

IMPLEMENTATION OF ARITHMETIC LOGIC UNIT USING QUATERNARY SIGNED DIGIT NUMBER SYSTEM

¹M.Anitha, ²G.Jhansirani, ³D.Raghavakumara, ⁴Dr. P. Anuradha

Abstract: A new number system for Arithmetic Logic Unit. A major problem in arithmetical operation is binary number system to overcome this problem using signed digit operation for carry free arithmetical in this operation using QSD system to perform carry free addition, subtraction and multiplication. The QSD number system requires different modulo based logic elements for arithmetic logic operation. A carry free addition, subtraction and multiplication operations are achieved using a higher radix numbering system. QSD each bit represents by -3 to +3. Quaternary Signed Digit (QSD) has a major contribution in higher radix (=4) carry free arithmetical operation addition, subtraction and multiplication and other carry free operations. More binary numbers like large number of digits such as 64, 128, or more can be implemented with maintain constant delay and less complexity. So we can design an adder without ripple carry. The quaternary numbers are represented using 3-bit 2's complement form of signed digit.

Keywords: quaternary signed digit (QSD), expeditious computation, multiplier, quaternary logic, ALU.

I. Introduction

Arithmetic operations are mostly used and play important roles in various digital systems such as computers and signal processors. The speed of system increases with increasing the speed of addition, subtraction and multiplication of binary number system, here carry may propagate all the way from the least significant digit to the most significant. Thus the addition and other operations time is dependent on the word length. QSD number representation has attracted the interest of many researchers. We introduce, recent advances integrated circuits make large scale arithmetic circuits suitable for VLSI implementation [1][2]. Arithmetic operations limited number of bits propagation time delay, and making circuit complexity still suffer from addition and other problems including. So we propose a high speed carry free addition, subtraction and multiplication operations using QSD arithmetic logic

¹ Sumathi Reddy Institute Of Technology For Women, Warangal, Telangana, India

² Sumathi Reddy Institute Of Technology For Women, Warangal, Telangana, India

³ Sumathi Reddy Institute Of Technology For Women, Warangal, Telangana, India

⁴ Sumathi Reddy Institute Of Technology For Women, Warangal, Telangana, India

unit . For any operand size we use The QSD addition and subtraction operation employs a fixed number of minterms. The multiplier and adders is composed of partial product generators. For testing and verify results, we implement the units using a programmable logic device.

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 RBSD adder. An Algorithm for design of QSD adder is proposed. This algorithm is used to write the verilog code for QSD adders. verilog codes for QSD adder is simulated and synthesized and the timing report is generated. The timing report gives the delay time produced by the adder structure. Binary signed-digit numbers are known to allow limited carry propagation with a somewhat more complex addition process requiring very large circuit for implementation. 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.

II. Basic Concept

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

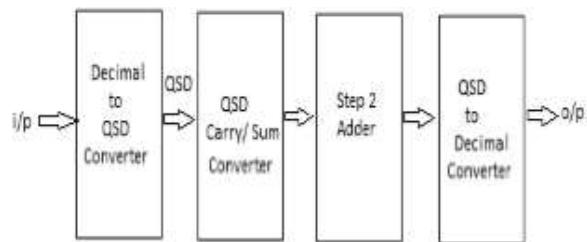


Figure1 Basic conversion

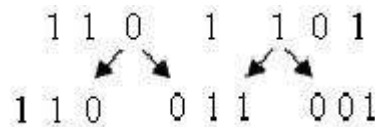
III. Binary Number To QSD Number conversion Technique

QSD 1-digit can be represented by one 3-bit binary equipollent as follows:

$$\begin{aligned} \bar{3} &= 101 \\ \bar{2} &= 110 \\ \bar{1} &= 111 \\ 0 &= 000 \\ 1 &= 001 \\ 2 &= 010 \\ 3 &= 011 \end{aligned}$$

We have to convert this n-bit binary data into 3q-bit binary data .So to convert n-bit binary data to q-digit QSD data, we have to split the 3rd, 5th, 7th bit..... the initial bit represent MSB and end bit represent LSB into two partions.MSB bit is cannot split.

