AN ABSTRACT OF THE THESIS OF

Thomas Robert Routhieaux (Name) for the M.S. (Degree) in Electrical and Electronic Engineering (Major)

Date thesis is presented August 6, 1965

Title Logic Circuit Design of a 41-Bit Residue Arithmetic Unit

Abstract approved [redacted]

The application of residue notation is another approach for solving the problem of carry propagation in arithmetic units. Most other methods for solving this problem are based on changes in adder design and the use of improved components.

Residue arithmetic has the inherent property of requiring no carries between moduli. This results in residue arithmetic requiring only a fraction of the carries necessary with conventional weighted arithmetic.

The basic operations of addition, subtraction, multiplication, and division are incorporated in the design of a 41-bit residue arithmetic unit. Using only four moduli, addition and subtraction can be accomplished four times faster than conventional addition and subtraction using synchronous accumulators without carry-look-ahead techniques. Multiplication can be accomplished approximately sixteen times faster. In some

cases, division is accomplished faster than conventional non-restoring division. On the average, however, residue division takes longer to accomplish than non-restoring conventional division.

Detection of overflow in addition, subtraction, and multiplication is based on a slightly different approach for representing signed numbers. Monitoring multiplicative overflow after every multiplication reduces the speed of residue multiplication from sixteen to approximately twelve times faster than conventional multiplication.

LOGIC CIRCUIT DESIGN OF A 41-BIT RESIDUE ARITHMETIC UNIT

by

THOMAS ROBERT ROUTHIEAUX

A THESIS

submitted to

OREGON STATE UNIVERSITY

in partial fulfillment of
the requirements for the
degree of

MASTER OF SCIENCE

June 1966

APPROVED: [redacted]

Professor of Electrical Engineering

Head of Department of Electrical and Electronic Engineering

In Charge of Major

[redacted]

Dean of Graduate School

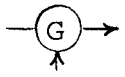
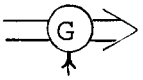
Date thesis is presented August 6, 1965

Typed by Erma McClanathan

TABLE OF CONTENTS

LIST OF FIGURES

SYMBOLS

:=	Is represented by.
≡	Congruent.
|x|	Residue of x or the residue of x with respect to a modulus when the particular modulus concerned is obvious.
	AND gate.
	OR gate.
	Delay line or delay unit.
	Parallel transfer of bits of a register, counter, or accumulator.
	Inhibit input to an AND gate.
	R-S flip-flop.
	Two-input AND gate.
	Two-input AND gates required for parallel transfers.
(-)	An output from a comparator indicating overflow when computing with positive numbers.
"1"	The "one" side of a flip-flop.
"0"	The "zero" side of a flip-flop.

LOGIC CIRCUIT DESIGN OF A 41-BIT RESIDUE ARITHMETIC UNIT

I. INTRODUCTION

Single-Base Notation

Using single-base notation, a number $N := d_n d_{n-1} \cdots d_1 d_o$ or more specifically, $N := \sum_{o}^{n} d_i b^i$ where b is an integer which is the base and $0 \leq d < b$. An example is the decimal number 1,069. $1{,}069 := 1 \cdot 10^3 + 0 \cdot 10^2 + 6 \cdot 10^1 + 9 \cdot 10^o$.

Mixed-Base Notation

In the mixed-base notation a number $N := d_n b_{n-1} b_{n-2} \cdots b_1 + d_{n-1} b_{n-2} \cdots b_1 + d_2 b_1 + d_1$ where each digit refers to the product of the succeeding bases.

Mixed-base notation will be utilized later in the discussion of magnitude determination of residue numbers (1,2,4).

Congruences

The concept of congruences depends on a reference number called the modulus which has a role similar to the base in the single and mixed-base systems. Two numbers are congruent if division by the modulus for each number results in the same remainder. Thus, two numbers, X and X^1 are mathematically shown to be congruent as

$X \equiv X^1 \equiv x \pmod{m}$ where x is the remainder when X or X^1 are divided by the modulus m. The remainder of usual interest is the remainder which is less than m but greater than 0. This remainder is called the least positive residue or simply, the residue of X or X^1 with respect to the modulus m. Another way of representing the residue of X with respect to m is $|X|_m$.

The following rules (1) can be proven rigorously but will only be given here for use in later manipulations.

Let: $|A|_m = a$; $|B|_m = b$

(1) The residue of the sum of two numbers is the residue of the sum of their residues.
Symbolically: $|A + B|_m = |a + b|_m$.
Example: $|20 + 8|_7 = |6 + 1|_7 = 0 = |28|_7$.

(2) The residue of the difference of two numbers is the residue of the difference of their residues.
Symbolically: $|A - B|_m = |a - b|_m$.
Example: $|20 - 8|_7 = |6 - 1|_7 = 5 = |12|_7$.

(3) The residue of the product of two numbers is the residue of the product of the residues.
Symbolically: $|A \cdot B|_m = |a \cdot b|_m$.
Example: $|20 \cdot 8|_7 = |6 \cdot 1|_7 = 6 = |160|_7$.

(4) A unique solution for division is possible only when the divisor and modulus are relatively

prime (common factor is one). This unique solution will not always be correct. If the divisor and modulus have a common factor = f, then f solutions exist provided that the dividend is evenly divisible by f. If the dividend is not evenly divisible by f, then an exact solution is not possible (with present knowledge). Approximations can be made which will be discussed later.

Note that $|m|_m = 0$; $|mX|_m = 0$; and $|X \pm km|_m = |X|_m$. $\left| |X|m \right|_m = |X|_m$ holds true when considering the least positive residue.

The residue of a negative number can be represented as $|-x|_m = |m-x|_m$.

Example: $|-20|_7 = |7-20|_7 = |7-13|_7 = |7-6|_7 = |1|_7 = 1$.

At this point note that $|20|_7 = |13|_7 = |6|_7 = 1$ as expected from a knowledge of congruences. Thus, there isn't any way to determine if the residue of one with respect to modulus seven represents the number 20, 13, six, one, etc.

Multiple Radix Notation

The multiple radix notation consists of a set of radices most commonly labeled as $m_1, m_2, m_3, \cdots, m_n$.

A necessary requirement for the radices in a non-redundant multiple radix system is that no two moduli have a common factor between them except one (relatively prime).

When a number $N := x_n x_{n-1} \cdots x_2 x_1$ each digit represents the least positive residue of N with respect to its corresponding radix. Thus, $X_1 = |N|_{m_1}$.

The compound radix M is the product of the radices. Thus, $M = m_n \cdot M_{n-1} \cdots M_2 \cdot m_1 = \prod_{i=1}^{n} m_i$.

An important characteristic of the multiple radix notation is as follows: A number N can be only uniquely represented if $0 \leq N < M$ when all the radices are relatively prime.

Multiple radix notation will be used in the remaining sections and will be hereafter called the residue notation.

Arithmetic Properties

The previous rules given in the section on congruences concerning the arithmetic properties of least positive residues applies to each digit and corresponding modulus in the residue notation.

Let two numbers A and B be represented by their residues with respect to their radices m_n, m_{n-1}, $\cdots m_2$, m_1. $A := a_n, a_{n-1}, \cdots, a_1$, and $B := b_n, b_{n-1}, \cdots, b_1$. The sum or difference, $S = A \pm B$, and product, $P = A \cdot B$, are also represented by their residues with respect to

the radices. $S := S_n, S_{n-1}, \cdots S_1$, and $P := P_n, P_{n-1}, \cdots, P_1$. To obtain each digit in the sum or product, the arithmetic properties are applied to each set of digits in the operands.

$$S_i = \left| |A|_{m_i} \pm |B|_{m_i} \right|_{m_i} = |a_i \pm b_i|_{m_i}$$

$$P = \left| |A|_{m_i} - |B|_{m_i} \right|_{m_i} = |a_i \cdot b_i|_{m_i}$$

Again, the numbers A, B, S, or P must be less than M for these numbers to be uniquely determined. Also, note that any carriers with respect to any one of the moduli can be ignored.

Representation in Machine Language

Each digit of N where $N := d_n d_{n-1} \cdots d_1$ can be encoded in a weighted binary code such as the 2^o, 2^1, 2^2, 2^3, $\cdots 2^k$ binary code (hereafter called the binary code). Thus, the residue number five := 101 in binary code.

In the 7 5 3 2 residue system the decimal number 16 := 2/ 1/ 1/0 or 010/ 001/ 01/0. By examining the binary code of seven with respect to modulus seven, 111 is not needed since $|7|_7 = 0$. Therefore, six is the largest number to be encoded. Except for moduli of the form 2^k, k = 1, 2, $\cdots$, some inherent inefficiency is noted for encoding residue numbers into the binary code.

Signed Numbers

Sign and Magnitude Notation. An extra bit is used to indicate the sign of the encoded number. A zero and a one are used to represent a positive and negative number respectively. The number N is uniquely represented if $0 \leq N < M$. As an example, in the 7 5 3 2 system + 17 := 0 / 3 / 2 / 2 / 1 or 0 / 011/ 010/ 10/ 1 and -17 := 1 /3 / 2 /2 / 1 or 1 / 011 / 010/ 10 / 1.

Complement Residue Notation. In this method each residue digit is subtracted from its corresponding radix.

Symbolically: $-A := m_n - a_n, m_{n-1} - a_{n-1}, \cdots, m_1 - a_1$.

Thus, $A + \bar{A} = 0$ where $\bar{A}$ is the complement of A. Terms such as $m_n - a_n = \left| m_n - a_n \right|_{m_n}$ are commonly called the additive inverse of a mod m_n.

Using the complement representation the range of magnitude is reduced to $\frac{M}{2}$ since by convention (1,2,6) all positive numbers are represented by numbers 0 - $(\frac{M}{2} - 1)$, inclusive. Negative numbers are represented by numbers $\frac{M}{2} - (M-1)$, inclusive.

Sign and Complement Notation. The rules for addition and subtraction using the complement representation are simpler than the rules for sign and magnitude notation. With this in mind, the complement notation is used as a basis for the sign and complement notation. This notation is identical to the complement notation except

an extra sign bit is used. This redundancy of sign information makes overflow determination much easier.

The rules for addition and subtraction in sign and complement notation are as follows:

(1) a. Subtract - after complementing the subtrahend and its sign bit, perform addition on the non-sign binary bit positions.

b. Add - addition performed directly on the non-sign binary bit positions.

(2) a. If the operands now added are of the same sign, the result is correct if it has the sign of the operands.

b. An incorrect result occurs when the sign of the result is different from the sign of the operands.

(3) If the operands added are oppositely signed, the sum is always correct and properly signed.

Modified Sign and Complement Notation. As an aid in detecting multiplicative overflow, none of the previous signed number notations will be utilized. Instead, a modified sign and complement notation will be used.

With this notation positive numbers are represented by numbers 0 - $(\frac{M}{4} - 1)$, inclusive. Negative numbers are represented by numbers $(M - (\frac{M}{4} - 1)) - (M-1)$, inclusive. Complementation is performed in the same manner as in the sign and complement notation. However, now there exists a region $\frac{M}{4} - (M-(\frac{M}{4} - 1) - 1)$, inclusive, which will be used in the detection of multiplicative overflow. The rules given for the sign and magnitude notation still

hold true.

With these simple rules in mind, it seems apparent that sign determination with residue numbers presents no more difficulties than conventional representations. However, to determine the sign of the result requires magnitude determination of the result which is not easily obtained.

Magnitude Determination

As discussed by G. Lindamood and G. Shapiro (4), the digits in the residue notation give no direct indication of the magnitude of the number which it represents. As an example, in the residue notation system where $m_1 = 13$, $m_2 = 11$, and $m_3 = 7$ the representations for the decimal numbers 65, 72, and 21 are shown.

65 := 0/10/2.
72 := 7/6/2.
21 := 8/10/0.

Magnitude determination appears to be most readily accomplished by converting the multiple radix notation to mixed radix notation. Mixed radix notation is similar to the mixed-base notation except the bases are replaced by the moduli. In general form the number $X := d_n \prod_{i=1}^{n-1} m_i + \cdots + d_2 m_1 + d_1$ in the mixed radix notation where $0 \leq d_n < m_n$. The magnitude range, M, is the same as the magnitude range in residue notation provided the moduli used in the two number notations are the same. Upon

obtaining the mixed radix notation of two numbers the most significant digits can be compared to determine which number is greater. This comparison is analogous to the relative magnitude determination in the decimal number system.

