Energy efficient modified partial Product generator for redundant binary multipliers

S.Geethamani
APPLIED ELCETRONICS
SHREE VENKATESHWARA HI-TECH ENGINEERING COLLEGE
ERODE, TAMILNADU, INDIA
sathkurube@gmail.com

Abstract: **Due to its high modularity and carry-free addition, a redundant binary (RB) representation can be used when designing high performance multipliers. The conventional RB multiplier requires an additional RB partial product (RBPP) row, because an error-correcting word (ECW) is generated by both the radix-4 Modified Booth encoding (MBE) and the RB encoding. This incurs in an additional RBPP accumulation stage for the MBE multiplier. In this paper, a new RB modified partial product generator (RBMPPG) is proposed; it removes the extra ECW and hence, it saves one RBPP accumulation stage. Therefore, the proposed RBMPPG generates fewer partial product rows than a conventional RB MBE multiplier. Simulation results show that the proposed RBMPPG based designs significantly improve the area and power consumption when the word length of each operand in the multiplier is at least 32 bits; these reductions over previous NB multiplier designs incur in a modest delay increase (approximately 5%). The power-delay product can be reduced by up to 59% using the proposed RB multipliers when compared with existing RB multipliers.**

1 Introduction

Digital multipliers are widely used in arithmetic units of microprocessors, multimedia and digital signal processors. Many algorithms and architectures have been proposed to design high-speed and low-power multipliers [1-13]. A normal binary (NB) multipli-cation by digital circuits includes three steps. In the first step, partial products are generated; in the second step, all partial products are added by a partial product re-duction tree until two partial product rows remain. In the third step, the two partial product rows are added by a fast carry propagation adder. Two methods have been used to perform the second step for the partial product reduction. A first method uses 4-2 compressors, while a second method uses redundant binary (RB) numbers [5-6]. Both methods allow the partial product reduction tree to be reduced at a rate of 2:1.

The redundant binary number representation has been introduced by Avizienis [1] to perform signed-digit arithmetic; the RB number has the capability to be represented in different ways. Fast multipliers can be designed using redundant binary addition trees [2-3]. The redundant binary representation has also been ap-plied to a floating-point processor and implemented in VLSI [4]. High performance RB multipliers have become popular due to the advantageous features, such as high modularity and carry-free addition [5-9].

A RB multiplier consists of a RB partial product (RBPP) generator, a RBPP reduction tree and a RB-NB converter. A Radix-4 Booth encoding or a modified Booth encoding (MBE) is usually used in the partial product generator of parallel multipliers to reduce the number of partial product rows by half [5-6] [10-13]. A RBPP row can be obtained from two adjacent NB partial product rows by inverting one of the pair rows [5-6]; an N-bit convention-al RB MBE (CRBBE-2) multiplier requires _ /4_ RBPP rows. An additional error-correcting word (ECW) is also required by both the RB and the Booth encoding [5-6] [14]; therefore, the number of RBPP accumulation stages (NRBPPAS) required by a power-of-two word-length (i.e., 2 -bit) multiplier is given by:

NRBPPAS = _log (/4 + 1)= n − 1, if _ = 2 (1)

If the additional ECW can be removed, an RBPP accu-mulation stage is saved, so resulting in improvements of complexity and critical path delay for a RB multiplier. For example, a conventional 32-bit RB multiplier has 4 RBPP accumulation stages; if the ECW is removed, then the number of RBPP accumulation stages is reduced to 3, i.e., the stage count is decreased by 25%. Note that the problem of extra ECW does not exist in standard signifi-cand size (i.e., 24×24-bit and 54×54-bit) RB multipliers as used in floating point-arithmetic units [5-6].

Alternatively, a high-radix Booth encoding technique can reduce the number of partial products. However, the number of expensive hard multiples (i.e., a multiple that is not a power of two and the operation cannot be per-formed by simple shifting and/or complementation) in-creases too [14-16]. Besli et al. [16] noticed that some hard multiples can be obtained by the differences of two sim-ple power-of-two multiplies. A new radix-16 Booth en-coding (RBBE-4) technique without ECW has been pro-posed in [14]; it avoids the issue of hard multiples. A radix-16 RB Booth encoder can be used to overcome the hard multiple problem and avoid the extra ECW, but at the cost of doubling the number of RBPP rows. There-fore, the number of radix-16 RBPP rows is the same as in the radix-4 MBE. However, the RBPP generator based on a radix-16 Booth encoding has a complex circuit struc-ture and a lower speed compared with the MBE partial product generator [10] when requiring the same number of partial products.

This paper focuses on the RBPP generator for design
[8-10] to speed up the partial product reduction tree and decrease power dissipation. Optimized designs of 4-2 exact
compressors have been proposed in [8, 11 - 16]. [17] [18] have also considered compression for approximate multiplication. In [17], an approximate signed multiplier has been proposed for use in arithmetic data value speculation (AVDS); multiplication is performed using the Baugh-Wooley algorithm. However, no new design is proposed for the compressors for the inexact computation. Designs of approximate compressors have been proposed in [18]; however, these designs do not target multiplication. It should be noted that the approach of [7] improves over [17] [18] by utilizing a simplified multiplier block that is amenable to approximate multiplication.

Initially in this paper, two novel approximate 4-2 compressors are proposed and analyzed. It is shown that these simplified compressors have better delay and power consumption than the optimized (exact) 4-2 compressor designs found in the technical literature [8]. These approximate compressors are then used in the restoration module of a Dadda multiplier; four different schemes are proposed for inexact multiplication. Extensive simulation results are provided at circuit-level for figures of merit, such as delay, transistor count, power dissipation, error rate and normalized error distance under CMOS feature sizes of 32, 22 and 16 nm. The application of these multipliers to image processing is then presented. The results of two examples of multiplication of two images are reported; these results show that the third and fourth approximate multipliers yield an output product image that has a very high quality and resemblance to the image generated by an exact multiplier, i.e. excellent values for the average NED and the Peak Signal-to-Noise Ratio (PSNR) are found (for the PSNR more than 50db). The analysis and simulation results show that the proposed approximate designs for both the compressor and the multiplier are viable candidates for inexact computing.

