FAST ADDERS USING QUATERNARY SIGNED DIGIT NUMBER SYSTEM WITH REVERSIBLE LOGIC GATE

Neetu Thomas[1]
[1] M.Tech Scholar, Jaipur National University,
Jaipur, Rajasthan, India
neetuthomas07@gmail.com

Abstract :- **The requirement of a high speed digital circuitry has become a cardinal need as portable multimedia & communication applications that incorporates the process of computing & information processing. The issue related to latest computers has led to degradation in performance of arithmetic functions such as subtraction, addition, division & multiplication on the basis of higher consumption of power, time delay in carry propagation & complicacy in bigger circuitry. The system elaborates the carry free n digits addition/subtraction in the form of carry free propagation which is a necessary factor related to speed of digitized system. In this document, QSD numbers of which radix is 4 are implemented in arithmetic operations for executing carry free arithmetic operations. The QSD numbers ranges in between -3 to 3. In a n-digit QSD number, every digit is presented as digit set of [-3, -2, -1, 0, 1, 2, 3]. In this document, we are working over improvising the performance of QSD addition by implementation of reversible logic gate. QSD addition is implemented for 4-bit. In general circumstances, high delay performance is produced by QSD. We worked over minimizing the delay in 4 bit QSD adder. The pipelining methodology & Peres gate is implemented for minimizing the delay.**

Keyword :- *Carry free addition, Fast computing, FPGA, Quaternary Signed Digit, VHDL, VLSI*

I. INTRODUCTION

In various computers & other processors, adders are implemented in ALUs and also in other portions of processors for calculating the address, table indices & same kind of operations.

Even though adders can be generated for various numerical presentations like excess-3 or binary-coded decimal, most of the basic adders work over binary digits. In the cases were negative numbers are presented through two's or one's complement, it is necessary to transform an adder to adder-subtractor. Some other signed number presentations need a more complicated structurized adder.

A. Half Adder

The half adder adds up two single binary digits that are termed as A & B. it produces to outcomes, Carry (C) & Sum (S). This carry signal leads to an overflow in the subsequent digits of multi-digit addition. The value obtained from this sum is S + 2C. The most generalized design of half adder is displayed on the right hand side that is comprised of XOR gate for S & an AND gate for C. By adding an OR gate for combination of the carry outcomes, a full adder is made by combining two half adders.

The half adder adds two input bits and generates a carry and sum, which are the two outputs of a half adder. The input variables of a half adder are called the augends and addend bits. The output variables are the sum and carry.

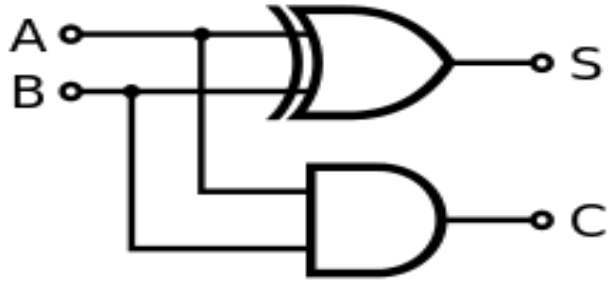

Fig 1:- Half Adder

INPUT		OUTPUT	
A	**B**	**C**	**S**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Table 1:- Half adder truth table

B. Full Adder

A full adder is implemented for adding binary numbers & accounts where values are carried in & out as well. A one-bit full adder adds up three 1-bit numbers that are termed as A, B &C_{in}; where A & B are referred as operands and C_{in} is the bit that is carried in from past least significant level. The full adder is generally a component in cascade of adders that adds up 8, 16, 32 etc. bit binary digits. This circuitry generates a 2-bit output, sum & output carry which is presented by signals C_{out} & S, where

$$sum = 2 * C_{out} + S \quad (1)$$

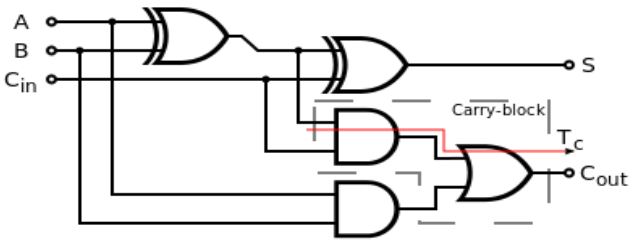

Fig 2:- Full adder

A full adder can be implemented in many different ways such as with a custom transistor-level circuit or composed of other gates. One example implementation is with

$$S = A \text{ xor } B \text{ xor } C_{in} \quad (2)$$

and

$$C_{out} = (A.B) + (Cin .(A \text{ xor } B)) \quad (3)$$

In such implementation, the final OR gate before carry-out outcome may get replaced by an XOR gate without making any transformations to resulted logic. By making use of only two forms of gates is easy if the circuitry is implemented by deployment of basic IC chips that are comprised of only a single gate type for each chip.

