value of 4 after the addition of a 1. To normalizethis, we replace 4 by a 0 and subtract a 1 from the digit on its left. This subtraction might once again introduce negative digits which will be normalized using the previous method but this process will definitely terminate.

(iii) Lastly, we replace each digit in the normalized representation by its equivalent binary representation in CBNS, i.e., 0 → 0000, 1 → 0001, 2 → 1100, and 3 →1101.

For example,

$$2012_{10} = (1,3,3,1,3,0)_{Base\ 4} = (-1,3,-3,1,-3,0)_{Base\ -4}$$
$$= (1,2,0,1,2,1,0)_{Normalized} = 0001\ 1100\ 0000\ 0001\ 1100\ 0001\ 0000$$
$$= 11100000000011100000100 00_{Base\ (-1+j)}$$



To convert a negative integer into CBNS format, we simply multiply the representation of the corresponding positive integer with 11101 (equivalent to $(-1)_{Base\,(-1+j)}$) according to the multiplication algorithm for CBNS[1].

For example,

$$-2012_{10} = 111000000000111000001 \times 11101$$
$$= 1100000000001101110100 00_{Base\ (-1+j)}$$

To obtain binary representation of a positive or negative imaginary number in CBNS, we multiply the corresponding CBNS representation of positive or negative integer with 11 (equivalent to $(+j)_{Base\,(-1+j)}$) or 111 (equivalent to $(-j)_{Base\,(-1+j)}$) according to the multiplication algorithm for CBNS [1]. Thus

$$+j2012_{10} = 111000000000111000001 \times 11$$
$$= 1000000000001000011000 0_{Base\ (-1+j)}$$

$$-j2012_{10} = 111000000000111000001 \times 111$$
$$= 1110100000001110100011100 00_{Base\ (-1+j)}$$

After obtaining CBNS representations for all types of integers (real and imaginary), it is now possible for us to represent an integer complex number (both real and imaginary parts of the complex number are integers) simply by adding the real and imaginary CBNS representations according to the addition algorithm for CBNS [1]. Thus

$$2012_{10}+j2012_{10}$$
$$= 1110000000001110000010000_{Base\ (-1+j)} + 1000000000010000110000_{Base\ (-1+j)}$$
$$= 1110100000001110100011100000_{Base\ (-1+j)}$$

### 3. Multiple-Bit Shift Right Operations in CBNS

To analyze the effects of shift-right operations on a complex number represented in CBNS format, we wrote a computer program in C++ language which allowed for variations in magnitude and sign of both real and imaginary components of a complex number to be generated automatically in a linear fashion, and then decomposed the complex binary number after the shift-right operation into its real and imaginary components. We restricted the length of the original binary bit array 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 illustrate 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 right will yield $00 \ldots 011010001000100_{Base\ (-1+j)}$ensuring that total array-size remains 800 bits. This is done by removing one 0 from the right-side and inserting one 0 on the left-side of the number. Similarly, shifting of the original binary array by 2,3,4,5,6,7,8-bits to the right will yield respectively:

$000 \ldots 01101000100010_{Base\ (-1+j)}$
$0000 \ldots 0110100010001_{Base\ (-1+j)}$
$00000 \ldots 011010001000_{Base\ (-1+j)}$
$000000 \ldots 01101000100_{Base\ (-1+j)}$
$0000000 \ldots 0110100010_{Base\ (-1+j)}$
$00000000 \ldots 011010001_{Base\ (-1+j)}$
$000000000 \ldots 01101000_{Base\ (-1+j)}$

All the time we make sure that the total array-size remains 800 bits by removing 2,3,4,5,6,7,8 bits, respectively from the right-side of the original array.

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-right operations (1 to 8 bits).

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

| Before Shift-Right | | After Shift-Right by 1-bit | | After Shift-Right by 2-bits | |
|---|---|---|---|---|---|
| Real | Imaginary | Real | Imaginary | Real | Imaginary |
| + | 0 | – | – | 0 | + |
| – | 0 | + | + | 0 | – |
| 0 | + | + | – | – | 0 |
| 0 | – | – | + | + | 0 |
| + | + | 0 | – | – | + |
| + | – | – | 0 | + | + |
| – | + | + | 0 | – | – |
| – | – | 0 | + | + | – |

| Before Shift-Right | | After Shift-Right by 3-bits | | After Shift-Right by 4-bits | |
|---|---|---|---|---|---|
| Real | Imaginary | Real | Imaginary | Real | Imaginary |
| + | 0 | + | – | – | 0 |
| – | 0 | – | + | + | 0 |
| 0 | + | + | + | 0 | – |
| 0 | – | – | – | 0 | + |
| + | + | + | 0 | – | – |
| + | – | 0 | – | – | + |
| – | + | 0 | + | + | – |
| – | – | 0 | + | + | + |

**Table 1 (continued).** Effect on signs of complex numbers in CBNS format after shift-right operations

| Before Shift-Right | | After Shift-Right by 5-bits | | After Shift-Right by 6-bits | |
|---|---|---|---|---|---|
| Real | Imaginary | Real | Imaginary | Real | Imaginary |
| + | 0 | + | + | 0 | – |
| – | 0 | – | – | 0 | + |
| 0 | + | – | + | + | 0 |
| 0 | – | + | – | – | 0 |
| + | + | 0 | + | + | – |
| + | – | + | 0 | – | – |
| – | + | – | 0 | + | + |
| – | – | 0 | – | – | + |

| Before Shift-Right | | After Shift-Right by 7-bits | | After Shift-Right by 8-bits | |
|---|---|---|---|---|---|
| Real | Imaginary | Real | Imaginary | Real | Imaginary |
| + | 0 | – | + | + | 0 |
| – | 0 | + | – | – | 0 |
| 0 | + | – | – | 0 | + |
| 0 | – | + | + | 0 | – |
| + | + | – | 0 | + | + |
| + | – | 0 | + | + | – |
| – | + | 0 | – | – | + |
| – | – | + | 0 | – | – |

Shift-right 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 different mathematical relationships. To find out the effects of shift-right 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 (Fig. 1). The complex numbers obtained after shift-right operations were analysed by obtaining mathematical equations describing their behavior, as given in Figs. 2-9.

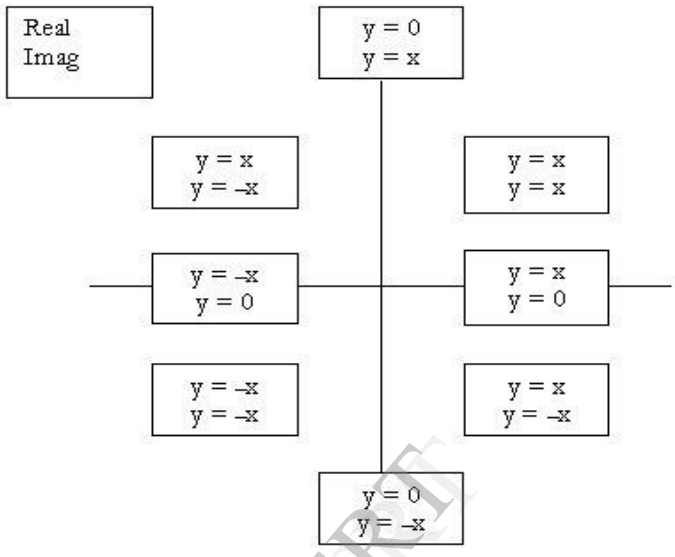




Fig.1. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (before shift-right)

