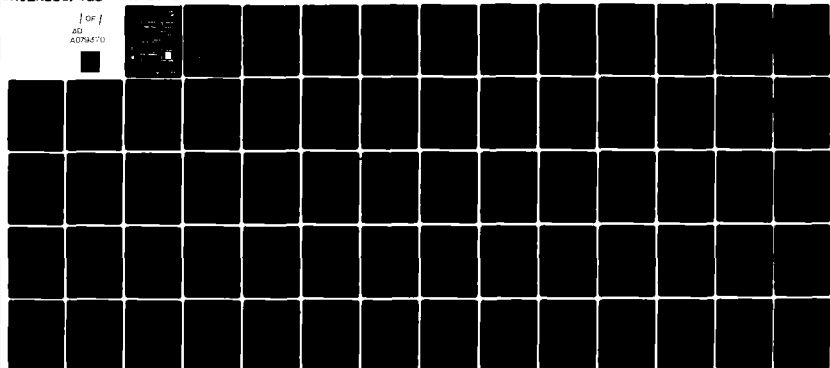


D-0079 370

MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB F/G 9/2  
A COMPACT HARDWARE REALIZATION FOR APPROXIMATE DIVISION.(U)  
OCT 79 A H HUNTOON F19628-80-C-0002  
TN-1979-57 ESD-TR-79-263 NL

UNCLASSIFIED

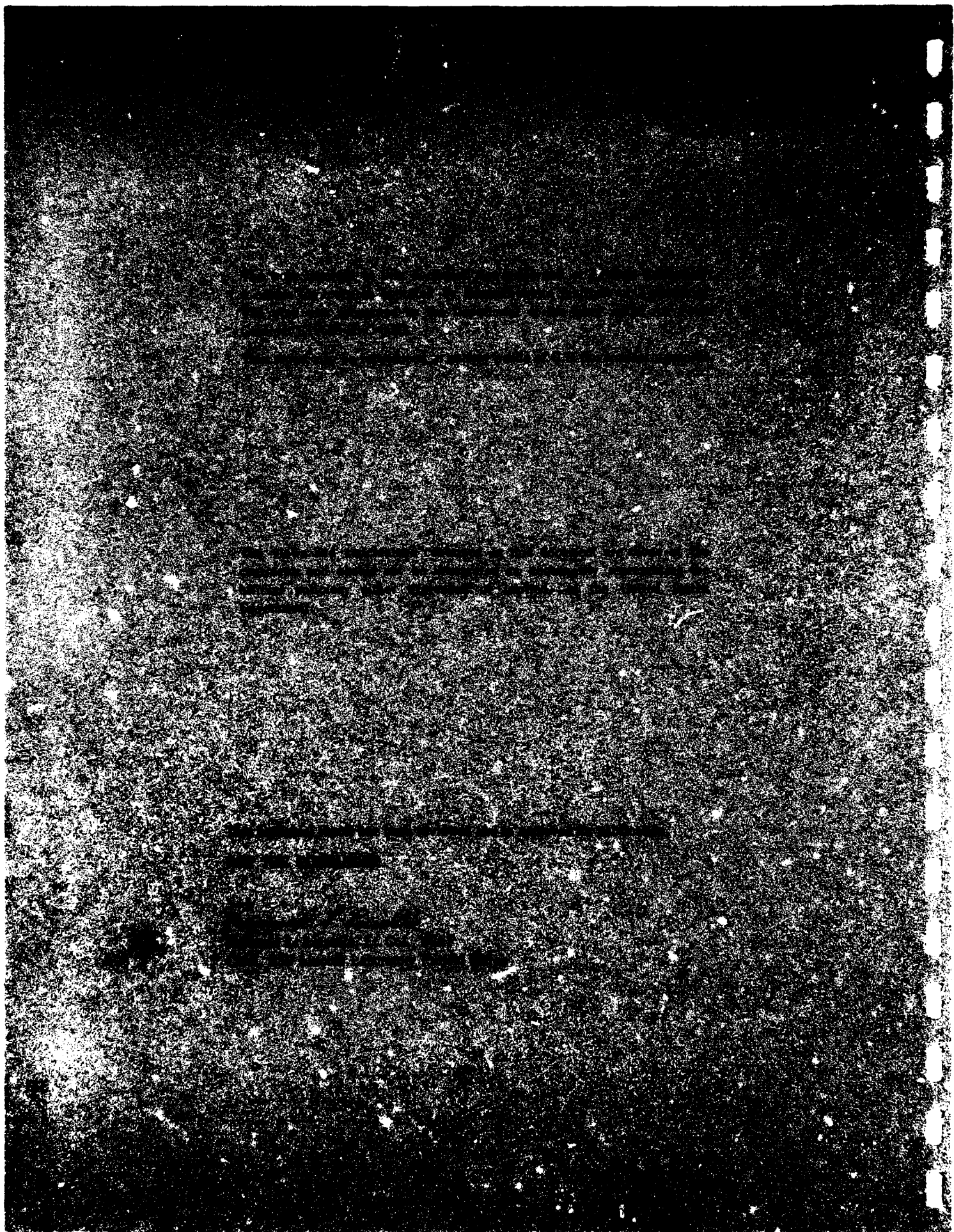
[ OF ]  
20  
AD794570



END  
DATE  
FILMED  
2-80

ODC

ADA 079370



MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
LINCOLN LABORATORY

A COMPACT HARDWARE REALIZATION  
FOR APPROXIMATE DIVISION

A. H. HUNTOON  
*Group 27*

TECHNICAL NOTE 1979-57

9 OCTOBER 1979

Approved for public release; distribution unlimited.

LEXINGTON

MASSACHUSETTS

## ABSTRACT

The design and implementation of hardware for approximate digital division is described. Using standard, commercially-available T<sup>2</sup>L components, a compact divider has been developed which is well-suited for use within 5 MHz systems having three-state busses. After forming double-length denominator reciprocals by table look-up to high accuracies, single and extended-precision quotients are made available by consecutive numerator multiplications using a single-chip array multiplier.

Accession for  
 RESEARCH  
 DEPT. OF  
 HEALTH  
 AND  
 HUMAN SERVICES  
 NATIONAL  
 ARCHIVES  
 COLLEGE PARK, MARYLAND

## CONTENTS

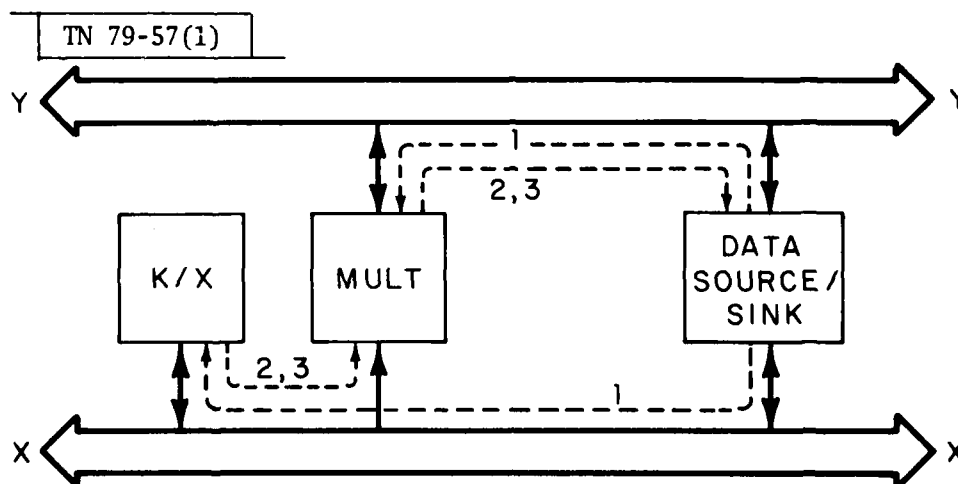
|                              |     |
|------------------------------|-----|
| Abstract                     | iii |
| I. Introduction              | 1   |
| II. K/X Approximation        | 4   |
| III. K/X Quantization        | 11  |
| IV. Address Mapping          | 18  |
| V. Implementation            | 23  |
| A. PROM and PMUX Programming | 26  |
| B. Reciprocal Function Modes | 28  |
| C. Multiplier Modes          | 30  |
| VI. Results                  | 32  |
| VII. Summary                 | 35  |
| APPENDIX A                   | 36  |
| APPENDIX B                   | 40  |
| APPENDIX C                   | 42  |
| APPENDIX D                   | 44  |
| APPENDIX E                   | 52  |
| APPENDIX F                   | 54  |
| APPENDIX G                   | 62  |

## I. INTRODUCTION

For a large number of digital signal processing algorithms, it has generally been true that the infrequent need for a full-precision 2's complement divide capability has resulted either in this function being implemented in software, or else denominator reciprocal values being prestored in memory for later add-shift numerator multiplication to achieve "real time" division. For those applications wherein a fast, compact hardware divide capability is essential, the choices have been somewhat limited. For example, one may implement either a successive-approximation or a non-restoring division algorithm using one or two handfals of MSI T<sup>2</sup>L devices or utilize a single-chip multiplier/divider (e.g., the MMI #67516), but in either case the expected quotient calculation time exceeds two microseconds.

A table look-up method is described in this paper for forming the approximate reciprocal of any 16-bit 2's complement input using 12 MSI T<sup>2</sup>L chips (plus SSI gates). Access time is slightly more than 100 ns. When used together with a single-chip 16 x 16 array multiplier, single-precision quotient calculations require about 250 ns.

Figure 1 illustrates how the scaled reciprocal function ( $K/X$ ) may be used within a two-bus  $\mu C$  structure ( $K = 2^{-15}$ ).



| <u>INST No.</u> | <u>ACTION</u>                            |
|-----------------|------------------------------------------|
| 1               | $K/X \leftarrow X, MULT \leftarrow Y$    |
| 2               | $MULT \leftarrow K/X, Y \leftarrow MULT$ |
| 3               | $MULT \leftarrow K/X, Y \leftarrow MULT$ |

Fig. 1. Two-bus microcomputer structure showing data flow for single and double-precision 'pseudodivides'.

In the context of a 5 MIPS microcomputer currently being developed, nominal 200 ns instruction times provide generous overhead allowances for data bus transfers and control delays, while expanding the total allotted time for reciprocal estimation and multiplication to 400 ns. All functional blocks shown in Figure 1 communicate via three-state data buses X and Y, where K/X is equipped with an input register, the multiplier is fully registered on both input and output ports, and the data source/sink is intended to represent one of several possible two-ported data elements, such as a data memory or CPE register file. After operands X and Y are transferred and K/X is formed during instruction 1, a single-precision quotient is generated during instruction 2 by loading K/X into the multiplier, doing the multiplication, and delivering the result to the data sink. Instruction 3 accesses the extended-precision portion of the K/X table for multiplication by the same numerator Y to complete the formation of an approximate double-precision quotient.

When the reciprocal table is used for the "pseudodivide" operations described here, scaling factor K is eliminated by exploiting the output select feature (MSP/LSP) of the multiplier to rescale the resultant output,  $\hat{Q} = Y/X$ . A transfer function such as K/X whose hardware is internally registered and equipped with a three-state port may be applied within single-bus structures as well as dedicated (non-programmable) hardware.

## II. K/X APPROXIMATION

There is ample incentive to minimize the required amount of table storage space for a reciprocal function which will accommodate 16-bit signed 2's complement inputs. Even a single-precision (16-bit) unsampled table requires  $2^{20}$  bits of storage or sixty-four 16K-bit programmable read-only memory devices, apart from support circuitry. By imposing a reasonable error constraint, we can piecewise-approximate the K/X curve and get a dramatic table memory (PROM) saving. Thus, assuming for now that output data precision is not an issue (i.e., assuming "infinite precision"), we seek to approximate K/X with closely spaced points in the region of steepest slope (smallest X), but to use progressively more sparse spacing of points as the slope diminishes and  $|X|$  approaches one. For the moment, let us consider only the positive region of the function  $R = K/X$ . Figure 2 shows the alternate point sampling that would be likely within a region of steeper slope.

Reduction of the required number of table entries involves 1) the choice of compromise K/X values, and 2) the choice of progressively wider K/X sample spacing as X increases from decimal zero to one. A PROM table addressing algorithm is needed which will compress an unsampled,  $2^{16}$ -word table into a much smaller one. For logical simplicity, let the address quantization be binary-weighted.

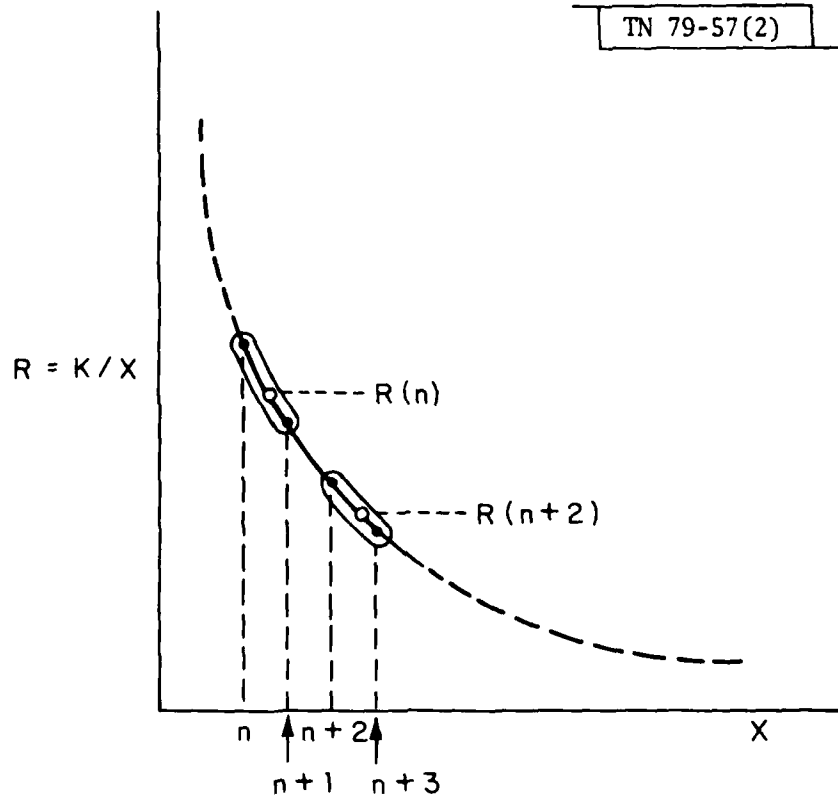


Fig. 2. Illustration of alternate point sampling for  $R = K/X$  ( $K = 2^{-15}$ ,  $X = n \cdot 2^{-15}$ ).