A full adder is formulated by combining two of the half adders by linking A & B to input of one half adder, that is linked to sum from that to an input to 2nd adder, linking C_i to other input & OR to the two carry outputs. The critical path in a full adder goes through both of the XOR gate & ends over the sum bit S. it is presumed that XOR gate needs 3 delays to get complete where delay is inflict by critical path of full adder which is equal to

$$T_{F.A} = 2 \, . \, T_{XOR} = 2.3D = 6D \quad (4)$$

The subcomponents of carry block is comprised of 2 gates & so it has a delay of

$$T_C = 2D \quad (5)$$

C. Quaternary Numeral System

Quaternary is taken as a base-4 number system. The numbers 0, 1, 2 & 3 are used to present any of the number. The number 4 is the maximum digit in subsidizing range and one of the number which is a high composite number & a square as well (as other is taken as 36), that makes quaternary as a rational selection as a base over this scale. Since it is double in the size than binary, still radix economy of both is same. Though, it doesn't work fine in localizing prime numbers (further best is the primordial base six, senary).

Various kind of characteristics like capability of presenting a real number having canonical presentation (that is almost unique) & presentation of rational & irrational numbers are been shared by the quaternary with all of the fixed radix numerical systems. Look over binary & decimal topic for explanation of these characteristics.

Arithematic operations have a vital role in several digitized systems like process controllers, computers, image processing & signal processors computer graphics. Latest improvisations in the techniques of integrated circuitries made the bigger circuits based over arithmetic functions to be implemented over VLSI [1]. Though such arithmetic functions are still dealing with issues like limited bits, time delay in propagation & complicacy in circuit. Now, flexibility in FPGAs has also supported enhancement in customized hardware giving high ratio of performance. By choosing arithmetic algorithms that suits FPGA technology & implementing optimal mapping techniques, high performance FPGA application ban be produced [7].

High speed QSD arithmetic and logical unit that has the ability of carry free addition, borrow free subtraction, multiplication functions & up-down count. The operation of addition and subtraction by QSD incorporates a defined number of inters for any of the size of an operand. The signed digit number system provides the chances of carry free addition. QSD Adder/ QSD Multiplier circuitries are designed in order to execute arithmetic functions at a high speed. In the number system of QSD, carry chain propagation is eliminated that further leads to minimization of computational time. It further improvises speed of machine [2].

D. Quaternary Signed Digit Numbers

QSD digits are present by making use of 3-bit 2's complement notation. Each digit is presented by:

$$D = \sum_i^n x_i 4^i \quad (6)$$

Here x_i can be taken as any of the value from the set of {3, 2, 1, 0, 1, 2, 3} for generating a suitable decimal representation. A QSD negative number is obtained by complementing a QSD positive number which means the 3 = -3, 2 = -2, and 1 = -1. For digitized implementation, bigger sized digits like 64, 128 and more can be utilized trough a consistent delay. A signed digit number system constituted over higher radix value like QSD helps in more information storage density, low complicacy, less system constituents & low cascading gates & operations. This methodology helps in implementation of area effective & high speed adders & multipliers.

For Example

$$(1\overline{233})_{QSD} = (23)_{10}$$
$$= 1*4^3 + \overline{2} * 4^2 + \overline{3} * 4^1 + 3 * 4^0$$
$$= 64-32-12+3 = 23$$
$$= (\overline{1}\ 2\ 3\ \overline{3})_{QSD} = (-23)_{10}$$

D. Adder Design

A faster Arithmetic set helps in elaboration of application domain for faster multipliers in processing of digitized signals, modular exponential, matrix inversion, calculation of Eigen values, digit filters etc. Square roots, calculation of reciprocals, inverse square toots & other elementary functions through implementation of small sized tables, small multipliers & for few functions, a final multiplication is obtained. Addition is taken as a cardinal operation in the field of digital computing. As the size of digits become large, a need of carry free addition is there. This carry free addition can be obtained by exploitation of redundancy in QSD numbers & QSD addition.

$$(6)_{10} = (12)_{QSD} = (2\overline{2}) \quad (7)$$

Two steps are involved in the carry free addition. In the first step, an intermediate carry & sun from augends & addend is produced. In the next step, intermediate sum of present digit & carry from lower significant digit is combined [3] [6]. There are two rules defined for prevention of carry from rippling. As per the first rule, value of magnitude of intermediate sum must be equal to or less than 2. As per the 2nd rule, value of magnitude of carry should be equal or less that 1. Further, magnitude of outcome obtained in the second step must not be higher than 3 that is presented by a single digit QSD number. Thus, no additional carry is needed. In the 1st step, the possible input pairs of augends & added are taken into account.