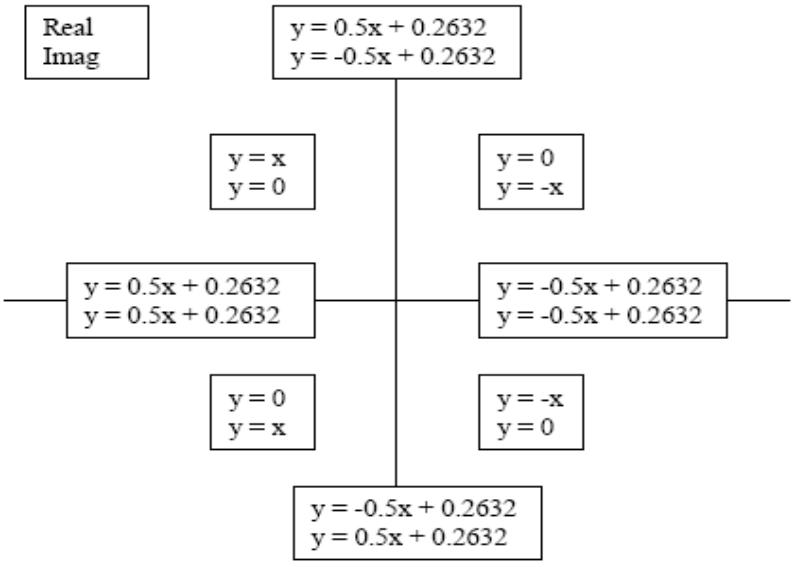


Fig. 2. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (after shift-right by 1-bit)

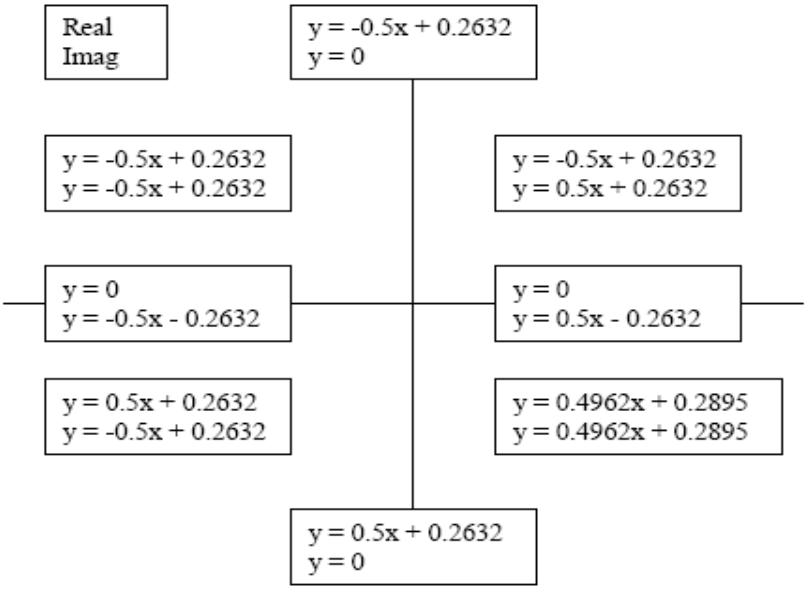


Fig.3. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (after shift-right by 2-bits)

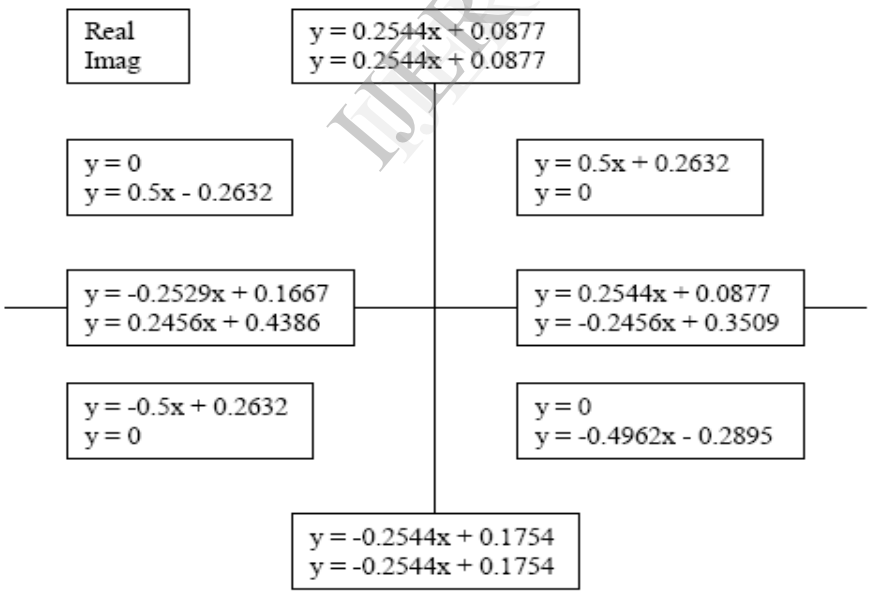


Fig. 4. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (after shift-right by 3-bits)

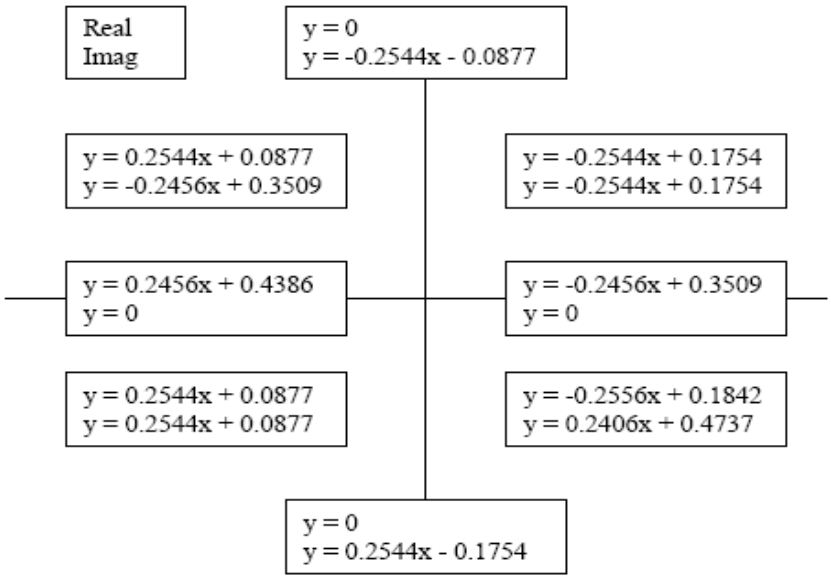


Fig.5. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (after shift-right by 4-bits)



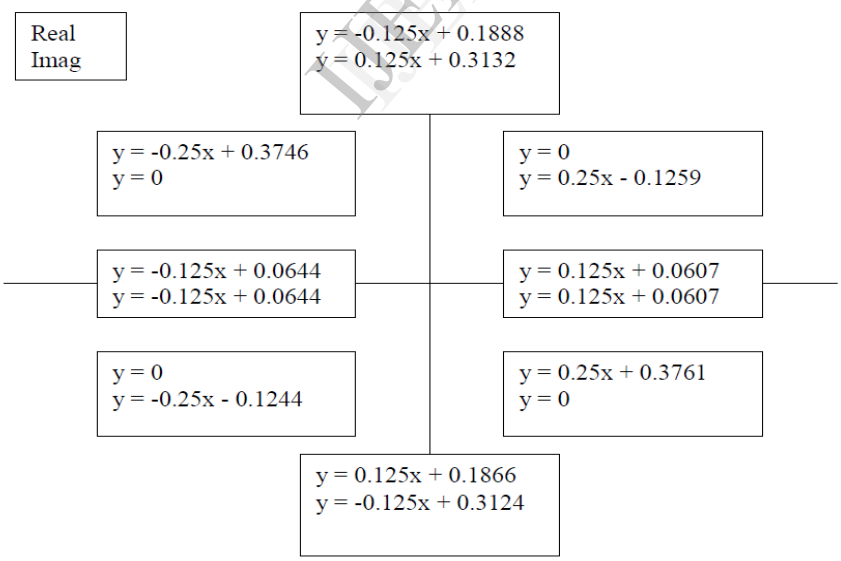


Fig. 6. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (after shift-right by 5-bits)

