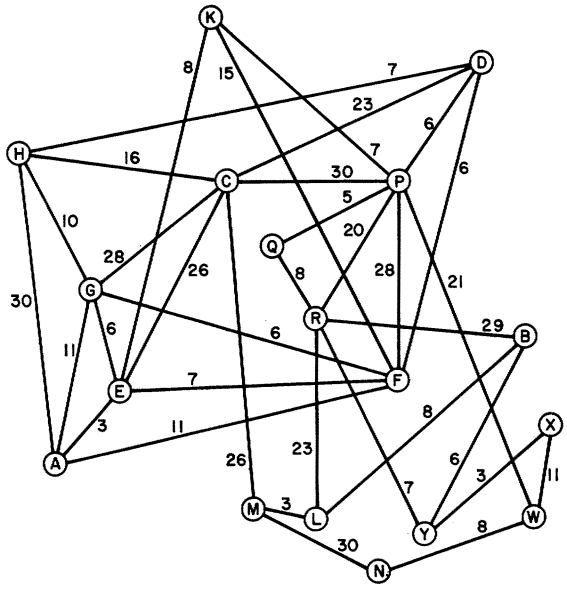

Fig. 4. A second example graph.

This modulo 3 technique does not work for graphs of unequal edge lengths, however, since the unequal lengths can put the labels "out of synchronization" (which can be verified by adding edge BH of length 1 to the first example problem). In the algorithm described here, the 2-bit vertex labels indicate if an edge has been traversed and, if so, in what direction, with synchronization being provided by edge length modification.

While this algorithm can be used with sequential computer, it has been designed expressly to exploit the highly parallel search and arithmetic properties of associative memory so as to gain improved execution speeds; and it is with associative memory that the saving in storage required for graph representation becomes important because of the higher cost per bit.

References

[1] E. F. Moore, "The shortest path through a maze," *Ann. Computation Laboratory of Harvard University*, vol. 30. Cambridge. Mass.: Harvard University Press, 1959, pp. 285–292.

[2] M. Pollack and W. Wiebenson, "Solutions of the shortest-route problem—a review," *Operations Research*, vol. 8, pp. 224–230, March–April 1960.

[3] C. Y. Lee, "An algorithm for path connections and its applications," *IEEE Trans. Electronic Computers*, vol. EC-10, pp. 346–365, September 1961.

[4] S. B. Akers, Jr., "A modification of Lee's path connection algorithm," *IEEE Trans. Electronic Computers (Short Notes)*, vol. EC-16, pp. 97–98, February 1967.

[5] M. H. Lewin, "Retrieval of ordered lists from content-addressed memory," *RCA Rev.*, pp. 215–229, June 1962.

[6] R. H. Fuller and G. Estrin, "Some applications for content-addressable memories," *1963 Fall Joint Computer Conf., AFIPS Proc.*, vol. 24. Baltimore, Md.: Spartan, 1963, pp. 495–508.

A New Algorithm for Inner Product

S. WINOGRAD, MEMBER, IEEE

***Abstract*—In this note we describe a new way of computing the inner product of two vectors. This method cuts down the number of multiplications required when we want to perform a large number of inner products on a smaller set of vectors. In particular, we obtain that the product of two $n\times n$ matrices can be performed using roughly $n^3/2$ multiplications instead of the n^3 multiplications which the regular method necessitates.**

***Index Terms*—Algorithm, inner product, matrix inversion, matrix multiplication, solution of linear equations.**

Manuscript received January 18, 1968; revised March 13, 1968.

