Analysis of Column Compression Multipliers

K'Andrea C. Bickerstaff and Earl E. Swartzlander, Jr.
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712, USA

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

Abstract

Column compression multipliers are frequently used in high-performance computer systems due to their short worst case delay. This paper examines the area, delay, and power characteristics of Dadda and Wallace column compression multipliers in deep submircon technology. Our analysis shows that Wallace multipliers have slightly more area and approximately the same worst case delay as Dadda multipliers. It also shows the importance of considering parasitic capacitances when determining the delay of column compression multipliers, since parasitics can increase the delay of the multiplier by over 60%. As multiplier size increases, the ratio of power to area also increases, due to longer interconnect lines and increased glitching.

1. Introduction

In 1964, Wallace [1] defined a column compression architecture for fast multiplication. The multiplication process begins with the generation of all partial products in parallel using an AND gate array. Next, the partial products are reduced to two numbers by the application of (3,2) and (2,2) counters. Finally, the two numbers are summed using a fast carry-propagate adder to form the final product. The column compression architecture offers total delays which are proportional to the logarithm of the operand word length. Therefore, column compression multipliers are faster than array multipliers, whose delay grows linearly with operand size.

In 1965, Dadda [2] refined Wallace's method by proposing a unique placement strategy for the reduction stage counters. Using Dadda's technique, the number of (3,2) and (2,2) counters is minimized, but the fast carry-propagate adder is larger. The Wallace and Dadda multipliers achieve optimal speed for both pipelined and non-pipelined implementations.

When first introduced, column compression multipliers were difficult to design, had high interconnect overhead, and could not be efficiently pipelined. However, advances in computer-aided design and VSLI process technology have helped alleviate these problems. In the literature, reports of fast CMOS implementations [3, 4, 5], alternative design schemes [6, 7, 8, 9, 10, 11] and strategies for pipelining [12, 13] column compression multipliers have appeared with increasing frequency.

Missing from the literature is the analysis of column compression multipliers which details the trends and magnitude of area, delay, and power for deep submicron design. Such analysis will facilitate the design community's ability to estimate performance and select appropriate multiplier architectures. The practical examination of the multipliers using today's typical process technologies will also establish new targets for research and improvement.

This paper examines the area, delay, and power characteristics of Dadda and Wallace column compression multipliers in deep submicron technology. In Section 2, the Dadda column compression scheme is detailed. Section 3 describes the Wallace column compression scheme. Section 4 reports the simulation results for 16 multiplier designs. Finally, Section 5 summarizes the analysis. Throughout the paper, it is assumed that the number of bits in the multiplier is equal to the number of bits in the multiplicand.

0-7695-1150-3/01 $10.00 © 2001 IEEE

2. Dadda multipliers

For the reduction of the N by N partial product matrix, Dadda proposes a sequence of matrix heights that are determined by working back from the final two-row matrix. In order to implement the minimum number of reduction stages, the height of each intermediate matrix is limited to the largest integer that is no more than 1.5 times the height of its successor. Table 1 lists the number of reduction stages, S, needed for an N-bit multiplier. For example, a 32 by 32 Dadda multiplier uses 8 reduction stages, with matrix heights of 28, 19, 13, 9, 6, 4, 3, and finally 2.

Table 1. Number of reduction stages.

Bits in Multiplier (N)	Number of Stages (S)
2	0
3	1
4	2
$5-6$	3
$7-9$	4
$10-13$	5
$14-19$	6
$20-28$	7
$29-42$	8
$43-63$	9
$64-94$	10

The reduction process for a Dadda multiplier is developed using the following recursive algorithm [14]:

1. Let $d_1 = 2$ and $d_{j+1} = \lfloor 1.5 \cdot d_j \rfloor$, where d_j is the matrix height for the j^{th} stage from the end. Find the smallest j such that at least one column of the original partial product matrix has more than d_j bits.

2. In the j^{th} stage from the end, employ (3,2) and (2,2) counters to obtain a reduced matrix with no more than d_j bits in any column.

