Efficient New Design and Verification of Sign-Digit-Adder for Two Symmetric Redundant Radix-4 Numbers

Qasem Abu Al-Haija' Yathrip Al-Zahouri Maamoun Ahmed
Eng_Qasem1982@Yahoo.com *Ythrip_Zahouri@Hotmail.co.uk* *Maamoun.Ahmed@Bcs.org*

Jordan University of Science and Technology, Department of Computer Engineering, Jordan, Irbid 22110, P. O. Box 3030

***Abstract*—The carry propagation during the addition operation limits the speed of arithmetic operations. In Digital arithmetic, several approaches were proposed to provide the best implementation for addition operation of two operands with minimum carry propagation delay. As one of the major solutions for this problem is to use Signed-Digit-Adder (SDA) in which the (i^{th}) digit of the sum is exclusively dependant on the (i^{th}) digit. In this paper, we propose an Efficient Hardware Design and Verification of Signed-Digit-Adder for two Signed-Digit Radix-4 Operands benefiting from the inherent parallelism of the SDA's operation and the use of basic structural logic circuits at the gate level. This new design will form an adder for two singed-operands of symmetric redundant number representation system with a fixed radix-4 system. Simulation results shown that the Proposed work enhance the critical path delay of the SDA-unit by 55%.**

***Keywords*— Digital Arithmetic, Signed Digit Adder, Operand, Operation, Propagation Delay, Symmetric Redundant Number Representation System, Fixed Radix System, Carry Ripple Adder, Two's Complement Representation, Radix-4 System.**

I. INTRODUCTION

A Computer Arithmetic [2] is a subfield of digital computer organization that deals with the hardware realization of arithmetic functions. A major thrust is the design of hardware algorithms and circuits to enhance the speed of numeric operations. A Computer Arithmetic is divided into two major categories [1, 2]: Hardware which concerns of designing efficient circuits for arithmetic operations (+, -, x, /, sqrt, log...), and Software which forms the numerical methods for solving systems of equations [1, 2].

The arithmetic operations [1] form the largest field of computer digital arithmetic which operates on one, two, or more operands depending on the operation. The operation [1] is selected from an allowable set, which is usually includes addition, subtraction, multiplication, division, and so on. Addition process for two operands which is an arithmetic operation - that has resulted in a digital arithmetic numbers - has two steps to be calculated: Obtain carries then Compute sum.

The main problem with addition operation is the propagation delay of the carry through the adder circuit which forms the largest time in performing the addition operation. There are a lot of researches that concerns of effective designs for adders with the smallest time of carry propagation delay.

Signed-Digit-Adder (SDA) for two signed-digit operands [1] is one of the most effective adders that deal with two singed-operands of a symmetric redundant number representation system with a fixed radix system. It eliminated the carry obtain step by using such technique that perform the addition process without producing a carry, so that the transfer digit is propagates just one position. The operation of Signed-Digit-Adder appears in figure 1.

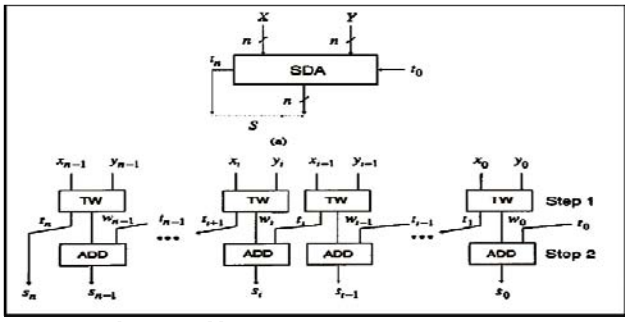

Fig.1.Signed-Digit Adder

This paper will focus on designing an efficient Signed-Digit-Adder for two radix-4 signed-digit operands of a symmetric redundant number representation system with a fixed radix system. Based on the algorithm of Signed-Digit-Adder [1], there are 3 cases that have to be compared using a comparator hardware technique that will perform the appropriate operation of two signed-digit numbers. Figure 2 shows the addition process which has two steps to complete the operation: Firstly-Obtain carries and then Compute sum. Signed-Digit-Adder eliminated the first step (obtain carry) which has the highest propagation delay (Tc) by such mechanism which make the (i^{th}) digit of the sum exclusively depends on the (i^{th}) digit.

