A Novel High-Speed Low-Power Binary Signed-Digit Adder

Somayeh Timarchi[1], Parham Ghayour[1], and Asadollah Shahbahrami [2]

[1]*Department of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran*

[2]*Department of Computer Engineering, University of Guilan, Rasht, Iran*

***Abstract*— Addition is one of the most important arithmetic operations in digital computation. Optimization of adders' speed, power, and area is a challenging task. To this end, redundant number system has been proposed in the literatures. In this paper, we propose a new redundant binary signed-digit adder that not only utilizes specific encoding for the input operands, but also uses a new efficient adder structure. Using this technique we can generate low power signed digit adders that perform fast additions. The comparisons show delay, power and area reduction both on FPGA and Synopsys Design Vision tool.**

***Keywords*— Redundant addition; binary signed digit number system; high-speed low-power arithmetic; FPGA; VLSI**

I. INTRODUCTION

Addition operation has a significant role in digital arithmetic operations like multiplication and division. However, carry propagation is an important issue in addition operations which depends on the operand length. The case results in low-speed and high-complexity circuit.

Using redundant representation, like Binary Signed-Digit (BSD) and high-radix signed-digit number systems can eliminate the carry propagation in arithmetic operations [1-3]. The BSD representation has been used to organize constant time adders due to the capability of carry-free addition and regular VLSI layout. Redundant representations have also some benefits in residue number system [4-8].Therefore, it is important to use an appropriate encoding and efficient design for BSD addition.

The BSD number system is defined for radix $r = 2$ with digit set ranging over the set {-1,0,1}. In order to use BSD number system, the digit set needs to be encoded in binary representation. Several encoding schemes have been discussed in the literatures [1, 4, 9, 10] for BSD. Another important problem is to design a high performance carry-free adder utilizing the encoding schemes.

There have been many efforts to optimize the speed, power and area of BSD adders using different methods which have used various encodings, such as Binary Canonical Signed Digit (BCSD) [9] and Quaternary Signed Digit (QSD) [10] representations. One of the most efficient conventional BSD adder proposed in [4, 5] is composed of n BSD full adders (SDFAs).

In this paper a new high-speed low-power BSD adder is proposed which utilizes a specific encoding for BSD operands. The encoding is based on the recently presented bit representation in [11]. The proposed adder structure utilizes standard non-redundant binary addition components like optimized counters and compressors.

The paper is organized as follows. In Section II we provide a technical background of BSD adders and, briefly review the relevant prior works in the field. In Section III, our new proposed BSD adder is described. In Section IV, we present the simulation and comparison and finally, in Section V the concluding remarks are presented.

II. BACKGROUNDS

A. SD Number System

Signed-digit representation is a class of number representations which limits carry propagation during the addition and subtraction. Using redundant representation for the operands we can remove the carry-propagation chains. The speed of addition and subtraction will be improved by redundancy. In this representation each digit of a positional constant radix r number representation can have q values. In order to have carry-free addition, q has to be in the following range [1]:

$$r + 2 < q < 2r - 1$$

A signed-digit number Z represented by n digits (i= 0, 1,… , n) has the numerical value:

$$Z = \sum_{i=0}^{n} Z_i \times r^i$$

B. SD Adder

Redundant binary adders have wide applications in arithmetic circuits because of their constant time addition property. The recently published BSD adder is a n-digit modulo m BSD adder (MSDA) with n BSD full adders (SDFAs), which each SDFA consists of two sub-circuits, ADD1* and ADD2* [5] as shown in Fig. 1.

In this figure, $t_i \in \{-1,0,1\}$ and is used to express the sign of $x_i + y_i$ as

$$t_i = \begin{cases} 1 & : \ x_i + y_i < 0 \\ 0 & : \ x_i + y_i \geq 0 \end{cases}$$

The modular BSD adder can be simply converted to a non-modular BSD adder by removing the end-around carry addition (considering $\mu = 0$ and $v_{-1} = 0$ in Fig. 1).

