ISSN No: 2348-4845
International Journal & Magazine of Engineering,
Technology, Management and Research
A Peer Reviewed Open Access International Journal

VLSI Implementation of Fast Addition Subtraction and Multiplication (Unsigned) Using Quaternary Signed Digit Number System

A. Leela Bhardwaj Reddy
Pg Scholar, Dept Of ECE,
PBR Visvodaya Institute Of Technology & Science
Kavali, Nellore, AP, India.

V. Narayana Reddy
Associate Professor, Dept Of ECE,
PBR Visvodaya Institute Of Technology & Science
Kavali, Nellore, AP, India.

Abstract:

The high performance Arithmetic units are essential since the speed of the digital processor depends heavily on the speed of the Arithmetic units used is the system. Digital arithmetic operations are very important in the design of digital processors and application-specific systems. Arithmetic circuits form an important class of circuits in digital systems. Adders which are a part of Arithmetic unit 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. The proposed design of QSD Arithmetic unit involves significantly less area and time complexity for higher word sizes due to fewer requirements of bits and low storage density.

With the binary number system, the computation speed is limited by formation and propagation of carry especially as the number of bits increases. Using a quaternary Signed Digit number system one may perform carry free addition, borrow free subtraction and multiplication. In QSD, each digit can be represented by a number from -3 to 3. Carry free addition and other operations on a large number of digits such as 64, 128, or more can be implemented with constant delay and less complexity.

1.INTRODUCTION

1.1 Introduction

These high performance Arithmetic units are essential since the speed of the digital processor depends heavily on the speed of the Arithmetic units used is the system. Digital arithmetic operations are very important in the design of digital processors and application-specific systems [1]. Arithmetic circuits form an important class of circuits in digital systems. 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 [2]. 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]. Some central processing units are comprised of two components, an arithmetic unit and a logic unit. Other processors may have an arithmetic unit for calculating fixed-point operations and another AU for calculating floating-point computations. Some PCs have a separate chip known as the numeric coprocessor. This coprocessor contains a floating-point unit for processing floating-point operands. The coprocessor increases the operating speed of the computer because of the coprocessor ability to perform computation faster and more efficiently. The redundancy associated with signed-digit numbers offers the possibility of carry free addition. The redundancy provided in signed-digit representation allows for fast addition and subtraction because the sum or difference digit is a function of only the digits in two adjacent digit positions of the operands for a radix greater than 2, and 3 adjacent digit positions for a radix of 2. Thus, the add time for two redundant signed-digit numbers is a constant independent of the word length of the operands, which is the key to high speed computation. The advantage of carry free addition 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 BSD adder. An Algorithm for design of QSD adder is proposed. Binary signed-digit numbers are known to allow limited carry propagation with a somewhat more complex addition process requiring very large circuit for implementation [4] [10]. 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 [14].

1.2 Literature Survey

Based on the concept of Quaternary Signed Digit Number System, Quaternary is the base-4 numeral system. It uses the digits 0, 1, 2 and 3 to represent any real number.

Normally most of the numbers deal in decimal number system. First question which will come to our mind is what is number system? Well, a number system is a system which determines the rules and symbols for numbers on how all are going to use them.

1.3 Number System

The symbols for decimal number system are Arabic but they were taken from India, presumably. A number system consists of symbols for representing numbers and a dot for representing fractional numbers. Minus sign is used to represent negative numbers. A number system ranges from $-\infty$ to $+\infty$. It is best represented by a straight line given below. As you see the stretch of number axis, it makes me wonder what infinity is. Infinity in itself is not a number to be honest in the sense that it is more of a concept. Infinity is such a large number that cannot be written or achieved by anything of physical world. Infinity is immeasurable. Each point on this axis represents a number. It may be integer or fractional number. I hope you know what an integer is. An integer is a whole number like -1, -2, 0, 5, 7 etc. Real numbers have fractional parts like 1.234. I cannot go much into these basics. The important fact to note is that between any two points there exists infinite numbers. In other words between any two numbers there exists infinite numbers. For example, between 1.2 and 1.3 there are 1.21, 1.22, 1.23..., and 1.29. Moreover between 1.21 and 1.22 there are 1.211, 1.212, 1.213 and so on. Now let us come back to number system discussion. It enables us to represent a point on this axis. The numbers I have written are supposedly in decimal number system. Base of decimal number system is 10. Why because it consists of 10 distinct symbols 0 through 9. Similarly we can have any other number system. Popular number systems in computers are binary, octal and hexadecimal not to mention decimal.

1.4 Motivation

The major challenges in VLSI design are reducing the area of chip and increasing speed of the circuit. Reducing area can be achieved by optimization techniques and number of instructions executed per second increases as speed increases. The performance of a digital system depends upon performance of adders.

Adders are also act as basic building blocks for all arithmetic circuits for example DSP processors. Binary adders are easy to implement because of logic levels involved in it ‘0’ and ‘1’, but they have their own limitations in the area of circuit complexity and chip area which ultimately increases propagation delay of the circuit. Is it time to move beyond zeroes and ones? This is the title of Bernard Cole article’s published in 2003 on the official site of the Embedded Development Community [25].

The conclusion is “I think that the economics of semiconductor manufacturing now is forcing us to move beyond zero and one. Shouldn't we also take another look at multi-valued logic?” This very thought brought many researches to work upon multi-valued logic to bring a new era of technology. This multi-valued logic is focused in this paper recognizing it as a fundamental advancement in circuit technology. Many authors have directed their efforts to the implementation of Multi-Valued logic looking for benefit from all advantages it possess over the binary logic. It is possible for ternary logic to achieve simplicity and energy efficiency in digital design since the logic reduces the complexity of interconnects and chip area, in turn, reducing the chip delay [13].

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

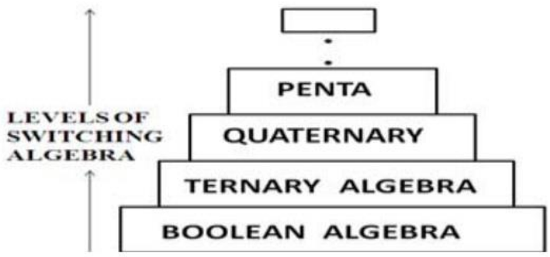

Fig 1.1 Levels of Switching Algebra

Expanding the existing logic levels are shown in Fig.1.1, higher processing rates could be achieved in various applications like memory management, communication throughput and domain specific computation. An evident advantage of a quaternary representation over binary is economy of digits. Quaternary representation admits sign convention also.The drawback of binary adders can be reduced by increasing the range of the numbers used. Signed number system can be used for this purpose. Signed digit numbers allows redundancy of numbers which allow possibility of carry-free addition, but the signed digit number system allow limited carry propagation with some complex addition process which requires large for its implementation. To overcome all these limitations Multi Valued Logic number system is used. It has advantages in many areas as high density along with increasing the speed of operation. One such number system is QSD (Quaternary Signed Digit) number system.

