SPRA008

Matrix Multiplication

with the TMS32010 and TMS32020

Digital Signal Processing
Application Report

TEXAS
INSTRUMENTS

Matrix Multiplication with the TMS32010 and TMS32020

Charles Dana Crowell
Digital Signal Processing
Applications Engineering

TEXAS
INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes in the devices or the device specifications identified in this publication without notice. TI advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current.

TI warrants performance of its semiconductor products, including SNJ and SMJ devices, to current specifications in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems such testing necessary to support this warranty. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed.

In the absence of written agreement to the contrary, TI assumes no liability for TI applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor device might be or are used.

INTRODUCTION

Matrix multiplication is useful in applications such as graphics, numerical analysis, or high-speed control. The purpose of this application report is to illustrate matrix multiplication on two digital signal processors, the TMS32010 and TMS32020.

Both the TMS32010 and TMS32020 can multiply any two matrices of size $M \times N$ and $N \times P$. The programs for the TMS32010 and TMS32020, included in the appendices, can multiply large matrices and are only limited by the amount of internal data RAM available. Assuming a 200-ns cycle time, the TMS32010 and TMS32020 can calculate $[1 \times 3] \times [3 \times 3]$ in 5.4 microseconds.

Before discussing the two versions of implementing a matrix multiplication algorithm, a brief review of matrix multiplication is presented along with three examples of graphics applications.

MATRIX MULTIPLICATION

The size of a matrix is defined by the number of rows and columns it contains. For example, the following is a 5×3 matrix since it contains five rows and three columns.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{bmatrix}$$

Any two matrices can be multiplied together as long as the second matrix has the same number of rows as the first has of columns. This condition is called conformability. For example, if a matrix A is an $M \times N$ matix and a matrix B is an $N \times P$ matrix, then the two can be multiplied together with the resulting matrix being of size $M \times P$.

$$A = \begin{bmatrix} 3 & 4 \\ 2 & 7 \end{bmatrix} \qquad B = \begin{bmatrix} 4 \\ 6 \end{bmatrix} \qquad AB = \begin{bmatrix} 36 \\ 50 \end{bmatrix}$$

$M \times N = 2 \times 2 \qquad N \times P = 2 \times 1 \qquad M \times P = 2 \times 1$

Example: (3)(4) + (4)(6) = 36

Given the two conformable matrices A and B, the elements of $C = A \times B$ are given by:

$$C_{ij} = \sum_{k=1}^{N} a_{ik} \times b_{kj}$$

for i = 1,...,M and j = 1,...,P

Q12 FORMAT

Applications often require multiplication of mixed numbers. Since the TMS32010 and TMS32020 implement fixed-point arithmetic, the programs in the appendices assume a Q12 format, i.e., 12 bits follow an assumed binary point. The bits to the right of the assumed binary point represent the fractional part of the number and the four bits to the left represent the integer part of the number. An example of Q12 format is as follows:

0 0 0 1.1 1 0 1 1 1 1 0 0 0 0 0 = 1.866

ASSUMED BINARY POINT

```
          0000.110111100000 = 0.866 in Q12
      ×   0000.100000000000 = 0.5 in Q12
-------------------------------------------
00000000.011011110000000000000000 = 0.433 in Q24
```

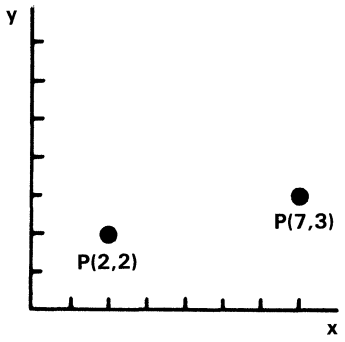
The result of a Q12 by Q12 multiplication is a number in a Q24 format that can easily be converted to Q12 by a logical left-shift of four. The first four bits will be lost as well as the last twelve, but these bits are insignificant for Q12. Note that the programs in the appendices provide no protection against overflow; therefore, the design engineer should implement a format that best fits the application.

GRAPHICS APPLICATIONS

Operations in graphics applications, such as translation, scaling, or rotation, require matrix manipulations to be performed in a limited amount of time. Therefore, the TMS32010 and TMS32020 processors are ideal for these applications. Graphics applications, such as scaling and rotation of points in a coordinate system, require multiplication of matrices. Translation is typically implemented by addition of two matrices. However, when points are represented in a homogeneous coordinate system, translation can be implemented by multiplication. In a homogeneous coordinate system, a point P(x,y) is represented as P(X,Y,1). This type of coordinate system is desirable since it relates translation with scaling and rotation.

Translation can be defined as the moving of a point or points in a coordinate system from one location to another without rotating. This is accomplished by adding a displacement value D_X to the X coordinate of a point and adding a displacement value D_Y to the Y coordinate, thus moving the point from one location to another. Figure 1 shows both addition and multiplication methods of translation and an example of each.