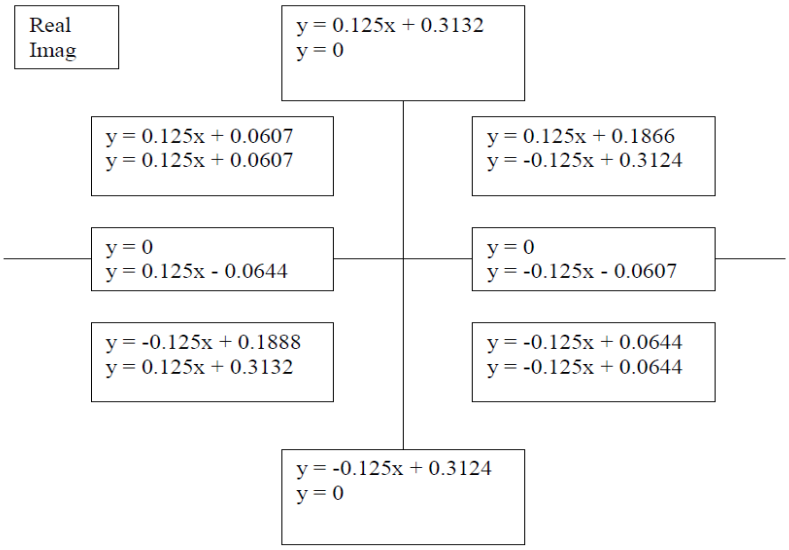


Fig.7. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (after shift-right by 6-bits)

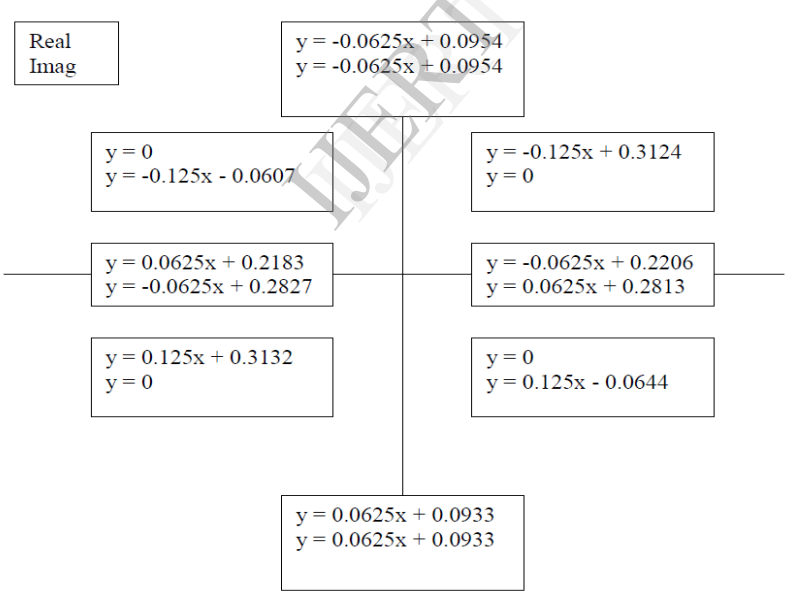


Fig. 8. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (after shift-right by 7-bits)

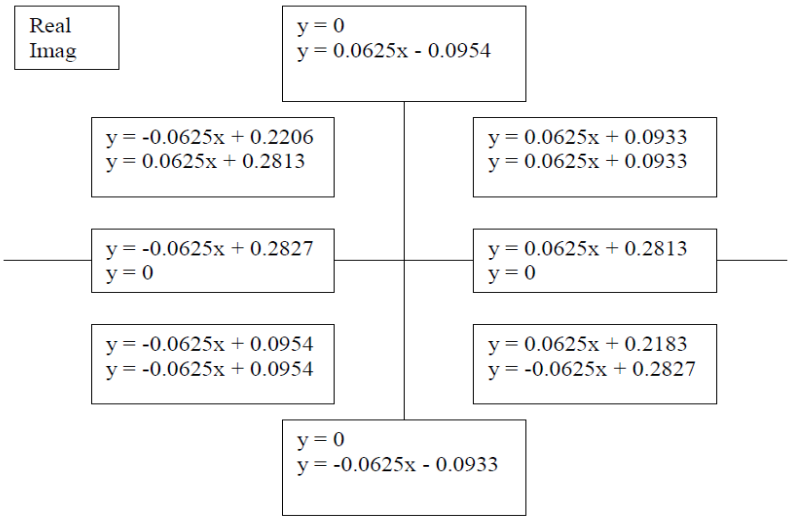


Fig. 9. Mathematical equations describing variations in magnitudes of complex numbers in the Cartesian plane (after shift-right by 8-bits)

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

Fig. 10 and Fig. 11 illustrate the effect on the real and imaginary parts of the complex numbers after 1-8-bits shift-right operations for positive and negative real only complex numbers (no imaginary part), respectively.

Fig. 12 and Fig. 13 present the effect on the real and imaginary parts of the complex numbers after 1-8-bits shift-right operations for positive and negative only imaginary complex numbers (no real part), respectively.

The four cases of ±Real±Imaginary complex numbers represented in CBNS format before the shift, and effects of 1-8 bits shift-right operations on the sign and magnitude of complex numbers are presented in Figs. 14,15,16, and 17 respectively.

