Reduced Power Dissipation Through Truncated Multiplication

Michael J. Schulte and James E. Stine
Electrical Engineering and Computer Science Department
Lehigh University
Bethlehem, PA 18015, USA

John G. Jansen
Lucent Technologies
Allentown, PA 18104, USA

Abstract

Reducing the power dissipation of parallel multipliers is important in the design of digital signal processing systems. In many of these systems, the products of parallel multipliers are rounded to avoid growth in word size. The power dissipation and area of rounded parallel multipliers can be significantly reduced by a technique known as truncated multiplication. With this technique, the least significant columns of the multiplication matrix are not used. Instead, the carries generated by these columns are estimated. This estimate is added with the most significant columns to produce the rounded product. This paper presents the design and implementation of parallel truncated multipliers. Simulations indicate that truncated parallel multipliers dissipate between 29 and 40 percent less power than standard parallel multipliers for operand sizes of 16 and 32 bits.

1: Introduction

High-speed parallel multipliers are fundamental building blocks in digital signal processing systems [1]. In many cases, parallel multipliers contribute significantly to the overall power dissipation of these systems [2]. As transistor counts, clock frequencies, and the desire for portability increase, so does the need for low-power parallel multipliers.

Parallel multipliers are typically implemented as either array multipliers [3], [4] or tree multipliers [5] - [7]. For both types of parallel multipliers, Booth-encoding can be employed to reduce the number of partial products [8], [9]. Estimates given in [10] - [12] indicate that array multipliers dissipate more power than tree multipliers and that Booth-encoded multipliers dissipate more power than multipliers that are not Booth-encoded.

Various techniques have been developed to reduce the power dissipation of parallel multipliers. Several of these techniques reduce power dissipation by eliminating spurious transitions [13] - [15]. Other research has focused on developing novel multiplier architectures and sign-extension techniques to reduce power dissipation and improve performance [16] - [19]. Another approach is to develop low-power 3-2 counters and 4-2 compressors, which are key components in parallel multipliers [20] - [22]. Although each of these techniques helps reduce power dissipation, further reductions will be needed for future digital signal processing systems.

This paper examines reductions in power dissipation that can be achieved through the use of truncated multiplication. Sections 2 gives an overview of truncated multipliers, and Section 3 discusses their implementation. Section 4 compares the power dissipation, delay, and area of truncated multipliers to standard parallel multipliers. Section 5 gives conclusions.

2: Truncated multipliers

In the discussion to follow, it is assumed that an unsigned n-bit multiplicand A is multiplied by an unsigned n-bit multiplier B to produce an unsigned $2n$-bit product P. For fractional numbers, the values for A, B, and P are

$$A = \sum_{i=0}^{n-1} a_i 2^{-n+i} \qquad B = \sum_{i=0}^{n-1} b_i 2^{-n+i} \qquad P = \sum_{i=0}^{2n-1} p_i 2^{-2n+i} \tag{1}$$

The multiplication matrix for $P = A \times B$ is shown in Figure 1a. For most high-speed applications, parallel multipliers are used to produce the product.

In many computer systems, the $2n$-bit products produced by the parallel multipliers are rounded to n bits to avoid growth in word size. As presented in [23] - [26], truncated multiplication provides an efficient method for reducing the hardware requirements of rounded parallel multipliers. With truncated multiplication, only the $n + k$ most significant columns of the multiplication matrix are used to compute the product. The error produced by omitting the $n - k$ least significant columns and rounding the final result to n bits is estimated, and this estimate is added with the $n + k$ 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.

With the Constant Correction Truncated Multiplier presented in [24], a constant is added to columns $n - 1$ to $n - k$ of the multiplication matrix. The constant helps compensate for the error introduced by omitting the $n - k$ least significant columns (called reduction error), and the error due to rounding the product to n 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. As described in [24], this gives

$$E_{total} = -0.25 \sum_{i=0}^{n-k-1} (i+1)2^{-2n+i} - 2^{-n-1}(1 - 2^{-k}) \tag{2}$$

