# VLSI Implementation of Fast Addition using Quaternary Signed Digit Number System

P.Usha
*Assistant Professor*
*Department of Electronics and Communication Engineering*
*KG Reddy College of Engineering and Technology, Moinabad, Hyderabad, Telangana, INDIA.*

**Abstract- With the binary number system, the speed of arithmetic operations are limited by formation and propagation of the carry. Using quaternary signed digit (QSD) number system both carry free addition and borrow free subtraction can be achieved. The QSD number system requires a special set of prime modulo based logic for arithmetic operations. Using a high radix number system such as Quaternary Signed Digit(QSD),a carry free arithmetic operation can be achieved. Arithmetic Operations such as addition and subtraction for large numbers like 64 and 128 can be computed without the propagation of carry using QSD number system. Design is simulated and analyzed using Xilinx 13.2 ISE Simulator**.

**KEY WORDS-Carry free addition, QSD, Redundancy, VLSI**

## I. INTRODUCTION

HESE high performance adders are essential since the speed of the digital processor depends heavily on the speed of the adders used is the system. Also, it serves as a building block for synthesis of all other arithmetic operations. Adders are most commonly used in various electronic applications e.g. Digital signal processing in which adders are used to perform various algorithms like FIR, IIR etc. In past, the major challenge for VLSI designer is to reduce area of chip by using efficient optimization techniques. Then the next phase is to increase the speed of operation to achieve fast calculations like, in today's microprocessors millions of instructions are performed per second. Speed of operation is one of the major constraints in designing DSP processors[11].

The speed of the digital processors depends mostly on the speed of the adders used in the system. Most of the arithmetic operations suffers from problems like limited number of bits, circuits complexity and delay. Carry look ahead adder produces a less propagation delay, but it is limited to small number of bits due to the circuit complexity. In this paper, a high speed QSD adder is proposed which is capable of performing carry free addition and borrow free subtraction using QSD numbers. For any operand size the QSD addition/subtraction operation employs a fixed number of minterms. In QSD number System carry propagation chains are eliminated.

*Theorem*

It offers the advantage of reduced circuit complexity both in terms of transistor count and interconnections. QSD number uses 25% less space than BSD to store number [10] and it can be verified by the theorem described as under- QSD numbers save 25% storage compared to BSD To represent a numeric value N log 4N number of QSD digits and 3 log 4N binary bits are required while for the same log 2N BSD digits and 2 log 2N binary bits are required in BSD representation. Ratio of number of bits in QSD to BSD representation for an arbitrary number N is,

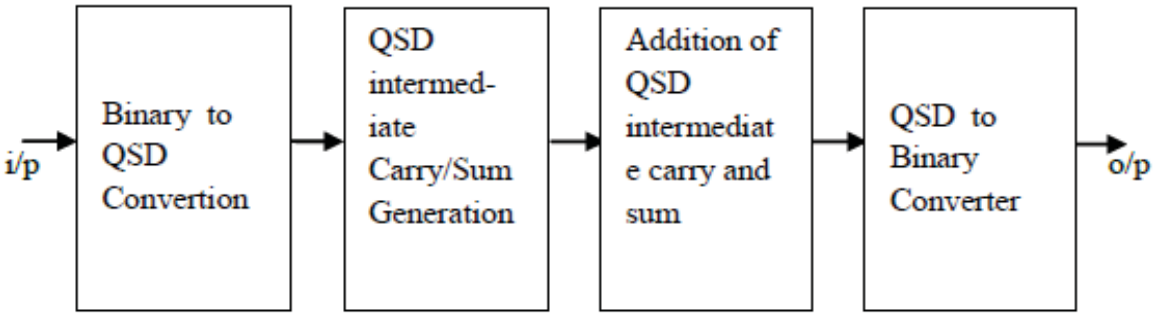



Fig1. General Block Diagram of QSD Addition

## II. PREVIOUS WORK

The speed of the digital processors depends mostly on the speed of the adders used in the system. Most of the arithmetic operations suffers from problems like limited number of bits, circuits complexity and delay. Carry look ahead adder produces a less propagation delay, but it is limited to small number of bits due to the circuit complexity. Modern computers are based on binary number system(radix =2). It has two logical states '0' and '1'. In such system, '1' plus '1' is '0' with carry '1' (i.e. 1+1=10). This carry should have to add with another '1', as a result further carry '1' generates. This creates the delay problem in computer circuits. In adders Binary Signed Digit Numbers are known to allow limited carry propagation with more complex addition process sand very large circuit for implementation. Some of the limitations of this system are computational speed which limits formation and propagation of carry especially as the number of bits increases. Therefore it provides large complexity and low storage density.