The author was with IBM Research Center, Yorktown Heights, N. Y. 10598 He is now with the Dept. of Elec. Engrg. and Computer Sciences, University of California, Berkeley, Calif. 94720

I. The Algorithm

Let $x=(x_1, \cdots, x_n)$ and $y=(y_1, \cdots, y_n)$ be two vectors. For each vector we compute the number

$$\xi = \sum_{j=1}^{\lfloor n/2 \rfloor} x_{2j-1} \cdot x_{2j}$$

(where $\lfloor t \rfloor$ denotes the integer part of t), and

$$\eta = \sum_{j=1}^{\lfloor n/2 \rfloor} y_{2j-1} \cdot y_{2j}.$$

The inner product (x, y) is then given by

$$(x, y) = \begin{cases} \sum_{j=1}^{\lfloor n/2 \rfloor} (x_{2j-1} + y_{2j})(x_{2j} + y_{2j-1}) - \xi - \eta & \text{if } n \text{ is even} \\ \sum_{j=1}^{\lfloor n/2 \rfloor} (x_{2j-1} + y_{2j})(x_{2j} + y_{2j-1}) - \xi - \eta + x_n y_n & \text{if } n \text{ is odd.} \end{cases}$$

Consider the case where N n-dimensional vectors are given, and it is desired to perform T inner products involving these vectors. The total number of multiplications required is then $N\lfloor n/2 \rfloor + T\lfloor (n+1)2 \rfloor = Nn + (T-N)\lfloor (n+1)/2 \rfloor$ as compared with $Tn = Nn + (T-N)n$. So if $T > N$, the new method requires fewer multiplications than the regular method. The total number of additions required is $N(\lfloor n/2 \rfloor - 1) + T(n + \lfloor n/2 \rfloor + 1)$, while the regular method requires only $T(n-1)$ additions. If $T >> N$, then the total number of operations in the new method is about the same as the total number of operations in the regular method; therefore, the new method is faster when the time required to multiply is longer than the time required to add.

II. Applications

A. Matrix Multiplication

Let A be an $m\times n$ matrix, and B an $n\times p$ matrix. Performing the product $A \cdot B$ is equivalent to giving $N = m+p$ vectors and performing $m \cdot p$ inner products. The total number of multiplications is $(m+p)n + (m \cdot p - m - p)\lfloor (n+1)/2 \rfloor$ compared with $m \cdot p \cdot n$ multiplications in the regular way of performing matrix multiplication. The number of additions is $(m+p)(\lfloor n/2 \rfloor - 1) + mp(n + \lfloor n/2 \rfloor + 1)$ compared with $mp(n-1)$ in the regular way. If we assume m, n, and p are large, then the new method requires about $\frac{1}{2}$ mnp multiplications and $\frac{3}{2}$ mnp additions, while the regular method requires mnp multiplications and mnp additions.

B. Matrix Inversion

Let A be an $n\times n$ matrix to be inverted. Gaussian elimination method requires n^3 multiplications (we count a division as a multiplication) and $n^3 - 2n^2 + n$ additions. If $n = m \cdot k$ then we can view A as an $m\times m$ matrix whose entries are $k\times k$ matrices. We can invert A by Gaussian elimination of this $m\times m$ matrix, where addition and multiplication means addition and multiplication between $k\times k$ matrices and inversion means inverting a matrix. Applying the above method for the multiplications and assuming that the inversion of the $k\times k$ matrices is done by the regular Gaussian elimination, we obtain (assuming k is even) that

$$\frac{n^3}{2} + \frac{3}{2} n^2 + \frac{n}{2} (k^2 - k)$$

multiplications and

$$\frac{3}{2} n^3 \left(1 + \frac{4}{3k}\right) - n^2 \left(\frac{1}{2} + \frac{3}{k}\right) + n \left(2 - \frac{5}{2} k - \frac{1}{2} k^2\right)$$

additions are required.

C. Solutions of Linear Equations

Consider the system of linear equations $Ax = b$, where A is an $n\times n$ matrix. Solving these equations by Gaussian elimination requires $\frac{1}{3}(n^3 + 3n^2 - n)$ multiplications and $\frac{1}{6}(2n^3 + 3n^2 + n)$ additions.