1.4 Objective

The objective is to design carry free adder using QSD number system to achieve fast addition with the help of Verilog, which integrates novel design of high speed QSD adder and multiplier for higher input bit sequences. The programming objective of the VLSI Implementation of fast addition using QSD number system into the following categories

- QSD Adder Unit
- QSD Multiplication Unit
- Ripple carry Adder Unit
- Synthesis Reports
- Physical designing

1.5 Thesis Organization

Chapter 2 Deals with the Overview of the Quaternary Signed Digit Number System, it's comparison with Binary Signed Digit Number System, and advantages of QSD number system. Chapter 3 also deals with the basic Rules and steps to be followed for carry free QSD addition, subtraction and multiplication. Chapter 4 deals with the Overview of the Ripple Carry Adder, Full adder and half adder.Chapter 5 deals with the work done throughout the project, description of verilog language and VCS Synopsys. Chapter 6 deals with the block diagrams, simulation results, synthesis reports of QSD adder, multiplier, and ripple carry adder, physical design reports of QSD adder.Chapter 7 summarizes the work and gives conclusion.

2.QUATERNARY SIGNED DIGIT NUMBER SYSTEM

2.1 Introduction of QSD

Quaternary is the base-4 numeral system. It uses the digits 0, 1, 2 and 3 to represent any real number. It shares with all fixed-radix numeral systems many properties, such as the ability to represent any real number with a canonical representation (almost unique) and the characteristics of the representations of rational numbers and irrational numbers. See decimal and binary for a discussion of these properties.

Relation to binary

As with the octal and hexadecimal numeral systems, quaternary has a special relation to the binary numeral system. Each radix 4, 8 and 16 is a power of 2, so the conversion to and from binary is implemented by matching each digit with 2, 3 or 4 binary digits, or bits. For example, in base 4,

$$3021_4 = 11\ 00\ 10\ 01\ 00_2.$$

Although octal and hexadecimal are widely used in computing and computer programming in the discussion and analysis of binary arithmetic and logic, quaternary does not enjoy the same status.By analogy with *bit*, a quaternary digit is sometimes called a *crumb*.

How do Base 4 work

Bases have to do with how you write numbers in a number system, and how the place values work in that system. Let's start with the system you already know. We usually work in base 10. In base 10, the place values are ones, tens, hundreds, thousands and so on. So when we see a number like 437, it really means 'four hundreds, 3 tens and 7 ones.' We understand that to be worth 'four hundred and thirty seven'. The place values are determined by raising the base to powers. In base 10, ones is 10^0, tens is 10^1, hundreds is 10^2, thousands is 10^3 and so on.When we start to count in base 10, we can write as, 1, 2, 3, 4, 5, 6, 7, 8, 9. Each of those stands for how many ones we have. The number 8 means 8 ones, or 8 * 1. But when we go past 9 to the number 10, we don't have a single digit that stands for '10 ones.' So instead, we use a two-digit number, 10, which stands for '1 ten and 0 ones.' Once we get to 99, we have reached '9 tens and 9 ones.'Going past that, we move to a three-digit number, 100, which means '1 hundred, 0 tens and 0 ones.'It's kind of hard to think about this, because your brain just does it without thinking about it, but that's what's really going on.So what happens in base 4? The place values are again given by raising 4 to powers.

4^0 = 1
4^1 = 4
4^2 = 16
4^3 = 64

So, the number 23 in base 4 is NOT worth twenty-three. It's only twenty three in base 10, where it means '2 tens and 3 ones.' In base 4, 23 (which is read as 'two-three') means '2 fours and 3 ones.' So it has a value of 2*4 + 3*1 or 8 + 3 or 11. Now think about how we count in base 4. We start with 1, 2, 3. But there is no digit '4' to use--the number 4 is written '1 four and 0 ones,' so it's 10. I know this may be confusing, but here are the numbers from 1 to 10 in base 4:

Base 10	Base 4	Meaning
1	1	(1 one) = 1*1
2	2	(2 ones) = 2*1
3	3	(3 ones) = 3*1
4	10	(1 four and 0 ones) = 1*4 + 0
5	11	(1 four and 1 ones) = 1*4 + 1
6	12	(1 four and 2 ones) = 1*4 + 2
7	13	(1 four and 3 ones) = 1*4 + 3
8	20	(2 fours and 0 ones) = 2*4 + 0
9	21	(2 fours and 1 one) = 2*4 + 1
10	22	(2 fours and 2 ones) = 2*4 + 2

Binary logic is restricted to only two logical states; Multi-Valued Logic (MVL) replaces these with finite and infinite numbers of values [1]. Multi-valued logic is a higher radix (R>2) logic system. Non-binary data requires less physical storage space than binary data [2-4]. Depending upon the radix number R, the number system are named as ternary (R = 3), quaternary (R = 4) etc. Ternary logic is based on ternary number system. Quaternary logic is based on Quaternary number system. Quaternary is the base 4 redundant number system. The degree of redundancy usually increases with the increase of the radix [24]. The signed digit number system allows us to implement parallel arithmetic by using redundancy. QSD numbers are the Signed Digit numbers with the digit set as: {-3, -2, -1, 0, 1, 2, 3} respectively.

