OPEN ACCESS

International
Journal
Of Modern Engineering Research (IJMER)

VLSI Design of Fast Addition Using QSD Adder for Better Performance

G. Jayaprakash[1], B. Adinarayana[2]
[1,2] *PG Student, Assistant Prof, Santhiram Engineering College, Nandyal.*

Abstract: *The high speed digital circuits became more prominent with incorporating information processing and computing. Arithmetic circuits play a very critical role in both general-purpose and application specific computational circuits. The modern computers lead to the deterioration in performance of arithmetic operations such as addition, subtraction, multiplication, division, on the aspects of carry propagation delay, large circuit complexity and high power consumption. Designing this adder using QSD number representation allows fast addition/subtraction which is capable of carry free addition and borrows free subtraction because the carry propagation chain are eliminated, hence it reduce the propagation time.*

Keywords: *QSD, carry/borrow free addition/ subtraction.*

I. Introduction

Arithmetic operations are widely used and play important role in various digital systems such as computers and signal processors. Designing this Arithmetic unit using QSD number representation has attracted the interest of many researchers. Additionally, recent advances in technologies for integrated circuits make large scale arithmetic circuits suitable for VLSI implementation. In this paper, we propose a high speed QSD adder which is capable of carry free addition, borrow free subtraction. The QSD addition/subtraction operation employs a fixed number of minterms for any operand size. In QSD number system carry propagation chain are eliminated which reduce the computation time substantially, thus enhancing the speed of the machine. Signed digit number system offers the possibility of carry free addition. QSD Adder / QSD Multiplier circuits are logic circuits designed to perform high-speed arithmetic operations. In QSD number system carry propagation chain are eliminated which reduce the computation time substantially, thus enhancing the speed of the machine.

II. Signed Digit Number

Signed digit representation of number indicates that digits can be prefixed with a – (minus) sign to indicate that they are negative. Signed digit representation can be used to accomplish fast addition of integers because it can eliminate carriers.

$$(1\bar{1}\bar{2}2)_2 = 1\times2^3 + 1\times2^2 - 2\times2^1 + 1\times2^0$$
$$= 8+4-4+2$$
$$= 10$$

III. QSD Number System

QSD numbers are represented using 3-bit 2's complement notation. Each number can be represented by

$$D = \Sigma xi\, 4^i$$

Where xi can be any value from the set {3, 2, 1, 0, 1, 2,3} for producing an appropriate decimal representation. A QSD negative number is the QSD complement of QSD positive number i.e.3 = -3, 2 = -2, 1 = -1. For digital implementation, large number of digits such as 64, 128, or more can be implemented with constant delay. A high speed and area effective adders and multipliers can be implemented using this technique. For digital implementation, large number of digits such as 64, 128, or more can be implemented with constant delay. A higher radix based signed digit number system, such as quaternary signed digit (QSD) number system, allows higher information storage density, less complexity, fewer system components and fewer cascaded gates and operations. A high speed and area effective adders and multipliers can be implemented using this technique.
Also we can obtain redundant multiple representation of any integer Quantity using this QSD number system Examples of n digit QSD number are as follows: etc. 3 1021310,1123,0112, 0132

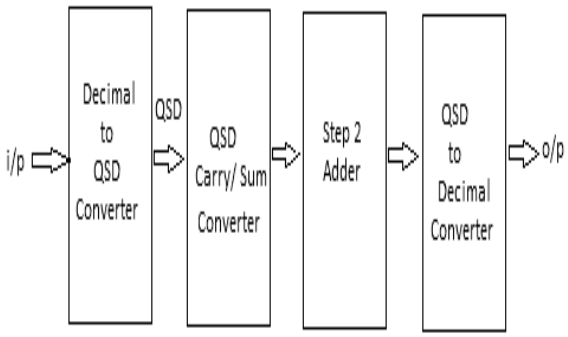
For example:

$$(1\bar{3}\bar{2})_{QSD} = 1\times4^3 + 3\times4^2 - 3\times4^1 + 2\times4^0$$
$$= 64+48-12+2$$
$$= (102)_{10}$$

The basic quaternary operators are very similar to binary operators and they are obtained from Boolean algebra.

IV. Basic Concept

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



V. Adder/Subs Tractor Design

In arithmetic operation of digital computation addition is the most important operation. A carry-free addition is desirable as the number of digits is large. The carry-free addition can achieve by exploiting redundancy of QSD number and QSD addition. The redundancy allows multiple representations of any integer quantity i.e., 610 = 12QSD = 22QSD. There are two steps involved in the carry-free addition. The first step generates an intermediate carry and sum from the addend and augends. The second step combines the intermediate sum of the current digit with the carry of the lower significant digit. To prevent carry from further rippling, we define two rules. The first rule states that the magnitude of the intermediate sum must be less than or equal to 2. The second rule states that the magnitude of the carry must be less than or equal to 1.Consequently, the magnitude of the second step output cannot be greater than 3 which can be represented by a single-digit QSD number; hence no further carry is required. In step 1, all possible input pairs of the addend and augends are considered [4]. The output ranges from -6 to 6 as shown in Table 1.