As in the previous section, if $n=m\cdot k$ we perform the Gaussian elimination method on the $m\times m$ matrix whose entries are $k\times k$ matrices. Assuming that k is even this method requires

$$\frac{n^3}{6}+\frac{n^2}{4}(7k-k^2)+\frac{n}{6}(11k^2+3k+6)$$

multiplications and

$$\frac{n^3}{2}+n^2\left(\frac{3}{2}-\frac{1}{4k}-\frac{1}{2k}\right)+n\left(\frac{3}{2}k^2-\frac{20}{6}k+\frac{7}{4}-\frac{1}{2k}\right)$$

additions.

ACKNOWLEDGMENT

The author wishes to thank M. O. Rabin for his suggestions for the format of this note.

A Variable Counter Design Technique

EDWARD L. RENSCHLER, MEMBER, IEEE

***Abstract*—Two design techniques are presented which allow one counter circuit to divide a fixed-reference frequency by a wide range of counts. In the examples used the output frequency is preselected on three 10-position selector switches, providing for division of the input reference frequency by N, where $1\leq N\leq 999$. The techniques described are not dependent on the type of digital logic used and are therefore applicable to any family of binary logic modules.**

***Index Terms*—Counter, divider, programmable counter.**

An externally programmable counter is desirable in many frequency-synthesizing systems. For this counter to be of value, one must be able to change the count state, in steps of one, over a range of three or four hundred. In addition, this count change must be done externally to the counter itself. This means that no wiring can be changed and that the count decision must be made through logic. Two techniques that accomplish this are presented here. The techniques are described in general terms and will be directly applicable to any form of binary logic. The counters used as examples in this paper have the constraint

$$1\leq N\leq 999, \tag{1}$$

where N is the desired count.

The basic counter system is shown in Fig. 1. A description of the system operation is as follows. The desired count N is selected by setting each of the three 10-position input switches to the desired decimal number. Only three input switches are used here because of the constraint given in (1). N can be represented, for a 3-digit number, as

$$N=\sum_{i=0}^{2}\sum_{j=0}^{3}10^i 2^j B_{ij} \tag{2}$$

which is a 12-bit BCD word.

Each of the three stages of decimal–BCD conversion logic will be identical. One stage is illustrated in Fig. 2. The logic used here is a positive-logic NAND function.

The first technique for performing this variable-count function is shown in Fig. 3. When the shift pulse arrives at the input gating logic, each of the BCD numbers is preset into the appropriate counter stage. Each counter is a clocked BCD decade *down* counter. All three counter stages are driven from the same clock through gating, as opposed to the ripple-counter approach. This makes the counter system completely synchronous and is done deliberately to avoid the well-known problems of ripple counters. The count continues until the three stages are simultaneously in the zero state. When this occurs, the counter has counted down from the preset number N to 0. The zero state is detected by the zero state detector, and a trigger is produced which fires the "one shot," which in turn produces the shift pulse necessary to preset the number N again into the counter. The counter is reloaded during the zero count, and upon the next clock pulse the divide-by-N process begins again.

The one shot is necessary to insure that the number N is properly preset into the counter. If the one shot were not included, the dura-

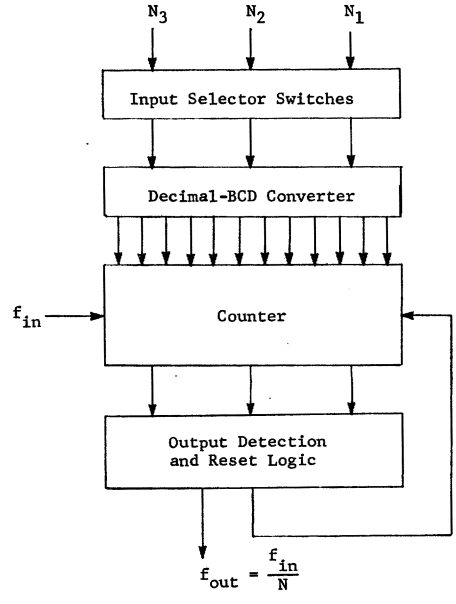

Fig. 1. Basic programmable counter.

