COO-1018-1204

Report No. 397

A STUDY OF METHODS FOR SELECTION OF QUOTIENT DIGITS DURING DIGITAL DIVISION

by

Daniel E. Atkins

June 1970

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

COO-1018-1204

Report No. 397

A STUDY OF METHODS FOR SELECTION OF QUOTIENT DIGITS DURING DIGITAL DIVISION*

by

Daniel E. Atkins

June 1970

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

*This work was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science, June 1970, and was supported in part by the National Science Foundation under Grant Nos. US NSF GJ 812 and US NSF GJ 813, and in part by the U.S. Atomic Energy Commission under Contract USAEC AT(11-1-1018).

[illegible]

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

ACKNOWLEDGMENT

The author gratefully acknowledges the continued support of the Department of Computer Science during the past five years and particularly thanks the following professors, colleagues and employees of the department.

First to his thesis adviser, Professor James E. Robertson, whose guidance, encouragement, and friendship are highly valued. Second, to Professor Bruce H. McCormick, whose never failing loyalty, support, and enthusiasm are equally valued.

The author's colleague, V. G. Tareski is the author of the very efficient prime implicant generation algorithm so essential to this work and mentioned in Section 4. He was also a source of encouragement and enlightening discussions. C. R. Baugh, T. K. Liu, and T. Ibaraki developed the program used to solve the massive covering problems encountered in the course of minimization. B. G. DeLugish has assisted in offering valuable discussions and in the arduous task of proofreading.

The final typing is the fine work of Mary Ann Davis and Betty Gunsalus. The excellent drawings were done by Stanislav Zundo and the equally excellent offset printing is the work of Dennis Reed.

And finally, acknowledgment is due Miss Peppermint Patty whose incisive comments, presented below, have been a source of comfort in times of confusion.

PEANUTS
NUMBERS ARE BEAUTIFUL..
I LIKE TWOS THE BEST...THEY'RE SORT OF GENTLE..THREES AND FIVES ARE MEAN, BUT A FOUR IS ALWAYS PLEASANT..I LIKE SEVENS AND EIGHTS, TOO, BUT NINES ALWAYS SCARE ME...TENS ARE GREAT...
HAVE YOU DONE THOSE DIVISION PROBLEMS FOR TOMORROW?
NOTHING SPOILS NUMBERS FASTER THAN A LOT OF ARITHMETIC!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

A STUDY OF METHODS FOR SELECTION OF QUOTIENT DIGITS DURING DIGITAL DIVISION

Daniel Ewell Atkins, III, Ph.D.
Department of Computer Science
University of Illinois, 1970

This study concerns a class of non-restoring division schemes in which redundancy is introduced into the representation of the quotient thereby permitting quotient digits to be selected from highly truncated versions of the divisior and partial remainders. The mechanism for selection of quotient digits is a limited precision model of the full precision division which it controls by the generation of simple microprogram instructions. A major advantage of this approach to division is a high degree of congruity with commonly used multiplication structures, including those making use of limited propagation adder-subtracters, for example, carry-save adders.

A cost versus performance analysis for a large class of quotient selection mechanisms (model divisions) is developed. The class is defined in terms of a block diagram and a set of ten design parameters. By varying the structure of the sub-blocks and the values of the parameters, the model division scheme ranges from that of forming quotient digits by multiplying the dividend by the inverse of the divisor, to that of a direct table look-up of the quotient digit. So called hybrid structures exist between these two cases. Algorithms are described which synthesize near minimal cost realizations of the most complicated sub-blocks: a combinatorial logic network to produce appropriate estimates of the reciprocal of the divisor, and a combinatorial logic network to generate a quotient digit directly as a function of the bits in estimate of the divisor and partial remainder. Formulas are given for the cost of the remaining sub-blocks. For a given type structure the primary determinant of performance is the radix of the model division, $r = 2^k$, where k is the number of bits of quotient produced per access to the model division.

A FORTRAN implementation of the synthesis routines is used to obtain the near minimal cost for several different structures and sets of design parameter values. The numerical results, together with the insight gained in obtaining them, are applied to hypothesize a formula for minimal cost. The analysis includes a multi-variable expression which relates cost to the radix of the model division, r, the degree of redundancy in the quotient representation, and the magnitude and direction of the maximum truncation error in the divisor and partial remainder estimates. The cost formulas, together with easily derived performance formulas, are used to tabulate expected cost and performance for a variety of structures. It is found that for most schemes the cost varies exponentially with performance and consequently, that many of the higher radix schemes are not practicable. A radix 4, direct table look-up, however, can be built with about ten, 10-input gates, and assuming 10 ns. logic, could produce 60 bits of quotient in about 4 μs. The study is concluded with suggestions for further investigation.