Table 1. The outputs of all possible combinations of a pair of addend (A) and augend (B)

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

The range of the output is from - 6 to 6 which can be represented in the intermediate carry and sum in QSD format as show in Table 2.Some numbers have multiple representations, but only those that meet the defined rules are chosen. The chosen intermediate carry and sum are listed in the last column of Table 2. Both inputs and outputs can be encoded in 3-bit 2's complement binary number.

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

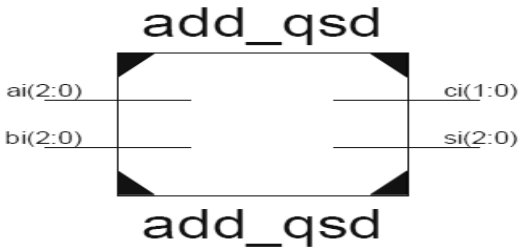


Figure 1. The intermediate carry and sum generator.

THE INTERMEDIATE CARRY AND SUM BETWEEN -6 TO +6

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

In step 2, the intermediate carry from the lower significant digit is added to the sum of the current digit to produce the final result. The addition in this step produces no carry because the current digit can always absorb the carry-in from the lower digit. Table-3 shows all possible combinations of the summation between the intermediate carry and the sum.

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:

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

$$= (2\,\bar{2}\,3\,\bar{1})_{QSD}$$

$$(233)_{10} = 3 \times 4^3 + 3 \times 4^2 + \bar{2} \times 4^1 + 1 \times 4^0$$

$$= (3\,3\,\bar{2}\,1)_{QSD}$$

Hence, $(-233)_{10} = (\bar{3}\,\bar{3}\,2\,\bar{1})_{QSD}$

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

A = 107		2	$\bar{2}$	3	$\bar{1}$
B = -233		$\bar{3}$	$\bar{3}$	2	$\bar{1}$
Decimal Sum		-1	-5	5	-2
IC	0	$\bar{1}$	1	0	
IS		$\bar{1}$	$\bar{1}$	1	$\bar{2}$
S		$\bar{2}$	0	1	$\bar{2}$
C_{out}	0				

The sum output is (2 01 2) QSD which 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.

VI. Logic Design And Implementation Using Of Single Digit QSD Adder Unit

There are two steps involved in the carry-free addition. The first step generates an intermediate carry and sum from the addend and augends. The second step combines the intermediate sum of the current digit with the carry of the lower significant digit. To prevent carry from further rippling, two rules are defined. The first rule states that the magnitude of the intermediate sum must be less than or equal to 2. The second rule states that the magnitude of the carry must be less than or equal to 1. Consequently, the magnitude of the second step output cannot be greater than 3 which can be represented by a single-digit QSD number; hence no further carry is required. In step 1, all possible input pairs of the addend and augends are considered.

The range of input numbers can vary from -3 to +3, so the addition result will vary from -6 to +6 which needs two QSD digits. The lower significant digit serves as sum and most significant digit serves as carry. The generation of the carry can be avoided by mapping the two digits into a pair of intermediate sum and intermediate carry such that the nth intermediate sum and the (n-1)th intermediate carry never form any carry generating pair (3,3), (3,2), (3,1), (3 , 3), (3, 2), (3,1). If we restrict the representation 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, addend and augends, and the outputs, the intermediate carry and sum are shown in binary format in Table II.

To remove the further carry propagation the redundancy feature of QSD numbers is used. We restrict the representation such that all the intermediate carries are limited to a maximum of 1, and the intermediate sums are restricted to be less than 3, then the final addition will become carry free. The QSD representations according to these rules are shown in Table 4.3 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 A2, A1, A0 and the augends 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 4.3. 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].

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:

$$IC_2 = a_2 b_2 \left(\overline{a_0 b_0 a_1 b_1}\right) + \left(\overline{a_1 + b_1}\right)\left(a_2 \overline{b_0} + b_2 \overline{a_0}\right)$$

$$IC_1 = a_2 b_2 \left(\overline{a_0 b_0 a_1 b_1}\right) + \left(\overline{a_1 + b_1}\right)\left(a_2 \overline{b_0} + b_2 \overline{a_0}\right)$$

$$IC_0 = IC_2 + \overline{a_2}\,\overline{b_2}(a_1 b_1 + b_1 b_0 + b_0 a_1 + b_1 a_0 + a_1 a_0)$$

Minterms for intermediate sums are:

$$IS_0 = a_0 \overline{b_0} + \overline{a_0} b_0$$

$$IS_1 = \left(a_1 \overline{b_1} + \overline{a_1} b_1\right)\overline{a_0 b_0} + \left(\overline{a_1 \overline{b_1} + \overline{a_1} b_1}\right) a_0 b_0$$

$$IS_2 = IS_0\left(\overline{a_1} b_1 + a_1 \overline{b_1}\right) + b_2 \overline{a_1 b_0} + a_2 \overline{b_1}\,\overline{a_0} + a_0 b_0 \overline{a_1 b_1}(a_2 +$$