Magnitude determination is necessary no matter which form of residue number notation is used. The modified sign and complement notation appears to be much simpler to mechanize in the determination of the sign of the result and overflow in addition and subtraction and multiplication. Since the signs of the operands are easily obtained by inspection of the sign bits, a conversion of the result to mixed radix notation is all that is necessary. A comparison of the result with $\frac{M}{4}$ and $\frac{3}{4}$ M in mixed radix notation is then made to determine the sign of the result and whether overflow has occurred.

Conversion to Mixed Radix Notation

Having discussed the necessity of magnitude determination, the method for accomplishing this must be defined. Conversion to mixed radix notation appears to be the best method existing.

By examining X where $X := d_n \prod_{i=1}^{n-1} m_i + \cdots + d_3 m_1 m_2 + d_2 m_1 + d_1$ in mixed radix notation, the least positive residue with respect to m_1 is d_1. Thus, the first mixed radix digit, d_1, is found by performing the $|X|_{m_1}$

operation. The second digit, d_2, is found by subtracting the value of d_1 from X, dividing this difference by m_1, and taking the residue with respect to m_2, $\left|\frac{X-d_1}{m_1}\right|_{m_2}$.

This process is repeated until all the mixed radix digits, d_1 through d_n, are found. Knowing these digits permits relative magnitude determination with another mixed radix number.

As an example, a specific number represented in residue notation will be converted to the mixed radix notation. Let X = 108 which := 4/9/3 where $m_1 = 7$, $m_2 = 11$, and $m_3 = 13$. If $|X|_{m_1}$ is performed where X := $d_3m_2m_1 + d_2m_1 + d_1$ in the mixed radix notation, it is easily seen that d_1 is found. However, d_1 is already known since $|X|_{m_1} = 3$ in residue notation where X := 4/9/3 = $|X|_{m_3} / |X|_{m_2} / |X|_{m_1} = x_3 / x_2 / x_1$. Thus, the residue notation digit $x_1 = d_1$. Subtracting $d_1 = 3$ from X, remembering that X is in residue form, gives the residue number 1/6/0.

At this point in the conversion a division by m_1 is required. In residue notation, division is not always possible (1,2). However, by forming the multiplicative inverses $\left|\frac{1}{m_1}\right|_{11}$ and $\left|\frac{1}{m_1}\right|_{13}$ where $\left|m_1 \cdot \left|\frac{1}{m_1}\right|_{11}\right|_{11} = 1$ and $\left|m_1 \cdot \left|\frac{1}{m_1}\right|_{13}\right|_{13} = 1$, the division by m_1 can now be accomplished by multiplying by the appropriate

multiplicative inverses.

From the preceding definition of multiplicative inverses $\left|\frac{1}{m_1}\right|_{11} = \left|\frac{1}{7}\right|_{11} = 8$ and $\left|\frac{1}{m_1}\right|_{13} = \left|\frac{1}{7}\right|_{13} = 2$. Multiplying the appropriate digits of the residue number 1/6/0 (obtained by subtracting $d_1 = 3$ from each residue digit of the original residue number 4/9/3) yields $|1 \cdot 2|_{13} / |6 \cdot 8|_{11} / 0 = 2/4/0$. Thus, $d_2 = 4$.

Subtracting $d_2 = 4$ from the non-zero digits of the present form of the residue number yields $|2-4|_{13} / |4-4|_{11} / 0 = 11/0/0$. Using $\left|\frac{1}{m_2}\right|_{13} = \left|\frac{1}{11}\right|_{13} = 6$ results in $11 \cdot 6/0/0 = 1/0/0$. Thus, $d_3 = 1$. The residue number 4/9/3 has now been converted to the mixed radix number 1/4/3. Converting 1/4/3 to decimal notation gives $1\ (7 \cdot 11) + 4\ (7) + 3 = 108$. This specific conversion example is shown in Figure 1 emphasizing the sequential nature of a residue to mixed radix conversion. Note that two of the subtractions are accomplished in parallel followed by two multiplications in parallel.

Modulus (mod) adders shown in Figure 1 are essentially conventional adders with some modifications. For the purpose of illustration the mod 11 adder is discussed.

Since 10 is the maximum number to be held in a register associated with a mod 11 adder, four flip-flops are required. When two numbers such as three and six

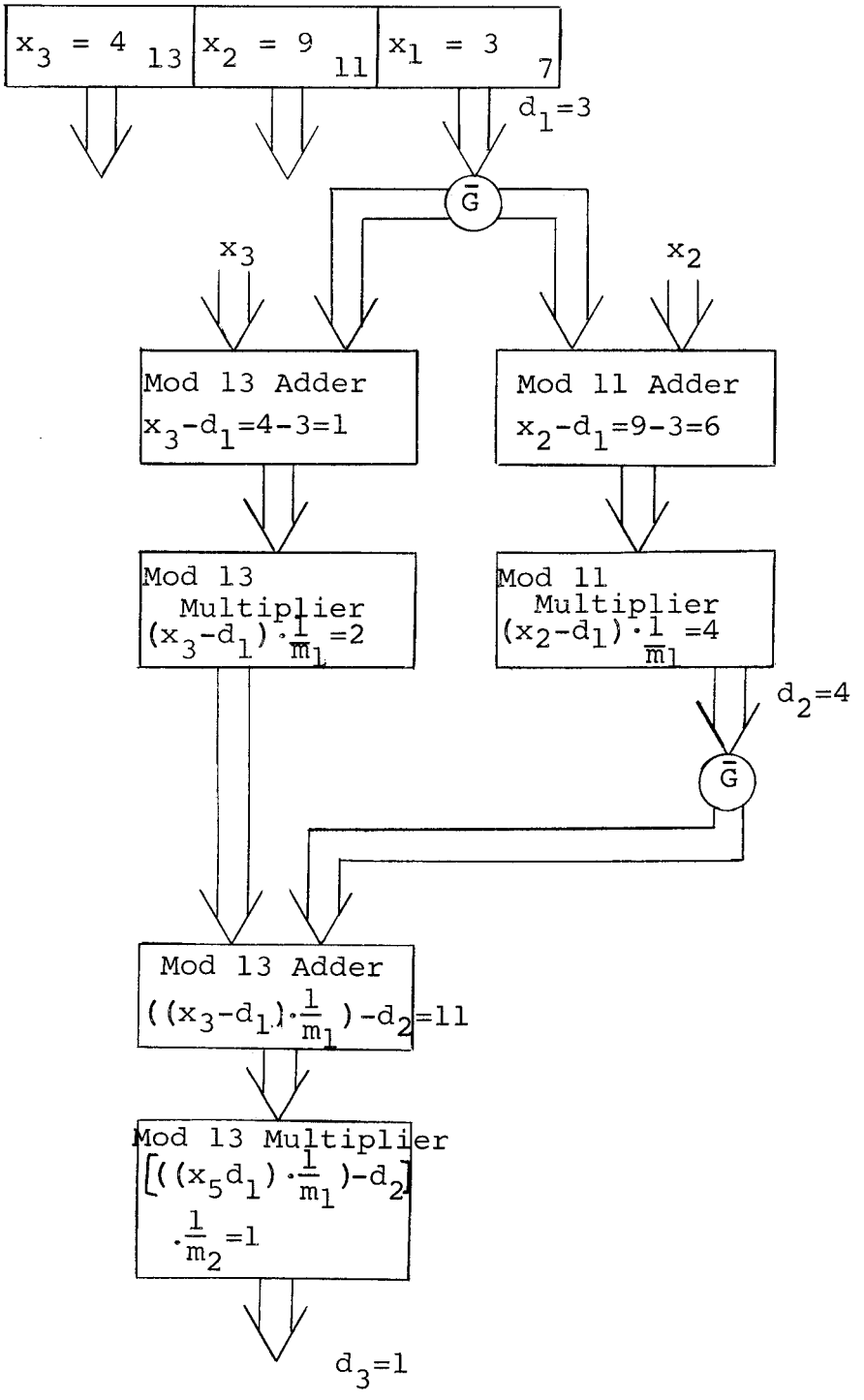


Figure 1. Conversion of the residue number 4/9/3 to the mixed radix number 1/4/3.

are added in a four-bit adder the correct sum, nine, is obtained. When the two numbers four and seven are added the sum, 11, is obtained. This sum is not correct for a mod 11 adder since the correct sum is zero. The logic necessary to correct for this condition and other incorrect sums involves adding five to the incorrect sum to obtain the correct sum.

Incorrect Sums	Incorrect Binary Representation	Correction Necessary	Correct Representation
11	1011	add five	0000
12	1100	↓	0001
13	1101	↓	0010
14	1110	↓	0011
15	1111	↓	0100
16	0000	↓	0101
17	0001	↓	0110
18	0010	↓	0111
19	0011	↓	1000
20	0100	↓	1001

By examining the examples shown the logic required for correction is significant. This is especially true when it is noted that some incorrect binary representations can also be correct binary representations.

Without a doubt, the mod 11 adder is "inefficient". As emphasized by R. D. Merrill, Jr. (5) the most "efficient" moduli should be chosen for a specific residue number system. Obviously the moduli such as 2^k or 2^k-1 are most "efficient". It must be remembered that only one modulus in residue notation can be of the form 2^k. This is to fulfill the requirement that all moduli

must be relatively prime (or prime) for a non-redundant residue system.

A mod 15 adder (of the form 2^k-1) needs correction for the "all one" condition, 1111. An AND gate can be used to sense this condition producing a carry-in to the least significant stage of the adder. Of course, this correction is not necessary if zero is defined as 0000 or 1111. If any carry-outs from the most significant stage of the four-bit adder are "wired" to the carry-in position of the least significant stage, the adder will function properly as a mod 15 (2^k-1) adder.

II. A SPECIFIC RESIDUE ARITHMETIC UNIT

Now that the basic rules and problems of residue arithmetic have been discussed, the logic circuit design of a residue arithmetic unit will be pursued.

Choosing the Moduli

As discussed earlier, the moduli should be of the form 2^k or $2^k - 1$ to be efficient. The number of moduli should be kept small to make number conversion simpler. On the other hand, if too few moduli are used, each modulus adder will have so many bit positions that the inherent advantage of residue arithmetic will be lost. Another requirement for the moduli is the necessity of having the moduli relatively prime. Also, to make efficient use of a given number of binary storage devices, N, it is desirable to make the product of the moduli, M, as close to $2^N - 1$ as possible.

While considering different moduli, it is also very important to consider the specific ordering $(m_1, m_2, \cdots)$ of every set of moduli considered. For any given set of moduli it is quite possible that one particular ordering of the moduli will greatly simplify the process of mixed radix conversion. The conversion from a conventional binary number to a residue number can be simplified with similar considerations.

Using the criteria given, a specific set of four moduli was chosen where $m_1 = 1024$, $m_2 = 2047$, $m_3 = 1023$ and $m_4 = 511$. Note that the largest decimal number which can be represented uniquely by this set of moduli is M-1 = 1,095,757,200,383. This compares favorably with the maximum number represented by an "equivalent" 41-bit "conventional" word where $2^{40}-1 = 1,099,511,627,775$.

Since the modified sign and complement notation will be used, only one-fourth of the range, 0 - (M-1), inclusive, can be used. Numbers 0 - (M/4-1), inclusive, represent positive numbers (sign bit is "0") and (M - (M/4-1)) - (M-1), inclusive, represent negative numbers (sign bit is "1").

Addition

Since the modulus 1024 is of the form 2^k where k is equal to ten, addition with the modulus 1024 adder is analogous to a ten-bit adder (0-9 bit positions) with one exception. Any carries from the most significant bit position are ignored.

The remaining three moduli are of the form 2^k-1 where k is equal to 11, ten, and nine for the 2047, 1023, and 511 respectively. Addition is similar to conventional k-bit binary addition with end-around carry. This is more readily seen by examining 2^k-1 where k = 3.

Since $|8|_7 = 1$, the result has to be changed from zero which normally represents eight in a three-bit adder to one by adding a one to the carry-in position. It is evident that zero will exist in two forms: (1) as zero and (2) as 2^k-1 or seven in the case of the three-bit adder. This double zero condition can easily be corrected by using simple logic to sense the 2^k-1 condition (111 for the three-bit adder) creating an output which clears the adder and at the same time prevents an end-around carry.