Similar to translation, scaling can be implemented by matrix multiplication. Points can be scaled by multiplying



ADDITION METHOD

$[X_{NEW}\ Y_{NEW}] = [X_{OLD}\ Y_{OLD}] + [D_x\ D_y]$

where $D_x = 5$ and $D_y = 1$

MULTIPLICATION METHOD

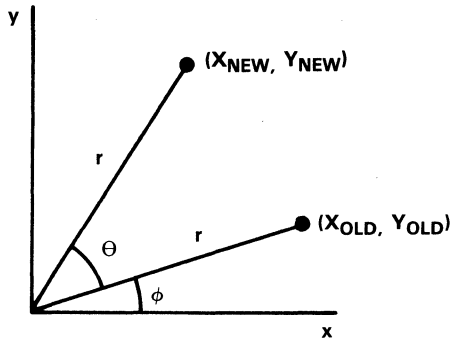
$$[X_{NEW}\ Y_{NEW}\ 1] = [X_{OLD}\ Y_{OLD}\ 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix}$$

where $D_x = 5$ and $D_y = 1$

Figure 1. Translation of Coordinates

each coordinate of a point (or points) by a scaling value S_x and S_y. Scaling an object is similar to stretching or shrinking an object. The coordinates of each point that makes up the object are multiplied by a scaling value which scales the object to a larger or smaller scale. Figure 2 shows the scaling of an object from one size to another.

Rotation of the coordinates of a point (or points) about an angle theta can also be accomplished by a matrix multiplication. The following set of equations results with the matrix multiplication required to rotate an object about any angle.



$X_{OLD} = r\cos\phi$
$Y_{OLD} = r\sin\phi$

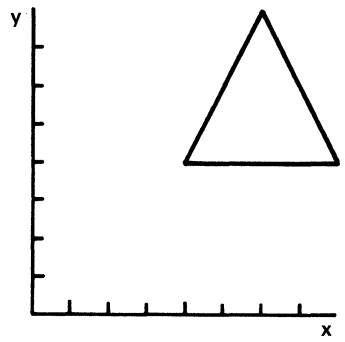
$X_{NEW} = r\cos(\Theta+\phi) = r\cos\phi\cos\Theta - r\sin\phi\sin\Theta$
$Y_{NEW} = r\sin(\Theta+\phi) = r\cos\phi\sin\Theta + r\sin\phi\cos\Theta$

$X_{NEW} = X_{OLD}\cos\Theta - Y_{OLD}\sin\Theta$
$Y_{NEW} = X_{OLD}\sin\Theta - Y_{OLD}\cos\Theta$

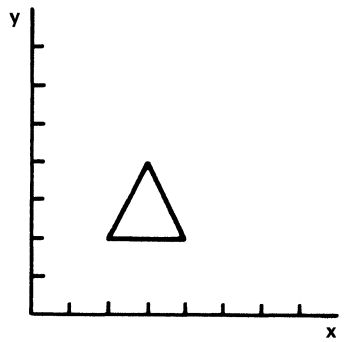
OR

$$[X_{NEW}\ Y_{NEW}\ 1] = [X_{OLD}\ Y_{OLD}\ 1] \cdot \begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

BEFORE SCALING



AFTER SCALING



Let the scaling factors S_x and $S_y = 0.5$

$$[X_{NEW}\ Y_{NEW}\ 1] = [X_{OLD}\ Y_{OLD}\ 1] \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

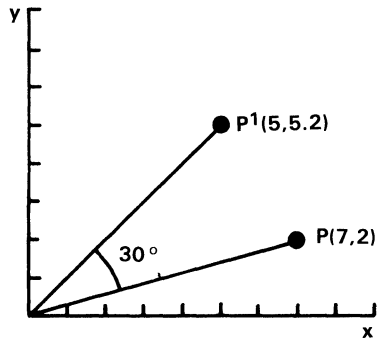
$$[X\ Y\ 1] = [4\ 4\ 1] \cdot \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$[X\ Y\ 1] = [2\ 2\ 1]$

Figure 2. Scaling From One Size To Another

Figure 3 shows an implementation of these equations to rotate an object 30 degrees about the origin.