3. Let $j = j - 1$ and repeat step 2 until a matrix with only two rows is generated.

Habibi and Wintz [15] established that Dadda's strategy for column compression is optimum in that it uses the minimum number of (3,2) counters.

For Dadda multipliers there are N^2 bits in the original partial product matrix and $4 \cdot N - 3$ bits in the final two row matrix. Since each (3,2) counter takes three inputs and produces two outputs, the number of bits in the matrix is reduced by one with each applied (3,2) counter. Therefore, the total number of (3,2) counters is

$$\#(3,2) = N^2 - 4 \cdot N + 3$$

The length of the carry-propagate adder is

$$\text{CPA length} = 2 \cdot N - 2$$

The number of (2,2) counters used in Dadda's reduction method is determined by the following observations:

1. In the first stage, (2,2) counters are applied in columns d_S to N, where d_i is the number of bits specified by the Dadda sequence for the i^{th} reduction stage from the end.

2. In the i^{th} reduction stage from the end (2,2) counters are used in columns d_i to $d_{i+1} - 1$.

By observations (1) and (2), the entire reduction process requires one (2,2) counter in each column for columns 2 to N. Therefore, the total number of (2,2) counters equals $N - 1$.

The dot diagram for a 12 by 12 Dadda multiplier is shown in Figure 1. Dot diagrams are a useful tool for depicting the placement of (3,2) and (2,2) counters in parallel multipliers. Each partial product bit is represented by a dot. The outputs of each (3,2) counter are represented as two dots connected by a plain diagonal line. The outputs of each (2,2) counter are represented as two dots connected by a crossed diagonal line. The 12 by 12 Dadda multiplier takes five reduction stages, with matrix heights of 9, 6, 4, 3, and 2. The reduction uses 99 (3,2) counters, 11 (2,2) counters, and a 22-bit carry-propagate adder. The total delay for the generation of the final product is the sum of one AND gate delay, one (3,2) counter delay for each of the five reduction stages, and the delay through the final 22-bit carry-propagate adder. As noted in [8], the middle inputs to the carry-propagate adder arrive later, which effectively reduces the worst case delay of the carry-propagate adder.

3. Wallace multipliers

With Wallace tree multipliers, rows are grouped into sets of three during each reduction stage. Within each three row set, (3,2) counters reduce columns with three bits to two bits and (2,2) counters reduce columns with only two bits. Rows that are not part of a three row set are transfered to the next stage without modification. The height of the matrix in the j^{th} reduction stage, w_j, is defined by the following recursive equations:

$$\begin{aligned} w_0 &= N \\ w_{j+1} &= 2 \cdot \lfloor w_j/3 \rfloor + w_j \bmod 3 \end{aligned}$$

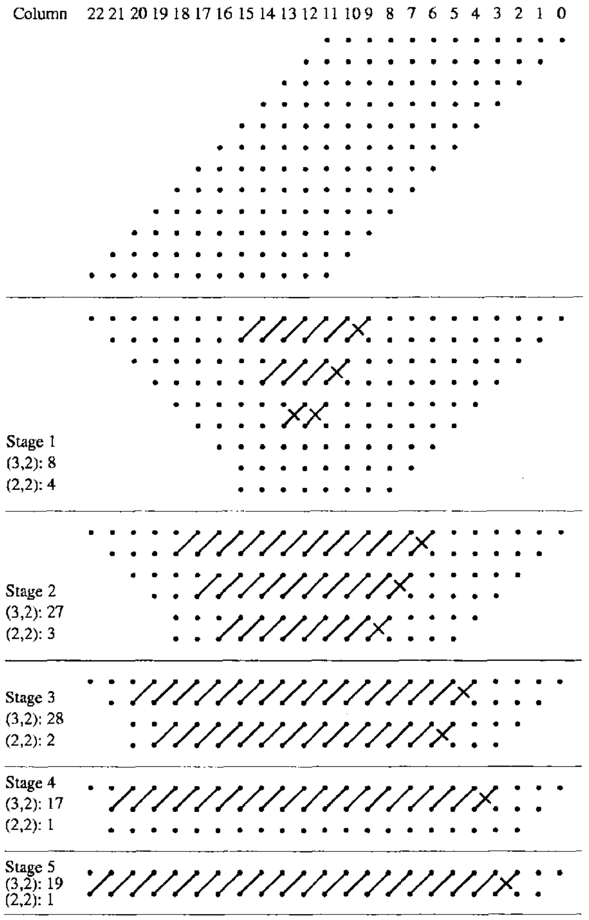

Figure 1. Dot diagram of a 12 by 12 Dadda multiplier.