Subtraction

Subtraction is performed by adding the additive inverse of the subtrahend to the minuend. The additive inverse for subtracting mod 2^k is $2^k - |x|_{2^k}$ or the two's complement of the binary representation for $|x|_{2^k}$. Likewise for subtracting mod 2^k-1 the additive inverse is $2^k-1 - |x|_{2^k-1}$ which is the one's complement of the binary representation of $|x|_{2^k-1}$.

When subtraction is performed in the modified sign and complement residue notation the sign bit of the subtrahend must be complemented (triggered), also.

Accumulators

In the following sections the frequent use of the

word accumulator will be noted. Any one of the numerous designs existing could be used for the accumulators. It should be recalled that each stage of an accumulator has a flip-flop which holds the binary bits of the augend. Addition occurs when the addend is gated to the accumulator and an add command and carry command (if necessary) occur at the appropriate times. In the discussions to follow, it is assumed the add and carry commands required are derived from the signal causing the transfer of the addend to the accumulator. If the one's complement of an addend is gated to the mod 1024 accumulator, it is assumed the two's complement is "automatically" formed by simultaneously adding a "one" to the least significant stage.

The accumulators are synchronous where the time for addition or subtraction is determined by the maximum carry propagation time. Thus, the time for addition or subtraction is roughly proportional to the number of carries possible in the accumulator. For the set of four moduli considered, addition or subtraction can be accomplished in approximately one-fourth the time required for conventional 41-bit addition or subtraction. This assumes no additional time is required for end-around carry.

Multiplication

Conventional "bit-by-bit" parallel binary multiplication of two 41-bit words involves addition and shifting. This results in a product of 2n-1 = 81 bits including the sign bit (considering each 41-bit operand to have one sign bit position).

With residue multiplication this addition and shifting procedure is done for each pair of corresponding modulus digits as previously discussed. Since the multiplication with respect to each modulus can be accomplished in parallel, the time for multiplication is approximately $(\frac{10}{40})^2 = \frac{1}{16}$ of the time required for multiplication of a conventional 41-bit multiplicand and 41-bit multiplier. This comparison is made on the assumption that synchronous adders or accumulators are used without carry-look-ahead in both the residue and conventional systems.

Another advantage besides the faster multiplication is the simplification of control logic necessary for multiplication. Residue multiplication using the modified sign and complement notation can be considered as the multiplication of two positive numbers. Thus, any algorithms peculiar to a particular type of multiplication involving negative numbers is avoided. This applies whether the operand(s) are changed to positive numbers

$(0 \leq \text{operand} < \frac{M}{4})$ or left as negative numbers $(\frac{3}{4} M \leq \text{operand} < M)$.

Before multiplication is begun, the operand sign bits are inspected. If a sign bit is "one", the non-sign bit positions are complemented. Thus, multiplication is always accomplished with "positive" numbers. The original operand sign bits are retained so that the product (providing no overflow has occurred) may be complemented and given a negative sign bit if the original operand sign bits are unlike in sign.

To clarify the basic mechanics of residue multiplication, an arrangement of registers and accumulators together with the multiplication control logic is shown in Figure 2.

Observation of Figure 2 reveals that each "modulus multiplier" terminates multiplication at different times (except for m_1 and m_3) since the length of the multipliers is ten, 11, ten, and nine binary bits in length for moduli m_1, m_2, m_3, and m_4 respectively.

The pulse generator, PG, generates one pulse upon receipt of a multiply command. The sequence of add and shift or just shift is repeated until the decoder decodes a count of 11, which terminates the multiplication. Of course, multiplication is terminated in each modulus multiplier when the counter attains the appropriate count.

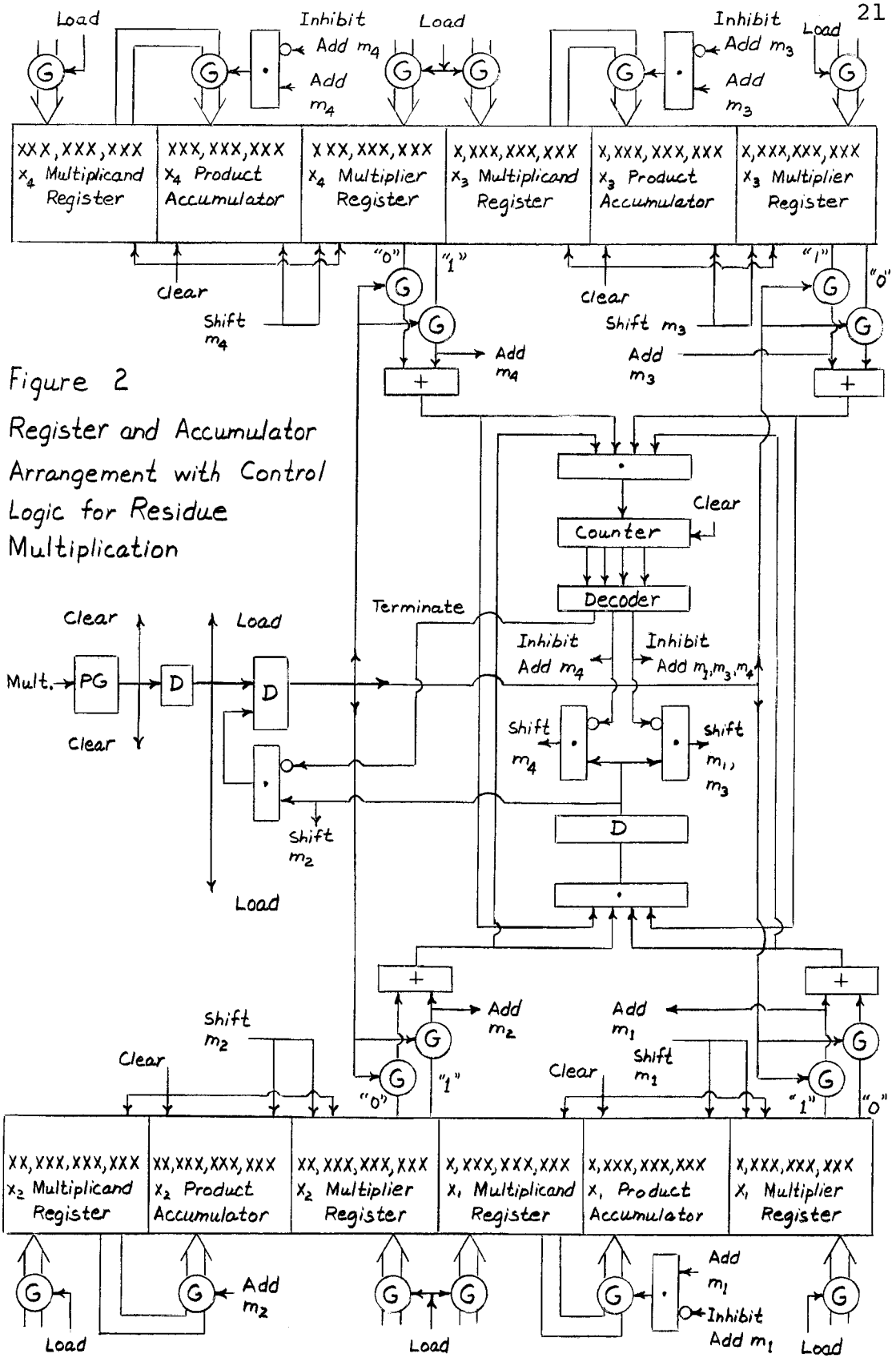


Figure 2

Register and Accumulator Arrangement with Control Logic for Residue Multiplication

Since the residue arithmetic unit handles whole numbers, all shifts for multiplication are left-shifts with circular shifts for the mod 2^k-1 accumulators. The left-shifts are analogous to the right-shifts required in a fractional computer.

The details of overflow detection in multiplication will be investigated in a later section.

Mixed Radix Converter

It has been shown that mixed radix conversion is basically a series of subtractions and multiplications. To shorten the conversion process it is desirable to pick a set of moduli where the multiplications involve multipliers which are powers of two. The multiplications now become merely shift operations.

For moduli $m_1 = 1024$, $m_2 = 2047$, $m_3 = 1023$, and $m_4 = 511$, the multiplicative inverses are as follows:

$$\left|\frac{1}{1024}\right|_{2047} = 2^1 \qquad \left|\frac{1}{2047}\right|_{1023} = 2^0$$

$$\left|\frac{1}{1024}\right|_{1023} = 2^0 \qquad \left|\frac{1}{2047}\right|_{511} = 341$$

$$\left|\frac{1}{1024}\right|_{511} = 2^{-1} \text{ or } 2^8 \qquad \left|\frac{1}{1023}\right|_{511} = 2^0$$

With this set of moduli all multiplications are shift operations except one where a multiplication by 341 mod 511 is required. However,

$|341|_{511}$:= 101010101 is easily mechanized with a multiplier using a sequence of "add-shift left two bit positions".

As an example of a residue to mixed radix conversion using the multiplicative inverses given, the decimal number $\frac{M}{4}$ = 273,939,300,096 will be written in residue form and then converted to mixed radix notation.

$\frac{M}{4} := x_4 / x_3 / x_2 / x_1 = 0 / 0 / 0 / 768$ in residue notation. The problem is to convert to mixed radix notation where $\frac{M}{4} := d_4 / d_3 / d_2 / d_1$ as shown on page 24.

Therefore, $\frac{M}{4} := d_4 / d_3 / d_2 / d_1 = 127/767/511/768$

Check: $\frac{M}{4} = 127\ (1023 \cdot 2047 \cdot 1024) + 767\ (2047 \cdot 1024) + 511\ (1024) + 768$

$\frac{M}{4} = 273,939,300,096$

Subtractions are made by forming the additive inverse with respect to the appropriate moduli. Since the moduli involved with subtractions are of the form 2^k-1, the additive inverse is obtained by taking the one's complement of the subtrahend. It is important to realize that the subtrahend must be placed in proper residue form (with respect to m_2, m_3, or m_4) before complementing. Since the modulus accumulators differ in length, two secondary accumulators are required to properly form the one's complements of the subtrahends.

Shifting operations require circular shifts. In

Mod 511

$\underline{x_4 = 0}$

$x_4 - d_1 = 0 + (511 - |768|)$

$= 254$

$\left|\frac{x_4 - d_1}{m_1}\right|_{m_4} = |254 \cdot 256|$

$= |65,024| = 127$

$\left|\frac{x_4 - d_1}{m_1}\right|_{m_4} - d_2 =$

$127 + (511 - 511)$

$= 127$

$\left|\left(\left|\frac{x_4 - d_1}{m_1}\right|_{m_4} - d_2\right) \cdot \frac{1}{m_2}\right|_{m_4} =$

$|127 \cdot 341| = |43,307|$

$= 383$

$\left|\left|\left(\left|\frac{x_4 - d_1}{m_1}\right|_{m_4} - d_2\right) \cdot \frac{1}{m_2}\right|_{m_4} - d_3\right|_{m_4} =$

$383 + (511 - |767|) = 127$

$\left|\frac{\left|\frac{\left|\frac{x_4 - d_1}{m_1}\right|_{m_4} - d_2}{m_2}\right|_{m_4} - d_3}{m_3}\right|_{m_4}$

$= 127 \cdot 1 = \underline{127} = d_4$

Mod 1023

$\underline{x_3 = 0}$

$x_3 - d_1 = 0 + (1023 - 768)$

$= 255$

$\left|\frac{x_3 - d_1}{m_1}\right|_{m_3} = |255 \cdot 1|$

$= 255$

$\left|\left|\frac{x_3 - d_1}{m_1}\right|_{m_3} - d_2\right| =$

$255 + (1023 - 511)$

$= 767$

$\left|\left(\left|\frac{x_3 - d_1}{m_1}\right|_{m_3} - d_2\right) \cdot \frac{1}{m_2}\right|_{m_3} =$

$767 \cdot 1 =$

$\underline{767} = d_3$

Mod 2047

$\underline{x_2 = 0}$

$x_2 - d_1 = 0 + (2047 - 768)$

$= 1279$

$\left|\frac{x_2 - d_1}{m_1}\right|_{m_2} = |1279 \cdot 2|$

$= \underline{511} = d_2$

Mod 1024

$\underline{x_1 = 768 = d_1}$

$x_1 = 768 = d_1$

other words, any bits shifted out of the most significant bit position or the least significant bit position must be recirculated into the least significant and most significant bit positions respectively.