Sum	QSD representation	QSD coded number
-6	$\overline{2}\,2\,,\overline{1}\overline{2}$	$\overline{1}\overline{2}$
-5	$\overline{2}3,\overline{1}\overline{1}$	$\overline{1}\overline{1}$
-4	$\overline{1}0$	$\overline{1}0$
-3	$\overline{1}1,0\overline{3}$	$\overline{1}1$
-2	$\overline{1}2,0\overline{2}$	$0\overline{2}$
-1	$\overline{1}3,0\overline{1}$	$0\overline{1}$

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	$12,2\bar{3}$	12

Table 2 :- The Intermediate Carry And Sum Between -6 To 6

Both of the inputs & outcomes can be encoded in 3-bit 2's complement binary number type. Mapping in the addend, augends, inputs & outputs, sum & intermediate carry are presented in binary form Table 2. As the value of intermediate carry remains in between -1 & 1, only 2-bot binary presentation is needed. At last, five 6-variable Boolean expressions can be extracted.

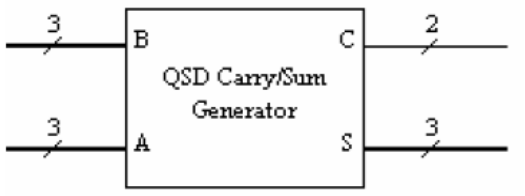

Fig 3. The intermediate carry and sum generator

In the 2nd step, sum generator & intermediate carry from lower significant digit is added to sum of current digit for generating the last final outcome no carry is produced by the addition in this step as carry-in coming from lower digit s absorbed by present digit. All the possible sets of summation generated in between sum & intermediate carry are presented.

Fig 4. The second step QSD added.

It is possible to extract three 5-variable Boolean expressions. N-QSD carry & sum generators along the n-1 second step adders is needed for implementing an n-digit QSD adder that is presented in the outcomes which comes out to be an n+1 digit number. The outcome obtained by this addition lie in between -3 to 3. As taking a carry is not permitted in this step, the outcome is a single digit QSD output. The inputs are, intermediate carry & sum that are 2-bit & 3-bit binary respectively. The outcome obtained is 3-bit binary representation of QSD number [4] [5].

It is possible to extract three 5-variable Boolean expressions. the implementation of the Boolean equations is done with the help of FPGA tools. The compilation & simulation of Modelism and Leonardo spectrum design is performed. There are two steps in the addition for design of adders. The 2nd step adder is presented in figure 2. N-QSD carry & sum generators along the n-1 second step adders is needed for implementing an n-digit QSD adder that is presented in figure 3. The outcome obtained is n+1 digit number.

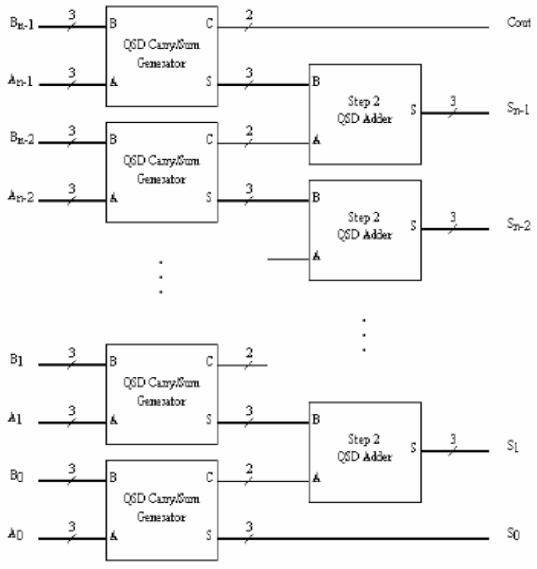

Fig 5 :- n-digit QSD adder

II. PROBLEM STATEMENT

The restraints of speed in the latest computers for performing the arithmetic functions like addition, subtraction & multiplication leads tp carry propagation delay. Arithmetic operations of carry free form can be attained through implementation of higher radix number system like QSD (Quaternary Signed Digit). In the base document, fast adders that are constituted over QSD systems. In QSD, every digit is presented by a number lying in between -3 and 3. Carry free addition & other functions over large digits like 64, 128 and more can be applied with consistent delay & low level of complicacy. The designing of such circuitries is conducted through FPGA tools. Simulation of the design is performed by making use of modelism software & synthesized by Leonardo Spectrum. Delay is obtained from 6ns in base paper. Delay from the base paper is obtained as 6ns. The main problem is related to area & delay as mentioned in base paper. We worked for minimizing the area & delay in circuitry described in base paper.