The final sum which is carry free is generated from those outputs i.e. Intermediate carry (IC2, IC1, and IC0) and Intermediate sum (IS2, IS1, and IS0). Therefore it has six input and three output bits.

$$S_0 = IC_0 \overline{IS_0} + \overline{IC_0} IS_0$$

$$S_1 = IC_1 \oplus IS_1 \oplus IC_0 IS_0$$

$$S_2 = IC_2 \oplus IS_2 \oplus \left(IC_1 IS_1 + (IC_1 \oplus IS_1) IC_0 IS_0\right)$$

Addition operation for higher order digit does not wait for the completion of addition operation of the immediate lower order digit resulting in a parallel addition of each individual pair of digits.

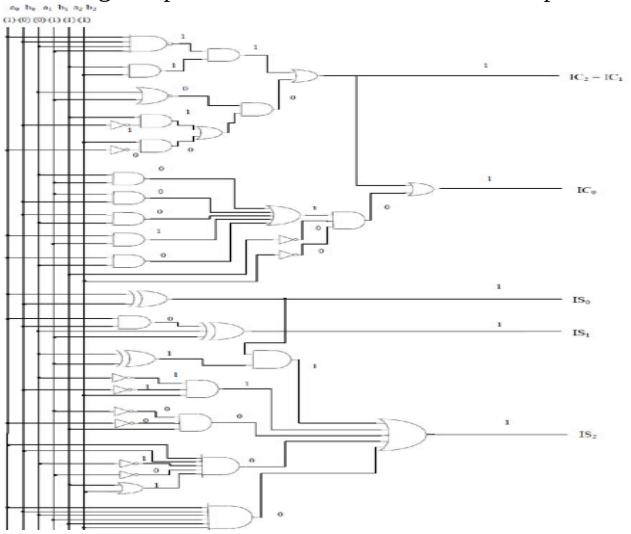


Fig: Data Flow of single digit QSD adder cell.

VII. Simulation Results

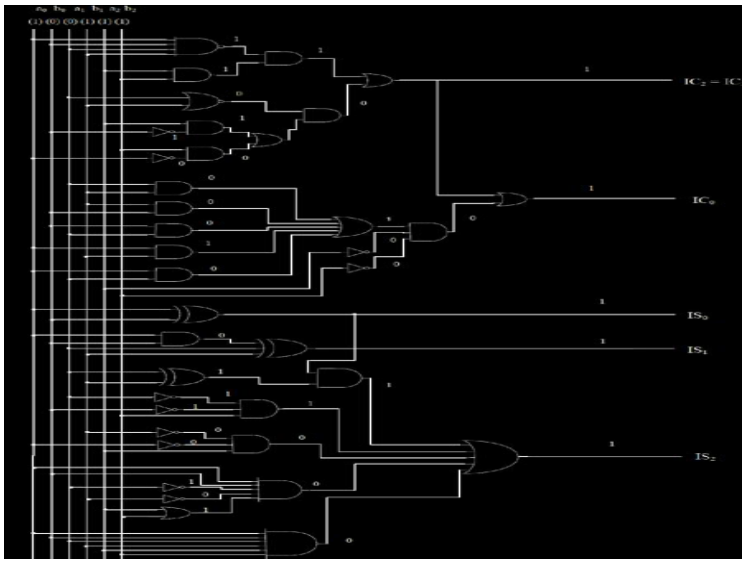


Figure 1: Data Flow of single digit QSD adder cell.

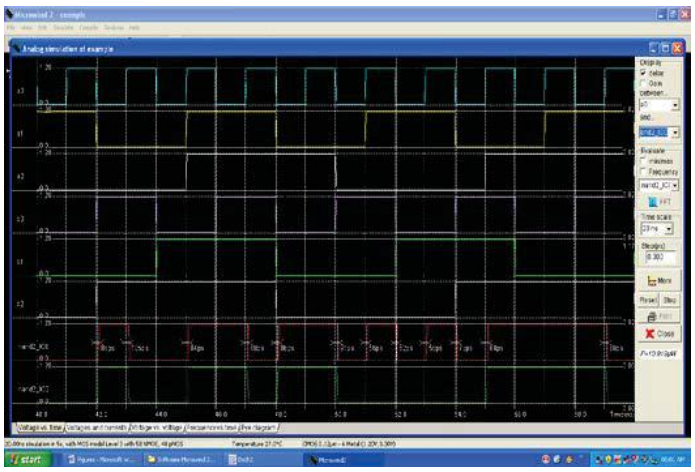


Figure 2: Simulation result of NAND implementation of step1 QSD adder

VIII. Conclusion

In the proposed design of Quaternary Signed Digit adder using NAND-NAND implementation for single digit addition, the dynamic power dissipation is 36.255W at 5GHz frequency. These circuits consume less energy and less energy and power, and shows better performance power dissipation is obtained using Micro wind.

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.