An arrangement of accumulators for a mixed radix converter (MRC) is shown in Figure 3.

Referring to Figure 3, a typical conversion process is outlined for the MRC shown.

Timing Signal	Action(s) Taken
t_o	Clear all accumulators and mod 1024 register.
t_1	Add x_1, x_2, and x_4 to the appropriate primary accumulators, mod 1024 register and secondary mod 511 accumulator.
t_2	Add the one's complement of the mod 1024 register to the primary mod 2047 and 1023 accumulators. Add the one's complement of the secondary mod 511 accumulator to the primary mod 511 accumulator.
t_3	Shift the contents of the mod 2047 primary accumulator left one bit position (SL - 1). Shift the contents of the primary mod 511 accumulator right one bit position (SR - 1). Clear the secondary accumulators.
t_4	Add the contents of the primary mod 2047 accumulator, d_2, to the secondary accumulators.
t_5	Add the one's complement of the secondary mod 1023 and 511 accumulators to primary mod 1023 and 511 accumulators.
t_6	Clear the secondary accumulators.

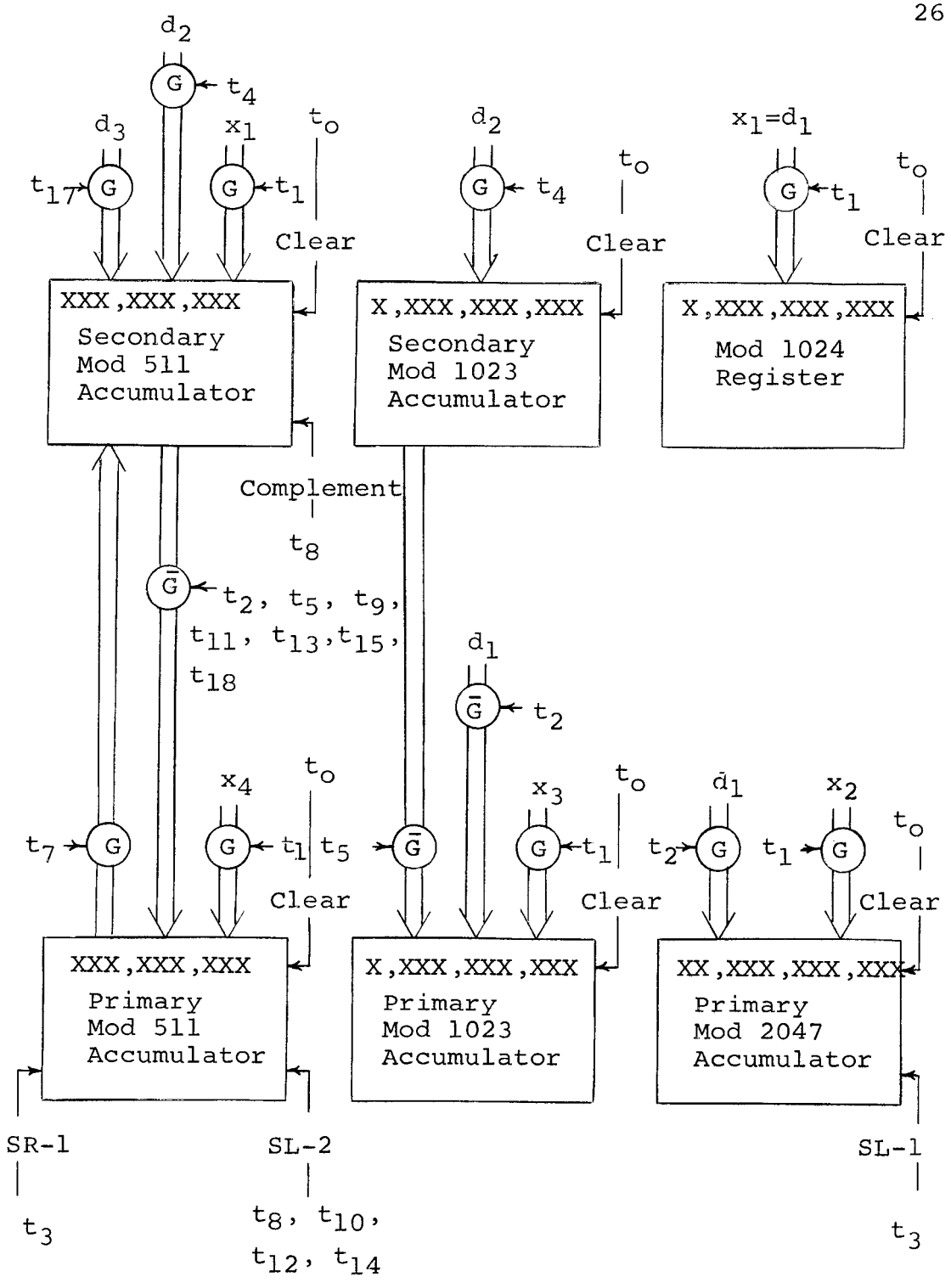


Figure 3. Accumulators and register required for a mixed radix converter using $m_1 = 1024$, $m_2 = 2047$, $m_3 = 1023$, and $m_4 = 511$.

t_7	Add the contents of the primary mod 511 accumulator to the secondary mod 511 accumulator.
t_8	Form the one's complement of the contents of the secondary mod 511 accumulator. Shift the contents of the primary mod 511 accumulator left two bit positions (SL - 2).
t_9	Add the one's complement of the secondary mod 511 accumulator to the primary mod 511 accumulator.
t_{10}	Shift the contents of the primary mod 511 accumulator left two bit positions (SL - 2).
t_{11}	Same as t_9.
t_{12}	Same as t_{10}.
t_{13}	Same as t_9.
t_{14}	Same as t_{10}.
t_{15}	Same as t_9.
t_{16}	Clear the secondary mod 511 accumulator.
t_{17}	Add the contents of the primary mod 1023 accumulator, d_3, to the secondary mod 511 accumulator.
t_{18}	Same as t_9.

Examination of the functions of each accumulator reveals the following facts:

(1) Primary accumulators mod 511, mod 1023, and mod 2047 are the same as the $2^k - 1$ accumulators previously discussed. In addition, the mod 2047 requires a shift left one-bit position (SL - 1) capability and the mod 511 requires a shift right one-bit position (SR - 1) capability. The primary mod 2047 accumulator also requires the 11th bit complemented for the actions at t_2.

(2) The secondary mod 1023 accumulator requires only the capability of adding the first ten binary bits of the primary mod 2047 accumulator to a carry-in to the least significant bit position. Note that a carry-in may exist if the 11th bit of the primary mod 2047 accumulator contains a "one". The presence of an 11th bit "one" at the least significant stage carry-in should be slightly delayed to prevent simultaneous occurrence with a "full count" condition carry-in (see section on addition).

(3) The secondary mod 511 accumulator requires capabilities similar to the secondary mod 1023 accumulator except that carry-ins may exist for the two least significant bit positions. This, of course, is due to the 11 bit positions of the mod 2047 accumulator while the mod 511 accumulator has only nine bit positions. To prevent an almost simultaneous occurrence of two carry-ins to the second stage of the secondary mod 511 accumulator, a slight delay in the appearance of the 11th bit to the second stage carry-in is desirable. Analogous to the secondary mod 1023 accumulator the presence of the tenth bit "one" to the least significant stage carry-in position should be delayed. Of course, the 11th bit carry-in to the second least significant stage must be delayed sufficiently to allow for the possibility of two other carry-ins to occur to the second least significant stage.

The number of residue additions required for mixed radix conversion totals nine. Note that any number added

to zero is considered as a transfer, not an addition.

The timing signals can be easily derived from the signal that gates the residue number into the MRC. The input gating signal can be used immediately to clear the accumulators and register before the residue number is actually gated into the MRC. In the illustrations that follow only the input gating signal to the MRC is shown. It is understood that a clear signal, t_o, to the MRC occurs slightly before the gating signal gates the input residue number.

Appendix I shows the contents of each accumulator for each timing interval in the conversion of $\frac{M}{4}$.

Overflow

Now that addition, subtraction, multiplication, and conversion to mixed radix notation have been discussed, it is necessary to investigate the problem of overflow.

Addition and Subtraction

In the residue number system it is apparent that the carry-outs from the various modulus accumulators during addition or subtraction give no indication of overflow. These carry-outs only indicate that a particular modulus has been exceeded. They give no indication that the result has overflowed. This is contrary to the similar condition with conventional weighted number systems where

a carry-out from the most significant stage indicates overflow has occurred.

The maximum correct result when both operands are positive is $\frac{M}{4} - 1$. However, an incorrect positive result, $\frac{M}{4} - 1 \lessdot (+)\ \text{result} \lessdot \frac{M}{2}$, is possible. This corresponds to an incorrect negative result, $\frac{M}{2} \leqslant (-)\ \text{result} \lessdot \frac{3}{4} M + 1$, when the operands are negative. Thus, only "one place" of overflow can occur in addition or subtraction.

It is easily seen that overflow occurs when the result is in the interval $\frac{M}{4} - 1 \lessdot \text{result} \lessdot \frac{3}{4} M + 1$.

Overflow = $(\text{Result} \geqslant \frac{M}{4}) \cdot (\text{Result} \leqslant \frac{3}{4} M)$.

From the preceding section $\frac{M}{4}$:= 127 / 767 / 511 / 768 := 001111111 / 101111111 / 00111111111 / 1100000000 in mixed radix notation.

Since $\frac{M}{4}$:= $\underbrace{00}_{\bar{a}\bar{b}}\underbrace{1111111}_{A}$ / $\underbrace{10}_{c\bar{d}}\underbrace{111111111}_{B}$ / $\underbrace{00}_{\bar{e}\bar{f}}\underbrace{11111111111}_{C}$ / $\underbrace{11}_{gh}\underbrace{00000000}_{\bar{D}}$

$$(\text{Result} \geqslant \tfrac{M}{4}) = \bar{a}\,\bar{b}\,A\,c\,\bar{d}\,B\,\bar{e}\,\bar{f}\,C\,g\,h\,\bar{D} + a + b + \bar{a}\,\bar{b}\,A\,c\,d + \bar{a}\,\bar{b}\,A\,c\,\bar{d}\,B\,e + \bar{a}\,\bar{b}\,A\,c\,\bar{d}\,B\,\bar{e}\,f + \bar{a}\,\bar{b}\,A\,c\,\bar{d}\,B\,\bar{e}\,\bar{f}\,C\,g\,h\,D.$$

Simplifying: $(\text{Result} \geqslant \frac{M}{4}) = Ac\left[B\,(Cgh + e + f) + d\right] + a + b.$

$\frac{3}{4}$ M := 383 / 255 / 1535 / 256 in mixed radix notation.

Thus, $\frac{3}{4}M$:= $\underbrace{10}_{a\bar{b}}\underbrace{1111111}_{A}$ / $\underbrace{00}_{\bar{c}\bar{d}}\underbrace{1111111111}_{B}$ / $\underbrace{10}_{e\bar{f}}\underbrace{1111111111}_{C}$ / $\underbrace{01}_{\bar{g}h}\underbrace{00000000}_{\bar{D}}$.

$$(\text{Result} \leq \tfrac{3}{4} M) = a\,\bar{b}\,A\,\bar{c}\,\bar{d}\,B\,e\,\bar{f}\,C\,\bar{g}\,h\,\bar{D} + \bar{a} + a\,\bar{b}\,\bar{A} + a\,\bar{b}\,A\,\bar{c}\,\bar{d}\,\bar{B} + a\,\bar{b}\,A\,\bar{c}\,\bar{d}\,B\,\bar{e} + a\,\bar{b}\,A\,\bar{c}\,\bar{d}\,B\,e\,\bar{f}\,\bar{C} + a\,\bar{b}\,A\,\bar{c}\,\bar{d}\,B\,e\,\bar{f}\,C\,\bar{g}\,\bar{h}.$$

Simplifying: $(\text{Result} \leq \tfrac{3}{4} M) = \bar{b}\,\bar{c}\,\bar{d}\left[\bar{f}\left\{\bar{g}\,(\bar{D} + \bar{h}) + \bar{C}\right\} + \bar{B} + \bar{e}\right] + \bar{a} + \bar{b}\,\bar{A}.$

Since overflow = $(\text{Result} \geq \tfrac{M}{4}) \cdot (\text{Result} \leq \tfrac{3}{4} M)$,

$$\text{Overflow} = A\,c\,\bar{a}\left[d + B\,(C\,g\,h + e + f)\right] + a\,\bar{b}\,\bar{c}\,\bar{d}\left[\bar{f}\,(\bar{g}\left\{\bar{D} + \bar{h}\right\} + \bar{c}) + \bar{B} + \bar{e}\right] + a\,\bar{b}\,\bar{A} + b\,\bar{a}.$$

Twenty AND gates and six OR gates with a fan-in of five or less are sufficient to implement the equation for additive overflow.

