PERGAMON

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers and Electrical Engineering

Computers and Electrical Engineering 29 (2003) 303–315

www.elsevier.com/locate/compeleceng

A signed binary addition circuit based on an alternative class of addition tables

Mitch Thornton [*]

Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, Mississippi 39762-9571, USA

Received 8 November 2000; accepted 13 March 2001

Abstract

Redundant signed digit number systems have been used as a basis for the construction of fast arithmetic circuits for several years. In particular, addition circuits with no carry-ripple effects have been developed using signed binary arithmetic systems. This paper presents a general class of signed binary addition tables and provides a framework for constructing various tables. The existence of an entire class of tables provides a circuit designer with an additional degree of freedom while developing addition circuitry. The choice of the exact form of the addition table can be based on the dominant desired characteristics of the resultant circuit. An example of a circuit derived for area minimization is presented and compared to another signed binary addition circuit that was previously published. Both circuits were optimized and mapped to 20 different CMOS cell libraries. The experimental results indicate an average decrease in area of 26% and an average decrease in dynamic power consumption of 29% with an average increase in delay of only 4.4%.
© 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Signed digit number systems; Redundant number systems; Signed binary addition; High speed addition circuitry

1. Introduction

In an attempt to build processors that are faster, significant past research has been focused on developing addition circuitry with minimal delay [6,10,13–15,17,18,21]. One technique for implementing fast adders is to utilize signed digit number systems where the digit set contains a number of members that is larger than the radix value [1,11,16]. When such number systems are

[*] Tel.: +1-662-325-3912; fax: +1-662-325-2298.
E-mail address: mitch@ece.msstate.edu (M. Thornton).

used, particular quantities may be represented by more than one unique digit string. This redundancy can be used to produce addition circuits that do not require carry ripples to traverse the entire set of digits comprising the addend and augend. Examples of circuits using redundant signed binary arithmetic may be found in Refs. [4,5,7–9,12,19].

In particular, the work described in Refs. [4,9,19,8] utilized the signed binary digit (SBD) set comprised of the digits $\{\overline{1}, 0, 1\}$. Two different addition tables were used in these approaches and they were chosen such that no carry ripples would occur. By examining the two digits in the augend and addend to the immediate right (one radix power less), intermediate sum and carry digits of the current addend and augend can be chosen such that no carry ripple will occur. As is shown in this paper, there are many other possible signed binary addition tables that exhibit this property. An entire class of new addition tables will be discussed and some specific examples are given.

The remainder of this paper is organized as follows: Section 2 will provide a brief review of the properties of signed binary number systems and will describe their use in past addition circuits as well as discuss why this number system can be used to generate fast addition circuits. In Section 3, a new class of addition tables is presented with a description of the theory of their construction included. Next, we describe experimental results where we computed several different addition tables and chose one that resulted in a circuit with superior area and delay characteristics as compared to the the circuit in Ref. [8]. Finally, the paper will conclude with a discussion of the results obtained and their usefulness.

2. Signed binary arithmetic

The number system commonly used in the design of digital logic is the binary system consisting of a digit set, $\{0, 1\}$ with a radix value of 2. Using this digit set and radix value, specific quantities can be represented as a fixed string of digits. Each digit string is a shorthand notation for a radix polynomial. For example, the value 9 in the familiar radix-10 system could be expressed as a radix polynomial in the binary system as given in Eq. (1).

$$9 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \tag{1}$$

In this radix-2 system, there exists a unique digit string for any given value since the number of members in the digit set is equal to the radix value. This statement holds for any arbitrary number system with this characteristic and is discussed in detail in Refs. [1,16].