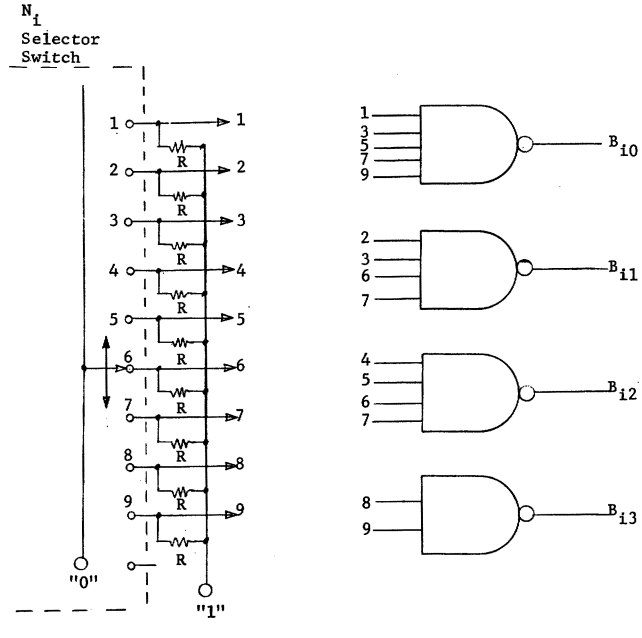

Fig. 2. Decimal–BCD conversion logic.

Manuscript received October 9, 1967; revised April 18, 1968.
The author is with the Semiconductor Products Div., Motorola, Inc., Phoenix, Ariz.

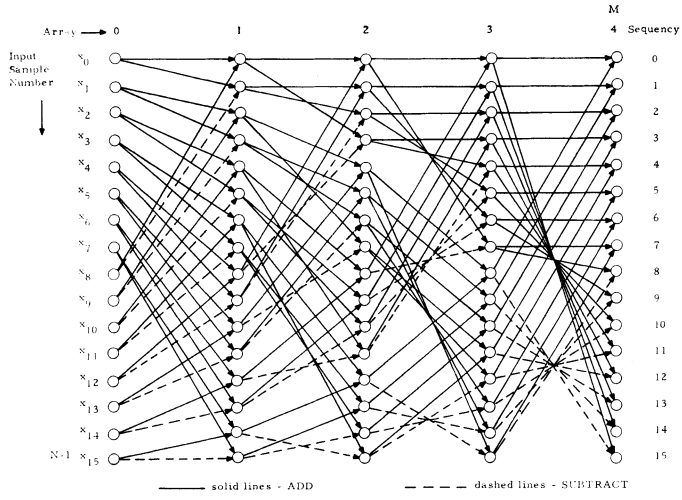

Fig. 1. Ordered Hadamard transform computation.

N = 2^M = Number of sample points.
L = Index of array used to compute values for array LP.
NZN = Number of computation groups for array being computed.
J = Index of point in array being computed

Reserve storage for an (M+1) by N array for XR.
Read input data into XR (1, J).
Output data appears in XR (M2, J) where M2 = M + 1.

Program

```
NH = N/2
DO 120 L = 1, M
LP = L + 1
LM = L - 1
NY = 0
NZ = 2**LM
NZI = 2*NZ
NZN = N/NZI
DO 110 I = 1, NZN
NX = NY + 1
NY = NY + NZ
JS = (I-1) *NZI
JD = JS + NZI + 1
DO 100 J = NX, NY
JS = JS + 1
J2 = J + NH
XR (LP, JS) = XR (L, J) + XR (L, J2)
JD = JD - 1
XR (LP, JD) = XR (L, J) - XR (L, J2)  See Note
100 CONTINUE
110 CONTINUE
120 CONTINUE
```

NOTE: For R-transform use XR (LP, JD) = ABS (XR(L, J) - XR (L, J2))

Fig. 2. FORTRAN program for ordered Hadamard transform and ordered R-transform.