For the positive region, assuming 2's complement fractional inputs, define index  $n = 0 \rightarrow 32767$ ,  $X = n \cdot 2^{-15}$ , and  $K = 2^{-15}$  such that

$$R = K/X = 2^{-15}/n \cdot 2^{-15} = 1/n \quad \left| \begin{array}{l} 32767 \\ n=0 \end{array} \right. \quad (1)$$

If the quantization factor ( $\Delta$ ) represents the number of contiguous  $X$  points spanned for a given table address,  $E_A$  denotes normalized approximation error, and  $\hat{R}$  is the approximate value for  $R$  over the same span, then for uniform normalized peak error,

$$E_A = (1/n - \hat{R}) / (1/n) = [\hat{R} - 1/(n + \Delta - 1)] / [1/(n + \Delta - 1)] \quad (2)$$

from which

$$\hat{R} = 2/[2n + (\Delta - 1)] \quad (3)$$

and

$$E_A = (\Delta - 1) / [2n + (\Delta - 1)] \quad (4)$$

Reciprocal function  $K/X$  can be organized into several neighboring zones, each having its own unique  $\Delta$ . As  $X = n \cdot 2^{-15}$  increases from  $n = 1$ , this first zone (#1) requires that  $\Delta = 1$  (from (4),  $\Delta = 2$  would imply that  $E_A = 0.33$  at  $n = 1$ , an unacceptably large error). Zone 1 is somewhat special in that  $\Delta = 1$  results in a 1:1 relationship between table addresses and data, and no approximation error exists. The end point for zone 1 may be deduced by calculating the beginning value

of  $n$  for zone 2, subject to an error constraint. From (4), for

$$E_A \leq (\Delta-1)/[2n+(\Delta-1)], \quad n \geq (1-E_A)(\Delta-1)/2E_A \quad (5)$$

Thus if we require  $E_A \leq 1/2\%$ , then  $n \geq 100$ . Using the nearest larger binary value, choose  $n = 128$  as the starting point of zone 2.

To minimize the number of table entries as  $X = n \cdot 2^{-15}$  increases toward 1, we wish to increase  $\Delta$  to  $2\Delta$  at the earliest opportunity. For uniform peak error, the zone boundary can be determined by solving (4) for  $n$  when  $\Delta = 2\Delta$ , where  $n_i$  and  $n_{i+1}$  are the first address points in zones  $i$  and  $i+1$ , respectively:

$$E_A = (\Delta_i-1)/[2n_{i+1}+(\Delta_i-1)] = (2\Delta_i-1)/[2n_{i+1} + (2\Delta_i-1)] \quad (6)$$

$$(n_{i+1})/n_i = (2\Delta_i-1)/(\Delta_i-1) \quad (7)$$

The starting point of zone 3 ( $\Delta = 4$ ) determines the end point of zone 2, and from (7) is  $n_{i=3} = 128 (4-1)/(2-1) = 384$ . Equation 7 determines the beginning points of subsequent zones in a similar manner, with uniform peak error throughout, predetermined by the starting point of zone 2. Because zone 2 begins at  $n = 128$ , universal  $E_{A_{\max}} = 0.00389$ . Table 1 shows the range of  $n$  values for the 9 zones needed to generate the positive portion of an approximate  $K/X$  function. Fortunately, zone 9 may be merged with zone 8 by letting the last stored  $\hat{R}$

TABLE 1  
NUMBER RANGE AND QUANTIZATION FACTORS ( $\Delta$ )  
FOR A 9-ZONE APPROXIMATE RECIPROCAL FUNCTION

| n                           | $\Delta$ | Zone | S           | Input X* |    |    |    |    |   |   |   |   |   |   |   |   |   |   |   |
|-----------------------------|----------|------|-------------|----------|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|
|                             |          |      |             | 14       | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |   |
| 0 → 127                     | 1        | 1    | 0<br>⋮<br>0 | 0        | 0  | 0  | 0  | 0  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 128 → 383                   | 2        | 2    | 0<br>⋮<br>0 | 0        | 0  | 0  | 0  | 0  | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 384 → 895                   | 4        | 3    | 0<br>⋮<br>0 | 0        | 0  | 0  | 0  | 0  | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 896 → 1919                  | 8        | 4    | 0<br>⋮<br>0 | 0        | 0  | 0  | 0  | 0  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1920 → 3967                 | 16       | 5    | 0<br>⋮<br>0 | 0        | 0  | 0  | 0  | 1  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3968 → 8063                 | 32       | 6    | 0<br>⋮<br>0 | 0        | 0  | 0  | 1  | 1  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8064 → 16255                | 64       | 7    | 0<br>⋮<br>0 | 0        | 0  | 1  | 1  | 1  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16256 → 32639               | 128      | 8    | 0<br>⋮<br>0 | 0        | 1  | 1  | 1  | 1  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32640 → 32767               | 256      | 9    | 0<br>⋮<br>0 | 1        | 1  | 1  | 1  | 1  | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *2's complement, fractional |          |      |             |          |    |    |    |    |   |   |   |   |   |   |   |   |   |   |   |

value span  $n = 32512 \rightarrow 32767$ ,  $\Delta = 256$ . Each zone requires 128 storage locations, and the 8-zone arrangement reduces the needed storage by a factor of 32 with respect to unsampled storage. For both positive and negative regions of the function  $K/X$ , with approximation error of less than 1/2% and a data wordlength of 16 bits, 32K bits of PROM table storage is sufficient.

Figure 3 shows the function  $R(x)$  (at somewhat distorted scale) for which the approximate function  $\hat{R}(x)$  is to be computed and stored in PROM tables. The intent here is to illustrate the numerical range of interest, and to take note of departures from the ideal curve attributable to 2's complement asymmetry along with a limit when  $X = 0$ . For the positive region,  $X$  ranges from 0 to  $1-2^{-15}$ , with  $R(0)$  and  $R(2^{-15})$  constrained to the largest representable positive fraction,  $1-2^{-15}$ . For the negative region,  $X$  ranges from  $-2^{-15}$  to  $-1$ , and  $R(x)$  is the straightforward scaled reciprocal. Excluding  $R(0)$  and  $R(2^{-15})$ , the asymmetry about the origin may be expressed as

$$R(A^+) = -R(A^+-1), \quad (8)$$

where  $A^+$  is any given address within the positive region.

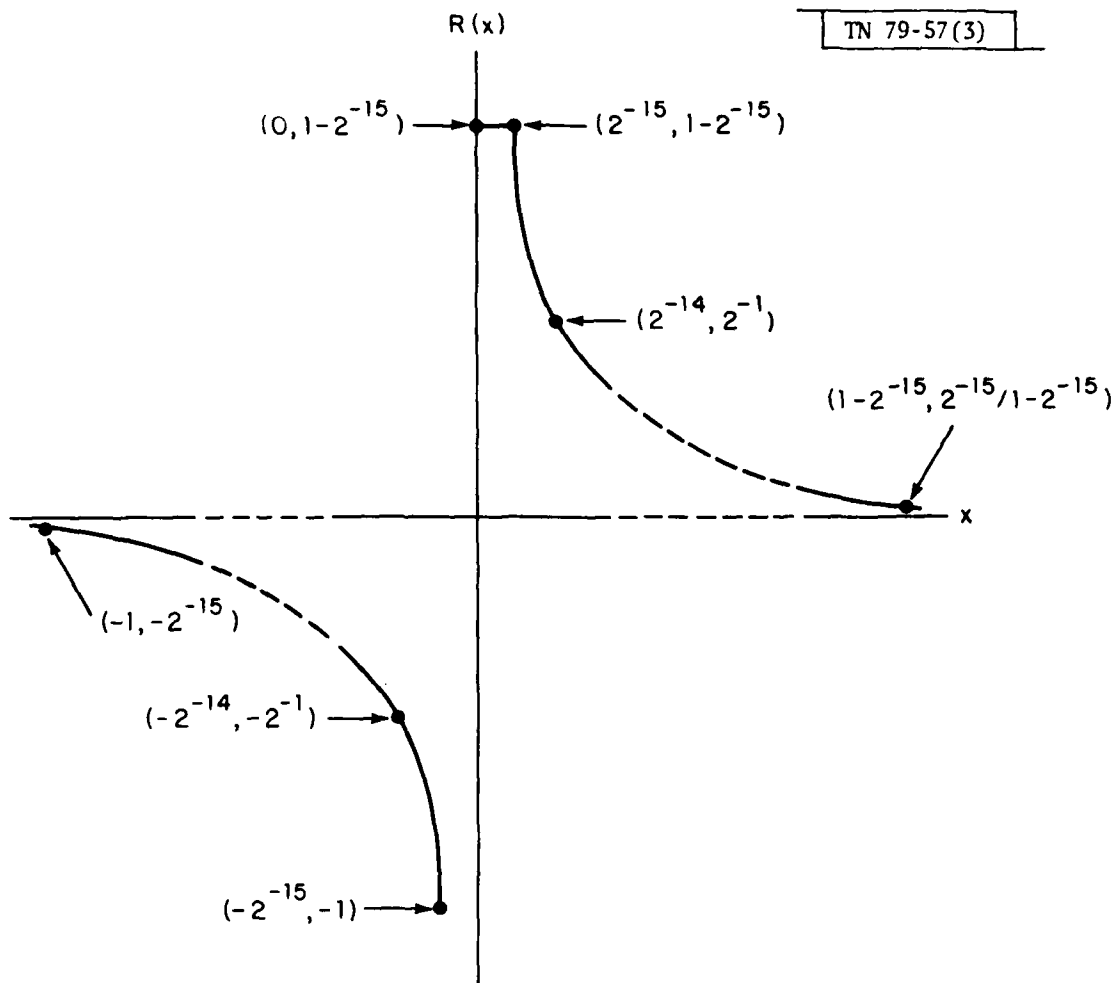


Fig. 3. Numerical range of interest for reciprocal function  $R(x)$ .

### III. K/X QUANTIZATION

It is important to consider the accuracy limits of  $R = K/X$ , i.e., the precision with which one may represent the scaled reciprocal. Let us define the normalized quantization error as

$$E_Q = (R - R_Q)/R, \quad (9)$$

where  $R$  is the exact (infinite precision) value of  $K/X$  and  $R_Q$  is its quantized value. As indicated in Fig. 4, the absolute amount of quantization error in a single-precision  $R$  table may approach the value of the LSB ( $2^{-15}$ ). But on a percentage basis, this error has the least impact when the absolute value of  $X$  is small, i.e., when  $R$  is large.

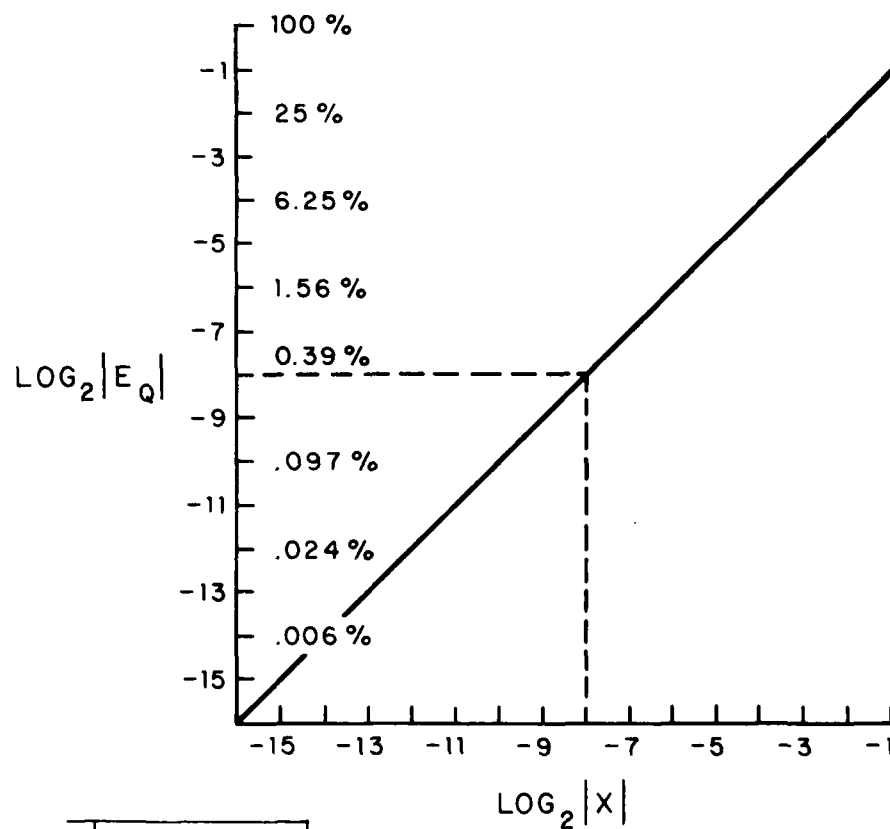
Tables 2 and 3 indicate the expected worst-case total errors (approximation plus quantization) for single and double-precision realizations, respectively. Recalling from (1) that

$$R = K/X = 1/I \left| \begin{array}{l} I_{\max} \\ I=0 \end{array} \right. \quad (10)$$

where

$$X = I \cdot 2^{-15}, \quad (11)$$

both tables show normalized error as a function of  $I_{\max}$  and  $X_{\max}$ , together with the input values ( $I_{pk}$  and  $X_{pk}$ ) at which total error  $E_T$  is maximum. As expected, Table 2 shows approxi-



TN 79-57(4)

Fig. 4. Maximum quantization error ( $E_Q$ ) versus denominator ( $x$ ) magnitude for single-precision reciprocal,  $R$ .