## III. PROPOSED WORK

A special higher radix-based (quaternary) representation of binary signed-digit numbers not only allows carry-free addition and borrow-free subtraction but also offers other important advantages such as simplicity in logic and higher storage density. Carry free arithmetic operations can be achieved using a higher radix number system such as Quaternary Signed Digit (QSD). In present study, QSD number system eliminates carry propagation chain which reduces the computation time substantially, thus enhancing the speed of the machine. QSD Adder or QSD Multiplier circuits are logic circuits designed to perform high-speed arithmetic operations. A higher radix based signed digit number system, such as quaternary signed digit (QSD) number system, allows higher information storage density, less complexity. A high speed area effective adders and multipliers can be implemented using this technique. The advantage of carry freeaddition offered by QSD numbers is exploited in designing a fast adder circuit. Additionally adder designed with QSD number system has a regular layout which is suitable for VLSI implementation which is the great advantage over the RBSD adder. An Algorithm for design of QSD adder is proposed. This algorithm is used to write the VHDL code for QSD adders. VHDL codes for QSD adder is simulated and synthesized and the timing report is generated.

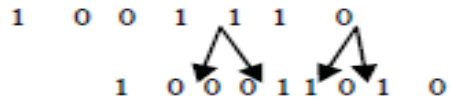
*Technique for conversion from decimal to quaternary number:*

1 digit QSD number can be represented by using a 3-bit binary equivalent are

0 = 000
1 = 001
2 = 010
3 = 011
-3= 101
-2= 110
-1= 111

The 3q-bit binary data is converted from the n-bit binary data, thus equivalent q- digit QSD data is getting from conversion of n-bit binary data. We must split the 3rd,5th,7th bit i,e odd bit(from LSB to MSB)into two portion .we

con not split the MSB bit this is major thing, for the odd bit is 0 then it is splited into 0&0 and if it is 1 then it is splited into 1&0.for example:(1001110)2.

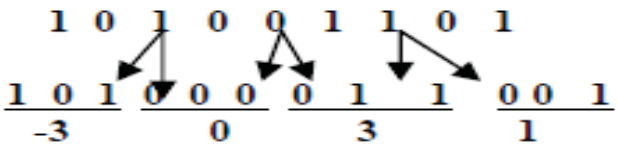



So we have to split the given binary data q-times. For example, the splitting is 1 time for conversion of a 2-digit quaternary number; the splitting is 2 times for conversion of a 3-digit quaternary number and so on. In each splitting one extra bit is generated. For conversion from binary to QSD, the required number of binary bits are,

**n = 3q{1*(n-1)}**

So, for converting binary number into its equivalent QSD the number of bits should be 3,5,7,9 etc. According to the above equation every 3-bit can be converted into its QSD.

Examples for conversion from decimal to quaternary number: 1) Let (-179)10=(101001101)2 have to be converted into its equivalent QSD. It has 9 binary bits. Its 3rd bit is 1,5th bit is 0,7th bit is 1.So from equation(2),its equivalent QSD is of 4- digit.i.e,




*Adder/Subtractor Design :*

The most important operation in the arithmetic operation of digital computation is addition operation. As the number of digits is large a carry free addition is desirable.by the exploitation of QSD number and QSD addition achieve the carry free addition. Multiple representation of any integer quantity allowed by redundancy ,i.e,610=12QSD=22QSD. In carry free addition it involves the two steps.the first step involve the to generate the intermediate carry and intermediate sum from the addend and augends. In the second step it combines the intermediate sum of the current digit with the carry of the lower significant digit. Here helps to prevent carry from further rippling, we define two rules.the first rule states that the magnitude of the intermediate sum must be equal to or less than 2.second rule states that the magnitude of the intermediate carry must be less than or equal to 1.accordling to 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[4]. In the step 2 ,the intermediate carry from the lower significant digit is added to the intermediate sum of the current digit to produce final result.the addition in this step produces no carry because the current digit can always absorb the carry-in from the lower digit.