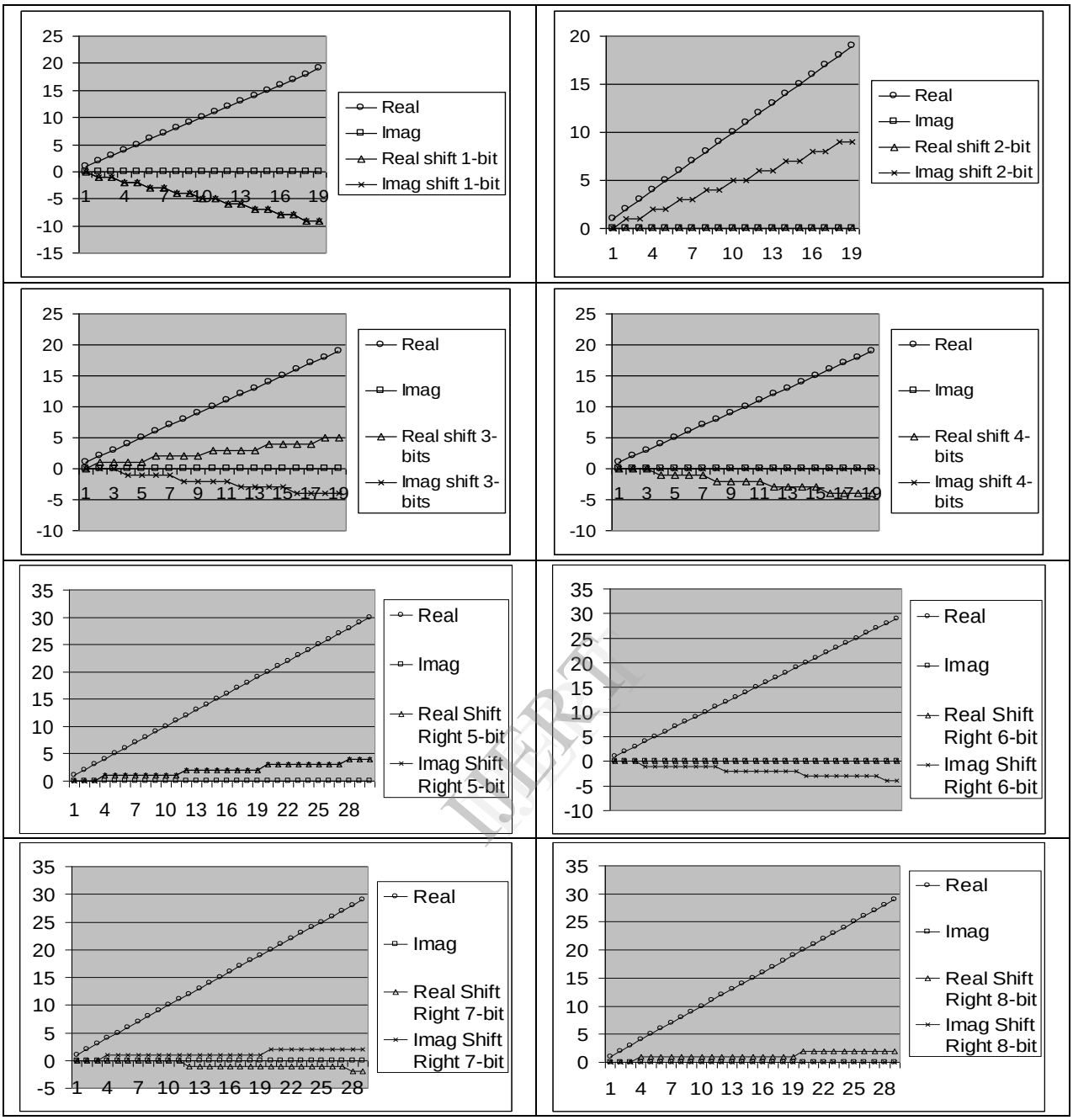


Fig. 10. Effects of shift-right operation on sign and magnitude of a positive real-only complex number (1-8 bits)

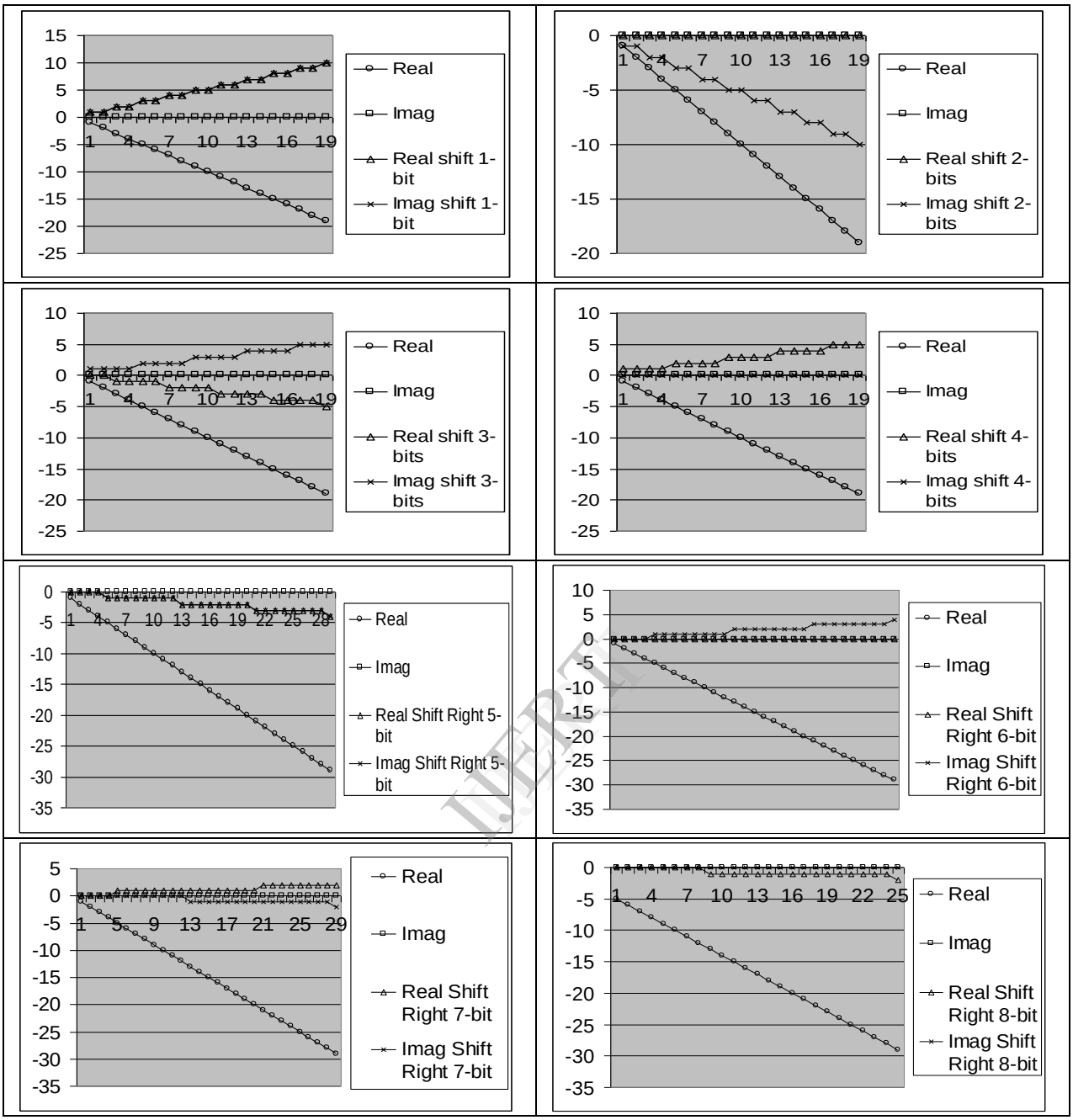


Fig. 11. Effects of shift-right operation on sign and magnitude of a negative real-only complex number (1-8 bits)