TABLE 2  
ERROR ANALYSIS FOR SINGLE-PRECISION K/Y

| IMAX  | IPK   | XMAX        | XPK         | EA       | EQ       | ET       |
|-------|-------|-------------|-------------|----------|----------|----------|
| 1     | 1     | 0.0         | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 2     | 1     | 0.000030518 | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 4     | 4     | 0.000091553 | 0.000122070 | 0.0      | 0.000061 | 0.000061 |
| 8     | 6     | 0.000213623 | 0.000183105 | 0.0      | 0.000091 | 0.000091 |
| 16    | 12    | 0.000457764 | 0.000366211 | 0.0      | 0.000305 | 0.000305 |
| 32    | 30    | 0.000946045 | 0.000915527 | 0.0      | 0.000824 | 0.000824 |
| 64    | 59    | 0.001922607 | 0.001800537 | 0.0      | 0.001709 | 0.001709 |
| 128   | 114   | 0.003875732 | 0.003479004 | 0.0      | 0.003387 | 0.003387 |
| 256   | 247   | 0.007781982 | 0.007537842 | 0.002028 | 0.007005 | 0.009033 |
| 512   | 489   | 0.015594482 | 0.014923096 | 0.003064 | 0.014025 | 0.017090 |
| 1024  | 993   | 0.031219482 | 0.030303955 | 0.003516 | 0.027734 | 0.031250 |
| 2048  | 1937  | 0.062469482 | 0.059112549 | 0.003859 | 0.050829 | 0.054687 |
| 4096  | 3649  | 0.124969482 | 0.111358643 | 0.002051 | 0.107323 | 0.109375 |
| 8192  | 6561  | 0.249969482 | 0.200225830 | 0.002357 | 0.196862 | 0.199219 |
| 16384 | 10945 | 0.499969482 | 0.334014893 | 0.002870 | 0.329161 | 0.332031 |
| 32768 | 16385 | 0.999969482 | 0.500030518 | 0.003861 | 0.496139 | 0.500000 |

TABLE 3  
ERROR ANALYSIS FOR DOUBLE-PRECISION K/X

| IMAX  | IPK   | XMAX        | XPK         | EA       | EQ       | ET       |
|-------|-------|-------------|-------------|----------|----------|----------|
| 1     | 1     | 0.0         | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 2     | 1     | 0.000030518 | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 4     | 1     | 0.000091553 | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 8     | 1     | 0.000213623 | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 16    | 1     | 0.000457764 | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 32    | 1     | 0.000946045 | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 64    | 1     | 0.001922607 | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 128   | 1     | 0.003875732 | 0.000030518 | 0.0      | 0.0      | 0.0      |
| 256   | 129   | 0.007781982 | 0.003936768 | 0.003891 | 0.000000 | 0.003891 |
| 512   | 129   | 0.015594482 | 0.003936768 | 0.003891 | 0.000000 | 0.003891 |
| 1024  | 897   | 0.031219482 | 0.027374268 | 0.003891 | 0.000000 | 0.003891 |
| 2048  | 897   | 0.062469482 | 0.027374268 | 0.003891 | 0.000000 | 0.003891 |
| 4096  | 3969  | 0.124969482 | 0.121124268 | 0.003891 | 0.000001 | 0.003892 |
| 8192  | 8065  | 0.249969482 | 0.246124268 | 0.003891 | 0.000003 | 0.003894 |
| 16384 | 8065  | 0.499969482 | 0.246124268 | 0.003891 | 0.000003 | 0.003894 |
| 32768 | 32513 | 0.999969482 | 0.992218018 | 0.003906 | 0.000000 | 0.003906 |

mation error ( $E_A$ ) to be zero for zone 1, and constraining  $X$  to be no larger than  $2^{-8}$  results in quantization error ( $E_Q$ ) of less than 0.4% as indicated in Fig. 4. Note also that while approximation error is minimal throughout the usable input range, quantization error rapidly becomes unacceptable with increasing  $X$ . However, for those applications requiring good accuracy over the full dynamic range of  $X$ , Table 3 shows that extended precision approximation of  $K/X$  yields improved results. Simply stated, a single-precision approximation to  $K/X$  can resolve to within  $2^{-15}$  (the LSB), whereas a double-precision approach is accurate to within  $2^{-30}$ . The penalty for extended precision is an additional multiply operation when doing a "pseudodivide".

The Fortran programs for generating Tables 2 and 3 are shown in Appendix A.

For single-precision results to have tolerable limits on error, some constraints must be imposed on the range of allowable divider inputs shown in Fig. 4. If, after compensating for scaling we say that 16-bit quotient

$$Q = Y/X \quad , \quad (12)$$

and avoidance of overflow requires that

$$|Y| < |X| \quad , \quad (13)$$

then for quantization error  $E_Q < 0.4\%$  as chosen for this particular design,

$$|X| \leq 2^{-8} \quad (14)$$

and

$$|Q| \geq 2^{-7} \quad (15)$$

In the context of a given  $\mu C$  program requiring a fast divide, a priori knowledge and maintenance of the relative magnitudes of X and Y can permit straightforward, fast generation of single-precision quotients. If, however, a number of instructions are expended in first shifting divider operands, the hardware divide feature may have less appeal. For this reason, the K/X address map function has been designed to accommodate the full denominator (X) dynamic range, and permits the addressing of a double-precision (32-bit) K/X table simply by including two additional PROMs. The divider hardware then provides a single-precision  $\hat{Q}$  in two instructions (400 ns), and an extended precision  $\hat{Q}$  with

one additional 200 ns instruction, subject only to the initial requirement for  $|Y| < |X|$ . If, on the other hand, we had elected to limit the input range to  $|X| \leq 2^{-8}$  (see Fig. 4), a simple 256 x 16 PROM table with no address mapping would have sufficed.

#### IV. ADDRESS MAPPING

Figure 5 is a simplified block diagram showing the generation of  $K/X$  by cascading address map and table memory sections. The principal design task here is the derivation of a mapping function whose implementation will have reasonable speed and a minimal number of integrated circuits. By examining the 2's complement representation of  $X$  as described in Table 1, the logical relationships for address mapping may be deduced and minimized. Although only numbers for the positive region are considered here, the same mapping logic may be used for the negative region after inverting input word  $X$  under control of sign bit  $X_s$ . Figure 5 shows an 11-bit table address, where bit  $A_{10}$  determines the table region (+, -), address bits  $A_9$  through  $A_7$  control zone selection, and bits  $A_6$  through  $A_0$  constitute the zone address.

Let seven intermediate variables be defined as follows:

$$\begin{aligned}
 C_1 &= \bar{X}_8 X_7 + X_8 \bar{X}_7 \\
 C_2 &= \bar{X}_9 X_8 X_7 + X_9 (\bar{X}_8 + \bar{X}_7) \\
 C_3 &= \bar{X}_{10} X_9 X_8 X_7 + X_{10} (\bar{X}_9 + \bar{X}_8 + \bar{X}_7) \\
 C_4 &= \bar{X}_{11} X_{10} X_9 X_8 X_7 + X_{11} (\bar{X}_{10} + \bar{X}_9 + \bar{X}_8 + \bar{X}_7) \\
 C_5 &= \bar{X}_{12} X_{11} X_{10} X_9 X_8 X_7 + X_{12} (\bar{X}_{11} + \bar{X}_{10} + \bar{X}_9 + \bar{X}_8 + \bar{X}_7) \\
 C_6 &= \bar{X}_{13} X_{12} X_{11} X_{10} X_9 X_8 X_7 + X_{13} (\bar{X}_{12} + \bar{X}_{11} + \bar{X}_{10} + \bar{X}_9 + \bar{X}_8 + \bar{X}_7) \\
 C_7 &= X_{14} X_{13} X_{12} X_{11} X_{10} X_9 X_8 X_7
 \end{aligned} \tag{16}$$

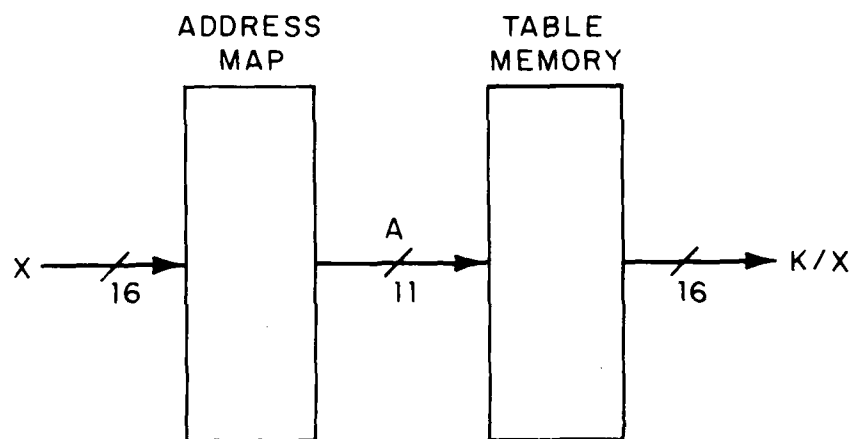


Fig. 5. Simplified reciprocal function generator block diagram.

Then

$$A_{10} = X_s, \quad (17)$$

and zone select bits are

$$\begin{aligned} A_9 &= \bar{X}_{14}\bar{X}_{13}\bar{X}_{12}C_4 + \bar{X}_{14}\bar{X}_{13}C_5 + \bar{X}_{14}C_6 \\ &\quad + \bar{X}_{14}X_{13}X_{12}X_{11}X_{10}X_9X_8X_7 + X_{14} \\ A_8 &= \bar{X}_{14}\bar{X}_{13}\bar{X}_{12}\bar{X}_{11}\bar{X}_{10}C_2 + \bar{X}_{14}\bar{X}_{13}\bar{X}_{12}\bar{X}_{11}C_3 \\ &\quad + \bar{X}_{14}C_6 + \bar{X}_{14}X_{13}X_{12}X_{11}X_{10}X_9X_8X_7 + X_{14} \\ A_7 &= \bar{X}_{14}\bar{X}_{13}\bar{X}_{12}\bar{X}_{11}\bar{X}_{10}\bar{X}_9C_1 + \bar{X}_{14}\bar{X}_{13}\bar{X}_{12}\bar{X}_{11}C_3 \\ &\quad + \bar{X}_{14}\bar{X}_{13}C_5 + \bar{X}_{14}X_{13}X_{12}X_{11}X_{10}X_9X_8X_7 + X_{14} \end{aligned} \quad (18)$$

The required zone address bits are shown in Table 4 below.

| TABLE 4<br>ZONE ADDRESS BITS AS A FUNCTION OF ZONE<br>NUMBER AND INPUT VARIABLES X,C |                |                |                |                |                |                |                |                                    |
|--------------------------------------------------------------------------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|------------------------------------|
| Zone<br>bits \ Zone                                                                  | 1              | 2              | 3              | 4              | 5              | 6              | 7              | 8,9                                |
| A <sub>6</sub>                                                                       | X <sub>6</sub> | $\bar{X}_7$    | C <sub>1</sub> | C <sub>2</sub> | C <sub>3</sub> | C <sub>4</sub> | C <sub>5</sub> | (C <sub>6</sub> + C <sub>7</sub> ) |
| A <sub>5</sub>                                                                       | X <sub>5</sub> | X <sub>6</sub> | $\bar{X}_7$    | C <sub>1</sub> | C <sub>2</sub> | C <sub>3</sub> | C <sub>4</sub> | (C <sub>5</sub> + C <sub>7</sub> ) |
| A <sub>4</sub>                                                                       | X <sub>4</sub> | X <sub>5</sub> | X <sub>6</sub> | $\bar{X}_7$    | C <sub>1</sub> | C <sub>2</sub> | C <sub>3</sub> | (C <sub>4</sub> + C <sub>7</sub> ) |
| A <sub>3</sub>                                                                       | X <sub>3</sub> | X <sub>4</sub> | X <sub>5</sub> | X <sub>6</sub> | $\bar{X}_7$    | C <sub>1</sub> | C <sub>2</sub> | (C <sub>3</sub> + C <sub>7</sub> ) |
| A <sub>2</sub>                                                                       | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | X <sub>5</sub> | X <sub>6</sub> | $\bar{X}_7$    | C <sub>1</sub> | (C <sub>2</sub> + C <sub>7</sub> ) |
| A <sub>1</sub>                                                                       | X <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | X <sub>5</sub> | X <sub>6</sub> | $\bar{X}_7$    | (C <sub>1</sub> + C <sub>7</sub> ) |
| A <sub>0</sub>                                                                       | X <sub>0</sub> | X <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | X <sub>5</sub> | X <sub>6</sub> | ( $\bar{X}_7$ + C <sub>7</sub> )   |

The foregoing expressions are the result of examining the relationships between binary input values (X) and required address values (A), with particular emphasis upon the X states which mark the zone boundaries. Simply stated, knowledge of the zone within which any input X falls determines 1) the page in table memory containing the appropriate K/X value, and 2) the modulus by which changes in X advance the address count. Consider, for example, the bottom row of table 4. As we proceed along the K/X curve from zone 1 to zone 8, the table address (with LSB =  $A_0$ ) is incremented modulo 1,2,4,8,... etc., as expected.