Figures 4 and 5 show a segment of straight-line TMS32010 and TMS32020 code, respectively. These programs calculate the coordinate rotation example using a Q12 format. Note that once the matrices are loaded into memory, the procssors can calculate the results in 5.4 microseconds. The segment of TMS32020 code in Figure 5 implements the MAC instruction. For small matrices, the MAC instruction in conjunction with the RPT instruction gains little due to the overhead timing of the MAC instruction. However, for larger matrices, this method is most efficient since the MAC instruction becomes single-cycle in the repeat mode. For applications that only require translation, scaling, or rotation of coordinates, straight-line code as in Figures 4 and 5 is more efficient than the larger programs in the appendices.



$$[X\ Y\ 1] = [7\ 2\ 1] \cdot \begin{bmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[X\ Y\ 1] = [5.0\ 5.2\ 1]$$

Figure 3. Implementation of Rotation Matrix

```
NO$IDT       32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107     09:54:24  02-25-85
                                                                PAGE 0001

0001                ********************************************************************
0002                *                                                                  *
0003                *       THIS ROUTINE ASSUMES THE INPUTS ARE IN Q12.                 *
0004                *       THE FIRST NINE INPUTS SHOULD BE THE ROTATION               *
0005                *       MATRIX (HOMOGENEOUS COORDINATES), ENTERED BY               *
0006                *       COLUMNS. THE LAST THREE INPUTS SHOULD BE THE               *
0007                *       OLD X AND Y COORDINATES.                                   *
0008                *                                                                  *
0009                ********************************************************************
0010 0000 6E00  ROTATE LDPK    0
0011      000C  ANS    EQU     12
0012 0001 6880         LARP    0
0013 0002 7000         LARK    AR0,0    * POINT AT BEGINNING OF ROTATION MATRIX.
0014 0003 7109         LARK    AR1,9    * POINT AT BEGINNING OF OLD COORDINATES.
0015 0004 40A8         IN      *+,PA0   * INPUT ROTATION MATRIX AND OLD
0016 0005 40A8         IN      *+,PA0   * COORDINATES.
0017 0006 40A8         IN      *+,PA0
0018 0007 40A8         IN      *+,PA0
0019 0008 40A8         IN      *+,PA0
0020 0009 40A8         IN      *+,PA0
0021 000A 40A8         IN      *+,PA0
0022 000B 40A8         IN      *+,PA0
0023 000C 40A8         IN      *+,PA0
0024 000D 40A8         IN      *+,PA0
0025 000E 40A8         IN      *+,PA0
0026 000F 40A8         IN      *+,PA0
0027 0010 7F89         ZAC             * CLEAR ACCUMULATOR.
0028 0011 7000         LARK    AR0,0
0029 0012 6AA1         LT      *+,1     * CALCULATE NEW X COORDINATE.
0030 0013 6DA0         MPY     *+,0
0031 0014 6CA1         LTA     *+,1
0032 0015 6DA0         MPY     *+,0
0033 0016 6CA1         LTA     *+,1
0034 0017 6DA0         MPY     *+,0
0035 0018 7F8F         APAC
0036 0019 5C0C         SACH    ANS,4    * CONVERT TO Q12 AND OUTPUT RESULT.
0037 001A 480C         OUT     ANS,PA0
```

Figure 4. TMS32010 Code for Rotation

```
0038 001B 7F89          ZAC
0039 001C 7109          LARK    AR1,9    * CALCULATE NEW Y COORDINATES.
0040 001D 6AA1          LT      *+,1
0041 001E 6DA0          MPY     *+,0
0042 001F 6CA1          LTA     *+,1
0043 0020 6DA0          MPY     *+,0
0044 0021 6CA1          LTA     *+,1
0045 0022 6DA0          MPY     *+,0
0046 0023 7F8F          APAC
0047 0024 5C0C          SACH    ANS,4    * CONVERT TO Q12 AND OUTPUT RESULT.
0048 0025 480C          OUT     ANS,PA0
0049 0026 7F89          ZAC
0050 0027 7109          LARK    AR1,9    * FINISH HOMOGENEOUS MATRIX.
0051 0028 6AA1          LT      *+,1
0052 0029 6DA0          MPY     *+,0
0053 002A 6CA1          LTA     *+,1
0054 002B 6DA0          MPY     *+,0
0055 002C 6CA1          LTA     *+,1
0056 002D 6DA0          MPY     *+,0
0057 002E 7F8F          APAC
0058 002F 5C0C          SACH    ANS,4
0059 0030 480C          OUT     ANS,PA0
0060 0031 7F8D          RET
NO ERRORS, NO WARNINGS
```

