Modified Booth Truncated Multipliers

Alok A. Katkar and James E. Stine
VLSI Computer Architecture, Arithmetic, and CAD Research Laboratory
Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, Illinois 60616, USA
{akatkar,jstine}@ece.iit.edu

ABSTRACT

Truncated multiplication provides an efficient method for reducing the power dissipation and area of rounded parallel multipliers in digital signal processing systems. With this technique, the products of parallel multipliers are rounded to a shorter word size and the least-significant columns of the multiplication matrix are not used. This technique provides significant savings in terms of power dissipation for unsigned multiplication. Although previous implementations involved unsigned and signed array and tree multipliers, this technique can be equally applied to multiplication using Booth-encoding. This paper presents the design and implementation of parallel and truncated multipliers that use Booth-encoding and compressors for signed multiplication. Initial estimates indicate that truncated parallel multipliers dissipate less power than standard parallel multipliers for operand sizes of 16 bits.

Categories and Subject Descriptors

B.2.4 [**Arithmetic and Logic Structures**]: High-Speed Arithmetic; B.5.1 [**Register-Transfer-Level Implementation**]: Design; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*algorithms implemented in hardware, VLSI*

General Terms

Algorithms, Performance, Design

Keywords

Arithmetic, VLSI

1. INTRODUCTION

Digital signal processing (DSP) is used in a variety of applications such as cellular telephones, radios, and video applications. According to the semiconductor industry association's economic forecast, the growth for DSP processors will continue to grow at an increasing rate [1]. However, as the demand for new, smaller, and faster processors appear, the need for processors which can handle fast arithmetic is apparent [2].

Most DSP computations involve the use of high-speed parallel multipliers to accumulate numbers, therefore, the design of fast and efficient multipliers is imperative [3]. In fact, approximately 8.72% of all instructions in typical scientific programs are multiplies [4]. Thus, hardware designers have recognized this and have devoted considerable silicon area to building high-speed multipliers [5]. However, multipliers are a major source of power dissipation in digital signal processors [6], [7]. Therefore, the design of parallel multipliers for DSP applications should be efficient while still being able to handle low-power applications.

There are a variety of different implementations for parallel multipliers. However, most algorithms involve a shift and an add technique where the multiplicand is conditionally added to the obtain the final result. Although there are many algorithms for accomplishing this, there is no reduction in the height of partial products that need to be summed to produce the final result. The Booth algorithm attempts to reduce the number of partial-products by recoding the multiplier so that groups of its bits select multiples of the multiplicand [8]. Although, Booth encoding can typically reduce 45% to 80% the number of operations in a typical DSP application [9], Booth-encoded multipliers typically dissipate more power than multipliers that are not Booth-encoded [10], [11].

Although there have been many techniques to reduce the power dissipation of parallel multipliers, one technique that is well-suited for digital signal processing systems is truncated multiplication [12], [13], [14]. In [15], truncated multiplication is shown to significantly reduce the power over standard parallel multiplier for different operand sizes. This paper examines further reductions in power dissipation that can be achieved through the use of truncated multiplication using Booth-encoding. Section 2 gives an overview of truncated multipliers, and Section 3 discusses the proposed method. Section 4 gives results using a AMI 0.6 μm standard cell library. Finally, Section 5 presents conclusions.

2. TRUNCATED MULTIPLIERS

Parallel multipliers are typically implemented as either carry-save array or tree multipliers [16]. In many computer systems, the $(n+m)$-bit products produced by parallel multipliers are rounded to r bits to avoid growth in word size. As presented in [17], truncated multiplication provides an efficient method for reducing the hardware requirements of rounded parallel multipliers. With truncated multiplication, only the $r+k$ most-significant columns of the multiplication matrix are used to compute the product. The error produced by omitting the $m+n-r-k$ least-significant columns and rounding the final result to r bits is estimated, and this estimate is added along with the $r+k$ most-significant columns to produce the rounded product. Although this leads to additional error in the rounded product, various techniques have been developed to help limit this error.

