DESIGN OF AN ACCURATE, COST-EFFECTIVE RADIX-4 BOOTH MULTIPLIER

[1]G Sravani, [2]K. Bhagya Lakshmi

[1]PG Scholar, M.Tech, Dept of ECE, Shadan Women's College of Engineering and Technology, Hyderabad, TS.
annu.sravanireddy@gmail.com

[2]Asst Professor, Dept of ECE, Shadan Women's College of Engineering and Technology, Hyderabad, TS.
bhagyalakshmi99@gmail.com

ABSTRACT

In many different applications, multiplication is a crucial procedure. As a result, the multiplier is a key element of many hardware systems. Radix-4 booth multipliers are frequently used for signed integer multiplication because they cut the number of partial products in half. In order to maximize energy economy, a number of approximative multipliers for radix-4 booth multiplication have recently been introduced. These multipliers are not suited for lossless applications due to the mistakes they exhibit. Accurate radix-4 booth multiplication is crucial for these applications. In this research, we present a multiplier and an efficient technique for precise radix-4 booth multiplication. The algorithm minimizes the amount of bits involved in addition during multiplication. The method also has enough storage space in its multiplier for a decreased design complexity. When the multiplier's accuracy is evaluated throughout a broad spectrum and in all conceivable multiplicand-multiplier pair combinations, it shows promising results of 100%. The multiplier is also demonstrated to be economical. In order to demonstrate our multiplier's accuracy, we compare it to the current radix-4 booth multipliers in terms of important error characteristics.

1. INTRODUCTION

Several machine learning and digital signal processing applications are highly influenced by the multiplication process. For instance, multiply-accumulate operations account for more than 90% of the computations in convolutional neural networks. Hence, the multiplier is a key component in various hardware platforms. Radix-4 booth multiplication is widely used for the multiplication of signed binary integers since it reduces the number of partial products by half compared to the conventional multiplication process.

Research in radix-4 booth multiplication made a shift from accurate to approximate computing design paradigm which constructs energy-efficient circuits at the cost of a little accuracy. Researchers have devised several radix-4 booth multipliers based on this emerging paradigm [1]–[10]. However, the errors featured by these multipliers are not tolerable by lossless applications. Besides, cost is another important parameter in the context of multiplier design. Therefore, there is a need for a cost-effective multiplier for accurate radix-4 booth multiplication.

In an attempt to address the above requirement, we introduce an accurate, cost-effective radix-4 booth multiplier in this paper. Initially, we devise an optimized algorithm for accurate radix-4 booth multiplication by reducing the number of bits that participate in the addition process during radix-4 booth multiplication while having sufficient storage area for reduced hardware design complexity. The accuracy of the multiplier is tested for a wide range and all possible cases of multiplicand multiplier pairs. The cost of the multiplier circuit and its operations are critically analyzed. Besides, the proposed multiplier's accuracy is compared with the current radix-4 multipliers in key error parameters viz., Mean Relative Error Distance (MRE/MRED), Normalized Mean Error Distance (NMED), and Probability Relative Error Distance (PRED). are widely used in arithmetic units of microprocessors, multimedia and digital signal processors; moreover, high performance and low power multipliers are in high demand for embedded systems. It is becoming extremely difficult to further improve performance and reduce the power consumption of multipliers under the requirement of full accuracy; however, the requirements of high precision and exactness are not so strict for many applications related to human perception, such as multimedia signal processing and machine learning. High precision and exactness in the operations of digital logic circuits are related to the generally accepted requirement of correctness of information processing; numerous error-tolerant applications can be found in computing and by relaxing the requirement of strict accuracy, performance and power consumption can be substantially improved [1]. This design principle is generally known as approximate or inexact computing [2].

As the basic operations of an arithmetic processor, addition and multiplication are very important for achieving high performance. Addition has been extensively studied for approximate computing for reduction in power consumption and delay [3-5]. New metrics including error distance (ED), mean error distance (MED) and normalized error distance (NED) have been proposed for evaluating the designs of approximate adders [6]. Approximate multiplication has not been extensively studied despite its importance for arithmetic processing and systems; multiplication is more complex compared with addition, because it

requires the accumulation of partial product rows. The most widely used high performance multiplier consists of a modified Booth encoding (MBE) to reduce the number of partial product rows by half as the first step [7, 8].

The current designs for an approximate multiplier can be categorized as truncation and non-truncation schemes. A truncation-based design relies on a simple approximation in which either the lower part of the partial products is removed, or the least significant partial products are estimated by a constant (so referred to as fixed-width multiplier design [9]); however, the error generated by the truncated partial product rows can be rather large. Therefore, error compensation strategies have been proposed to increase the accuracy of truncated multipliers; an inexact array multiplier has been proposed by ignoring some of the least significant columns of the partial products and considering them as a constant [3]. In [10], the truncated multiplier utilizes a correction constant selected according to both the reduction and rounding errors. However, this truncated multiplier incurs a very large error if the partial products in the least significant columns are all ones or all zeros; therefore, a truncated multiplier with variable correction has been also proposed in [11]. Recently, a few error compensation strategies have been proposed to further improve the accuracy of fixed-width Booth multipliers [9, 12-15]. An error is compensated with the outputs of the Booth encoders in [9]; the error compensation circuit proposed in [12] mainly uses a simplified sorting network. An adaptive conditional-probability estimator has been proposed in [15] to compensate the quantization error of a fixed-width Booth multiplier. These truncated Booth multipliers use error compensation circuits to improve the accuracy. However, the extra compensation circuits require additional hardware; approximate computing can be employed to reduce such overhead.

A non-truncation scheme utilizes approximate circuits to assemble an approximate multiplier. An approximate 2×2 multiplier has been proposed in [16] by simplifying its logic expression using a Karnaugh-Map (K-map); this circuit is then used as a basic block for larger size multipliers. Compressors or counters are widely used to accelerate the accumulation of partial products in the design of a high-speed multiplier [17, 18]; an inexact 4:2 counter has been used to design an approximate 4×4 Wallace multiplier for assembling larger size multipliers [17]. Two approximate 4-2 compressors are proposed in [18] and used in a Dadda tree of 8×8 array multipliers. An 8×8 multiplier using approximate adders that ignore carry propagation between partial products, has been proposed in [19]. Hybrid Approximate Multiplier (HAM) schemes are proposed by exploring perforation, logic approximation and voltage over-scaling techniques in [20]. It is found that higher power savings for the same error values can be achieved by applying hybrid approximation techniques than using only single design techniques. A dynamic range unbiased multiplier (DRUM) with a dynamic range selection scheme is proposed in [21]; this scheme based on approximation in operands has an unbiased error distribution. In [22], approximate radix-8 Booth multipliers are proposed by using an approximate 2-bit adder for generating the odd multiples (×3) of the multiplicand.