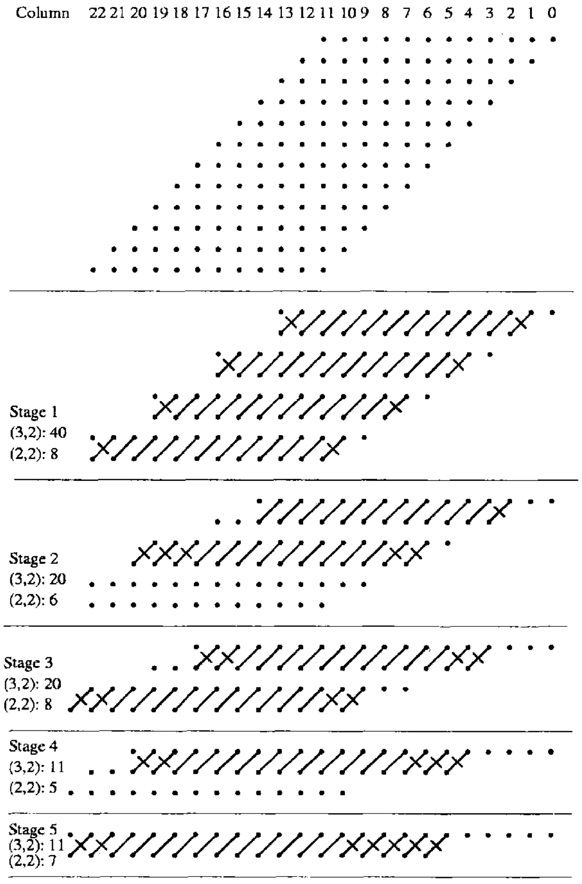

Figure 2. Dot diagram of a 12 by 12 Wallace multiplier.

For example, a 32 by 32 Wallace multiplier uses eight reduction stages with matrix heights of 22, 15, 10, 7, 5, 4, 3, and 2.

The dot diagram for a 12 by 12 Wallace multiplier is shown in Figure 2. This multiplier takes five reduction stages, with matrix heights of 8, 6, 4, 3, and 2. In the second reduction stage, which has eight rows, the last two rows are left over and not reduced until the following stage. The entire reduction uses 102 (3,2) counters, 34 (2,2) counters, and an 18-bit carry-propagate adder. The total delay for the generation of the final product is the sum of one AND gate delay, one (3,2) counter delay for each of the five reduction stages, and the delay through the final 18-bit carry-propagate adder.

In general, the hardware required for Wallace tree multipliers depends on N and S. As presented in [7], the number of (3,2) counters and the word-length of the carry-propagate adder for $3 \leq N \leq 5$ are

$$\begin{aligned} \#(3,2) &= N^2 - 4 \cdot N + 3 + S \\ \text{CPA length} &= 2 \cdot N - 2 - S \end{aligned}$$

For $5 < N$, they are

$$\begin{aligned} \#(3,2) &= N^2 - 4 \cdot N + 2 + S \\ \text{CPA length} &= 2 \cdot N - 1 - S \end{aligned}$$

or

$$\begin{aligned} \#(3,2) &= N^2 - 4 \cdot N + 1 + S \\ \text{CPA length} &= 2 \cdot N - 1 - S \end{aligned}$$

depending on the number of bits in the final two row matrix. The number of (2,2) counters is at least N and often much greater than N. For example, a 32 by 32 Wallace multiplier requires 164 (2,2) counters.

Table 2 gives the component counts of 8 by 8, 16 by 16, 32 by 32, and 64 by 64 Dadda and Wallace multipliers. Compared to Dadda multipliers, Wallace multipliers require more (3,2) and (2,2) counters, but have a smaller carry-propagate adder. For both types of multipliers, the number of AND gates is N^2.