One method to compensate for truncation is Constant Correction Truncated (CCT) Multipliers [12]. In this method, a constant is added to columns $n+m-r-1$ to $n+m-r-k$ of the multiplication matrix. The constant helps compensate for the error introduced by omitting the $n+m-r-k$ least-significant columns (called reduction error), and the error due to rounding the product to r bits (called rounding error). The expected value of the sum of these error E_{total} is computed by assuming that each bit in A, B and P has an equal probability of being one or zero. Consequently, the expected value of the total error is the sum of expected reduction error and the expected rounding error as

$$\begin{aligned} E_{total} &= E_{reduction} + E_{rounding} \\ E_{total} &= \frac{1}{4}\sum_{q=0}^{S-k-1}(q+1)\cdot 2^{-m-n+q} + \frac{1}{2}\cdot\sum_{z=S-k}^{S-1}2^{-m-n+z} \end{aligned}$$

where $S = m+n-r$ [14]. The constant C_{total} is obtained by rounding E_{total} to $r+k$ fractional bits, such that

$$C_{total} = -\frac{round(2^{r+k}\cdot E_{total})}{2^{r+k}}$$

where $round(x)$ indicates that x is rounded to the nearest integer. Although the value of k can be chosen to limit the maximum absolute error to a specific precision, this paper assumes the maximum absolute error is limited to one unit in the last place (i.e., 2^{-r}).

Another method to compensate for the truncation is using the Variable Correction Truncated (VCT) Multiplier [13]. With this type of multiplier, the values of the partial product bits in column $m+n-r-k-1$ are used to estimate the error due to leaving off the $m+n-r-k$ least-significant columns. This is accomplished by adding the partial products bits in column $m+n-r-k-1$ to column $m+n-r-k$. To compensate for the rounding error, a constant is added to columns $m+n-r-2$ to $m+n-r-k$ of the multiplication matrix. The value for this constant is

$$C_{total} = 2^{-S-1}(1-2^{-k+1})$$

which corresponds to the expected value of the rounding error truncated to $r+k$ bits.

Another method, called a Hybrid Correction Truncated (HCT) Multiplier, uses both constant and variable correction techniques to reduce the overall error [14]. In order to implement a HCT multiplier, a new parameter is introduced, p, that represents the percentage of variable correction to use for the correction. This percentage is utilized to choose the number of partial products from column $m+n-r-k-1$ to be used to add into column $m+n-r-k$. The calculation of the number of variable correction bits is the following utilizing the number of bits used in the variable correction method, N_{HCT}

$$N_{HCT} = floor(N_{VCT}\times p)$$

Similar to both the CCT and the VCT multipliers, a HCT multiplier uses a correction constant to compensate for the error. However, since the correction constant will be based on a smaller number bits than a VCT multiplier, the correction constant is modified as follows

$$C_{VCT'} = 2^{-r-k-2}\cdot N_{HCT}$$

This produces a new correction constant based on the difference between the new variable correction constant and the constant correction constant.

$$C_{total} = \frac{round((C_{CCT} - C_{VCT'})\cdot 2^{r+k})}{2^{r+k}}$$

Most DSP and embedded systems involve the use of signed and unsigned binary numbers. Therefore, multiplication requires some mechanism to compute two's complement multiplication. A common implementation for two's complement multipliers is to use the basic mathematical equation for multiplication and use algebra to formalize a structure. The most popular of these implementation are called Baugh-Wooley multipliers [18]. Each structure utilizes the same tree structure, however, several columns require complementation as well as adding compensation constants for sign-extension.

3. BOOTH-ENCODED TRUNCATED MULTIPLIER IMPLEMENTATIONS

Booth's algorithms can also be used to handle negative binary representations by utilizing a Signed-Digit (SD) number system [8]. Signed-Digit (SD) number systems allow both positive and negative digits within the number For example, the value 5 in radix 10 can be represented in binary as $011\overline{1}$. The main idea behind Booth's algorithm is to eliminate long strings of ones by changing or recoding the bit representation.