Fig. 1 is a diagram showing the computational steps for finding the Hadamard transform or the R-transform of an input function defined by 16 samples. Subscripts of the rows in this diagram are numbered from 0 to $(N-1)$, where $N=2^M$ (M an integer) is the number of input samples, and the columns are numbered from 0 to M. The arrays (columns) represent the N input values and the N results of each of M transformation steps. The value at each node of arrays 1 through M is found by calculating the sum or difference of two values in the preceding array from a row i and a row $(i+N/2)$. In the diagram the solid arrows represent addition and the dashed arrows represent subtraction so that when a subtraction is involved, the value connected to the dashed arrow is subtracted from the value connected by the solid arrow. If the R-transform is being computed, absolute values are used for the results of each computation which is indicated as a subtraction. The values in array M are the Hadamard transform (or R-transform) values in order of sequency.

The method of computation which has been described was developed during a study of the identification of radars by means of video waveforms. The ordered transform was desired to observe the sequency characteristics of the video waveforms and to determine the effects of filtering and other operations on the correlation properties of the transforms of sampled waveforms.

Fig. 2 is a FORTRAN program for computing the ordered Hadamard transform and the ordered R-transform as described above. Note that in the program the rows of Fig. 1 are numbered from 1 to N, and the columns are numbered from 1 to $(M+1)$ to avoid zero subscript values. This program was used for a relatively small number of input samples where the number of storage locations required was not a problem. For applications where the available storage is a limiting factor, it has been suggested that the storage efficiency could be increased by alternating the computations between two arrays of length N. As the computation cannot be performed "in place," the number of storage locations required is at least twice the number of input samples.

L. J. ULMAN
Westinghouse Defense and Space Center
Aerospace Division, MS-360
Baltimore, Md. 21203

REFERENCES

[1] H. F. Harmuth, "A generalized concept of frequency and some applications," *IEEE Trans. Information Theory*, vol. IT-14, pp. 375–382, May 1968.
[2] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proc. IEEE*, vol. 57, pp. 58–68, January 1969.
[3] J. L. Shanks, "Computation of the fast Walsh-Fourier transform," *IEEE Trans. Computers* (Short Notes), vol. C-18, pp. 457–459, May 1969.
[4] H. Reitboeck and T. P. Brody, "A transformation with invariance under cyclic permutation for applications in pattern recognition," Westinghouse Research Labs, Scientific Paper 68-1F1-ADAPT-P1, September 17, 1968.

On Winograd's Algorithm for Inner Products

***Abstract*—This correspondence demonstrates an improvement on Winogard's algorithm for inner products as applied to the multiplication of two matrices.**

The total multiplication required is $n/2(n^2+2n-1)$, which is $n/2$ multiplication less than was required by the original algorithm and which reduces to seven multiplications for the product of two 2×2 matrices.

***Index Terms*—Product of matrices, sufficient number of arithmetical operations.**

Strassen [1] has shown that $4\cdot7\times n^{\log_2 7}$ arithmetical operations are sufficient in order to compute the coefficients of the product of two square matrices of order n.

Strassen's result is an iteration on the processes of finding the coefficients of the product of two 2×2 matrices in seven multiplications.

His algorithm is as follows.