## V. IMPLEMENTATION

The schematic diagram of Fig. 6 is the result of considering several approaches to minimization of the address mapping logic, using standard components. While intuition initially suggested the use of barrel shifters or FPLAs, the use of fast PROMs in conjunction with programmable multiplexers yielded the most compact design. The zone address pattern of Table 4 is suggestive of a large multiplexer, and indeed the PMUXs of Fig. 6 permit generation of all zone address bits except  $A_6$ . PROMs A and B provide all the variables of Eqs. 16 and 18, plus  $A_6$ . The 16K-bit table storage PROMs represent the best density/speed combination currently available, and provide needed three-state outputs. The 25LS2519 input registers are particularly well suited to this application because of their output invert feature. While the address mapping logic is configured to accept binary inputs in magnitude format, its usefulness is extended by the input register's ability to supply the 1's complement of negative inputs under sign bit control. This same sign bit ( $X_{15}$ ) selects the table region (+,-) via address bit  $A_{10}$ .

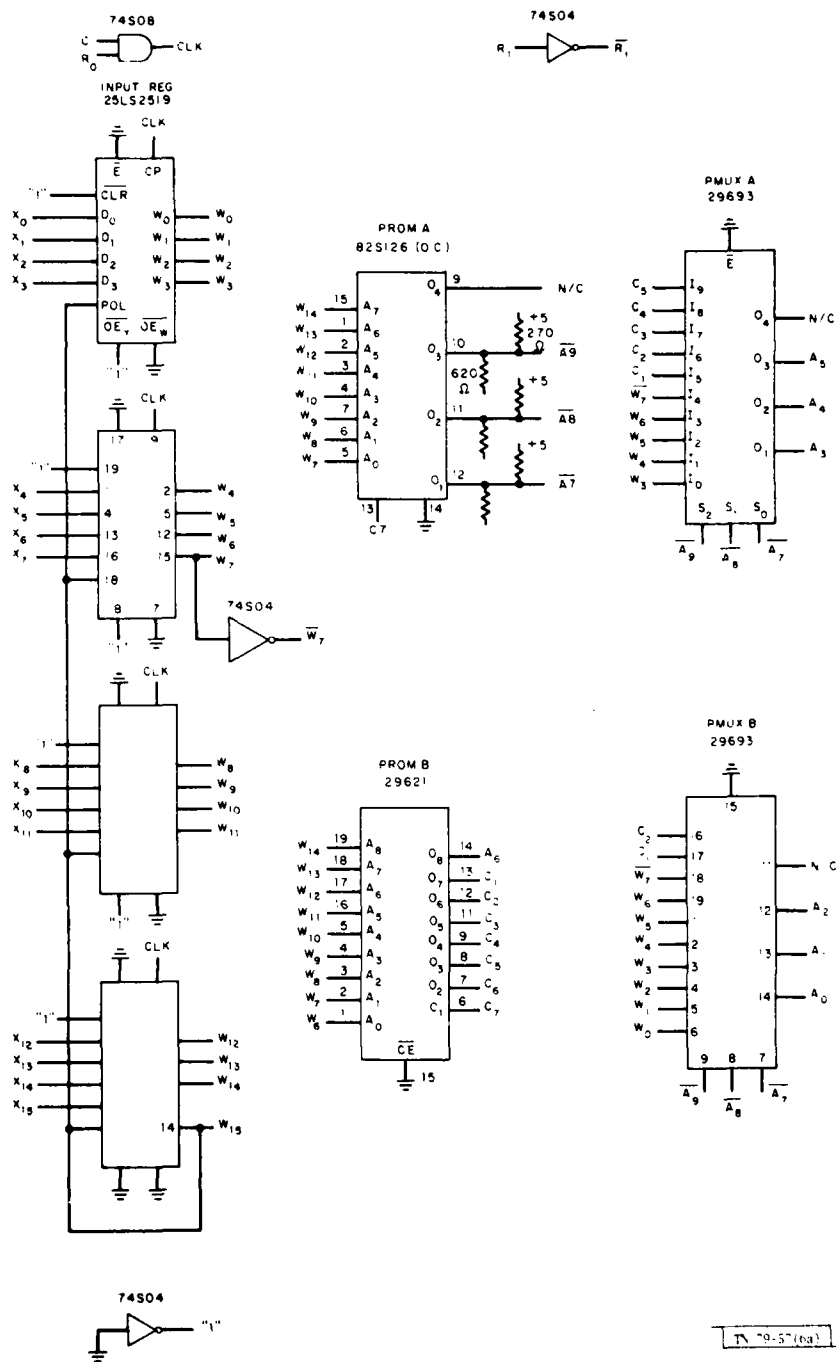


Fig. 6. Logic diagram for double-precision approximate reciprocal function.

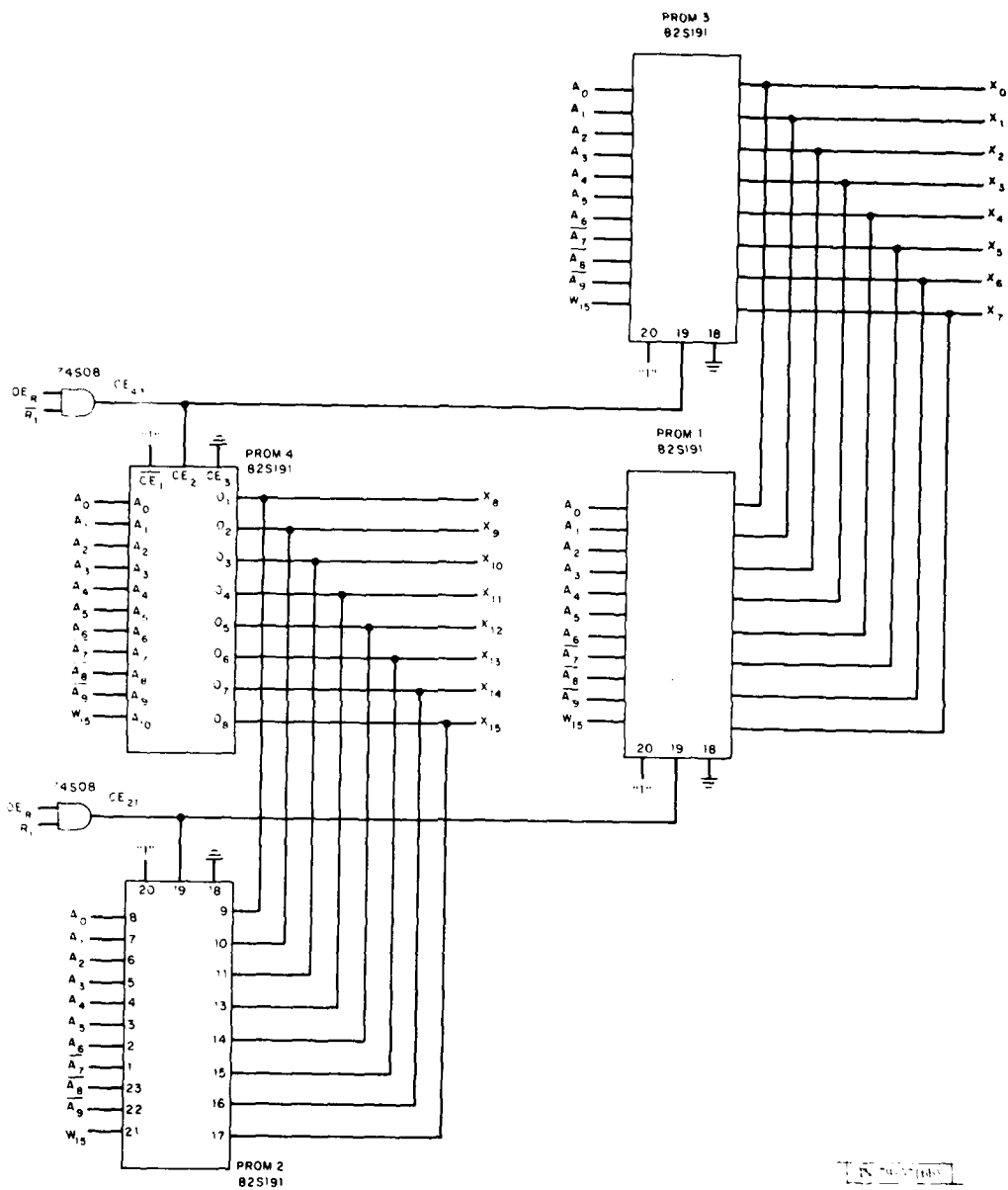


Fig. 6. Continued.

One additional hardware trick is needed to realize the zone 8,9 column of Table 4. As mentioned earlier, a residual zone 9 ( $32640 \leq n \leq 32767$ ) exists for which intermediate variable  $C_7$  is uniquely true. This small region can be merged with zone 8 with no discernible error penalty. In Table 4, the presence of the  $C_7$  terms within the zone 8,9 column is the assertion of zone 9. However, because PMUX input ports are already fully utilized (so much so that  $A_6$  is generated via PROM B instead of PMUX A or B), another method for including the contribution of  $C_7$  must be used. This is done by storing zone select bits ( $A_9 \rightarrow A_7$ ) within PROM A in their complemented form, and applying  $C_7$  to PROM A as a chip enable. The programming of PMUXs and table PROMs is then done in accordance with  $\bar{A}_9, \bar{A}_8, \bar{A}_7$  instead of  $A_9, A_8, A_7$ , and whenever  $C_7$  is asserted, zone selection is constrained to zone 8 via the PROM A chip enable. Zone address bits  $A_0$  through  $A_6$  are unaffected.

#### A. PROM and PMUX Programming

Referring to Fig. 6, PROM A is required for zone selection within the table PROMs, whereas PROM B is used principally for the development of intermediate address mapping variables. With the exception of bit  $A_6$ , PMUXs A and B generate the table address shown in Table 4. The requisite data patterns to be stored were written in accordance with Tables 1 and 4 and Eqs. 16 and 18, and are shown in Appendix B. For purposes of

electrical programming, the 29693 PMUX is viewed as a 256 x 4 PROM.

From Fig. 6, Table PROMs 4 and 3 are selected for single-precision readouts, while PROMs 2 and 1 contain extended-precision data. A Fortran program called TABLE1 (Appendix C) calculates the decimal, integer, and hex values required for the single-precision PROM pair, based on parameters such as number of zones and the quantization factor within each zone. TABLE1 outputs four disk files entitled BYTE43 DEC, BYTE43 INT, BYTE4 HEX, and BYTE3 HEX. The first two outputs provide fractional and integer decimal representations of the entire single-precision list for checking purposes. Because these files are somewhat voluminous, they are not included in the appendix. The remaining two files are the result of splitting the integer file into two pieces and adopting the hexadecimal format, such that the resultant tables are directly useful as data files for programming the two byte-wide memories, PROMs 3 and 4. These listings are shown in Appendix D.

Fortran program TABLE2 (Appendix E) is nearly the same as TABLE1, except that its purpose is to generate the data patterns needed for extended-precision PROMs 2 and 1. It does so by calculating to high precision all the table values needed for double-precision, then deducting the values already stored in PROMs 4 and 3, and using the residue as data for PROMs 2

and 1. The TABLE2 output files are labeled BYTE21 DEC, BYTE21 INT, BYTE2 HEX, and BYTE1 HEX. The latter two are shown in Appendix F. As a point of interest, program TABLE2 resolves each table entry to within  $2^{-30}$  instead of  $2^{-31}$ , because a sign bit must be included within both the most and least significant 16-bit words made available on the K/X data bus.

The array multiplier to be used in conjunction with the reciprocal function unit for doing 'pseudodivides' is the TRW model MPY-16HJ. Figure 7 shows the block diagram for this single-chip LSI device which can typically provide 32-bit products within 125 ns. Two particularly useful features in the present context are fully-registered inputs and outputs, plus the availability of three-state outputs. Further reference to these features will be made when defining multiplier operating modes.

#### B. Reciprocal Function Modes

Control of the reciprocal function unit involves some simple choices relating to 1) the status of the three-state port (input vs output), 2) whether or not the data register stores new information, and 3) single or extended-precision output (MSR or LSR). Table 5 below outlines some basic modes based upon the availability of an input clock and control bits  $R_0$ ,  $R_1$ , and  $OE_R$ .

