ISSN:2231-0711
www.ijcset.net

# Arithmetical Operations in Quaternary System Using VHDL

Snehal B. Sahastrabudhey, K. M. Bogawar

*Electronics Department (VLSI)*
*Priyadarshini College of Engg. ,Nagpur, India*

snehalfeb16@gmail.com , karuved@rediffmail.com

*Abstract*: **While performing the several arithmetic operations such as addition, subtraction and multiplication the speed of modern computers are limited because of carry propagation delay. A carry-free arithmetic operation can be achieved using a higher radix number system such as Quaternary Signed Digit (QSD). In QSD, each digit can be represented by a number from -3 to 3. Using this number system any integer can be represented in multiple ways. With constant or fix delay and less complexity carry free addition, multiplication and other operations can be implemented on large number of digits. This paper deals with the implementation of QSD based arithmetic operations. The designs are simulated and synthesized using VHDL software, Modelsim software is used for simulation.**

## I. INTRODUCTION

**A**rithmetic operations are widely used and plays an important role in various digital systems such as computers and signal processors. **A**rithmetic operations suffer from known problems including limited number of bits, propagation time delay, and circuit complexity. Carry look ahead helps to improve the propagation delay, but is bounded to a small number of digits due to the complexity of the circuit. Signed digit number system offers the possibility of Carry free addition.

In this paper, we propose a high speed QSD arithmetic logic unit which is capable of carry free addition, borrow free subtraction. The QSD addition/subtraction operation employs a fixed number of minterms for any operand size. In QSD number system carry propagation chain are eliminated which reduce the computation time substantially, thus enhancing the speed of the machine.

## II. SIGNED DIGIT NUMBERS

Signed digit representation of number indicates that digits can be prefixed with a –(minus) sign to indicate that they are negative. Signed digit representation can be used in low-level software & hardware to accomplish fast addition of integers because it can eliminate carriers. Example,

$$
\begin{aligned}
(101\bar{1})_2 &= 1\times 2^3 + 0\times 2^2 + 1\times 2^1 - 1\times 2^0 \\
&= 8 + 0 + 2 - 1 \\
&= 9
\end{aligned}
$$

## III. QUATERNARY SIGNED DIGIT NUMBERS

Quaternary digits {0,1,2,3} are represented as 2-bit equivalent in binary where as QSD are represented using 3-bit 2's complement notation. Each number can be represented by:

$$D = \sum_{i}^{n} x_i \, 4^i ,$$

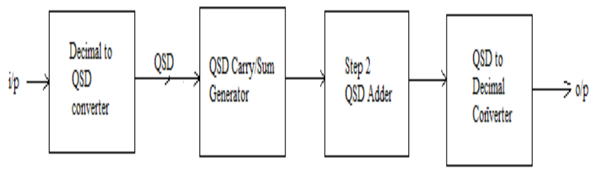
where *xi* can be any value from the set $\{\bar{3}, \bar{2}, \bar{1}, 0,1,2,3\}$ for producing an appropriate decimal representation. A QSD negative number is the QSD complement of the QSD positive number i.e.,

Large number of digits such as 64, 128, or more can be implemented with constant delay.

$$
\begin{aligned}
(1\bar{2}\bar{3}3)_{QSD} &= (23)_{10} \\
&= 1\times 4^3 + \bar{2}\times 4^2 + \bar{3}\times 4^1 + 3\times 4^0 \\
&= 64 - 32 - 12 + 3 \\
(\bar{1}23\bar{3})_{QSD} &= (-23)_{10} \quad = 23
\end{aligned}
$$

## IV. BASIC CONCEPT

For performing any operation in QSD, first convert the binary or any other input into quaternary signed digit.



## V. ADDER/SUBSTRACTOR DESIGN

One of the most important arithmetic operations in digital computation is addition. As the number of digits become large during addition, a carry-free addition is highly desirable. Carry-free addition can be achieved by exploiting the redundancy of QSD numbers and the QSD addition. The redundancy allows multiple representations of any integer quantity.

There are two steps involved in the carry-free addition. The first step generates an intermediate carry and sum from the addend and augends. The second step combines the intermediate sum of the current digit with the carry of the lower significant digit.

To prevent carry from further rippling, we define two rules. The first rule states that the magnitude of the intermediate sum must be less than or equal to 2. The second rule states that the magnitude of the carry must be less than or equal to 1. Consequently, the magnitude of the second step output cannot be greater than 3 which can be represented by a single-digit QSD number; hence no further carry is required. In step 1, all possible input pairs of the addend and augends are considered. The output ranges from –6 to 6 as shown in Table 1.