Figure 4. TMS32010 Code for Rotation (Concluded)

```
NO$IDT      32020 FAMILY MACRO ASSEMBLER   PC0.7 84.348    16:07:15  02-25-85
                                                                PAGE 0001

0001            ************************************************************
0002            *                                                          *
0003            *      THIS ROUTINE ASSUMES THE INPUTS ARE IN Q12.          *
0004            *      THE FIRST NINE INPUTS SHOULD BE THE ROTATION        *
0005            *      MATRIX (HOMOGENEOUS COORDINATES), ENTERED BY        *
0006            *      COLUMNS. THE LAST THREE INPUTS SHOULD BE THE        *
0007            *      OLD X AND Y COORDINATES.                            *
0008            *                                                          *
0009            ************************************************************
0010 0000 5589  ROTATE LARP    1           * USE AUXILIARY REGISTER 1.
0011      000C  ANS    EQU     12
0012 0001 CA00         ZAC                 * INITIALIZE ACCUMULATOR.
0013 0002 C806         LDPK    6
0014 0003 D100         LRLK    AR1,>300    * LOAD ROTATION MATRIX INTO B1.
     0004 0300
0015 0005 CB08         RPTK    8
0016 0006 80A0         IN      *+,PA0
0017 0007 D100         LRLK    AR1,>200    * LOAD COORDINATES INTO BLOCK B0.
     0008 0200
0018 0009 CB02         RPTK    2
0019 000A 80A0         IN      *+,PA0
0020 000B CE05         CNFP                * CONFIGURE B0 AS PROGRAM MEMORY.
0021 000C A000         MPYK    >0          * CLEAR P REGISTER.
0022 000D D100         LRLK    AR1,>300
     000E 0300
0023 000F CB02         RPTK    2
0024 0010 5DA0         MAC     >FF00,*+    * CALCULATE THE NEW X COORDINATE.
     0011 FF00
0025 0012 CE15         APAC
0026 0013 6C0C         SACH    ANS,4
0027 0014 E00C         OUT     ANS,PA0     * OUTPUT NEW X COORDINATE.
0028 0015 A000         MPYK    >0          * CLEAR P REGISTER.
0029 0016 CA00         ZAC
0030 0017 CB02         RPTK    2
0031 0018 5DA0         MAC     >FF00,*+    * CALCULATE NEW Y COORDINATE.
     0019 FF00
0032 001A CE15         APAC
0033 001B 6C0C         SACH    ANS,4
0034 001C E00C         OUT     ANS,PA0     * OUTPUT NEW Y COORDINATE.
0035 001D A000         MPYK    >0          * CLEAR P REGISTER.
0036 001E CA00         ZAC
0037 001F CB02         RPTK    2
0038 0020 5DA0         MAC     >FF00,*+    * FINISH HOMOGENEOUS MATRIX.
     0021 FF00
0039 0022 CE15         APAC
0040 0023 6C0C         SACH    ANS,4
0041 0024 E00C         OUT     ANS,PA0
0042 0025 CE26         RET
NO ERRORS, NO WARNINGS
```

Figure 5. TMS32020 Code for Rotation

To combine translation, scaling, and rotation, a more general matrix can be implemented.

GENERAL MATRIX FOR
TWO-DIMENSIONAL SYSTEMS

$$\begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

The upper 2×2 matrix is a combination rotation matrix and scaling matrix. The t_x and t_y values are the translation values. A three-dimensional general matrix can be developed similar to the two-dimensional translation, scaling, and rotation matrix.

GENERAL MATRIX FOR
THREE-DIMENSIONAL SYSTEMS

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

IMPLEMENTATION OF THE MATRIX MULTIPLICATION ALGORITHM FOR THE TMS32010

The implementation of the algorithm for the TMS32010 shown in Figure 6 assumes that the two matrices to be multiplied together are of size $M \times N$ and $N \times P$. Three major

