Hardware Implementation for High-Speed Parallel Adder for QSD 2D Data Arrays

Mohammed A. Al-Ibadi
Computer Engineering Department
University of Basrah
Basrah, Iraq
m.a.al-ebadi@ieee.org

Abstract— **In this paper, a fast parallel 2D-array computation has been implemented based on QSD number system and FPGA hardware. The QSD numbers are coded in binary bit streams in order to be processed inside the digital hardware. The same parallel addition algorithm for adding two QSD numbers is used for adding the binary codded QSD numbers and its two steps has been implemented successfully in FPGA chip. The syntheses and implementation of the hardware parallel adder shows that the proposed parallel adder has a high performance than the software version, and the size of data arrays depends on the number of logic elements of the FPGA chip.**

Keywords—Parallel computing, QSD numbers, Addition algorithm, FPGA.

I. Introduction

Current technologies are going toward developing high speed processing units to fulfill the requirement of today demands for ultra-high speed devices [1]. With respect to digital system units, the arithmetic unit takes long processing time when dealing with long numbers as well the large array of numbers of form conventional binary number system. This occurs because of the carry/barrow propagation delay which needs sequential operation of the intermediate results [2]. It is clear that the computations will take long time when the numbers under processing has long bit length. In order to overcome this bottleneck, Signed-Digit (SD) number system had been proposed and used in various digital systems with different implementations, such as digital and optical computing system [3-4].

Signed-Number (SD) system has the general form [5]:

$$d = \sum_{i=0}^{n-1} x_i r^i \qquad (1)$$

where,
d= the SD number,
n= number of SD digit of the SD number
r= the radix of the SD number system,
x_i= the ith digit of the SD number.

The radix *r* can take any value such as 2 (Modified Signed-Digit (MSD)), 3 (Trinary Signed-Digit (TSD)), and 4 (Quaternary Signed-Digit (QSD)) number systems.

An arithmetic unit will be more powerful if it can process higher radix SD number system. For QSD number system, the SD number will be formed from seven digits ($\bar{3}$, $\bar{2}$, $\bar{1}$, 0, 1, 2, 3), such as $(1\bar{2}\bar{3}\bar{1})_{QSD}$ which equals to $(19)_{10}$.

Singed-Number system (SD) has its special features [6]:

1- The carry propagation of the arithmetic operations can be limited with few steps independently to the length of the numbers, such as three, two, or one step.

2- Because of the limitation in carry propagation, parallel arithmetic operations can be done for all digits of the two numbers.

3- The number has a redundant equal values such as $(19)_{10}$=$(1\bar{2}\bar{3}\bar{1})_{QSD}$, $(011\bar{1})_{QSD}$, $(0103)_{QSD}$, and $(02\bar{3}\bar{1})_{QSD}$, and this feature led to construct special intermediate results for the arithmetic operation to be done in limit steps and in parallel manner as explain in the next section.

II. Parallel Two-Step Addition for QSD Numbers

Parallel two-step addition operation of QSD number consists of two steps to produce the addition result [7]. During the first step, the addend and augend corresponding digits are added and produced intermediate sums and carries. In the second step, the intermediate sums and curries will be added and find the final result. Equation (2) explain the two steps of addition [8].

$$\begin{aligned} &\text{step1:} \quad x_i + y_i = c_i + s_i \\ &\text{step2:} \quad s_i + c_{i-1} = z_i \end{aligned} \qquad (2)$$

where x_i , y_i, c_i, s_i, and z_i are the i^{th} SD digit of addend, augend, intermediate carry, intermediate sum, final sum respectively, and c_{i-1} is the carry of the previous i^{th-1} digit.

Fig. 1 shows the block diagram of the two-step QSD adder for two *n*-digit numbers. Note that the resulted QSD number will be (*n+1*)-digit as maximum length [9]. Tables 1 and 2 show the addition results for the first and second steps, respectively.