**TABLE I.**
THE OUTPUTS OF ALL POSSIBLE COMBINATIONS OF A PAIR OF ADDEND (A) AND AUGENDS (B)

| A \ B | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|
| -3 | -6 | -5 | -4 | -3 | -2 | -1 | 0 |
| -2 | -5 | -4 | -3 | -2 | -1 | 0 | 1 |
| -1 | -4 | -3 | -2 | -1 | 0 | 1 | 2 |
| 0 | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| 1 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
| 2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

The range of the output is from -6 to 6 which can be represented in the intermediate carry and sum in QSD format as show in Table II.

**TABLE II.**
OUTPUTS OF ALL POSSIBLE COMBINATIONS OF A PAIR OF INTERMEDIATE CARRY (A) AND SUM (B)

| A \ B | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| -1 | -3 | -2 | -1 | 0 | 1 |
| 0 | -2 | -1 | 0 | 1 | 2 |
| 1 | -1 | 0 | 1 | 2 | 3 |

Some numbers have multiple representations, but only those that meet the defined rules are chosen. The chosen intermediate carry and sum are listed in last column of table III.

**TABLE III.**
THE INTERMEDIATE CARRY AND SUM BETWEEN -6 TO 6

| Sum | QSD represented number | QSD coded number |
|---|---|---|
| -6 | $\bar{2}2,\bar{1}\bar{2}$ | $\bar{1}\bar{2}$ |
| -5 | $\bar{2}3,\bar{1}\bar{1}$ | $\bar{1}\bar{1}$ |
| -4 | $\bar{1}0$ | $\bar{1}0$ |
| -3 | $\bar{1}1,0\bar{3}$ | $\bar{1}1$ |
| -2 | $\bar{1}2,0\bar{2}$ | $0\bar{2}$ |
| -1 | $\bar{1}3,0\bar{1}$ | $0\bar{1}$ |
| 0 | 00 | 00 |
| 1 | $01,1\bar{3}$ | 01 |
| 2 | $02,1\bar{2}$ | 02 |
| 3 | $03,1\bar{1}$ | $1\bar{1}$ |
| 4 | 10 | 10 |
| 5 | $11,2\bar{3}$ | 11 |
| 6 | $12,2\bar{2}$ | 12 |

In step 1 both inputs and outputs can be encoded in 3-bit 2'scomplement binary number. The mapping between the inputs, addend and augends, and the outputs, the intermediate carry and sum are shown in binary format in Table IV. Since the intermediate carry is always between numbers -1 and 1, it requires only a 2-bit binary representation. Finally, five 6-variable Boolean expressions can be extracted. The intermediate carry and sum circuit is shown in Figure 1.

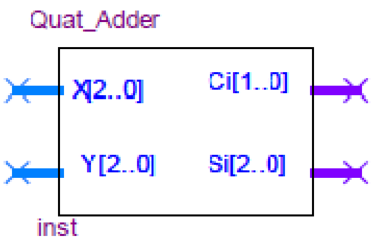


**Figure 1.** Intermediate Carry and Sum Generator

In step 2, the intermediate carry from the lower significant digit is added to the sum of the current digit to produce the final result. The addition in this step produces no carry because the current digit can always absorb the carry-in from the lower digit. Table 4 shows all possible combinations of the summation between the intermediate carry and the sum.