The constant C_{total} is obtained by rounding $-E_{total}$ to $n + k$ fractional bits, such that

$$C_{total} = -\frac{round(2^{n+k} E_{total})}{2^{n+k}} \tag{3}$$

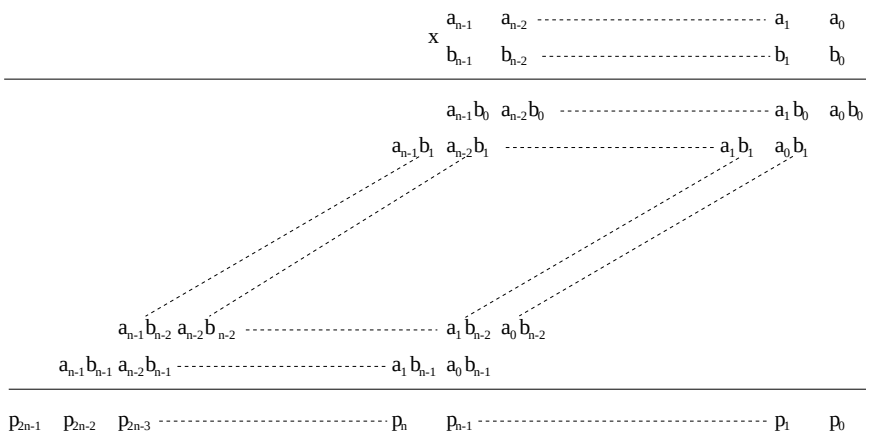
where $round(x)$ indicates that x is rounded to the nearest integer. The multiplication matrix for a truncated multiplier that uses this method is shown in Figure 1b.

In [26], the Variable Correction Truncated Multiplier is introduced. With this type of multiplier, the values of the partial product bits in column $n - k - 1$ are used to estimate the error due to leaving off the $n - k$ least significant columns. This is accomplished by adding the partial products bits in column $n - k - 1$ to column $n - k$. To compensate for the rounding error that occurs when truncating the products bits in columns $n - 1$ to $n - k$, a rounding constant, C_{round}, is added to the multiplication matrix. Since each product bit has an equal probability of being one or zero and the rounding constant cannot go beyond column $n - k$, the value used for C_{round} is

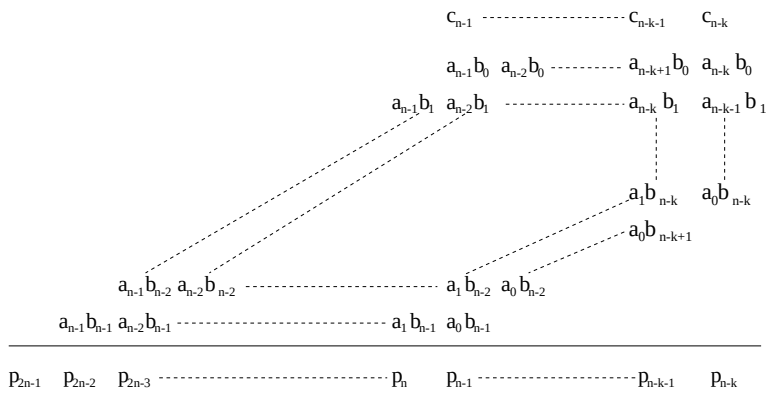
$$C_{round} = 2^{-n-1}(1 - 2^{-k+1}) \tag{4}$$

which corresponds to the additive inverse of the expected value of the rounding error, truncated after column $n - k$. The correction constant is added by putting ones in columns $n - 2$ to $n - k$, as shown in Figure 1c.

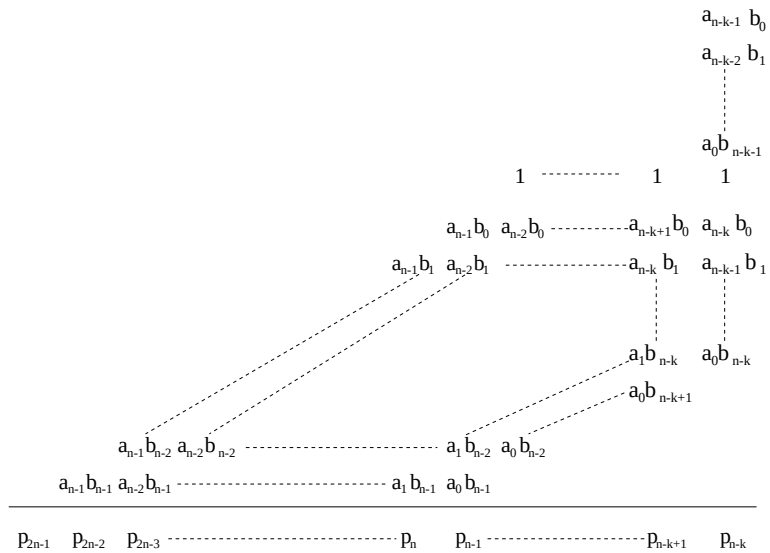
Compared to Constant Correction Truncated Multipliers, Variable Correction Truncated Multipliers have less average, mean square and maximum error for given values of n and k, but require more hardware. As discussed in [27], array multipliers can be implemented more efficiently as Variable Correction Truncated Multipliers and tree multipliers can be implemented more efficiently as Constant Correction Truncated Multipliers.