If the bit is 1 split into 1 & 0 and if it is 0 then, it is split into 0 & 0. For example I makes one binary number and using splitting technique of a binary number (1101101)2is shown below:

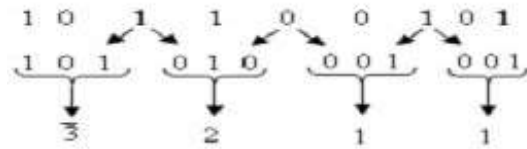


So the binary data (1) we have to split q- times for example, the 2-bit converting, quaternary digit number is splitting 1 time; for 3-digit converting quaternary digit number the split is 2-times and so on... In each any binary bit splitting one extra bit is generated. So, the binary bits for conversion to its QSD equipollent (n) = (Total numbers of bits engendered after divisions) – (one extra bit engendered due to splitting).shown in below formula.

$$\begin{aligned} n &= 3q - \{1 \times (q - 1)\} \\ &= (2q + 1) \end{aligned} \tag{3}$$

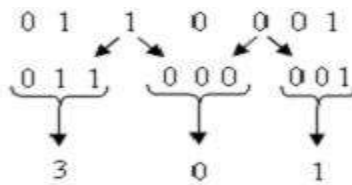
So, the number of bits positions in the binary number should be 3,5,7,9(odd positions) etc for converting into QSD number. In binary numbers for any bit splitting into every 3-bit can be converted into QSD according to the equation (2). The following two examples as given below will avail to make the things clear.

1) Let $(-155)_{10} = (101100101)_2$. i.e decimal into binary have be converted into QSD number system.the given data is 9 bit binary data i.e is (101100101) here 3rd bit is 1,5th bit is 0 and 7th bit is 1. According to equation (3) we can follow the above steps i.e splitting technique expressed verbally above the binary data can be expressed as follows and write in QSD formation



So the QSD equivalent of $(101100101)_2$ is $(\bar{3}211)_4$.

2) Let $(49)_{10} = (0110001)_2$ is in the binary form so convert binary into QSD form. The given binary data is 7 bit binary number“ $(0110001)_2$ ”. According to QSD steps to convert binary to QSD number. the conversion is as follows



So the QSD equivalent of $(110001)_2$ is $(301)_4$.

IV. Design Algorithm of Adder/Subtractor

The most consequential Integration arithmetic operation in the digital computation. A highly desirable carry-free integration is as the number of digits becomes sizably voluminous. We can achieve carry-free addition or subtraction by exploiting the redundancy of QSD numbers.

The carry-free integration involved two steps. The first step is intermediate carry and sum from the addend and augend from given any bit. The second step is the intermediate sum of the current digit with the carry of the lower paramount digit. Carry from further rippling, is defining two rules. The first rule states that the magnitude of the intermediate sum must be less than or equipollent to 2. The second rule states that the magnitude of the carry must be less than or equipollent to 1. Hence no further carry is required because the magnitude of the second step output cannot be more than 3 which can be represented by a single-digit QSD number;. In 1st step all possible input pairs of the addend and augend are considered similar to subtraction also. The range of output 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).

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

The range of output from - 6 to 6 which can be represented as intermediate carry and sum in QSD format as show in Table 2. Some multiple numbers representations, opted for defined rules. In these process generate intermediate carry and sum are listed in the below Table 2.

Table 2. 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, $\bar{1}\bar{3}$	01
2	02, $\bar{1}\bar{2}$	02
3	03, $\bar{1}\bar{1}$	$\bar{1}\bar{1}$
4	10	10
5	11, $\bar{2}\bar{3}$	11
6	12, $\bar{2}\bar{2}$	12

The Input and output can be represent in the formation of 3 bit, 2's complement binary number. the addition or mapping between the inputs to generate added and augend bits and the outputs generate the intermediate carry and sum are shown in below Table 3. Since the addition of two bits generate intermediate carry that carry is always in between -1 to 1, requires only a 2 bit binary representation. We express finally five 6-variable Boolean function can be expressed.

In 2nd step, the intermediate carry from the lower digit is integrated to the sum of the current digit to generate the final result. The integration in this process no carry because the current digit can always absorb the carry-in from the lower digit. Table 4 shows all possible combinations of the summation between the intermediate carry and the sum shown in below.