This paper is organized as follows. Section 2 is a review of existing schemes for (exact) compressors. The two new designs of an approximate 4-2 compressor are presented in Section 3.Multiplication and four different approximate multipliers are proposed in Section 4. Simulation results for the approximate compressors and multipliers are provided in Section 5. The application of the proposed approximate multipliers to image processing is presented in Section 6. Section 7 concludes the manuscript.

II. Exact Compressors

The main goal of either multi-operand carry-save addition or parallel multiplication is to reduce *n* numbers to two numbers; therefore, *n*-2 compressors (or *n*-2 counters) have been widely used in computer arithmetic. A*n*-2 compressor (Figure 1) is usually a slice of a circuit that reduces *n* numbers to two numbers when properly replicated. In slice *i* of the circuit, the *n*-2 compressor receives *n* bits in position *i* and one or more carry bits from the positions to the right, such as *i* – 1 or *i* – 2. It produces two output bits in positions *i* and *i* + 1 and one or more carry bits into the higher positions, such as *i* + 1 or i + 2.For the correct operation of the circuit shown in Figure 1, the following inequality must be satisfied

… 3 2 4 8 … (1)

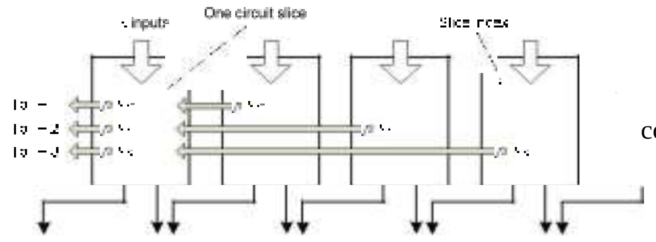

Figure 1.Schematic diagram of *n*-2 compressors in a multi operand addition circuit [13]

Where denotes the number of carry bits from slice *i*to slice *i*+ *j*.

A widely used structure for compression is the 4-2 compressor; a 4-2 compressor (Figure 2) can be implemented with a carry bit between adjacent slices ( ). The carry bit from the position to the right is denoted as c while the carry bit into the higher position is denoted as c_{out}. The two output bits in positions i and i + 1are also referred to as the *sum* and *carry* respectively.

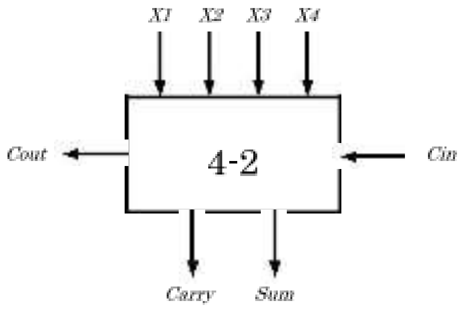

Figure2.4-2 compressor

The following equations give the outputs of the 4-2 compressor, while Table 1 shows its truth table.

(2)

1 2 3 4 (3)

1 2 3 1 2 1 (4)

1 2 3 4 1 2 3 4 4

The common implementation of a 4-2 compressor is accomplished by utilizing two full-adder (FA) cells (Figure 3) [8]. Different designs have been proposed in the literature for 4-2 compressor [8, 11-16].

Figure 4 shows the optimized design of an exact4-2 compressor based on the so-called XOR-XNOR gates [8]; a XOR-XNOR gate simultaneously generates the XOR and XNOR output signals. The design of [8] consists of three XOR-XNOR (denoted by XOR*) gates, one XOR and two 2-1 MUXes. The critical path of this design has a delay of 3 , where is the unitary delay through any gate in the design.

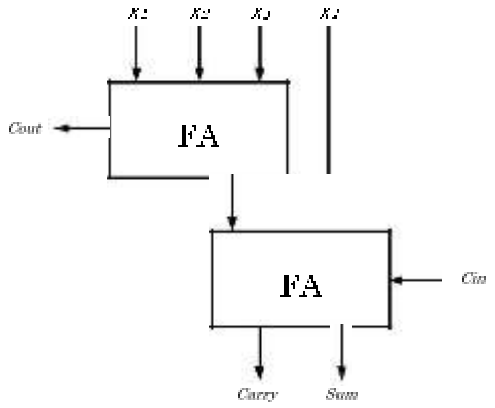

Figure 3. Implementation of 4-2 Compressor

TABLE I
TRUTH TABLE OF 4-2 COMPRESSOR

c_{in}	X_4	X_3	X_2	X_1	c_{out}	*carry*	*sum*
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	1	0	0
0	0	1	0	0	0	0	1
0	0	1	0	1	1	0	0
0	0	1	1	0	1	0	0
0	0	1	1	1	1	0	1
0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	1	0	1
0	1	1	0	0	0	1	0
0	1	1	0	1	1	0	1
0	1	1	1	0	1	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0
1	0	0	1	1	1	0	1
1	0	1	0	0	0	1	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	0	0	1	1
1	1	0	1	1	1	1	0
1	1	1	0	0	0	1	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

III. PROPOSED APPROXIMATE COMPRESSORS

In this section, two designs of an approximate compressor are proposed. Intuitively to design an approximate 4-2 compressor, it is possible to substitute the exact full-adder cells in Figure3 by an approximate full-adder cell (such as the first design proposed in [2]). However, this is not very efficient, because it produces at least 17 incorrect results out of 32 possible outputs, i.e. the error rate of this inexact compressor is more than 53% (where the *error rate* is given by the ratio of the number of erroneous outputs over the total number of outputs). Two different designs are proposed next to reduce the error rate; these designs offer significant performance improvement compared to an exact compressor with respect to delay, number of transistors and power consumption.

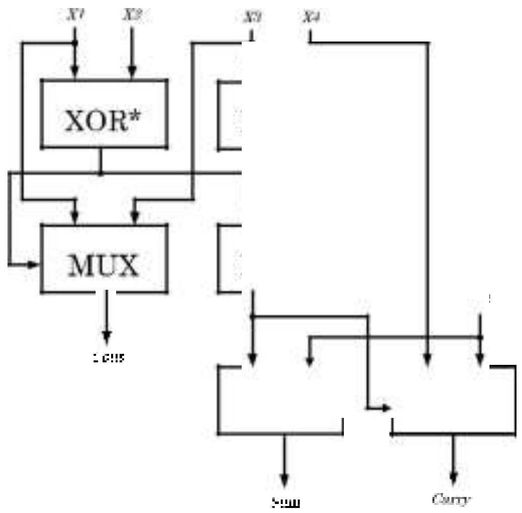

Figure4. Optimized 4-2 compressor of [8]

A. Design 1

As shown in Table I, the *carry* output in an exact compressor has the same value of the input c_{in} in 24 out of 32 states. Therefore, an approximate design must consider this feature. In Design 1, the *carry* is simplified to c_{in} by changing the value of the other 8 outputs.