III. PROPOSED METHODLOGY

QSD works as an addition that is carry free and is implemented for enhancing the speed of system. Generally, QSD is implemented for carry & sum till now in a basic full adder for the purpose of addition.

A. Methodology 1

The Peres gate constituted over full adder is implemented for improving the performance under condition of delay. We are designing a full adder Peres gate based over the reversible logic.

The input vector is considered to be I (A, B, C) & the vector obtained for output is O (P, Q, R). The outcome is presented by P = A, Q = AB & R = AB C. The cost of Quantum for Peres gate is 4. In the suggested design Peres gate is implemented as it has minimal quantum cost.

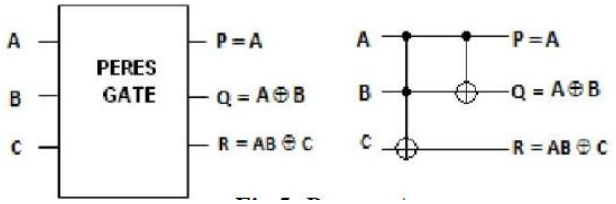

Fig 6 :- Peres Gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	1	1	0	0

Table 3 :- Peres gate Truth Table

A full-adder that applies two Peres gates as presented in the diagram. The quantum realization of this presents that cost of quantum is 8 where to Peres gates are applied. A 4*4 reversible gate termed as PFAG with quantum cost of 8 is applied to form the multiplier.

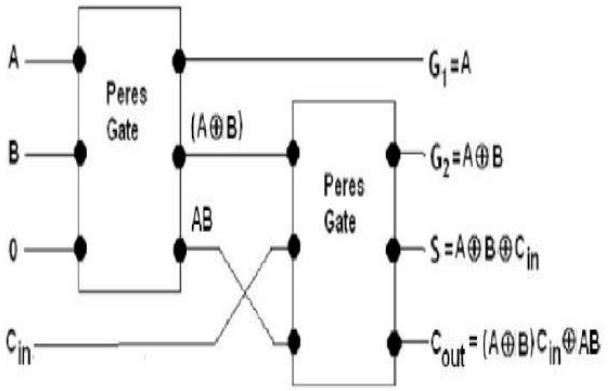

Fig 7 :- Full adder using two Peres gates

B. Methodlogy2

Here the method of pipelining is implemented for minimizing the delay in the circuitry. Initially, the pipelining is applied in QSD carry sum generator & the applied in bit transfer values for purpose of sum & carry. In this process we are able for transferring multiple clocks through only one clock.

IV. RESULTS

A. Existing Design

This existing design is a generalized design that performs basic operations of addition for adding up the bits. The RTL schema for this existing design is presented in the diagram

In the present design, the inputs of a1, a2, a3, a4 are applied in the form of 1, 0, 3, 1 & b1, b2, b3, b4 in the form of 3, 2, 2, 1. As observed from the session of waveform the outcome is generated in the two forms i.e. carry & sum. In the operation of addition sum will be 3130 where carry is 0.