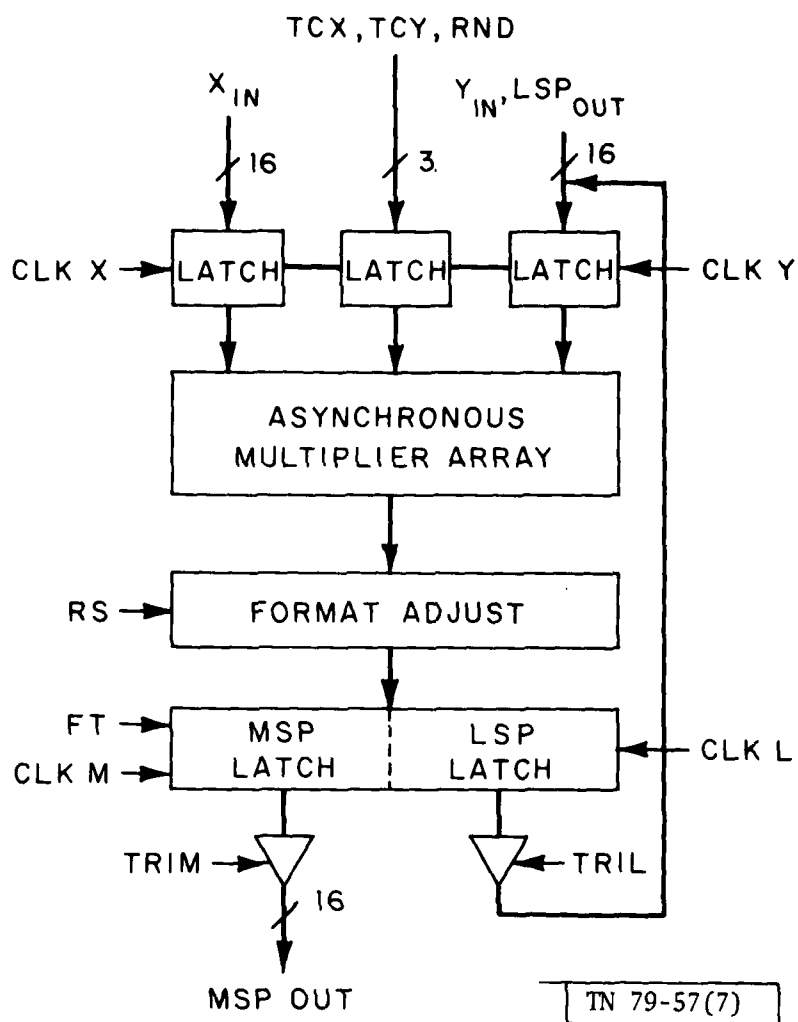


Fig. 7. TRW MPY-16 HJ 16 x 16 bit parallel multiplier block diagram.

| TABLE 5<br>ESSENTIAL CONTROL STATES FOR RECIPROCAL FUNCTION UNIT |       |       |     |           |           |           |
|------------------------------------------------------------------|-------|-------|-----|-----------|-----------|-----------|
| $OE_R$                                                           | $R_1$ | $R_0$ | CLK | $CE_{43}$ | $CE_{21}$ | Mode      |
| 1                                                                | 0     | 0     | 0   | 1         | 0         | Read MSR  |
| 1                                                                | 1     | 0     | 0   | 0         | 1         | Read LSR  |
| 0                                                                | X*    | 0     | 0   | 0         | 0         | Hold      |
| 0                                                                | X     | 1     | ↑   | 0         | 0         | Calc/Hold |
| *X = don't care                                                  |       |       |     |           |           |           |

As shown in Fig. 6,  $R_0$  gates an external clock (C) and controls updating of the data register.  $R_1$  determines the MSR/LSR choice when accessing the table PROMs in conjunction with bus control  $OE_R$ . Referring to the three instructions of Fig. 1 for forming approximate quotients, the corresponding reciprocal unit commands in accordance with Table 5 are Calc/Hold, Read MSR, Read LSR.

### C. Multiplier Modes

Given a set of externally-applied clocks and control states, the internal registers and three-state capability of the MPY-16HJ may be used to full advantage for calculating approximate quotients. Table 6 defines several useful multiplier modes. The use of separate input and output clocks ( $C_i$  and  $C_o$ ) is assumed, three-state ports become outputs when TRIM or TRIL are zero, and X denotes "don't care" conditions.

| TABLE 6<br>CONTROL STATES FOR TRW MULTIPLIER |                  |                    |      |      |          |
|----------------------------------------------|------------------|--------------------|------|------|----------|
| CLK <sub>x</sub>                             | CLK <sub>y</sub> | CLK <sub>1,m</sub> | TRIM | TRIL | Mode     |
| C <sub>i</sub>                               | 0                | C <sub>o</sub>     | 1    | 1    | Load X   |
| 0                                            | C <sub>i</sub>   | C <sub>o</sub>     | 1    | 1    | Load Y   |
| C <sub>i</sub>                               | C <sub>i</sub>   | C <sub>o</sub>     | 1    | 1    | Load X,Y |
| X                                            | X                | 0                  | 1    | 1    | Hold     |
| X                                            | X                | 0                  | 0    | 1    | Read MSP |
| X                                            | X                | 0                  | 1    | 0    | Read LSP |

Multiplication occurs on any Load instruction, and the resultant product is retained in the output register(s). The Hold condition retains data in the output registers, and is generally used when no multiplier outputs are driving the bus. The Read operations simply amount to suppressing the output register clocks and enabling the appropriate three-state driver to access either the most or least significant product.

Table 6 depicts only a subset of the total number of unique operations possible using the MPY-16HJ multiplier. An underlying assumption of Fig. 1 is that some of the modes of Table 6 may be concatenated within any given 200 ns nominal microcomputer instruction cycle. In particular, for the three-instruction loop of Fig. 1, the multiplier does Load Y, Load X/Read LSP, and Load X/Read LSP operations while the reciprocal unit is doing Calc/Hold, Read MSR, and Read LSR.

The split multiplier instruction Load X/Read LSP is faster than others such as Load Y/Read LSP because the Y bus is not shared between input and output traffic, and no three-state output enable delay is incurred. Another multiplier feature which may be exploited is the feedthrough provision. If local storage of a product (in the multiplier output register) is not needed, but instead a result is to be stored at some remote point, the FT line on the MPY-16HJ renders the output register transparent, making the multiplier output available earlier.

When doing division, the scale factor (K) associated with the single-precision reciprocal function is effectively removed by shifting the multiplier output by  $2^{+15}$ , i.e., by interpreting the LSP in accordance with MSP binary weights. For extended precision calculations, the reciprocal table output is interpreted as being unscaled.

## VI. RESULTS

Using commercial grade integrated circuits, bench tests have shown the clock-to-output throughput delay of the one reciprocal function built to be no greater than 110 nanoseconds at 25°C. The expected delay value over the commercial temperature range should not exceed 165 ns. For those instances where X has been stored in the K/X holding register for a long time (e.g., when fetching an extended-precision result), access via table PROM chip enable may take as little as 50 ns.

Tests over a 0° to 70°C temperature range indicate that the MPY-16HJ multiplier typically requires 125 ns to calculate 32-bit products.\* The worst-case data sheet specification for non-pipelined multiplies is 175 ns.

In light of the above, the nominal 200 ns interval cited earlier for forming reciprocals and for doing multiplies continues to be a reasonable estimate.

Beyond questions of operating speed, several test cases have been run to verify the numerical validity of the basic design, with particular emphasis on the effects of approximation and quantization errors. Appendix G discusses the results of two test cases, including such issues as table memory zone selection, scaling, and error calculations.

The integrated circuit placement diagram of Fig. 8 is indicative of the physical size of the divider. The layout shown requires about 12 in.<sup>2</sup> of circuit board area.

Total D.C. power dissipation for the divider using available componentry is slightly in excess of 8 watts. This number can be reduced to 5 watts in the future by employing SPROMs which have a fast power-down capability. Raytheon currently offers some of the needed devices which have a built-in power switch, and can potentially save 70% of PROM power.

---

\*The MPY-16HJ has been thoroughly tested at several temperatures using all possible input operand combinations as part of a separate characterization effort.

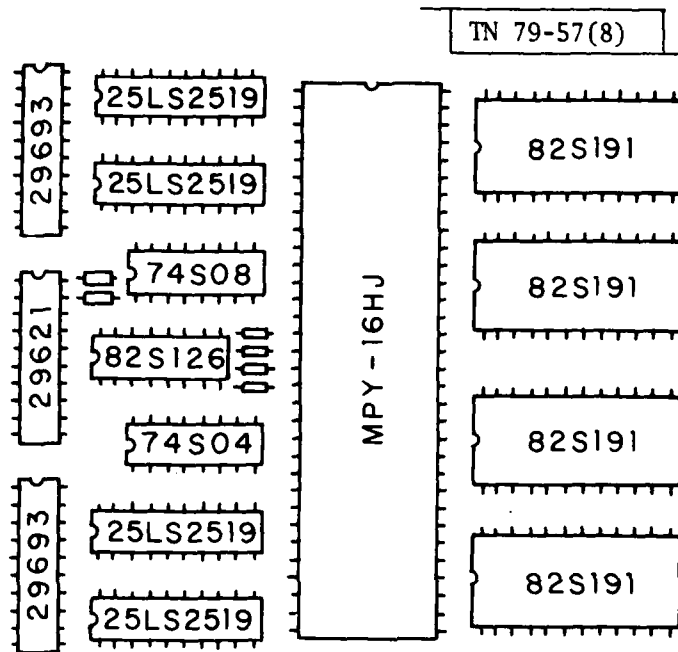


Fig. 8. Integrated Circuit placement diagram for digital divider (actual size).

## VII. SUMMARY

For those applications where fast, approximate division is essential, the reciprocal table look-up/multiplication approach to generating quotients is a viable alternative. The design described in this paper is of tractable size (15 DIPs) and power (8 watts), and provides double-precision outputs having total errors no greater than 0.4%. If produced in significant quantities, the device count could be further reduced through the use of semi-custom integrated circuits, particularly for the address mapping function. Better accuracies may be obtained at the expense of additional table storage. For each halving of the approximation error, twice as many PROM table entries are needed.

The hardware implementation shown requires less than 200 ns for reciprocal formation and for each subsequent multiplication. This speed regime is well matched to the capabilities of bipolar (LSTTL) microcomputers or to dedicated hardware systems running with 5 MHz throughput rates.

APPENDIX A  
 FORTRAN PROGRAMS FOR GENERATING TABLES 3 AND 4

```

REAL*8 R,RA,RAQ,EA,ET,EQ,ERROR,EPEAK,XPEAK,EAP,ETP,EQP
REAL*8 DEL,RDEL,X1,X2
WRITE(6,1)
1 FORMAT(/,1X,'ERROR ANALYSIS FOR APPROXIMATE 1/X',/)
WRITE(6,2)
2 FORMAT(1X,'DATA TABULATED RE MAX INPUT X - SINGLE PRECISION',/)
WRITE(6,3)
3 FORMAT(1X,'EA,EQ,ET=APPROX,QUANT,& TOTAL WORST-CASE ERRORS',/)
WRITE(6,4)
4 FORMAT(3X,'IMAX',4X,'IPK',6X,'XMAX',9X,'XPK',9X,'EA',8X,'EQ',8X,
1 'ET',/)
DO 10 L=1,16
IM=2**(L-1)
XM=(IM-1.)*2.**(-15)
EPEAK=-1.
I1=1
I2=128
J=1
IMAX=2**(L-1)
IEND=128
IF(IMAX.LT.IEND)IEND=IMAX
DO 20 I=1,IEND
IF(I.GT.2)GO TO 21
R=1.-(2.**(-15))
GO TO 22
21 R=1./(I-1)
22 RA=R
CALL QUANT(RA,RAQ)
GO TO 50
20 CONTINUE
IF(I.EQ.IMAX)GO TO 40
15 J=J+1
I3=2*(I2-I1)+1
I1=I2+1
I2=I1+I3
IF(IMAX.LT.I2)I2=IMAX
IF(I2.EQ.32640)I2=32768
DO 30 I=I1,I2
R=1./(I-1)
DEL=2.**(J-1)
IF(I.GE.32513)DEL=256.
IDEL=DEL
RDEL=1./DEL
X1=I/DEL
IX1=I/IDEL

```

ERROR16 FORTRAN

```

      X2=X1-IX1
      IF(X2.NE.RDEL)GO TO 30
      RA=2./(2.*(I-1)+DEL-1.)
      CALL QUANT(RA,RAQ)
      GO TO 50
30  CONTINUE
      IF(I.EQ.IMAX)GO TO 40
      GO TO 15
50  EA=(R-RA)/R
      ET=(R-RAQ)/R
      EQ=ET-EA
      ERROR=ET
      IF(ET.LT.0)ERROR=-ET
      IF(ERROR.LE.EPEAK)GO TO 51
      EPEAK=ERROR
      IPEAK=I
      XPEAK=I*2.**(-15)
      EAP=EA
      ETP=ET
      EQP=EQ
51  IF(J.EQ.1)GO TO 20
      GO TO 30
40  WRITE(6,41)IM,IPEAK,XM,XPEAK,EAP,EQP,ETP
41  FORMAT(1X,I6,1X,I6,1X,F12.9,1X,F12.9,1X,F9.6,1X,F9.6,1X,F9.6)
10  CONTINUE
      STOP
      END
      SUBROUTINE QUANT(RA,RAQ)
      REAL*8 RA,RAQ,RS,RR,C
      DIMENSION CO(16)
      RS=RA
      RC=1.
      IF(RS.GE.0)GO TO 60
      RC=-1.
      RS=-RS
60  C=0
      DO 61 N=2,16
      CO(N)=2.**(-(N-1))
      IF(RS.LT.CO(N))GO TO 62
      RS=RS-CO(N)
      C=C+CO(N)
62  RAQ=RC*C
61  CONTINUE
      RETURN
      END

```

ERROR16 FORTRAN

```

      REAL*8 R,RA,RAQ,EA,ET,EQ,ERROR,EPEAK,XPEAK,EAP,ETP,EQP
      REAL*8 DEL,RDEL,X1,X2
      WRITE(6,1)
1  FORMAT(/,1X,'ERROR ANALYSIS FOR APPROXIMATE 1/X',/)
      WRITE(6,2)
2  FORMAT(1X,'DATA TABULATED RE MAX INPUT X - DOUBLE PRECISION',/)
      WRITE(6,3)
3  FORMAT(1X,'EA,EQ,ET=APPROX,QUANT,& TOTAL WORST-CASE ERRORS',/)
      WRITE(6,4)
4  FORMAT(3X,'IMAX',4X,'IPK',6X,'XMAX',9X,'XPK',9X,'EA',8X,'EQ',8X,
1  'ET',/)
      DO 10 L=1,16
      IM=2**(L-1)
      XM=(IM-1.)*2.**(-15)
      EPEAK=-1.
      I1=1
      I2=128
      J=1
      IMAX=2**(L-1)
      IEND=128
      IF(IMAX.LT.IEND)IEND=IMAX
      DO 20 I=1,IEND
      IF(I.GT.2)GO TO 21
      R=1.-(2.**(-15))
      GO TO 22
21  R=1./(I-1)
22  RA=R
      CALL QUANT(RA,RAQ)
      GO TO 50
20  CONTINUE
      IF(I.EQ.IMAX)GO TO 40
15  J=J+1
      I3=2*(I2-I1)+1
      I1=I2+1
      I2=I1+I3
      IF(IMAX.LT.I2)I2=IMAX
      IF(I2.EQ.32640)I2=32768
      DO 30 I=I1,I2
      R=1./(I-1)
      DEL=2.**(J-1)
      IF(I.GE.32513)DEL=256.
      IDEL=DEL
      RDEL=1./DEL
      X1=I/DEL
      IX1=I/IDEL