*System Design Of Qsd Adder*

The carry propagation chains are eliminated in QSD number system which reduces the computation timesubstantially, thus enhancing the speed of the machine [31]. The QSD numbers are ranges from -3 to 3,the intermediate carry and intermediate sum are added,the addition results are varies from -6 to +6[30]. The output for all possible combination of two numbers are showen in table I. in QSD the decimal number range -3 to +3 are representing by one digit number.after the addition the result is ranges from -6 to +6,two QSD numbers are needed.in the result of two digit QSD the LSB digit represent the sum bit and the MSB digitrepresent the carry bit.this carry bit is prevented by propagating from lower digit position to higher digit position QSD number representation is used. the addition of two QSD numbers done by using two steps:

**Step 1:** in the first step to generates an intermediate carry and intermediate sum from the input QSD digits i.e., addend and augend.

**Step 2:** the Second step which helps to combines intermediate sum of current digit with the intermediate

So the addition of two QSD numbers is done in two stages. First stage of adder generates intermediate carry and intermediate sum from the input digits. Second stage of adder adds the intermediate sum of current digit with the intermediate carry of lower significant digit. To remove the further rippling of carry there are two rules to perform QSD addition in two steps:

**Rule 1:** First rule states that the magnitude of the intermediate sum must be less than or equal to 2 i.e., it should be in the range of -2 to +2.

**Rule 2:** Second rule states that the magnitude of the intermediate carry must be less than or equal to 1 i.e., it should be in the range of -1 to +1.

TABLE I
THE INTERMEDIATE CARRY AND SUM BETWEEN -6 TO +6

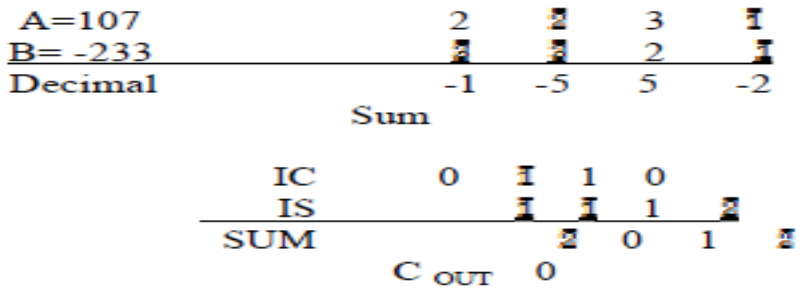
| *Sum* | *QSD represented number* | *QSD coded number* |
|---|---|---|
| -6 | $\bar{2}2, \bar{1}\bar{2}$ | $\bar{1}\bar{2}$ |
| -5 | $\bar{2}3, \bar{1}\bar{1}$ | $\bar{1}\bar{1}$ |
| -4 | $\bar{1}0$ | $\bar{1}0$ |
| -3 | $\bar{1}1, 0\bar{3}$ | $\bar{1}1$ |
| -2 | $\bar{1}2, 0\bar{2}$ | $0\bar{2}$ |
| -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 |

According to these two rules the intermediate sum and intermediate carry from the first step QSD adder can have the range of -6 to +6. But by exploiting the redundancy feature of QSD numbers we choose such QSD represented number 655 which satisfies the above mentioned two rules. When the second step QSD adder adds the intermediate sum of current digit, which is in the range of -2 to +2, with the intermediate carry of lower significant digit, which is in the range of -1 to +1, the addition result cannot be greater than 3 i.e., it will be in the range of -3 to +3[]. The addition result in this range can be represented by a single digit QSD number; hence no further carry is required. In the step 1 QSD adder, the range of output is from -6 to +6 which can be represented in the intermediate carry and sum in QSD format as shown in table I. We can see in the first column of TableI that some numbers have multiple representations, but only those that meet the above defined two rules are chosen. The chosen intermediate carry and intermediate sum are listed in the last column of Table I as the QSD coded number. This addition process can be well understood by following examples:

Example: To perform QSD addition of two numbers A = 107 and B = -233 (One number is positive and one number is negative). First convert the decimal number to their equivalent QSD representation:

$$(107)_{10} = 2\times4^3+\bar{2}\times4^2+3\times4^1+\bar{1}\times4^0$$
$$= (2\ \bar{2}\ 3\ \bar{1})_{QSD}$$
$$(233)_{10} = 3\times4^3+3\times4^2+\bar{2}\times4^1+1\times4^0$$
$$=(3\ 3\ \bar{2}\ 1)$$
$$\text{Hence } (-233)=(\bar{3}\ \bar{3}\ 2\ \bar{1})_{QSD}$$

Now the addition of two numbers can be done as follows:




The sum output is QSDwhich is equivalent to (-126)10 and carry output is 0. From these examples it is clear that the QSD adder design process will carry two stages for addition. The first stage generates intermediate carry and sum according to the defined rules. In the second stage the intermediate carry from the lower significant digit is added to the intermediate sum of current digit which results in carry free output. In this step the current digit can always absorb the carry-in from the lower digit.

*Logic Design And Implementation Using Of Single digit Qsd Adder Unit*

The QSD carry free addition involved the two steps .the first step helps to generates intermediate carry and intermediate sum from the added and augend. second step involves the to combines the intermediate sum of the current digit with the carry of the lower significant digit. To prevent from further rippling of the carry, two rules are also defined the according to first rule it states that the magnitude of intermediate sum must be equal to 2.the second rule defined as the magnitude of carry must be equal to 1.therefore ,,the magnitude of the second step output cannot be larger than the 3 which is represented by single-digit QSD number: due to this no further carry is required. The range of input number must in between -3 to +3,so the addition result must be in the range of -6 to +6 which needs two QSD digits. The lower significant digit is treated as sum and most significant bit is acts as carry. It contain one disadvantage also that is the generation of the carry can be avoided by mapping the two digit a pair of intermediate sum and intermediate carry such that the nth intermediate sum and the (n-1)th intermediate carry is never be form any carry generating pair (3,3),(3,2),(3,1),(3,3),(3,2),(3,1).

The finally we are getting the carry free addition, and here both inputs and output can be encoded in 3-bit 2's complement binary number. The intermediate carry and sum are shown in binary formate shown in Table II. At the input side, the addend Ai is represented by 3 variable input as A2, A1, A0 and the augend Bi is represented by 3 variable input as B2, B1, B0. At the output side, the intermediate carry IC is represented by IC2, IC1, IC0 and the intermediate sum IS is represented by IS2, IS1, IS0. The six variable expressions for intermediate carry and intermediate sum in terms of inputs (A2, A1, A0, B2, B1 and B0) can be derived from Table II. So we get the six output expressions for IC2, IC1, IC0, IS2, IS1 and IS0. As the intermediate carry can be represented by only 2 bits, the third appended bit IC2 is equal to IC1 so the expression for both outputs will be the same[5]. The verilog code for intermediate carry and sum generator in step 1 adder, by taking the six inputs (A2, A1, A0, B2, B1 and B0) and six outputs (IC2, IC1, IC0, IS2, IS1 and IS0), has been written. The Verilog code is compiled and simulated using Xilinx software. The design is synthesized on FPGA device xc9536xvPC44 in Xilinx XC9500XV technology. Using 6 variable K-map, the logic equations specifying a minimal hardware realization for generating the intermediate carry and intermediate sum are derived. The minterms for the intermediate carry (IC2 , IC1, IC0) are:

$IS0=a_0b_0{}^1+a_0{}^1b_0$

$IS1=(a_1b_1{}^1+a_1{}^1b_1)(a_0b_0)^1+\ (a_1b_1{}^1+a_1{}^1b_1)\ {}^1(a_0b_0)$

$IS2=IS0(a_1{}^1b_1+a_1b_1{}^1)+b_2(a_1b_0)^1+\ a_1(b_1a_0)^1+a_0b_0(a_1b_1)^1$

The minterms for intermediate carry are:

$IC_2=IC_1=(a_2b_2)(a_0b_0a_1b_1)\ {}^1+(a_1+\ b_1)\ {}^1(a_2b_0{}^1+a_0{}^1b_2)$

$IC_0=IC_2+(a_2b_2)^1\ (a_1b_1+b_1b_0+a_1b_0+a_0b_1+\ a_1a_0)$

The final sum which is carry free is generated fron
bits.