Multiplication

In multiplication overflow is not limited to "one place" of overflow. The absolute value of the product may be as great as $(\frac{M}{4} - 1)^2$. Thus, the problem of multiplicative overflow is considerably more complicated than additive overflow.

The detection of additive overflow for the sign and complement notation requires less logic gates than for the modified sign and complement notation (11 AND and three OR gates versus 20 AND and six OR gates). However, the increased number of gates required for the modified sign and complement notation is more than offset by the easier detection of multiplicative overflow.

To begin the discussion of multiplicative overflow

several terms are defined. Let: Z^1 = the machine product of X·Y which may be incorrect due to the possibility of overflow. Let: $2^{E(X)-1} < X \leq 2^{E(X)}$; $2^{E(Y)-1} < Y \leq 2^{E(Y)}$; $2^{E(K)} < \frac{M}{4} \leq 2^{E(K)+1}$.

By examining the relationships given and remembering that all negative operands are changed to positive operands before multiplication, except for the sign bit, the following facts are revealed:

(1) If $Z^1 < X$ or $Z^1 < Y$ or $Z^1 = 0$ when $X \neq 0$ and $Y \neq 0$, overflow has occurred.

(2) If $Z^1 \geq X$ and $Z^1 \geq Y$, overflow may have occurred.

a. If $E(X) + E(Y) \leq E(K)$, no overflow occurs.

b. Noting that X and Y may be as small as $2^{E(X)-1} + 1$ and $2^{E(Y)-1} + 1$ respectively, the condition where $E(K) < E(X) + E(Y) < E(K) + 3$ may indicate overflow has occurred.

c. If $E(X) + E(Y) \geq E(K) + 3$, overflow has occurred.

Figure 4 shows the logic necessary for determining E (X) and E (Y). The inputs to the logic blocks are the mixed radix digits from the primary accumulators and mod 1024 register of the mixed radix converter. Only the logic for the inputs from d_4 (located in the primary mod 511 accumulator) is shown. The numbers six, five, etc. are the 2^6 2^5, etc. binary "zero" inputs from the primary mod 511 accumulator. Since X or Y is $< \frac{M}{4} = 273,939,300,096 < 2^{38}$, the largest "one" binary bit

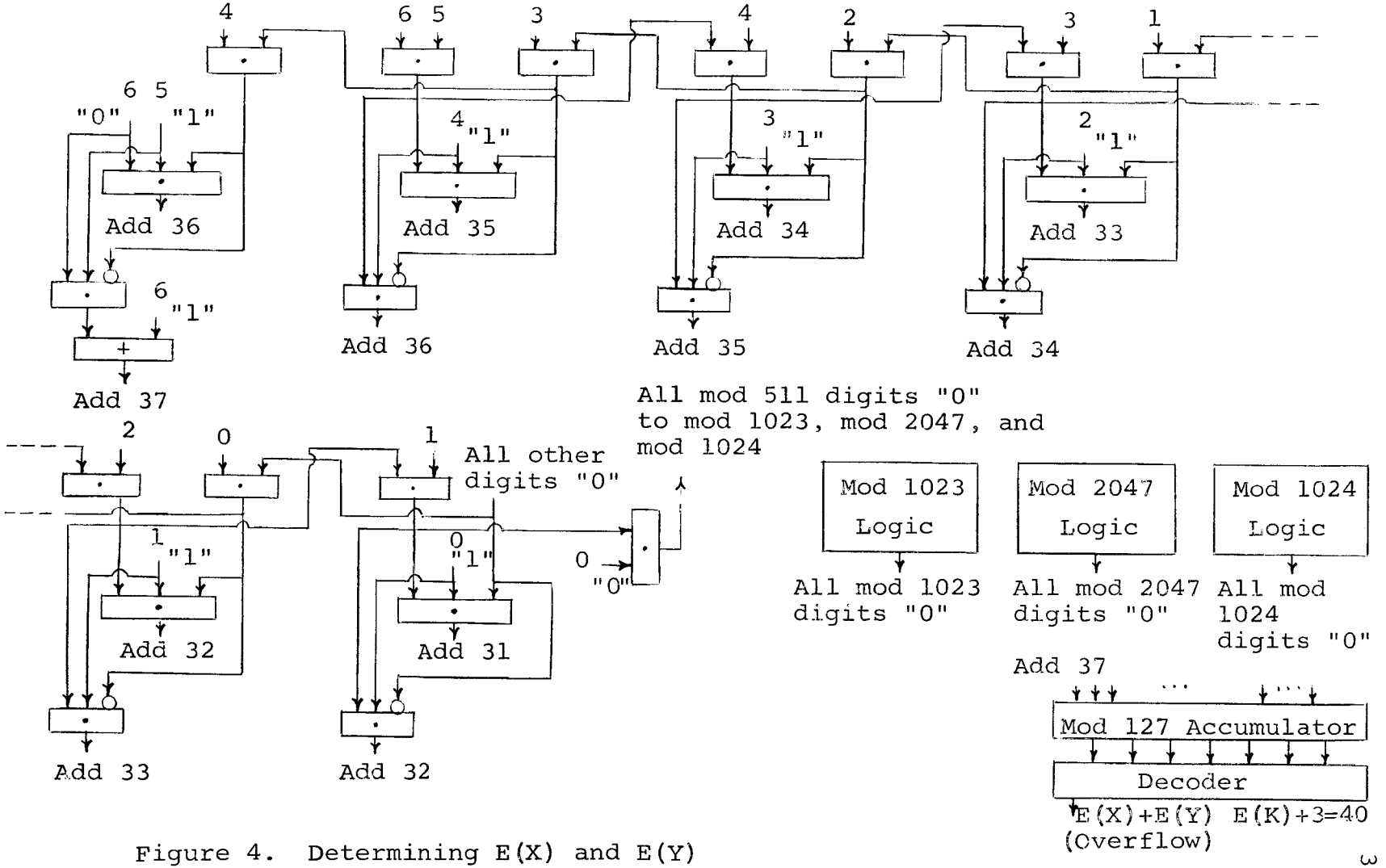


Figure 4. Determining E(X) and E(Y)

possible in the primary mod 511 accumulator after conversion of X and Y is 2^6. This is true since d_4 is weighted by $1023 \cdot 2047 \cdot 1024 = (2^{10} - 1)(2^{11} - 1) \cdot 2^{10} < 2^{31}$. Also, $d_4 = 127$ was previously determined in the example given for mixed radix conversion. Therefore, 2^6 corresponds to a value less than 2^{37}.

The inputs labeled "all other digits zero" are inputs that indicate all the remaining binary bits in the other modulus accumulators are "zero". An "all mod 511 digits 'zero'" indication is needed for inputs to the mod 1023, 2047, and 1024 logic as shown. It is now obvious that the mod 1023 and mod 2047 logic require "all digits of greater significance zero" and "all inputs of less significance zero" inputs.

The outputs labeled "add 37", "add 36", etc. are the values of E (X) or E (Y) determined by the logic. Note that only one value of E (X) or E (Y) is determined for any given value of X or Y. The two values determined for E (X) and E (Y) are then added to a mod 127 parallel accumulator which gives the sum of E (X) + E (Y). A decoder decodes the contents of the accumulator after completion of addition giving an output overflow alarm for $E(X) + E(Y) \geq E(K) + 3$.

The logic for determining E (X) or E (Y) requires 136 AND gates and four OR gates with a maximum fan-in of

three inputs and a maximum fan-out of three outputs.

Detection of multiplicative overflow in the interval $E(K) \leq E(X) + E(Y) \leq E(K) + 3$ appears more difficult. However, upon closer examination of the interval any overflow occurring can quite easily be detected. If $E(X) + E(Y) = E(K) + 1$, then $2^{E(K)-1} < X \cdot Y \leq 2^{E(K)+1} \leq \frac{M}{2}$. If $E(X) + E(Y) = E(K) + 2$, then $2^{E(K)} < X \cdot Y \leq 2^{E(K)+2} < M$. Thus any overflow is detected by $X \cdot Y = Z^1 \geq \frac{M}{4}$. This, of course, applies when all negative operands are changed to positive operands, except the sign bit, before multiplication. Complementing negative operands is shown in Figure 5.

The conditions where $Z^1 < X$ or $Z^1 < Y$ can be determined by subtracting the larger operand from Z^1 and noting the sign of the result. If the result is negative, overflow occurs. Overflow can be quickly determined for $Z^1 = 0$ when $X \neq 0$ and $Y \neq 0$ since the residue representation of zero is 0/0/0/0. The logic for this, although not shown, can easily be accomplished.

Figure 5 shows the procedure and equipment used for detecting $Z^1 \geq \frac{M}{4}$ for $E(X) + E(Y) = E(K) + 1$ or $E(K) + 2$ and $Z^1 < X$ or $Z^1 < Y$. Two mixed radix converters, an X register, a Z secondary accumulator, and logic for detecting numbers $\geq \frac{M}{4}$ are utilized. Only one MRC converter is required, but the additional equipment enables detection of multiplicative overflow to be accomplished

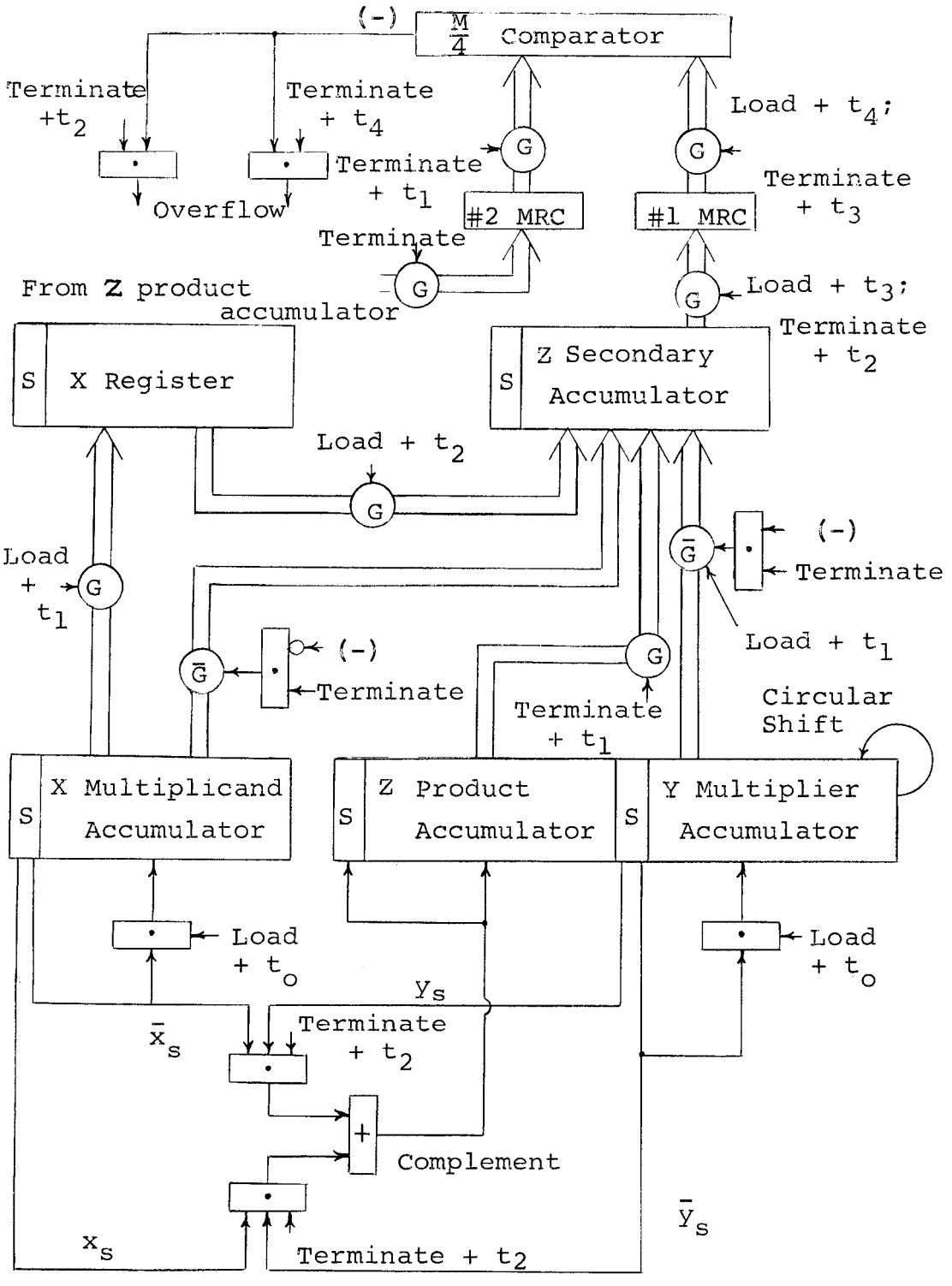


Figure 5. Complementing Negative Operands, Correcting the Product, and Determining if $Z^1 < X$, $Z^1 < Y$, or $Z^1 \geq \frac{M}{4}$ for Multiplication.

approximately three times faster. Further justification of the additional equipment will become apparent when division is discussed.