```

ERROR32 FORTRAN

```

      X2=X1-IX1
      IF(X2.NE.RDEL)GO TO 30
      RA=2./(2.*(I-1)+DEL-1.)
      CALL QUANT(RA,RAQ)
      GO TO 50
30  CONTINUE
      IF(I.EQ.IMAX)GO TO 40
      GO TO 15
50  EA=(R-RA)/R
      ET=(R-RAQ)/R
      EQ=ET-EA
      ERROR=ET
      IF(ET.LT.0)ERROR=-ET
      IF(ERROR.LE.EPEAK)GO TO 51
      EPEAK=ERROR
      IPEAK=I
      XPEAK=I*2.**(-15)
      EAP=EA
      ETP=ET
      EQP=EQ
51  IF(J.EQ.1)GO TO 20
      GO TO 30
40  WRITE(6,41)IM,IPEAK,XM,XPEAK,EAP,EQP,ETP
41  FORMAT(1X,I6,1X,I6,1X,F12.9,1X,F12.9,1X,F9.6,1X,F9.6,1X,F9.6)
10  CONTINUE
      STOP
      END
      SUBROUTINE QUANT(RA,RAQ)
      REAL*8 RA,RAQ,RS,RR,C
      DIMENSION CO(32)
      RS=RA
      RC=1.
      IF(RS.GE.0)GO TO 60
      RC=-1.
      RS=-RS
60  C=0
      DO 61 N=2,32
      CO(N)=2.**(-(N-1))
      IF(RS.LT.CO(N))GO TO 62
      RS=RS-CO(N)
      C=C+CO(N)
62  RAQ=RC*C
61  CONTINUE
      RETURN
      END

```

ERROR32 FORTRAN

APPENDIX B  
HEXADECIMAL DATA FILES FOR PROM A, PROM B,  
AND PMUX PROGRAMMING

PMUX DATA

7777777677777765  
7777765377776537  
7776537777653777  
7653777765377777  
5377777737777777  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000

PROMA DATA

7665555444444443  
3333333333333332  
2222222222222222  
2222222222222221  
1111111111111111  
1111111111111111  
1111111111111111  
1111111111111110  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000  
0000000000000000

PROMB DATA

00804040C0C020202020E0E0E0E01010  
101050505050B0B0B0B0F0F0F0F00808  
08084848484828282828686868689898  
9898D8D8D8D8B8B8B8B8F8F8F8F80404  
04044444444424242424646464641414  
14145454545434343434747474748C8C  
8C8CCCCCCCCACACACACECECECEC9C9C  
9C9CDCDCDCDCBCBCBCBCFCFCFCFC0202  
02024242424222222222626262621212  
12125252525232323232727272720A0A  
0A0A4A4A4A4A2A2A2A2A6A6A6A6A1A1A  
1A1A5A5A5A5A3A3A3A3A7A7A7A7A8686  
8686C6C6C6C6A6A6A6A6E6E6E6E69696  
9696D6D6D6D6B6B6B6B6F6F6F6F68E8E  
8E8ECECECECEAEAEAEAEAEAEAE9E9E  
9E9EDEDEDEDEBEBEBEBEFEFEFEFE0000  
00004040404020202020606060601010  
10105050505030303030707070700808  
08084848484828282828686868681818  
18185858585838383838787878780404  
04044444444424242424646464641414  
14145454545434343434747474740C0C  
0C0C4C4C4C4C2C2C2C2C6C6C6C6C1C1C  
1C1C5C5C5C5C3C3C3C3C7C7C7C7C8282  
8282C2C2C2C2A2A2A2A2E2E2E2E29292  
9292D2D2D2D2B2B2B2B2F2F2F2F28A8A  
8A8ACACACACAAAAAAAEAEAEAE9A9A  
9A9ADADADADABABABABAFAFAF8686  
8686C6C6C6C6A6A6A6A6E6E6E6E69696  
9696D6D6D6D6B6B6B6B6F6F6F6F68E8E  
8E8ECECECECEAEAEAEAEAEAEAE9E9E  
9E9EDEDEDEDEBEBEBEBEFEFEFEFE8181

APPENDIX C  
 FORTRAN PROGRAM FOR CALCULATING  
 SINGLE-PRECISION TABLE ENTRIES

```

INTEGER*2 YINT(2048)
LOGICAL*1 YFROM(32)
EQUIVALENCE (YINT(1),YFROM(1))
REAL*16 Y(2048),CF,C(16),XI,XK,DEL,R
C(1)=16128.
C(2)=-192.
C(3)=-4256.
C(4)=-4240.
C(5)=-3208.
C(6)=-2180.
C(7)=-1410.
C(8)=-897.
C(9)=-114943.
C(10)=-65727.
C(11)=-37023.
C(12)=-20623.
C(13)=-11399.
C(14)=-6275.
C(15)=-3457.
C(16)=-1920.
CF=2.
M=1
DO 20 J=1,16
  IF(J .GE. 9) M=2
  L2=128*J
  L1=L2-127
  L3=8*M-J
  DO 20 I=L1,L2
    XI=I
    DEL=2.**L3
    XK=C(J)+XI*DEL
    IF(I.EQ.897 .OR. I.EQ.898)GO TO 15
    IF(I.GT.897 .AND. I.LE.1024)GO TO 10
    IF(I.GE.1921 .AND. I.LE.2048)GO TO 40
    IF(I.EQ.128 .OR. I.EQ.1152)GO TO 30
    IF(M.EQ.2)CF=-2.
    Y(I)=CF/(2.*XK+DEL-1.)
  
```

TABLE1 FORTRAN

```

      GO TO 20
10  Y(I)=1./XK
15  Y(897)=1.-(2.**(-15))
      Y(898)=Y(897)
      GO TO 20
40  Y(I)=-1./XK
      GO TO 20
30  Y(I)=CF/(2.*XK+255.)
20  CONTINUE
      DO 70 I=1,2048
      WRITE(3,3)Y(I)
3   FORMAT(F36.33)
      N=1
      ICUM=0
      R=Y(I)
      IF(Y(I).GE.0)GO TO 25
      N=-1
      R=-Y(I)
25  R=R-2.**(-15)
      IF(R.LT.0)GO TO 35
      ICUM=ICUM+1
      GO TO 25
35  YINT(I)=N*ICUM
70  WRITE(4,4)YINT(I)
4   FORMAT(I8)
      DO 60 I=1,4065,32
      WRITE(1,2) YFROM(I),YFROM(I+2),YFROM(I+4),YFROM(I+6),
1YFROM(I+8),YFROM(I+10),YFROM(I+12),YFROM(I+14),
1YFROM(I+16),YFROM(I+18),YFROM(I+20),YFROM(I+22),
1YFROM(I+24),YFROM(I+26),YFROM(I+28),YFROM(I+30)
60  WRITE(2,2) YFROM(I+1),YFROM(I+3),YFROM(I+5),YFROM(I+7),
1YFROM(I+9),YFROM(I+11),YFROM(I+13),YFROM(I+15),
1YFROM(I+17),YFROM(I+19),YFROM(I+21),YFROM(I+23),
1YFROM(I+25),YFROM(I+27),YFROM(I+29),YFROM(I+31)
2   FORMAT(32Z2)
      STOP
      END

```

TABLE1 FORTRAN

APPENDIX D  
HEXADECIMAL DATA FILES FOR PROM 3  
AND PROM 4 PROGRAMMING

|                                  |    |
|----------------------------------|----|
| 02010101010101010101010101010101 |    |
| 01010101010101010101010101010101 |    |
| 01010101010101010101010101010101 |    |
| 01010101010101010101010101010101 |    |
| 01010101010101010101010101010101 | 8+ |
| 01010101010101010101010101010101 |    |
| 01010101010101010101010101010101 |    |
| 01010101010101010101010101010101 |    |
| 01010101010101010101010101010101 |    |
| 01010101010101010101010101010101 |    |
| 04040303030303030303030303030303 |    |
| 03030303030303030303030303030303 |    |
| 03030303030303030303030303030202 |    |
| 02020202020202020202020202020202 | 7+ |
| 02020202020202020202020202020202 |    |
| 02020202020202020202020202020202 |    |
| 02020202020202020202020202020202 |    |
| 02020202020202020202020202020202 |    |
| 08080808070707070707070707070707 |    |
| 07070707070706060606060606060606 |    |
| 06060606060606060606060606060605 |    |
| 05050505050505050505050505050505 | 6+ |
| 05050505050505050505050505050505 |    |
| 05040404040404040404040404040404 |    |
| 04040404040404040404040404040404 |    |
| 04040404040404040404040404040404 |    |
| 11101010101010100F0F0F0F0F0F0F0F |    |
| 0F0E0E0E0E0E0E0E0E0E0D0D0D0D0D0D |    |
| 0D0D0D0D0D0D0C0C0C0C0C0C0C0C0C0C |    |
| 0C0C0C0B0B0B0B0B0B0B0B0B0B0B0B0B | 5+ |
| 0B0B0A0A0A0A0A0A0A0A0A0A0A0A0A0A |    |
| 0A0A0A0A0A09090909090909090909   |    |
| 09090909090909090909090909080808 |    |
| 08080808080808080808080808080808 |    |

BYTE3 HEX

|                                  |    |
|----------------------------------|----|
| 24242323232222222221212120202020 |    |
| 1F1F1F1F1E1E1E1E1E1D1D1D1D1C1C1C |    |
| 1C1C1B1B1B1B1B1B1A1A1A1A1A1A1919 |    |
| 19191919181818181818181717171717 | 4+ |
| 17171616161616161616151515151515 |    |
| 15151514141414141414141414131313 |    |
| 13131313131313131212121212121212 |    |
| 12121212111111111111111111111111 |    |
| 555453525150504F4E4D4D4C4B4A4A49 |    |
| 48484747464545444443424241414040 |    |
| 3F3F3E3E3D3D3C3C3C3B3B3A3A393939 |    |
| 38383737373636363535353434333333 | 3+ |
| 33323232313131303030302F2F2F2E2E |    |
| 2E2E2D2D2D2D2C2C2C2C2B2B2B2B2A2A |    |
| 2A2A2A292929292928282828272727   |    |
| 27272626262626262525252525252424 |    |
| FFFBF7F3F0ECE9E5E2DFDCD9D6D4D1CE |    |
| CCC9C7C4C2C0BDBBB9B7B5B3B1AFADAC |    |
| AAABA6A5A3A1A09E9D9B9A9897959493 |    |
| 91908F8E8C8B8A898887868483828180 | 2+ |
| 7F7E7D7C7B7A7A797877767574737372 |    |
| 7170706F6E6D6D6C6B6A6A6968686766 |    |
| 66656464636362616160605F5F5E5E5D |    |
| 5C5C5B5B5A5A595958585757565655   |    |
| FFFF00AA009955490038CCA2AAD82488 |    |
| 00871C8C6618D190551EECB92694421  |    |
| 00E0C3A88E755E48331F0CFAE8D8C8B9 |    |
| AA9C8F82766A5E53493E342B22191008 | 1+ |
| 00FBF0E9E1DAD4CDC7C0BAB4AFA9A49E |    |
| 99948F8A86817D7874706C6864605C58 |    |
| 55514E4A4744413E3B3835322F2C2927 |    |
| 24211F1C1A181513110E0C0A08060402 |    |

BYTE3 HEX



```

DCDCDDDDDDDEDEDEDFDFDFDFE0E0E0E0
E1E1E1E1E2E2E2E2E3E3E3E3E4E4E4
E4E4E5E5E5E5E5E5E6E6E6E6E7E7E7
E7E7E7E7E8E8E8E8E8E8E9E9E9E9E9
E9E9EAEAEAEAEAEAEAEAEAEAEAEAEAE
EBEBEBEBEBEBEBEBEBEBEBEBEBEBEBEB
EBEBEBEBEBEBEBEBEBEBEBEBEBEBEBEB
EDEDDEDDEDDEDDEEEEEEEEEEEEEEEEEEE
EEEEEEEFEFEFEFEFEFEFEFEFEFEFEFEF
ACADADAFAFB0B1B1B2B3B4B4B5B6B6B7
B8B8B9B9BABABBBBBCBDBDBEBEBEBFBFCOC0
C1C1C2C2C3C3C4C4C5C5C5C6C6C7C7C7
C8C8C9C9C9CACACACBCBCCCCCCCCDCDCD
CDCECECECFCFCFD0D0D0D0D1D1D1D2D2
D2D2D3D3D3D3D4D4D4D4D5D5D5D5D6D6
D6D6D6D7D7D7D7D7D8D8D8D8D9D9D9D9
D9D9DADADADADADBDDBDBDBDBDBDCDCDC
03070B0F1216191C1F2225282B2E3033
36383B3D3F414446484A4C4E50525455
57595B5C5E5F6163646667686A6B6D6E
6F70727374757778797A7B7C7D7E7F80
8182838485868788898A8B8C8D8E8F
8F90919292939495959697979899999A
9B9B9C9C9D9E9E9F9FA0A1A1A2A2A3A3
A4A4A5A5A6A6A7A7A8A8A9A9AAAAABAB
0000560067ABB700C8345E5628DC7800
79E4449AE82F70ABE214436E97BCDF00
203D58728BA2B8CDE1F4061828384756
64717E8A96A2ADB7C2CCD5DEE7F0F800
0810171F262C333940464C51575C6267
6C71767A7F83888C9094989CA0A4A8AB
AFB2B6B9BCBFC2C5C8CBCED1D4D7D9DC
DFE1E4E6E8EBEIEFF2F4F6F8FAFCFE00