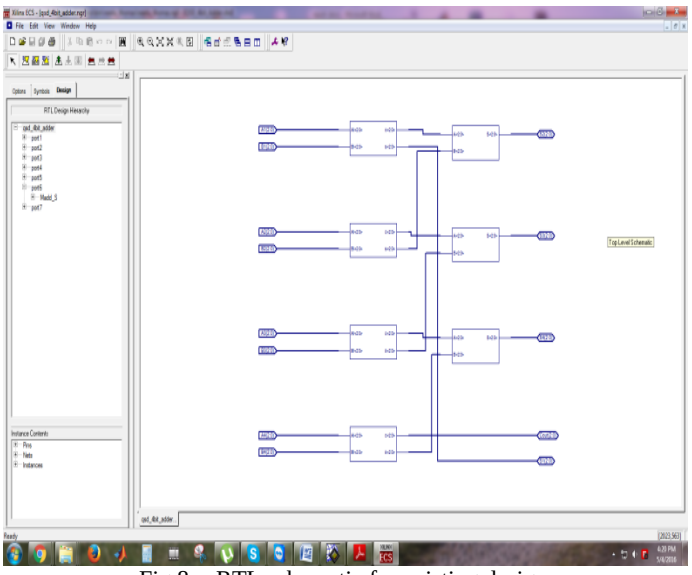

Fig 8 :- RTL schematic for existing design

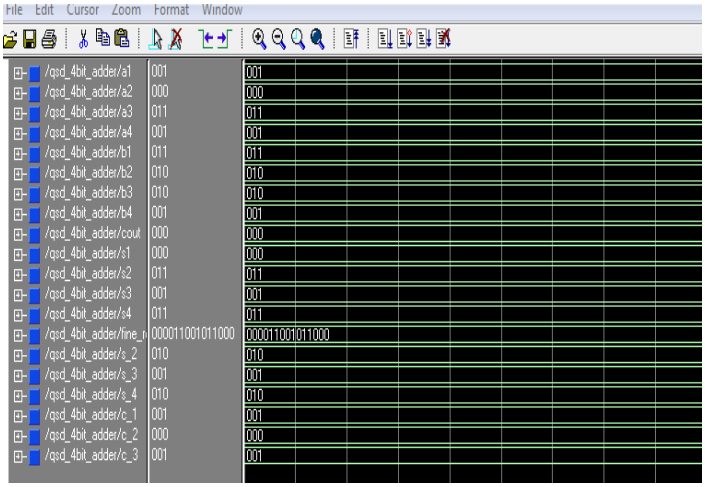

Fig 9 :- waveform generation for Existing design

Here final_result will be 000011001011000. On the side of input, 4 bits are provided where outcome received is also of 4 bits along with a carry. In the present design delay is 15.449 ns. As observed from the image the delay is 15.449 ns.

```
--------------------------------------------------------------------------------
Timing constraint: Default path analysis
Delay:              15.449ns (Levels of Logic = 9)
  Source:           B3<0> (PAD)
  Destination:      S4<2> (PAD)

  Data Path: B3<0> to S4<2>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             14   1.492   0.925  B3_0_IBUF (B3_0_IBUF)
     LUT3:I1->O             1   0.720   0.240  port3_Ker262663 (CHOICE700)
     LUT3:I0->O             1   0.720   0.240  port3_Ker262666 (CHOICE701)
     LUT4:I0->O             2   0.720   0.465  port3_Ker262687 (CHOICE703)
     LUT4:I3->O             2   0.720   0.465  port3_Ker2626120 (N7041)
     LUT2:I0->O             2   0.720   0.465  port3__n0048<3>60 (C_3<0>)
     LUT4:I0->O             1   0.720   0.240  port7_Madd_S_Mxor_Result<2>_Xo<1>_SW1 (N8923)
     LUT4:I3->O             2   0.720   0.465  port7_Madd_S_Mxor_Result<2>_Xo<1> (Fine_Result_11_OBUF)
     OBUF:I->O                  5.412          S4_2_OBUF (S4<2>)
    ----------------------------------------
    Total                     15.449ns (11.944ns logic, 3.505ns route)
                                       (77.3% logic, 22.7% route)
```

Fig 10 :- Delay report for the Existing design

B. Proposed Methodology

In the suggested design, reversible logic gates based over Peres are implemented for minimization of delay. As RTK schema for the current system is presented in the figure.

In the suggested design, the inputs of a1, a2, a3, a4 are applied in the form of 1, 0, 3, 1 & b1, b2, b3, b4 in the form of 3, 2, 2, 1. As observed from the session of waveform the outcome is generated in the two forms i.e. carry & sum. In the operation of addition sum will be 3130 where carry is 0. Hence final_result will be 000011001011000. On the side of input, 4 bits are provided where outcome received is also of 4 bits along with a carry. In the present design delay is 15.449 ns. As observed from the image the delay is 15.449 ns.