**TABLE 4.**
THE MAPPING BETWEEN THE INPUTS AND OUTPUTS OF THE INTERMEDIATE CARRY AND SUM

| INPUT | | | | OUTPUT | | | | |
|---|---|---|---|---|---|---|---|---|
| QSD | | Binary | | Decimal | QSD | | Binary | |
| $A_i$ | $B_i$ | $A_i$ | $B_i$ | Sum | $C_i$ | $S_i$ | $C_i$ | $S_i$ |
| 3 | 3 | 011 | 011 | 6 | 1 | 2 | 01 | 010 |
| 3 | 2 | 011 | 010 | 5 | 1 | 1 | 01 | 001 |
| 2 | 3 | 010 | 011 | 5 | 1 | 1 | 01 | 001 |
| 3 | 1 | 011 | 001 | 4 | 1 | 0 | 01 | 000 |
| 1 | 3 | 001 | 011 | 4 | 1 | 0 | 01 | 000 |
| 2 | 2 | 010 | 010 | 4 | 1 | 0 | 01 | 000 |
| 1 | 2 | 001 | 010 | 3 | 1 | -1 | 01 | 111 |
| 2 | 1 | 010 | 001 | 3 | 1 | -1 | 01 | 111 |
| 3 | 0 | 011 | 000 | 3 | 1 | -1 | 01 | 111 |
| 0 | 3 | 000 | 011 | 3 | 1 | -1 | 01 | 111 |
| 1 | 1 | 001 | 001 | 2 | 0 | 2 | 00 | 010 |
| 0 | 2 | 000 | 010 | 2 | 0 | 2 | 00 | 010 |
| 2 | 0 | 010 | 000 | 2 | 0 | 2 | 00 | 010 |
| 3 | -1 | 011 | 111 | 2 | 0 | 2 | 00 | 010 |
| -1 | 3 | 111 | 011 | 2 | 0 | 2 | 00 | 010 |
| 0 | 1 | 000 | 001 | 1 | 0 | 1 | 00 | 001 |
| 1 | 0 | 001 | 000 | 1 | 0 | 1 | 00 | 001 |
| 2 | -1 | 010 | 111 | 1 | 0 | 1 | 00 | 001 |
| -1 | 2 | 111 | 010 | 1 | 0 | 1 | 00 | 001 |
| 3 | -2 | 011 | 110 | 1 | 0 | 1 | 00 | 001 |
| -2 | 3 | 110 | 011 | 1 | 0 | 1 | 00 | 001 |
| 0 | 0 | 000 | 000 | 0 | 0 | 0 | 00 | 000 |
| 1 | -1 | 001 | 111 | 0 | 0 | 0 | 00 | 000 |
| -1 | 1 | 111 | 001 | 0 | 0 | 0 | 00 | 000 |
| 2 | -2 | 010 | 110 | 0 | 0 | 0 | 00 | 000 |
| -2 | 2 | 110 | 010 | 0 | 0 | 0 | 00 | 000 |
| -3 | 3 | 101 | 011 | 0 | 0 | 0 | 00 | 000 |
| 3 | -3 | 011 | 101 | 0 | 0 | 0 | 00 | 000 |
| 0 | -1 | 000 | 111 | -1 | 0 | -1 | 00 | 111 |
| -1 | 0 | 111 | 000 | -1 | 0 | -1 | 00 | 111 |
| -2 | 1 | 110 | 001 | -1 | 0 | -1 | 00 | 111 |
| 1 | -2 | 001 | 110 | -1 | 0 | -1 | 00 | 111 |
| -3 | 2 | 101 | 010 | -1 | 0 | -1 | 00 | 111 |
| 2 | -3 | 010 | 101 | -1 | 0 | -1 | 00 | 111 |
| -1 | -1 | 111 | 111 | -2 | 0 | -2 | 00 | 110 |
| 0 | -2 | 000 | 110 | -2 | 0 | -2 | 00 | 110 |
| -2 | 0 | 110 | 000 | -2 | 0 | -2 | 00 | 110 |
| -3 | 1 | 101 | 001 | -2 | 0 | -2 | 00 | 110 |
| 1 | -3 | 001 | 101 | -2 | 0 | -2 | 00 | 110 |
| -1 | -2 | 111 | 110 | -3 | -1 | 1 | 11 | 001 |
| -2 | -1 | 110 | 111 | -3 | -1 | 1 | 11 | 001 |
| -3 | 0 | 101 | 000 | -3 | -1 | 1 | 11 | 001 |
| 0 | -3 | 000 | 101 | -3 | -1 | 1 | 11 | 001 |
| -3 | -1 | 101 | 111 | -4 | -1 | 0 | 11 | 000 |
| -1 | -3 | 111 | 101 | -4 | -1 | 0 | 11 | 000 |
| -2 | -2 | 110 | 110 | -4 | -1 | 0 | 11 | 000 |
| -3 | -2 | 101 | 110 | -5 | -1 | -1 | 11 | 111 |
| -2 | -3 | 110 | 101 | -5 | -1 | -1 | 11 | 111 |
| -3 | -3 | 101 | 101 | -6 | -1 | -2 | 11 | 110 |

The result of addition in this step ranges from -3 to 3. Since carry is not allowed in this step, the result becomes a single digit QSD output. The inputs, the intermediate carry and sum, are 2-bit and 3-bit binary respectively. The output is a 3-bit binary represented QSD number.

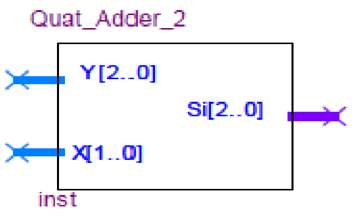


**Figure 2.** Second step of QSD adder

The implementation of an *n*-digit QSD adder requires *n* QSD carry and sum generators and *n*-1 second step adders as shown in Figure 3. The result turns out to be an *n*+1-digit number.