2.2 QSD Number Representation

In general, a signed-digit decimal number D can be represented in terms of an n digit quaternary signed digit number as

$$D=\sum_{i=0}^{n-1} x_i 4^i$$

Where x_i can be any value from the set { -3, -2, -1, 0, 1, 2, 3 }for producing an appropriate decimal representation. A QSD number can be represented in Binary in 2'b (2 bit) notation for unsigned QSD number. For digital implementation, QSD numbers are represented using 3-bit 2's complement notation. A QSD negative number is the QSD complement of the QSD positive number [3]. For example, using the primes to denote complementation, we have

3 ' = -3, 3' =3, 2 ' = -2, $\overline{2'}$ = 2 , 1'= -1, $\overline{1'}$ =1.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

2.3 Examples of QSD numbers

A few examples of QSD numbers Represented in Decimal Numbers are mentioned below for better understanding the conversion of QSD Numbers

1. Decimal $(107)_{10} = 1 \times 4^3 + 2 \times 4^2 + 3 \times 4^1 + 3 \times 4^0$
   $= (1233)_4QSD$

2. Negative – $(107)_{10} = 1 \times 4^3 + \bar{2} \times 4^2 + \bar{3} \times 4^1 + 3 \times 4^0$
   $= (1\ \bar{2}\ \bar{3}\ 3)_4QSD$

3. Decimal $(98)_{10} = 1 \times 4^3 + 2 \times 4^2 + 0 \times 4^1 + 2 \times 4^0$
   $= (1202)_4QSD$

4. Negative – $(98)_{10} = 1 \times 4^3 + 2 \times 4^2 + 0 \times 4^1 + 2 \times 4^0$
   $= (1\ 2\ 0\ 2)_4QSD$

2.4 Comparison of QSD with BSD

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.

QSD numbers save 25% storage compared to BSD

To represent a numeric value N $\lceil \log_4 N \rceil$ number of QSD digits and $3\lceil \log_4 N \rceil$ binary bits are required while for the same log $2N$ BSD digits and $2\lceil \log_2 N \rceil$ binary bits are required in BSD representation. Ratio of number of bits in QSD to BSD representation for an arbitrary number N is,

$$\frac{3\lceil \log_4 N \rceil}{2\lceil \log_2 N \rceil}$$

This roughly equals to ¾. Therefore, QSD saves ¼ of the storage used by BSD. The computation speed and circuit complexity increases as the number of computation steps decreases. The computation speed mainly depends on the number of bits required to represent a number, since the less the number of bit's the easy is the computation. In general the number of bits required by a QSD number system is less when compared to BSD number system, which in turn results in better speeds and performance.

2.5 Advantages of QSD Number System

The main advantage of Quaternary logic is that it reduces the number of required computation steps for developing digital design. Furthermore memory, control unit, and processor can be carried out faster if the Quaternary logic is easily employed and memory utilization also less than binary.

These advantages have been shown to be useful for the design of Quaternary computers, for digital filtering. Quaternary representation of admits sign convention also.

- Quaternary logic is mainly applied in new transforms for encoding and more efficient for Compression, error correction, and state assignment, representation of discrete information and in automatic telephony.
- Quaternary logic also offers greater utilization of transmission channels because of the higher information content carried by every line. It gives exact and more efficient error detection and correction codes and possesses potentially higher density of information storage.

We can achieve a carry free arithmetic operation by using higher radix number system such as QSD (Quaternary Signed Digit). Signed digit number system has redundancy associated with it. The redundancy provided in signed digit number system offers the possibility of carry free arithmetic operations which in terms allows for faster processing. In signed digit representation of the system the add time for two redundant signed digit numbers is a constant indepen dent of the word length of the operands which is the key to high speed computation. Binary signed digit numbers allows limited carry propagation with a more complex addition process which requires very large circuit for implementation [1][4].

A higher radix based representation of binary signed digit numbers such as quaternary allows carry free arithmetic operations as well as it offers the important advantage of logic simplicity and storage density[5]. Quaternary logic is a promising alternative for the complex binary circuit as it will reduce the circuit area and circuit cost and power efficiency at the same time

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

3.QSD ADDITION, SUBTRACTION AND MUL TIPLICATION

3.1 Basics for QSD addition

In QSD number system carry propagation chain are eliminated which reduce the computation time substantially, thus enhancing the speed of the machine. As range of QSD number is from -3 to 3, the addition result of two QSD numbers varies from -6 to +6. Table I depicts the output for all possible combinations of two numbers. The decimal numbers in the range of -3 to +3 are represented by one digit QSD number. As the decimal number exceeds from this range, more than one digit of QSD number is required. For the addition result, which is in the range of -6 to +6, two QSD digits are needed. In the two digits QSD result the LSB digit represents the sum bit and the MSB digit represents the carry bit. To prevent this carry bit to propagate from lower digit position to higher digit position QSD number representation is used. QSD numbers allow redundancy in the number representations. The same decimal number can be represented in more than one QSD representations. So such QSD represented number which prevents further rippling of carry is chosen.

3.2 Steps for Carry free addition

To perform carry free addition, the addition of two QSD numbers can be done in two steps [4]:

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

Step 2: Second step combines intermediate sum of current digit with the intermediate carry of the lower significant digit.

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.

3.3 Rules for carry free addition

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.

By considering the Rules for carry free addition, the representation of QSD Number is shown in below table 1.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, such QSD represented number 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 3.1.

It can see in the first column of Table 3.1 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. To prevent this carry bit to propagate from lower digit position to higher digit position QSD number representation is used. QSD numbers allow redundancy in the number representations. The same decimal number can be represented in more than one QSD representations. So such QSD represented number which prevents further rippling of carry is chosen. 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 3.1. It can see in the first column of Table 3.1 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. If the representation is restricted such that the intermediate carry is limited to a maximum of 1, and the intermediate sum is restricted to be less than 2, then the final addition will become carry free. Both inputs and outputs can be encoded in 3-bit 2’s complement binary number. The mapping between the inputs,

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

add end and augend, and the outputs, intermediate carry and sum are shown in binary format in Table 3.2. This addition process can be well understood by following examples:

SUM	Possible QSD Representations	QSD Number
-6	$\bar{1}\,\bar{2},\bar{2}\,2$	$\bar{1}\,\bar{2}$
-5	$\bar{1}\,\bar{1},\bar{2}\,3$	$\bar{1}\,\bar{1}$
-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	$0\,\bar{1},\bar{1}\,3$	$0\,\bar{1}$
0	$0\,0$	$0\,0$
1	$0\,1,1\,\bar{3}$	$0\,1$
2	$1\,\bar{2},0\,2$	$0\,2$
3	$0\,3,1\,\bar{1}$	$1\,\bar{1}$
4	$1\,0$	$1\,0$
5	$1\,1,2\,\bar{3}$	$1\,1$
6	$1\,2,2\,\bar{2}$	$1\,2$