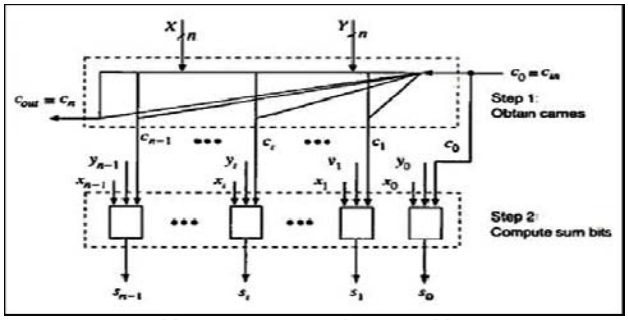

Fig.2.Steps in Addition using Signed-Digit Adder.

Also, the proposed work is motivated by many issues such as: the carry propagation during the addition limits the speed of arithmetic operation. A well-known property of Signed-Digit Adder is that the (i^{th}) digit of the sum is exclusively dependant on the (i^{th}) digit. Also, the digital arithmetic hardware units based on Signed-Digit-Adder is considered important for high speed and low-cost hardware realization. The Signed-Digit-Adder is well known to offer advantages in an arithmetic circuit implementation without the carry propagation. That why SDA became very important and interesting to design and verify.

In the next section, Section 2, the description and related works for SDA is briefly described as they appeared in the literatures. Section 3 presents our system modeling as the core of the implementation phase proposed in this work. In Section 4, we introduce the System equations that will be used in the implementation. The simulation environment and the implementation phase of all modules in our system are given in Section 5. A list of simulation results is given in Section 6, followed by the conclusion, Section 7.

II. RELATED WORKS

In the last years, many solutions tried to address the carry propagation delay problem that appeared in the addition arithmetic hardware unit. Many of these hardware algorithms used different forms [8] of adders to control the carry of the addition operation. Several solutions use SDA as one form of adders that has significant effect in minimizing the delay of addition operation. These solutions appear in [3, 4, 5, 6, and 7].

Katsumi Wasaki in [3] used the calculation models proposed for arithmetic logic units based on many sorted algebras to verify the structure and design of these circuits using the Mizar proof checking system. He proved the stability of a circuit for a redundant SDA circuit example based on definitions and theorems for logic operations, hardware gates, and signal lines. He used Mizar proof checking system as a formal verification tool.

Masaaki Niimura and Yasushi Fuwa in [4] proposed Radix-2^k sub signed-digit numbers in consideration of the hardware realization as a high speed adder algorithm using this Radix-2^k sub signed-digit numbers. Their method, one is “bit compare “at carry calculation and the other is carry calculation between two numbers. They showed that n digits Radix-2^k signed-digit numbers is expressed in n + 1 digits Radix-2k sub signed-digit numbers, and addition result of two n+1 digits Radix-2^k sub signed-digit numbers is expressed in n+1 digits.

Yoshinori Fujisawa and Yasushi Fuwa in [5] defined the radix-2^k signed-digit number (Radix-2^k SD number) and discussed a high-speed adder algorithm.

Dhananjay S. Phatak in [6] proposed a novel hybrid number representation with twos complement representation and the signed-digit representation as special cases. It provided bounding on the maximum length of carry propagation chains during addition to any desired value between 1 and the entire word length. He presented several static CMOS implementations of a two operand adder which employ the proposed representations.

Aldirdas Avizienist in [7] described a class of number representations which are called signed-digit representations. He discussed the properties of signed-digit representations and arithmetic operations with signed-digit numbers: addition, subtraction, multiplication, division, and round off. He presented a brief discussion of logical design problems for a SDA.

Our proposed work is trying to handle the problem of the carry propagation during the addition limits the speed of arithmetic operation by benefiting from the well-known property of SDA that is the (i^{th}) digit of the sum is exclusively dependant on the (i^{th}) digit. We propose a new hardware design and verification for SDA for two Symmetric Redundant Radix-4 Numbers.

III. PROPOSED SYSTEM MODELING

The basic idea behind this work is to design a structural circuit at the gate level that allows us to add two numbers (Operands), where these numbers are:

- Radix-4 number representation system (r = 4).
- Symmetric numbers: which means that these numbers will fall in the symmetric range, where the digit set of both operands are the same and from -a to +a. So in this work which based on Radix-4, the digit set will be D= {-3... +3\}.
- Redundant Representation which must be satisfy the following equation: a > (r-1)/2. So, for our system this equation is satisfied where 3 > 3/2.
- Two's Complement system where each digit in radix-4 system will need 3-bits to be represented in Two's Complement Binary system.