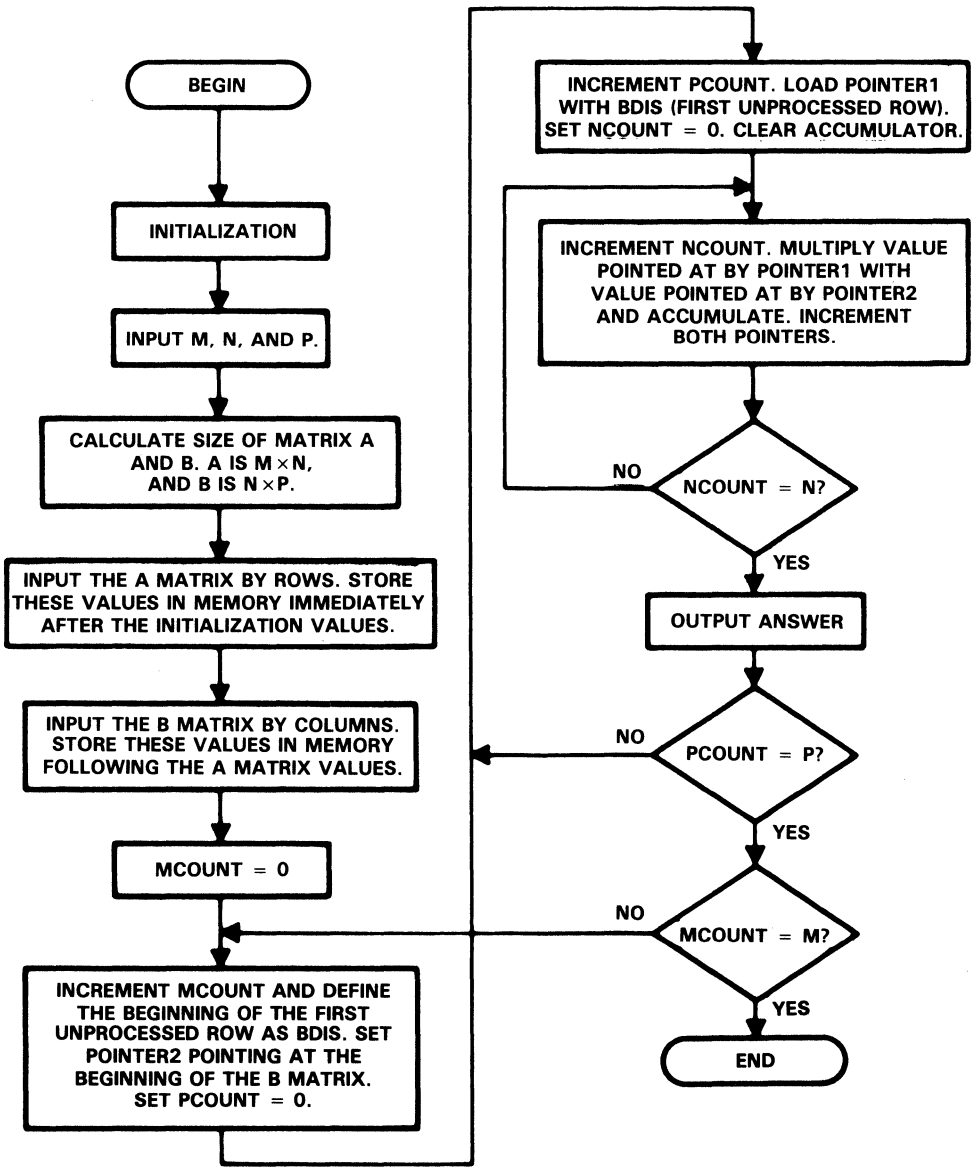


Figure 6. TMS32010 Flowchart

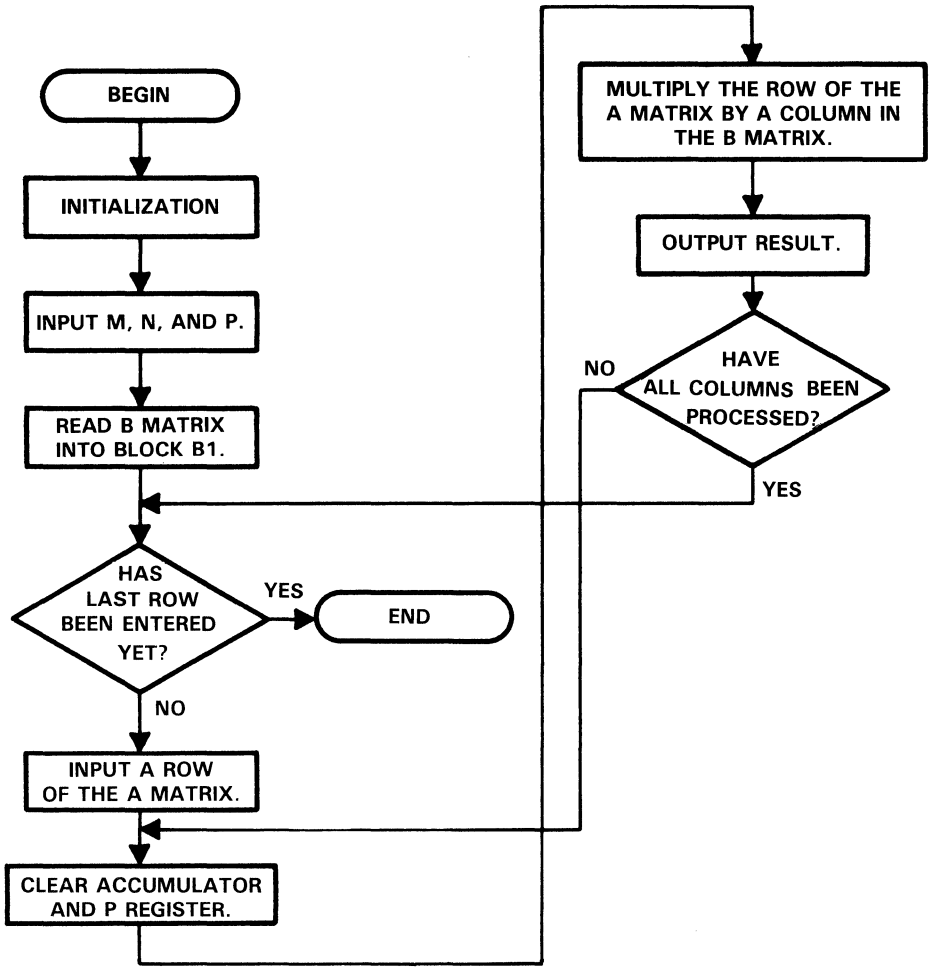


Figure 7. TMS32020 Flowchart

loops are included to multiply the two matrices. The outside loop control is labeled MCOUNT since it controls which row in the A matrix is being referenced during the multiplication. The secondary loop control is labeled PCOUNT because it counts how many columns in the B matrix have been processed. The inside loop control is labeled NCOUNT since it controls the multiplication of the values in the A matrix with the values in the B matrix.