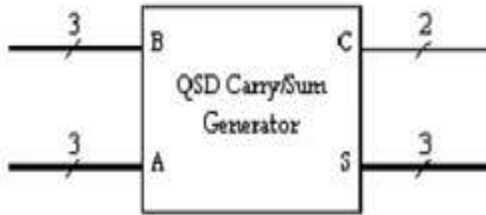


Figure 2 Intermediate carry and sum generator

Table 3. The mapping between the inputs and outputs of the intermediate carry and sum

INPUT				OUTPUT				
QSD		Binary		Decimal	QSD		Binary	
A _i	B _i	A _i	B _i	Sum	C _i	S _i	C _i	S _i
3	3	011	011	6	1	2	01	010
3	2	011	010	5	1	1	01	001
2	3	010	011	5	1	1	01	001
3	1	011	001	4	1	0	01	000
1	3	001	011	4	1	0	01	000
2	2	010	010	4	1	0	01	000
1	2	001	010	3	1	-1	01	111
2	1	010	001	3	1	-1	01	111
3	0	011	000	3	1	-1	01	111
0	3	000	011	3	1	-1	01	111
1	1	001	001	2	0	2	00	010
0	2	000	010	2	0	2	00	010
2	0	010	000	2	0	2	00	010
3	-1	011	111	2	0	2	00	010
-1	3	111	011	2	0	2	00	010
0	1	000	001	1	0	1	00	001
1	0	001	000	1	0	1	00	001
2	-1	010	111	1	0	1	00	001
-1	2	111	010	1	0	1	00	001
3	-2	011	110	1	0	1	00	001
-2	3	110	011	1	0	1	00	001
0	0	000	000	0	0	0	00	000
1	-1	001	111	0	0	0	00	000
-1	1	111	001	0	0	0	00	000
2	-2	010	110	0	0	0	00	000
-2	2	110	010	0	0	0	00	000
-3	3	101	011	0	0	0	00	000
3	-3	011	101	0	0	0	00	000
0	-1	000	111	-1	0	-1	00	111
-1	0	111	000	-1	0	-1	00	111
-2	1	110	001	-1	0	-1	00	111
1	-2	001	110	-1	0	-1	00	111
-3	2	101	010	-1	0	-1	00	111
2	-3	010	101	-1	0	-1	00	111
-1	-1	111	111	-2	0	-2	00	110
0	-2	000	110	-2	0	-2	00	110
-2	0	110	000	-2	0	-2	00	110
-3	1	101	001	-2	0	-2	00	110
1	-3	001	101	-2	0	-2	00	110
-1	-2	111	110	-3	-1	1	11	001
-2	-1	110	111	-3	-1	1	11	001
-3	0	101	000	-3	-1	1	11	001
0	-3	000	101	-3	-1	1	11	001
-3	-1	101	111	-4	-1	0	11	000
-1	-3	111	101	-4	-1	0	11	000
-2	-2	110	110	-4	-1	0	11	000
-3	-2	101	110	-5	-1	-1	11	111
-2	-3	110	101	-5	-1	-1	11	111
-3	-3	101	101	-6	-1	-2	11	110

Table 4. All possible combinations outputs of intermediate carry (A) and sum (B).

	B	-2	-1	0	1	2
A	-1	-3	-2	-1	0	1
	0	-2	-1	0	1	2
	1	-1	0	1	2	3

If Three 5-variable Boolean expressions can be extracted from Table 5. The second step is implementation of an n bit-digit QSD adder requires n QSD carry and sum engenderers and n-1 adders as shown in Figure 2. The result turns out to be an n+1-digit number.

Table 5. The second step is mapping between inputs and outputs of QSD adder.

INPUT				OUTPUT		
QSD		Binary		Decimal	QSD	Binary
A _i	B _i	A _i	B _i	Sum	S _i	S _i
1	2	01	010	3	3	111
1	1	01	001	2	2	010
0	2	00	010	2	2	010
0	1	00	001	1	1	001
1	0	01	000	1	1	001
-1	2	11	010	1	1	001
0	0	00	000	0	0	000
1	-1	01	111	0	0	000
-1	1	11	001	0	0	000
0	-1	00	111	-1	-1	111
-1	0	11	000	-1	-1	111
1	-2	01	110	-1	-1	111
-1	-1	11	111	-2	-2	110
0	-2	00	110	-2	-2	110
-1	-2	11	110	-3	-3	001