Table 3.1 QSD Number Representation for Carry free Addition

Example: To perform QSD Addition of two numbers A = 113 and B = 107 (both numbers are +'ve, and performing addition like 113+107). First convert the decimal number to their equivalent QSD representation:

$(113)_{10} = 1 \times 4^3 + 3 \times 4^2 + 0 \times 4^1 + 1\times 4^0 = (1\ 3\ 0\ 1)_{QSD}$
$(107)_{10} = 1 \times 4^3 + 2 \times 4^2 + 3 \times 4^1 + 0\times 4^0 = (1\ 2\ 2\ 3)_{QSD}$

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

```
   A = 113          1 3 0 1
   B = 107          1 2 2 3
 ______________________________
   Decimal sum         2 5 2 4

   IC     0  1  0  1
   IS        2  1  2  0
 ______________________________
   sum        3  1  3  0

   carry    0
```

The sum output is $(3\ 1\ 3\ 0)_{QSD}$ which is equivalent to $(220)_{10}$ and carry output is 0. From these examples it is clear that the QSD addition design process also will carry two stages for subtraction. The QSD representations according to these rules are shown in Table 3.1 for the range of -6 to +6. As the range of intermediate carry is from -1 to +1, it can be represented in 2 bit binary number but we take the 3 bit representation for the bit compatibility with the intermediate sum. At the input side, the addend Ai is represented by 3 variable input as A_2, A_1, A_0 and the augend Bi is represented by 3 variable input as B_2, B_1, B_0. At the output side, the intermediate carry IC is represented by IC_2, IC_1, IC_0 and the intermediate sum IS is represented by IS_2, IS_1, IS_0.The six variable expressions for intermediate carry and intermediate sum in terms of inputs (A_2, A_1, A_0, B_2, B_1 and B_0) can be derived from Table 3.2. So we get the six output expressions for, IC_2, IC_1, $IC_{0,}$ IS_2, IS_1 and IS_0.Using 6 variables K-map, the logic equations can be derived for the intermediate carry and intermediate sum. Using these equations block diagram can be designed.

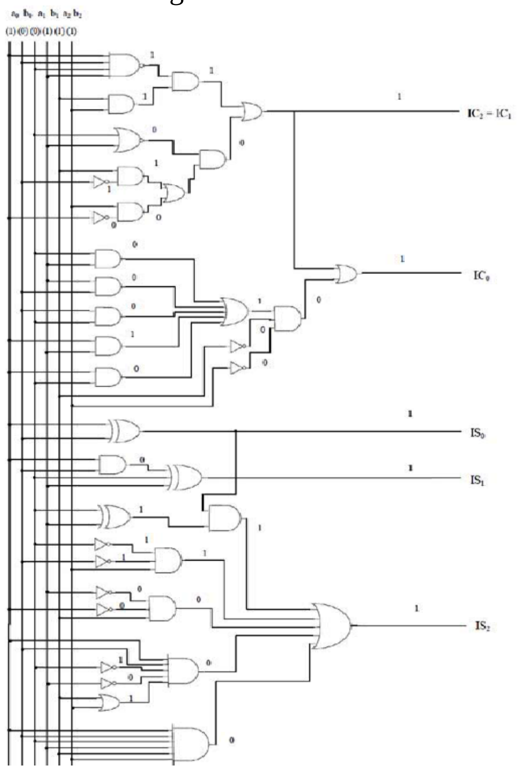

Fig 3.1 RTL schematic view of QSD adder

3.4 QSD Subtraction

The Subtraction is similar to Addition that is carried out with a negative number i.e., the two's complement of the given number. The mapping between the inputs, addend and augend, and the outputs, the intermediate

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

borrow and subtraction results are shown in binary format in Table 3.3.

In QSD number system carry propagation chain are eliminated which reduce the computation time substantially, thus enhancing the speed of the machine. As range of QSD number is from -3 to 3, the subtraction result of two QSD numbers also varies from -6 to +6. Table 3.3 depicts the output for all possible combinations of two numbers.

This subtraction process can be well understood by following examples

Example: To perform QSD Subtraction of two numbers A = 127 and B = 128 (both numbers are +'ve, and performing subtraction like 127-128). First convert the decimal number to their equivalent QSD representation:

INPUT				OUTPUT				
QSD		BINARY		DECIMAL	QSD		BINARY	
A	B	A	B	SUM	S1	S0	S1	S0
-3	-3	101	101	-6	-1	-2	111	110
-3	-2	101	110	-5	-1	-1	111	111
-3	-1	101	111	-4	-1	0	111	000
-3	0	101	000	-3	-1	1	111	001
-3	1	101	001	-2	0	-2	000	110
-3	2	101	010	-1	0	-1	000	111
-3	3	101	011	0	0	0	000	000
-2	-3	110	101	-5	-1	-1	111	111
-2	-2	110	110	-4	-1	0	111	000
-2	-1	110	111	-3	-1	1	111	001
-2	0	110	000	-2	0	-2	000	110
-2	1	110	001	-1	0	-1	000	111
-2	2	110	010	0	0	0	000	000
-2	3	110	011	1	0	1	000	001
-1	-3	111	101	-4	-1	0	111	000
-1	-2	111	110	-3	-1	1	111	001
-1	-1	111	111	-2	0	-2	000	110
-1	0	111	000	-1	0	-1	000	111
-1	1	111	001	0	0	0	000	000
-1	2	111	010	1	0	1	000	001
-1	3	111	011	2	0	2	000	010
0	-3	000	101	-3	-1	1	111	001
0	-2	000	110	-2	0	-2	000	110
0	-1	000	111	-1	0	-1	000	111
0	0	000	000	0	0	0	000	000
0	1	000	001	1	0	1	000	001
0	2	000	010	2	0	2	000	010
0	3	000	011	3	1	-1	001	111
1	-3	001	101	-2	0	-2	000	110
1	-2	001	110	-1	0	-1	000	111
1	-1	001	111	0	0	0	000	000
1	0	001	000	1	0	1	000	001
1	1	001	001	2	0	2	000	010
1	2	001	010	3	1	-1	001	111
1	3	001	011	4	1	0	001	000
2	-3	010	101	-1	0	-1	000	111
2	-2	010	110	0	0	0	000	000
2	-1	010	111	1	0	1	000	001
2	0	010	000	2	0	2	000	010
2	1	010	001	3	1	-1	001	111
2	2	010	010	4	1	0	001	000
2	3	010	011	5	1	1	001	001
3	-3	011	101	0	0	0	000	000
3	-2	011	110	1	0	1	000	001
3	-1	011	111	2	0	2	000	010
3	0	011	000	3	1	-1	001	111
3	1	011	001	4	1	0	001	000
3	2	011	010	5	1	1	001	001
3	3	011	011	6	1	2	001	010