However, there is little study in the technical literature on the approximate design of a radix-4 Booth multiplier as one of the most popular schemes for signed multiplication. In the first step of a radix-4 Booth multiplier, a radix-4 modified Booth encoding (MBE) is used to generate the partial products [23]; a radix-4 MBE can reduce the number of partial products by a factor of two. The implementation of the MBE significantly affects the area, delay and power consumption of Booth multipliers [24]; in the traditional MBE algorithm, an extra partial product bit is generated at the least significant bit (LSB) position of each partial product row due to the negative encoding. This leads to an irregular partial product array as requiring a complex reduction tree.

1.2 EXISTING SYSTEM:

In Existing system two approximate radix-4 MBE algorithms are proposed and analyzed. Booth multipliers are designed based on the proposed radix-4 MBEs, in which a regular partial product array is achieved by using the proposed approximate Wallace tree structure. The error characteristics are analyzed with an approximation factor that is related to the inexact bit width of the Booth multipliers. Simulation results at 45 nm CMOS technology on delay, area, power consumption are also provided. Case studies for image processing are presented to show the validity of the proposed approximate radix-4 Booth multipliers. This paper is an extension of our previous work presented in [25]; the main differences and novel contributions are summarized as follows:

1) A more efficient approximate radix-4 Booth encoder is proposed in this paper. The designs of both approximate radix-4 Booth encoders are presented and extensively analyzed.

2) Approximate Booth multipliers are proposed using approximate Booth encoders, in which the features of an approximate regular tree structure are illustrated in detail.

3) An approximation factor is proposed to assist in the design of the approximate Booth multipliers and facilitate its error analysis.

4) The proposed approximate Booth multipliers are comprehensively evaluated with respect to both hardware implementation and error analysis.

5) The proposed approximate Booth multipliers are applied in image processing.

1.3 PROPOSED SYSTEM

We propose an accurate, cost-effective radix-4 booth multiplier in this paper. Initially, we devise an optimized algorithm for accurate radix-4 booth multiplication by reducing the number of bits that participate in the addition process during radix-4 booth multiplication while having sufficient storage area for reduced hardware design complexity. The accuracy of the multiplier is tested for a wide range and all possible cases of multiplicand multiplier pairs. The cost of the multiplier circuit and its operations are critically analyzed. Besides, the proposed multiplier's accuracy is compared with the current radix-4 multipliers in key error parameters viz., Mean Relative Error Distance (MRE/MRED), Normalized Mean Error Distance (NMED), and Probability Relative Error Distance (PRED).

2. LITERATURE SURVEY

Liu et al. [1] presented "two approximate radix-4 booth encoding algorithms" viz.," R4ABE1 and R4ABE2". They also designed "two approximate booth multipliers: R4ABM1 and R4ABM2", based on the algorithms with an approximate wallace tree structure in place. In this work, the energy efficiency of the multipliers is decided by a parameter called the approximation factor. Among the two multipliers, R4ABM2 produces relatively more error. For the approximation factor 2, it produces MRED, NMED, and PRED of 0.34410□2, 0.254310□5, and 99.79% respectively. Whereas, for the approximation factor 32, it produces the highest error with MRED, NMED, and PRED of 2.52107, 1787910□5, and 0.123% respectively.

In [2], the authors devised "an approximate hybrid high radix encoding" to generate the partial products for signed multiplication. In this technique, "accurate radix-4 encoding" is used to encode the Most Significant Bits (MSBs), and "approximate higher radix encoding" is used to encode the Least Significant Bits (LSBs). "Three approximate multipliers: RAD64, RAD256 and RAD1024", are presented. Of the three multipliers, RAD1024 produces the highest error with an MRE of 0.93% and PRED of 93.26%. The authors did not consider evaluating NMED in this work. Ansari et al. [3] presented "an improved, approximate 4:2 compressor", in which generate and propagate signals are used for encoding the inputs. This has resulted in the reduced number of flawed rows in the truth table of the compressor. Based on this compressor, the authors designed" an approximate multiplier called CABM". CABM produces MRED and NMED of 0.014 and 0.18 respectively. "Three approximate radix-4 booth multipliers: ABM-M1, ABM-M2, and ABM-M3", are proposed in [4]. In each of the multipliers, a unique approximation technique is employed which encompasses reduction in the logic complexity of "booth partial product generator" and "partial product accumulation". Among the three multipliers, ABM-M2 generates the highest error with MRED and NMED of 2.689_10-2 and 1087.270_10-6 respectively.

The authors of [5] designed "two approximate multipliers: LOBO10 and LOBO12", using "radix-4 booth encoding" and "logarithmic product approximation". The multipliers encode the MSBs using accurate radix-4 booth encoding and approximate the higher radix partial products arising from the LSBs using logarithmic approximation. Of the two multipliers, LOBO12 generates the highest error with MRE, NMED, and PRED of 0.44-96.09%, 18.45_10□4, and 90.99% respectively.

[6] introduces "two approximate multipliers: HLR-BM1 and HLR-BM2", based on hybrid low radix encoding. In this work, the odd multiples of radix-8 are approximated to their nearest powers of two to fix the low-performance issue of radix-8 booth encoding. Among the two multipliers, HLRBM1 produces the highest error with MRE, NMED, and PRED of 0.03%, 0.79_10□5, and 98.35% respectively. The aforementioned approaches are inclined to approximate computing design paradigm which aims to construct energy efficient circuits at the cost of a little accuracy. Not all applications are inherently resilient to accuracy loss. For instance, lossless applications such as medical image compression have zero tolerance to accuracy loss. Besides, the approximate encoding techniques discussed in the aforementioned approaches are based on accurate radix-4 encoding. This signifies the importance of accurate radix-4 encoding in the context of signed multiplication.

"Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing", W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi

Approximate computing is an attractive design methodology to achieve low power, high performance (low delay) and reduced circuit complexity by relaxing the requirement of accuracy. In this paper, approximate Booth multipliers are designed based on approximate radix-4 modified Booth encoding (MBE) algorithms and a regular partial product array that employs an approximate Wallace tree. Two approximate Booth encoders are proposed and analyzed for error-tolerant computing. The error characteristics are analyzed with respect to the so-called approximation factor that is related to the inexact bit width of the Booth multipliers. Simulation results at 45 nm feature size in CMOS for delay, area and power consumption are also provided. The results show that the proposed 16-bit approximate radix-4 Booth multipliers with approximate factors of

12 and 14 are more accurate than existing approximate Booth multipliers with moderate power consumption. The proposed R4ABM2 multiplier with an approximation factor of 14 is the most efficient design when considering both power-delay product and the error metric NMED. Case studies for image processing show the validity of the proposed approximate radix-4 Booth multipliers.

"Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers", V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi

Approximate computing forms a design alternative that exploits the intrinsic error resilience of various applications and produces energy-efficient circuits with small accuracy loss. In this paper, we propose an approximate hybrid high radix encoding for generating the partial products in signed multiplications that encodes the most significant bits with the accurate radix-4 encoding and the least significant bits with an approximate higher radix encoding. The approximations are performed by rounding the high radix values to their nearest power of two. The proposed technique can be configured to achieve the desired energy-accuracy tradeoffs. Compared with the accurate radix-4 multiplier, the proposed multipliers deliver up to 56% energy and 55% area savings, when operating at the same frequency, while the imposed error is bounded by a Gaussian distribution with near-zero average. Moreover, the proposed multipliers are compared with state-of-the-art inexact multipliers, outperforming them by up to 40% in energy consumption, for similar error values. Finally, we demonstrate the scalability of our technique.

"Low-Power Approximate Multipliers Using Encoded Partial Products and Approximate Compressors", M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han,

Approximate computing has been considered to improve the accuracy-performance tradeoff in error-tolerant applications. For many of these applications, multiplication is a key arithmetic operation. Given that approximate compressors are a key element in the design of power-efficient approximate multipliers, we first propose an initial approximate 4:2 compressor that introduces a rather large error to the output. However, the number of faulty rows in the compressor's truth table is significantly reduced by encoding its inputs using generate and propagate signals. Based on this improved compressor, two 4 × 4 multipliers are designed with different accuracies and then are used as building blocks for scaling up to 16 × 16 and 32×32 multipliers. According to the mean relative error distance (MRED), the most accurate of the proposed 16 × 16 unsigned designs has a 44% smaller power-delay product (PDP) compared to other designs with comparable accuracy. The radix-4 signed Booth multiplier constructed using the proposed compressor achieves a 52% reduction in the PDP-MRED product compared to other approximate Booth multipliers with comparable accuracy. The proposed multipliers outperform other approximate designs in image sharpening and joint photographic experts group applications by achieving higher quality outputs with lower power consumptions. For the first time, we show the applicability and practicality of approximate multipliers in multiple-input multiple-output antenna communication systems with error control coding.

"Design and Analysis of Area and Power Efficient Approximate Booth Multipliers", S. Venkatachalam, E. Adams, H. J. Lee, and S.-B. Ko,

Approximate computing is an emerging technique in which power-efficient circuits are designed with reduced complexity in exchange for some loss in accuracy. Such circuits are suitable for applications in which high accuracy is not a strict requirement. Radix-4 modified Booth encoding is a popular multiplication algorithm which reduces the size of the partial product array by half. In this paper, three Approximate Booth Multiplier Models (ABM-M1, ABM-M2, and ABM-M3) are proposed in which approximate computing is applied to the radix-4 modified Booth algorithm. Each of the three designs features a unique approximation technique that involves both reducing the logic complexity of the Booth partial product generator and modifying the method of partial product accumulation. The proposed approximate multipliers are demonstrated to have better performance than existing approximate Booth multipliers in terms of accuracy and power. Compared to the exact Booth multiplier, ABM-M1 achieves up to a 23 percent reduction in area and 15 percent reduction in power with a Mean Relative Error Distance (MRED) value of $7{:}9 \times 10^{-4}$. ABM-M2 has area and power savings of up to 51 and 46 percent respectively with a MRED of $2{:}7 \times 10^{-2}$. ABM-M3 has area savings of up to 56 percent and power savings of up to 46 percent with a MRED of $3{:}4 \times 10^{-3}$. The proposed designs are compared with the state-of-the-art existing multipliers and are found to outperform them in terms of area and power savings while maintaining high accuracy. The performance of the proposed designs are demonstrated using image transformation, matrix multiplication, and Finite Impulse Response (FIR) filtering applications.

"On the Design of Logarithmic Multiplier Using Radix-4 Booth Encoding", R. Pilipovic and P. Bulic

This paper proposes an energy-efficient approximate multiplier which combines radix-4 Booth encoding and logarithmic product approximation. Additionally, a

datapath pruning technique is proposed and studied to reduce the hardware complexity of the multiplier. Various experiments were conducted to evaluate the multiplier's error performance and efficiency in terms of energy and area utilization. The reported results are based on simulations using TSMC-180nm. Also, the applicability of the proposed multiplier is examined in image sharpening and convolutional neural networks. The applicability assessment shows that the proposed multiplier can replace an exact multiplier and deliver up to a 75% reduction in energy consumption and up to a 50% reduction in area utilization. Comparative analysis with the state-of-the-art multipliers indicates the potential of the proposed approach as a novel design strategy for approximate multipliers. When compared to the state-of-the-art approximate non-logarithmic multipliers, the proposed multiplier offers smaller energy consumption with the same level of applicability in image processing and classification applications. On the other hand, some state-of-the-art approximate logarithmic multipliers exhibit lower energy consumption than the proposed multiplier but deliver significant performance degradation for the selected application cases.

3. PROPOSED METHODOLOGY

3.1 BLOCK DIAGRAM

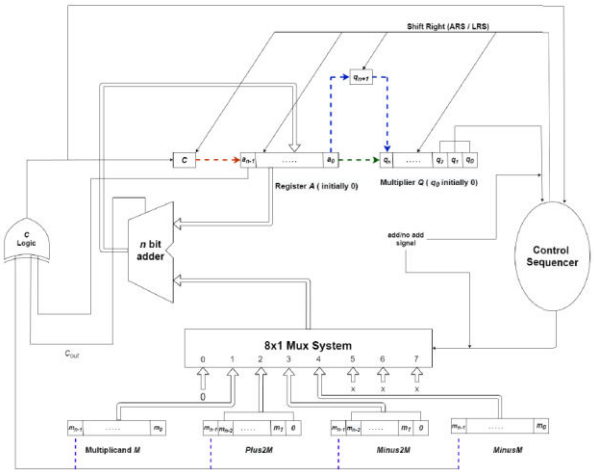


Fig.: Proposed radix-4 booth multiplier

3.2 PROPOSED ALGORITHM AND MULTIPLIER:

A. ALGORITHM

The proposed algorithm is illustrated in Algorithm 1. We have the input variables ‘m’ and ‘q’, two decimal integers converted into two’s complement binary integers of length ‘n’ bits each, where ‘n’ is calculated as the maximum of the number of bits used to represent ‘m’ and ‘q’ as binary numbers.