We start our system model as we see from figure 3 according to the proposed methodology by initially add the two digits Xi, Yi which they are originally represented in Symmetric Redundant Radix-4 system with the range [-

3, 3] so we need 3-bit to represent them in two's complement binary system.

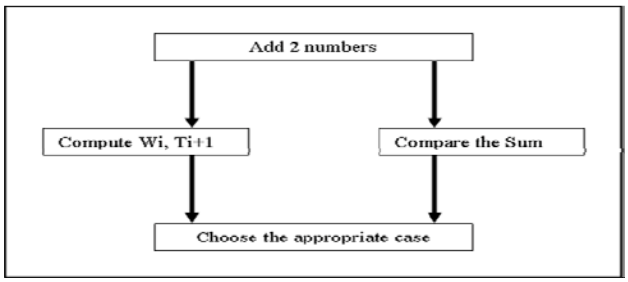

Fig.3.The Proposed Methodology

Also, the addition of two numbers each with 3-bit will result maximally in 4-bit number (Sum and Carry Out), so that, according to the algorithm, Firstly: this result (Zi) must be compared with (-a+1) and (+a-1) in the same level. Secondly: the result (Zi) will be added to (-r) resulting in (Zi+ (-r)) or added to (+r) resulting in (Zi+r) using CRA-4 bit or it will be transferred as is (Zi).
The comparison process will be done in parallel with the addition process (CRA-4) and then the result of comparison will be used as selection lines of the multiplexers which they judge the appropriate result that will be taken. By that way, we determined the value of Wi and Ti+1. These operations appear more clearly in fig.4.

Our System Model consists of the following modules:

- **Multiplexer Module**: The 4-Bit multiplexer module (figure 5) consists of 2 Invertors gates in parallel, 4 AND gates in parallel, and one OR gate to choose one result.

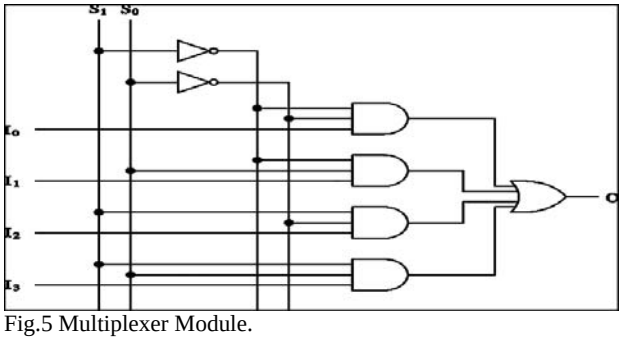

Fig.5 Multiplexer Module.

- **CRA 3-bit implementation Module**: It is a carry Ripple Adder (Figure 6) for 3-bit from Xi, 3-bit From Yi, and Carry-in where each bit from {Xi, Yi, Ci} needs One Full adder, so we need 3-Full Adders in a ripple fashion to add those 3-bit.

- **CRA 4-bit implementation Module**: Its a Carry Ripple Adder (Figure 7) for 4-bit from Zi, 4-bit From r{-4 or r}, and Carry-in where each bit from {Zi, Ri, Ci} needs One Full adder, so we need 4-Full Adders in a ripple fashion to add those 4-bit.

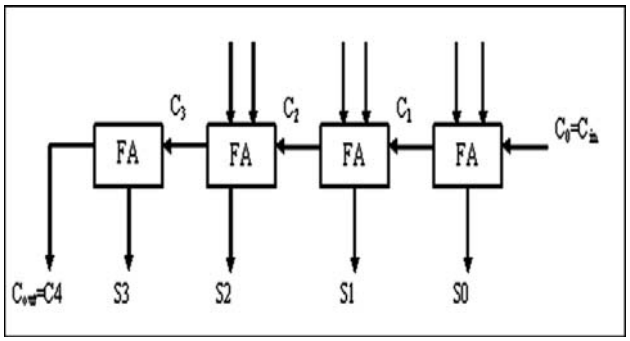

Fig.7 CRA4-bit implementation Module.