(a) Standard Multiplication Matrix



(b) Constant Correction Truncated Multiplication Matrix



(c) Variable Correction Truncated Multiplication Matrix

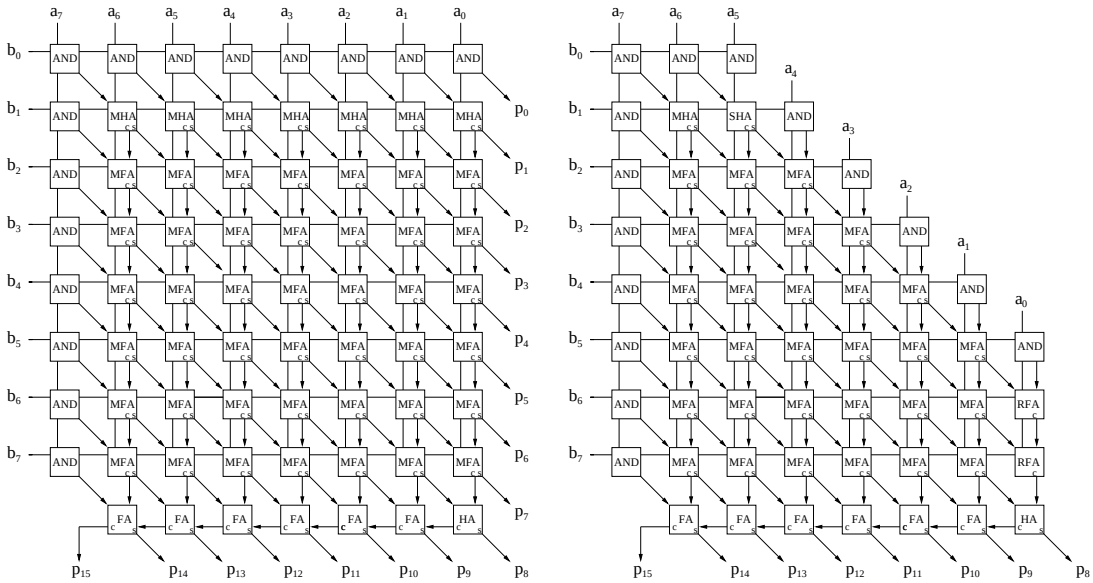
Figure 1. Multiplication Matrices

3: Truncated multiplier implementations

Figure 2a shows the block diagram of a standard 8 by 8 array multiplier. The cells along each diagonal in the array multiplier correspond to a column in the multiplication matrix. In this diagram, a modified half adder (MHA) cell consists of an AND gate and a half adder. The AND gate generates a partial product bit, and the half adder adds the generated partial product bit and a partial product bit from the previous row to produce a sum bit and a carry bit. Similarly, a modified full adder (MFA) consists of an AND gate, which generates a partial product bit, and a full adder which adds the partial product bit and the sum and carry bits from the previous row. The bottom row of adders produces the most significant half of the product. To improve performance, this row of adders is sometimes replaced by a fast n-bit carry-propagate adder. An n by n array multiplier requires n^2 AND gates, n half adders, and $n^2 - 2n$ full adders.