The following arrays are used: M (‘n’ bits) to hold the binary representation of ‘m’, C (1 bit) to hold the carry, and A (‘n’ bits) to hold the intermediate summation results of the partial products in binary representation. The array Q to hold the binary representation of multiplier ‘q’ has some additional components. After converting ‘q’ to a binary number, an implicit zero is added to the right of the LSB.

The crux of radix-4 booth multiplication is to convert the multiplier in its binary form into a recoded multiplier by checking whether successive triplets (groups of 3 bits) are being formed or not, starting from LSB. Therefore, after adding the implicit zero, the aforesaid condition is checked, and if it’s not satisfied, sign extension is done to complete the triplet. The process of forming the triplets is shown in Fig. 1.

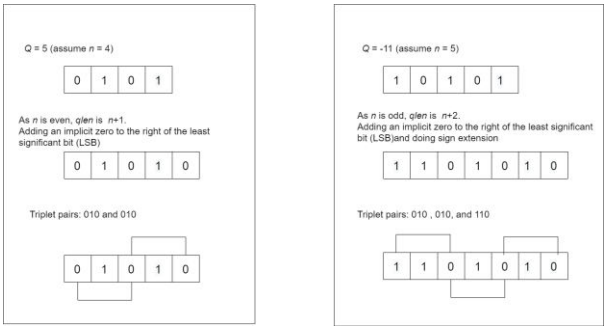


Fig. 1: Triplets formation.

Sign extension will be needed only when 0n0 is odd. Thus, Q is n + 1 bits long if 0n0 is even, and n + 2 bits long if 0n0 is odd.

To save repeated computations, the values of +2M, □2M, □M are computed beforehand as follows:

_ □M is the 2’s complement of multiplicand 0m0 and is stored as a binary number in array MinusM (0n0 bits)

_ +2M is the binary representation of multiplicand 0m0, with an added zero at the LSB, and is stored in array Plus2M (n + 1 bits)

_ □2M is the binary representation of the 2’s complement of multiplicand 0m0, with an added zero at the LSB, and is stored in the array Minus2M (n + 1 bits)

In each iteration, the last 3 digits of Q determine which operation will take place. The variable Cout will have the carry-out value after the addition of A and M=MinusM=Plus2M=Minus2M. The variable C takes three intermediate variables and performs eXclusive-OR (XOR) operation on them, denoted by _ operator in the algorithm. C determines whether to carry out Logical Right Shift (LRS) or Arithmetic Right Shift (ARS). The final product is the concatenation (denoted by $ operator) of the values in arrays A and Q, where the higher-order half is stored in A and the lower-order half in Q, except for the last bit of Q.

B. MULTIPLIER

Fig. 2 introduces the multiplier for the proposed radix-4 booth multiplication algorithm, for multiplication of two signed integers, both having 0n0 bits in their binary representation. The hardware mainly consists of an 8 _ 1 Multiplexer (Mux), an n-bit adder, a carry logic, a control sequencer and registers C (1 bit), A (0n0 bits), M (0n0 bits), Plus2M (n+1 bits), Minus2M (n+1 bits), MinusM (0n0 bits), qn+1 (1 bit) and Q (n + 1 bits). A and Q are shift registers, concatenated as shown in Fig. 2. These registers are analogous to the arrays declared in the algorithm.

The register Q holds the binary representation of the multiplier, including the implicit zero to the right of LSB. If 0n0 is odd, then the input and output lines to and from the register qn+1 are enabled, and the shift line from a0 to qn is disabled. In this case, qn+1 holds the sign extension bit for Q. If 0n0 is even, then the input and output lines to and from qn+1 are disabled, and the shift line from a0 to qn is enabled. The register M holds the binary representation of the multiplicand, register MinusM the 2's complement representation of M, register Plus2M the binary representation of M after logical left shift by one-bit position, and register Minus2M the binary representation of MinusM after logical left shift by one-bit position.

At the start, the multiplier and the implicit zero are loaded into register Q, the multiplicand into register M, and registers C and A are cleared to 0. The carry logic is essentially an XOR gate whose output is directed to C and the control sequencer. The product is computed in L cycles (whose value is ceil(n=2)), for which the underlying logic is present in the control sequencer. At the beginning of each cycle, the last 3 bits (starting from q0) of Q are sent to the control sequencer to determine which operation will take place.

A signal from the control sequencer to determine whether the addition of M, Plus2M, MinusM, Minus2M or no addition takes place, is sent to the 8 x 1 Mux and accordingly M, Plus2M, MinusM, Minus2M or 0 is sent as the output of the 8 x 1 Mux to the n-bit adder, and its MSB is sent to the carry logic. The MSB of the current value in register A is sent to the carry logic, and A is sent as input to the n-bit adder. The sum from the n-bit adder is sent to A, and the carry-out
is sent to the carry logic. The output of the carry logic is sent to C.

Depending on the output of the carry logic, a signal is sent from the control sequencer to C, A, and Q. If the output of carry logic is "0", and if 0n0 is even, then the logical right shift of A and Q takes place. If 0n0 is odd, then the logical right shift of A, qn+1, and Q by two-bit positions takes place. If the output is "1", and if 0n0 is even, then the arithmetic right shift of C, A, and Q takes place. If 0n0 is odd in this case, then the arithmetic right shift of C, A, qn+1, and Q by two-bit positions takes place.

Together, registers A and Q hold the intermediate partial product while the last 3 bits of Q (starting from LSB) generate the add/no-add signal, which determines the next operation that will take place.

After L cycles, the higher-order half of the product is held in register A, and the lower-order half in register Q (qn to q1 bits) when 0n0 is even, and in registers qn+1 and Q (qn to q1 bits) when 0n0 is odd.

****

Algorithm 1 Optimized radix-4 booth multiplication.