$S_0=IC_0IS_0{}^1+\ IC_0{}^1IS_0$

$S_1=IC_1\oplus\ \ IS\oplus\ \ IC_0IS_0$

$S_2=\ IC_2\oplus\ \ IS\oplus\ \ (IC_1IS_1+(IC_1+\ \ IS_1)IC_0IS_0)$

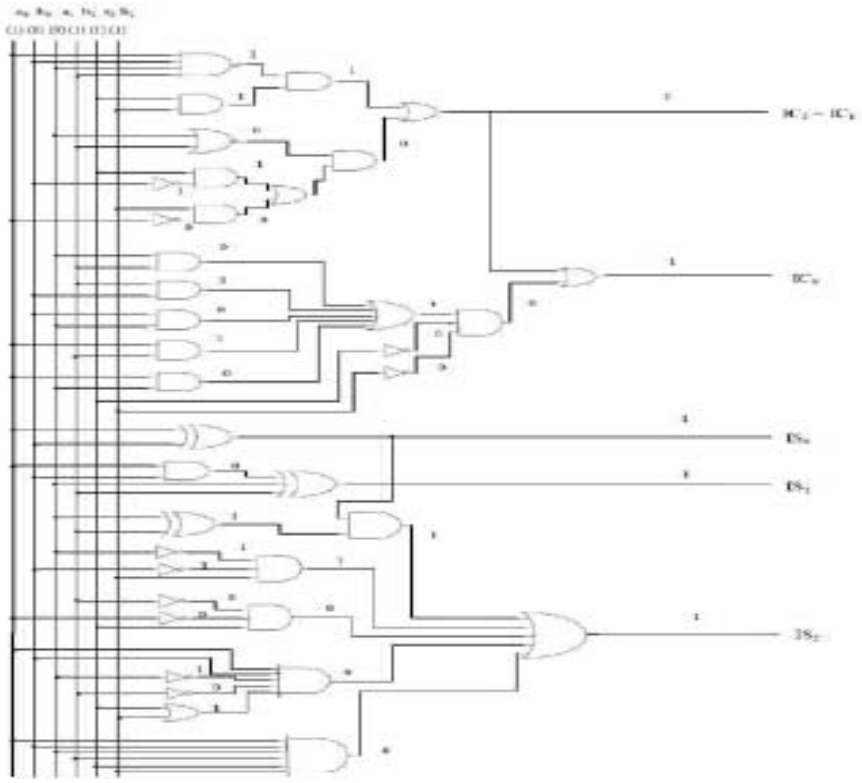


Figure 1: Data Flow of single digit QSD adder cell.

Table Ii: The Conversion Between The Inputs And Outputs Of The Intermediate Carry And Intermediate Sum