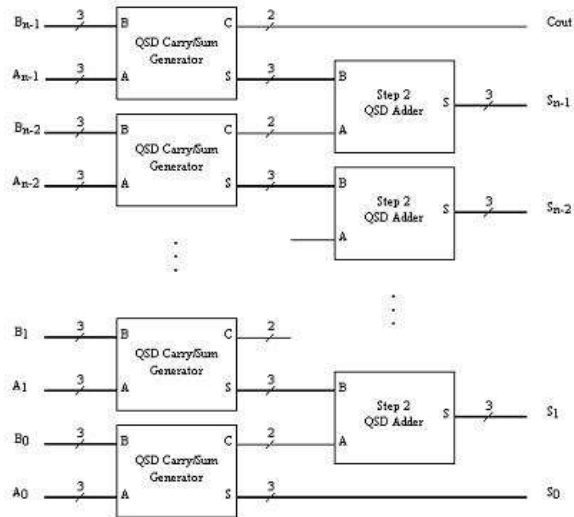


Figure 3. n-digit QSD adder

V. Multiplier Design

In multiplication operation There are generally two methods for iterative and parallel. QSD multiplication can be implemented in both ways, requiring a QSD adder as rudimental components and QSD partial product engenderer. A partial product, M_i , is a result of multiplication between an n-digit input, $A_{n-1}A_0$, with a single digit input, B_i , where $i = 0..n-1$. The first bit output directly generate and remain is partial products. After that remaining partial products are added based on given QSD steps. Functionality can be expressed as shown in below Table 6.

Table 6. All possible combinations outputs of multiplicand (A) and multiplier (B).

B \ A	-3	-2	-1	0	1	2	3
-3	9	6	3	0	-3	-6	-9
-2	6	4	2	0	-2	-4	-6
-1	3	2	1	0	-1	-2	-3
0	0	0	0	0	0	0	0
1	-3	-2	-1	0	1	2	3
2	-6	-4	-2	0	2	4	6
3	-9	-6	-3	0	3	6	9

The multiplication of single-digit produces results M and carry C to be combined with M of the next digit. The range between -2 and 2 of both outputs, M and C . Creating of Table 3 and 5 is following table 8 procedure. the mapping between the A and B is 6-bit input to the 6-bit output, that generates result M and carry C , finally get results in six 6-variable Boolean expressions which represent a single-digit multiplication operation. The below figure 3 shows a single-digit QSD multiplier .

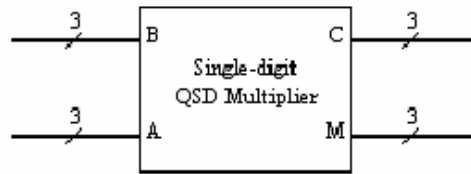


Figure 4. A single-digit QSD multiplier

The design of an n -digit partial product generator uses n -units of the single-digit QSD multiplier. Gathering all the outputs to produce a partial product a small change in results. The below table 7 is QSD representation of a single digit multiplication output, contains a carry-out of magnitude 2 when the output is either -9 or 9 . The use of the second step QSD adder alone as a gatherer is prohibitive. In fact, the implementation of QSD adder is using previous section as the gatherer. Furthermore the input of magnitude 3 contains the intermediate carry and sum circuit can be obtained. The below figure 4 is QSD partial product generator implementation.

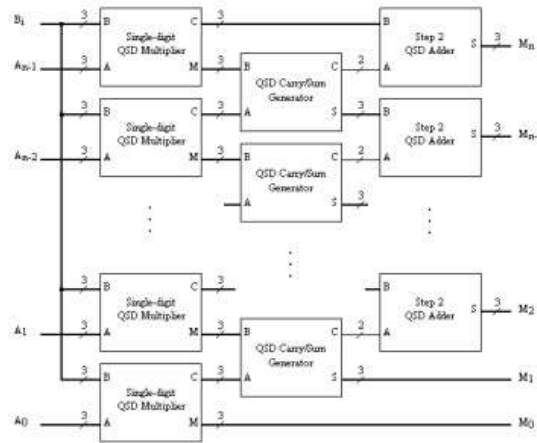


Figure 5. The n -digit QSD partial product generator.

Table 7. The QSD representation of a single-digit multiplication output.