978-1-4673-1482-4/12/$31.00 ©2012 IEEE

TABLE I

RULES FOR ADDING BSD NUMBER [5]

	$abs(x_i)=abs(y_i)$[1]	$abs(x_i)\neq abs(y_i)$	
		$(x_i+y_i)\times(x_{i-1}+y_{i-1})\geq 0$	$(x_i+y_i)\times(x_{i-1}+y_{i-1})\geq 0$
u_i	t_{i-1}	$t_{i-1}-xpy_i^2$	$t_{i-1}+xpy_i$
v_i	$t_{i-1}+xpy_i/2$	t_i	t_i+xpy_i

(Note1: abs(n) is the absolute value of the n. Note 2: $xpy_i=x_i+y_i$)

TABLE II

SYMBOLIC AND DOT NOTATION OF POSIBIT, NEGABIT AND UNIBIT [7]

Bit Name	Lower and upper values	Dot Notation	Symbolic Notation	Lower Value representation	Upper Value Representation
Posibit	{0,1}	•	x_i^+	0 (0)	1(1)
Negabit	{-1,0}	○	x_i^-	$\underline{0}$ (-1)	$\underline{1}$ (0)

In that paper a new addition rule for simplifying the BSD adder and generating the intermediate sum and carry with a binary number representation is proposed that is shown in Table I.

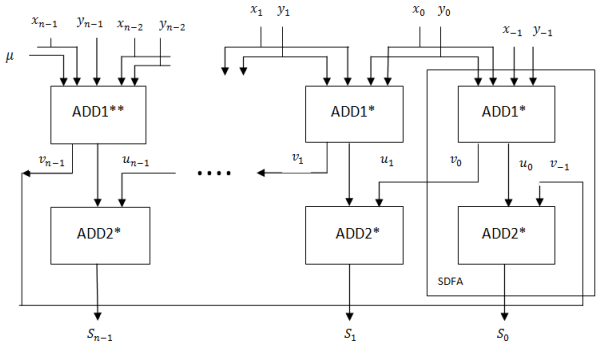

Fig. 1 Modulo m Signed-Digit Adder (MSDA) [5].

Each MSDA depicted in Fig. 1 is constructed from n SDFA blocks. The input of SDFA (ADD1*) is four redundant digits (x_{i-1}, y_{i-1}, x_i, and y_i) and the output of ADD1* (intermediate sums) are u_i and v_i . The algorithm of computing u_i and v_i is described in the following.

ADD1*: in each digit u_i and v_i are calculated by table I, where, $i \in \{0,1, \ldots, n\text{-}1\}$;

ADD2*: u_i is subtracted from v_{i-1} that meets:

$$s_i = v_{i-1} - u_i$$

it is assumed that $v_{-1} = 0$.

From Fig. 1 and the addition rules, it is concluded that the i-th final sum digit depends only on the digits in i-th and (i-1)-th positions. The final sum can be computed with a guarantee that no carry propagation chain will occur and the completion time will be independent of the operand size. Thus, a constant time BSD adder can be constructed based on the previous addition method using SDFA blocks.

III. Proposed BSD Adder

A. BSD Encoding

In the proposed BSD adder we utilize a specific encoding for representing BSD numbers. According to Table II, a bit in the range {0,1} and a bit in the range {-1,0} are called posibit and negabit, respectively. A BSD digit x_i is represented by a posibit, x_i^+ and a negabit, x_i^- with equal weight. It should be emphasized that a negabit with inverted encoding is employed for BSD representation. It means that a negabit with two values -1 and 0 requires one bit where they are represented by 0 and 1, respectively. Each BSD digit x_i is represented with [x_i^+ x_i^-]. According to Table II, the BSD three values, -1, 0, and 1 are represented by [00], [01] and [11], respectively.

This notation has advantages not found in other redundant binary representation. It is possible to easily invert a value by reversing all the bits of the represented value using NOT gates. This allows building adder/subtracter unit more easily.