(5)

Since the *Carry* output has the higher weight of a binary bit, an erroneous value of this signal will produce a difference value of two in the output. For example, if the input pattern is "01001" (row 10 of Table II), the correct output is "010" that is equal to 2. By simplifying the *carry* output to c_{in}, the approximate compressor will generate the "000" pattern at the output (i.e. a value of 0). This substantial difference may not be acceptable; however, it can be compensated or reduced by simplifying the c_{out} and *sum* signals. In particular, the simplification of *sum* to a value of 0 (second half of Table II) reduces the difference between the approximate and the exact outputs as well as the complexity of its design. Also, the presence of some errors in the *sum* signal will results in a reductions of the delay of producing the approximate *sum* and the overall delay of the design (because it is on the critical path).

1 2 3 4 (6)

In the last step, the change of the value of c_{out} in some states, may reduce the error distance provided by approximate *carry* and *sum* and also more simplification in the proposed design.

1 2 3 4 (7)

Although the above mentioned simplifications of *carry* and *sum* increase the error rate in the proposed approximate compressor, its design complexity and therefore the power consumption are considerably decreased. This can be realized by comparing (2)-(4) and (5)-(7).Table II shows the truth table of the first proposed approximate compressor. It also shows the difference between the inexact output of the proposed approximate compressor and the output of the exact compressor. As shown in Table II, the proposed design has 12 incorrect outputs out of 32 outputs (thus yielding an error rate of 37.5%). This is less than the error rate using the best approximate full-adder cell of [2].

TABLE II
TRUTH TABLE OF THE FIRSTAPPROXIMATE 4-2 COMPRESSOR

c_{in}	X_4	X_3	X_2	X_1	C_{out}'	*carry'*	*sum'*	*Difference*
0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	1	0
0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	1	-1
0	0	1	0	0	0	0	1	0
0	0	1	0	1	1	0	0	0
0	0	1	1	0	1	0	0	0
0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	0	1	1	0	0	0
0	1	0	1	0	1	0	0	0
0	1	0	1	1	1	0	1	0
0	1	1	0	0	0	0	1	-1
0	1	1	0	1	1	0	1	0
0	1	1	1	0	1	0	1	0
0	1	1	1	1	1	0	1	-1
1	0	0	0	0	0	1	0	1
1	0	0	0	1	0	1	0	0
1	0	0	1	0	0	1	0	0
1	0	0	1	1	0	1	0	-1
1	0	1	0	0	0	1	0	0
1	0	1	0	1	1	1	0	1
1	0	1	1	0	1	1	0	1
1	0	1	1	1	1	1	0	0
1	1	0	0	0	0	1	0	0
1	1	0	0	1	1	1	0	1
1	1	0	1	0	1	1	0	1
1	1	0	1	1	1	1	0	0
1	1	1	0	0	0	1	0	-1
1	1	1	0	1	1	1	0	0
1	1	1	1	0	1	1	0	0
1	1	1	1	1	1	1	0	-1

(5)-(7) are the logic expressions for the outputs of the first design of the approximate 4-2 compressor proposed in this manuscript.

The gate level structure of the first proposed design (Figure 6) shows that the critical path of this compressor has still a delay of 3 , so it is the same as for the exact compressor of Figure 5. However, the propagation delay through the gates of this design is lower than the one for the exact compressor. For example, the propagation delay in the *XOR** gate that generates both the *XOR* and *XNOR* signals in [8], is higher than the delay through a *XNOR* gate of the proposed design. Therefore, the critical path delay in the proposed design is lower than in the exact design and moreover, the total number of gates in the proposed design is significantly less than that in the optimized exact compressor of [8].

B. Design 2

A second design of an approximate compressor is proposed to further increase performance as well as reducing the error rate. Since the *carry* and c_{out} outputs have the same weight, the proposed equations for the approximate *carry* and c_{out} in the previous part can be interchanged. In this new design, *carry* uses the right hand side of (7) and c_{out} is always equal to c_{in}; since c_{in} is zero in the first stage, c_{out} and c_{in} will be zero in all stages. So, c_{in} and c_{out} can be ignored in the hardware design. Figure 7shows the block diagram of this approximate 4-2 compressor and the expressions below describe its outputs.

1 2 3 4 (9)(8)

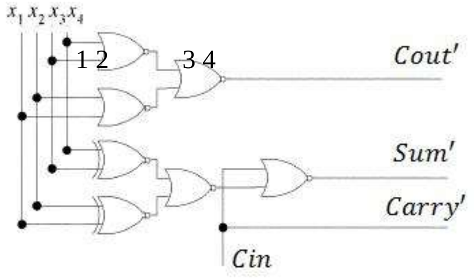

Figure 6. Gate level implementation of Design 1

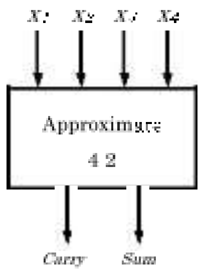

Figure7. Approximate 4-2 compressor, Design 2

Note that (9) is the same as (7) and (8) is the same as (6) for c_{in}= 0. Figure 8 shows the gate level implementation of the second proposed design. The delay of the critical path of this approximate design is 2 , so it is 1 less than the previous designs; moreover, a further reduction in the number of gates is accomplished.

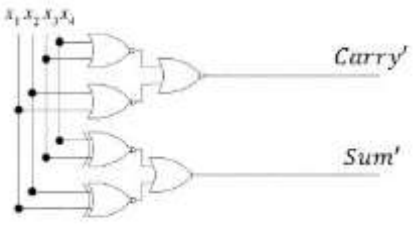

Figure 8. Gate level implementation of Design 2

Table III shows the truth table of the second approximate design for a 4-2 compressor; this Table also shows the difference between the exact decimal value of the addition of the inputs and the decimal value of the outputs produced by the approximate compressor. For example when all inputs are

1, the decimal value of the addition of the inputs is 4. However, the approximate compressor produces a 1 for the *carry* and *sum*. The decimal value of the outputs in this case is 3; Table II shows that the difference is -1.