| INPUT | | | | OUTPUT | | | | |
|---|---|---|---|---|---|---|---|---|
| QSD | | Binary | | Decimal | QSD | | Binary | |
| $A_1$ | $B_1$ | $A_2$ | $B_2$ | Sum | $C_3$ | $S_3$ | $C_4$ | $S_4$ |
| 3 | 3 | 011 | 011 | 6 | 1 | 2 | 001 | 010 |
| 3 | 2 | 011 | 010 | 5 | 1 | 1 | 001 | 001 |
| 2 | 3 | 010 | 011 | 5 | 1 | 1 | 001 | 001 |
| 3 | 1 | 011 | 001 | 4 | 1 | 0 | 001 | 000 |
| 1 | 3 | 001 | 011 | 4 | 1 | 0 | 001 | 000 |
| 2 | 2 | 010 | 010 | 4 | 1 | 0 | 001 | 000 |
| 1 | 2 | 001 | 010 | 3 | 1 | -1 | 001 | 111 |
| 2 | 1 | 010 | 001 | 3 | 1 | -1 | 001 | 111 |
| 3 | 0 | 011 | 000 | 3 | 1 | -1 | 001 | 111 |
| 0 | 3 | 000 | 011 | 3 | 1 | -1 | 001 | 111 |
| 1 | 1 | 001 | 001 | 2 | 0 | 2 | 000 | 010 |
| 0 | 2 | 000 | 010 | 2 | 0 | 2 | 000 | 010 |
| 2 | 0 | 010 | 000 | 2 | 0 | 2 | 000 | 010 |
| 3 | -1 | 011 | 111 | 2 | 0 | 2 | 000 | 010 |
| -1 | 3 | 111 | 011 | 2 | 0 | 2 | 000 | 010 |
| 0 | 1 | 000 | 001 | 1 | 0 | 1 | 000 | 001 |
| 1 | 0 | 001 | 000 | 1 | 0 | 1 | 000 | 001 |
| 2 | -1 | 010 | 111 | 1 | 0 | 1 | 000 | 001 |
| -1 | 2 | 111 | 010 | 1 | 0 | 1 | 000 | 001 |
| 3 | -2 | 011 | 110 | 1 | 0 | 1 | 000 | 001 |
| -2 | 3 | 110 | 011 | 1 | 0 | 1 | 000 | 001 |
| 0 | 0 | 000 | 000 | 0 | 0 | 0 | 000 | 000 |
| 1 | -1 | 001 | 111 | 0 | 0 | 0 | 000 | 000 |
| -1 | 1 | 111 | 001 | 0 | 0 | 0 | 000 | 000 |
| 2 | -2 | 010 | 110 | 0 | 0 | 0 | 000 | 000 |
| -2 | 2 | 110 | 010 | 0 | 0 | 0 | 000 | 000 |
| -3 | 3 | 101 | 011 | 0 | 0 | 0 | 000 | 000 |
| 3 | -3 | 011 | 101 | 0 | 0 | 0 | 000 | 000 |
| 0 | -1 | 000 | 111 | -1 | 0 | -1 | 000 | 111 |
| -1 | 0 | 111 | 000 | -1 | 0 | -1 | 000 | 111 |
| -2 | 1 | 110 | 001 | -1 | 0 | -1 | 000 | 111 |
| 1 | -2 | 001 | 110 | -1 | 0 | -1 | 000 | 111 |
| -3 | 2 | 101 | 010 | -1 | 0 | -1 | 000 | 111 |
| 2 | -3 | 010 | 101 | -1 | 0 | -1 | 000 | 111 |
| -1 | -1 | 111 | 111 | -2 | 0 | -2 | 000 | 110 |
| 0 | -2 | 000 | 110 | -2 | 0 | -2 | 000 | 110 |
| -2 | 0 | 110 | 000 | -2 | 0 | -2 | 000 | 110 |
| -3 | 1 | 101 | 001 | -2 | 0 | -2 | 000 | 110 |
| 1 | -3 | 001 | 101 | -2 | 0 | -2 | 000 | 110 |
| -1 | -2 | 111 | 110 | -3 | -1 | 1 | 111 | 001 |
| -2 | -1 | 110 | 111 | -3 | -1 | 1 | 111 | 001 |
| -3 | 0 | 101 | 000 | -3 | -1 | 1 | 111 | 001 |
| 0 | -3 | 000 | 101 | -3 | -1 | 1 | 111 | 001 |
| -3 | -1 | 101 | 111 | -4 | -1 | 0 | 111 | 000 |
| -1 | -3 | 111 | 101 | -4 | -1 | 0 | 111 | 000 |
| -2 | -2 | 110 | 110 | -4 | -1 | 0 | 111 | 000 |
| -3 | -2 | 101 | 110 | -5 | -1 | -1 | 111 | 111 |
| -2 | -3 | 110 | 101 | -5 | -1 | -1 | 111 | 111 |
| -3 | -3 | 101 | 101 | -6 | -1 | -2 | 111 | 110 |