A SBD system allows each digit to carry its own arithmetic sign information rather than assigning a single arithmetic sign to an entire quantity. Therefore, the signed binary digit set is $\{\overline{1}, 0, 1\}$, where $\overline{1}$ indicates a negative value with unity magnitude. Since the radix is 2 for a binary system, but the number of signed binary digits is 3, this system is redundant. Because the system is redundant, it is possible to represent some values with more than one digit string (or radix polynomial). As an example consider Eq. (2) which shows the equivalence of two radix polynomials both representing the value of 1.

$$1 = 1 \cdot 2^1 + \overline{1} \cdot 2^0 = 0 \cdot 2^1 + 1 \cdot 2^0 \tag{2}$$

In positional notation, the relationship expressed in Eq. (2) states that the number 01 is the same as the number $1\overline{1}$. This redundancy can be used to perform addition whereby the carry ripple is confined to two adjacent digits. This type of addition is much faster than ripple carry forms and does not suffer from extremely large fanout and area requirements due to routing that carry-lookahead forms can require.

2.1. Signed binary addition

Any arithmetic operation can be defined by a table containing all possible pairs of digits and a corresponding table entry defining the outcome of the operation. In terms of addition, such tables are referred to as addition tables and contain all possible combinations of the addend and augend with a corresponding sum digit. One of the first signed binary addition (SBA) tables was given in Ref. [9]. In this table, each sum was represented as an intermediate sum digit and an intermediate carry digit. For two cases, $1 + 0$ and $0 + \bar{1}$, two possible intermediate sum and carry digits are included, the selection of which is determined by the arithmetic signs of the next lower ordered augend and addend. These two digits are part of intermediate values that are combined with a guarantee of no ripple in the final stage. In this way, the final step of combining the intermediate carry and sum digits to form a final sum can be accomplished with a guarantee that no carry ripples will occur. Table 1 is a reproduction of the addition table that originally appeared in Ref. [9].

As an example, consider the addition of the following two signed digit numbers:

$$\begin{array}{rl}
10\bar{1}\bar{1}100 & \text{addend} \\
\underline{0\bar{1}\bar{1}010\bar{1}} & \text{augend} \\
110\bar{1}00\bar{1} & \text{intermediate sum} \\
\underline{0\bar{1}\bar{1}01000} & \text{intermediate carry} \\
000000\bar{1} & \text{final sum}
\end{array}$$

As is apparent from the example, the formation of each final sum digit in the ith position depends only upon the addend and augend digits in the ith and $(i - 1)$th positions. Using this method, a fast adder of arbitrary length can be built and is illustrated by the block diagram in Fig. 1.

Table 1 is not the only possible signed binary addition table that exhibits the behavior of limiting the carry ripple to one digit only. In the work in Ref. [5], a SBD addition table is

Table 1
Signed binary addition table used to prevent long carry propagation chains

Addend + augend digits in position i	Sign information of digits in position $i - 1$	Intermediate carry digit, c_{i+1}	Intermediate sum digit, s_i
$\bar{1} + \bar{1}$	Not used	$\bar{1}$	0
$\bar{1} + 0$	Either are negative	$\bar{1}$	1
$\bar{1} + 0$	Neither are negative	0	$\bar{1}$
$0 + 0$	Not used	0	0
$1 + \bar{1}$	Not used	0	0
$1 + 0$	Either are negative	0	1
$1 + 0$	Neither are negative	1	$\bar{1}$
$1 + 1$	Not used	1	0

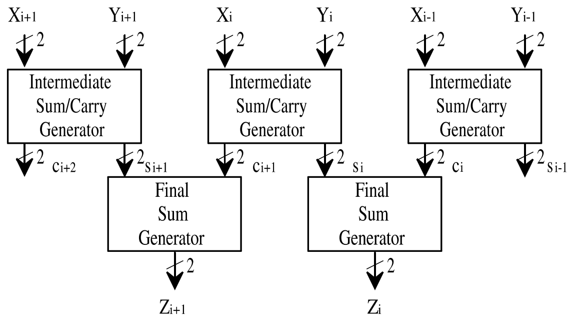

Fig. 1. Parallelism of signed binary adder.