TABLE III
TRUTH TABLE OF SECOND PROPOSED 4-2 COMPRESSOR

X_4	X_3	X_2	X_1	*carry'*	*sum'*	*difference*
0	0	0	0	0	1	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	-1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	1	-1
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

This design has therefore 4 incorrect outputs out of 16 outputs, so its error rate is now reduced to 25%. This is a very positive feature, because it shows that on a probabilistic basis, the imprecision of the proposed design is smaller than the other available schemes.

IV. MULTIPLICATION

In this section, the impact of using the proposed compressors for multiplication is investigated. A fast (exact) multiplier is usually composed of three parts (or *modules*) [8].

- Partial product generation.
- A Carry Save Adder (CSA) tree to reduce the partial products' matrix to an addition of only two operands
- A Carry Propagation Adder (CPA) for the final computation of the binary result.

In the design of a multiplier, the second module plays a pivotal role in terms of delay, power consumption and circuit complexity. Compressors have been widely used [9, 10] to speed up the CSA tree and decrease its power dissipation, so to achieve fast and low-power operation. The use of approximate compressors in the CSA tree of a multiplier results in an approximate multiplier.

A 8×8 unsigned Dadda tree multiplier is considered to assess the impact of using the proposed compressors in approximate multipliers. The proposed multiplier uses in the first part AND gates to generate all partial products. In the second part, the approximate compressors proposed in the previous section are utilized in the CSA tree to reduce the partial products. The last part is an exact CPA to compute the final binary result. Figure 9(a) shows the reduction circuitry of an exact multiplier for n=8. In this figure, the reduction part uses half-adders, full-adders and 4-2 compressors; each partial product bit is represented by a dot. In the first stage, 2 half-adders, 2 full-adders and 8 compressors are utilized to reduce the partial products into at most four rows. In the second or final stage, 1 half-adder, 1 full-adder and 10 compressors are used to compute the two final rows of partial products. Therefore, two stages of reduction and 3 half-adders, 3 full-adders and 18 compressors are needed in the reduction circuitry of an 8×8Dadda multiplier.

In this paper, four cases are considered for designing an approximate multiplier.

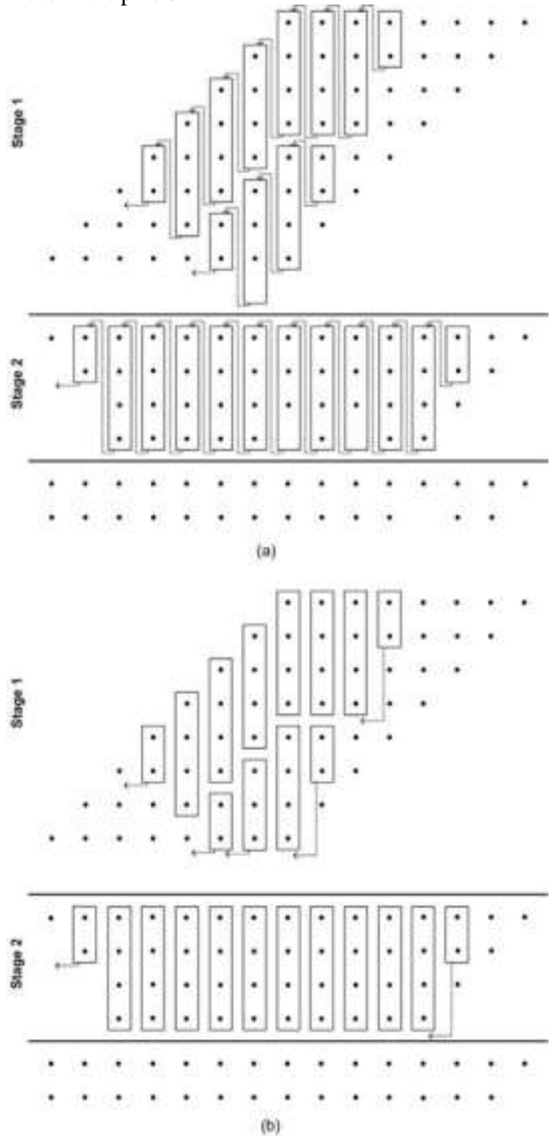

Figure 9. Reduction circuitry of an 8×8Dadda multiplier, (a) using Design 1 compressors, (b) using Design 2 compressors

- In the first case (Multiplier 1), Design 1 is used for all 4-2 compressors in Figure 9(a).
- In the second case (Multiplier 2), Design 2 is used for the 4-2 compressors. Since Design 2 does not have c_{in} and c_{out}, the reduction circuitry of this multiplier requires a lower number of compressors (Figure 9(b)). Multiplier 2 uses 6 half-adders, 1 full-adder and 17 compressors.
- In the third case (Multiplier 3), Design 1 is used for the compressors in the$n$-1 least significant columns. The other n most significant columns in the reduction circuitry use exact 4-2 compressors.

- In the fourth case (Multiplier 4), Design 2 and exact 4-2 compressors are used in the*n-1* least significant columns and the*n* most significant columns in the reduction circuitry respectively.

The objectives of the first two approximate designs are to reduce the delay and power consumption compared with an exact multiplier; however, a high error distance is expected. The next two approximate multipliers (i.e. Multipliers 3 and 4) are proposed to decrease the error distance. The delay in these designs is determined by the exact compressors that are in the critical path; therefore, there is no improvement in delay for these approximate designs compared with an exact multiplier. However, it is expected that the utilization of approximate compressors in the least significant columns will decrease the power consumption and transistor count (as measure of circuit complexity). While the first two proposed multipliers have better performance in terms of delay and power consumption, the error distances in the third and fourth designs are expected to be significantly lower.