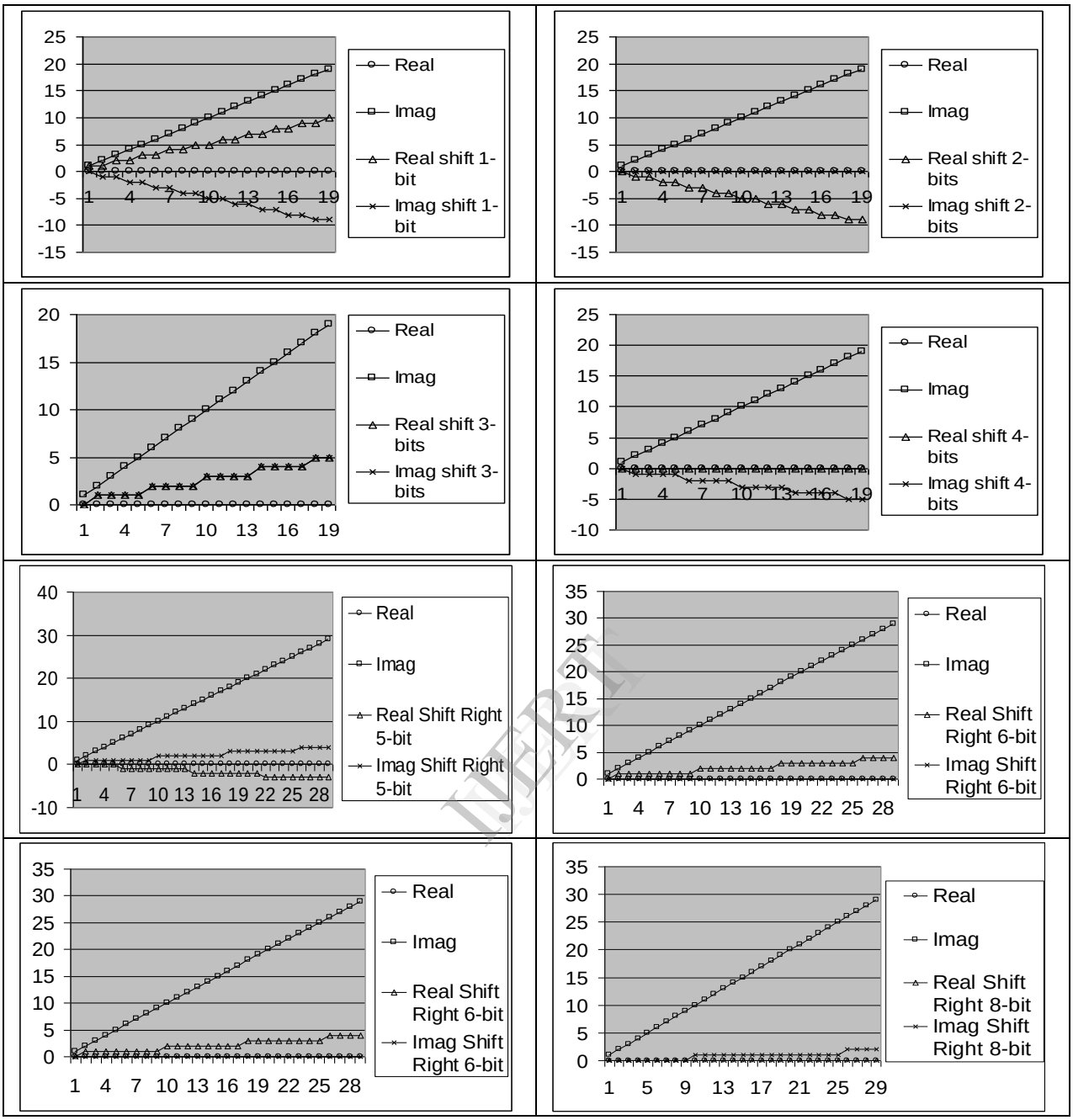


Fig. 12. Effects of shift-right operation on sign and magnitude of a positive imaginary-only complex number (1-8 bits)

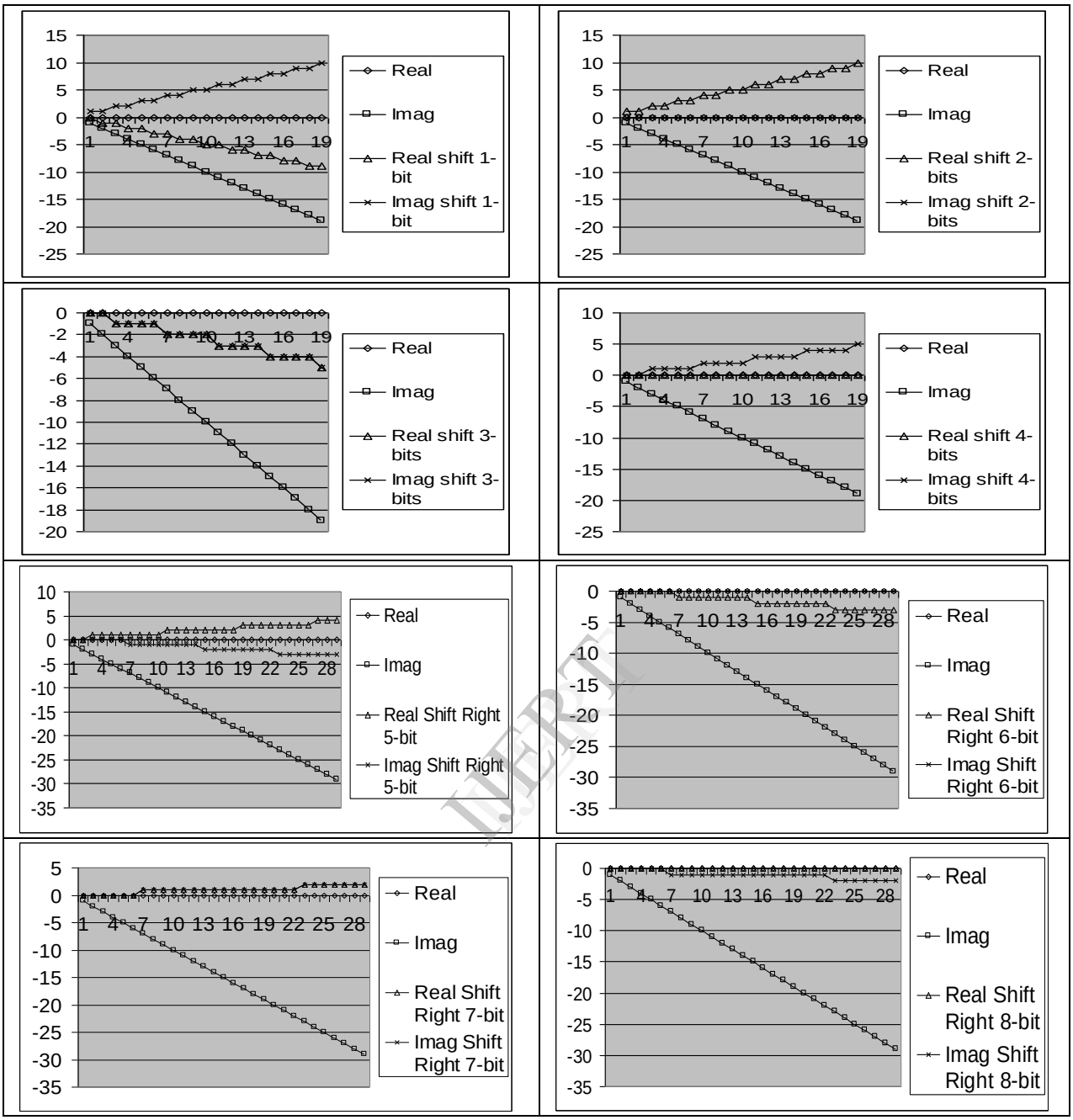


Fig. 13. Effects of shift-right operation on sign and magnitude of a negativeimaginary-only complex number (1-8 bits)