978-1-7281-7116-6/20/$31.00 ©2020 IEEE

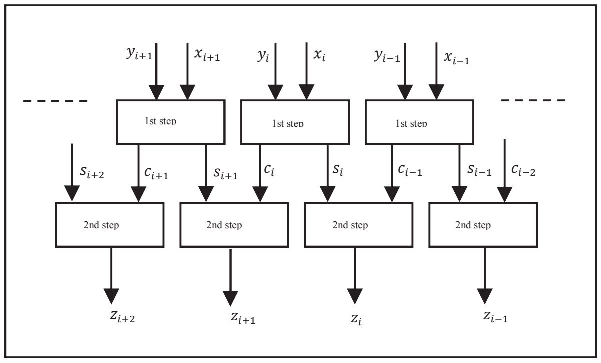

Fig. 1. Block diagram of the two-step QSD adder

TABLE I. FIRST STEP RESULTS OF QSD ADDITION

SD digits to be added	**Intermediate results**	
(x_i , y_i)	**s_i**	**c_i**
$(3,3)$	2	1
$(3,2),(2,3)$	1	1
$(3,1),(1,3),(2,2)$	0	1
$(3,0),(0,3),(2,1),(1,2)$	$\bar{1}$	1
$(3, \bar{1}), (\bar{1},3),(2,0),(0,2),(1,1)$	2	0
$(3, \bar{2}),(\bar{2},3),(2, \bar{1}),(\bar{1},2),(1,0),(0,1)$	1	0
$(3, \bar{3}),(\bar{3},3),(2, \bar{2}),(\bar{2},2),(1, \bar{1}),(\bar{1},1),(0,0)$	0	0
$(\bar{3},2),(2, \bar{3}),(\bar{2},1),(1, \bar{2}),(\bar{1},0),(0, \bar{1})$	$\bar{1}$	0
$(\bar{3},1),(1, \bar{3}),(\bar{2},0),(0, \bar{2}),(\bar{1}, \bar{1})$	$\bar{2}$	0
$(\bar{3},0),(0, \bar{3}),(\bar{2}, \bar{1}),(\bar{1}, \bar{2})$	1	$\bar{1}$
$(\bar{3}, \bar{1}),(\bar{1}, \bar{3}),(\bar{2}, \bar{2})$	0	$\bar{1}$
$(\bar{3}, \bar{2}),(\bar{2}, \bar{3})$	$\bar{1}$	$\bar{1}$
$(\bar{3}, \bar{3})$	$\bar{2}$	$\bar{1}$

III. BINARY CODED QSD DIGITS

As explained in the previous section, the QSD numbers may have signed (negative and positive) digits in its contents. The sign of the digits should be represented as a digital data in computing systems that process such number type [10]. The key of the design is the converting QSD digits into its equivalent binary bits to be capable to process as a digital data. We suggest that, each QSD digit can be represented by 3-bit binary coded QSD digit, two bits for the value and the last bit for the sign. Table 3 Explain the proposed binary coded QSD digits.

TABLE II. SECOND STEP RESULTS OF QSD ADDITION

Intermediate results	**Final result**
(s_i, c_{i-1})	**z_i**
$(2,1)$	3
$(2,0),(1,1)$	2
$(2, \bar{1}),(1,0),(0,1)$	1
$(1, \bar{1}),(\bar{1},1),(0,0)$	0
$(\bar{2},1),(\bar{1},0),(0, \bar{1})$	$\bar{1}$
$(\bar{2},0),(\bar{1}, \bar{1})$	$\bar{2}$
$(\bar{2}, \bar{1})$	$\bar{3}$

By this way, we will consider the signed QSD digits presented in Tables 1 and 2 as their equivalent binary coded QSD digits listed in Table 3.

IV. DESIGN AND IMPLEMENTATION A HARDWARE ARCHITECTURE FOR QSD ARRAY ADDER

A. Design overview

Fig. 2 explains the proposed hardware design for QSD array adder based on an FPGA chip. The hardware design is consist mainly from three parts as describes below:

1- The memory locations of the two input QSD array:

The two arrays to be added enter the chip from external interface as a binary coded form as explained in section (III). The two arrays will be stored in distributed memory locations using the dedicated memory blocks of the FPGA chip. According to data bus width, the QSD array sizes, and the clock time period, the total time of loading the two arrays can be calculated. The distributed memory locations is done by using only the first memory location of each memory block of the FPGA chip for storing a part of the QSD array. These memory locations can be configured with a suitable data width as the design requirements. The total number of the memory blocks is depending on the targeted FPGA chip. The write and other control signals of the memory blocks are supplied and synchronized from outside the FPGA chip. By this way, the two QSD array will be stored completely in this part of the proposed QSD array adder.