described where all of the intermediate sum digits are restricted to the set $\{0, \overline{1}\}$ and all of the intermediate carry bits are restricted to the set $\{0, 1\}$. This ensures that during the formation of the final sum no carry will occur since the two possibilities $1 + 1$ and $\overline{1} + \overline{1}$ never exist. However, in order to always form an intermediate sum word with all non-zero values being negative valued and an intermediate carry word with all non-zero digits being positive valued, the use of a borrow digit is employed. Addition using this type of table requires two steps; first the borrow digits are formed and second, the intermediate carry and sum digits are chosen based upon the value of the borrow digit. Because the intermediate sum values are restricted to the digit set $\{\overline{1}, 0\}$ and the intermediate carry values are restricted to $\{0, 1\}$, these two intermediate words are guaranteed to be negative and positive valued respectively. Like the intermediate carry digit, the borrow digit in the *i*th place depends only upon the addend and augend digits in the $(i-1)$th position. Another advantage of this scheme is that single bits may be used internally to represent the borrow, intermediate sum, and intermediate carry bits since each of these represent only two possible SBD values. In the circuit described in Ref. [5], the intermediate carry digits were members of the set $\{0, 1\}$ while the intermediate sum and borrow digits were always values from the set $\{\overline{1}, 0\}$. The addition table for this type of adder is given in Table 2 followed by an example. As an example,

Table 2
Signed binary addition table using intermediate borrows and carries

Addend + augend digits in position i, x_i, y_i	Intermediate borrow out, b_{i+1}	Intermediate borrow in, b_i	Intermediate carry digit, c_{i+1}	Intermediate sum digit, s_i
$\overline{1}+\overline{1}$	$\overline{1}$	$\overline{1}$	0	$\overline{1}$
$\overline{1}+\overline{1}$	$\overline{1}$	0	0	0
$\overline{1}+0$	$\overline{1}$	$\overline{1}$	0	0
$\overline{1}+0$	$\overline{1}$	0	1	$\overline{1}$
$0+0$	0	$\overline{1}$	0	$\overline{1}$
$0+0$	0	0	0	0
$1+\overline{1}$	$\overline{1}$	$\overline{1}$	1	$\overline{1}$
$1+\overline{1}$	$\overline{1}$	0	1	0
$1+0$	0	$\overline{1}$	0	0
$1+0$	0	0	1	$\overline{1}$
$1+1$	0	$\overline{1}$	1	$\overline{1}$
$1+1$	0	0	1	0

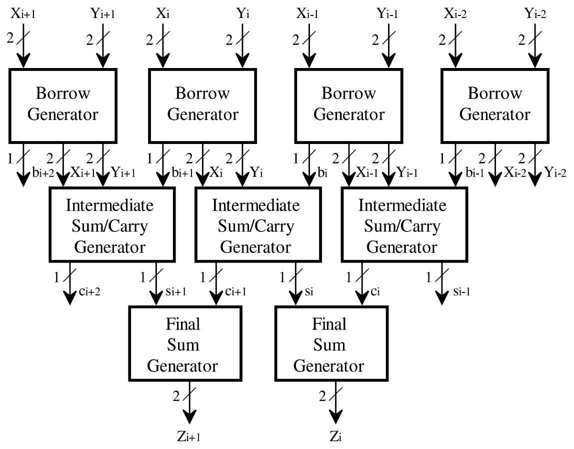

Fig. 2. Parallelism of signed binary adder using intermediate borrow and carry.

the addition rules given in Table 2 will be used to compute the sum of the same signed binary numbers as used in the previous example.