Table 2. Component counts of Dadda and Wallace multipliers.

Multiplier	# (3,2)	# (2,2)	CPA
8 by 8 Dadda	35	7	14
8 by 8 Wallace	38	15	11
16 by 16 Dadda	195	15	30
16 by 16 Wallace	200	54	25
32 by 32 Dadda	899	31	62
32 by 32 Wallace	906	164	55
64 by 64 Dadda	3843	63	126
64 by 64 Wallace	3850	459	117

4. Multiplier simulations

In total, sixteen multipliers designs were generated for simulation. The multipliers were designed using the standard cell libraries of two process technologies. The CMOS process technologies utilized reflect today's most fabricated transistor sizes and voltage configurations: 1) 0.25 micron, 2.5V and 2) 0.18 micron, 1.8V. All simulations were run at the "typical" process corner with the temperature set to 25°C.

All primary inputs and outputs of the multipliers were gated by D flip-flops. The inputs for large multipliers pose fanout problems, driving from the D flip-flops into the AND gate array. Therefore, it was necessary to buffer each input for every additional eight AND gates. For a 32-bit multiplicand, each bit of the multiplier is driven from the D flip-flop into four buffers, each driving eight AND gates.

The final fast carry propagate adder was implemented as a Carry Lookahead Adder (CLA) [16]. The lookahead logic blocks were organized to receive a maximum of four propagate (P) and generate (G) pairs from modified full adders. Lookahead logic blocks to support two and three pairs of propagate and generate signals were also available for the most significant P and G pairs, as needed.

Perl scripts were written to automatically generate Hspice and Verilog versions of the multipliers for a range of operand sizes and process technologies. In order to determine delay and power characteristics, the Hspice circuits were evaluated using the Synopsys tools, Timemill and PowerMill. The Verilog designs were imported into the Cadence Design System place and route tool, Silicon Ensemble. For the Dadda multipliers implemented in 0.25 micron CMOS technology, Arcadia was used to extract parasitic capacitances for back annotation into delay and power simulations.

Tables 3 and 4 give the area of Wallace and Dadda multipliers in 0.25 and 0.18 micron CMOS technology. The area measurements are performed after place and route and include interconnect area and unused space. For comparison purposes the percent increase in area of Wallace multipliers to Dadda multipliers is also reported. As shown in these tables, Wallace multipliers require from 4% to 7% more area than Dadda multipliers, for operand sizes from 8 to 64 bits. The increase in area of Wallace multipliers is primarily due to the large number of (2,2) counters they require. As the operand size becomes larger, the percent increase in area of the Wallace multipliers to the Dadda multipliers also grows.

Table 3. Area of Dadda and Wallace multipliers for 0.25 micron technology.

Multiplier	Area (μm^2)		Percent
N by N	Dadda	Wallace	Increase
8 by 8	19,252	20,087	4.3
16 by 16	67,133	70,703	5.3
32 by 32	243,088	257,673	6.0
64 by 64	973,399	1,038,617	6.7

Table 4. Area of Dadda and Wallace multipliers for 0.18 micron technology.

Multiplier	Area (μm^2)		Percent
N by N	Dadda	Wallace	Increase
8 by 8	9,017	9,377	4.0
16 by 16	31,980	33,574	5.0
32 by 32	121,313	129,132	6.4
64 by 64	472,973	506,081	7.0

Table 5 gives normalized area measurements of Dadda multipliers in 0.25 and 0.18 micron CMOS technology, where the area for each multiplier is normalized to the area of the 8 by 8 Dadda multiplier in 0.18 micron CMOS technology. The ratio of the areas in the two technologies is also given for each operand size. Based on the data in this table, doubling the operand size increases the total area by less than a factor of four. This occurs because although the number of AND gates and (3,2) counters increases quadratically with an increase in operand size, the number of (2,2) counters and the carry-propagate adder length increase linearly. Going from 0.25 to 0.18 micron CMOS technology decreases the area by about a factor of two, which is slightly more than $(0.25/0.18)^2$.