TABLE III. THE PROPOSED CONVERSION OF THE QSD DIGITS INTO BINARY BITS.

QSD Digit	**Sing bit**	**Digit bits**	**Binary coded QSD digit**
3	0	11	011
2	0	10	010
1	0	01	001
0	0	00	000
$\bar{1}$	1	01	101
$\bar{2}$	1	10	110
$\bar{3}$	1	11	111

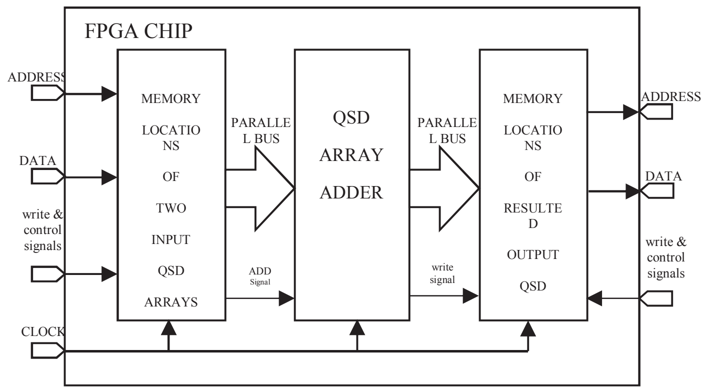

Fig. 2. Block diagram of the proposed hardware architecture for QSD array adder

2- Parallel QSD array adder:

This part contains the parallel adder of the two QSD arrays. Its two inputs are the two QSD arrays to be added and stored in memory parts. The two arrays will enter completely at the same time to the first step of the QSD adder. Each two corresponding binary coded QSD digits will be added simultaneously. As a result, the intermediate sums and carries of all added numbers, whole arrays, will be produced at the same time, i.e. two new arrays will be generated, one for intermediate sums and the other for intermediate carries. The time from entering the two arrays to generation the arrays for intermediate sums and carries is one clock period. This is because all digits of all numbers of the two arrays will be added in parallel manner during just one clock period.

During the second period, the second step operation of the QSD adder will be done over the two intermediate arrays of sums and carries which generated from the first step, and the produced array will be the binary coded QSD array of the addition results.

It should be noticed that, if the array size of the two inputs are $(m \times k \times n)$, the final result array will be $(m \times k \times (n + 1))$, where m and k are the number of rows and columns of the input QSD arrays and n is the number of the QSD digits per a number. In binary coded QSD point of view, each number of the resulted array will be increased by 3 bits which represent the additional QSD digit of the resulted numbers. An example of two 16-digit QSD numbers addition is explained below:

$$\begin{array}{rl} 1320\bar{2}01201\bar{1}12\bar{3}\bar{1}31 & \\ 03123\bar{3}1100\ 1\bar{1}20111 & + \\ \hline 023\bar{1}1\bar{1}122001\bar{1}1102 & \end{array}$$

that equivalent in decimal number system to:

$$\begin{array}{rl} 2005267773 & \\ 0894701077 & + \\ \hline 2899968850 & \end{array}$$

3- The memory locations of the resulted output QSD array:

This part of the QSD adder is for storing the final result binary coded QSD array. It has the same design principles of the first part of the QSD array adder which consists of distributed memory locations for storing the binary coded QSD array. The final results array will be stored completely during one clock period to the memory locations of memory block of the FPGA chip. Then, result array will be read by the external hardware as a binary coded QSD array of the resulted addition operation. Also, depending on data width selected, the time of completely extracting the resulted array can be calculated accurately by calculated the number of clocks periods required for finishing this operation.

B. FPGA-Based Implementation

In order to implement the proposed hardware architecture of the QSD array adder, an FPGA chip is used as a targeted chip for implementation. Altera Cyclone III (EP3C16F484C6) (DE0 Board) is used and a complete hardware design is described using VHDL language and the implementation is done by Quartus II design tool.

Fig. 3 show the FPGA-based implementation of the two steps of the QSD adder for adding two numbers each is 16 QSD digits (binary recoded) and generating the sum and carry and the final result.

After synthesis, mapping, placing, and routing stages of FPGA design implementation, the logic elements, storage bits, and I/O pins of the FPGA chip that used by implementing a QSD array adder for array size $(10 \times 2 \times 16)$ are as explained in Table 4.