- **4-bit Comparator Module**: The following circuit (Figure 8) tests the inequality a > b, by first testing to see whether a3 is 1 and b3 is 0 and doing the same test for lower bits if these bits are equal.

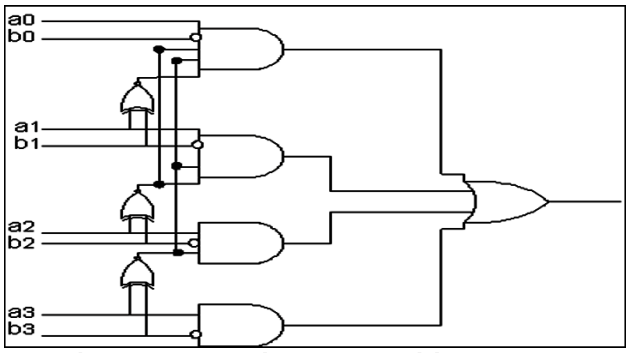

Fig.8 4-bit Comparator implementation Module.

- **Full Adder Module**: Figure 9 shows the internal gate design of basic Full Adder.

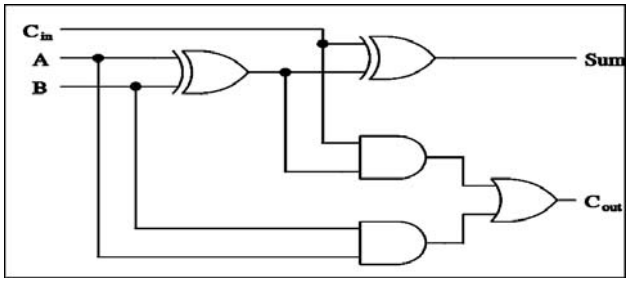

Fig.9 Full Adder Module.

$$(T_{i+1}, Wi) = \begin{cases} (0, X+Y) & ; \quad -2 <= X+Y <= 2 \\ (1, X+Y-4) & ; \quad 2 < X+Y <= 6 \\ (-1, X+Y+4) & ; \quad -6 <= X+Y < -2 \end{cases}$$

Fig.10 Proposed Algorithm

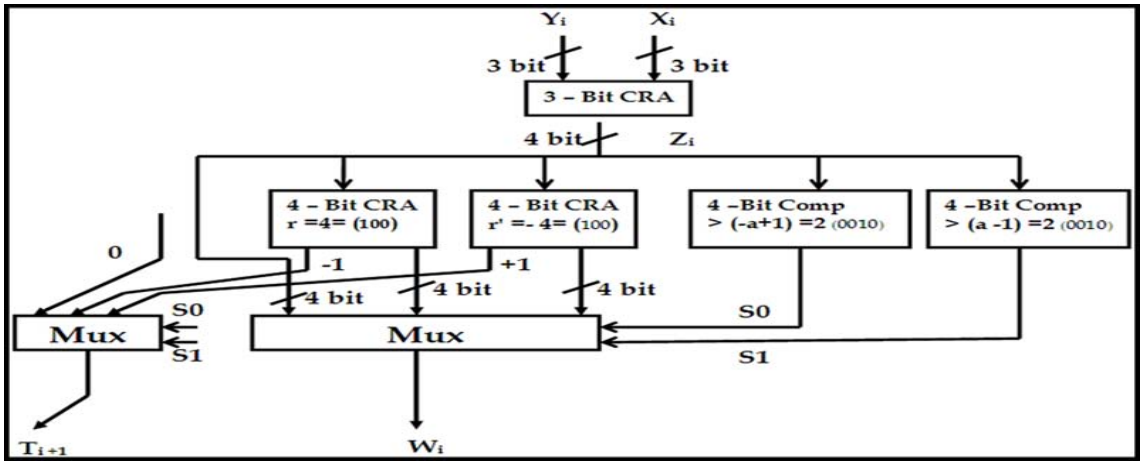

Fig.4 Our System Model

As we see from previous model the critical path delay can be computed as the following: (note that the comparators work in parallel with CRA-4 bit; done in the fly), so:

Critical Path Delay =
CRA (3-Bit) + CRA (4-Bit) + 1Mux

= 3FA + 4FA + 1Mux = 7FA + 1 Mux
= 14XOR + 1MUX
= 14XOR + 1INV + 1AND + 1OR
= 14(0. 3 + 0.029 * 2) + 1(0.04 + 0.028*1) + 1(0.16 + 0.027*2) + 1(0.16 + 0.028*1)
= 5.474 ns

As an average it takes about 5.5 CC, which make this design have to be minimized as much as possible, so we have to decrease this delay by minimizing the slowest gates used such as XOR. That's what we will work on it in the System equations.