1. INTRODUCTION

1.1 Background

Since division is the mathematical inverse of multiplication, one might hope that the cost of implementing both a multiplication and division operation would not be much different than the cost of implementing multiplication alone. Furthermore, for a given operand length, one might expect the executions times for the operations to be about the same. In actual practice this hope has not been realized, largely due to the fact that division, unlike multiplication, is inherently a trial-and-error process.

In multiplication, a product is accumulated by the successive addition of multiples of the multiplicand to a partial product. The selection of which multiple to add is dependent upon a digit, radix r, of the multiplier --a quantity which is known apriori.

Now consider a recursive relationship for a class of division techniques based upon subtraction. This relationship is defined by

$$p_{j+1} = rp_j - q_{j+1}d, \quad j = 0,1,\ldots,m-1 \tag{1.1}$$

in which

p_0 is the dividend,

p_j is the partial remainder used in the j^{th} recursion,

p_m is the remainder,

j is the recursion index,

q_j is the j^{th} quotient digit,

d is the divisor, and

r is the radix.

In forming the partial remainder, p_{j+1}, a multiple of the divisor is subtracted from the previous partial remainder shifted left by one digital position. The selection of which multiple to subtract is dependent upon a digit of the quotient; but it is precisely this quotient digit that we must compute. It is not known apriori. As it stands this relationship for division does not adequately specify how q_{j+1} is selected until we add a restriction such as $|p_{j+1}| < |d|$. The important point here is that division not only requires an addition or subtraction as in multiplication, but also the selection of a quotient digit such that the value of the contents of the accumulator after the subtraction is within a specified range. If it is not within this range then some correction is required.

Several effective techniques have been developed for accelerating the execution of multiplication. Foremost among them are the following:

1. Use of adders or subtracters which postpone carry or borrow propagation until a terminal step.
2. The use of a higher radix (greater than 2) so that several bits of the multiplier are retired in one iteration.
3. The introduction of redundancy* into the multiplier by multiplier recoding.

The success of such techniques in multiplication raises the question of their applicability to division. A significant contribution to the answer was made with the discovery of SRT division.

*Redundancy or redundant representation refers to a number representation in which each radix r digit may assume more than r different values.

In the middle 1950's D. Sweeney of IBM, J. E. Robertson of the University of Illinois [1]*, and T. D. Tocher [2] of Imperial College, London, independently discovered a binary division technique especially suited for implementation in an electronic digital computer. SRT division was named by C. V. Freiman of IBM in a paper discussing its statistical properties [3] although an example of the technique may actually have been presented by Nadler [4] in a 1956 paper describing a computer designed and built by the Institute of Mathematical Machines of the Czechoslovak Academy of Science under the direction of Dr. Antonin Svoboda. Whether or not the Nadler work is equivalent to SRT is obscured by the fact that it is discussed in conjunction with a stored-carry adder and accumulator.

The basis of SRT division is the discovery that introducing redundancy into the representation of the quotient yields more freedom in the selection of a quotient digit at each step of the recursion. In SRT division this freedom is used to increase the probability of a zero quotient digit, for which the next partial remainder is produced merely by a shift rather than by a subtraction followed by a shift. This flexibility is in contrast to conventional restoring or non-restoring division which require a full-precision subtraction for each quotient bit generated. Even though we are considering a binary number system, digit values for SRT division are 1, 0, $\overline{1}$ (the overbar denotes negation, i.e. -1), and thus we have redundancy.

In 1965, Robertson [5] extended the concepts inherent in SRT division to higher radix division schemes. The fundamental tenets of the method remain, namely, that by introducing redundancy into the representation

*Numbers in brackets refer to entries under References.

of the quotient, the selection of a quotient digit at each step of the recursion need not be precise. For the higher radix cases, a larger set of quotient digits is necessary and thus the probability of a zero quotient digit is reduced to the extent that adder bypass no longer yields significant speed improvement. However, the redundancy may still be put to advantage; it permits the selection of a quotient digit based only upon high-order digits of the divisor and high-order digits of the shifted partial remainder.