If A, B are matrices of order $m2^{k+1}$ to be multiplied, write

$$A=\begin{pmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{pmatrix},\quad B=\begin{pmatrix}B_{11} & B_{12}\\ B_{21} & B_{22}\end{pmatrix},\quad AB=\begin{pmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{pmatrix},$$

where the A_{ik}, B_{ik}, C_{ik} are matrices of order $m2^k$. Then compute

$$\begin{aligned}
I&=(A_{11}+A_{22})(B_{11}+B_{22})\\
II&=(A_{21}+A_{22})B_{11}\\
III&=A_{11}(B_{12}-B_{22})\\
IV&=A_{22}(-B_{11}+B_{21})\\
V&=(A_{11}+A_{12})B_{22}\\
VI&=(-A_{11}+A_{21})(B_{11}+B_{12})\\
VII&=(A_{12}-A_{22})(B_{21}+B_{22})\\
C_{11}&=I+IV-V+VII\\
C_{21}&=II+IV\\
C_{12}&=III+V\\
C_{22}&=I+III-II+VI.
\end{aligned}$$

Manuscript received August 4, 1969; revised November 13, 1969. This work was supported by the Office of Naval Research, Information Systems Branch, under Contract N00014-C-0294.

The purpose of this correspondence is to demonstrate an improvement on Winograd's algorithm [2] for inner products as applied to the multiplication of two matrices.

The improved algorithm permits the multiplication of 2×2 matrices in seven multiplications as a special case.

Winograd's algorithm is as follows. Let

$$x = (x_1, \cdots, x_n) \quad \text{and} \quad y = (y_1, \cdots, y_n)$$

be two vectors. For each vector we compute the number

$$\xi = \sum_{j=1}^{[n/2]} x_{2j-1} \cdot x_{2j}$$

where $[t]$ denotes the integer part of t, and

$$\eta = \sum_{j=1}^{[n/2]} y_{2j-1} \cdot y_{2j}.$$

The inner product (x, y) is then given by

$$(x, y) = \begin{cases} \sum_{j=1}^{[n/2]} (x_{2j-1} + y_{2j})(x_{2j} + y_{2j-1}) - \xi - \eta & \text{if } n \text{ is even} \\ \sum_{j=1}^{[n/2]} (x_{2j-1} + y_{2j})(x_{2j} + y_{2j-1}) - \xi - \eta + x_n y_n & \text{if } n \text{ is odd.} \end{cases}$$

Thus, if $\bar{X}$ and $\bar{Y}$ are $n\times n$ matrices, performing the product $\bar{X}\cdot\bar{Y}$ is equivalent to having $2n$ vectors and performing n^2 inner products. Since there are only n different ξ's and n different η's, it follows that the total number of multiplications is

$$2n^2 + \left[\frac{n+1}{2}\right](n^2 - 2n).$$

The same count of the required number of multiplications can be accomplished as follows. Without loss of generality, for n even, let $\bar{X}$ and $\bar{Y}$ be two $n\times n$ matrices such that

$$\bar{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & \cdots & x_{1n} \\ x_{21} & x_{22} & & \cdots & & x_{2n} \\ x_{31} & x_{32} & & \cdots & & x_{3n} \\ \vdots & & & & & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & & x_{nn} \end{bmatrix}$$

and

$$\bar{Y} = \begin{bmatrix} y_{11} & y_{21} & y_{31} & \cdots & y_{n1} \\ y_{12} & y_{22} & y_{32} & & \\ y_{13} & y_{23} & y_{33} & & \vdots \\ y_{14} & \vdots & \vdots & & \\ \vdots & \vdots & \vdots & & \\ y_{1n} & y_{2n} & & & y_{nn} \end{bmatrix}.$$

Then the following are the terms of the resultant $\bar{X}\cdot\bar{Y}$ matrix:

$$\sum_{j=1}^{n/2} (A - B) \tag{1}$$

where

$$A = (x_{l(2j-1)} + y_{k(2j)}) \cdot (x_{l(2j)} + y_{k(2j-1)}) \quad \text{for } 1 \le k \le n \text{ and } 1 \le l \le n$$

and

$$B = (x_{l(2j-1)} \cdot x_{l(2j)} + y_{k(2j-1)} \cdot y_{k(2j)}) \quad \text{for } 1 \le k \le n \text{ and } 1 \le l \le n.$$

For a fixed j in (1) there are n^2 different A terms, but only $2n$ different products that are required for the B terms. Since j runs from 1 to $n/2$, the total multiplications required are: $n/2(n^2+2n)$, for even n as noted before.