IV. OUR SYSTEM EQUATIONS

The system module has X = (X2, X1, X0) and Y = (Y2, Y1, Y0) as an input and the output is also two vectors: the intermediate values W = (W2, W1, W0) and transfer value T = (T1, T0). This values will be used later to generate the final sum. This system uses radix-4 signed digit adder where:
R = 4, symmetric digit set Di = {-3, -2...2, 3}, and a = 3.

In this case - Our case: the two operands are X, Y, the Sum X+Y will be calculated like this (Figure 10):

Where Si = Wi + Ti, Ti in {-1, 0, 1}, and Wi in {-2, ..., 2}.

Our System is divided to three main steps:

Step One: Calculate the Z = X + Y.
For this step we use 3-bit carry ripple adder the sum Z= (Z3, Z2, Z1, Z0) as in the figure 11. Where the Full Adder (FA) consist of two half adders to generate the sum Si = [(Xi XOR Yi) XOR Ci], and the carry to the next position Ci+1 = [(Xi XOR Yi) AND Ci) OR (Xi AND Yi)]. In this system, the last bit of Z which is Z3, the sign of the result.

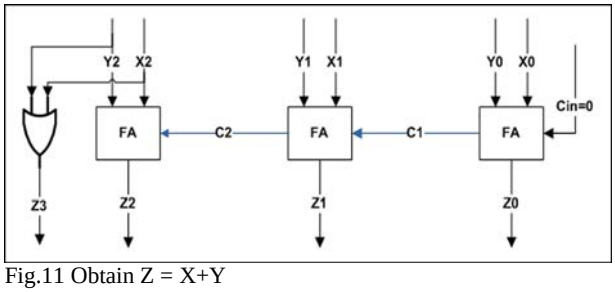

Fig.11 Obtain Z = X+Y

Step Two: The selection.
In this step, we need to know where the result falls in, so we need to compare it to the boundaries, two modules are used to generate two selection signals S0, S1:

Greater Than 2: S0=1: This module (Figure 12) will compute the signal S0, which is equivalent to the case when the sum result Z is greater than 2.
Z in {3, 4, 5, 6}, S0 = 'Z3 AND ((Z1 AND Z2) OR (Z1 AND (Z0 XOR Z2))).

Less Than -2: S1=1: This module (Figure 13) will compute the signal S1, which is equivalent to the case when the sum result Z is less than -2.
Z in {-6, -5, -4, -3}, S1 = Z3 AND (Z1 XOR Z2).

These selection lines used to calculate X + Y - 4 and X + Y + 4 (Figure 14). So it is just invert the Z2 in both cases (Z3 is not included in W cause W in {-2... 2}, need just 3 bits).

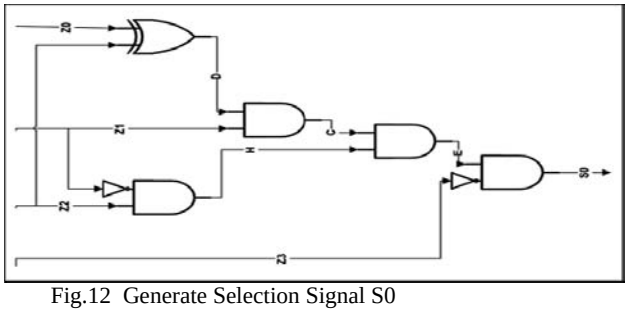

Fig.12 Generate Selection Signal S0

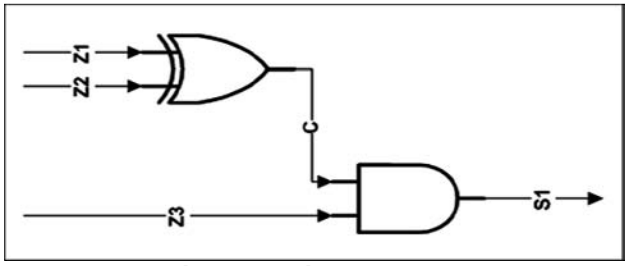

Fig.13 Generate Selection Signal S1

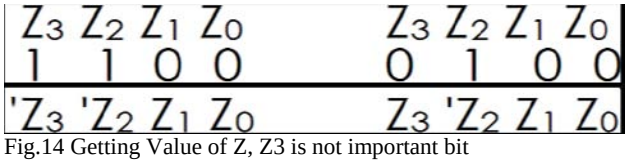

Fig.14 Getting Value of Z, Z3 is not important bit

Step Three: The Final result.