Input: Two integers 0m0 and 0q0 in decimal form.
Output: Product of 0m0 and 0q0.
1. Convert 0m0 and 0q0 into binary numbers of 0n0 bits
2. if (n%2 = 0), then
qlen <- n + 1
else
qlen <- n + 2
end if
3. Declare arrays M[n], C[1], A[n], Q[qlen]
4. A <- 0
5. M <- binary equivalent of 0m0
6. L <- ceil(n=2)
7. C <- 0
8. Q[0 : qlen □ 2] <- binary equivalent of 0q0
9. Q[qlen □ 1] <- 0
10. Check whether, in Q, groups of 3 bits are being formed or not. If not, do sign extension
11. Compute the values of +2M, □2M, □M and store them in arrays Plus2M[n + 1], Minus2M[n + 1], and MinusM[n]
12. while L > 0, do
if Q[qlen □ 3 : qlen □ 1] = 000 or 111, then
c <- 0
a <- A[0]
else if Q[qlen □ 3 : qlen □ 1] = 001 or 010, then
c <- M[0]
a <- A[0]
A <- A +M
else if Q[qlen □ 3 : qlen □ 1] = 011, then
c <- Plus2M[0]
a <- A[0]
A <- A + Plus2M
else if Q[qlen □ 3 : qlen □ 1] = 100, then
c <- Minus2M[0]
a <- A[0]
A <- A +Minus2M
else if Q[qlen □ 3 : qlen □ 1] = 101 or 110, then
c <- MinusM[0]
a <- A[0]
A <- A +MinusM
end if
Cout <- carry out after binary addition

C <- Cout _ a _ c
if C = 0, then
LRS(A$Q) by 2 times
else
ARS(CAQ) by 2 times
end if
L <- L □ 1
end while
13. res <- (A$Q[0 : qlen □ 2])
14. prod <- decimal form of res
15. return prod
****

B. APPROXIMATE RADIX-4 BOOTH MULTIPLIERS

A Booth multiplier consists of three parts: partial product generation using a Booth encoder, partial product accumulation using compressors and final product generation using a fast adder. The approximate design of radix-4 Booth encoding is studied in this section. A more efficient approximate radix-4 Booth encoding method is proposed in this section by carefully considering the error characteristics. Furthermore, an approximate partial product array produced by the Booth encoding is also designed to make it regular, such that a reduction stage is saved. Approximate Booth multipliers are designed based on the approximate Booth encoder and the regular approximate partial product array.

Review of Radix-4 Booth Multiplication

Booth encoding has been proposed for improving the performance of multiplication of two′s complement binary numbers [7]; it has been further improved by the MBE or radix-4 Booth encoding [8]. The Booth encoder plays an important role in the Booth multiplier, which reduces the number of partial product rows by half. Consider the multiplication of two N-bit integers, i.e.,
a multiplicand A and a multiplier B in two′s complement; this is given as follow

$$A = -a_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i\, 2^i,$$
$$B = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i\, 2^i.$$

In a Booth encoder, each group is decoded by selecting the partial products as -2A, -A, 0, A, or 2A. The negation operation is performed by inverting each bit of A and adding a ‘1’ (defined as Neg) to the LSB [23, 24].
The circuit diagrams of the radix-4 Booth encoder and decoder are provided in [23]. The output (i.e., the partial product, pp_{ij}) of the Booth encoder is given as follows:

$$pp_{ij} = (b_{2i} \oplus b_{2i-1})(b_{2i+1} \oplus a_j) + \overline{(b_{2i} \oplus b_{2i-1})}(b_{2i+1} \oplus b_{2i})(b_{2i+1} \oplus a_{j-1}).$$

Approximate Radix-4 Booth Encoding Method 1

The K-map of the first approximate radix-4 Booth encoding (R4ABE1) method is shown in Table 1, where ○0 denotes an entry in which a ′1′ is replaced by a ′0′. Only four entries are modified to simplify the Booth encoding; the strategy for the first approximate design is to make the truth table as symmetrical as possible and introduce a small error. Thus, the advantage of the R4ABE1 design is that a very small error occurs, as only four entries are modified; however, all modifications change a ′1′ to a ′0′, so the absolute value of approximate product is always smaller than its exact counterpart.
The modified truth table shows two XOR functions; therefore, the output of R4ABE1 is given as follows:

$$\begin{aligned} pp_{ij} &= a_j \overline{b_{2i+1} b_{2i}} b_{2i-1} + a_j \overline{b_{2i+1}} b_{2i} \overline{b_{2i-1}} \\ &\quad + \bar{a}_j b_{2i+1} b_{2i} \overline{b_{2i-1}} + \bar{a}_j b_{2i+1} \overline{b_{2i}} b_{2i-1} \\ &= (b_{2i} \oplus b_{2i-1})(b_{2i+1} \oplus a_j). \end{aligned}$$

Compared with the exact MBE (Eq. 3), R4ABE1 can significantly reduce both the complexity and the critical path delay of Booth encoding. The error rate (defined as the probability that at least a bit differs from the exact result, given a uniform distribution of inputs), denoted by Pe, is
given by:

$$P_e = 4/32 = 12.5\%.$$

The gate level structure of R4ABE1 is shown in Fig. 1(a); the conventional design of MBE [23] consists of four XNOR-2 gates, one XOR-2 gate, one OR-3 gate, one OR-2 gate and one NAND-2 gate. The R4ABE1 design only requires two XOR-2 gates and one AND-2 gate. If the 2-input XOR and XNOR gates are implemented using transmission gates and the other gates are implemented as a complex gate, the transistor count of the MBE for a full CMOS implementation is 34 [23], while the transistor count of the proposed R4ABE1 is only 12, i.e., a reduction of over 64% in terms of circuit complexity is achieved.
2), the conventional MBE has a normalized delay of 2.5; therefore, its critical path delay is reduced to 1.7-unit delay in the R4ABE1 design, i.e. an improvement of 32% in delay.

Approximate Radix-4 Booth Encoding Method 2

The truth table of the second approximate radix-4 Booth encoding (R4ABE2) method is shown in Table 3, where ⃝1 denotes a ′0′ entry that has been replaced by a ′1′; eight entries in the K-map are modified to simplify the logic of the Booth encoding. The strategy for R4ABE2 is that in addition to having a symmetric truth table at a small error, the number of prime implicants (identified by rectangle) should be as small as possible too. Although the error introduced by R4ABE2 is nearly doubled compared with R4ABE1, the modification is achieved by not only changing a ′1′ to a ′0′, but also changing a ′0′ to a ′1′. Thus, the approximate product can be either larger or smaller than the exact product and errors can complement each other in the partial product reduction process. Therefore, when using R4ABE2 in a Booth multiplier, the error may not be larger than for a Booth multiplier with R4ABE1.