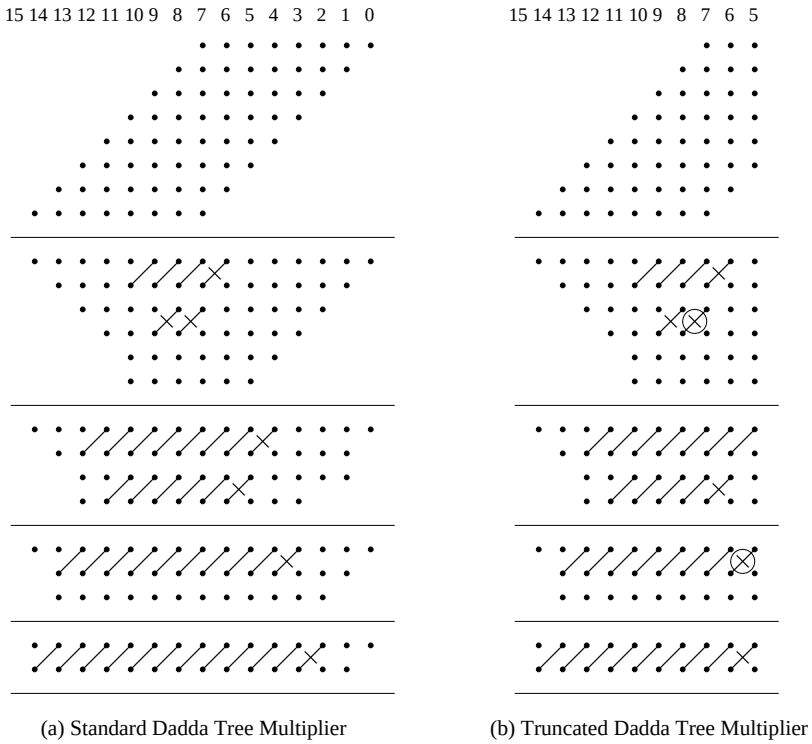
The Variable Correction Truncated Multiplication method provides an efficient method for reducing the power dissipation and hardware requirements of rounded array multipliers. With this method, the diagonals that produce the $t = n - k$ least significant product bits are eliminated. To compensate for this, the AND gates that generate the partial products for column $t - 1$ are used as inputs to the modified adders in column t. Since the k remaining modified full adders on the right-hand-side of the array do not need to produce product bits, they are replaced by modified reduced full adders (RFAs), which produce a carry, but do not produce a sum. To add the constant that corrects for rounding error, $k - 1$ of the MHAs in the second row of the array are changed to modified specialized half adders (SHAs). SHAs are equivalent to MFAs that have an input set to one [7]. Array multipliers that use this method require $t(t-1)/2$ fewer AND gates, $(t-1)(t-2)/2$ fewer full adders, and $(t-1)$ fewer half adders than standard array multipliers [26].

Figure 2b shows the block diagram of a 8 by 8 array multiplier that uses the Variable Correction Truncated Multiplication method. For this multiplier, $n = 8$, $k = 2$, and $t = 6$, which results in a hardware savings of 15 AND gates, 10 full adders, and 5 half adders. The two MFAs on the right-hand-side of the array are replaced by RFAs. The rounding correction constant $C_{round} = 0.25 \times 2^{-8}$, is added by changing one of the MHAs in the second row to a SHA. For this example, only one MHA is modified since $C_{round} = 0.25 \times 2^{-8}$ has a single '1'. This multiplier has a maximum absolute error of approximately 0.723×2^{-8}. In comparison, an 8 by 8 rounded multiplier has a maximum absolute error of 0.5×2^{-8}.



(a) Standard Array Multiplier (b) Truncated Array Multiplier

Figure 2. 8 by 8 Array Multipliers.



(a) Standard Dadda Tree Multiplier (b) Truncated Dadda Tree Multiplier

Figure 3. 8 by 8 Dadda Tree Multipliers.

With tree multipliers, the bits of the multiplicand and multiplier are ANDed to generate an n word by n bit partial product matrix. After this, half adders and full adders are used to reduce the partial product matrix to two rows, which are summed using a carry-propagate adder. Figure 3a shows the dot diagram of an 8 by 8 tree multiplier that uses Dadda's method of partial product reduction [6]. In this figure, each partial product is represented by a dot, the outputs of each full adder are represented by two dots connected by a plain diagonal line, and the outputs of a half adder are represented by two dots connected by crossed diagonal line. An n by n multiplier that uses Dadda's method of partial product reduction requires n^2 AND gates to generate the partial products, $n^2 - 4n + 3$ full adders and $n - 1$ half adders to reduce the partial products, and a $(2n - 2)$-bit carry-propagate adder to produce the product [7].