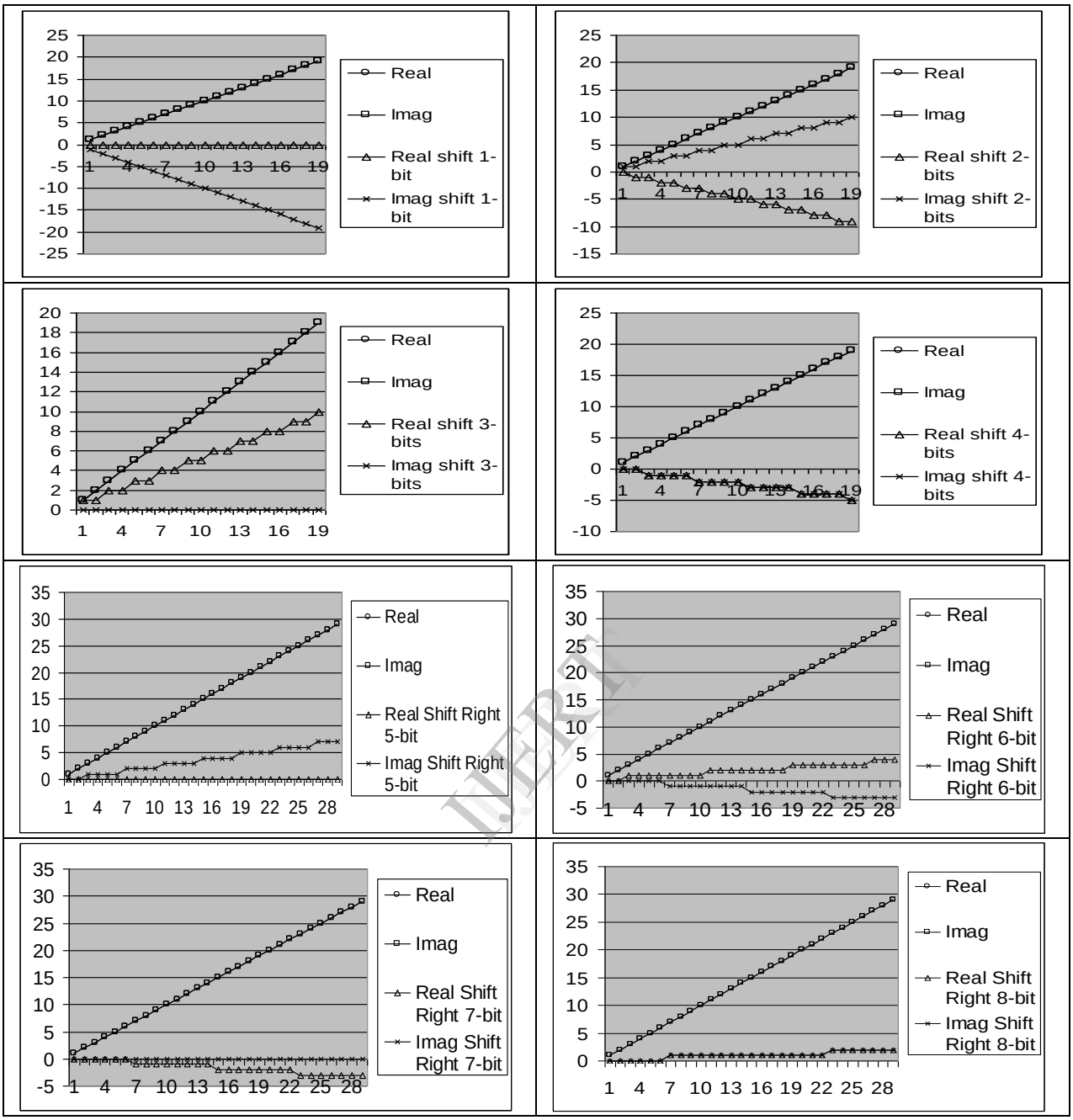


Fig. 14. Effects of shift-right operation on sign and magnitude of a +Real+Imaginary complex number (1-8 bits)

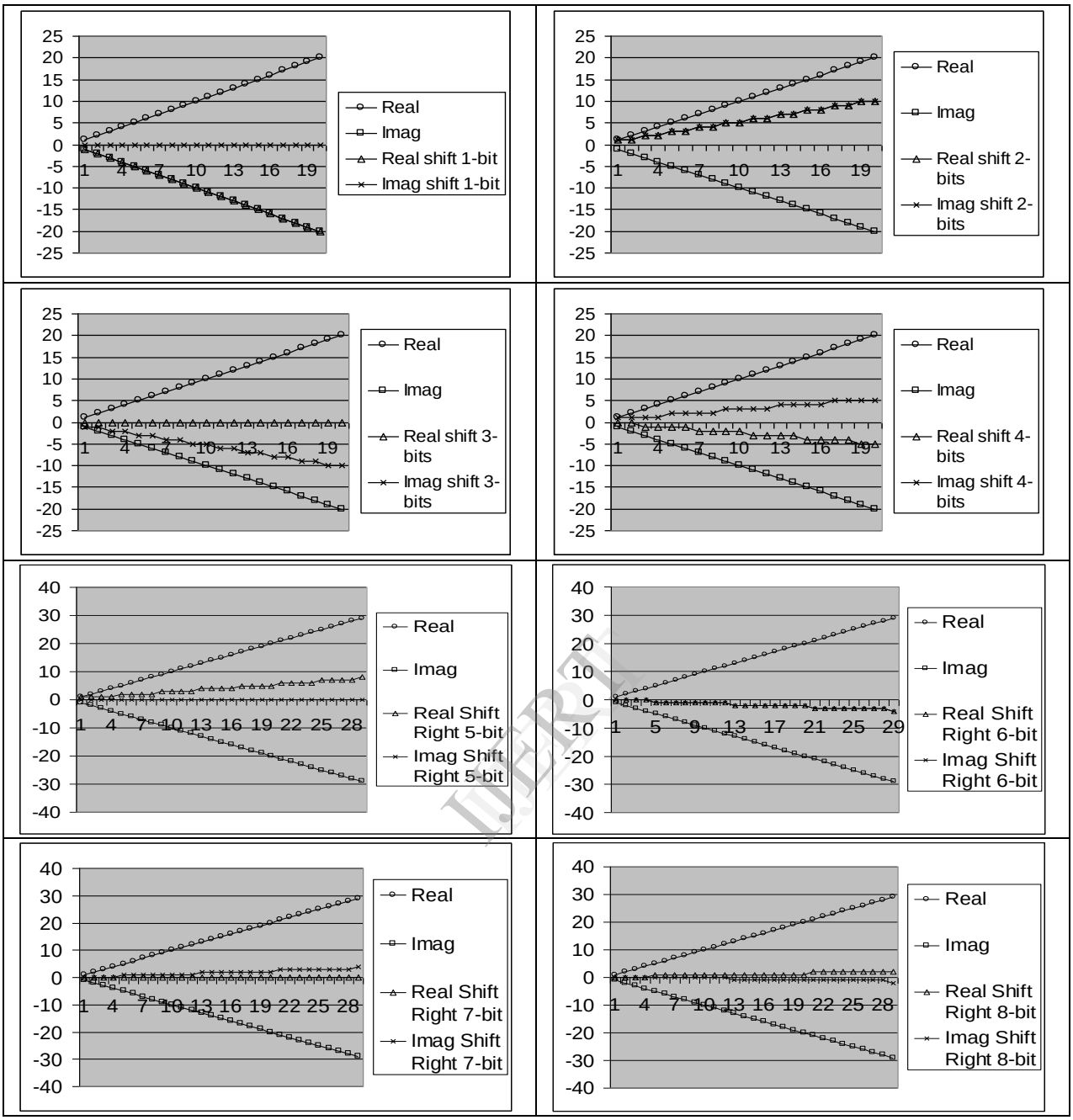


Fig. 15. Effects of shift-right operation on sign and magnitude of a +Real–Imaginary complex number (1-8 bits)