Examination of the equation for "Result $\geq \frac{M}{4}$" developed in the previous section indicates the detection of numbers $\geq \frac{M}{4}$ can be accomplished with 12 AND gates and three OR gates with a fan-in and fan-out of five or less.

Note that the various modulus accumulators are not shown in Figure 5, but are shown in one block as the X multiplicand accumulator, etc. Also, note that Y is available for overflow detection since a circular shift is used in the Y multiplier accumulator.

The timing pulses shown can easily be derived from the timing required for multiplication. The words "load" and "terminate" refer to Figure 2 describing multiplication.

Division

The problem is to find $Z = \frac{X}{Y}$. From discussions in the literature (5,6) it is known that Z can be solved quite readily if X is evenly divisible by Y and Y is relatively prime with all the moduli. By forming the multiplicative inverse of Y, $\left|\frac{1}{Y}\right|_{m_i}$, $\left|Z\right|_{m_i}$ is determined simply by performing $\left|\left|X\right|_{m_i} \cdot \left|\frac{1}{Y}\right|_{m_i}\right|_{m_i}$. If X is not evenly divisible by Y or $\left|\frac{1}{Y}\right|_{m_i}$ does not exist, then the

division can not be performed in this manner.

A few examples of the preceding facts are shown:

$\left|\frac{8}{3}\right|_{11} = \left|8\cdot 4\right|_{11} = \left|32\right|_{11} = 10$ which is incorrect.

$\left|\frac{6}{3}\right|_{11} = \left|6\cdot 4\right|_{11} = \left|24\right|_{11} = 2$ which is correct.

$\left|\frac{6}{3}\right|_{9} = \left|6\cdot\left|\frac{1}{Y}\right|_{9}\right|_{9}$ which does not exist since $Y = 3$ is not relatively prime with $m = 9$.

In the multiple radix or residue notation it must be remembered that these facts apply to each modulus. The prime factors of the present set of moduli, where $m_1 = 1024$, $m_2 = 2047$, $m_3 = 1023$, and $m_4 = 511$, are 2, 3, 7, 11, 23, 31, 73, 89. Consequently, $\left|\frac{1}{Y}\right|_{m_i}$ frequently does not exist. In fact, $\left|\frac{1}{Y}\right|_{m_i}$ only exists approximately 22% of the time.

If $\left|\frac{1}{Y}\right|_{m_i}$ does exist, the next step is to form $Z^1 = \left|\left|X\right|_{m_i}\cdot\left|\frac{1}{Y}\right|_{m_i}\right|_{m_i}$ for all the moduli. In other words, the machine product $Z^1 = X\cdot\left|\frac{1}{Y}\right|_{m_i}$ is formed. Thus, $Z^1\cdot Y = CM + X$ where C is a unique integer of $Z^1 = \frac{CM+X}{Y}$. If Z^1 is the correct quotient ($Z^1 = Z = \frac{X}{Y}$), then $C = 0$. It is now easily seen that to determine if X is evenly divisible by Y ($Z^1 = Z = \frac{X}{Y}$) requires the detection of multiplicative overflow. Therefore, if no multiplicative overflow occurs in forming the product

$Z^1 = X \cdot \left|\frac{1}{Y}\right|_{m_i}$, C = 0 and Z^1 is the correct quotient of $\frac{X}{Y}$. This is essentially the same reasoning used in the proof of Theorem I by Y. A. Keir, P. W. Cheney, and M. Tannenbaum (6).

To summarize, the following can be stated. If $\left|\frac{1}{Y}\right|_{m_i}$ exists and no multiplicative overflow occurs in forming the machine product $Z^1 = X \cdot \left|\frac{1}{Y}\right|_{m_i}$, $Z^1 = Z = \frac{X}{Y}$. Thus, under these conditions division can be accomplished in the time necessary to obtain $\left|\frac{1}{Y}\right|_{m_i}$, form the product Z^1, and examine for multiplicative overflow. The quickest and most economical method for obtaining $\left|\frac{1}{Y}\right|_{m_i}$ is by addressing a core memory. Unfortunately it appears to be the best method known.

Obviously another method of division must be used if $\left|\frac{1}{Y}\right|_{m_i}$ does not exist or X is not evenly divisible by Y. A method which consists of finding the binary residues of Z can be used. This method is not new, but the procedure utilized is simpler than the method found in the literature (6).

Any number, such as Z, consists of the sum of various powers of two, 2^J. The number Z can be found by simply finding the required values of 2^J, where J = 1, 2, 3 $\cdots$37 for the number system currently considered, and summing the values found.

The first step to accomplish is to determine the maximum value of 2^J for a given number, Z. This maximum value of 2^J is called $2^{E(Z)-1}$, where $2^{E(Z)-1} \leq Z < 2^{E(Z)}$. Recalling the logic required in forming E (X) and E (Y) suggests a simple method for obtaining $2^{E(Z)-1}$. Since $2^{E(Z)-1} = \frac{2^{E(X)-1}}{2^{E(Y)-1}}$ or $= \frac{1}{2} \cdot \frac{2^{E(X)-1}}{2^{E(Y)-1}}$, the importance of $E(X) - 1 - [E(Y) - 1] = E(X) - E(Y)$ is apparent. $E(X) - E(Y)$ can easily be formed by first adding the one's complement of E (Y) to the mod 127 accumulator. Then, E (X) is added to the accumulator. If the sign of $X - 2^{E(X)-E(Y)} \cdot Y$ is found to be positive, then $2^{E(Z)-1} = 2^{E(X)-E(Y)}$. Otherwise, $2^{E(Z)-1} = \frac{1}{2} \cdot 2^{E(X)-E(Y)}$.

The second largest value of 2^J is found by testing $(X - 2^{E(X)-E(Y)} \cdot Y) - 2^{E(X)-E(Y)-1} \cdot Y$ or $(X - 2^{E(X)-E(Y)-1} \cdot Y) - 2^{E(X)-E(Y)-2}$. If the quantity formed is positive, then $2^{E(X)-E(Y)-1}$ or $2^{E(X)-E(Y)-2}$ is the second largest value of 2^J. If the quantity is negative, $2^{E(X)-E(Y)-2}$ is tested. This process continues until 2^0 is tested.

The accuracy of the quotient, as in conventional systems, depends on the accuracy in scaling and whether X is evenly divisible by Y. Unlike conventional methods of division in fixed point machines, the scaled value of X should be greater than Y. For greatest accuracy X and

Y should be scaled as follows: $2^{E(K)} \leq X < \frac{M}{4}$ and $0 < Y \leq 2^{E(\frac{K-1}{2})}$. It must be remembered that this fixed point arithmetic unit is defined to compute in whole numbers rather than fractions.

Since there is not a fixed base in residue notation, scaling can not be accomplished by multiplying or dividing by powers of the fixed base. Instead, scaling is accomplished by multiplying or dividing by one of the moduli or by the product of several moduli.

The following sequence of actions for division is given with reference to Figure 6.

Timing	Action(s) Taken
t_o	Clear accumulators, R-S flip-flops, counters, and X register.
t_1	Load operands into the operand accumulators.
t_2	Transfer Y to the memory address register (MAR). Convert X and Y to mixed radix notation.
t_3	Find E (Y).
t_4	Form the one's complement of E (Y) in the mod 127 accumulator.
t_5	Find E (X).
t_6	Add E (X) to the contents of the mod 127 accumulator. Transfer X to the X register.
t_7	Multiply - $2^{E(X)-E(Y)}$. Y, where the clear signal and usual load operation to the Y multiplier accumulator are inhibited and the load operation for the X multiplicand accumulator is altered by loading only the appropriate complement bits determined by the decoder.

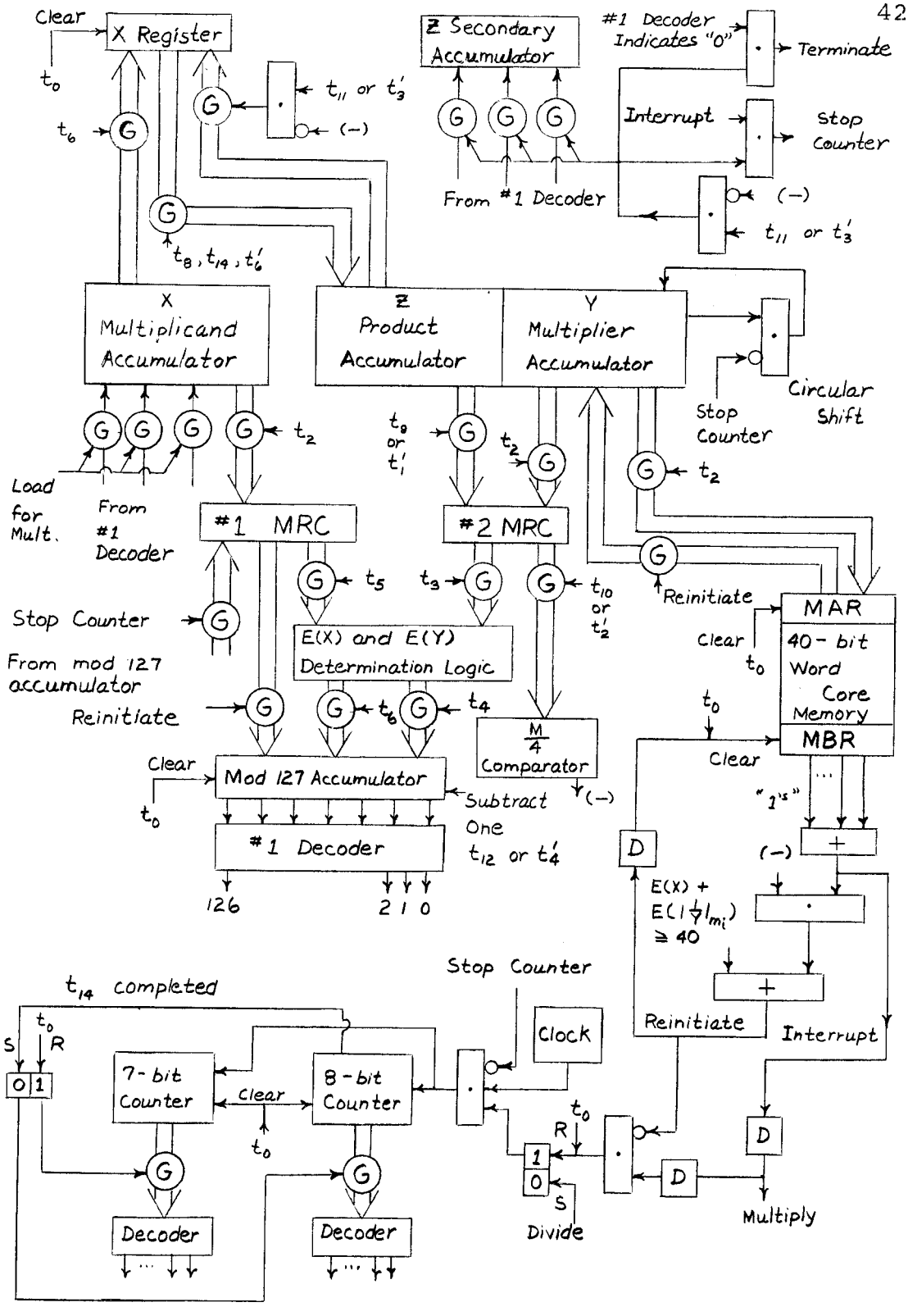


Figure 6 Division

t_8	Add X, located in the X register to the Z product accumulator.
t_9 or t_1^1	Convert $X - 2^{E(X)-E(Y)} \cdot Y$ to mixed radix notation.
t_{10} or t_2^1	Compare with $\frac{M}{4}$.
t_{11} or t_3^1	If the $\frac{M}{4}$ comparator indicates positive, transfer the contents of the Z product accumulator to the X register and add the appropriate bits determined by the decoder to the Z secondary accumulator.
t_{12} or t_4^1	Subtract one from the mod 127 accumulator.
t_{13} or t_5^1	Multiply $- 2^{E(X)-E(Y)-1} \cdot Y$ with same changes as t_7.
t_{14} or t_6^1	Add contents of the X register to the Z product accumulator. Observe that X is located in the X register if the $\frac{M}{4}$ comparator is negative. If the $\frac{M}{4}$ comparator is positive, $X - 2^{E(X)-E(Y)}$ is located in the X register.