The modified truth table contains only two prime implicant rectangle; therefore, the output of R4ABE2 is given as follows:

$$pp_{ij} = a_j\overline{b_{2i+1}} + \bar{a}_j b_{2i+1} = b_{2i+1} \oplus a_j.$$

R4ABE2 further reduces the complexity and critical path delay compared with R4ABE1 (Eq. 4). The error rate is now given by:

$$P_e = 8/32 = 25\%.$$

R4ABE2 only requires one XOR-2 gate by using transmission gates, so the transistor count of R4ABE2 is 4. R4ABE2 reduces the complexity of the Booth encoder by over 88% compared with MBE. By using the normalized gate delay model in Table 2 [23], the critical path delay of R4ABE2 is 1.0; so, it improves the delay by 60% compared with MBE.

Approximate Regular Partial Product Array

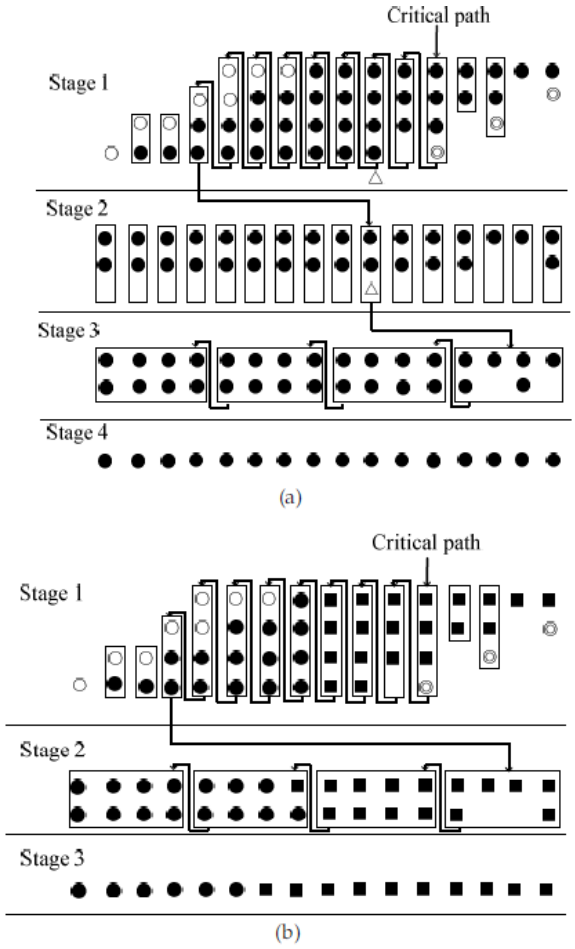
For a more regular partial product array (requiring a smaller reduction stage), the Neg term in the (N/2+1)th row of the approximate design of a Booth multiplier can be ignored (shown as △ in Fig. 2(a)). For an N-bit radix-4 Booth multiplier when N is a power of 2, removing the extra Neg term significantly reduces the critical path, area and power when the 4-2 compressor is used for the partial product accumulation. In the approximate partial product array (Fig. 2(b)), one reduction stage is saved; this significantly reduces the complexity and critical path delay.

The error rate of the approximate partial product array with the ignored Neg bit is 37.5% and its logic function is given by:

$$Neg_{N/2-1} = b_{2N+1}\overline{b_{2N}} + b_{2N+1}\overline{b_{2N-1}} = b_{2N+1}\overline{b_{2N}b_{2N-1}}.$$

Design of Approximate Booth Multipliers

R4ABE1 and R4ABE2 are applied to a Booth multiplier design. In the approximate Booth multiplier, the proposed approximate Booth encoders, i.e. R4ABE1 and R4ABE2, are used in the first part to generate the inexact partial products. The approximate Booth encoders can then be used in all or only part of the partial product generation process; therefore, an approximation factor p (p=1, 2, ..., 2N) is defined as the number of least significant partial product columns that are generated by the approximate Booth encoders. The approximate partial products are accumulated with the exact 4-2 compressor. The last part uses an exact carry-lookahead adder (CLA) to compute the final product result.



Two types of approximate Booth multipliers are proposed:

1. The first approximate radix-4 Booth multiplier (R4ABM1) uses R4ABE1 (to generate the p least significant partial product columns) and the regular approximate partial product array. The exact MBE is used for generating the 2N-p most significant partial

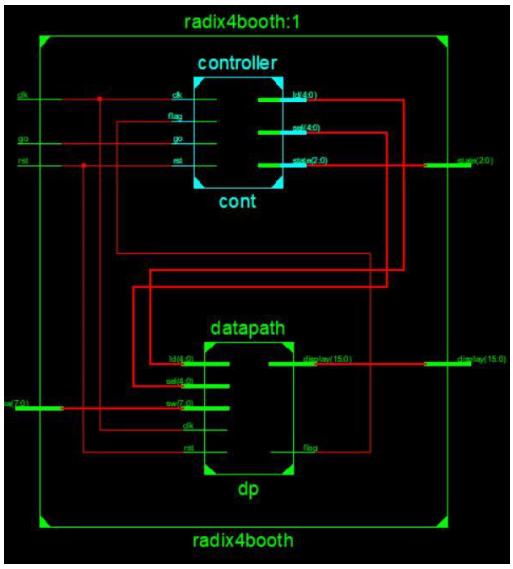
product columns. The exact 4-2 compressors are used to accumulate both approximate and exact partial products.

2. Similar as R4ABM1, the second approximate radix-4 Booth multiplier (R4ABM2) uses R4ABE2 (to generate the p least significant partial product columns), the regular approximate partial product array and the exact 4-2 compressors.

As the error is controlled by the approximation factor p, a reasonable accuracy can be achieved for different applications; Fig. 2(b) shows an approximate 8×8 Booth multiplier using either R4ABE1, or R4ABE2 with p=8.