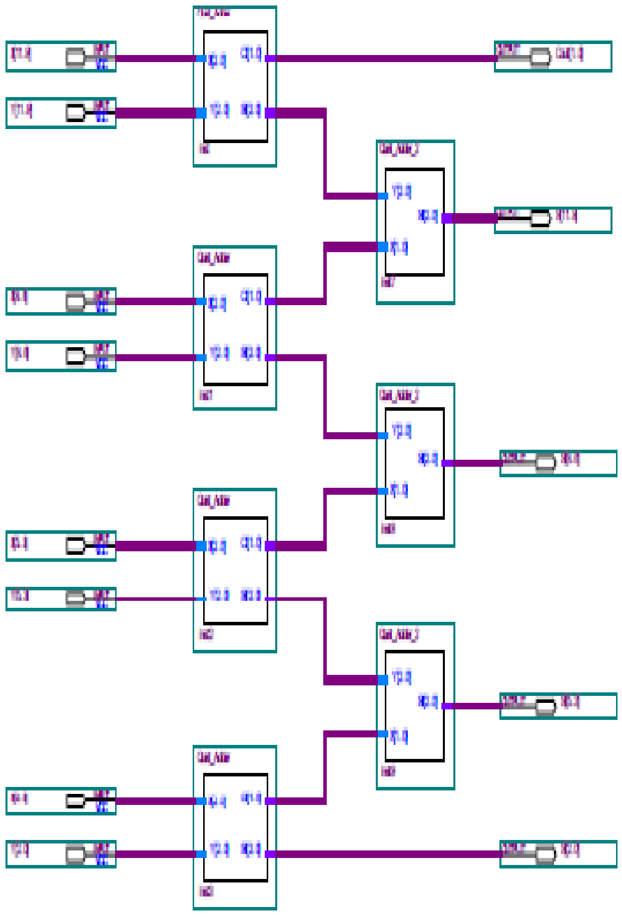


**Figure 3**. Four digit QSD adder

## VI. Simulation Results

The QSD adder written in VHDL, compiled and simulation using modelsim. The QSD adder circuit simulated and synthesized. The simulated result for 4-bit QSD adders as shown in figure 4.

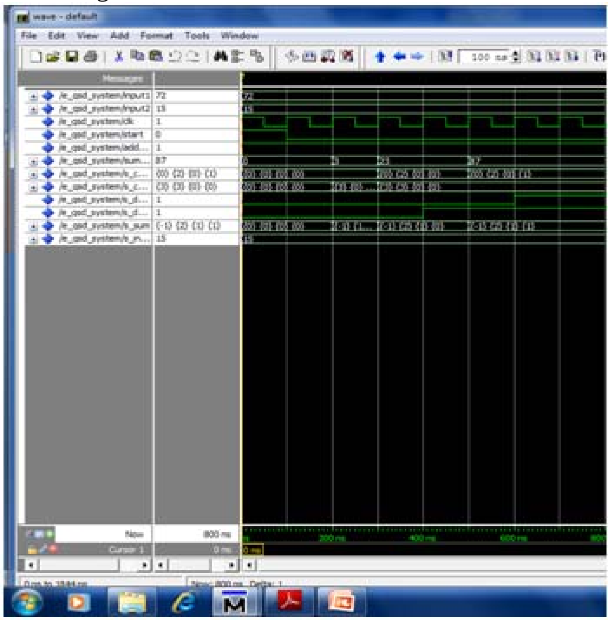


**Figure 4:** Simulated result QSD adder

## VII. Conclusion

The proposed QSD adder is better than other binary adders in terms of number of gates and higher number of bits addition within constant time. Efficient design for adder block to perform addition or multiplication will increase operation speed. QSD number uses 25% less space than BSD to store number, higher number of gates can be tolerated for further improvement of QSD adder.

## References

1. I.M. Thoidis, D. Soudris, J.M. Fernandez , A. Thanailakis, "The circuit design of multiple-valued logic voltage-mode adders," 2001 IEEE International Symposium on Circuits and Systems, pp 162-165,Vol. 4 , 2001.
2. A. Avizienis, "Signed-Digit Number Representation for Fast Parallel Arithmetic, "IRE Transaction Electron. Comp., EC-10, pp. 389-400, 1961.
3. A. K. Cherri, *"Canonical quaternary arithmetic based on optical content-addressable memory (CAM)"* Proceedings of the 1996 National Aerospace and Electronics Conference, pp. 655-661, Vol. 2, 1996.
4. *FLEX 10K Embedded Programmable Logic FamilyData Sheet*, version 4.1, http://www.altera.com, March 2001.
5. J. U. Ahmed, A. A. S. Awwal, *"Multiplier design using RBSD number system"*, Proceedings of the 1993 National Aerospace and Electronics Conference, pp. 180-184, Vol. 1, 1993.