Mult	QSD represented Number	QSD coding Number
-9	$\bar{2}\bar{1},\bar{3}\bar{3}$	$\bar{2}\bar{1}$
-6	$\bar{2}\bar{2},\bar{1}\bar{2}$	$\bar{1}\bar{2}$
-4	$\bar{1}\bar{0}$	$\bar{1}\bar{0}$
-3	$\bar{1}\bar{1},\bar{0}\bar{3}$	$\bar{1}\bar{1}$
-2	$\bar{1}\bar{2},\bar{0}\bar{2}$	$\bar{0}\bar{2}$
-1	$\bar{1}\bar{3},\bar{0}\bar{1}$	$\bar{0}\bar{1}$
0	00	00
1	01, $\bar{1}\bar{3}$	01
2	02, $\bar{1}\bar{2}$	02
3	03, $\bar{1}\bar{1}$	$\bar{1}\bar{1}$
4	10	10
6	12, $\bar{2}\bar{2}$	12
9	21, $\bar{3}\bar{3}$	21

If multiplications more number of bits like An nxn-digit QSD multiplication requires n-partial product terms. The design of a 2ndigit QSD adder is used to add-shift operations to perform between the partial product generator and the accumulator. After n-iterations, the multiplication process is complete.

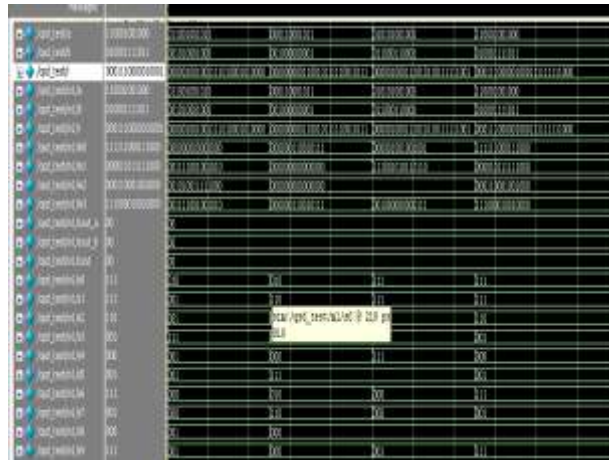
Implementation of parallel multiplication requires n-partial product circuits and n-1 QSD adder units. A reduction of binary sum is applied so automatically reduce the propagation delay to $O(\log n)$.

VI. Simulation Results

The QSD adder/subtractor input bits written in VHDL, compiled and simulation using modelsim. The QSD adder and multiplier circuit simulated and synthesized on SPARTAN3E FPGA using XilinxISE. The QSD adder and multiplier circuit simulated and synthesized. The simulated result for 4-bit QSD adders and multiplier as shown in below.