Repeat t_1^1 through t_6^1 with revised values until division is terminated.

Depending on the memory cycle time, the multiplicative inverse of Y, $\left|\frac{1}{Y}\right|_{m_i}$, will appear (if it exists) in the memory buffer register (MBR) sometime during the timing sequence given. If $\left|\frac{1}{Y}\right|_{m_i}$ appears in the MBR, then the timing sequence is interrupted and a multiplication cycle is begun. The usual loading operation for multiplication is altered by loading $\left|\frac{1}{Y}\right|_{m_i}$, located in the MBR, into the Y multiplier accumulator. This assumes X is readily available in its usual location before loading

(possibly in an instruction register).

The method for interrupting the timing sequence depends on the type of system used to generate the timing signals required. If a clock, counter, and decoder arrangement is used, the timing sequence can easily be interrupted or terminated by inhibiting the input clock pulse to the counter. In order that the timing sequence may be continued later at the point where it is interrupted, the timing sequence is interrupted in the following manner:

(1) With the presence of $\left|\frac{1}{Y}\right|_{m_i}$ in the MBR (interrupt signal generated) and the action of t_{11} or t_3^1 completed, the clock input to the counter is inhibited. (stop counter signal generated).

(2) The stop counter signal not only inhibits the clock input to the counter, but transfers the contents of the mod 127 accumulator to the # 1 MRC mod 1024 register for temporary storage.

If multiplicative overflow exists in forming $Z^1 = X \cdot \left|\frac{1}{Y}\right|_{m_i}$, then the timing sequence must be re-initiated. The timing sequence can be reinitiated by detecting $E(X) + E\left(\left|\frac{1}{Y}\right|_{m_i}\right) \geq E(K) + 3 = 40$ OR if the $\frac{M}{4}$ comparator output signal AND the interrupt signal is present. The reinitiated signal transfers the contents of the # 1 MRC mod 1024 register to the mod 127

accumulator and transfers Y from the MAR to the Y multiplier accumulator. Observe that $\left|\frac{1}{Y}\right|_{m_i}$, unlike Y, is shifted out and lost. Consequently, the Y multiplier accumulator is cleared and ready to accept Y from the MAR. The reinitiate signal is slightly delayed before clearing the MBR. After clearing the MBR, the stop counter signal no longer exists. Thus, the timing sequence is resumed where it was interrupted.

Termination of division can occur with the presence of one of two conditions:

(1) If $\left|\frac{1}{Y}\right|_{m_i}$ exists and no reinitiate signal occurs, division is terminated. The quotient is located in the Z product accumulator.

(2) If $\left|\frac{1}{Y}\right|_{m_i}$ does not exist, division is terminated when the decoder decodes the contents of the mod 127 accumulator as zero and the actions of t_{11} or t_3^1 have been completed. The quotient is located in the Z secondary accumulator.

Although not shown, the change in gating for normal loading operations requires only simple logic. Normally, the product of any multiplication is immediately converted to mixed radix notation and compared with $\frac{M}{4}$. However, during division this operation is inhibited. Sign detection occurs only at the times indicated.

It is easily seen that division can be accomplished in the time required to obtain $\left|\frac{1}{Y}\right|_{m_i}$ from memory and

form the residue product of $X \cdot \left|\frac{1}{Y}\right|_{m_i}$. However, at least 88% of the time, the binary residues of Z need to be determined. This process may only involve a few iterations if E (X) - E (Y) and consequently $2^{E(Z)-1}$, the maximum binary residue of Z, is kept small. This, of course, depends on the specific problem and scaling involved.

III. INPUT-OUTPUT NUMBER CONVERSION

Although input-output conversion is not technically part of the arithmetic unit, it is considered beneficial to consider the problem of number conversion.

A deciding factor in the use of a residue arithmetic unit is the amount of hardware necessary for the unit to function properly in a particular environment. Thus, before a residue arithmetic unit can be utilized intelligently and economically some knowledge of input-output number conversions must be gained.

Input Conversion

Weighted Binary Code to Residue Notation

As previously shown, a positive number in the modified sign and complement residue notation is represented by numbers 0 - $(\frac{M}{4} - 1)$, inclusive, and a "zero" sign bit. Negative residue numbers are represented by numbers $(\frac{3}{4} M + 1) - (M - 1)$, inclusive, and a "one" sign bit. This corresponds to the equivalent decimal number range (for $m_1 = 1024$, $m_2 = 2047$, $m_3 = 1023$, $m_4 = 511$) of $-\frac{M}{4} =$ - 273,939,300,096 through $+ (\frac{M}{4} - 1) =$ 273,939,300,095.

Thus, before conversion is begun the binary coded number $x := x_{39}2^{39} + x_{38}2^{38} + x_{37}2^{37} + \cdots + x_1 2^1 + x_o 2^o$, where x_{40} is the sign bit, must be examined to determine

if the specified limits are exceeded. If one of the limits is exceeded the binary number must be scaled to bring the number within the specified limits before conversion can proceed.

$$\frac{M}{4} := \underbrace{00}_{\bar{A}}\underbrace{11111111}_{B}\ \underbrace{00}_{c}\ \underbrace{1}_{d}\ \underbrace{0000000}_{E}\ \underbrace{11}_{F}\ \underbrace{0}_{g}\ \underbrace{11111111}_{H}\ \underbrace{1}_{m}\ \underbrace{00000000}_{P}$$

in the binary code. The logic equation when the binary number x exceeds $-\frac{M}{4}$ or $+\frac{M}{4} - 1$ is as follows:

$$\text{Exceed} - \left(\frac{M}{4}\right) = \text{"1"}\left[A + BC + BdE + BdFg + BdFHmP\right]$$

$$\text{Exceed} + \left(\frac{M}{4}-1\right) = \text{"0"}\left[A + BC + BdE + BdFg + BdFHm\right]$$

The limits are exceeded when "Exceed $-\frac{M}{4}$" or "Exceed $+\left(\frac{M}{4}-1\right)$" occurs. Simplifying:

$$\text{Exceed limits} = A + B\left[C + d\ (E + F\ (g + H\ m))\right]$$

Using a maximum of five inputs to any AND or OR gate, the above equation can be realized with 11 AND and eight OR gates. If the binary coded number is negative and in complemented form, the number must be complemented before conversion.

Now that the binary coded number is ready for conversion, a few facts should be stated before proceeding with the actual conversion. Where $k \angle 9$ in 2^k, then $\left|2^k\right|_{2^9-1}$ is 2^k. For example: $\left|2^7\right|_{2^9-1} = \left|128\right|_{511} = 128$. Where $k \geq 9$ in 2^k, then $\left|2^k\right|_{2^9-1} = \left|2^{(k-9)}\right|$. For example: $\left|2^9\right|_{2^9-1} = \left|2^{9-9}\right|_{511} = 1$ and

$$\left|2^{12}\right|_{2^9-1} = \left|2^{12-9}\right|_{511} = \left|2^3\right|_{511} = 8.$$

In conversion of the binary number x, the binary bits will be added to the appropriate nine bit positions in the mod 511 accumulator, to the appropriate ten bit positions in the mod 1023 and mod 1024 accumulators, and to the appropriate 11 bit positions in the mod 2047 accumulators.

Binary bits x_o, x_9, x_{18}, x_{27}, and x_{36} are all added to the least significant bit position (2^o) of the mod 511 accumulator. This is done since the residue of these bits with respect to modulus 511 is one. Thus, to determine the least significant bit of the residue number with respect to modulus 511 requires three additions and three gating operations. First, x_o and x_9 are simultaneously gated to the least significant position and added. Next, x_{18} and x_{27} are simultaneously gated and added to the least significant bit. Lastly, x_{36} is gated and added to the least significant bit position. This process can easily be accomplished since the least significant stage of the accumulator has a carry-in input.

In a similar manner, appropriate bits are added to the eight other bit positions of the mod 511 accumulator.

A table for the conversion of binary x to y mod 511 = $\left|x\right|_{511}$ can now be written.

Mod 511 Binary Bit Positions	Bits of Binary Number x Added
2^0	$(x_0 + x_9) + (x_{18} + x_{27}) + x_{36}$
2^1	$(x_1 + x_{10}) + x_{19} + x_{28} + x_{37}$
2^2	$(x_2 + x_{11}) + x_{20} + x_{29} + x_{38}$
2^3	$(x_3 + x_{12}) + x_{21} + x_{30} + x_{39}$
2^4	$(x_4 + x_{13}) + x_{22} + x_{31}$
2^5	$(x_5 + x_{14}) + x_{23} + x_{32}$
2^6	$(x_6 + x_{15}) + x_{24} + x_{33}$
2^7	$(x_7 + x_{16}) + x_{25} + x_{34}$
2^8	$(x_8 + x_{17}) + x_{26} + x_{35}$

Note that the binary bits within the parentheses are added simultaneously, in parallel, to the appropriate stages of the accumulator. The other bits are added consecutively, in parallel, to the appropriate stages. Obviously, the type of adder or accumulator will determine how many bits can be added simultaneously, in parallel, to each stage of the accumulator.

Other conversion tables are shown.

Mod 2047 Binary Bit Positions	Bits of Binary Number x Added
2^0	$(x_0 + x_{11}) + x_{22} + x_{33}$
2^1	$(x_1 + x_{12}) + x_{23} + x_{34}$
2^2	$(x_2 + x_{13}) + x_{24} + x_{35}$
2^3	$(x_3 + x_{14}) + x_{25} + x_{36}$
2^4	$(x_4 + x_{15}) + x_{26} + x_{37}$
2^5	$(x_5 + x_{16}) + x_{27} + x_{38}$
2^6	$(x_6 + x_{17}) + x_{28} + x_{39}$
2^7	$(x_7 + x_{18}) + x_{29}$
2^8	$(x_8 + x_{19}) + x_{30}$
2^9	$(x_9 + x_{20}) + x_{31}$
2^{10}	$(x_{10} + x_{21}) + x_{32}$

Mod 1023 Binary Bit Positions	Bits of Binary Number x Added
2^0	$(x_0 + x_{10}) + x_{20} + x_{30}$
2^1	$(x_1 + x_{11}) + x_{21} + x_{31}$
2^2	$(x_2 + x_{12}) + x_{22} + x_{32}$
2^3	$(x_3 + x_{13}) + x_{23} + x_{33}$
2^4	$(x_4 + x_{14}) + x_{24} + x_{34}$
2^5	$(x_5 + x_{15}) + x_{25} + x_{35}$
2^6	$(x_6 + x_{16}) + x_{26} + x_{36}$
2^7	$(x_7 + x_{17}) + x_{27} + x_{37}$
2^8	$(x_8 + x_{18}) + x_{28} + x_{38}$
2^9	$(x_9 + x_{19}) + x_{29} + x_{39}$

Since $|2^k|_{1024} = 0$ for $k \geq 10$, only the x_o through x_9 bits need to be gated and added to the corresponding ten bit positions of the mod 1024 accumulator.