Table 3.2 QSD Addition along with equivalent binary representations

$(127)_{10} = 1 \times 4^3 + 3 \times 4^2 + 3 \times 4^1 + 3\times 4^0 = (1\ 3\ 3\ 3)_{QSD}$

$(128)_{10} = 1 \times 4^3 + 2 \times 4^2 + 3 \times 4^1 + 0\times 4^0 = (2\ 0\ 0\ 0)_{QSD}$

Hence, $(-128)_{10} = (\bar{2}\ 0\ 0\ 0)_{QSD}$

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

A = 127		1	3	3	3
B = -128		2	0	0	0
Decimal sum		-1	3	3	3
	IC	0	1	1	1
	IS	-1	-1	-1	-1
	sum	0	0	0	-1
	carry	0			

The sum output is $(0\ 0\ 0\ 1)_{QSD}$ which is equivalent to $(-1)_{10}$ and carry output is 0. From these examples it is clear that the QSD subtractor design process also will carry two stages for subtraction.

INPUT				OUTPUT				
QSD		BINARY		DECIMAL	QSD		BINARY	
A	B	A	B	SUBTRACTION	D1	D0	D1	D0
-3	-3	101	101	0	0	0	000	000
-3	-2	101	110	-1	0	-1	000	111
-3	-1	101	111	-2	0	-2	000	110
-3	0	101	000	-3	-1	1	111	001
-3	1	101	001	-4	-1	0	111	000
-3	2	101	010	-5	-1	-1	111	111
-3	3	101	011	-6	-1	-2	111	110
-2	-3	110	101	1	0	1	000	001
-2	-2	110	110	0	0	0	000	000
-2	-1	110	111	-1	0	-1	000	111
-2	0	110	000	-2	0	-2	000	110
-2	1	110	001	-3	-1	1	111	001
-2	2	110	010	-4	-1	0	111	000
-2	3	110	011	-5	-1	-1	111	111
-1	-3	111	101	2	0	2	000	010
-1	-2	111	110	1	0	1	000	001
-1	-1	111	111	0	0	0	000	000
-1	0	111	000	-1	0	-1	000	111
-1	1	111	001	-2	0	-2	000	110
-1	2	111	010	-3	-1	1	111	001
-1	3	111	011	-4	-1	0	111	000
0	-3	000	101	3	1	-1	001	111
0	-2	000	110	2	0	2	000	010
0	-1	000	111	1	0	1	000	001
0	0	000	000	0	0	0	000	000
0	1	000	001	-1	0	-1	000	111
0	2	000	010	-2	0	-2	000	110
0	3	000	011	-3	-1	1	111	001
1	-3	001	101	4	1	0	001	000
1	-2	001	110	3	1	-1	001	111
1	-1	001	111	2	0	2	000	010
1	0	001	000	1	0	1	000	001
1	1	001	001	0	0	0	000	000
1	2	001	010	-1	0	-1	000	111
1	3	001	011	-2	0	-2	000	110
2	-3	010	101	5	1	1	001	001
2	-2	010	110	4	1	0	001	000
2	-1	010	111	3	1	-1	001	111
2	0	010	000	2	0	2	000	010
2	1	010	001	1	0	1	000	001
2	2	010	010	0	0	0	000	000
2	3	010	011	-1	0	-1	000	111
3	-3	011	101	6	1	2	001	010
3	-2	011	110	5	1	1	001	001
3	-1	011	111	4	1	0	001	000
3	0	011	000	3	1	-1	001	111
3	1	011	001	2	0	2	000	010
3	2	011	010	1	0	1	000	001
3	3	011	011	0	0	0	000	000

Table 3.3 QSD Subtraction with equivalent binary representation

The decimal numbers in the range of -3 to +3 are represented by one digit QSD number. As the decimal number exceeds from this range, more than one digit of QSD number is required. For the subtraction result,

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

which is in the range of -6 to +6, two QSD digits are needed. In the two digits QSD result the LSB digit represents the difference bit and the MSB digit represents the carry bit. To prevent this borrow bit to propagate from lower digit position to higher digit position QSD number representation is used.

3.5 QSD Multiplication

Multiplication is the third basic mathematical operation of arithmetic, the others being addition, subtraction and division. The multiplication of two whole numbers is equivalent to the addition of one of them with itself as many times as the value of the other one. The multiplication of integers (including negative numbers), rational numbers (fractions) and real numbers is defined by a systematic generalization of this basic definition. Multiplication can also be visualized as counting objects arranged in a rectangle (for whole numbers) or as finding the area of a rectangle whose sides have given lengths. The area of a rectangle does not depend on which side is measured first, which illustrates the commutative property.The QSD representation of a single digit multiplication output, shown in Table 3.4, contains a carry-out of magnitude 2 when the output is either -9 or 9. This prohibits the use of the second step QSD adder alone as a gatherer. In fact, we can use the complete QSD adder from the previous section as the gatherer. Furthermore, the intermediate carry and sum circuit can be optimized by not considering the input of magnitude 3.

This multiplication process can be well understood by following examples:

Mult	QSD represented Number	QSD coding Number
-9	$\bar{2}\bar{1},\bar{3}3$	$\bar{2}\bar{1}$
-6	$\bar{2}2,\bar{1}\bar{2}$	$\bar{1}\bar{2}$
-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
6	$12,2\bar{2}$	12
9	$21,3\bar{3}$	21

Table 3.4: QSD representation of a single-digit multiplication output

Example: To perform QSD Multiplication of two unsigned numbers A = 5 and B = 14 (both numbers are +'ve, and performing multiplication like 5*14). First convert the decimal number to their equivalent QSD representation:

$(5)_{10} = 1 \times 4^1 + 1 \times 4^0 = (1\ 1)_{QSD}$

$(14)_{10} = 3 \times 4^1 + 2 \times 4^0 = (3\ 2)_{QSD}$

Now the multiplication of two QSD numbers can be done as follows:

A = 5 1
1
B = 14 3 2

Decimal multiplication
2 2
3
3

Partial product 3
5 2

0 1
1 0
0
-1 1 2

Product
1 0 1 2
Carry 0

The output is $(1\ 0\ 1\ 2)_{QSD}$ which is equivalent to $(70)_{10}$ and carry output is 0. From these examples it is clear that the QSD product design process also will carry two stages for multiplication.