IMPLEMENTATION OF THE MATRIX MULTIPLICATION ALGORITHM FOR THE TMS32020

The implementation of the algorithm for the TMS32020 is somewhat different since its advanced instruction set allows for a more efficient method of computing matrix multiplication. The TMS32020 version in Figure 7 also assumes that the two matrices to be multiplied are of size $M \times N$ and $N \times P$. This program takes a row of the A matrix, loads it into block B0 of data memory, and then multiplies this row by all columns in the B matrix. The TMS32020 continues this process until all the rows in the A matrix have been multiplied by all the columns in the B matrix. The TMS32020 version is similar to the TMS32010 in that the A matrix must be entered by rows and the B matrix by columns. This allows for a faster execution time. Figure 7 shows the basic implementation of the matrix multiplication algorithm that the TMS32020 uses to multiply two matrices.

Since the programs in the appendices treat the matrices differently, a memory map is included to help in understanding the two versions. Figure 8 shows how the matrices should look in memory after they have been entered. Note that for the TMS32020 version, the A matrix values reside in program memory since the CNFP (configure as program memory) instruction was implemented. Note also that only one row of the A matrix is in this block since the program enters one row at a time.

For the following matrices,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

the memory would be configured in this manner for the TMS32010 and TMS32020.

TMS32010
DATA MEMORY

LOCATION (IN HEX)	VALUE
>00F	a_{11}
>010	a_{12}
>011	a_{21}
>012	a_{22}
>013	b_{11}
>014	b_{21}
>015	b_{12}
>016	b_{22}
>017	b_{13}
>018	b_{23}

TMS32020

DATA MEMORY		PROGRAM MEMORY	
LOCATION (IN HEX)	VALUE	LOCATION (IN HEX)	VALUE
>308	b_{11}	>FF00	a_{i1}
>309	b_{21}	>FF01	a_{i2}
>30A	b_{12}		
>30B	b_{22}		
>30C	b_{13}		
>30D	b_{23}		

Figure 8. Memory Maps

SUMMARY

The TMS32010 and TMS32020 processors can be used to multiply large matrices efficiently. A brief review of matrix multiplication has been given to assist in the understanding of fundamental matrix multiplication. Three examples of graphics applications have been presented since these applications often require multiplication of matrices.

The TMS320 family has the power and flexibility to cost-effectively implement a wide range of high-speed graphics, numerical analysis, digital signal processing, and control applications. Since the TMS32010 and TMS32020 combine the flexibility of a high-speed controller with the numerical capability of an array processor, a new approach to applications such as graphics can now be considered.

REFERENCES

1. J.D. Foley and A. Van Dam, *Fundamentals of Interactive Commputer Graphics*, Addison-Wesley Publishing Company, Inc. (1982).
2. S.D. Conte and Carl de Boor, *Elementary Numerical Analysis*, McGraw-Hill, Inc. (1980).