V. SIMULATION RESULTS

In this section, he designs of the two approximate compressors (Section III) and the four approximate multipliers (Section IV) are simulated using HSPICE. Predictive Technology Models (PTMs) at different CMOS feature sizes (32 nm, 22 nm and 16 nm) are utilized in the HSPICE simulation.

A. Approximate Compressors

The two approximate compressors of this paper and the best low-power exact compressor of [8] (implemented by using *XOR-XNOR* gates) are simulated at a 1 GHz frequency; a fan-out of 4 is utilized in all simulations. The simulation results of the delay, power consumption and power-delay product (PDP) are given in Table IV by using the PTMs at 32 nm, 22 nm and 16 nm.

TABLE IV
SIMULATION RESULTS (@32 NM)

Design	*Delay(ps)*	*Power(μW)*	*PDP(aJ)*
	@32 nm		
Exact Design [8]	60.36	2.98	180
Design 1	58.32	1.27	74
Design 2	44.35	1.14	50
	@22 nm		
Exact Design [8]	55.82	1.50	84
Design 1	56.79	0.62	35
Design 2	41.69	0.58	24
	@16 nm		
Exact Design [8]	47.59	0.95	45
Design 1	37.16	0.39	14
Design 2	24.44	0.36	9

As expected, the second proposed design (Design 2) has the best delay, power consumption and PDP; these improvements are irrespective of feature size. This approximate design is 62% faster than the exact compressor at 16 nm CMOS technology and 44% faster on average for the three feature sizes considered. Moreover on average, Design 2 is also 35% faster than Design 1. The two proposed approximate designs achieve significant improvement in terms of power consumption; on average at different feature sizes, the power consumption of Design 1 is 57% less than the exact compressor, while Design 2 has a power consumption that is 60% less than the exact design of [8].

Table V compares these designs in terms of number of transistors, as a measure of circuit complexity. The exact compressor [8] uses 10 transistors to implement each *XOR** gate, 6 transistors to implement the *XOR* gate and 8 transistors to implement each *MUX* gate [8]; therefore, the exact compressor utilizes 52 transistors. A 50% improvement in circuit complexity is accomplished by Design 2, as reflected by the lower number of transistors. This is expected because the second approximate design has no c_{in} and c_{out} with only 4 inputs and 2 outputs (the exact compressor has 5 inputs and 3 outputs).

TABLE V
COMPARISON OF NUMBER OF TRANSISTORS

Design	*Number of transistors*
Exact Design [8]	52
Design 1	28
Design 2	26

B. Approximate Multipliers

The four proposed approximate multipliers are simulated for *n=8*. The delay, power consumption and number of transistors are investigated for these approximate designs as well as the exact multiplier. A comparison of the error distance (as measure of reliability [1]) of the proposed multipliers with other approximate multipliers is also pursued.

- *Delay*

The delay of the reduction circuitry (second module) of a Dadda multiplier is dependent on the number of reduction stages and the delay of each stage. In Multipliers 1 and 2, the approximate compressors are used in all columns; therefore, the delay of the stages is equal to the delay of the approximate compressors. However, in Multipliers 3 and 4, the delay of the stages is equal to the delay of the exact compressors. So, the use of these approximate compressors in the *n*/2 LSBs cause no improvement in terms of delay compared to an exact multiplier. The delay improvement in the reduction circuitry of each multiplier (at 32 nm CMOS technology) compared to an exact adder is shown in Table VI.

TABLE VI
DELAY IMPROVEMENT IN REDUCTION CIRCUITRY

Design	*Improvement (%)*
Multiplier 1	3.38
Multiplier 2	26.52
Multiplier 3	0
Multiplier 4	0

- *Power Consumption*

The power consumption of each multiplier is determined by the number and type of compressors used. Multipliers 1 and 2 use only approximate compressors so they have power consumption lower than Multipliers 3 and 4. Table VII shows the power consumption improvement of each multiplier at 32 nm feature size with respect to an exact adder; this confirms that an approximate multiplier in the reduction circuitry will result in a considerable power saving.

TABLE VII
POWER CONSUMPTION IMPROVEMENT IN REDUCTION CIRCUITRY

Design	*Improvement (%)*
Multiplier 1	52.49
Multiplier 2	58.58
Multiplier 3	17.50
Multiplier 4	26.15

• *Transistor Count*

The transistor count is used in this paper as metric of circuit complexity. The first two approximate multipliers have a lower transistor count compared with Multipliers 3 and 4. Table VIII shows the transistor count improvement of the reduction circuitry of each multiplier compared to an exact adder.

TABLE VIII
TRANSISTOR COUNT IMPROVEMENT IN REDUCTION CIRCUITRY

Design	*Improvement (%)*
Multiplier 1	42.11
Multiplier 2	48.15
Multiplier 3	14.03
Multiplier 4	22.42

• *Error Distance*

Four additional approximate multipliers are simulated to compare the error distance. The multiplier (Multiplier 5) proposed in [7] is simulated for *n=8.* The truncated multiplier with constant correction [5] (Multiplier 6) and the truncated multiplier with variable correction [6] (Multiplier 7) are also simulated for *n=8* and *k=1*. A further approximate multiplier (Multiplier 8) is simulated to investigate the impact of using the proposed approximate compressors compared with other approximate compressors. This *8×8* Dadda multiplier uses 4-2 compressors made of two approximate full-adders (Figure 3). The first full-adder design proposed in [2] is used in this approximate multiplier. Table IX summarizes the eight approximate multipliers assessed in this manuscript, i.e. the four proposed designs and the other four approximate multipliers together with their salient features.