We consider now a modification on the above algorithm as follows. Let

$$C = (x_{l(2j-1)} - y_{k(2j)}) \cdot (x_{l2j} - y_{k(2j-1)}).$$

Then

$$(A - B) = \tfrac{1}{2}(A + C) \tag{2}$$

and

$$B = \tfrac{1}{2}(A - C). \tag{3}$$

We observe now that for fixed j, if given the following $2n-1$ B terms,

$$(x_{1(2j-1)} \cdot x_{1(2j)} + y_{k(2j-1)} \cdot y_{k(2j)}) \quad \text{for all } 1 \le k \le n \tag{4}$$

and

$$x_{l(2j-1)} \cdot x_{l(2j)} + y_{l(2j-1)} \cdot y_{l(2j)}) \quad \text{for all } 2 \le l \le n, \tag{5}$$

it is possible by substitution only to solve for the rest of the $n^2 - (2n-1)$ B terms. This is true since for fixed j, each product that appears in a B term appears n times in different B terms so that the $(2n-1)$ B terms as selected in (4) and (5) will suffice to solve by substitution for the rest of the $n^2-(2n-1)$ B terms.

We can now adapt the following procedure for the solution.

1) For fixed j:
   a) In n^2 multiplication solve for all n^2 A terms.
   b) In $(2n-1)$ multiplication solve for the $(2n-1)$ C terms such that
      i) j is fixed
      ii) $l=1$ and $1\le k\le n$
      iii) $2\le l=k\le n$.
   c) From b) solve for the $(2n-1)$ B terms by the use of (3).
   d) By substitution solve for the rest of the $n^2-(2n-1)$ B terms.
   e) By subtraction solve for all n^2 $(A-B)$ terms.

2) Repeat 1) for $1<j\le n/2$.

Thus the total multiplication required is $n/2$ (n^2+2n-1), which is $n/2$ multiplication less than was required by the original algorithm, and which reduces to seven multiplications for the product of two 2×2 matrices. (We have considered only multiplication of nonconstant terms, thus the multiplications by $\frac{1}{2}$ are not represented in this count.)

Example: Let

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} A & C \\ B & D \end{pmatrix} = \begin{pmatrix} aA + bB, & aC + bD \\ cA + dB, & cC + dD \end{pmatrix}.$$

The seven required multiplications are:

$$\begin{aligned} A_1 &= (a + B)\cdot(A + b) \\ A_2 &= (a + D)\cdot(C + b) \\ A_3 &= (c + B)\cdot(A + d) \\ A_4 &= (c + D)\cdot(C + d) \\ C_1 &= (a - B)\cdot(A - b) \\ C_2 &= (a - D)\cdot(C - b) \\ C_3 &= (c - B)\cdot(A - d). \end{aligned}$$

Therefore

$$\begin{aligned} B_1 &= \tfrac{1}{2}(A_1 - C_1) \\ B_2 &= \tfrac{1}{2}(A_2 - C_2) \\ B_3 &= \tfrac{1}{2}(A_3 - C_3) \\ B_4 &= B_2 + B_3 - B_1, \end{aligned}$$

and

$$\begin{aligned} aA + bB &= A_1 - B_1 \\ aC + bD &= A_2 - B_2 \\ cA + dB &= A_3 - B_3 \\ cC + dD &= A_4 - B_4. \end{aligned}$$

ABRAHAM WAKSMAN
Stanford Research Institute
Menlo Park, Calif. 94025

REFERENCES

[1] V. Strassen, "Gaussian elimination is not optimal," *Seminar für Angewandte Mathematik der Universität Zürich*, December 1968.
[2] S. Winograd, "A new algorithm for inner product," IBM Res. Rept. RC-1943, November 21, 1967.