$10\overline{1}\,\overline{1}100$ addend
$\underline{0\overline{1}\,\overline{1}010\overline{1}}$ augend
$\underline{0\overline{1}\,\overline{1}\,\overline{1}00\overline{1}0}$ borrow
$00\overline{1}\,\overline{1}0\overline{1}\,\overline{1}$ intermediate sum
$\underline{00011010}$ intermediate carry
$0000000\overline{1}$ final sum

The main difference between these two methods are the use of a borrow to ensure that all intermediate sum digits are 0 or $\overline{1}$ and all intermediate carry digits are 0 or 1 in the latter case while the previous method simply chooses a form of representation for the intermediate carry and sum digits that will never cause a carry ripple. A comparison of the block diagram for the previous method as shown in Fig. 1 and that of this method as shown in Fig. 2 indicates that both techniques formulate the final sum digit based only upon the information contained in two consecutive addend and augend digits. However, the latter method requires a borrow generator to be included in the circuitry versus a digit selection portion as is used in the past method.

3. An alternative class of signed binary addition tables

The basic paradigm used in the method described in Ref. [5] is to ensure that all non-zero digits in the intermediate carry word have an opposite sign from those in the intermediate sum word. This was accomplished through the use of a borrow digit. However, many other possibilities exist for suitable addition tables. As an example, it is possible to choose the intermediate sum and borrow digits to be members of the set $\{0, 1\}$ and the intermediate carry digits to be members of the set $\{\overline{1}, 0\}$. It is also possible to allow the intermediate borrow digits to belong to the set $\{\overline{1}, 0, 1\}$ which is the entire set of SBDs.

In constructing alternative addition tables based upon this paradigm, it is useful to refer to an algebraic equation that describes the addend and augend digits as a function of the intermediate

Table 3
Notation used to represent various intermediate values for signed binary addition

Symbol	Description
x_i	SBD in the *i*th position of the addend word
y_i	SBD in the *i*th position of the augend word
b_i	SBD in the *i*th position of the borrow word
s_i	SBD in the *i*th position of the intermediate sum word
c_i	SBD in the *i*th position of the intermediate carry word
z_i	SBD in the *i*th position of the final sum word

values. Before this equation is given, the notation used for each of the values is defined in Table 3. Note that the subscripts for each symbol indicate the power of the radix when the value is written in radix polynomial form.

Using the notation in Table 3, it is possible to express the relationship between the addend and augend digits and the intermediate values as shown in Eq. (3).

$$x_i + y_i = 2(c_{i+1} + b_{i+1}) + s_i - b_i \tag{3}$$

It should be noted that the sum $x_i + y_i$ can contain values from the set $\{\overline{2}, \overline{1}, 0, 1, 2\}$ which is not the signed binary digit set. However, Eq. (3) is extremely useful for building alternative addition tables since it may be used to find values for b_i, b_{i+1}, s_i, and c_{i+1} from the set $\{\overline{1}, 0, 1\}$ that satisfy $x_i + y_i$ regardless of whether its value is one of the members of the SBD set.

Using the procedure outlined above, a program was written allowing for the formation of alternative addition tables. In the first example, an addition table is given that requires the borrow and intermediate sum digits to be restricted to the values of $\{0, 1\}$ and the intermediate carry to be restricted to $\{\overline{1}, 0\}$. The result is given in Table 4.

Since the only requirement for this class of addition tables is that the non-zero digits in the intermediate sum and carry words have dissimilar arithmetic signs, it is not necessary to restrict the borrow digits to a subset of the SBDs. The following two addition tables illustrated in Tables 5 and 6, are examples where the intermediate borrow digits are members of the set $\{\overline{1}, 0, 1\}$.

The three alternative addition tables presented here are new and are examples of tables from a larger class. The defining relationship is that all entries must satisfy Eq. (3) and that each non-zero digit in the intermediate sum and carry words have a dissimilar arithmetic sign. Based on these criteria it is possible to formulate many other addition tables. As an example, this class of addition tables was used to derive a signed binary addition circuit that always produces sum values with an even parity representation [20].

As an example the addition performed previously will be performed again using Table 6.