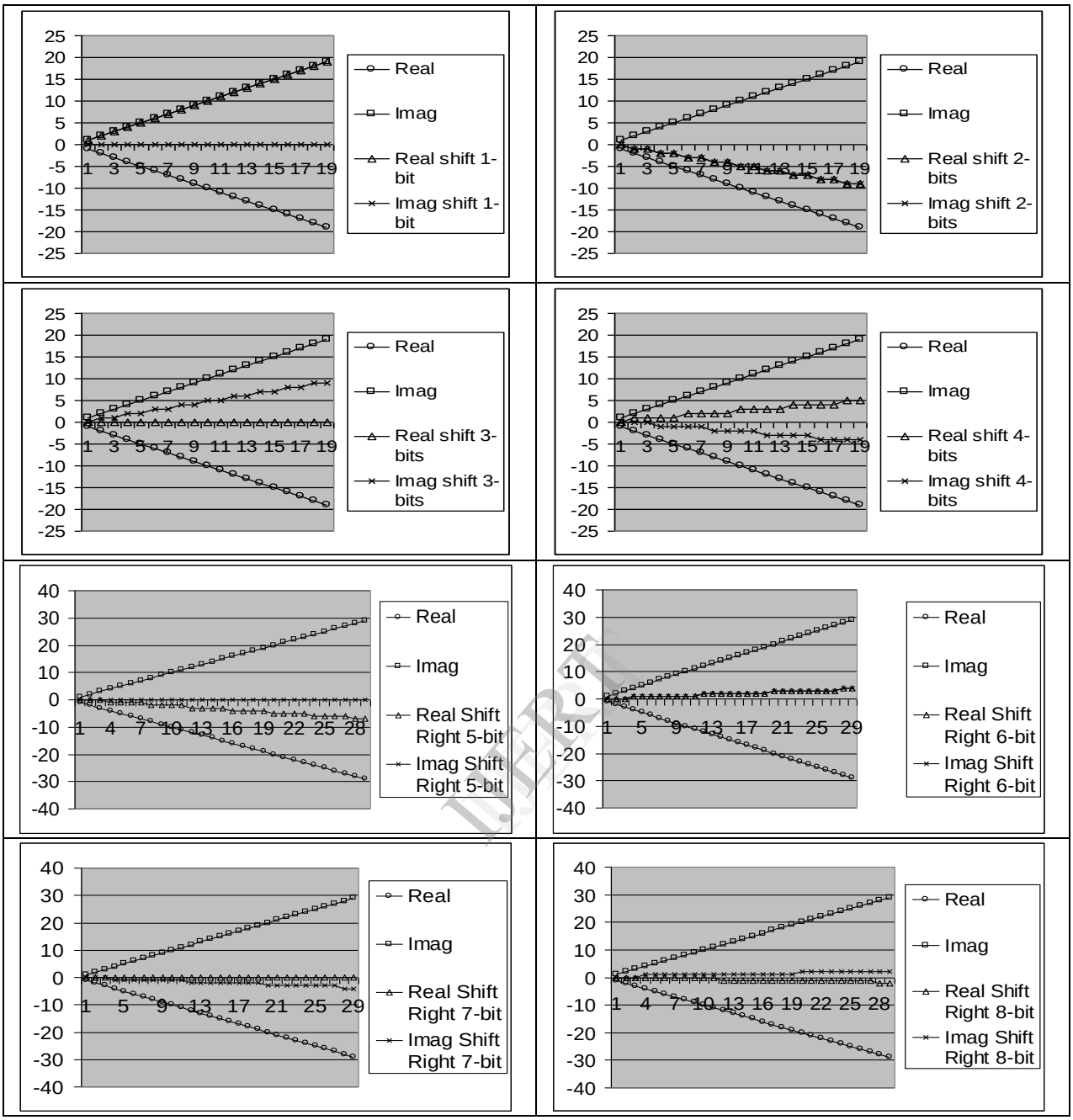


Fig.16. Effects of shift-right operation on sign and magnitude of a –Real+Imaginary complex number (1-8 bits)

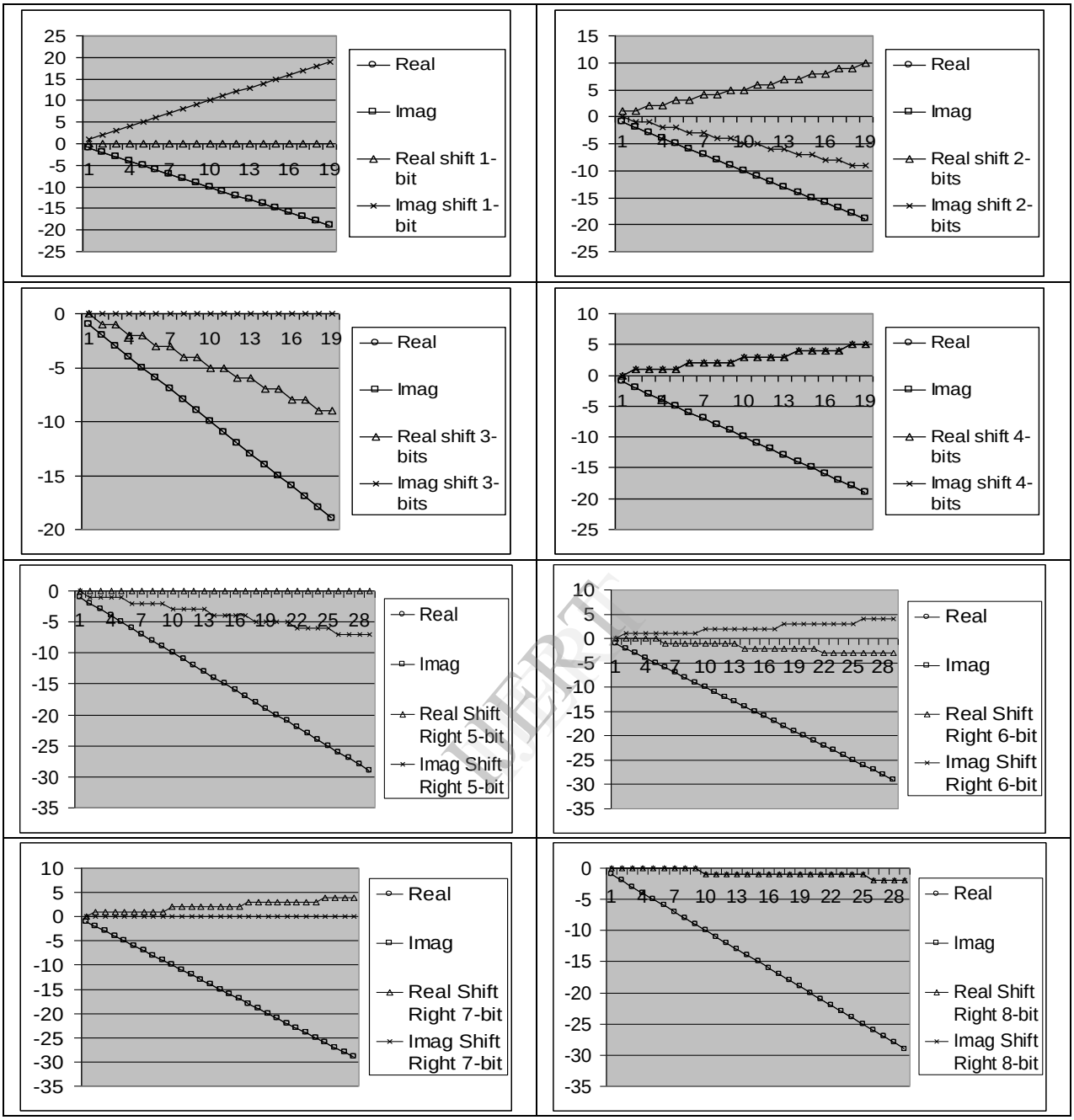


Fig. 17. Effects of shift-right operation on sign and magnitude of a –Real–Imaginary complex number (1-8 bits)