The QSD Multiplication is similar to that of normal Multiplication, but the digits here are the Quaternary Signed Digit Numbers. The single bit QSD Multiplication in Quaternary results in two bit QSD result, whereas the multiplication result lies between -9 to +9. The QSD Multiplication is shown in Table 3.5 with corresponding binary notations.

ISSN No: 2348-4845
International Journal & Magazine of Engineering,
Technology, Management and Research
A Peer Reviewed Open Access International Journal

INPUT				OUTPUT				
QSD		BINARY		DECIMAL	QSD		BINARY	
A	B	A	B	MULTIPLICATION	M1	M0	M1	M0
-3	-3	101	101	9	2	1	010	001
-3	-2	101	110	6	1	2	001	010
-3	-1	101	111	3	1	-1	001	111
-3	0	101	000	0	0	0	000	000
-3	1	101	001	-3	-1	1	111	001
-3	2	101	010	-6	-1	-2	111	110
-3	3	101	011	-9	-2	-1	110	111
-2	-3	110	101	6	1	2	001	010
-2	-2	110	110	4	1	0	001	000
-2	-1	110	111	2	0	2	000	010
-2	0	110	000	0	0	0	000	000
-2	1	110	001	-2	0	-2	000	110
-2	2	110	010	-4	-1	0	111	000
-2	3	110	011	-6	-1	-2	111	110
-1	-3	111	101	3	1	-1	001	111
-1	-2	111	110	2	0	2	000	010
-1	-1	111	111	1	0	1	000	001
-1	0	111	000	0	0	0	000	000
-1	1	111	001	-1	0	-1	000	111
-1	2	111	010	-2	0	-2	000	110
-1	3	111	011	-3	-1	1	111	001
0	-3	000	101	0	0	0	000	000
0	-2	000	110	0	0	0	000	000
0	-1	000	111	0	0	0	000	000
0	0	000	000	0	0	0	000	000
0	1	000	001	0	0	0	000	000
0	2	000	010	0	0	0	000	000
0	3	000	011	0	0	0	000	000
1	-3	001	101	-3	-1	1	111	001
1	-2	001	110	-2	0	-2	000	110
1	-1	001	111	-1	0	-1	000	111
1	0	001	000	0	0	0	000	000
1	1	001	001	1	0	1	000	001
1	2	001	010	2	0	2	000	010
1	3	001	011	3	1	-1	001	111
2	-3	010	101	-6	-1	-2	111	110
2	-2	010	110	-4	-1	0	111	000
2	-1	010	111	-2	0	-2	000	110
2	0	010	000	0	0	0	000	000
2	1	010	001	2	0	2	000	010
2	2	010	010	4	1	0	001	000
2	3	010	011	6	1	2	001	010
3	-3	011	101	-9	-2	-1	110	111
3	-2	011	110	-6	-1	-2	111	110
3	-1	011	111	-3	-1	1	111	001
3	0	011	000	0	0	0	000	000
3	1	011	001	3	1	-1	001	111
3	2	011	010	6	1	2	001	010
3	3	011	011	9	2	1	010	001

Table 3.5 QSD Multiplication with equivalent binary representations

As range of QSD number is from -3 to 3, the multiplication result of two QSD numbers also varies from -9 to +9. Table 3.5 depicts the output for all possible combinations of two numbers. The decimal numbers in the range of -3 to +3 are represented by one digit QSD number. As the decimal number exceeds from this range, more than one digit of QSD number is required. For the multiplication result, which is in the range of -9 to +9, two QSD digits are needed. In the two digits QSD result the LSB digit represents the m (0) bit and the MSB digit represents the m (1) bit. To prevent this other bit to propagate from lower digit position to higher digit position QSD number representation is used. The QSD Multiplication is shown in Table 3.5 with corresponding binary representations.

4. RIPPLE CARRY ADDER

4.1 Introduction:

Multiple full adder circuits can be cascaded in parallel to add an N-bit number. For an N- bit parallel adder, there must be N number of full adder circuits. A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. Each carry bit gets rippled into the next stage.It is possible to create a logical circuit using multiple full adders to add N-bit numbers. Each full adder inputs a Cin, which is the Cout of the previous adder. This kind of adder is called a ripple-carry adder, since each carry bit "ripples" to the next full adder. Note that the first (and only the first) full adder may be replaced by a half adder (under the assumption that Cin = 0).

4.2 Ripple Carry Adder:

The layout of a ripple-carry adder is simple, which allows for fast design time; however, the ripple-carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic.In a ripple carry adder the sum and carry out bits of any half adder stage is not valid until the carry in of that stage occurs. Propagation delays inside the logic circuitry are the reason behind this. Propagation delay is time elapsed between the application of an input and occurrence of the corresponding output. Consider a NOT gate, When the input is “0” the output will be “1” and vice versa. The time taken for the NOT gate’s output to become “0” after the application of logic “1” to the NOT gate’s input is the propagation delay here. Similarly the carry propagation delay is the time elapsed between the application of the carry in signal and the occurrence of the carry out (Cout) signal. Circuit diagram of a 4-bit ripple carry adder is shown below fig 4.1.

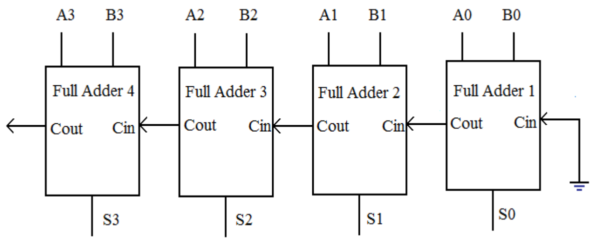

Fig 4.1: 4-bit ripple carry adder

Sum out S0 and carry out Cout of the Full Adder 1 is valid only after the propagation delay of Full Adder 1. In the same way, Sum out S3 of the Full Adder 4 is valid only after the joint propagation delays of Full Adder 1 to Full Adder 4. In simple words, the final result of the ripple carry adder is valid only after the joint propagation delays of all full adder circuits inside it.