Appendix A

```
NO$IDT      32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    10:03:42  02-25-85
                                                                PAGE 0001

0001            ****************************************************
0002            *   ALL INPUTS AND OUTPUTS FOR THIS PROGRAM SHOULD *
0003            *   BE OR ARE IN Q12 FORMAT EXCEPT FOR THE M, N,   *
0004            *   AND P INPUTS, WHICH SHOULD BE Q0.              *
0005            ****************************************************
0006 0000               AORG    0
0007       0000 M       EQU     >0
0008       0001 N       EQU     >1
0009       0002 P       EQU     >2
0010       0003 C1      EQU     >3
0011       0004 C2      EQU     >4
0012       0005 C3      EQU     >5
0013       0006 ANS     EQU     >6
0014       0007 ADIS    EQU     >7
0015       0008 BDIS    EQU     >8
0016       0009 CDIS    EQU     >9
0017       000A TEMP    EQU     >A
0018       000B COI     EQU     >B
0019       000C COS     EQU     >C
0020       000D T       EQU     >D
0021       000E ONE     EQU     >E
0022            *
0023            * INITIALIZATION
0024            *
0025 0000 6E00          LDPK    0
0026 0001 6880          LARP    0
0027 0002 7E0F          LACK    15
0028 0003 500C          SACL    COS
0029 0004 500D          SACL    T
0030 0005 7E01          LACK    1
0031 0006 500E          SACL    ONE
0032            *
0033            * MATRIX A IS M x N AND MATRIX B IS N x P.
0034            * THESE STATEMENTS READ IN THE SIZES OF
0035            * THE TWO MATRICES.
0036            *
0037 0007 4000          IN      M,PA0
0038 0008 4001          IN      N,PA0
0039 0009 4002          IN      P,PA0
0040            *
0041            * CALCULATE THE LENGTH OF THE A MATRIX AND
0042            * STORE THIS VALUE IN ADIS.
0043            *
0044 000A 6A00          LT      M
0045 000B 6D01          MPY     N
0046 000C 7F8E          PAC
0047 000D 5007          SACL    ADIS
0048            *
0049            * CALCULATE THE LENGTH OF THE B MATRIX AND
0050            * STORE THIS VALUE IN BDIS.
0051            *
0052 000E 6A01          LT      N
0053 000F 6D02          MPY     P
0054 0010 7F8E          PAC
0055 0011 5008          SACL    BDIS
0056            *
0057            * POINT AT THE END OF THE INITIAL DATA.
0058            *
0059 0012 380C          LAR AR0,COS
```

```
0060                *
0061                * READ THE A MATRIX VALUES INTO DATA RAM.
0062                * THIS MATRIX MUST BE ENTERED BY ROWS.
0063                * THE MATRIX VALUES WILL BE LOCATED IN
0064                * DATA RAM FOLLOWING THE INITIALIZATION
0065                * VALUES.
0066                *
0067 0013 200B  FST    LAC     COI
0068 0014 000E         ADD     ONE
0069 0015 500B         SACL    COI
0070 0016 4088         IN      *,PA0
0071 0017 68A8         MAR     *+
0072 0018 2007         LAC     ADIS
0073 0019 100B         SUB     COI
0074 001A FE00         BNZ     FST
     001B 0013
0075                *
0076                * RESET COUNTER TO READ IN THE B MATRIX VALUES.
0077                *
0078 001C 7F89         ZAC
0079 001D 500B         SACL    COI
0080                *
0081                * READ THE B MATRIX VALUES INTO DATA RAM.
0082                * UNLIKE THE A MATRIX, THESE VALUES MUST BE
0083                * ENTERED BY COLUMNS.  THESE VALUES WILL BE
0084                * LOCATED IN DATA RAM FOLLOWING THE A MATRIX VALUES.
0085                *
0086                *
0087 001E 200B  SND    LAC     COI
0088 001F 000E         ADD     ONE
0089 0020 500B         SACL    COI
0090 0021 4088         IN      *,PA0
0091 0022 68A8         MAR     *+
0092 0023 2008         LAC     BDIS
0093 0024 100B         SUB     COI
0094 0025 FE00         BNZ     SND
     0026 001E
0095                *
0096                * MORE INITIALIZATION
0097                *
0098 0027 200D         LAC     T
0099 0028 1001         SUB     N
0100 0029 5003         SACL    C1
0101 002A 200D         LAC     T
0102 002B 0007         ADD     ADIS
0103 002C 500D         SACL    T
0104 002D 1001         SUB     N
0105 002E 5007         SACL    ADIS
0106                *
0107                * CALCULATE A x B
0108                *
0109                *
0110                *
0111                *
0112                *                      N
0113                *                    _____
0114                *                    \
0115                * OUTPUT(ij)   =      \  A(ik) x B(kj)
0116                *                     /
0117                *                    /
0118                *                    -----
0119                *                    k = 1
0120                *
0121 002F 2003  FS     LAC     C1
0122 0030 0001         ADD     N
```

```
 0123 0031 5003          SACL    C1
 0124 0032 6881          LARP    1
 0125 0033 390D          LAR     AR1,T
 0126 0034 6880          LARP    0
 0127 0035 7F89          ZAC
 0128 0036 5004          SACL    C2
 0129 0037 2004  SN      LAC     C2
 0130 0038 000E          ADD     ONE
 0131 0039 5004          SACL    C2
 0132 003A 3803          LAR     AR0,C1
 0133 003B 7F89          ZAC
 0134 003C 5006          SACL    ANS
 0135 003D 5005          SACL    C3
 0136 003E 2005  TH      LAC     C3
 0137 003F 000E          ADD     ONE
 0138 0040 5005          SACL    C3
 0139 0041 6506          ZALH    ANS
 0140 0042 6AA1          LT      *+,AR1
 0141 0043 6DA0          MPY     *+,AR0
 0142 0044 7F8F          APAC
 0143 0045 5806          SACH    ANS
 0144 0046 2005          LAC     C3
 0145 0047 1001          SUB     N
 0146 0048 FE00          BNZ     TH
      0049 003E
 0147             *
 0148             * LOAD ACCUMULATOR WITH HIGH WORD OF Q24 RESULT.
 0149             * LEFT-SHIFT FOUR TO CONVERT TO Q12.
 0150             * NOTE THAT ONLY THE 12 MSB'S ARE SIGNIFICANT.
 0151             *
 0152 004A 2406          LAC     ANS,4
 0153 004B 5006          SACL    ANS
 0154 004C 4806          OUT     ANS,PA0
 0155 004D 2004          LAC     C2
 0156 004E 1002          SUB     P
 0157 004F FE00          BNZ     SN
      0050 0037
 0158 0051 2003          LAC     C1
 0159 0052 1007          SUB     ADIS
 0160 0053 FE00          BNZ     FS
      0054 002F
 0161 0055 F900  QUIT    B       QUIT
      0056 0055
NO ERRORS, NO WARNINGS
```