$10\overline{1}\,\overline{1}100$	addend
$\underline{0\overline{1}\,\overline{1}010\overline{1}}$	augend
$\underline{10\overline{1}01000}$	borrow
1000001	intermediate sum
$\underline{\overline{1}\,\overline{1}0000\overline{1}0}$	intermediate carry
$000000\overline{1}1$	final sum

Table 4
Alternative signed binary addition table using intermediate borrows and carries

Addend + augend digits in position i, x_i, y_i	Intermediate borrow out, b_{i+1}	Intermediate borrow in, b_i	Intermediate carry digit, c_{i+1}	Intermediate sum digit, s_i
$\bar{1}+\bar{1}$	0	0	$\bar{1}$	0
$\bar{1}+\bar{1}$	0	1	$\bar{1}$	1
$\bar{1}+0$	0	0	$\bar{1}$	1
$\bar{1}+0$	0	1	0	0
$0+0$	0	0	0	0
$0+0$	0	1	0	1
$1+\bar{1}$	1	0	$\bar{1}$	0
$1+\bar{1}$	1	1	$\bar{1}$	1
$1+0$	1	0	$\bar{1}$	1
$1+0$	1	1	0	0
$1+1$	1	0	0	0
$1+1$	1	1	0	1

Table 5
Signed binary addition table with no restrictions on the borrow digits

Addend + augend digits in position i, x_i, y_i	Intermediate borrow out, b_{i+1}	Intermediate borrow in, b_i	Intermediate carry digit, c_{i+1}	Intermediate sum digit, s_i
		$\bar{1}$	0	$\bar{1}$
$\bar{1}+\bar{1}$	$\bar{1}$	0	0	0
		1	1	$\bar{1}$
		$\bar{1}$	0	0
$\bar{1}+0$	$\bar{1}$	0	1	$\bar{1}$
		1	1	0
		$\bar{1}$	0	$\bar{1}$
$0+0$	0	0	0	0
		1	1	$\bar{1}$
		$\bar{1}$	0	$\bar{1}$
$1+\bar{1}$	0	0	0	0
		1	1	$\bar{1}$
		$\bar{1}$	0	0
$1+0$	0	0	1	$\bar{1}$
		1	1	0
		$\bar{1}$	0	$\bar{1}$
$1+1$	1	0	0	0
		1	1	$\bar{1}$

4. Experimental results

This section will discuss the experimental results of this work where we formed 192 different signed binary addition tables and synthesized each in an attempt to find adder circuitry with superior area and delay characteristics as compared to the circuit described in Ref. [8].

Table 6
Alternative signed binary addition table with no restrictions on the borrow digits

Addend + augend digits in position i, x_i, y_i	Intermediate borrow out, b_{i+1}	Intermediate borrow in, b_i	Intermediate carry digit, c_{i+1}	Intermediate sum digit, s_i
		$\overline{1}$	$\overline{1}$	1
$\overline{1}+\overline{1}$	$\overline{1}$	0	0	0
		1	0	1
		$\overline{1}$	$\overline{1}$	0
$\overline{1}+0$	0	0	$\overline{1}$	1
		1	0	0
		$\overline{1}$	$\overline{1}$	1
$0+0$	0	0	0	0
		1	0	1
		$\overline{1}$	$\overline{1}$	1
$1+\overline{1}$	0	0	0	0
		1	0	1
		$\overline{1}$	$\overline{1}$	0
$1+0$	1	0	$\overline{1}$	1
		1	0	0
		$\overline{1}$	$\overline{1}$	1
$1+1$	1	0	0	0
		1	0	1

To automate the process of developing the SBA tables, a program was written in C++ that created all combinations of SBA tables given the permissible digit sets for the *carry*, *borrow* and *intermediate sum* values. The program also varied the bit encodings for each of the values in $\{\overline{1},0,1\}$. After the program computed each table, it translated them to a .pla file which was then used as input to the SIS synthesis tool. The synthesized and mapped resultant circuit was then compared to the results of mapping the adder circuit described in Ref. [8] to the same cell library. In all, 192 different SBA tables were created and the resultant circuits were synthesized.