## IV. SIMULATION RESULT

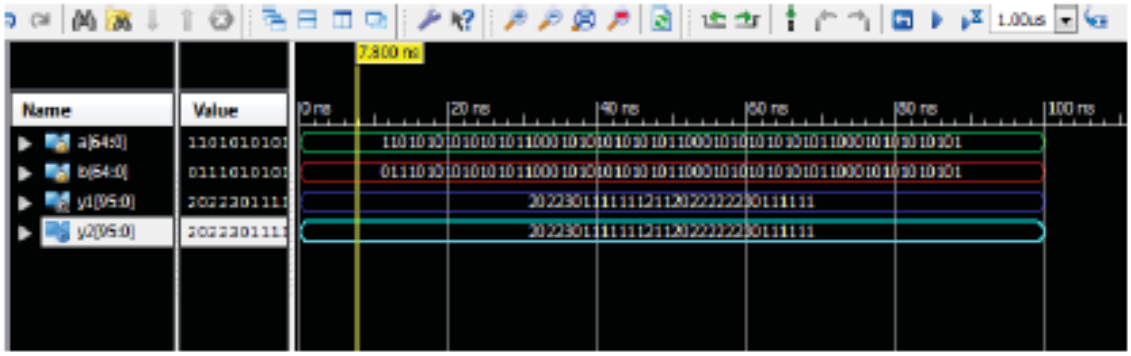



Fig 1:Simulation Result of conversion from binary to QSD

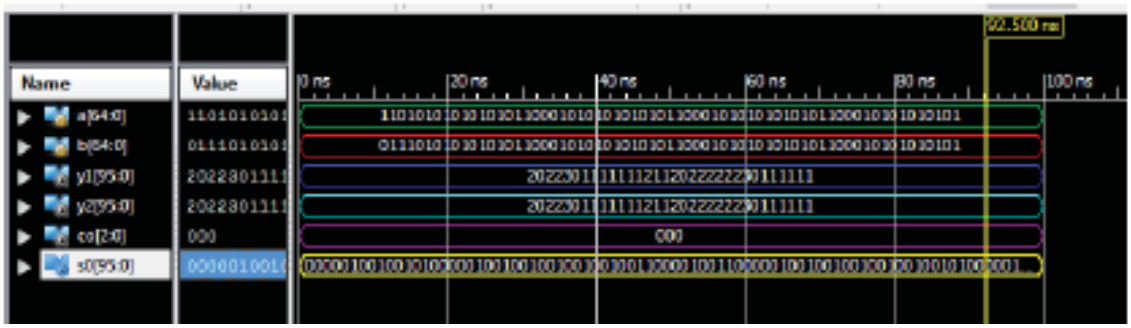



Fig.2:Simulation Result of QSD Addition

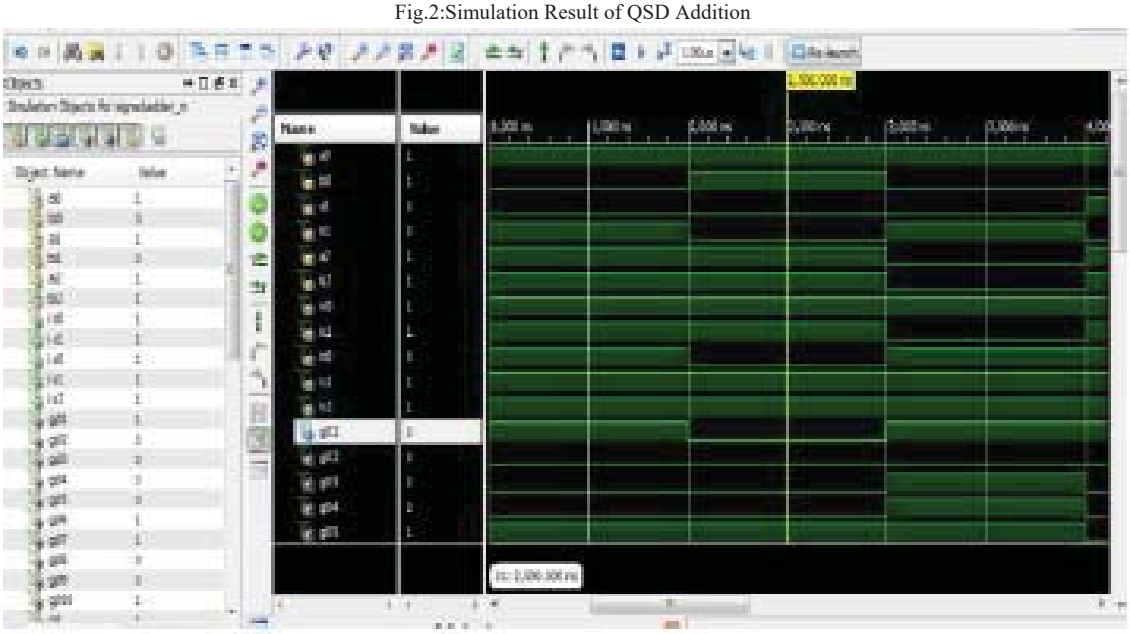


Fig 3: simulation result of single digit adder cell

## V. CONCLUSION

The proposed QSD adder is is written in verilog code and the design is simulated and synthesized using software Xilinx 13.2 ISE simulator. The QSD addition and subtraction are entrenched and proved. The QSD ALU design performance is better comparing to other designs. The QSD adder complexity is linearly proportional to the number of bits which are of the same order as the simplest BCD and other adder, such as the ripple carry adder. These QSD adders can be used as a building block for other arithmetic operations such as multiplication, division, etc. With the QSD addition scheme, some well-known arithmetic operations algorithms can be directly implemented.

## REFERENCES

[1] A. Avizinis “signed digit number representation for fast parallel arithmetic”, IRE Transactions on Elec. Comp..Vol EC-10,pp 389-400,sept-1961.

[2] A.A.S. Awwal and J.U. Ahmed, “fast carry free adder design using QSD number system ,”proceedings of the IEEE 1993 national aerospace and electronic conference, vol 2,pp 1085-1090,1993.

[3] Behrooz perhami “generalized signed digit number systems, a unifying frame work for redundant number reperesentation “.IEEE transactions on computers,vol 39,no.1,pp.89-98,January 19990.

[4] O. Ishizuka, A. Ohta, K. Tannno, Z. Tang, D. Handoko, *“VLSI design of a quaternary multiplier with direct generation of partial products,”*Proceedings of the 27th International Symposium on Multiple-Valued Logic, pp. 169-174, 1997.