TABLE IX
APPROXIMATE MULTIPLIERS AND THEIR FEATURES

Design	*Feature*
Multiplier 1	Design 1 in all columns
Multiplier 2	Design 2 in all columns
Multiplier 3	Design 1 in LSBs and exact compressor in MSBs
Multiplier 4	Design 2 in LSBs and exact compressor in MSBs
Multiplier 5 [7]	Approximate 2x2 multiplier blocks
Multiplier 6 [5]	Truncated multiplier with constant correction
Multiplier 7 [6]	Truncated multiplier with variable correction
Multiplier 8	Compressors made of approximate FAs [2]

The normalized error distance (NED) is used to compare these approximate multipliers. In [1], the NED is defined as the average error distance over all inputs, normalized by the maximum possible error. In this paper the NED is defined for each input. Therefore the average NED is equivalent to the NED defined in [1]. The maximum high (low) NED is also defined as the largest absolute value of NED for the case in which the erroneous result is more (less) than the exact result. Table X shows the average NED, the maximum high and low NEDs and the number of correct results (or outputs) of approximate multipliers for *n=8*. The number of correct outputs out of the total outputs represents the probability of correctness for each design. Based on Table X, the probability of correctness in Multiplier 1 is 0.16% (103 out of 65025) while the probability of correctness in Multiplier 4 is 14.3% (9320 out of 65025). Since the proposed approximate compressors produce erroneous results for all-zero input patterns (row 1 in Tables II and III), the proposed approximate multipliers will generate an erroneous result if at least one of the inputs is zero. However, in these cases (511 cases for *n*=8) the multiplier can produce correct result by adding a circuit for detecting the zero-valued inputs. Therefore, the zero-valued input patterns are not considered further in the simulation to investigate the proposed multipliers for a fair comparison.

TABLEX
NED FOR N = 8

Design	*Average NED*	*Max High NED*	*Max Low NED*	*correct outputs (out of 65025)*
Multiplier 1	0.6065×10^{-1}	0.1593	0.1375	103
Multiplier 2	0.5352×10^{-1}	0.1278	0.1329	458
Multiplier 3	0.9199×10^{-3}	0.3199×10^{-2}	0.2707×10^{-2}	5888
Multiplier 4	0.7827×10^{-3}	0.1845×10^{-2}	0.3076×10^{-2}	9320
Multiplier 5 [7]	0.1400×10^{-1}	0	0.2222	34400
Multiplier 6 [5]	0.1609×10^{-2}	0.3937×10^{-2}	0.9858×10^{-2}	0
Multiplier 7 [6]	0.1146×10^{-2}	0.3060×10^{-2}	0.4045×10^{-2}	769
Multiplier 8	0.1049	0.2263	0.1207	8

Based on Table X, Multiplier 4 has the lowest average NED among all approximate multipliers. The average NED of Multiplier 4 is 18 times better than Multiplier 5, 2 times better than Multiplier 6 and 1.5 times better than Multiplier 7. Multiplier 5 has the highest number of correct outputs. It has also the lowest maximum high NED. As the approximate output is always less than the exact output, the maximum high NED is 0 for this design; however, it has the worst maximum low NED among all considered designs.

A plot of the NED distribution is also generated (Figure 10) to compare the performance of the approximate multipliers. The range of the product in a 8×8 multiplier is between 0 and 65025 (unsigned values). All possible outputs are categorized in 127 intervals; in the first interval the output is between 0 and 512, in the second interval the output is between 513 and 1024 and so on. In the last interval the output is between 64513 and 65025. The average NED of each interval is then computed for the approximate multiplier. Figures 10a and 10b show that for Multipliers 1 and 2, the average NED increases only at very large or very small product values, i.e. these approximate multiplier incur on average in a small error in output compared to the exact calculation.

VI. APPLICATION: IMAGE PROCESSING

In this section the application of the proposed approximate multipliers to image processing is illustrated. A multiplier is used to multiply two images on a pixel by pixel basis, thus blending the two images into a single output image.

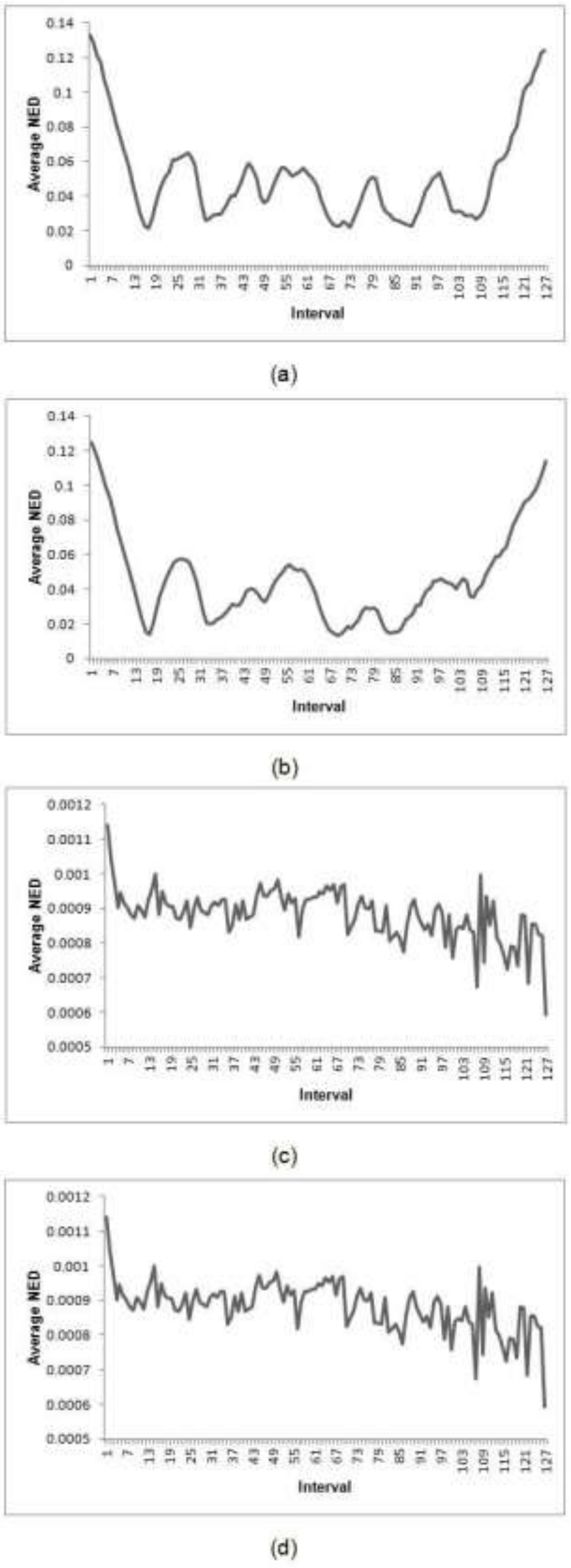

Figure10.Average NED distribution in 8×8 approximate multipliers. (a) Multiplier 1, (b) Multiplier 2, (c) Multiplier 3, (d) Multiplier 4

Figure 11 shows two examples: both input images and the resulting output image are provided. A program has been developed in C# .net and simulated in Microsoft Visual Studio 2010 using the 8 approximate multipliers at n=8. Figures 12 and 13 show the outputs for the two examples.