Tree multipliers can be efficiently implemented using the Constant Correction Truncated Multiplier method. The hardware saved with truncated Dadda tree multipliers is $t(t + 1)/2$ AND gates and $(t - 1)(t - 2)/2$ full adders. The number of half adders saved is between 1 and t, and depends on the values of n and k. The size of the carry-propagate adder is reduced by $t - 1$ bits, and the k least significant adders in the carry-propagate adder do not need to produce sum bits. To add the correction constant, m of the half adders are changed to specialized half adders, where m corresponds to the number of ones in C_{total}. Similar hardware savings can be achieved by multiplier trees that use other methods for reducing the partial product, such as Wallace tree multipliers [5] or multipliers that use compressors or higher order counters [28], [29], [30].

Figure 3b shows the dot diagram of an 8 by 8 truncated Dadda multiplier, which uses the Constant Correction Truncated Multiplication method [24]. For this multiplier, $n = 8$ and $k = 3$, so the $t = 5$ least significant columns of the dot diagram are eliminated. The correction constant $C_{total} = 0.625 \times 2^{-8}$ is added by changing the two circled half adders to specialized half adders. This multiplier has a maximum absolute error of approximately 0.754×2^{-8}. Compared to a standard 8 by 8 Dadda multiplier, this multiplier requires 15 fewer AND gates, 6 fewer full adders, 2 fewer half adders, and 4 fewer bits in the carry-propagate adder.

4: Power, delay, and area estimates

Previous research on truncated multipliers has focused on reducing their error and hardware requirements [23] - [26]. Reductions in power dissipation achieved by truncated multiplication, however, have not yet been explored. These reductions in power dissipation come as a direct consequence of the reductions in hardware and area obtained by truncated multipliers.

Power, delay, and area estimates were made to compare standard parallel multipliers and truncated parallel multipliers. For these estimates, the array multipliers use a ripple carry adder for the final addition, whereas the tree multipliers use a carry lookahead adder. The truncated array multipliers use the Variable Correction Truncated Multiplication method [26] and the truncated tree multipliers use the Constant Correction Truncated Multiplication method [24]. All multipliers were implemented using a 0.25 micron CMOS standard cell library, which uses four levels of metal. The nominal operating voltage for the library is 2.5 Volts at 25° C. The estimates given in this section were simulated with a worst-case condition of 2.3 Volts at 125° C.

Perl scripts were used to generate Module Compiler Language (MCL) code for each of the multipliers. MCL is a proprietary hardware description language in the Synopsys Module Compiler. The Module Compiler (MC) tool was then used to map the MCL code into the specific library. MC was also used to implement the final adders for the tree multipliers. The output of MC was a synthesizable verilog description, mapped to the targeted library. The verilog netlists were optimized for power using the PowerCompiler tool from Synopsys. The truncated multipliers were optimized first, with the constraints sets to minimize the power consumed, at the expense of timing and area. The standard multipliers were then constrained to meet the timing through the corresponding truncated multiplier, and then optimized for power. Layouts for the multipliers were generated by using the Apollo Place and Route tool from Avant!. The utilization factor and optimization iterations were held constant for all generated layouts.

Table 1 gives normalized, pre-layout delay, area, and power dissipation estimates for standard and truncated multipliers with operand sizes of 8, 16, and 32 bits. The ratios of the truncated multiplier estimates to the standard multiplier estimates are also given. The values for k are chosen to limit the maximum absolute error to one unit in the last place (i.e., 2^{-n}). Each pre-layout estimate is normalized by dividing it by the corresponding pre-layout estimate for a standard 16-bit array multiplier, which has a worst-case delay of 22.2 ns, an area of 5,317 grid units, and an average power dissipation of 15.8 mW. The pre-layout simulations used extracted cell description, which contained parasitic capacitors and diodes. The routing between the cells was assumed to be ideal (i.e., no routing capacitance).

Table 2 gives normalized, post-layout delay, area, and power dissipation estimates for standard and truncated multipliers with operand sizes of 16 and 32 bits. Each post-layout estimate is normalized by dividing it by the corresponding post-layout estimate for a standard 16-bit array multiplier, which has a worst-case delay of 20.7 ns, an area of 0.92 mm^2, and an average power dissipation of 15.5 mW. The post-layout simulation used extracted netlists for the entire design. The parasitics were extracted for the cells and the routing between the cells. The parasitic capacitors included coupling capacitors between signals. The normalized post-layout power estimates differed from the corresponding pre-layout estimates by less than 5%.