RESULTS

6.1 QSD Adder

6.1.1 Block Diagram

Input operands at 5000 ns

dec_1 [8:0] : a = 113.
dec_2 [8:0] : b = 107.
qsd_1 [11:0] :12-bit input operand a [12:0] = 12’o1301

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

qsd_2 [11:0] :12-bit input operand b [12:0] = 12’o1223

Output operands at 5000 ns

sum [11:0] : 12-bit output operand s[11:0] = 12’b011001011000.

carr_out [2:0] : 12-bit output operand c [2:0] = 3’b000.

Input operands at 20000 ns

dec_1 [8:0] : a = -98.

dec_2 [8:0] : b = -103.

qsd_1 [11:0] :12-bit input operand a [12:0] = 12’o7606

qsd_2 [11:0] :12-bit input operand b [12:0] = 12’o7675

Output operands at 20000 ns

sum [11:0] : 12-bit output operand s[11:0] = 12’b101000110111.

carr_out [2:0] : 12-bit output operand c[2:0] = 3’b000.

Input operands at 35000 ns

dec_1 [8:0] : a = 127.

dec_2 [8:0] : b = -128.

qsd_1 [11:0] :12-bit input operand a [12:0] = 12’o1333

qsd_2 [11:0] :12-bit input operand b [12:0] = 12’o6000

Output operands at 35000 ns

sum [11:0] : 12-bit output operand s[11:0] = 12’b000000000111.

carr_out [2:0] : 12-bit output operand c [2:0] = 3’b000.

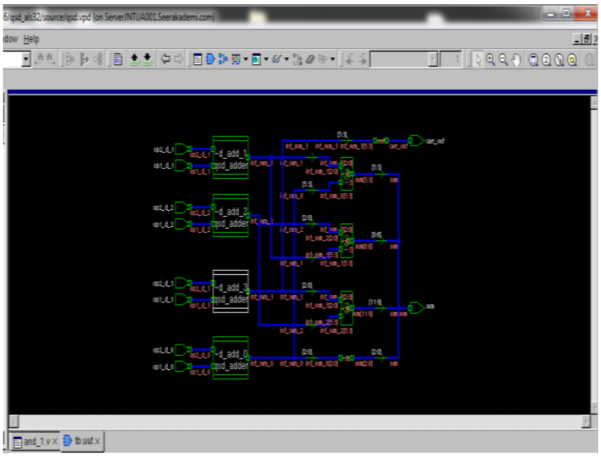

Fig: 6.1 Schematic view of QSD Adder

6.1.2 Simulation Results

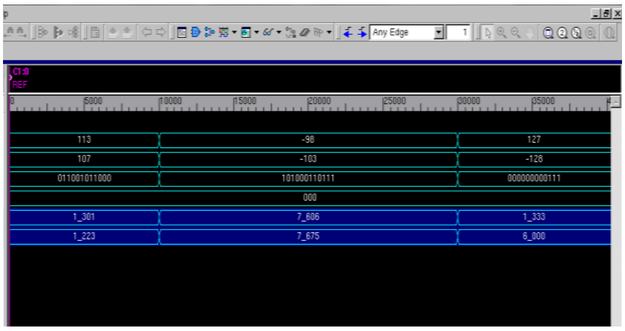

Fig 6.2 Simulation results of QSD Adder

6.2 QSD Unsigned Multiplier

6.2.1 Block Diagram

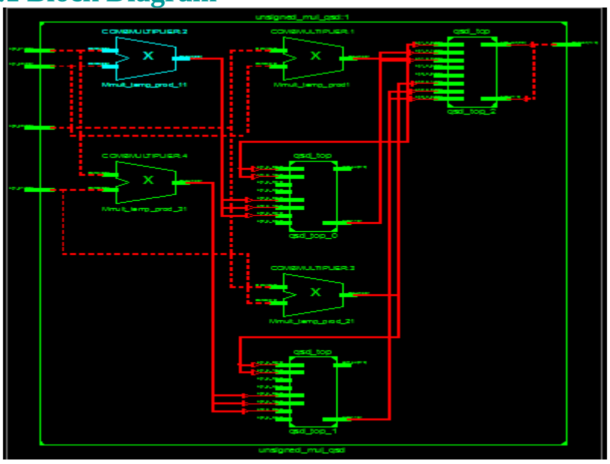

Fig: 6.3 Schematic view of QSD Unsigned Multiplier

6.2.2 Simulation results

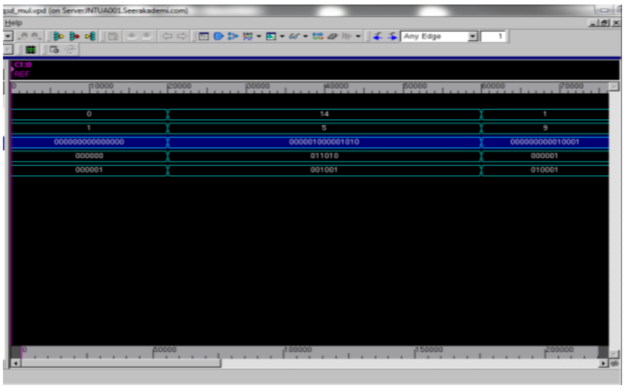

Fig 6.4 Simulation results of QSD Unsigned Multiplier

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

1. **Input operands at 10000 ns**
   dec_1 [3:0] : a = 0.
   dec_2 [3:0] : b = 1.
   mul_1 [5:0] :6-bit input operand a [5:0] = 6'b000000
   mul_2 [5:0] :6-bit input operand b [5:0] = 6'b000001
   Output operands at 10000 ns
   Prod [14:0] :15-bit output operand p [14:0] = 15'000000000000000
2. **Input operands at 40000 ns**
   dec_1 [3:0] : a = 14.
   dec_2 [3:0] : b = 5.
   mul_1 [5:0] :6-bit input operand a [5:0] = 6'b011010
   mul_2 [5:0] :6-bit input operand b [5:0] = 6'b001001
   Output operands at 40000 ns
   Prod [14:0] :15-bit output operand p [14:0] = 15'000001000001010
3. **Input operands at 70000 ns**
   dec_1 [3:0] : a = 1.
   dec_2 [3:0] : b = 9.
   mul_1 [5:0] :6-bit input operand a [5:0] = 6'b000001
   mul_2 [5:0] :6-bit input operand b [5:0] = 6'b010001
   Output operands at 70000 ns
   Prod [14:0] :15-bit output operand p [14:0] = 15'000000000010001