The result of this phase of the investigation produced several SBA tables that yielded circuits which had favorable characteristics when compared to the addition circuit in Ref. [8]. As an example, the SBA table given in Table 4 and the resulting synthesized circuit are shown in detail in order to provide specific data concerning the improvement in area, speed and power dissipation as compared to the circuit in Ref. [8]. The binary encodings used for this circuit for the input and output digits are $\overline{1} \leftarrow 00$, $0 \leftarrow 01$ and $1 \leftarrow 10$ (the 11 encoding was unused in this instance of the circuit). The allowable digit sets for the *carry* is $\{0,\overline{1}\}$, the *borrow* is $\{0,1\}$, and the *intermediate sum value* is $\{0,1\}$.

4.1. Addition circuit synthesis results

The chosen addition table and the circuit described in Ref. [8] are both described functionally in .eqn format in Fig. 3. The set of equations describing the rba circuit were obtained from Ref.

```
NAME=rba;

INORDER = xsi xai ysi yai pi1bar ci1bar;

OUTORDER = pibar cibar zsi zai;

pibar = !(xsi + ysi);

cibar = !(

        (!(xsi*ysi) * xai * yai)

        + ((xai ^ yai) * pi1bar)

        );

zsi = !(xai ^ yai ^ pi1bar) * ci1bar;

zai = !(

      (!( xai ^ yai ^ pi1bar) + ci1bar)

      *

      (!(!(xai^yai^pi1bar)*ci1bar))

      );
```

```
NAME = sba;

INORDER = ci bi ys ym xs xm;

OUTORDER = ci1 bi1 zs zm;

bi1 = !(ys + xs);

t1  = !(

        (!(ys + ym + xm))

      + (!(ym + xs + xm))

      );

t2  = !(ym * (!xm) * (!bi));

t3  = !((!bi) * xm * (!ym));

t4  = !((!xm) * (!ym) * bi);

t5  = !(bi * ym * xm);

ci1 = !(t1 * t2 * t3);

ui  = !(t2 * t3 * t4 * t5);

zm  = !(ci ^ ui);

zs  = !(ci + (!ui));
```

Fig. 3. EQN descriptions of the SBA circuit in Ref. [8] (left) and the NEW SBA (right).

[8] and, those for the new circuit sba resulted from our search program. The .eqn files were converted to .pla format and the ESPRESSO PLA minimization tool [2] was run on each in exact mode. The ESPRESSO results indicated that the circuit described in Ref. [8] reduced to 13 product terms requiring 44 different literals total. The circuit derived from the SBA table shown in Table 4 resulted in 15 product terms requiring a total of 56 different literals. Therefore, the two-level minimization results for the new circuit require two additional product terms. The .pla files resulting from invoking ESPRESSO with the exact flag are shown in Fig. 4.

Each of the .eqn descriptions were also used as initial input into the SIS logic synthesis tool [3], version 1.3. The standard optimization script, script.rugged was invoked and the circuits were technology mapped to 20 different standard cell libraries that are included in the SIS distribution. The synthesis results are shown in Table 7. Table 7 contains columns that give the name of the particular cell library used followed by the synthesis results in terms of number of area, power

```
.i 6
.o 4
.ilb xsi xai ysi yai pi1bar ci1bar
.ob pibar cibar zsi zai
.p 13
1111-- 0100
-1-110 0001
-0-010 0001
0-0--- 1000
-1-000 0001
-0-100 0001
-1-011 0011
-0-111 0011
-1-101 0011
-0-001 0011
---00- 0100
-0--0- 0100
-0-0-- 0100
.e
```

```
.i 6
.o 4
.ilb ci bi ys ym xs xm
.ob ci1 bi1 zs zm
.p 15
--00-0 1000
---000 1000
01-1-1 0010
11-1-1 0001
01-0-0 0010
00-1-0 1010
00-0-1 1010
11-0-0 0001
10-1-0 1001
10-0-1 1001
01-1-0 0001
01-0-1 0001
00-1-1 0001
--0-0- 0100
00-0-0 0001
.e
```

Fig. 4. PLA descriptions of the SBA circuit in Ref. [8] (left) and the NEW SBA (right).

dissipation and delay. The area measures are the total number of literals in the optimized expressions, the number of cells required from each library and a percentage of area reduction (in terms of the total cell sizes) of the new circuit versus the one in Ref. [8].