The average NED and the Peak Signal-to-Noise Ratio (PSNR) that is based on the Mean Squared Error (MSE) are computed to assess the quality of the output image and compare it with the output image generated by an exact multiplier. The equations for the MSE and PSNR are given in (10) and (11); in (10), m and p are the image dimensions and $I(i,j)$ and $K(i,j)$ are the exact and obtained values of each pixel respectively. In (11), MAX_I represents the maximum value of each pixel.

— (10)

MSE ∑ ∑—— , , (11)

PSNR 10 MSE

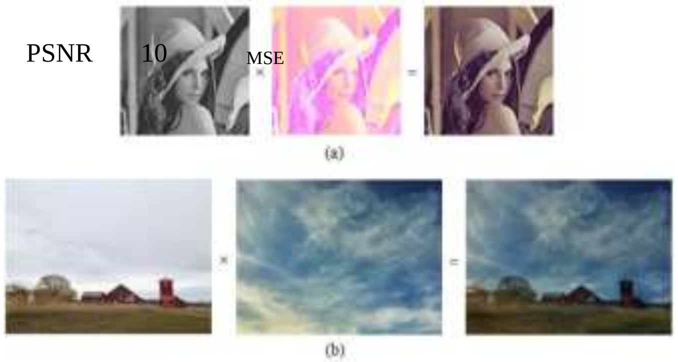

Figure11. Image multiplication (a) example 1, (b) example 2 (both using an exact multiplier)

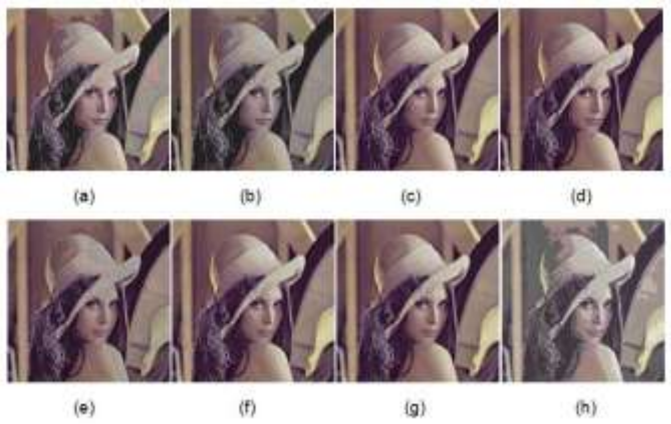

Figure12. Image multiplication results for example 1, (a) Multiplier 1, (b) Multiplier 2, (c) Multiplier 3, (d) Multiplier 4, (e) Multiplier 5, (f) Multiplier 6, (g) Multiplier 7, (h) Multiplier 8.

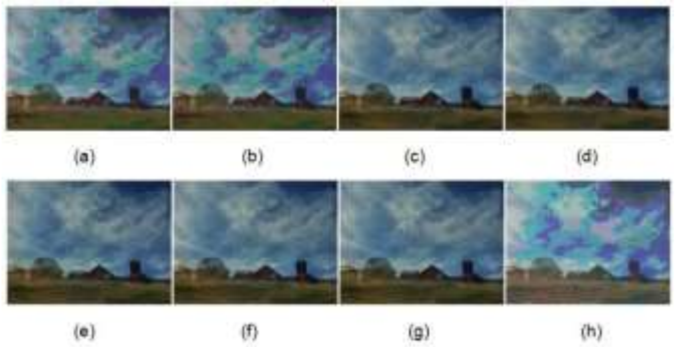

Figure13. Image multiplication results for example2, (a) Multiplier 1, (b) Multiplier 2, (c) Multiplier 3, (d) Multiplier 4, (e) Multiplier 5, (f) Multiplier 6, (g) Multiplier 7, (h) Multiplier 8

TABLE XI
PSNR AND AVERAGE NED FOR FIRST EXAMPLE

Design	*PSNR (dB)*	*Average NED($\times 10^{-2}$)*
Multiplier 1	25.3	4.4
Multiplier 2	26.3	3.7
Multiplier 3	53.9	0.10
Multiplier 4	53.2	0.12
Multiplier 5 [7]	26.3	2.3
Multiplier 6 [5]	48.3	0.28
Multiplier 7 [6]	52.3	0.15
Multiplier 8	21.2	7.6

TABLE XII
PSNR AND AVERAGE NED FOR SECOND EXAMPLE

Design	*PSNR (dB)*	*Average NED($\times 10^{-2}$)*
Multiplier 1	25.1	4.5
Multiplier 2	25.8	4.1
Multiplier 3	54.2	0.096
Multiplier 4	54.9	0.083
Multiplier 5 [7]	35.7	0.72
Multiplier 6 [5]	52.4	0.14
Multiplier 7 [6]	53.5	0.11
Multiplier 8	18.7	10.4

Tables XI and XII show that the PSNRs of the output images generated by Multipliers 3 and 4, are nearly 50 dB, a value that is acceptable for most applications. Consistently, Multiplier 1 has the worst PSNR among 4 proposed designs. As discussed previously, the proposed approximate multipliers have a higher error distance for very large and very small input values in the product operands. Therefore the pixels that have high RGB (Red-Green-Blue) model values (such as of a white color) or small RGB model values (such as those of a black color), show a larger inaccuracy than other pixels due to the approximate nature of the compressors. However, the error distance of Multipliers3 and 4 still remains very low.

VII. CONCLUSION

Inexact computing is an emerging paradigm for computation at nanoscale. Computer arithmetic offers significant operational advantages for inexact computing; an extensive literature exists on approximate adders. However, this paper has initially focused on compression as used in a multiplier; to the best knowledge of the authors, no work has been reported on this topic.

This paper has presented the novel designs of two approximate 4-2 compressors. These approximate compressors are utilized in the reduction module of four approximate multipliers. The approximate compressors show a significant reduction in transistor count, power consumption and delay compared with an exact design.