TABLE IV RESOURCE OCCUPATION OF THE FPGA CHIP (EP3C16F484C6) FOR QSD ARRAY ADDER IMPLEMENTATION

FPGA Resources Used	Usage Ratio
LEs usage ratio	89.3%
Storage bits usage ratio	0.57%
I/O pins usage ratio	22.5%

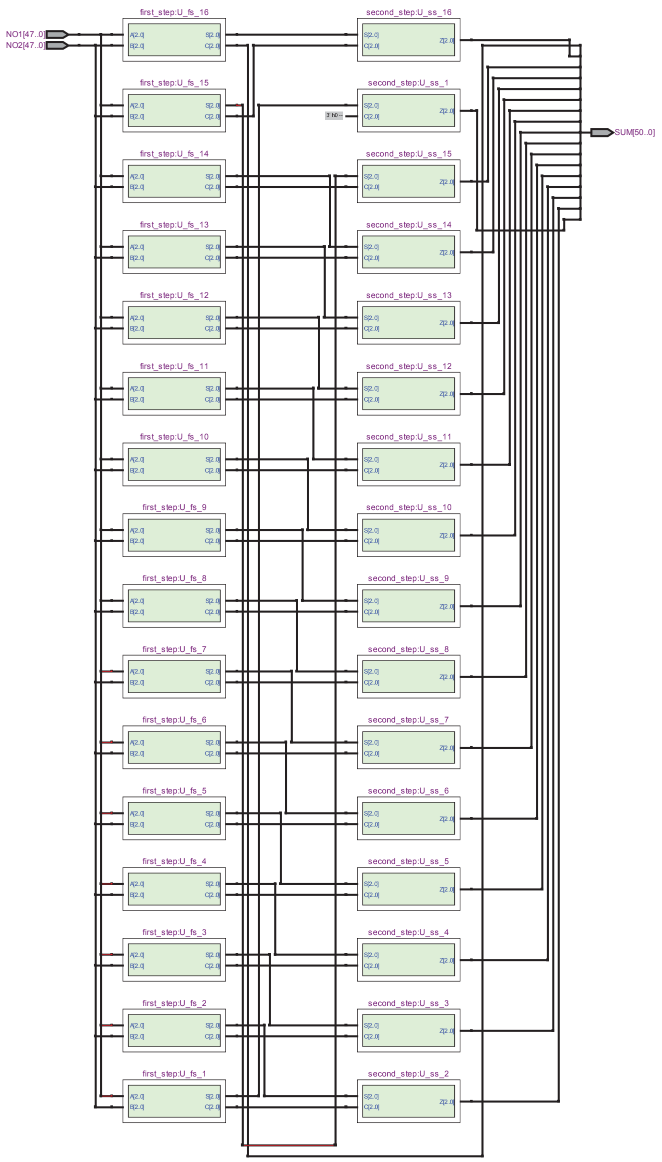

Fig. 3. The two steps adder of two QSD numbers (each 16 QSD digits binary recoded) implemented in FPGA chip

Referring to Fig. 2, the data bus width is selected as 32-bit for both input data (the two arrays to be added) and output data (the result array) of the first and third parts of the QSD array adder, respectively. The two input arrays will enter the FPGA chip divided to pieces, each piece is 32 bits. Each 32 bits will be stored in the first memory location of a single memory block. The EP3C16F484C6 FPGA has 56 memory block each with 9 Kbits called M9K blocks. These memory blocks can be configured by different ways. The 32-bit width of a memory location is selected as required configuration. So, for two $(10 \times 2 \times 16)$ input QSD arrays which both have 1920 bits, the total number of the array pieces are 60. By the same way, the result array which has 1020 bit will be extracted from the FPGA by 32 pieces of 32 bits. Thereby, the address bus for the input part is 6-bit and for the output part is 5-bit for providing 60 and 32 memory addresses, respectively.

C. Performance of the proposed QSD adder

It is clear know, the input arrays required 60 clock cycles for completing storing the two QSD arrays inside the distributed memory locations, and the output array required 32 clock cycles for extracting the complete result array outside the FPGA chip. Because of the parallel addition of the QSD adder of the two stored QSD arrays, it need only two clock cycles to complete the two step addition. Also, one clock cycle is required for reading all memory locations of the first part to reach simultaneously to the second part, and also one clock cycle is needed to write the complete result array into the distributed memory location of the third part. Therefore, the total clock cycles of the QSD array adder is 96 for loading two arrays, adding two arrays, and extracting final result array.