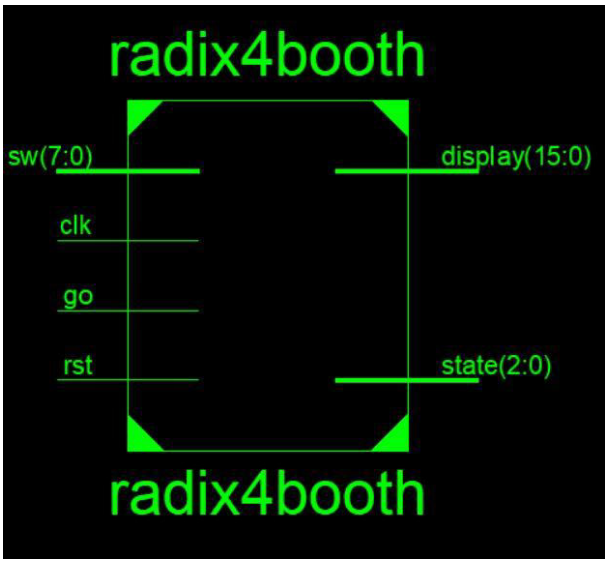
For p<7 of any word length, the delay of an approximate Booth multiplier is only improved by the approximate regular partial product array, since the critical path starts from p=7. For p>6, the delay of the entire multiplier can be further reduced due to the approximate Booth encoder. However, for both cases, the approximate design of a Booth encoder can significantly reduce the area and power consumption.

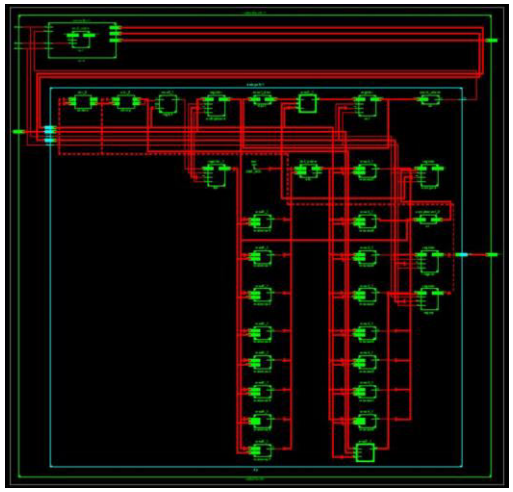
5 SIMULATION RESULTS:



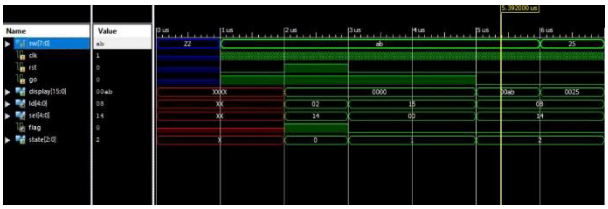
addertest Project Status (11/07/2022 - 12:21:43)			
Project File:	work.xise	Parser Errors:	No Errors
Module Name:	radix4booth	Implementation State:	Synthesized
Target Device:	xc6slx75-3csg484	* Errors:	No Errors
Product Version:	ISE 14.7	* Warnings:	107 Warnings (45 new)
Design Goal:	Balanced	* Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	* Timing Constraints:	
Environment:	System Settings	* Final Timing Score:	

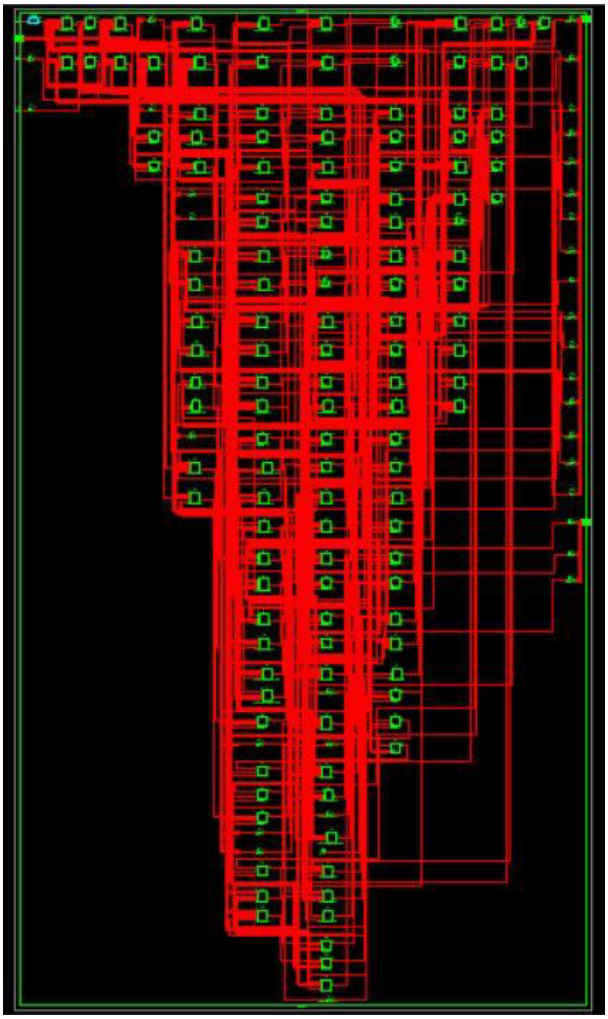
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	48	93296	0%
Number of Slice LUTs	81	46648	0%
Number of fully used LUT-FF pairs	41	88	46%
Number of bonded IOBs	30	328	9%
Number of BUFG/BUFGCTRLs	1	16	6%





Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	48	93,296	1%	
Number used as Flip Flops	48			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	75	46,648	1%	
Number used as logic	75	46,648	1%	
Number using O6 output only	71			
Number using O5 output only	0			
Number using O5 and O6	4			
Number used as ROM	0			
Number used as Memory	0	11,072	0%	
Number of occupied Slices	25	11,662	1%	
Number of MUXCYs used	0	23,324	0%	
Number of LUT Flip Flop pairs used	81			
Number with an unused Flip Flop	35	81	43%	
Number with an unused LUT	6	81	7%	
Number of fully used LUT-FF pairs	40	81	49%	
Number of unique control sets	7			
Number of slice register sites lost to control set restrictions	24	93,296	1%	
Number of bonded IOBs	30	328	9%	
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	





6. CONCLUSION

In this study, we presented a precise, economical radix-4 booth multiplier. By lowering the number of bits involved in the addition process while having enough storage space for a simplified hardware design, we were able to optimise the method for multiplying two signed binary numbers using radix-4 booth multiplication. For the same, we have created a high-level architecture. We created a Python programme to compute the product of two signed integers using the suggested method as the blueprint, allowing us to assess the technique's practicality. When we tested it against a sample size of 10,000 random integer pairs in the field of, we got encouraging results, i.e. 100% accuracy (-5000,5000). Additionally, we contrasted it with the existing radix-4 booth multipliers in a few critical error metrics, and the results demonstrated that our technique functioned more accurately. By examining the price of the multiplier circuit and how it functions, we were able to demonstrate how our technique is more cost-effective than the original approach. Convolutional neural network calculations are only one example of how the suggested approach might be applied in a variety of machine learning and digital signal processing applications. Additionally, it may be utilised in lossless applications with extremely high error tolerance. The hardware architecture will be simulated in future work, and its energy efficiency will be assessed.