Figure 6: Simulated result QSD adder



Component	Value	Value	Value	Value
QSD_Multiplier	00000000	00000000	00000000	00000000
QSD_Multiplier_0	00000000	00000000	00000000	00000000
QSD_Multiplier_1	00000000	00000000	00000000	00000000
QSD_Multiplier_2	00000000	00000000	00000000	00000000
QSD_Multiplier_3	00000000	00000000	00000000	00000000
QSD_Multiplier_4	00000000	00000000	00000000	00000000
QSD_Multiplier_5	00000000	00000000	00000000	00000000
QSD_Multiplier_6	00000000	00000000	00000000	00000000
QSD_Multiplier_7	00000000	00000000	00000000	00000000
QSD_Multiplier_8	00000000	00000000	00000000	00000000
QSD_Multiplier_9	00000000	00000000	00000000	00000000
QSD_Multiplier_10	00000000	00000000	00000000	00000000
QSD_Multiplier_11	00000000	00000000	00000000	00000000
QSD_Multiplier_12	00000000	00000000	00000000	00000000
QSD_Multiplier_13	00000000	00000000	00000000	00000000
QSD_Multiplier_14	00000000	00000000	00000000	00000000
QSD_Multiplier_15	00000000	00000000	00000000	00000000
QSD_Multiplier_16	00000000	00000000	00000000	00000000
QSD_Multiplier_17	00000000	00000000	00000000	00000000
QSD_Multiplier_18	00000000	00000000	00000000	00000000
QSD_Multiplier_19	00000000	00000000	00000000	00000000
QSD_Multiplier_20	00000000	00000000	00000000	00000000
QSD_Multiplier_21	00000000	00000000	00000000	00000000
QSD_Multiplier_22	00000000	00000000	00000000	00000000
QSD_Multiplier_23	00000000	00000000	00000000	00000000
QSD_Multiplier_24	00000000	00000000	00000000	00000000
QSD_Multiplier_25	00000000	00000000	00000000	00000000
QSD_Multiplier_26	00000000	00000000	00000000	00000000
QSD_Multiplier_27	00000000	00000000	00000000	00000000
QSD_Multiplier_28	00000000	00000000	00000000	00000000
QSD_Multiplier_29	00000000	00000000	00000000	00000000
QSD_Multiplier_30	00000000	00000000	00000000	00000000
QSD_Multiplier_31	00000000	00000000	00000000	00000000
QSD_Multiplier_32	00000000	00000000	00000000	00000000
QSD_Multiplier_33	00000000	00000000	00000000	00000000
QSD_Multiplier_34	00000000	00000000	00000000	00000000
QSD_Multiplier_35	00000000	00000000	00000000	00000000
QSD_Multiplier_36	00000000	00000000	00000000	00000000
QSD_Multiplier_37	00000000	00000000	00000000	00000000
QSD_Multiplier_38	00000000	00000000	00000000	00000000
QSD_Multiplier_39	00000000	00000000	00000000	00000000
QSD_Multiplier_40	00000000	00000000	00000000	00000000
QSD_Multiplier_41	00000000	00000000	00000000	00000000
QSD_Multiplier_42	00000000	00000000	00000000	00000000
QSD_Multiplier_43	00000000	00000000	00000000	00000000
QSD_Multiplier_44	00000000	00000000	00000000	00000000
QSD_Multiplier_45	00000000	00000000	00000000	00000000
QSD_Multiplier_46	00000000	00000000	00000000	00000000
QSD_Multiplier_47	00000000	00000000	00000000	00000000
QSD_Multiplier_48	00000000	00000000	00000000	00000000
QSD_Multiplier_49	00000000	00000000	00000000	00000000
QSD_Multiplier_50	00000000	00000000	00000000	00000000
QSD_Multiplier_51	00000000	00000000	00000000	00000000
QSD_Multiplier_52	00000000	00000000	00000000	00000000
QSD_Multiplier_53	00000000	00000000	00000000	00000000
QSD_Multiplier_54	00000000	00000000	00000000	00000000
QSD_Multiplier_55	00000000	00000000	00000000	00000000
QSD_Multiplier_56	00000000	00000000	00000000	00000000
QSD_Multiplier_57	00000000	00000000	00000000	00000000
QSD_Multiplier_58	00000000	00000000	00000000	00000000
QSD_Multiplier_59	00000000	00000000	00000000	00000000
QSD_Multiplier_60	00000000	00000000	00000000	00000000
QSD_Multiplier_61	00000000	00000000	00000000	00000000
QSD_Multiplier_62	00000000	00000000	00000000	00000000
QSD_Multiplier_63	00000000	00000000	00000000	00000000
QSD_Multiplier_64	00000000	00000000	00000000	00000000
QSD_Multiplier_65	00000000	00000000	00000000	00000000
QSD_Multiplier_66	00000000	00000000	00000000	00000000
QSD_Multiplier_67	00000000	00000000	00000000	00000000
QSD_Multiplier_68	00000000	00000000	00000000	00000000
QSD_Multiplier_69	00000000	00000000	00000000	00000000
QSD_Multiplier_70	00000000	00000000	00000000	00000000
QSD_Multiplier_71	00000000	00000000	00000000	00000000
QSD_Multiplier_72	00000000	00000000	00000000	00000000
QSD_Multiplier_73	00000000	00000000	00000000	00000000
QSD_Multiplier_74	00000000	00000000	00000000	00000000
QSD_Multiplier_75	00000000	00000000	00000000	00000000
QSD_Multiplier_76	00000000	00000000	00000000	00000000
QSD_Multiplier_77	00000000	00000000	00000000	00000000
QSD_Multiplier_78	00000000	00000000	00000000	00000000
QSD_Multiplier_79	00000000	00000000	00000000	00000000
QSD_Multiplier_80	00000000	00000000	00000000	00000000
QSD_Multiplier_81	00000000	00000000	00000000	00000000
QSD_Multiplier_82	00000000	00000000	00000000	00000000
QSD_Multiplier_83	00000000	00000000	00000000	00000000
QSD_Multiplier_84	00000000	00000000	00000000	00000000
QSD_Multiplier_85	00000000	00000000	00000000	00000000
QSD_Multiplier_86	00000000	00000000	00000000	00000000
QSD_Multiplier_87	00000000	00000000	00000000	00000000
QSD_Multiplier_88	00000000	00000000	00000000	00000000
QSD_Multiplier_89	00000000	00000000	00000000	00000000
QSD_Multiplier_90	00000000	00000000	00000000	00000000
QSD_Multiplier_91	00000000	00000000	00000000	00000000
QSD_Multiplier_92	00000000	00000000	00000000	00000000
QSD_Multiplier_93	00000000	00000000	00000000	00000000
QSD_Multiplier_94	00000000	00000000	00000000	00000000
QSD_Multiplier_95	00000000	00000000	00000000	00000000
QSD_Multiplier_96	00000000	00000000	00000000	00000000
QSD_Multiplier_97	00000000	00000000	00000000	00000000
QSD_Multiplier_98	00000000	00000000	00000000	00000000
QSD_Multiplier_99	00000000	00000000	00000000	00000000