The placing and routing operation of the FPGA implementation procedure reported that the maximum frequency of the clock signal is 142.8 MHz, i.e. the clock period approximately is 7 ns. This means the total processing time of the QSD array adder is 672 ns.

Fig. 4 illustrates the timing diagram for adding various QSD digits by the proposed QSD adder.

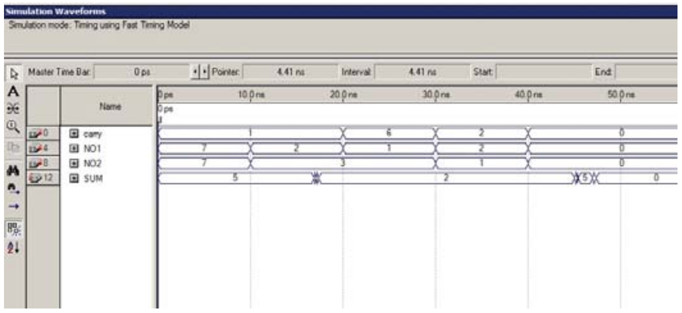

Fig. 4. Timing diagram of the QSD adder operation

V. Conclusions

It is clear that, the size of arrays and the length of the QSD numbers are depending on the FPGA chip and its logic resources. For larger arrays and longer numbers, other FPGA chips can be chosen which have more logic elements.

As expected, the FPGA-based hardware design has higher performance than the software program for array addition. For the two (10×2×16) QSD arrays addition, the FPGA-based array adder has 672 ns for complete array addition while the PC-based program for two (10×2×32) binary array (which equivalent of (10×2×16) QSD array) has 0.51314 ms.

REFERENCES

[1] K. Deepak, “Design and Analysis of High Speed and Low Power Reversible Vedic Multiplier Incorporating with QSDN Adder”, International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol.-8 Issue-4, pp. 273-279, February 2019.

[2] P. Dalmia,V. Vikas, A. Parashar, A. Tomar and N. Pandey, “Novel High speed Vedic Multiplier proposal incorporating Adder based on Quaternary Signed Digit number system,” 31th International Conference on VLSI Design, ieee computer society 2017, pp. 289-294, 2018.

[3] P. Gowthami and R. Satyanarayana, “Design of Digital Adder Using Reversible Logic”, IJERA, Vol. 6, Issue 2, pp.53-57, February 2016M.

[4] C. Sathish Kumar and P. Reddy, "Implementation of a Fast Adder Using QSD for Signed and Unsigned Numbers", International Journal of Science, Engineering and Technology Research (IJSETR), Vol. 3, Issue 11, pp. 3155-3160, November 2014.

[5] D. Jaina, K. Sethi and R. Panda, “Vedic Mathematics based multiply accumulate Unit,” IEEE International Conference on Computational Intelligence and Communication Systems, pp. 754-757, 2011.

[6] S. Saste1and A. Sawant, “Design and Implementation of Radix 4 Based Arithmetic Operations”, Advances in Intelligent Systems Research, Vol 137, pp. 800-809, 2017.

[7] M.Suneetha, S.Anilkumar, and P.Sivakrishna, “Design and Implementation of 2-Digit Adder using Quaternary Signed Digit Number System” International Journal of Electrical, Electronics and Computer Systems (IJEECS), Vol. 2, Issue 10, pp. 17-21, 2014.

[8] S.Mallesh and C. Narasimhulu, “Design of QSD Number System Addition using Delayed Addition Technique”, International Journal of Ethics in Engineering & Management Education, Vol. 1, Issue 10, pp. 1-4, October 2014.

[9] J. Moskal, E. Oruklu and J. Saniie, “Design and Synthesis of a Carry-Free SignedDigit Decimal Adder”, IEEE International symposium on Circuits and Systems, pp. 1089- 1092, 2007.

[10] A. Sruthi, Y. Raju and K. Mohan “Design and Implementation of ALU Using QSD in FPGA Technology” International Journal of Professional Engineering Studies (IJPRES), Vol. 5, Issue 3, pp. 18-24, AUG 2015.