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

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

| Type of Complex Number | | After Shift-Right by 1-bit | | After Shift-Right by 2-bits | |
|---|---|---|---|---|---|
| $Real_{old}$ | $Imaginary_{old}$ | $Real_{new}$ | $Imaginary_{new}$ | $Real_{new}$ | $Imaginary_{new}$ |
| + | 0 | $-½Real_{old}+¼$ | $-½Real_{old}+¼$ | 0 | $+½Real_{old}-¼$ |
| – | 0 | $-½Real_{old}+¼$ | $-½Real_{old}+¼$ | 0 | $+½Real_{old}-¼$ |
| 0 | + | $½Imag_{old}+¼$ | $-½Imag_{old}+¼$ | $-½Imag_{old}+¼$ | 0 |
| 0 | – | $½Imag_{old}+¼$ | $-½Imag_{old}+¼$ | $-½Imag_{old}+¼$ | 0 |
| + | + | 0 | $-Imag_{old}$ | $-½Real_{old}+¼$ | $½Imag_{old}+¼$ |
| + | – | $-Real_{old}$ | 0 | $½Real_{old}+¼$ | $½Imag_{old}+¼$ |
| – | + | $Real_{old}$ | 0 | $½Real_{old}+¼$ | $-½Imag_{old}+¼$ |
| – | – | 0 | $-Imag_{old}$ | $-½Real_{old}+¼$ | $½Imag_{old}+¼$ |

| Type of Complex Number | | After Shift-Right by 3-bits | | After Shift-Right by 4-bits | |
|---|---|---|---|---|---|
| $Real_{old}$ | $Imaginary_{old}$ | $Real_{new}$ | $Imaginary_{new}$ | $Real_{new}$ | $Imaginary_{new}$ |
| + | 0 | $¼\ Real_{old}$ | $-¼\ Real_{old}$ | $-¼Real_{old}$ | 0 |
| – | 0 | $¼\ Real_{old}$ | $-¼\ Real_{old}$ | $-¼Real_{old}$ | 0 |
| 0 | + | $¼\ Imag_{old}$ | $¼\ Imag_{old}$ | 0 | $-¼Imag_{old}$ |
| 0 | – | $¼\ Imag_{old}$ | $¼\ Imag_{old}$ | 0 | $-¼Imag_{old}$ |
| + | + | $½Real_{old}+¼$ | 0 | $-¼Real_{old}$ | $-¼Imag_{old}$ |
| + | – | 0 | $½Imag_{old}-¼$ | $-¼Real_{old}$ | $-¼Imag_{old}$ |
| – | + | 0 | $½Imag_{old}-¼$ | $-¼Real_{old}$ | $-¼Imag_{old}$ |
| – | – | $½Real_{old}+¼$ | 0 | $-¼Real_{old}$ | $-¼Imag_{old}$ |

**Table 2 (continued)** Characteristic equations describing complex numbers in CBNS format after shift-right operations

| Type of Complex Number | | After Shift-Right by 5-bits | | After Shift-Right by 6-bits | |
|---|---|---|---|---|---|
| $Real_{old}$ | $Imaginary_{old}$ | $Real_{new}$ | $Imaginary_{new}$ | $Real_{new}$ | $Imaginary_{new}$ |
| + | 0 | ⅛ $Real_{old}$ | ⅛ $Real_{old}$ | 0 | −⅛$Real_{old}$ |
| − | 0 | −⅛$Real_{old}$ | −⅛$Real_{old}$ | 0 | ⅛$Real_{old}$ |
| 0 | + | −⅛ $Imag_{old + ⅓}$ | ⅛ $Imag_{old + ⅓}$ | ⅛ $Imag_{old + ⅓}$ | 0 |
| 0 | − | ⅛ $Imag_{old + ⅓}$ | −⅛ $Imag_{old + ⅓}$ | −⅛ $Imag_{old + ⅓}$ | 0 |
| + | + | 0 | ¼$Imag_{old}$−⅛ | ⅛ $Real_{old + ⅓}$ | −⅛ $Real_{old}$ + ⅓ |
| + | − | ¼ $Real_{old}$ + ⅓ | 0 | −⅛ $Real_{old}$ | −⅛ $Imag_{old}$ |
| − | + | −¼ $Real_{old}$ + ⅓ | 0 | ⅛ $Real_{old}$ | ⅛ $Imag_{old}$ |
| − | − | 0 | −¼ $Imag_{old}$−⅛ | −⅛$Real_{old + ⅓}$ | ⅛ $Imag_{old + ⅓}$ |

| Type of Complex Number | | After Shift-Right by 7-bits | | After Shift-Right by 8-bits | |
|---|---|---|---|---|---|
| $Real_{old}$ | $Imaginary_{old}$ | $Real_{new}$ | $Imaginary_{new}$ | $Real_{new}$ | $Imaginary_{new}$ |
| + | 0 | ¼ | ¼ | ¼ | 0 |
| − | 0 | ¼ | ¼ | ¼ | 0 |
| 0 | + | 0 | 0 | 0 | 0 |
| 0 | − | 0 | 0 | 0 | 0 |
| + | + | −⅛ $Real_{old}$ + ⅓ | 0 | 0 | 0 |
| + | − | 0 | ⅛$Imag_{old}$ | ¼ | ¼ |
| − | + | 0 | −⅛$Imag_{old}$ | ¼ | ¼ |
| − | − | ⅛ $Real_{old}$ + ⅓ | 0 | 0 | 0 |

## 4. Conclusions

The role of complex numbers is vital in all types of engineering applications and the importance in these numbers in the professional realm of an engineer cannot be ignored. However, ever since the birth of computer, these numbers have been treated as distant relatives of the "real" world and nothing substantial has ever been done in the field of computer architecture and arithmetic to improve the performance of arithmetic operations involving these type of numbers. CBNS provides a viable alternative for a single-unit binary representation of complex numbers with the premise 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-right operations of 1-8 bits on a complex number represented in CBNS affect the signs and magnitudes of these numbers. Continuing with this work, we intend to do similar analysis for shift-left operations on complex numbers represented in CBNS so that a wholesome picture about the true benefits of CBNS are established within the engineering community.

## Acknowledgments

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 versions of this paper, related to only nibble-size (up to 4-bits) shift-right operations have appeared previously in the Proceedings of the IEEE SoutheastCon 2007 and the Journal of Computers (Academy Publisher) in 2008. 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-right operationsinvolving 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] D. Knuth, An imaginary number system, Communications of the ACM, 3, pp.345-347.

[3] W. Penney, A numeral system with a negative base, Mathematics Student Journal, 11(4), 1964, pp. 1-2.

[4] W. Penney, A binary system for complex numbers, Journal of the ACM, 12(2), 1964, pp. 247-248.

[5] V. Stepanenko, Computer arithmetic of complex numbers, Cybernetics and System Analysis, 32(4), 1996, pp. 585-591.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.