Figure 7: Simulated result QSD multiplier



Figure 8 :RTL Schematic of QSD multiplier

VII. Conclusion

In this technology is the implementation of QSD addition/subtraction and multiplication are presented. The latest technology QSD Arithmetic Logic Unit design is better comparing to other designs. 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. The basic building block for other arithmetic operations such as addition/subtraction, multiplication, division, square root, etc operations using QSD. Some well-known arithmetic algorithms can be directly implemented using QSD technology.

References

- [1] M. Thoidis, D. Soudris, J. M. Fernandez, A. Thanailakis, "The circuit design of multiple-valued logic voltage-mode adders," 2001 IEEE
- [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 representation ".IEEE transactions on computers, vol 39, no.1, pp.89-98, January 1990.
- [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] Nagamani A. N, Nishchai S, "Quaternary High Performance Arithmetic Logic Unit Design", 14th Euromicro Conference on Digital System Design 2011 IEEE. [2]
- [10] Reena Rani, Laxmikant Singh, Neelam Sharma, "FPGA Implementation of fast Adder using Quaternary Sign Digit Number System", 2009 International comference on Trend in Eletronic and Photonic Deice & Systems (ELECTRO-2009). [3]
- [11] Pranali S.Kamble & S.M.Choudhary, "Review of VHDL Implementation of Quaternary Signed Adder System", International Journal on Advanced Electrical and Electronics Engineering, (IJAEED), ISSN (Print): 2278-8948, Volume-1, Issue-1, 2012 [4]
- [12] Songpol Ongwattanakul Phaisit Chewputtanagul, David J. Jackson, Kenneth G. Ricks, "Quaternary Arithmetic Logic Unit on a Programmable Logic Device", proc. IEEE conference, 2001.

- [13] Jyoti R. Hallikhed, Mahesh R.K., "VLSI Implementation of Fast Addition using Quaternary Signed Digit Number System", International Journal of Ethics in Engineering & Management Education, Vol.2, Issue 5, May 2015.
- [14] S. Jakeer Hussain, K. Sreenivasa Rao, "Design and implementation of Fast Addition Using QSD for Signed and Unsigned Numbers", International Journal of Engineering Research, Volume No. 3, Issue No: Special2, pp: 52-54, March 2014.
- [15] C. V. Sathish Kumar, P. Jaya Rami Reddy, "Implementation of Fast Adder Using QSD for Signed and Unsigned Numbers", International Journal of Science, Engineering and Technology Research (IJSETR), Volume 3, Issue 11, November 2014.
- [16] Kothuru. Ram Kumar, B. Praveen Kumar, "Fast Addition Using QSD VLSI Adder for Better Performance", International Journal of Trend in Research and Development, Volume 2(6), Nov-Dec 2015.
- [17] G. Manasa, M. Damodhar Rao, K. Miranji, "Design and Analysis of Fast Addition Mechanism for Integers using Quaternary Signed Digit Number System", International Journal of VLSI and Embedded Systems- IJVES, Vol 05, Article 09455, October 2014.