```

4-

3-

2-

1-

BYTE3 HEX

5-







APPENDIX E  
 FORTRAN PROGRAM FOR CALCULATING  
 EXTENDED-PRECISION TABLE ENTRIES

```

INTEGER*2 YINT(2048)
LOGICAL*1 YFROM(32)
EQUIVALENCE (YINT(1),YFROM(1))
REAL*16 Y(2048),CF,C(16),XI,XK,DEL,R
C(1)=16128.
C(2)=-192.
C(3)=-4256.
C(4)=-4240.
C(5)=-3208.
C(6)=-2180.
C(7)=-1410.
C(8)=-897.
C(9)=-114943.
C(10)=-65727.
C(11)=-37023.
C(12)=-20623.
C(13)=-11399.
C(14)=-6275.
C(15)=-3457.
C(16)=-1920.
CF=2.
M=1
DO 20 J=1,16
  IF(J .GE. 9) M=2
  L2=128*J
  L1=L2-127
  L3=8*M-J
  DO 20 I=L1,L2
    XI=I
    DEL=2.**L3
    XK=C(J)+XI*DEL
    IF(I.EQ.897 .OR. I.EQ.898)GO TO 15
    IF(I.GT.897 .AND. I.LE.1024)GO TO 10
    IF(I.GE.1921 .AND. I.LE.2048)GO TO 40
    IF(I.EQ.128 .OR. I.EQ.1152)GO TO 30
    IF(M.EQ.2)CF=-2.
    Y(I)=CF/(2.*XK+DEL-1.)
    GO TO 20
10 Y(I)=1./XK
15 Y(897)=1.-(2.**(-15))
    Y(898)=Y(897)
    GO TO 20
  
```

TABLE2 FORTRAN

```

40 Y(I)=-1./XK
   GO TO 20
30 Y(I)=CF/(2.*XK+255.)
20 CONTINUE
   DO 70 I=1,2048
     R=Y(I)
     N=1
     IF(R.GE.0)GO TO 45
     N=-1
     R=-Y(I)
45 R=R-2.**(-15)
     IF(R.LT.0)GO TO 50
     GO TO 45
50 Y(I)=R+2.**(-15)
     IF(N.EQ.-1)Y(I)=-Y(I)
     WRITE(3,3)Y(I)
   3 FORMAT(F36.33)
     N=1
     ICUM=0
     R=Y(I)
     IF(Y(I).GE.0)GO TO 25
     N=-1
     R=-Y(I)
25 R=R-2.**(-30)
     IF(R.LT.0)GO TO 35
     ICUM=ICUM+1
     GO TO 25
35 YINT(I)=N*ICUM
70 WRITE(4,4)YINT(I)
   4 FORMAT(I8)
     DO 60 I=1,4065,32
       WRITE(1,2) YFROM(I),YFROM(I+2),YFROM(I+4),YFROM(I+6),
         1YFROM(I+8),YFROM(I+10),YFROM(I+12),YFROM(I+14),
         1YFROM(I+16),YFROM(I+18),YFROM(I+20),YFROM(I+22),
         1YFROM(I+24),YFROM(I+26),YFROM(I+28),YFROM(I+30)
60 WRITE(2,2) YFROM(I+1),YFROM(I+3),YFROM(I+5),YFROM(I+7),
         1YFROM(I+9),YFROM(I+11),YFROM(I+13),YFROM(I+15),
         1YFROM(I+17),YFROM(I+19),YFROM(I+21),YFROM(I+23),
         1YFROM(I+25),YFROM(I+27),YFROM(I+29),YFROM(I+31)
   2 FORMAT(32Z2)
     STOP
   END

```

TABLE2 FORTRAN

APPENDIX F  
HEXADECIMAL DATA FILES FOR PROM 1  
AND PROM 2 PROGRAMMING

|                                   |    |
|-----------------------------------|----|
| 03020A1A315075A2D6105199E73B95F5  |    |
| 5AC637AD29AA30BB4BE07A18BB620EBE  |    |
| 722AE7A76B33FFCFA279533112F6DEC9  |    |
| B7A89C948E8B8B8D939BA6B4C4D6EB03  | 8+ |
| 1D3A58799DC2EA14406F9FD2063C75AF  |    |
| EB2969A8EF347BC40E5BA9F8499CF046  |    |
| 9DF650AC0968C8298CF056BC258EF964  |    |
| D240AF2092057AEF65DD56CF4AC64381  |    |
| 1A0A09193866A3EF48B025A735D1792D  |    |
| EDB88F715E557647A9AC4F73379C81F   |    |
| 7FE758D051D96901A047F4A96426EFBF  |    |
| 9470533B2A1E18181D28394F6A8A9FD9  | 7+ |
| 083D75B3F53C87D72B83E040A50E7BEC  |    |
| 60D955D558DF6AF88A1FB753F29439E2  |    |
| 8D3CEEA25A14D191541AE2AD7B4B1DF3  |    |
| CBA5826142260CF5DFCCBCADA1968E88  |    |
| EB874524234280DD56EDA06E585B79B0  |    |
| 0067E77E2BEFC8B7BBD3004093F972FD  |    |
| 9A4908D9BAABACBDDDD0C4B97F25BD256 |    |
| E8B632EAAE7F5C45393843597AA5DB1B  | 6+ |
| 66BA1981F26DF27F16B55D0DC6885123  |    |
| FCDEC7B8B0B0B7C5DAF71A4475ACEA2E  |    |
| 79C9217EE14AB92EA828AE39C95FFA9B  |    |
| 40EB9A4F08C689511DEEC49E7C5F4631  |    |
| 081E7F291A51CB8987C43FF6E9157A16  |    |
| E8EF29963504022F8910C29FA6D62EAE  |    |
| 5420112760BC3BDB9D7F81A2E240BB54  |    |
| 0ADBC8D0F33086F67F20D9AA9291A6D2  | 5+ |
| 1369D555EA924F1F02F7001A4785D535  |    |
| A72ABC5F12D5A788797886A2CC054B9F  |    |
| 006EEA7207A95711D7A9877065657086  |    |
| A7D2084993E847B0239F25B44CEE994D  |    |

BYTE1 HEX

```

EDD26FBFB066B4A43259166543AC9E14
0C8275E0C217DD11B2BC2E053FDA042A
DCE748000B6816135DF2D2FB6B221D5C
DD9FA1E15F190F3EA749212F73EB9775 4+
85C637D7A6A2CB20A14D23224A9A12B0
755F6FA3FB7614D5B7BBE0258A0EB173
53506BA2F666F2985A362C3C65A80376
02A560321A1A2F5A9BF25DDD721CD9AA
2A6EF7B59589806B3CE65A8C71FB1FD2
0BBDE06A518E16E1E9248C18C3845632
11EFC48C41DC5BB6EAF2C86AD1FRE385
DDE7A00411C112028CAE65AD85E9D74C 3+
46C2BE372B987CD49FDA849A1C05560C
26A27DB84F428F34318329226D08F32B
AFB09AFDA99BD350111459DFA5AAEC6C
281F51BC613D519B1ACFB8D524A6593D
7F421F6A80C8B2B6530F7821A199AA7C
BC1949030306CE20C380247C5A90F358
978B0EFE389C0B66906FE6DD3AE6C9CD
DEE6D08B0428E62EEE199E6E7CBA1A9C 2+
0F8CF94D7B793EBEF0CB4556F418B8CD
5039801E0C45C0776583CC39C56B25ED
C0986F420CC872067FDA12240CC64EA1
BC9A3A97AF7E013619A8E0BE3F6223B0
0000005500CCAA4900C7665D55C42444
00C3E3D733C32E0BAA286297922C2221
001FE14171916B41991861F417C18582
550A1441B1CF4B12494796E511141041
00C00FBDF003A0B43838C80D357B2074 1+
CC870CC830C0FA648B026040420B4191
AA3F05B50AC3A065D8C0E71A25DA0985
24BC2335CBC0F24088AB8A08086E2004

```

BYTE1 HEX

0201FAEAD3B48E612EF3B26B1DC96F0F  
A93DCC56DA59D348B82389EB48A1F545  
91D81C5C97CF0334608AAFD2F00C2439  
4B5A666F747777756F675C4E3E2C16FF  
E5C8A988653F17EDC1936230FCC58D52  
16D8985613CD863DF3A75909B86611BC  
640BB155F89939D87511AC45DD73099D  
30C152E16FFC88129C24AB32B73BBE81  
F60706F7D7A96B20C65FEA67D83C94E0  
20547D9BAEB7B5A892724814D89243EC  
8C23B33AB930A00869C31560A5E31A4A  
7498B6CDDFEAF0F0EADFCFB99E7E582E  
FFCB925412CB8030DC8327C661F88B1B  
A62DB131AE279C0E7CE74EB31471CC24  
78C91863ABF13474B1EB23588ABAE712  
3A6083A4C2DEF81025384957646E767C  
59BCFD1D1CFCBD60E54E9ACBE1DCBE86  
35CD4DB507436879755C2FEE9A33BA2E  
91E221506F7D7C6A4A1ADB8E33C952CE  
3C9DF13873A2C5DCE7E8DCC6A5794302  
B863049C2AAF2A9C0666BE0D5492C8F7  
1D3B5261686861523D20FDD3A26A2CE8  
9D4CF59734CB5CE76CEC66DB4AB41978  
D32878C40A4C89C1F4234E7495B2CBE0  
19FE99EAF5BA3B797836B7FC06D66ECE  
F9EFB141A0CECC9D40B6012218E68B09  
61929F874BED6CCA06221EFBB959DC41  
8AB7C8BE9A5C03920865AAD8EFEEED8AB  
6811A4228CE2255470796F5426E79634  
C13EAA06528EBRD9E8E8D9BC915710BC  
5AEA6EE54FADFE437CA9CBE1ECEBDFC9  
A77B4403B862029926A92393FA57ACFB

8-

7-

6-

5-

BYTE1 HEX

4044912B1759F7F3521848E7F77D7BF4  
EC6664EAF97C380D2BB3D59136C6705  
4832C503EE86CFC875D6EDBB42827F38  
AEE4DB930D4B4F18A7FF200BC14290AB  
954ED8325D5B2DD24B9ABFBB8E39BD19  
50614D14B83895D1EAE2B970087FD813  
2F2E10D57D0A7BD10C2D3320F4AE50DA  
4BA5E71227250DDF9B42D451BA0E4E7B  
FD260DC65FE9740EC5A8C323D4E25944  
AC9C1F3DFF6F957A259EEC1827210CEE  
CFB3A09DAED9239128ECE20F761D0636  
B17B9708D3FA806ABA72972A2FA89802  
E74B309885FAF9849D46804FB4B04676  
44B0BD6BBD835095831C6053F34447FB  
648154DE211DD44674610B75A08B39AA  
E0DA99206D83620A7DBBC59C40B2F202  
8E27FFB8F94E47672C0B75D48CFC806D  
14C4C662D86B57D51F68E4C2336179A2  
03C305EC992CC37D75C78DE0D88E188B  
FC802B0F3ECBC64049EF43522AD96DF0  
71FB99573F5CB95E57AB658C2A47EB1E  
E84F5C1480A68C38B1FD22240BDA984A  
F59C46F8B480615A6FA500833110226A  
EDAEAFF480577AEED5D1461643D1C218  
0000AB003456B700399AA3AB3CDCBC00  
3D1D29CD3DD2F556D89E696ED4DEDFO0  
E11FRF8F6F95BF67E89FOCE93F7B7EAB  
F6ECBF4F31B5EEB7B96A1BEFECFOBF00  
40F17310FD604CC8C838F3CB85E08C34  
79F438D040069C75FEA0C0BEF5BF6F56  
C1FB4BF63D609B284019E6DB26F77BDC  
44DDCB35400EC0785576F8F892E0FC00

4-

5-

2-

1-

BYTE1 HEX

```

017F7D7B7977757371706E6C6A696765
6462615F5E5C5B59585655545251504E
4D4C4A4948474544434241403F3D3C3B
3A393837363534333231302F2E2D2C2C
2B2A29282726252423222121201F1E      8+
1D1D1C1B1A1A19181817161515141313
12111110100F0E0E0D0C0C0B0B0A0909
08080707060605040403030202010100
06027E7A76726E6A6763605C5955524F
4B4B45423F3C393633302D2A28252220
1D1A181513100E0C09070402007E7B79
777573716F6D6B69676563615F5D5B59      7+
58565452504F4D4B4A4846454342403E
3D3B3A383735343231302E2D2B2A2927
2625232221201E1D1C1B191817161513
1211100F0E0D0C0A0908070605040302
1C140C047C746C645D554E474039322B
251E17110B047E78726C67615B55504A
45403B35302B26211C18130E0905007C
77736F6A66625E5A56524E4A46423E3B      6+
3733302C2825211E1B1714110D0A0704
007D7A7774716E6B686563605D5A5755
524F4D4A474542403D3B383633312E2C
2A272523211E1C1A181513110F0D0B09
006E5C4B3A291808786859493A2C1D0F
007265574A3D3023160A7D7165594E42
372C21160B00766B61574D433930261D
140A01786F675E554D453C342C241C14      5+
0D057D766E676059524A443D362F2822
1B150E08027B756F69635D57514C4640
3B352F2A251F1A150F0A05007B76716C
67625E59544F4B46423D3934302B2723