In this step we will obtain the corresponding bits for W = (W2, W1, W0), and T = (T1, T0); this will be done through:

- The First Case: where Z > 2 so S0 = 1, S1= 0. We note that: W0 = Z0, W1 = Z1, W2 = 'Z2 , T0 = 1, T1= 0
- The second case: where Z < -2 so S0 =0, S1 = 1. We note that: W0 = Z0, W1 = Z1, W2 = 'Z2, T0 = 1, T1 = 1
- The final case: where 2 <= Z <= 2 so S0 = 0, S1 = 0. We note that: W0 = Z0, W1 = Z1, W2 = Z2, T0 = 0, T1 = 0

From the three cases above we note that:
1. W0 = Z0, W1 = Z1 in all cases.
2. T1 = S1.
3. T0 = S1 OR S0.
4. W2 = 'Z2 When S1 OR S0 equal 1.
5. W2 = Z2 When both S1, S0 equal 0.

For these results see figures 15, 16.

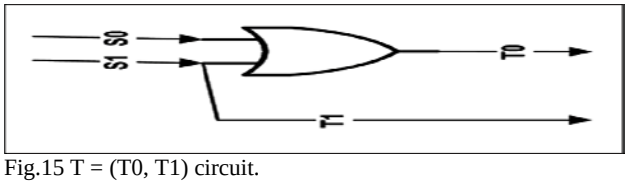

Fig.15 T = (T0, T1) circuit.

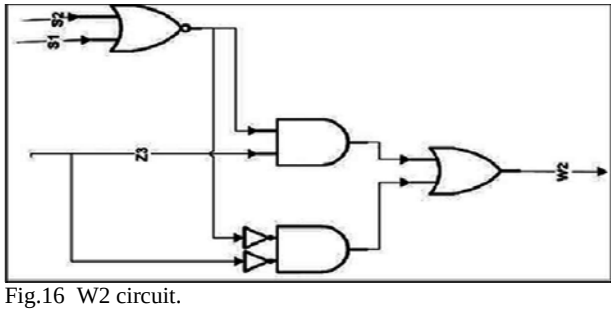

Fig.16 W2 circuit.

The final system modules and components explained in the figure below, Figure 17.

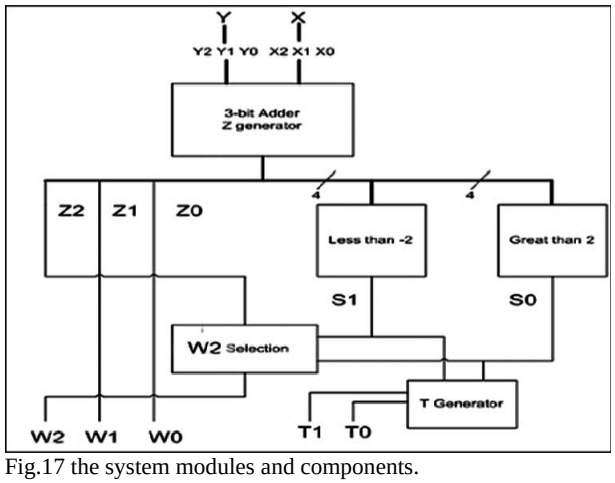

Fig.17 the system modules and components.

V. SIMULATION ENVIRONMENT & IMPLEMENTATION

The proposed implementation is programmed (Described) and implemented using VHDL [10] language which is a Hardware Description Language that was developed by the Institute of Electrical and Electronic Engineers (IEEE) as a standard language for describing the structure and behavior of digital electronic systems. It has many features appropriate for describing the behavior of electronic components ranging from simple logic gates to complete microprocessors and custom chips. Features of VHDL allow electrical aspects of circuit behavior (such as rise and fall times of signals, delays through gates, and functional operation) to be precisely described. The resulting VHDL simulation models can then be used as building blocks in larger circuits (using schematics, block diagrams or system-level VHDL descriptions) for the purpose of simulation.