6.3 Ripple Carry Adder

6.3.1 Block Diagram

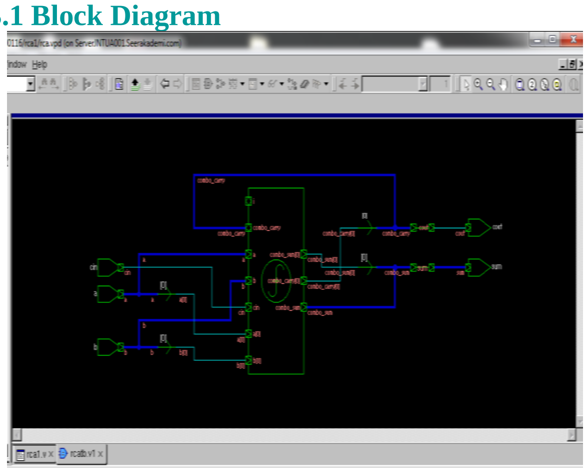

Fig 6.5 Schematic view of Ripple Carry Adder

6.3.2 Simulation Results

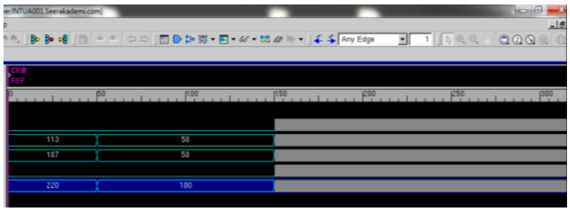

Fig 6.6 Simulation results of Ripple Carry Adder

1. **Input operands at 25 ns**
   a [11:0] : a = 113.
   b [11:0] : b = 107.
   Output operands at 25 ns
   sum [11:0] :12-bit output operand sum = 220
2. **Input operands at 100 ns**
   a [11:0] : a = 50.
   b [11:0] : b = 50.
   Output operands at 100 ns
   sum [11:0] :12-bit output operand sum = 100

7. CONCLUSION & FUTURE SCOPE

The implementation of QSD addition, subtraction and multiplication are designed, verified and compared with the Ripple carry adder. The modules are written in Verilog. These designs are simulated and synthesized using 32nm technology in VCS Synopsys tool. The test confirms the superior performance of the QSD adder implementation over other adders beyond 64-bits due to the carry-free addition scheme. The complexity of the QSD adder is linearly proportional to the number of bits which are of the same order as the simplest adder, the ripple carry adder.We can conclude that 32nm is having low power dissipation in the design of fast calculations using QSD number system. In future still lower power dissipation can be achieved without modifying and degrading the circuit functionality. Consequently this QSD adder can be used as a building block for all arithmetic operations. It can be applied for construction of a high performance multiprocessor. These high performance adders are essential in digital processors. These circuits consume less energy and power, and shows better performance. The delays of the proposed designs using QSD number system are less. 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 less space than BSD to store number higher number of gates can be tolerated for further improvement of QSD adder.

REFERENCES

[1] Sachin Dubey, Reena rani "VLSI Implementation of Fast Addition using Quaternary Signed Digit Number System" IEEE International Conference on Emerging Trends in Comp…ICECCN 2013.

[2] Neha W.Umredkar, M.A.Gaikwad, Ph.D. "Review of Quaternary Adders in Voltage Mode Multi-Valued Logic" International Journal of Computer Applications (0975 – 8887) "Recent Trends in Engineering Technology-2013".

[3] 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.

[4] 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.

[5] 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.

[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] Reilly, E.D. (2003). *Milestones in computer science and information technology*. Greenwood Press. p. 183. ISBN 1-57356-521-0.

[9] 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", Informatics, vol. 24, No. 4, pp. 467-473, 2000.

[10] 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.

[11] 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.

[12] Eurich, J.P. and Roth, G. (1990): "EDIF grows up". IEEE Spectrum, Vol. 27, Issue 11, pp. 68 - 72

[13] Behrooz perhami "generalized signed digit number systems, a unifying frame work for redundant number representation ".IEEE transactions on computers, vol 39, no.1, pp.89-98, January 1990.

[14] Behrooz Parhami, "Carry-Free Addition of Recoded Binary Signed- Digit Numbers", IEEE Transactions on Computers, Vol. 37, No. 11, pp. 1470-1476, November 1988.

[15] Barbacci, M., Grout S., Lindstrom, G., Maloney, M.P. "Ada as a hardware description language: an initial report," Carnegie-Mellon Univ., Dept. of Computer Science, 1984.

[16] Kai Hwang, "Computer Arithmetic Principles, Architecture and Design", ISBN 0-471-03496-7, John Wiley & Sons, 1979.

[17] Barbacci, M. "The Symbolic Manipulation of Computer Descriptions: ISPL Compiler and Simulator," Carnegie-Mellon Univ., Dept. of Computer Science, 1976.

[18] Barbacci, M. "A comparison of register transfer languages for describing computers and digital

systems," Carnegie-Mellon Univ., Dept. of Computer Science, March 1973.

[19] Bell, C. G.; Newell, A. (1971). *Computer Structures: Readings and Examples*. McGraw-Hill. ISBN 0-07-004357-4.

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

ABOUT AUTHORS

1.A. LEELA BHARDWAJ REDDY He received the B.TECH degree in ELECTRONICS AND COMMUNICATION ENGINEERING from JNT University, ANANTAPUR, and pursuing M.TECH degree in VLSI DESIGN from PBR VISVODAYA INSTITUTE OF TECHNOLOGY & SCIENCE, KAVALI, NELLORE, S.P.S.R.NELLORE [DT], ANDHRA PRADESH, INDIA, and Affiliated to JNT University, ANANTAPUR. He can be reachable at anam.bittu@gmail.com.

2. Mr.V NARAYANAA REDDY He received the B.TECH degree in AMIE from IEI and M.TECH degree in DIGITAL SYSTEMS AND COMPUTER ELECTRONICS (DSCE) from JNT University-ANANTAPUR. He is working as ASSOCIATE PROFESSOR in DEPT OF ECE, PBR VISVODAYA INSTITUTE OF TECHONOLOGY & SCIENCE (VITS),KAVALI, NELLORE, AP, INDIA. He can be reachable at vnr.nlr.kvl@gmail.com.