```

BYTE2 HEX

|                                  |    |
|----------------------------------|----|
| 360D653D16704A25015D3A1775533212 |    |
| 7252331476593B1F02664B30157A6047 |    |
| 2D147C644C341D066F58422C17026D58 |    |
| 432F1B0774614E3B28160472604E3D2C | 4+ |
| 1B0A7A695949392A1A0B7C6D5E4F4132 |    |
| 2416087A6C5F5244372A1D1104786B5F |    |
| 53473B2F23180C01766B60554A3F352A |    |
| 20150B01776D63594F453C322920160D |    |
| 001022374E67021F3E5E01254B721C46 |    |
| 73204F0032651A4F063F78336E2B6928 |    |
| 68286A2D71357B41085019632D784411 |    |
| 5E2C7B4B1B6B3D0F6134085C31065C33 | 3+ |
| 0A6139126B441E78532E0A6643207D5B |    |
| 391776553515755637187A5C3E210367 |    |
| 4A2E12765B40250B71573D230A715840 |    |
| 2B10786049321B046E57412B16006B56 |    |
| 000C2750074B1C796256545D6F0B305E |    |
| 145319673C187A64534945464D596A01 |    |
| 1C3C610937681E57145519612D7B4D22 |    |
| 7A553314785E46321F1002776E676360 | 2+ |
| 6061646A717A05111F2F4154687F162F |    |
| 4A660322426305294E741B446D184470 |    |
| 1E4D7D2E6012467B30661E560F48033E |    |
| 7A3775337232733476387B3F04490F55 |    |
| 00000055004C2A1200716674554E4944 |    |
| 0043385033303A592A5C275024772204 |    |
| 007C611D1C4F281A191C18055D162C18 |    |
| 555E2E411321686412707B3111174210 | 1+ |
| 000F3E0970730E420E70677414470D64 |    |
| 4C454E650C4002522E170B0B162C4C76 |    |
| 2A682F7E573720110909101F344F721A |    |
| 497D38783D08582E08674B3421120802 |    |

BYTE2 HEX

FF81828486888A8C8E8F91939596989A  
9B9D9EA0A1A3A4A6A7A9AAABADAEAFB1  
B2B3B5B6B7B8BABBBBCBDBEBFC0C2C3C4  
C5C6C7C8C9CACBCCCDCECFD0D1D2D3D3  
D4D5D6D7D8D9DADADBDCDDDEDEDFE0E1  
E2E2E3E4E5E5E6E7E7E8E9EAEAEBECEC  
EDEEEEEFEFF0F1F1F2F3F3F4F4F5F6F6  
F7F7F8F8F9F9FAFBFBFCFCFDFDFEFFFF  
F9FE8285898D9195989C9FA3A6AAAD80  
B4B7BABC0C3C6C9CCCFD2D5D7DADDDF  
E2E5E7EAECEFF1F4F6F8FBFDF818486  
888A8C8E90929496989A9C9EA0A2A4A6  
A7A9ABADAFB0B2B4B5B7B9BABCBD8FC1  
C2C4C5C7C8C9CBCECFD1D2D4D5D6D8  
D9DADCDDEDFE1E2E3E4E6E7E8E9EAECE  
EDEEEFF0F1F2F3F5F6F7F8F9FAFBFCFD  
E3EBF3FC848B939BA2AAB1B8BFC6CDD4  
DBE1E8EEF5FB81878D93999EA4AAAFB5  
BABFC5CACFD4D9DEE3E8ECF1F6FAFF83  
888C9095999DA1A5A9ADB1B5B9BDC1C5  
C8CCD0D3D7DADEE1E5E8EBEFF2F5F8FB  
FF8285888B8E9194979A9C9FA2A5A8AA  
ADB0B2B5B8BABDBFC2C4C7C9CCCE1D3  
D5D8DADCDFE1E3E5E7EAECEEF0F2F4F6  
8192A4B5C6D7E8FB8898A7B6C6D4E3F1  
FF8D9BA9B6C3D0DDEAF6838F9BA6B2BE  
C9D4DFEAF5FF8A949FA9B3BCC6D0D9E3  
ECF5FE879099A2AAB3B8C3CBD3DBE3EB  
F3FB828A9198A0A7AEB5BCC3CAD0D7DE  
E4EBF1F8FE848A90969CA2A8AEB4BABF  
C5CAD0D5DBE0E5EBF0F5FAFF84898E93  
989DA2A7ABB0B5B9BEC2C7CBCFD4D8DC

8-

7-

6-

5-

BYTE2 HEX

CEF79FC7EE94B9DE83A7CAEC8EB0D1F1  
91B1D0EE8CAAC7E4809CB8D3EE88A2BC  
D5EE869FB6CEE5FC93A9BFD5EB8095AA  
BED2E6FA8EA1B4C7D9ERFE90A1B3C4D5  
E6F78798A8B8C8D7E7F68594A3B2C0CF  
DDEBF98794A2AFBCC9D6E3F0FD8995A2  
AERAC6D1DDE9F4FF8B96A1ACB6C1CCD6  
E1EBF5808A949EA7B1B8C4CED7E1EAF3  
9B8BF8E2CBB196F9D9B895F1CAA2F9CE  
A1F3C493E0ADF8C28BD298DEA2E5A7E7  
A7E6A4E19DD893CC85BCF3AADF94C8FB  
ADD9F0C1F09FCEFC9D682AED983ADD7  
FFA8D0F79EC4EA90B5DAFEA2C5E88BAD  
CFF091B2D2F292B1D0EF8DABC8E6839F  
BCD8F48FABC6E0FB95AFC9E2FB94ADC5  
DDF58DA5BCD3EA8197ADC3D9EF8499AF  
FBE8C595D78DB6D3E5ECE8DAC3A2F9C7  
8DCAB0AFD6F791A4B2B9BAB6AD9E8AF1  
D4B18BDFB0FDC58ACB88C2F8ABDB88B1  
D7FB9CBAD5ED8397A8B6C3CDD5DAEDF  
DFDCD8D2CAC0B4A79887F5E1CCB59C83  
E7CBAD8EEDCBA884DEB790E7BD91E5B8  
89DAAAF8C693DFAAF4BD86CD94DA9FE3  
A6E9ABECADEDACEAA8E5A2DE99D38DC7  
0000AA00B3D5ED008E998BAAB1B6BB00  
BCC7AFCCCFCSA6D5A3D8AFDB88DDFB00  
839EE2E3B0D7E5E6E3E7FAA2E9D3E7AA  
A1D1BEECDE979BED8F84CEEEE8BDEF00  
F0C1F68F8CF1BDF18F988BEBB8F29BB3  
BAB19AF3BFFDADD1E8F4F4E9D3B389D5  
97D0B1A8C8DFEEF6F6EFE0CB808DE5B6  
82C787C2F7A7D1F798B4CBDEEDF7FD00

4-

3-

2-

1-

BYTE2 HEX

APPENDIX G  
SAMPLE CALCULATIONS FOR TWO PSEUDODIVIDE TEST CASES

Test Case 1

$$X = 3321 \text{ (integer)} = \text{CF9} = 0.101\ 348\ 876\ 953\ 125$$

$$Y = 1672 \text{ (integer)} = 688 = 0.051\ 025\ 390\ 625$$

$$\text{The exact decimal value for } 1672/3321 = Q = 0.503\ 462\ 812\ 405\ 901\ \dots$$

$$R = 2^{-15}/X = 0.000\ 301\ 108\ \dots$$

$$\hat{Q} = \hat{R} \cdot Y \text{ is to be calculated in two parts, with}$$

$$\hat{R} = \hat{R}_m + \hat{R}_1, \text{ and } \hat{Q} = \hat{Q}_m + \hat{Q}_1 = \hat{R}_m \cdot Y + \hat{R}_1 \cdot Y.$$

Some scaling maneuvers are needed. If we interpret the  $\hat{R}_m$  table as being scaled by  $2^{-15}$ , but the  $\hat{R}_1$  table as not scaled, then  $\hat{Q}_m$  can be retrieved from the multiplier LSP with assigned weights of  $2^{-1}$  to  $2^{-15}$ , and  $\hat{Q}_1$  may be found subsequently at the multiplier MSP with weights interpreted as  $2^{-16}$  to  $2^{-30}$ .

Integer  $X = 3321$  is the 88th location in reciprocal zone  $5^+$ , where quantization factor ( $\Delta$ ) is 16 and integer  $X$  ranges from 1920 to 3967. From the hex tables,  $\hat{R}_m = 0009$  and  $\hat{R}_1 = 6F88$ .

Accordingly,

$$\hat{Q}_m = (0009)(688) = 3AC8 = 0.459\ 228\ 515\ 625$$

$$\hat{Q}_1 = (6F88)(688) = 2D87040 = 05B0 = 0.044\ 433\ 593\ 750$$

and

$$\hat{Q} = \hat{Q}_m + \hat{Q}_1 = 0.503\ 622\ 109\ 375$$

$$\begin{aligned}\text{Error } E_T &= (\hat{Q} - Q)/Q \\ &= (0.503\ 622\ 109\ 375 - 0.503\ 462\ 812\ 405)/ \\ &\quad 0.503\ 462\ 812\ 405 \\ &= 0.00039\end{aligned}$$

Test Case 2

$$X = -3321 \text{ (integer)} = F307 = -0.101\ 348\ 876\ 953\ 125$$

$$Y = 1672 \text{ (integer)} = 688 = 0.051\ 025\ 390\ 625$$

$$\begin{aligned}\text{The exact decimal value for } 1672/(-3321) &= Q = \\ &-0.503\ 462\ 812\ 405\ 901\ \dots\end{aligned}$$

$$R = 2^{-15}/X = -0.000\ 301\ 108\ \dots$$

Integer  $X = -3321$  points to reciprocal zone 5<sup>-</sup>, 88th location.

From the hex tables,  $\hat{R}_m = FFF7$  and  $\hat{R}_1 = 90D9$ .

Thus

$$\begin{aligned}\hat{Q}_m &= (FFF7)(688) = 687C538 \Rightarrow C538 \text{ (selecting LSP)} \\ &= -0.459\ 228\ 515\ 625\end{aligned}$$

$$\begin{aligned}\hat{Q}_1 &= (90D9)(688) = D2A0948 \Rightarrow FA54 \text{ (selecting MSP)} \\ &= -0.044\ 311\ 523\ 437\ 500\end{aligned}$$

and

$$\hat{Q} = \hat{Q}_m + \hat{Q}_1 = -0.503\ 540\ 039\ 062\ 5$$

$$\begin{aligned}\text{Error } E_T &= (\hat{Q} - Q)/Q \\ &= (-0.5035400390625 + 0.5034628124059)/ \\ &\quad (-0.5034628124059) \\ &= 0.00015\end{aligned}$$

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| 12 REPORT DOCUMENTATION PAGE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                 | READ INSTRUCTIONS<br>BEFORE COMPLETING FORM                |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|------------------------------------------------------------|
| 1. REPORT NUMBER<br>(18) ESD-TR-79-263                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 2. GOVT ACCESSION NO.                                           | 3. RECIPIENT'S CATALOG NUMBER                              |
| 4. TITLE (and Subtitle)<br>(6) A Compact Hardware Realization for Approximate Division                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 5. TYPE OF REPORT & PERIOD COVERED<br>(9) Technical Note<br>TN- | 6. PERFORMING ORG. REPORT NUMBER<br>Technical Note 1979-57 |
| 7. AUTHOR(s)<br>(10) Albert H. Huntoon                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 8. CONTRACT OR GRANT NUMBER(s)<br>(15) F19628-80-C-0002         |                                                            |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Lincoln Laboratory, M.I.T.<br>P.O. Box 73<br>Lexington, MA 02173                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 10. PROGRAM ELEMENT, PROJECT, TASK<br>AREA & WORK UNIT NUMBERS  |                                                            |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Naval Electronic Systems Command<br>Department of the Navy<br>Washington, DC                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 12. REPORT DATE<br>(11) 9 October 1979                          | 13. NUMBER OF PAGES<br>70                                  |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)<br>Electronic Systems Division<br>Hanscom AFB<br>Bedford, MA 01731<br>(12) 61                                                                                                                                                                                                                                                                                                                                                                                                                                            | 15. SECURITY CLASS. (of this report)<br>Unclassified            | 15a. DECLASSIFICATION DOWNGRADING<br>SCHEDULE              |
| 16. DISTRIBUTION STATEMENT (of this Report)<br><br>Approved for public release; distribution unlimited.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                                 |                                                            |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                                                 |                                                            |
| 18. SUPPLEMENTARY NOTES<br><br>None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                 |                                                            |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number)<br><br><div style="display: flex; justify-content: space-between;"> <div>digital signal processing<br/>algorithms<br/>reciprocal estimator</div> <div>real time division<br/>array multiplier<br/>microcomputer</div> <div>divider<br/>digital divider<br/>division</div> </div>                                                                                                                                                                                                                                  |                                                                 |                                                            |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number)<br><br><p>The design and implementation of hardware for approximate digital division is described. Using standard, commercially-available T2L components, a compact divider has been developed which is well-suited for use within 5 MHz systems having three-state busses. After forming double-length denominator reciprocals by table look-up to high accuracies, single and extended-precision quotients are made available by consecutive numerator multiplications using a single-chip array multiplier.</p> |                                                                 |                                                            |

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE  
1 JAN 73

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20710-11

1/12