In reference [5], Robertson introduces the notion of a quotient selection mechanism with inputs consisting of estimates of the divisor and shifted partial remainder. He notes that the mechanism for selection of quotient digits may be thought of as a limited precision model of the full precision division. The procedures in the model need not be the same as the procedure of the full precision scheme which it controls. The model division generates simple microprogram instructions to the full-precision unit. His paper also presents an indirect, relative measure of the cost of selection of quotient digits.

The authors Master's Thesis in 1967 is based largely upon Robertson's work as described in references [1] and [5]. The complete thesis, including an example of a actual implementation of a model division scheme, is available in report form [6]; the more theoretical aspects of the work are available in journal form [7]. Implementation is also discussed in a more recent report in conjunction with the development of the arithmetic units of the Illiac III Computer [8], [9].

The author's paper [7] is largely tutorial. It presents a detailed review of Robertson's proof of the validity of the class of division techniques to which the model division approach is applicable. The proof will not be repeated in the present work. The paper also describes a graphical representation, the so-called P-D plot, suggested by C. V. Freiman [5], which is useful in describing the division procedure, and then develops expressions for the maximum number of bits of the divisor and partial remainder which must be inspected in order to determine a correct quotient digit for a given radix, a given lower limit on the divisor, and a given amount of redundancy* in the representation of the quotient. These expressions, which provide a worst-case measure of costs, also account for redundancy in the representation of the partial remainder such as produced by a member of the family of carry-save adders or borrow-save subtracters [10], [11], [12].

We are now in a position to consider the design of division schemes which are highly compatible with multiplication structures. The model division determines which multiples of the divisor are to be combined with the partial remainder. In this respect it is analogous to the multiplier recoder and may be thought of as a quotient recoder.∓ Multiplier recoding logic is usually entirely combinatorial and grows in complexity only linearly with the radix. The model division is complicated by the fact that the quotient digit is a function of both the divisor and the partial remainder and the fact that the partial remainder, unlike the divisor or multiplier, is not constant throughout a given operation. An analysis of the growth of the complexity of a model division with increase of radix is one aspect of this thesis work.

But despite these complications, the strong analogy between multiplier recoding and the concept of the model division leads to a division

*A measure of redundancy will be defined later in this paper.

∓Robertson has made a formal correspondence between multiplier recodings and quotient recodings produced by SRT division. See Ref. [13].

structure which is potentially highly compatible with a given multiplication scheme. The difference in the execution time between the iterative portion of multiplication and division is essentially the difference between the total time required to recode the multiplier and that to recode the quotient. The bulk of the logic accounting for the difference in cost of a multiplier and the cost of a multiplier and divider may then be associated with the cost of implementing the model division.

1.2 Present Work

With this background in mind, we now turn to an introduction to the present work. Section 2 begins by defining a class of full-precision multiplication-division structures. We then define a rather general block structure of a quotient selection mechanism suitable for use as a model division. The parameters of the model include the radix, the magnitude of the largest quotient digit, the range of the divisor, and the truncation error in the estimates of the divisor and partial remainder.

The flexibility of the model division approach and the generality of the model proposed in Section 2 offer a large number of design possibilities. A major goal of this work is to investigate the cost versus performance of various designs and attempt to extract analytic results. Such an attempt requires the definition of a measure of cost and performance. A useful cost measurement should, in some sense, be minimal, and therefore we must consider minimization criterion and a minimization algorithm. These topics are discussed in Section 3.

The first approach taken to determining cost and performance of various quotient selectors is that of computer-aided generation of a specific design followed by analysis. In Section 4 algorithms are described which, when supplied with parameter values, will generate logic definitions of the sub-blocks of the model. Most of the logic will be defined in a minimal sum-of-products form which could serve as input to a logic design program customized for a given class of logic.

To this point we will have developed a mechanism for generating and comparing various designs for a model division. The approach has been one of computer-aided design followed by computer-aided minimization. The results from the computer work are tabulated in Section 5. Although the design and minimization programs are quite efficient, the large number of design possibilities together with the large number of terms in the logic equations for the higher radix models strongly discourages an exhaustive analysis. An additional result described in Section 5 has been insight which led to development of analytic expressions for the cost of a structure.

Section 6 is a tabulation of estimates of cost and performance based upon the equations and computer generated results described in Section 5. The final selection is a summary and some conclusions as to the relative merits of various members of the family of model division schemes considered. The section includes a list of suggestions for further investigation.