Appendix B

```
NO$IDT      32020 FAMILY MACRO ASSEMBLER   PC0.7 84.348     11:22:01  02-25-85
                                                                  PAGE 0001

0001                ************************************************
0002                *   ALL INPUTS AND OUTPUTS FOR THIS PROGRAM    *
0003                *   SHOULD BE OR ARE IN Q12 FORMAT EXCEPT      *
0004                *   FOR THE M, N, AND P, WHICH SHOULD BE Q0.   *
0005                ************************************************
0006 0020                      AORG    32
0007       0000     M          EQU     >0
0008       0001     N          EQU     >1
0009       0002     P          EQU     >2
0010       0003     ANS        EQU     >3
0011       0004     BDM1       EQU     >4
0012       0005     ONE        EQU     >5
0013       0006     NM1        EQU     >6
0014       0007     PM1        EQU     >7
0015                *
0016                * INITIALIZATION
0017                *
0018 0020 C806               LDPK    6
0019 0021 D100               LRLK    AR1,>300
     0022 0300
0020 0023 5589               LARP    1
0021 0024 CA01               LACK    >1
0022 0025 6005               SACL    ONE
0023                *
0024                * READ SIZES OF MATRICES.
0025                *
0026 0026 CB02               RPTK    2
0027 0027 80A0               IN      *+,PA0
0028                *
0029                * MORE INITIALIZATION
0030                *
0031 0028 2001               LAC     M
0032 0029 0005               ADD     ONE
0033 002A 6001               SACL    M
0034 002B 2000               LAC     N
0035 002C 1005               SUB     ONE
0036 002D 6006               SACL    NM1
0037 002E 3C00               LT      N
0038 002F 3802               MPY     P
0039 0030 CE14               PAC
0040 0031 1005               SUB     ONE
0041 0032 6004               SACL    BDM1
0042 0033 2002               LAC     P
0043 0034 1005               SUB     ONE
0044 0035 6007               SACL    PM1
0045                *
0046                * READ IN THE B MATRIX.
0047                *
0048 0036 D100               LRLK    AR1,>308
     0037 0308
0049 0038 4B04               RPT     BDM1
0050 0039 80A0               IN      *+,PA0
0051 003A 2001   CALLER     LAC     M
0052 003B 1005               SUB     ONE
0053 003C 6001               SACL    M
0054 003D F680               BZ      QT
     003E 0052
0055                *
0056                * CALL ROUTINE TO READ IN A ROW
```

```
0057                    * OF THE A MATRIX.
0058                    *
0059 003F FE80                  CALL    IO
     0040 0053
0060 0041 D100                  LRLK    AR1,>308
     0042 0308
0061 0043 5589                  LARP    1
0062 0044 3007                  LAR     AR0,PM1
0063                    *
0064                    * CLEAR ACCUMULATOR AND P REGISTER.
0065                    *
0066 0045 A000  MUL             MPYK    0
0067 0046 CA00                  ZAC
0068                    *
0069                    * MULTIPLY A ROW BY A COLUMN.
0070                    *
0071 0047 4B06                  RPT     NM1
0072 0048 5DA0                  MAC     >FF00,*+
     0049 FF00
0073 004A CE15                  APAC
0074                    *
0075                    * OUTPUT RESULT.
0076                    *
0077 004B 6C03                  SACH    ANS,4
0078 004C E003                  OUT     ANS,PA0
0079 004D 5588                  LARP    0
0080                    *
0081                    * CHECK TO SEE IF ALL COLUMNS HAVE BEEN PROCESSED.
0082                    *
0083 004E FB99                  BANZ    MUL,*-,1
     004F 0045
0084                    *
0085                    *       GO GET NEXT ROW.
0086                    *
0087 0050 FF80                  B       CALLER
     0051 003A
0088 0052 CE1F  QT      IDLE
0089 0053 CE04  IO      CNFD
0090 0054 5589                  LARP    1
0091 0055 D100                  LRLK    AR1,>200
     0056 0200
0092 0057 4B06                  RPT     NM1
0093 0058 80A0                  IN      *+,PA0
0094 0059 CE05                  CNFP
0095 005A CE26                  RET
NO ERRORS, NO WARNINGS
```

April 1985
Revision *
1602253-9701
Printed in U.S.A.

TEXAS INSTRUMENTS

Creating useful products
and services for you.

SPRA008