- In terms of transistor count, the first design has a 46% improvement, while the second design has a 50% improvement.
- In terms of power consumption, the first design has a 57% improvement and the second design has a 60% improvement on average for CMOS implementation at feature sizes of 32, 22 and 16 nm.
- In terms of delay, the second design has a 44% improvement compared to the exact compressor and 35% improvement compared to the first design on average at different CMOS feature sizes of 32, 22 and 16 nm.

Four different approximate schemes have been proposed in this paper to investigate the performance of the approximate compressors for the aforementioned metrics for inexact multiplication. The approximate compressors have been utilized in the reduction module of a Dadda multiplier. The following conclusions can be drawn from the simulation results presented in this manuscript.

- The first and second proposed multipliers show a significant improvement in terms of power consumption and transistor count compared to an exact multiplier.
- The first and second multipliers have larger average NEDs (and thus, larger PSNRs), while the second multiplier that uses the second proposed approximate compressor for all bits, has the best delay.
- With relatively modest reductions in transistor count and power consumption, the third and fourth proposed multipliers have very low average NED values, thus presenting the best tradeoff for energy with accuracy.

Moreover, the application of these approximate multipliers to image processing has confirmed that two of the proposed designs achieve a PSNR of nearly 50dB in the output generated by multiplying two input images, thus viable for most applications.

Table XIII compares the four proposed approximate design with four other approximate designs found in the technical literature by ranking them under various metrics. Multiplier 4 is overall the best design with respect to all figures of merit for approximate multiplication as well as the two PSNR examples. Multiplier 5 has the best performance in terms of Max High NED and number of correct outputs; however, its rather poor performance for the other figures of merit causes its ranking to be in the middle once the PSNR examples are considered. Multiplier 3 is the second best design among the schemes considered in this manuscript. It offers overall good performance in most metrics of Table XIII. Current and future research addresses the tradeoffs of the different figures of merit in the proposed designs to establish conditions by which combined metrics can be attained. Moreover, physical designs of the approximate multipliers are being pursued to further confirm the analysis presented in this paper.

In conclusion, this paper has shown that by an appropriate design of an approximate compressor, multipliers can be designed for inexact computing; these multipliers offer significant advantages in terms of both circuit-level and error figures of merit. Although not discussed and beyond the scope of this manuscript, the proposed designs may also be useful in other arithmetic circuits for applications in which inexact computing can be used. The provision of an error indicator (as required for other applications) is a topic of current investigation.

TABLE XIII
RANKING OF APPROXIMATE MULTIPLIERS

Design	*Average NED*	*Max High NED*	*Max Low NED*	*Correct Outputs*	*PSNR example 1*	*PSNR example 2*
Multiplier 1	7	7	7	6	7	7
Multiplier 2	6	6	6	5	5	6
Multiplier 3	2	4	1	3	1	2
Multiplier 4	1	2	2	2	2	1
Multiplier 5 [7]	5	1	8	1	5	5
Multiplier 6 [5]	4	5	4	8	4	4
Multiplier 7 [6]	3	3	3	4	3	3
Multiplier 8	8	8	5	7	8	8

REFERENCES

[1] J. Liang, J. Han, F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," *IEEE Transactions on Computers*,vol. 63, no. 9, pp. 1760 - 1771, 2013.

[2] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," *Low Power Electronics and Design (ISLPED)* 2011 International Symposium on. 1-3 Aug. 2011.

[3] S. Cheemalavagu, P. Korkmaz, K.V. Palem, B.E.S. Akgul, and L.N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," in *Proc. IFIP-VLSI SoC*, Perth, Western Australia, Oct. 2005.

[4] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850-862, April 2010.

[5] M. J. Schulte and E. E. Swartzlander, Jr., "Truncated multiplication with correction constant," VLSI Signal Processing VI, pp. 388–396, 1993.

[6] E. J. King and E. E. Swartzlander, Jr., "Data dependent truncated scheme for parallel multiplication," in *Proceedings of the Thirty First Asilomar Conference on Signals, Circuits and Systems*, pp. 1178–1182, 1998.

[7] P. Kulkarni, P. Gupta, and MD Ercegovac, "Trading accuracy for power in a multiplier architecture", Journal of Low Power Electronics, vol. 7, no. 4, pp. 490--501, 2011.

[8] C. Chang, J. Gu, M. Zhang, "Ultra Low-Voltage Low- Power CMOS 4-2 and 5-2 Compressors for Fast Arithmetic Circuits," *IEEE Transactions on Circuits & Systems*, Vol. 51, No. 10, pp. 1985-1997, Oct. 2004.

[9] D. Radhakrishnan and A. P. Preethy, "Low-power CMOS pass logic 4-2 compressor for high-speed multiplication," in *Proc. 43rd IEEE Midwest Symp. Circuits Syst.*, vol. 3, 2000, pp. 1296–1298.

[10] Z. Wang, G. A. Jullien, and W. C. Miller, "A new design technique for column compression multipliers," *IEEE Trans. Comput.*, vol. 44, pp. 962–970, Aug. 1995.

[11] J. Gu, C. H. Chang, "Ultra Low-voltage, low-power 4-2 compressor for high speed multiplications," in *Proc. 36th IEEE Int. Symp. Circuits Systems*, Bangkok, Thailand, May 2003.

[12] M. Margala and N. G. Durdle, "Low-power low-voltage 4-2 compressors for VLSI applications," in *Proc. IEEE Alessandro Volta Memorial Workshop Low-Power Design*, 1999, pp. 84–90.

[13] B. Parhami, "Computer Arithmetic: Algorithms and Hardware Designs," 2nd edition, Oxford University Press, New York, 2010.

[14] K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors," in *Proc. of the 35th Asilomar Conf. on Signals, Systems and Computers*, vol. 1, 2001, pp. 129–133.

[15] Ercegovac, Miloš D., and Tomas Lang. *Digital arithmetic*. Elsevier, 2003.

[16] Baran, Dursun, Mustafa Aktan, and Vojin G. Oklobdzija. "Energy efficient implementation of parallel CMOS multipliers with improved compressors."*Proc. of the 16th ACM/IEEE international symposium on Low power electronics and design*. ACM, 2010.