2. DEFINITION OF THE DIVISION PROCEDURE

2.1 <u>Formal Definition of the Full Precision Division</u>

The members of the class of division algorithms which may be employed to perform the full-precision division are those defined by the recursive relationship (1.1) and the list of restrictions given below. The recursive relationship is repeated here for convenience.

$$p_{j+1} = rp_j - q_{j+1}\, d, \quad j = 0,1,\ldots,m-1 \tag{1.1}$$

in which

p_o is the dividend,

p_j is the partial remainder used in the jth recursion,

p_m is the remainder,

j is the recursion index,

q_j is the jth quotient digit (radix - r),

d is the divisor, and

r is the radix.

The quantity rp_j is referred to as the shifted partial remainder. Restrictions which apply are as follows:

1. Allowable quotient digits are

   $-n, -n+1, -n+2, \ldots, 0,1,2,\ldots, n$ where

   n is an integer such that $n \geq (r - 1)/2$. (2.1)

2. The dividend, p_o, must be in the range defined by

$$|p_o| \leq \rho \; |d| \tag{2.2}$$

where $\rho = n/(r-1)$. (2.3)

3. The divisor must be within a given range, i.e. the quantities a and b must be defined such that

$$a \leq |d| \leq b. \tag{2.4}$$

4. Every quotient digit, q_{j+1} for j from 0 through m-1, must be chosen such that p_{j+1} as defined by (1.1) is within the range

$$|p_{j+1}| \leq \rho \; |d|. \tag{2.5}$$

The derivation of these restrictions is given in [6] and [7]. Note that the forms of rp and d have not been limited to non-redundant representations. They may be in forms such as produced by carry-save adders or borrow-save subtracters.

2.2 Graphical Representation of the Division Procedure

This division procedure may be defined graphically with a construction suggested by C. V. Freiman [5]. The basis for the construction is the recursive relationship (1.1) together with the range restriction (2.5). The figure is essentially a plot of partial remainder versus divisor values and is thus designated a P-D plot.

Solving the recursive relationship for rp_j yields $rp_j = p_{j+1}' + q_{j+1}\,d$. For a fixed quotient digit, the upper limit of rp_j as a function of the divisor, d, occurs when p_{j+1} is maximum, i.e., when $p_{j+1} = \rho d$ and thus

$$rp_{j\ max} = (\rho + q_{j+1})d. \tag{2.6}$$

Likewise the lower limit is defined by

$$rp_{j\ min} = (-\rho + q_{j+1})d. \tag{2.7}$$

These linear functions of d may be plotted as a family of curves with q_{j+1} as a parameter ranging from -n through +n in steps of 1. The area between $rp_{j\ max}$ and $rp_{j\ min}$ for a given $q_{j+1} = i$ will be denoted the "q(i) region."

For given r, n, a, and b, the division procedure is specified by the corresponding P-D plot. A given value of d and rp_j will specify a point in a q(i) area. The quotient digit q_{j+1} is therefore i and is used in forming p_{j+1}.

Figure 1 is an example of a P-D plot with r = 4, n = 2, a = 1/2 and b = 1. The equations for the lines denoted 2', 2, etc. are defined in Table 1. Note that as a consequence of the redundancy introduced into the representation of the quotient there is overlap between adjacent quotient regions. Some pairs (d, rp_j) will specify a point for which either $q_{j+1} = i$ or $q_{j+1} = i - 1$ is a valid choice. It is this overlap which permits quotient digit selection to be made on the basis of <u>estimates</u> of the full precision divisor and shifted partial remainder.