As a compiling and simulation tool for VHDL, we used the ModelSim XE III 6.2g [11] which is known as a powerful tool developed by Mentor Graphics Company to offer an appropriate environment to validate the functional correctness of the design hardware.

In our implementation, the system module is named as SDA which contain five main components:

1- Adder_3 (X, Y: input, Z: output)
This component is used to generate the intermediate sum Z= X + Y, but it is adapted from the original 3-bit CRA to accommodate the sum range {-6... 6}.
The carry of the last bit is omitted and replaced by the sign of two operands.

2- Great_2 (Z: input, S0: output)
This component is used to generate the selection signal S0 as a function to intermediate sum bits, as an output S0 will be 1 only when Z > 2, otherwise S0 will be 0.

3- Less_2 (Z: input, S1: output)
This component is used to generate the selection signal S1 as a function to intermediate sum bits, as an output S1 will be 1 only when Z < -2, otherwise S1 will be 0.

4- Mux_w_2 (Z2, S0 S1: input, W2: output)
This component is used to select the third bit W2 of W, according to the selection signal S0 S1 the W2 will be either Z2 or 'Z2.

5- Mux_T (S0 S1: input, T0 T1: output)
This component is used to generate the transfer value Ti according to the selection signal S0 S1 the T will be either 00 (0), 01 (1) or 11 (-1).

VI. SIMULATION RESULTS

We can compute the critical path delay of our proposed model to show the benefits of using the basic gates and the new components with less delay.

Critical Path Delay =

3 XOR + 1 XOR + 3 AND + 1 OR + 1 AND + 1 OR + 1 OR + 1INV

= 4 XOR + 1 INV + 4 AND + 3 OR

= 4(0. 3 + 0.029 * 2) + 1(0.04 + 0.028*1) + 4(0.16 + 0.027*2) + 3(0.16 + 0.028*1)

= 2.92 ns

As an average it takes about 3.0 CC, which makes our design improved over the original design by (3/5.5 * 100 %) = 55%.

In the figures (18-20), we list some of our simulation results arranged from input (X, Y) to intermediate sum Z following with the selection signal S and finally the outputs W and T. We notice that the result is accommodating with the algorithm and the system module.

T T_0T_1	W $W_2W_1W_0$	S S_1S_0	Z $Z_3Z_2Z_1Z_0$	Y $Y_2Y_1Y_0$	X $X_2X_1X_0$
00	000	00	1000	101	011
00	001	00	1001	110	011
00	010	00	1010	111	011
01	111	01	0011	000	011
01	000	01	0100	001	011
01	001	01	0101	010	011
01	010	01	0110	011	011

Fig.18 Result Table for X= 3 with all possibilities

T T_0T_1	W $W_2W_1W_0$	S S_1S_0	Z $Z_3Z_2Z_1Z_0$	Y $Y_2Y_1Y_0$	X $X_2X_1X_0$
11	110	10	1010	101	101
11	111	10	1011	110	101
11	000	10	1100	111	101
11	001	10	1101	000	101
00	110	00	1010	001	101
00	111	00	1111	010	101
00	000	00	1000	011	101

Fig.19 Result Table for X=-3 all possibilities.

T T_0T_1	W $W_2W_1W_0$	S S_1S_0	Z $Z_3Z_2Z_1Z_0$	Y $Y_2Y_1Y_0$	X $X_2X_1X_0$
11	001	10	1101	101	000
00	110	00	1110	110	000
00	111	00	1111	111	000
00	000	00	0000	000	000
00	001	00	0001	001	000
00	010	00	0010	010	000

Fig.20 Result Table for X=0 with all possibilities

VII. CONCLUSIONS

Our Conclusions throughout this research can be listed as follow:

- Sign Digit Adders one of the adders that is represented using a redundant digit set. The operands might be in conventional representation, or one or both also use a redundant set. So having the output in redundant representation will reduce the addition time by reducing the length of the maximum carry propagation-chain.
- Sign Digit Adder is the appropriate adder to use whenever the output in redundant representation, typical cases of this are in accumulation, multioperand addition, multiplication, division, and square root.
- The use of Sign Digit Adder will increase the number of bits required for representation, which depends on the degree of redundancy.
- The use of Sign Digit Adder make it difficult for some operations, such as magnitude comparison and sign detection, they will be difficult to perform

in a redundant representations such that Sign Digit Adder.