Table 7
Comparison of synthesis results of the SBA circuits

Cell library	SBA circuit in Ref. [8]			New SBA circuit			Comparison of circuits		
	Literals	Cells	Power	Literals	Cells	Power	% area reduction	% power reduction	% delay increase
22-1	54	31	1.49	38	22	1.05	29.4	29.5	8.9
22-2	46	18	1.72	35	15	1.24	22.0	27.9	−2.3
33-1	51	28	1.47	37	21	1.06	26.6	27.9	8.9
33-2	41	18	1.75	33	14	1.25	25.4	28.6	−2.3
33-4	44	15	1.71	30	12	1.24	24.5	27.5	13.2
43-5	39	14	1.68	30	12	1.24	21.6	26.2	13.2
44-1	49	26	1.44	37	21	1.06	22.7	26.4	22.0
44-2	39	16	1.70	33	14	1.25	20.0	26.5	13.2
44-3	39	14	1.68	30	12	1.24	21.6	26.2	13.2
44-6	39	14	1.68	30	12	1.24	21.6	26.2	13.2
asynch	40	16	1.29	26	10	0.95	28.1	26.4	0.0
example	41	18	1.51	32	12	1.10	32.7	27.2	2.8
lib2	46	17	1.48	35	12	1.11	29.8	25.0	4.4
mcnc-subset	47	15	1.53	37	13	1.10	25.0	28.1	9.9
mcnc	46	14	1.52	37	13	1.10	25.0	27.6	10.0
minimal	53	30	1.48	37	21	1.04	30.2	29.7	7.9
msu	46	14	1.52	26	10	0.95	25.0	37.5	−21.6
nand-nor	53	30	1.50	37	21	1.06	30.2	29.3	14.0
stdcell2_2	41	16	1.48	26	10	0.95	28.1	35.8	−19.4
synch	45	16	1.59	26	10	0.95	30.4	40.3	−21.6

The power dissipation column contains an estimate of the dynamic power (in mW) assuming uncorrelated, equiprobable switching probabilities for the primary inputs and a source voltage value, V_{dd}, of 3.3 V with a throughput rate of 500 MHz. Also included is a column containing the relative reduction in power dissipation when the new circuit is compared against the previous implementation using the power estimates.

In terms of delay comparisons, the slack parameters were compared where slack is the difference between the required delay and the estimated delay from the circuit. The slack values were computed by SIS using a simple zero-delay model for input arrival times. A relative increase in delay of the newer circuit versus the previous one is given based on the most-negative slack values.

In all cases, the required area and estimated power dissipation was better for the new SBA circuit. In 14 of the 20 synthesis runs, the delay estimates produced using a zero-delay static slack model were worse for the new SBA circuit, one case produced the same slack and five cases were better. Over all of the 20 cell libraries used, the comparison of the two circuits yielded an average reduction in area of 26%, an average reduction in power dissipation of 29% with an average increase in delay of 4.4%.

5. Conclusion

A generalized method for producing addition tables using a redundant signed binary number system has been described. A systematic method for generating the tables through the use of an algebraic equation has been given and several example tables were derived using this technique. The obvious advantage of high speed addition using these tables can be augmented with other advantages allowing for more compact designs due to the added flexibility in the choice of the actual digits produced by the circuitry [20]. An example of a compact circuit with desirable area and power dissipation characteristics was shown as an example and compared to a previously published signed binary addition circuit. These results indicated that average area and power dissipation were reduced by 26% and 29% respectively with a penalty of increasing the overall relative delay of only 4.4%.