REFERENCES:

[1] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, “Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing,” IEEE Transactions on Computers, vol. 66, pp. 1435–1441, 2017.

[2] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, “Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers,” IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 26, pp. 421–430, 2018.

[3] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, “Low-Power Approximate Multipliers Using Encoded Partial Products and Approximate Compressors,” IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 8, pp. 404–416, 2018.

[4] S. Venkatachalam, E. Adams, H. J. Lee, and S.-B. Ko, “Design and Analysis of Area and Power Efficient Approximate Booth Multipliers,” IEEE Transactions on Computers, vol. 68, pp. 1697–1703, 2019.

[5] R. Pilipovic and P. Bulic, “On the Design of Logarithmic Multiplier Using Radix-4 Booth Encoding,” IEEE Access, vol. 8, pp. 64 578– 64 590, 2020.

[6] H. Waris, C. Wang, and W. Liu, “Hybrid Low Radix Encoding based Approximate Booth Multipliers,” IEEE Transactions on Circuits and Systems II: Express Briefs, to be published.

[7] G. Jain, M. Jain, and G. Gupta, “Design of Radix-4,16,32 Approx Booth Multiplier Using Error Tolerant Application,” in 6th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2017.

[8] S. Venkatachalam, H. J. Lee, and S.-B. Ko, “Power Efficient Approximate Booth Multiplier,” in IEEE International Symposium on Circuits and Systems (ISCAS), 2018.

[9] T. Zhang, W. Liu, J. Han, and F. Lombardi, “Design and Analysis of Majority Logic Based Approximate Radix-4 Booth Encoders,” in IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2019.

[10] N. R. Varghese and S. Rajula, “High Speed Low Power Radix 4 Approximate Booth Multiplier,” in 3rd International Conference on Electronics, Materials Engineering and Nano-Technology (IEMENTech), 2019.

[11] S. Venkataramani, S. T. Chakradhar, and K. Roy. “Computing approximately, and efficiently,” Proc.

Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015, pp.748-751.
[12] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," Proc. 18th IEEE European Test Symposium, 2013, pp.1-6.
[13] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, pp. 850-862, 2010.
[14] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy, "IMPACT: IM Precise Adders for Low-Power Approximate Computing," Proc. Int. Symp. Low Power Electronics and Design (ISLPED), pp. 1-3, 2011.
[15] W. Liu, L. Chen, C. Wang, M. O'Neill and F. Lombardi, "Design and analysis of floating-point adders", IEEE Trans. Computers, vol. 65, pp. 308-314, Jan. 2016.
[16] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Computers, vol. 63, pp. 1760-1771, Sep. 2013.
[17] A. Booth, "A signed binary multiplication technique," Quarterly J. Mechanics and Applied Mathematics, vol. 4, pp/236-240, June 1951.
[18] O. MacSorley, High-speed arithmetic in binary computers, Proc. IRE, vol. 49, pp. 67-91, 1961.
[19] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low error fixed-width modified Booth multiplier," IEEE Trans. VLSI Systems, vol. 12, no. 5, pp. 522-531, 2004.
[20] M. J. Schulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant," Proc. Workshop VLSI Signal Process. VI, 1993, pp. 388-396.
[21] E. J. King and E. E. Swartzlander Jr., "Data dependent truncated scheme for parallel multiplication," Proc. 31st Asilomar Conf. Signals, Circuits Syst., 1998, pp. 1178-1182.
[22] J.-P. Wang, S.-R. Kuang, and S.-C. Liang, "High-accuracy fixed-width modified Booth multipliers for lossy applications," IEEE Trans. VLSI Systems, vol. 19, no. 1, pp. 52-60, 2011.
[23] C.-Y. Li, Y.-H. Chen, T.-Y. Chang, and J.-N. Chen, "A probabilistic estimation bias circuit for fixed-width Booth multiplier and its DCT applications," IEEE Trans. Circuits and Systems II: Express Briefs, vol. 58, no. 4, pp. 215-219, 2011.
[24] Y.-H. Chen, C.-Y. Li, and T.-Y. Chang, "Area-effective and power efficient fixed-width Booth multipliers using generalized probabilistic estimation bias," IEEE J. Emerging and Selected Topics in Circuits and Systems, vol. 1, no. 3, pp. 277-288, 2011.
[25] Y.-H. Chen and T.-Y. Chang, "A high-accuracy adaptive conditional probability estimator for fixed-width Booth multipliers," IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 59, no. 3, pp. 594-603, 2012.
[26] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an under designed multiplier architecture," Proc. 24th Int. Conf. VLSI Design, 2011, pp. 346-351.
[27] C.-H. Lin and C. Lin, "High accuracy approximate multiplier with error correction," Proc. IEEE Int. Conf. Computer Design, 2013, pp. 33-38.
[28] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Computers, vol. 64, pp. 984-994, 2015.
[29] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," Proc. Design Automation and Test in Europe, 2014, pp. 95-98.
[30] G. Zervakis, S. Xydis, K. Tsoumanis, D. Soudris and K. Pekmestzi, "Hybrid Approximate Multiplier Architectures for Improved Power-Accuracy Trade-offs", in Proc. IEEE Int'l Symposium on Low Power Electronics and Design, 2015, pp. 79-84.
[31] S. Hashemi, R. Bahar and S. Reda, "DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications", in Proc. IEEE/ACM International Conference on Computer Design, 2015, pp. 418 - 425.
[32] H. Jiang, J. Han, F. Qiao, F. Lombardi, "Approximate radix-8 Booth multipliers for low-power and high-performance operation", IEEE Trans. Computers, vol. 65, pp. 2638 - 2644, Aug. 2016.
[33] W. Yeh and C. Jen, "High-speed Booth encoded parallel multiplier design," IEEE Trans. Comput., vol. 49, pp. 692-701, Jul. 2000.
[34] S. Kuang, J. Wang, and C. Guo, "Modified Booth multiplier with a regular partial product array," IEEE Trans. Circuits Syst. II: Express Briefs, vol. 56, pp. 404-408, May 2009.
[35] L. Qiang, C. Wang, W. Liu, F. Lombardi and J. Han, "Design and evaluation of an approximate Wallace-Booth multiplier", Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), May 2016, pp. 1974-1977.