Table 5. Normalized area of Dadda multipliers for 0.25 and 0.18 micron technology.

Multiplier	Normalized Area		Ratio of
N by N	0.25 μm	0.18 μm	0.25 to 0.18
8 by 8	2.14	1.00	2.14
16 by 16	7.45	3.55	2.10
32 by 32	26.96	13.45	2.00
64 by 64	107.95	52.45	2.06

Table 6 gives average power dissipation and area estimates for Dadda multipliers in 0.25 micron CMOS technology. The ratio of power to area is also reported. The power estimates include back annotated parasitic capacitances and were determined by applying 10,000 pairs of randomly generated operands to each multiplier. Two important power characteristics illustrated by Table 6 are

1. Doubling the operand size increases the average power dissipation by more than a factor of four.

2. Increasing the operand size increases the power to area ratio.

These two characteristics occur because as the multiplier size increases the length of the interconnect lines and the average amount of switching per gate also increases. The increased switching is prevalent in the later reduction stages and the fast carry-propagate adder, where gate inputs and outputs may switch several times before settling to their final value.

Table 6. Average power and area of Dadda multipliers for 0.25 micron technology.

Multiplier N by N	Power (mW)	Area (μm^2)	Power/Area mW/mm^2
8 by 8	1.9	19,252	99
16 by 16	8.3	67,133	124
32 by 32	35.0	243,088	144

Tables 7 and 8 give the estimated worst case delay of Wallace and Dadda multipliers in 0.25 and 0.18 micron CMOS technology. These estimates are based on Timemill simulation results and do not include parasitic capacitances or the delay of the input and output D flip-flops. Based on the data in these table, Wallace and Dadda multipliers have approximately the same worst case delay. For the multipliers examined, the difference in worst case delay of multipliers with the same operand size is less than or equal to 0.1 ns.

Table 7. Delay estimates of Dadda and Wallace multipliers for 0.25 micron technology without parasitics.

Multiplier	Delay (ns)	
N by N	Dadda	Wallace
8 by 8	1.8	1.8
16 by 16	2.7	2.8
32 by 32	3.5	3.5
64 by 64	5.1	5.0

Table 8. Delay estimates of Dadda and Wallace multipliers for 0.18 micron technology without parasitics.

Multiplier	Delay (ns)	
N by N	Dadda	Wallace
8 by 8	1.6	1.6
16 by 16	1.9	2.0
32 by 32	2.4	2.4
64 by 64	2.9	2.8

Table 9 gives normalized delay estimates of Dadda multipliers in 0.25 and 0.18 micron CMOS technology, where the delay for each multiplier is normalized to the delay of the 8 by 8 Dadda multiplier in 0.18 micron CMOS technology. The ratio of the delays in the two technologies is also given for each operand size. Based on the data in this table, going from 0.25 to 0.18 micron technology decreases the delay by a factor of 1.1 to 1.8, with larger relative decreases realized for larger multiplier sizes.

Table 10 gives delay estimates for Dadda multipliers in 0.25 micron CMOS technology with and without back annotated parasitic capacitances. The percent increase in delay due to parasitics is also reported. Based on the data in this table, parasitic capacitances increase the the delay of the multiplier from 26 to 61 percent for multiplier sizes from 8 to 32 bits. As the operand size increases, the percent increase in delay due to parasitic capacitance decreases.

Since Dadda multipliers have delays proportional to the logarithm of their operand size, it is expected that

Table 9. Normalized delay estimates of Dadda multipliers for 0.25 and 0.18 micron technology without parasitics.

Multiplier	Normalized Delay		Ratio of
N by N	0.25 μm	0.18 μm	0.25 to 0.18
8 by 8	1.1	1.0	1.1
16 by 16	1.7	1.2	1.4
32 by 32	2.2	1.5	1.5
64 by 64	3.2	1.8	1.8

Table 10. Delay estimates of Dadda multipliers for 0.25 micron technology.