B. Proposed BSD Adder

Although addition in non-redundant two's complement number system is proportional to log(n) (where n is the bit width), as discussed in the previous sections, addition in BSD representation can be done in constant time. This can be explained by the fact that the carry does not propagate through all the width of the addition unit. This does imply that the addition will finally be faster in BSD.

In this section, we utilized a new encoding for BSD representation. Therefore, there are four bits (two posibits and two negabits) in each position i, $i \in \{0,1, \ldots, n\text{-}1\}$ (where n is the operand width) which has to be accumulated. Since a negabit with inverted encoding is employed for BSD representation, standard counters and compressors can be utilized to accumulate the digits. So, a high-speed low-power BSD adder will be achieved for BSD operands.
In this paper we use standard 4-2 compressor for BSD addition. The diagram of the 4-2 compressor-based BSD adder is represented in Fig. 2 which employs two full-adders (FA).

The 4-2 compressor has 4-bit input x_i (x_i^+ x_i^-) and y_i (y_i^+ y_i^-), 2-bit outputs s_i^+ (posibit) and c_i^- (negabit), a single input carry c_{in}^+, and an output carry c_{out}^+. The five equally weighted inputs produce a sum output s_i^+ with the same weight as well as c_i^- and c_{out}^+ with double-weight. To speed-up BSD additions we can use an efficient structure for compressor. We utilized a different compressor which is represented in Fig. 3. The compressor is composed of six multiplexers [12].

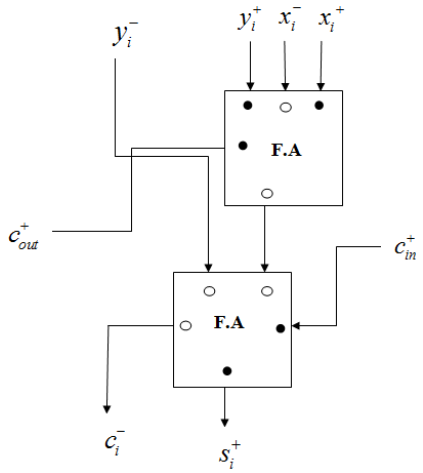

Fig. 2. FA-Based Compressor Used for BSD Digit Adder.

Example 1: Let's assume $x_i = 1$ and $y_i = -1$ and $c_{in}^+ = 1$. Find s_i^+, c_i^- and c_{out}^+.

Solution:
BSD representation using inverted encoding for $x_i = 1$ and $y_i = -1$ are as follow:

$x_i^+ : 1,\ x_i^- : 1$

$y_i^+ : 0, y_i^- : 0$

According to Fig. 2 we obtain:

$x_i^+ + x_i^- + y_i^+ \rightarrow D=0,\ c_{out}^+=1$

$D + c_{in}^+ + y_i^- \rightarrow s_i^+ = 1, c_i^- = 0$

In the FA-based compressor, s_i^+ and c_{out}^+ are posibits and c_i^- is a negabit. Therefore, the numerical value of output is:
$s_i^+ - 2\,(1 - c_i^-) + 2\,c_{out}^+ = 1 - 2\,(1) + 2(1) = 1$
which is the correct summation of inputs.

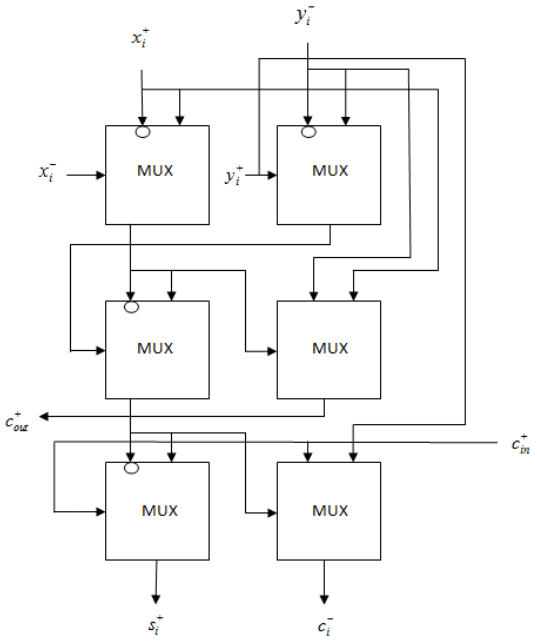

Fig. 3. MUX-Based Compressor with 6 Multiplexers Used for Each BSD Digit Adder [12].