References

[1] Avižienis A. Signed-digit number representations for fast parallel arithmetic. IRE Trans Electron Comput 1961;EC-10:389–400.

[2] Brayton RK, Hachtel GD, McMullen CT, Sangiovanni-Vincintelli AL. Logic minimization algorithms for VLSI synthesis. Boston, Massachusetts: Kluwer Academic; 1984.

[3] Brayton RK, Rudell R, Sangiovanni-Vincintelli AL, Wang AR. MIS: a multiple-level logic optimization system. IEEE Trans CAD 1987;CAD-6(6):1062–81.

[4] Briggs WS, Matula DW. A 17×69 bit multiply and add unit with redundant binary feedback and single cycle latency. Proc 11th Symp Comput Arithmetic 1993:163–70.

[5] Briggs WS, Matula DW. Signed digit multiplier. US Patent no. 5,144,576, September 1, 1992.

[6] Chan PK, Schlag MDF, Thomborson CD, Oklobdzija VG. Delay optimization of carry-skip adders and block carry-lookahead adders. Proc 10th Symp Comput Arithmetic, 1991. p. 154–64.

[7] Darley HM. Signed digit adder circuit. US Patent no. 4,979,140, December 18, 1990.

[8] Edamatsu H, Taniguchi T, Nishiyama T, Kuninobu S. A 33 MFLOPS floating point processor using redundant binary representation. Proc IEEE Int Conf Solid-State Circ, 1998. p. 152–3 and 342–3.

[9] Harata Y, Nakamura Y, Nagese H, Takigawa M, Takagi N. A high-speed multiplier using a redundant binary adder tree. IEEE J Solid-State Circ 1987;SC-22:28–34.

[10] Kantabutra V. Designing optimum carry-skip adders. Proc 10th Symp Comput Arithmetic 1991:146–53.

[11] Koren I. Computer arithmetic algorithms. Englewood Cliffs, New Jersey: Prentice-Hall; 1993.

[12] Kuninonu S, Nishyama T, Edamatsu T, Taniguchi T, Takagi N. Design of high speed MOS multiplier and divider using redundant binary representation. IEEE Trans Comput 1987;C-36:80–5.

[13] Lehman M, Burla N. Skip techniques for high-speed carry propagation in binary arithmetic circuits. IRE Trans Electron Comput 1962;EC-10:691–8.

[14] Ling H. High speed binary adder. IBM J Res Develop 1981;25:156–66.

[15] Ngai TF, Irwin MJ, Rawat S. Regular, area-time efficient carry-lookahead adders. J Parallel Distrib Comput 1986;3:92–105.

[16] Parhami B. Generalized signed-digit number systems: a unifying framework for redundant number representations. IEEE Trans Comput 1990;C-39:89–98.

[17] Skalansky J. Conditional-sum addition logic. IRE Trans Electron Comput 1960;EC-9:226–31.

[18] Swartzlander Jr EE (editor). Computer arithmetic, vol. I. Los Alamitos, California: IEEE Computer Society Press; 1990.

[19] Takagi N, Yasura H, Yajima S. High-speed VLSI multiplication algorithm with a redundant binary adder tree. IEEE Trans Comput 1985;C-34:789–96.

[20] Thornton MA. Signed binary addition circuitry with inherent even parity outputs. IEEE Trans Comput 1997;46(7):811–6.

[21] Waser S, Flynn MJ. Introduction to arithmetic for digital systems designers. New York: Holt, Rinehart and Winston; 1982.

Mitch Thornton received the Ph.D. in computer engineering from Southern Methodist University in 1995. Currently he is an Associate Professor of electrical and computer engineering at Mississippi State University. His research interests include computer architecture and arithmetic and electronic design automation algorithms.