Based on the post-layout estimates, the 16-bit and 32-bit truncated array multipliers dissipate 29% and 40% less power and require 32% and 37% less area than equivalent standard array multipliers. The 16-bit and 32-bit truncated tree multipliers dissipate 31% and 36% percent less power and require 27% and 36% less area than equivalent standard tree multipliers. As expected, the reductions in area and power dissipation from truncated multiplication are fairly close. This is because the area and the power dissipation are proportional to the amount of hardware used to implement the multiplier. The delays for the truncated multipliers varied from 9% less than to 5% more than equivalent standard multipliers. The difference in delays is primarily due to tradeoffs made by the synthesis tool when optimizing for power. The simulations also indicate that tree multipliers have significantly less power dissipation and delay than array multipliers, yet require only a small amount of additional area.

Multiplier			Delay			Area			Power		
n	k	Type	Stan.	Trun.	Ratio	Stan.	Trun.	Ratio	Stan.	Trun.	Ratio
8	2	Array	0.52	0.54	1.04	0.23	0.18	0.77	0.21	0.18	0.88
8	3	Tree	0.43	0.44	1.02	0.25	0.21	0.86	0.18	0.16	0.86
16	3	Array	1.00	1.02	1.02	1.00	0.69	0.69	1.00	0.71	0.71
16	4	Tree	0.71	0.61	0.87	1.05	0.76	0.72	0.64	0.45	0.70
32	4	Array	1.98	1.91	0.97	4.09	2.58	0.63	4.01	2.36	0.59
32	5	Tree	0.90	0.95	1.05	4.20	2.71	0.64	1.87	1.19	0.64

Table 1. Normalized Pre-Layout Multiplier Estimates.

Multiplier			Delay			Area			Power		
n	k	Type	Stan.	Trun.	Ratio	Stan.	Trun.	Ratio	Stan.	Trun.	Ratio
16	3	Array	1.00	1.05	1.05	1.00	0.68	0.68	1.00	0.71	0.71
16	4	Tree	0.64	0.58	0.91	1.02	0.75	0.73	0.67	0.46	0.69
32	4	Array	2.04	2.00	0.98	3.93	2.47	0.63	3.99	2.47	0.60
32	5	Tree	0.92	0.88	0.95	4.07	2.60	0.64	1.93	1.23	0.64

Table 2. Normalized Post-Layout Multiplier Estimates.

The multiplier power dissipations were estimated using PowerMill, a dynamic simulator provided by the Epic Technology Group of Synopsys. The simulator accepts a transistor level netlist, along with parasitic resistors, capacitors, and diodes. The stimuli to the simulator were pseudo-random, time-based vectors. The 8-bit and 16-bit multipliers were simulated for 50,000 ns, and the 32-bit multipliers were simulated for 25,000 ns. The actual number of vectors used to simulate each multiplier is computed by dividing the simulation time by the delay of the multiplier.

The multiplier delays were estimated using the PrimeTime tool from Synopsys. PrimeTime is a cell-based static timing tool. The pre-layout numbers were generated using wire load models provided as part of the library. The post-layout numbers were generated by back-annotating the routing delays to PrimeTime.

5: Conclusions

Truncated multiplication provides an efficient method for reducing the power dissipation and area of rounded parallel multipliers. Post-layout simulations indicate that truncated parallel multipliers dissipate between 29 and 40 percent less power than standard parallel multipliers for operand sizes of 16 and 32 bits. As the operand size increases, the relative reduction in power dissipation and area also increases. The techniques presented in this paper can also be applied to two's complement multipliers, Booth-encoded multipliers, and multipliers that use higher-order counters and compressors. Other methods for reducing power dissipation can be applied to truncated multipliers to further improve their power dissipation.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. MIP-9703421. This research is also supported by a grant from Lucent Technologies and the Pennsylvania Infrastructure Technology Alliance under Project No. AMD-003.

References

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

[2] 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.

[3] S. D. Peraris, "A 40 ns 17-bit Array Multiplier," *IEEE Transactions on Computers*, vol. 20, pp. 442–447, 1971.

[4] G. W. McIver, R. W. Miller, and T. G. O'Shaughnessey, "A Monolithic 16 by 16 Digital Multiplier," *IEEE International Solid-State Circuits Digest of Technical Papers*, pp. 231–233, 1974.