TABLE V
BSD Addition in EXAMPLE 2

position: i	4	3	2	1	0
Operand1: x	1	0	-1	1	1
Operand2: y	-1	0	1	0	1
x_i^+	1	0	0	1	1
x_i^-	1	1	0	1	1
y_i^+	0	0	1	0	1
y_i^-	0	1	1	1	1
Intermediate carry: c_{out}^+	0	0	1	1	0
Final sum: s					
s_i^+	0	0	1	0	0
c_i^-	1	1	1	1	1
Result	0	0	1	0	0

TABLE III

SYNTHESIS RESULTS OF THE BSD ADDERS ON FPGA.

Delay (ns)			Slices usage			Levels of logic		
[5]	Proposed MUX-based BSD adder	Proposed FA-based BSD adder	[5]	Proposed MUX-based BSD adder	Proposed FA-based BSD adder	[5]	Proposed MUX-based BSD adder	Proposed FA-based BSD adder
12.166	9.717	9.344	71	34	36	6	5	4

TABLE IV

SYNTHESIS RESULTS OF THE BSD ADDERS WITH SYNOPSIS.

Delay (ns)			Area(μm^2)			Power (mW)		
[5]	Proposed MUX-based BSD adder	Proposed FA-based BSD adder	[5]	Proposed MUX-based BSD adder	Proposed FA-based BSD adder	[5]	Proposed MUX-based BSD adder	Proposed FA-based BSD adder
0.4	0.28	0.25	3964.97	2564.77	1624.82	2.9827	2.1574	1.2947

Example 2 : let's assume $x = 15$ and $y = -11$ and $c_{in} = 0$. Find s, c and c_{out} .

Solution:

Using redundant representation for $x = 15$:

x: $(1\ 0\ -1\ 1\ 1)_{BSD}$

x_i^+ : 1 0 0 1 1

x_i^- : 1 1 0 1 1

Using redundant representation for $y = -11$:

y: $(-1\ 0\ 1\ 0\ 1)_{BSD}$

y_i^+ : 0 0 1 0 1

y_i^- : 0 1 1 1 1

Using five compressors of Fig. 3, we obtain Table V which gives us the answer**.**

IV. SIMULATION & COMPARISON

To evaluate the speed, area and power dissipation of the considered architectures, the structural VHDL descriptions of the BSD adders, consisting of BSD adder proposed in [5], proposed MUX-based, and proposed FA-based BSD adders have been first generated. After verifying the correctness of each description, we synthesized them on FPGA, Xilinx Spartan 3 [13-15]. The synthesis results are represented in Table III. As shown in the table, the proposed adders have overcome the adder of [5] in FPGA platform. This means it has less Delay, less resource usage, and less logic levels than the adder presented in [5].

We also synthesized the structures for 130nm CMOS technology with the Synopsys Design Vision tool. A typical corner (1.2V, 25 C) was considered. The results of total power, delay, and area for both proposed adders are included in Table IV. Delay, area, and power results are given in ns, μm^2, and mW, respectively.

The results indicate that area, delay and power of proposed methods decrease in comparison to the most efficient existing adder. We note that the new encoding approach offers several advantages as it make use of the same computational units as in the non-redundant adder, like standard compressors as indicated in Fig. 2 and Fig. 3. Moreover, we can apply any fast compressor to the proposed design. Therefore, the proposed adder has a simpler implementation than other existing BSD adders and requires a simple design and modification procedure.