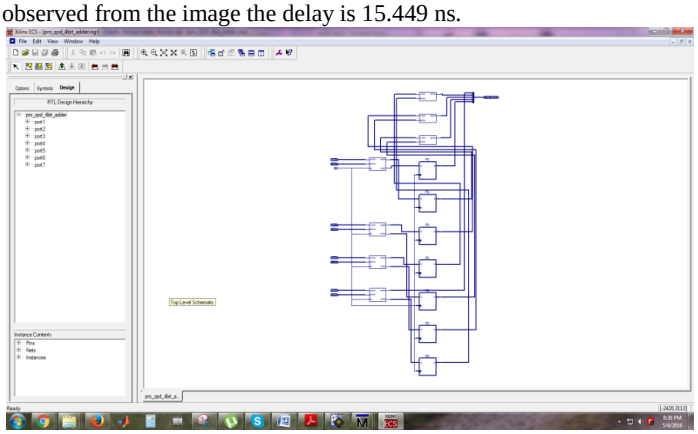

Fig 11 :- RTL schematic for Proposed design

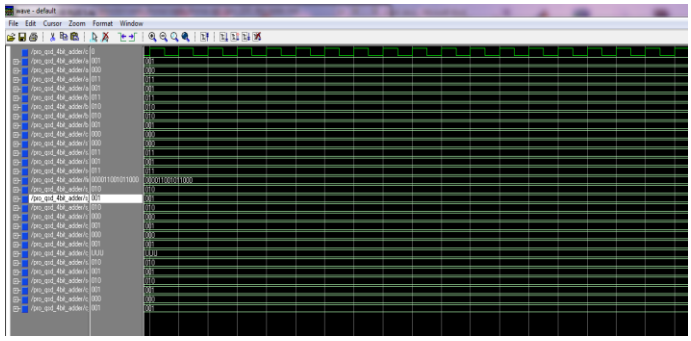

Fig 12 :- waveform generation for Proposed design

```
--------------------------------------------------------------------------------
Timing constraint: Default period analysis for Clock 'clk'
Delay:               1.361ns (Levels of Logic = 0)
  Source:            port4_s_2 (FF)
  Destination:       S4_reg_2 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: port4_s_2 to S4_reg_2
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDS:C->Q              1   0.619    0.240  port4_s_2 (port4_s_2)
     FD:D                      0.502           S4_reg_2
    ----------------------------------------
    Total                      1.361ns (1.121ns logic, 0.240ns route)
                                       (82.4% logic, 17.6% route)
```

Fig 13 :- Delay report for the Proposed design

	Delay(ns)
Existing Design	15.449 ns
Proposed Design	1.361 ns

Table 4 :- Delay comparison

V. CONCLUSION & FUTURE SCOPE

The suggested QSD along the reversible logic gates has a better scope than the binary addresses for the purpose of low addition of delay. The operational speed will be increased by an efficient design for implementing it for purpose of multiplication or addition. The suggested fast speed adders constituted over quaternary signed digit number system along the reversible logic gates. In the QSD every digit is presented by a number from -3 to 3. The process of carry free addition & other functions works over large quantity of digits like 64, 128 or further more that are applied by a constant delay & low level of complicacy. The design of this circuitry is presented through FPGA tools. These designs are simulated by applying modelism software & are formulated by XILINX.

In VLSI we implement Reversible logic gates for improvising the performance for power of the circuitry. The delay, power & area get minimized after implementation of reversible logic gates in the QSD. We further implement Peres gates for minimizing the delay & area. The delay in a general QSD will be 15.449ns & it becomes 1.361ns after implementation of suggested methodology.

In this research, we are working over enhancing the adder performance in the direction of development for digital design by applying reversible logic circuitries that acts as a support for quantum computing, CMOS with less power, cryptography, nanotechnology, DNA computing, optical computing, DSP (digital signal processing), communication, quantum dot cellular automata, computer graphics etc.

REFERENCES

[1] Chao Cheng; Parhi and K.K. “High-Throughput VLSI Architecture for FFT Computation”, IEEE Transactions on Volume 54, Issue 10, Page 863 – 867, October 2007.

[2] 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.

[3] Hwang K. (1979) Computer Arithmetic Principles Architecture and Design. New York : Wiley.

[4] N. Takagi, H. Yasuura, and S. Yajima, “High Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree, ” IEEE Trans. Comp., C-34, pp. 789-795, 1985.K. Elissa, “Title of paper if known,” unpublished.

[5] 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.

[6] 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.

[7] A. Avizienis, “Signed-Digit Number Representation for Fast Parallel Arithmetic, ”IRE Transaction Electron. Comp., EC-10, pp. 389-400, 1961.

[8] M. E. Louie and M. D. Ercegovac,” On Digit -Recurrence Division Implementations for Field Programmable gate Arrays” In Proc. Of the 11th symposium on Computer Arithmetic, PP. 202-209, Canada.