Multiplier	Delay (ns)		Percent
N by N	No Parasitics	Parasitics	Increase
8 by 8	1.8	2.9	61
16 by 16	2.7	3.6	33
32 by 32	3.5	4.4	26

doubling the operand size, N, will result in a constant increase in delay. That is

$$t_{delay} \approx \alpha + \beta \cdot \log_2(N)$$

For the data shown in Table 10, $\alpha = -0.70$ and $\beta = 0.85$ approximate the delays without parasitics to within 0.05 ns, while $\alpha = 0.60$ and $\beta = 0.75$ approximate the delays with parasitics to within 0.05 ns.

5. Summary

The area, delay, and power characteristics of Dadda and Wallace column compression multipliers have been presented using simulation data from 16 multipliers. Dominated by the counters in the reduction matrices, the area and power dissipated by column compression multipliers increase roughly quadratically as the operand word length increases. The ratios of power to area also increase with operand word length, due to longer interconnect lines and increased glitching. For each design, the simulated delay values reflect that the total delay is proportional to the logarithm of the operand word length. Back annotated delay simulations indicate that parasitic capacitances can increase the delay of column compression multipliers by over 60%. Our simulations also indicate that Wallace multipliers have from 4% to 7% more area than equivalent Dadda multipliers, and approximately the same worst case delay for operand sizes from 8 to 64 bits.

Acknowledgment

The authors thank Cirrus Logic, Inc. for the usage of design, simulation, and layout tools. In particular, the authors are grateful for the instruction and advice of layout specialists Mehran Jalaliani, Jim Fox, Si Q. Tran, and Jean C. Law.

References

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

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

[3] M. Nagamatsu, S. Tanaka, J. Mori, K. Hirano, T. Noguchi, and K. Hatanaka, "A 15-ns 32x32-b CMOS Multiplier with an Improved Parallel Structure," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 494–497, 1990.

[4] J. Mori, M. Nagamatsu, H. Hirano, S. Tanaka, M. Noda, Y. Toyoshima, K. Hashimoto, H. Hayashida, and K. Maeguchi, "A 10-ns 54x54-b Parallel Structured Full Array Multiplier with 0.6-um CMOS Technology," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 600–605, 1991.

[5] 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.

[6] K. C. Bickerstaff, M. J. Schulte, and E. E. Swartzlander, Jr., "Reduced Area Multipliers," in *Proceedings of the 1993 International Conference on Application Specific Array Processors*, pp. 478–489, IEEE, 1993.

[7] K. C. Bickerstaff, M. J. Schulte, and E. E. Swartzlander, Jr., "Parallel Reduced Area Multipliers," *Journal of VLSI Signal Processing*, vol. 9, pp. 181–191, 1995.

[8] V. G. Oklobdzija, "Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 3, pp. 292–301, 1995.

[9] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation for Fast Parallel Multipliers Using an Algorithmic Approach," *IEEE Transactions on Computers*, vol. 45, pp. 294–305, 1996.

[10] Z. Wang, G. A. Jullien, and W. C. Miller, "A New Design Technique for Column Compression Multipliers," *IEEE Transactions on Computers*, vol. 44, pp. 962–970, 1995.

[11] 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 on Computer Arithmetic*, pp. 43–50, 1991.

[12] P. R. Cappello and K. Steiglitz, "A VLSI Layout for a Pipelined Dadda Multiplier," *ACM Transactions on Computer Systems*, vol. 1, pp. 157–174, 1983.

[13] L. Breveglieri, L. Dadda, and V. Piuri, "Column Compression Pipelined Multipliers," in *Proceedings 1995 International Conference on Application Sepecific Array Processors*, pp. 93–103, 1995.

[14] E. E. Swartzlander, Jr., "Merged Arithmetic," *IEEE Transactions on Computers*, vol. C-29, pp. 946–950, 1980.

[15] A. Habibi and P. A. Wintz, "Fast Multipliers," *IEEE Transactions on Computers*, vol. C-19, pp. 153–157, 1970.

[16] E. E. Swartzlander, Jr., "Computer Arithmetic," in *Computer Science and Engineering Handbook* (A. B. Tucker, Jr., ed.), ch. 4, CRC Press, 1997.