V. CONCLUSION

In this paper we presented a new BSD adder that utilizes a specific encoding for BSD numbers as well as a specific adder structure. Using this BSD adder we achieved higher speed, lower power and lower area BSD adder than the most efficient existing adder. The significant improvements were obtained on both FPGA and Synopsys. Besides, we succeeded to decrease about %60 area, %37.5 delay and, %30 power consumption of the most efficient existing BSD adder.

These improvements are achieved based on selecting an efficient encoding for BSD operands. The encoding results in utilizing standard counters and compressors for redundant BSD operands instead of employing more complex counters for them as proposed in the related articles. Therefore, applying [00], [01] and [11] to the three values, -1, 0, and 1, respectively, cause to a high performance BSD adder with less hardware overhead, delay, and power consumption in comparison to the previous published structures. These important achievements can

REFERENCES

[1] Avizienis, A., 1961. Signed-digit number representations for fast parallel arithmetic. *IRE Tmnsactions on EIectronic Computers,* EC-10:389-400.

[2] Timarchi, S., Navi, K., 2009. A New Aalgorithm for Determining All Possible Symmetric Hybrid Redundant Numbers, *IEICE Electronics Express,* vol.6, no.1, pp.8-13, Jan. 10, 2009.

[3] Timarchi, S., Navi, K., and Kavehei, O., 2009. Maximally Redundant High-Radix Signed-Digit Adder: New Algorithm and Implementation, *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, May 13-15, Tampa,Florida.

[4] Wei, Sh., 2009. Modular Multiplier Using a Modified Residue Addition Algorithm with Signed-Digit Number Representation, *IMECS*, pp. 494-499, Mar. 2009.

[5] Zhang, M., Wei, Sh., 2010. "High-Speed Modular Multipliers Based on a New Binary Signed-Digit Adder Tree Structure", pp.615-619.

[6] Timarchi S., and Navi, K., "Efficient class of redundant residue number system," *IEEE International Symposium on Intelligent Signal Processing (WISP),* Madrid, Spain, pp. 475-480, 3-5 Oct. 2007.

[7] S.Timarchi, K.Navi, "Arithmetic Circuits of Redundant SUT-RNS," *IEEE Transactions On Instrumentation And Measurment,* vol. 58, no. 9,pp. 2959-2968, Sep. 2009.

[8] Timarchi, S., and Fazlali, M., "Power-Area-Delay Efficient Modulo 2^n-1 Multiplier for Multiply-Accumulate Unit", *The 15th CSI International Symposium on Computer Architecture and Digital Systems (CADS),* IPM, Tehran, 23-24 Sep., 2010.

[9] Iacono, D.L., Ronchi, M., "Binary Canonic Signed Digit Multiplier for High-speed Digital Signal Processing," *47th IEEE International Midwest Symposium on Circuits and Systems*, pp.205-208, 25-28 Jul. 2004.

[10] Rani R., Laxmi Kant Singh,Neelam Sharma, "FPGA Implementation of Fast Adders using Quaternary Signed Digit Number System," *International Conference on Emerging Trends in Electronic and Photonic Devices & Systems*, pp.132-135, 22-24 Dec. 2009.

[11] Jaberipur, G. and Parhami, B., "Posibits, Negabits, and Their Mixed Use in Efficient Realization of Arithmetic Algorithms," *the 15th CSI International Symposium on Computer Architecture and Digital Systems (CADS)*, pp. 3-9, Sep. 2010.

[12] Ohkubo, N., et al., "A 4.4 ns CMOS 54×54 multiplier using pass-transistor multiplexer," IEEE JSSC, vol. 30, no. 3, pp. 251-257, Mar. 1995.

[13] www.xilinx.com/products/silicon-devices/fpga/index.htm

[14] Wiśniewski, Remigiusz, Synthesis of compositional microprogram control units for programmable devices. Zielona Góra: University of Zielona Góra. pp. 153. ISBN 978-83-7481-293-1.

[15] edu/~vaughn/challenge/fpga_arch.html FPGA Architecture for the Challenge