And as a future works for this research, this work can be extended and enhanced by:

- Extend the capabilities of this work to include any radix representation system.
- Imply all Sign Digit Adder to work in one coherent system that accept any two operands which may be two signed digit numbers, two conventional numbers or one signed digit and one convectional number.
- Trying to use the Asymmetric Digit set where Xi, Yi will be in the range from {--b... +a} rather than use the symmetric digit set where Xi, Yi in the range {-a... +a}.
- While this approach cannot be applied in the symmetric binary system {-1, 0, +1} where it needs to use the double recording algorithm, so it's better to include this special case.

VIII. REFERENCES

[1] Milos D. ERCEGOVAC, Tomas LANG, " Digital Arithmetic," Morgan Kaufmann Publishers, by Elsevier Science (USA), Vol1, Ch2, pages (51-136), 2004.
[2] Mary Jane Irwin, "Computer Arithmetic,", "www.cse.psu.edu/~mji"s, spring, 2005, CSE 575.
[3] Katsumi Wasaki, " A Verification for Redundant Signed Digit Adder Circuits", "wasaki@cs.shinshu-u.ac.jp, Man, Faculty of Engineering, Shinshu University, Nagano, Japan, 2005.
[4] Masaaki Niimura,Yasushi Fuwa ,"High Speed Adder Algorithm with Radix-2^k Sub Signed-Digit Number", Faculty of Engineering ,Shinshu University, Nagano, Japan, 2003.
[5] Yoshinori Fujiswa and Yasushi Fuwa, "Definitions of Radix-2^k Signed-Digit Number and its Adder Algorithm," Faculty of Engineering, Shinshu University, Nagano, Japan, 2001.
[6] Dhananjay S. Phatak," Hybrid Signed Digit Number Systems: A Unified Framework for Redundant Number Representations with Bounded Carry Propagation Chains", Electrical Engineering Dept., State University of New York, phatak@ee.binghamton.edu, IEEE Transactions on Computers, vol.43, No.8, August 1994.
[7] Algirdas Avizienist," Signed Digit Number Representations For fast parallel Arithmetic", IRE Transactions on Electronic computers, May 1961.
[8] Martin Horauer and Dietmar Loy, "ADDER SYNTHESIS", Institute of Computer Technology., University of Technology Vienna, Emails: horauer,loy@ict.tuwien.ac.at, Supported by the Austrian Science Foundation (FWF).
[9] Linda Null,Julia Lobur " The Essentials of Computer Organization and Architecture,", "Pennsylvania State University", Jones and Bartlett Publishers, Canada, 2003.
[10] Douglas L. Perry, " VHDL: Programming by Example," Forth Edition: Tool usage for simulation and synthesize, Copyright by the McGraw-Hill Companies, 2002.
[11] Mentor Graphic Corporation, "ModelSim Tutorial Software Version 6.2g," www.mentor.com, February 2007.
[12] Dr.Shankar Balachandran, " CS 130: Computer Systems - III," Dept. of Computer Science & Engineering, IIT Madras, and Sep 13, 2006.
[13] www.xilinx.com, " Foundation Series ISE 3.1i In-Depth Tutorial: Watch Design", www.xilinx.com, Company Tutorial, UG101 (v1.0) July 21, 2000.}

Author Biography (Corresponding Author)

Eng. Qasem Abu Al-Haija' is computer engineer and information security / Cryptography researcher. He was born in a city north of Jordan called Irbid in 1982. He received his B.S. in Electrical and Computer Engineering from Jordanian Mu'tah University in February of 2005. Then he worked as a network engineer in a leading institute at KSA, and as a lecturer before he joined the graduate program at Jordan University of Science & Technology (JUST) in September 2007 Eng. Qasem received his M.S. degree in Computer engineering from Jordan University of Science & Technology under the direction of Dr. Lo'ai Tawalbeh in December 2009. Eng. Qasem research interests include Cryptography and Security, Computer Arithmetic and Finite Fields, Hardware implementations for cryptography, Wireless Sensor Networks, FPGA design, Elliptic Curve Cryptography, computer architecture, digital arithmetic algorithms. Hobbies: Reading-Writing, Swimming, Football.