A disadvantage to Booth's algorithm is that the algorithm is inefficient when zeroes and ones are interspersed randomly within a given input. This can be improved by examining three bits of the multiplier at a time instead of two [19], [20]. This reduces the number of partial products and is called radix 4 *Modified Booth's Algorithm* [19]. Instead of utilizing a controlled adder or subtractor, a multiplexor is utilized to choose a 0 or a multiple of the multiplicand. The output of the multiplexor can then be fed into an adder tree. Radix 4 Booth encoded digits have values from $\{\overline{2},\overline{1},0,1,2\}$.

Radix 4 digits can be obtained directly from two's complement values. In this case, groups of three bits are examined, with one bit overlap between groups. The general idea is to reduce the number of partial products by grouping the bits of the multiplier into pairs and selecting the partial products from the set $\{0, M, 2M\}$ where M is the multiplicand. Because this implementation typically outputs 2 bits for decoding the bits, it sometimes is called a *Booth* 2 multiplier. Each partial product is shifted two bit positions with respect to the previous row. In general, there will be $\lfloor\frac{n+2}{2}\rfloor$ partial product rows where n is the operand length [19].

One of the interesting components to Booth multipliers is that it employs sign extension to make sure that the sign is propagated appropriately. Since SD notation is utilized, the sign of the most-significant bit must be sign extended to allow the proper result to occur. Figure 1 shows an 8-bit by 8-bit signed radix-4 modified Booth multiplier dot diagram illustrating how sign extension is utilized within the partial product matrix where each partial product is represented as a dot. Each partial product is 9 bits since numbers as large as two times the multiplicand can be handled.

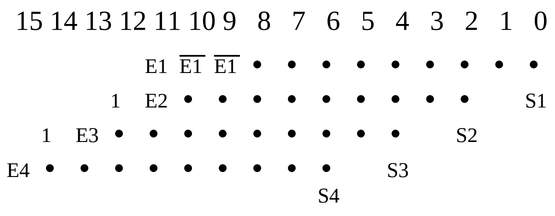

Figure 1: Dot Diagram of Signed Booth 2 Multiplier with $n = m = 8$.

Sign extension is shown in Figure 1 by two values SX and EX where X represents the row in the partial product matrix. The sign extension is handled by S which indicates whether the recoded Booth value is negative or positive. Each S term is then added to the unit in the last place (ulp) for proper two's complement conversion. It is important to note that sign extension can clear out the bits in the partial product matrix when the multiplicand is negative and the multiplier selects a negative value. Consequently, a simple EXCLUSIVE-NOR between the sign bit of the multiplicand and the high order bit of the partial product selection bits in the multiplier generates the one to clear the leading ones correctly as indicated by the E in Figure 1 [19]. Moreover, since the most-significant bit of the input operands is the sign bit, Booth 2 multipliers reduce the number of partial product rows to $\lfloor \frac{n}{2} \rfloor$.

Truncated Booth multipliers can utilize the same methods discussed in Section 2. However, since truncation can remove the bits from the two's complement conversion, the constant has to be modified. For example, Figure 2 shows the block diagram of a truncated 8 by 8 Booth 2 multiplier that uses the constant correction method (i.e. CCT) with $r = 9$, $k = 2$. A truncated multiplier removes $S1$, $S2$, and $S3$ from the partial product matrix causing the total error to be altered. Therefore, the total error for a CCT Booth multiplier is modified by adding an extra term, E_{ulp} to account for removing these bits:

$$\begin{aligned}
E_{total} &= E_{reduction} + E_{rounding} + E_{ulp} \\
E_{total} &= \frac{1}{4} \sum_{q=0}^{S-k-1} \lceil \frac{q+1}{2} \rceil \cdot 2^{-m-n+q} + \frac{1}{2} \cdot \sum_{z=S-k}^{S-1} 2^{-m-n+z} \\
& \quad \frac{3}{8} \cdot \sum_{q=0}^{min\{\lfloor \frac{S-k-1}{2} \rfloor, \lfloor \frac{n}{2} - 1 \rfloor\}} 2^{-m-n+q \cdot 2}
\end{aligned}$$

The value of 3/8 is the probability mass function that Booth recoding requires sign-extension. The constant that provides the correction, C_{total}, is obtained by rounding E_{total} to $r+k$ fractional bits as before. For example, the correction constant shown in Figure 2, indicated by the circled dot, is $C_{total} = 0.5 \times 2^{-9}$.