Inspection of the tables reveals the time for conversion is the time necessary to complete four gating operations and four additions in a mod 511 accumulator. This conversion time could be shortened if the accumulators could add more bits simultaneously or if more accumulators were used.

Conversion from Decimal to Residue Numbers

For some environments in which a residue arithmetic unit is utilized a decimal to residue number conversion capability is desirable.

The decimal number, Z, can be written in single-base notation as $Z := d_n b^n + d_{n-1} b^{n-1} + \cdots + d_1 b^1 + d_o$ where b is the base ten for a decimal number. The residue of Z mod m_i can be written as $|Z|_{m_i} = |d_n b^n + d_n b^{n-1} + \cdots d_1 b^1 + d_o|_{m_i} = \left| |d_n b^n|_{m_i} + |d_{n-1} b^{n-1}|_{m_i} + \cdots + |d_1 b^1|_{m_i} + |d_o|_{m_i}|_{m_i} \right.$.

For any term such as $d_j b^j$, b^j is constant for every decimal number subject to conversion. Of course, d_j varies as the decimal number changes. Consequently, $|d_j b^j|_{m_i}$ is a term which is a constant multiplied by a variable. Therefore, the digit d_j of any decimal number

can simply be relabeled to obtain the residue with respect to m_i. Each relabeled decimal digit needs to be added in a mod m_i adder to obtain the residue with respect to m_i for the given decimal number Z. Ordinarily, if Z is n decimal digits in length, n mod m_i residue additions are required for the conversion process. If all digits of Z are available simultaneously, parallel additions in separate mod m accumulators can speed-up the conversion. If the digits of Z are available in one of several serial forms, counters that operate mod m_i may be used to count to the correct residue number.

It should be observed that the general statements of this section have been applied in the preceding section to a special case of single base notation where the base, b, is two.

Output Conversion

Output conversion is necessary for two reasons. First, to convert computation results to a recognizable form. Second, to convert to another number system suitable for use in other portions of a computing system.

One method of output conversion is outlined by R. D. Merrill, Jr. (5). It consists of a mixed radix conversion as previously discussed and a mixed radix to weighted binary code conversion. The latter conversion

is accomplished by utilizing an algorithm based on the particular moduli and word length used.

To develop the algorithm it is necessary to begin by observing $N := d_4m_3m_2m_1 + d_3m_2m_1 + d_2m_1 + d_1$ in mixed radix notation more closely. Let: $m_1 = 1024$, $m_2 = 2047$, $m_3 = 1023$, and $m_4 = 511$. Note that each bit of the binary coded mixed radix digit d_4 is weighted by $m_3 \cdot m_2 \cdot m_1$. Since $m_3 \cdot m_2 \cdot m_1 = (2^{10}-1)\ (2^{11}-1) \cdot 2^{10} = (2^{10}-2^0)\ (2^{11}-2^0) \cdot 2^{10} = (2^{21}-2^{11}-2^{10}+2^0) \cdot 2^{10}$, each of the nine binary bits of d_4 needs to be gated and added to the appropriate bit positions (determined by the weight for d_4) of a 40-bit accumulator.

Similarly, the weights for d_3, d_2, and d_1 are $(2^{11}-2^0) \cdot 2^{10}$, 2^{10} and 2^0 respectively.

Now an equation can be written for N. $N = \left[(d_4 2^{21} + d_3 2^{11} + d_2) - ((d_4 2^{11} + d_3) + d_4 2^{10}) + d_4\right] \cdot 2^{10} + d_1$.

By careful observation of the binary bit lengths of each mixed radix digit some of the terms of the above equation are not really added together. Instead they are gated to the appropriate stages of the 40-bit accumulator. As an example, the terms $d_4 2^{21} + d_3 2^{11} + d_2$ are merely gated to the appropriate stages from zero to the 30th. stage. The output conversion equation for N is readily seen to consist of two additions, one

subtraction, and a shift left ten binary places.

If $N \geq \frac{3}{4} M + 1$ then the correct weighted binary representation of N can be obtained by subtracting N from M.

IV. CONCLUSIONS

Residue addition (subtraction) and multiplication without overflow detection can be accomplished in approximately one-fourth and one-sixteenth the time required for conventional addition (subtraction) and multiplication respectively. If additive overflow is checked after each addition (subtraction), the time for residue addition (subtraction) is increased by approximately 900%. This is due to the nine additions and subtractions required for a residue to mixed radix conversion. However, by checking for additive overflow less frequently, the advantage of using residue addition (subtraction) is realized. Since multiplicative overflow can be detected in approximately the time required for ten residue additions (subtractions), residue multiplication can be accomplished in approximately $(\frac{10}{40} \cdot \frac{12.5}{40}) = \frac{5}{64}$ of the time required for conventional multiplication.

Twenty 41-bit additions (subtractions) are required to form a 21-bit quotient, including the sign bit, using a conventional non-restoring method of addition. If $\left|\frac{1}{Y}\right|_{m_i}$ exists and X is evenly divisible by Y, residue division can be accomplished in approximately 2.75 41-bit additions and the time necessary to obtain $\left|\frac{1}{Y}\right|_{m_i}$ from memory. If $\left|\frac{1}{Y}\right|_{m_i}$ does not exist or X is not evenly divisible by Y, then the time for residue division is

considerably greater. In this case the method of determining the binary residues of the quotient is utilized. This method can be accomplished in the same time as conventional non-restoring division if the maximum binary residue is less than 2^3.

Input-output conversion does not appear to be a significant problem when using residue number systems. However, the mixed radix conversion involved in the residue to decimal conversion can add considerable expense and time for output conversion.

Perhaps the biggest improvement over any other residue systems discussed in the literature is a method of detecting multiplicative overflow. It has been hinted in the literature (4,6) that a super-modulus could be used for this detection. If a super-modulus is made large enough, all cases of multiplicative overflow could be detected. The super-modulus required for this would be impractically large. To detect only two places of overflow a modulus of five for the arithmetic unit discussed would work quite well. This super-modulus complicates mixed radix conversion, thereby making it more practical to alter the number range for positive and negative numbers.

Certainly magnitude determination of residue numbers is one of the greatest handicaps when using

radix notation. Improvement in this area would enable more moduli to be used for a given number range, thereby making greater use of the inherent advantage of residue notation.

BIBLIOGRAPHY

1. Flores, Ivan. The logic of computer arithmetic. Englewood Cliffs, N.J., Prentice-Hall, 1963. 493 p.

2. Garner, H. L. The residue number system. Institute of Radio Engineers Transactions on Electronic Computers EC-8:140-147. 1959.

3. Keir, Y. A., P. W. Cheney and M. Tannenbaum. Division and overflow detection in residue number systems. Institute of Radio Engineers Transactions on Electronic Computers EC-11: 501-507. 1962.

4. Lindamood, George E. and George Shipiro. Magnitude comparison and overflow detection in modular arithmetic computers. Society for Industrial and Applied Mathematics Review 5:342-350. 1963.

5. Merrill, Roy D., Jr. Improving digital computer performance using residue number theory. Institute of Electrical and Electronics Engineers Transactions on Electronic Computers EC-13:93-101. 1964.

6. Szabo, Nicholas. Sign detection in nonredundant residue systems. Institute of Radio Engineers Transactions on Electronic Computers EC-11:494-500. 1962.

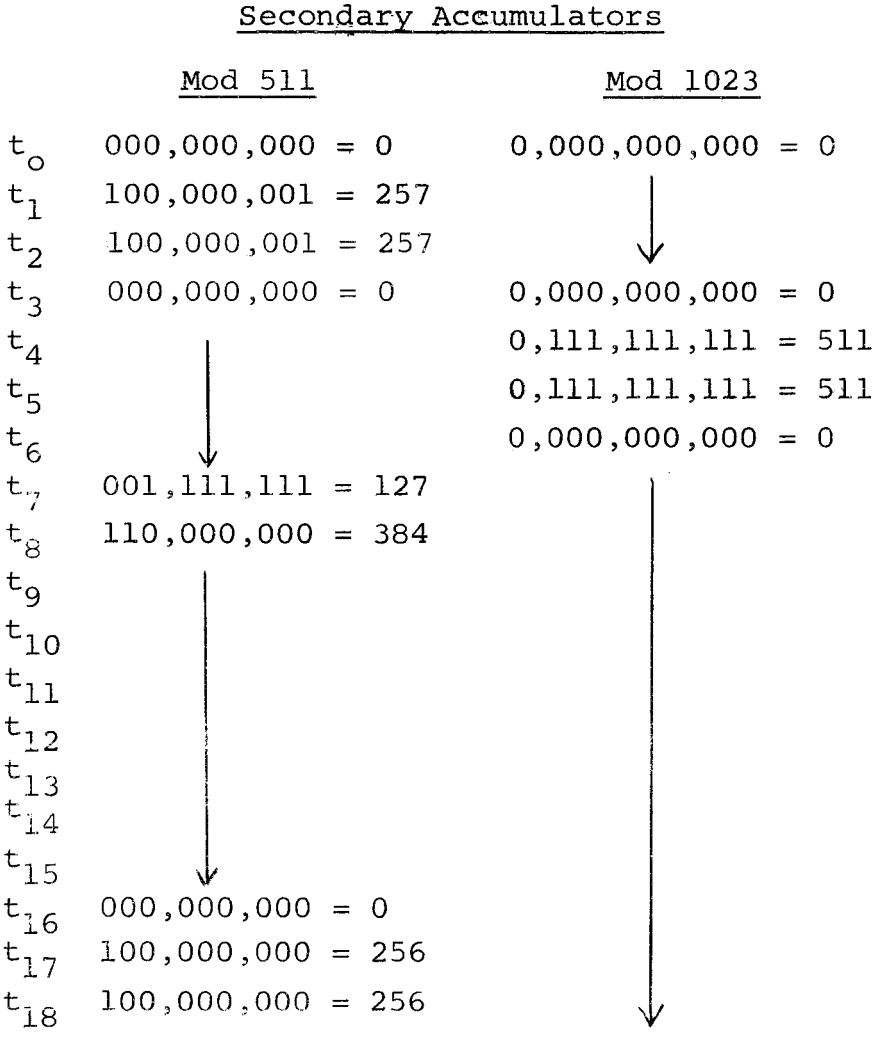
APPENDIX

APPENDIX I

Conversion of $\frac{M}{4}$ from Residue to Mixed Radix Notation

The contents of the accumulators and register shown in Figure 3 are given during the conversion of $\frac{M}{4}$ to mixed radix notation.

The numbers shown refer to the contents in each accumulator or register at the completion of action(s) for each timing signal.



Primary Accumulators

	Mod 511	Mod 1023
t_o	000,000,000 = 0	0,000,000,000 = 0
t_1	000,000,000 = 0	0,000,000,000 = 0
t_2	011,111,110 = 254	0,011,111,111 = 255
t_3	001,111,111 = 127	0,011,111,111 = 255
t_4	001,111,111 = 127	0,011,111,111 = 255
t_5	001,111,111 = 127	1,011,111,111 = 767
t_6	001,111,111 = 127	↓
t_7	001,111,111 = 127	
t_8	111,111,100 = 508	
t_9	001,111,100 = 124	
t_{10}	111,110,000 = 496	
t_{11}	001,110,000 = 112	
t_{12}	111,000,000 = 448	
t_{13}	001,000,000 = 64	
t_{14}	100,000,000 = 256	
t_{15}	101,111,111 = 383	
t_{16}	↓	
t_{17}		
t_{18}	001,111,111 = 127	

	Mod 2047	Mod 1024 Register
t_o	00,000,000,000 = 0	0,000,000,000 = 0
t_1	00,000,000,000 = 0	1,100,000,000 = 768
t_2	10,011,111,111 = 1279	↓
t_3	00,111,111,111 = 511	
t_4	↓	
.		
.		
.		
.		
t_{18}	↓	↓