[5] C. S. Wallace, "Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, pp. 14–17, 1964.

[6] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1965.

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

[8] A. D. Booth, "A Signed Binary Multiplication Technique," *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, pp. 236–240, 1991.

[9] H. Sam and A. Gupta, "A Generalized Multibit Recoding of Two's Complement Binary Numbers and Its Proof with Application in Multiplier Implementations," *IEEE Transactions on Computers*, vol. 39, no. 8, pp. 1006–1015, 1990.

[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] P. C. H. Meier, R. A. Rutenbar, and L. R. Carley, "Exploring Multiplier Architecture and Layout for Low Power," in *Proceedings of the IEEE 1996 Custom Integrated Circuits Conference*, pp. 513–516, 1996.

[12] 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.

[13] C. Lemonds and S. S. Shetti, "A Low Power 16 by 16 Multiplier Using Transition Reduction Circuitry," in *Proceedings of the International Workshop on Low Power Design*, pp. 139–142, 1994.

[14] G. E. Sobelman and D. L. Raatz, "Low-Power Multiplier Design Using Delayed Evaluation," in *Proceedings of the International Symposium on Circuits and Systems*, vol. 3, pp. 1564–1567, 1995.

[15] T. Sakuta, W. Lee., and P. T. Balsara, "Delay Balanced Multipliers for Low Power/Low Voltage DSP Core," in *1995 IEEE Symposium on Low Power Electronics*, vol. 4, pp. 36–37, 1995.

[16] J. Iwamura *et al.*, "A High Speed and Low Power CMOS/SOS Multiplier-Accumulator," *Microelectronics Journal*, vol. 14, no. 6, pp. 49–57, 1983.

[17] E. de Angel and E. E. Swartzlander, Jr., "Low Power Parallel Multipliers," in *VLSI Signal Processing, IX.*, pp. 199–208, 1997.

[18] E. de Angel, "Low Power Digital Multipliers," in *Application Specific Processors.* (E. E. Swartzlander, Jr., ed.), pp. 91–114, Kluwer Academic Publishers, 1997.

[19] E. Abu-Shama, M. B. Maaz, and M. A. Bayoumi, "A Fast and Low Power Multiplier Architecture," in *Proceedings of the 39th Midwest Symposium on Circuits and Systems*, pp. 26–32, 1997.

[20] S.-F. Hsiao, M.-R. Jiang, and J.-S. Yeh, "Design of High-Speed Low-Power 3-2 Counter and 4-2 Compressor for Fast Multipliers," *Electronics Letters*, vol. 34, no. 4, pp. 341–343, 1998.

[21] I. S. Abu-Khater, A. Bellaouar, and M. I. Elmasry, "Circuit Techniques for CMOS Low-Power High-Performance Multipliers," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 10, pp. 1535–1546, 1996.

[22] S. J. Jou, C. Y. C. E. C. Yang., and C. C. Su, "A Pipelined Multiplier-Accumulator Using a High-speed, Low-Power Static and Dynamic Full Adder Design," *IEEE J. Solid-State Circuits*, vol. 32, no. 1, pp. 114–118, 1997.

[23] 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.

[24] M. J. Schulte and E. E. Swartzlander, Jr., "Truncated Multiplication with Correction Constant," in *VLSI Signal Processing, VI*, pp. 388–396, 1993.

[25] S. S. Kidambi, F. El-Guibaly, and A. Antoniou, "Area-Efficient Multipliers for Digital Signal Processing Applications," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 2, pp. 90–95, 1996.

[26] 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.

[27] M. J. Schulte, J. G. Jansen, and J. E. Stine., "Implementing Truncated Multipliers," tech. rep., Lehigh University, 1998.

[28] E. E. Swartzlander, Jr., "Parallel Counters," *IEEE Transactions on Computers*, vol. 32, pp. 1021–1024, 1973.

[29] M. Mehta, V. Parmar, and E. E. Swartzlander, Jr., "High-speed Multiplier Design Using Multi-Input Counter and Compressor Circuits," in *Proceedings of the 10th International Symposium Computer Arithmetic*, pp. 43–50, 1991.

[30] P. J. Song and G. D. Micheli, "Circuit and Architecture Trade-offs for High-Speed Multiplication," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 9, pp. 1184–1198, 1991.