Tree multipliers are excellent structures for summing partial products. Unfortunately, tree multipliers can have irregular interconnections due to non regularity within the reduction matrix. Compressors are sometimes utilized instead of counters to make interconnections more regular. A

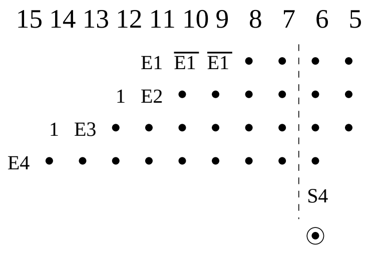

Figure 2: Dot Diagram of Truncated Signed Booth 2 Multiplier with $n = m = 8$, $r = 9$, and $k = 2$.

(p, q) compressor takes p inputs and produces q outputs. In addition, it takes k carry-in bits and produces k carry-out bits. The $(4, 2)$ compressor, which is probably the most common type of compressor, takes 4 input bits and 1 carry-in bit, and produces 2 output bits and 1 carry-out bit [21]. The fundamental difference between compressors and traditional multi-operand adders is that the carry-out bit does not depend on the carry-in bit.

Compressors must be interconnected properly so that it can compute the correct values for the final carry-propagate adder. Unfortunately, trees create three dimensional structures and interconnections between each row for a given partial product matrix can be difficult to manage. Fortunately, (4:2) compressors can be placed to reduce the overall height so it linear and more manageable. The overall idea is to partition each column into groups of four sub arrays and use (4:2) compressor to combine the sub array. Compressor trees organize each row into groups of four and connect each output in parallel to form a tree structure. Since the organization of compressors is more regular, they lend themselves to efficient implementations at the custom-level, thereby, being popular choices in many datapaths.

4. RESULTS

Several designs were implemented to examine the impact upon a typical Application Specific Integrated Circuit (ASIC) designs. A sub-micron standard-cell libraries is selected to examine the impact of these designs [22]. Synthesis is performed with Synopsys Design Compiler and Cadence Silicon Ensemble is used in script mode and performs both placement and routing. Verilog netlists are generated for several truncated and non-truncated multipliers.

Layouts are generated for each multiplier and parasitically extracted to obtain accurate numbers for area, delay, and speed in AMI 0.6 μm technology. Delay numbers are obtained utilizing Synopsys' Pathmill. Pathmill is a cell-based static timing tool that utilizes netlists to achieve accurate delay estimates. Power dissipation is obtained using Synopsys' Powermill and simulated for $20,000$ ns. The stimuli to the simulator is pseudo-random, time-based vectors. Each simulator accepts transistor level netlists along with parasitic resistor and capacitor extraction.

Table 1 gives power, delay and area estimates for truncated Booth 2 multipliers with operand sizes of 16 bits. Again, the values for k are chosen to limit the maximum absolute error to one unit in the last place. Compared to non-truncated Booth 2 multipliers, truncated multipliers dissipate between 19 and 27 percent less power, and has between

Truncation Method	k	p	Power (mW)	Delay (ns)	Area (mm^2)
-	-	-	273.12	18.70	0.837
CCT	4	-	199.41	17.20	0.689
VCT	3	-	220.61	17.47	0.639
HCT	3	0.7	205.14	17.46	0.625

Table 1: Post-layout Estimates for $n = m = r = 16$ Signed Booth 2 Truncated and Non-Truncated Multipliers.

18 and 25 percent less area. The worst-case delay is not affected because the critical path is not dramatically altered. Similar designs can be also be utilized for higher-radix and unsigned Booth multipliers.

5. SUMMARY

For multimedia and digital signal processing application that do not require correctly rounded multiplication, truncated multipliers offer a significant hardware savings while introducing a small amount of additional error. Simulations indicate that for applications where correct rounding of the result is not needed, truncated multipliers have significant savings in terms of area, delay, and power.