[5] A.A.S Awwal, Syed M. Munir, A.T.M. Shafiqul Khalid, Howard E.Michel and O. N. Garcia, “Multivalued Optical Parallel Computation Using An Optical Programmable Logic Array”, Informatica, vol. 24, No.4, pp. 467-473, 2000.

[6] F. Kharbash and G. M. Chaudhry, “Reliable Binary Signed Digit Number Adder Design”, IEEE Computer Society Annual Symposium on VLSI, pp 479-484, 2007.

[7] John Moskal, Erdal Oruklu and Jafar Saniie, “Design and Synthesis of a Carry-Free Signed-Digit Decimal Adder”, IEEE International symposium on Circuits and Systems, pp 1089-1092, 2007.
[8] Kai Hwang, “Computer Arithmetic Principles, Architecture and Design”, ISBN 0-471-03496-7, John Wiley & Sons, 1979.
[9] P. K. Dakhole, D.G. Wakde, “ Multi Digit Quaternary adder on Programmable Device : Design and verification”, International Conference on Electronic Design, pp. 1-4, Dec 2008.
[10] Behrooz Parhami, “Carry-Free Addition of Recoded Binary Signed- Digit Numbers”, IEEE Transactions on Computers, Vol. 37, No. 11 pp.1470-1476, November 1988.
[11] Reena Rani, Neelam Sharma, L.K.Singh, “FPGA Implementation of Fast Adders using Quaternary Signed Digit Number System” IEEE proceedings of International Conference on Emerging Trends in Electronic and Photonic Devices & Systems (ELECTRO-2009), pp 132-135, 2009.
[12] Reena Rani, Neelam Sharma, L.K.Singh, “Fast Computing using Signed Digit Number System” IEEE proceedings of International Conference on Control, Automation, Communication And Energy Conservation -2009, 4th-6th June 2009.
[13] Reena Rani, L.K. Singh and Neelam Sharma, “A Novel design of High Speed Adders Using Quaternary Signed Digit Number System ”International Journal of Computer and Network Security(IJCNS), Vol. 2, No. 9, pp.62-66, September 2010.