This paper examines modified Booth-encoded multipliers utilized with compressors. Power, delay, and area estimates are made to compare standard parallel tree multipliers against truncated multipliers. The value of k is chosen to limit the maximum absolute error to one unit in the last place, however, the correction constant is altered to handle additional error introduced during sign extension. Other methods for reducing power dissipation can be applied to truncated Booth multipliers to further improve their power dissipation such as canonical operand encodings and optimal (4:2) compressor structures.

6. REFERENCES

[1] ITRS, "International technology roadmap for semiconductors," tech. rep., ITRS, 2003.

[2] G.-K. Ma and F. J. Taylor, "Multiplier Policies for Digital Signal Processing," *IEEE ASSP Magazine*, vol. 7, no. 1, pp. 6–19, 1990.

[3] Y. N. Chang, J. H. Satayanarayana, and K. K. Parhi, "Systematic design of high-speed and low-power digit-serial multipliers," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 12, pp. 1585–1596, 1998.

[4] S. F. Oberman, H. Al-Twaijry, and M. J. Flynn, "The SNAP Project: Design of Floating-Point Arithmetic Units," in *Proceedings of the 13th Symposium on Computer Arithmetic*, pp. 156–165, 1997.

[5] H. Al-Twaijry and M. J. Flynn, "Multipliers and Datapaths," Tech. Rep. CSL-TR94-654, Stanford University, 1994.

[6] S. S. Mahant-Shetti, P. T. Balsara, and C. Lemonds, "High performance low power array multiplier using temporal tiling," *IEEE Transactions on VLSI*, vol. 7, no. 1, pp. 121–124, 1999.

[7] C. J. Nicol and P. Larsson, "Low power multiplication for FIR filters," in *Proceedings of the 1997 International Symposium on Low Power Electronics and Design*, pp. 76–79, 1997.

[8] A. D. Booth, "A signed binary multiplication technique," *Q. J. Mech. Appl. Math.*, vol. 4, pp. 236–240, 1951.

[9] K. Muhammad and K. Roy, "On complexity reduction of FIR digital filters using constrained least squares solution," in *Proceedings of the 1997 International Conference on Computer Design*, pp. 196–201, 1997.

[10] T. K. Callaway and E. E. Swartzlander, Jr., "Power-delay characteristics of CMOS multipliers," in *Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, pp. 26–32, 1997.

[11] J. H. Satyanarayana and K. K. Parhi, "A theoretical approach to estimation of bounds on power consumption in digital multipliers," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 6, pp. 473–481, 1997.

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

[13] 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.

[14] J. E. Stine and O. M. Duverne, "Variations on truncated multiplication," in *Euromicro Symposium on Digital System Design*, pp. 112–119, 2003.

[15] M. J. Schulte, J. G. Hansen, and J. E. Stine, "Reduced power dissipation through truncated multiplication," in *IEEE Alessandro Volta Memorial International Workshop on Power Design*, pp. 61–69, 1998.

[16] K. Bickerstaff, M. J. Schulte, and E. E. Swartzlander, Jr., "Parallel Reduced Area Multipliers," *Journal of VLSI Signal Processing*, vol. 9, pp. 181–192, April 1995.

[17] Y. C. Lim, "Single-precision multiplier with reduced circuit complexity for signal processing applications," *IEEE Transactions on Computers*, vol. 41, no. 10, pp. 1333–1336, 1992.

[18] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Transactions on Computers*, vol. C-22, pp. 1045–1047, 1973.

[19] G. W. Bewick, *Fast multiplication: algorithms and implementation.* PhD thesis, Stanford University, 1994.

[20] O. L. MacSorley, "High-Speed Arithmetic in Binary Computers," *IRE Proceedings*, vol. 49, pp. 67–91, 1961.

[21] A. Weinberger, "A 4:2 carry-save adder module," *IBM Technical Disclosure Bulletin*, vol. 23, no. 8, pp. 3811–3814, 1982.

[22] J. Grad and J. E. Stine, "A Standard Cell Library for Student Projects," in *International Conference on Microelectronic Systems Education*, pp. 98–99, IEEE Computer Society Press, 2003.