2.3 Formal Definition of the Quotient Selection Procedure

The quotient selection mechanism may be defined as a device that

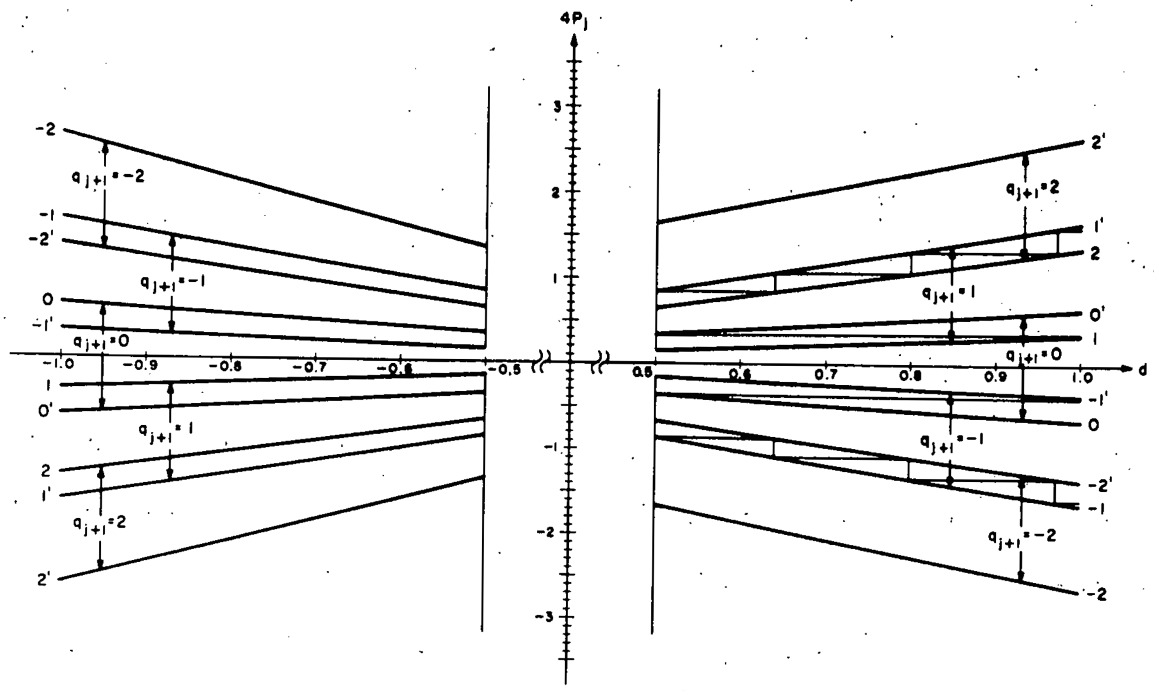


Figure 1. P-D Plot with r=4, n=2

$$rp_j = \pm \frac{n}{r-1} d + q_{j+1} d$$

Designation in Figure 1	q_{j+1}	p_{j+1}	Equation $rp_j =$
2'	2	2/3 d	8/3 d
2	2	-2/3 d	4/3 d
1'	1	2/3 d	5/3 d
1	1	-2/3 d	1/3 d
0'	0	2/3 d	2/3 d
0	0	-2/3 d	-2/3 d
$\overline{1}'$	$\overline{1}$	2/3 d	-1/3 d
$\overline{1}$	1	-2/3 d	-1/3 d
$\overline{2}'$	$\overline{2}$	2/3 d	-4/3 d
$\overline{2}$	$\overline{2}$	-2/3 d	-8/3 d

Table 1. Equations Defining the Regions of Figure 1.

when given estimates of the divisor and shifted partial remainder of "sufficient" precision, will produce a quotient digit, i, such that restriction (2.5) is satisfied. The definition of sufficient precision is given in the following.

With a, b, n, and r given, the P-D plot is specified. Let D be the set of all divisor values for a given operand length and range specified by (a, b). Let P be the set of all values of allowable shifted partial remainders. The area of the P-D plot is the Cartesian product of P and D, i.e. the area is the set

$$P \times D = \{(rp,d) \mid rp \in P \text{ and } d \in D\}. \qquad (2.8)$$

Every element of P x D is contained in one or more q(i) regions; thus each element implies a set $I = \{i \mid (rp, d) \text{ is within the } q(i) \text{ region}\}$. In Figure 1, every pair (d, rp) will be contained in either one or two q(i) regions. This will be the case for all examples discussed in this study, however, for $\rho = n/(r - 1)$ greater than 1, a given (d, rp) would be contained within two or more q(i) regions.

The inputs to the quotient selection mechanisms are estimates of the divisor and shifted partial remainder. Let $\hat{d}$ and $\hat{rp}$ denote these estimates, respectively, and let $Q(\hat{rp}, \hat{d})$ be the output of the quotient selection mechanism (a quotient digit) for given estimates $(\hat{rp}, \hat{d})$.

The set of $\hat{rp}$ and $\hat{d}$ values are of sufficient precision and the quotient selection procedure is correct if for every pair $(rp, d) \in P \times D$, there exists an ordered pair $(\hat{rp}, \hat{d})$ such that $Q(\hat{rp}, \hat{d}) = i$, where i belongs to the set I implied by (rp, d).

In actual practice, $\hat{d}$ and $\hat{rp}$ are formed by uniformly truncating, or truncating and rounding d and rp, respectively. Assume that a binary representation of d is truncated between position δ and $\delta + 1$ to the right of the binary point, and that a binary representation of rp is truncated between position ε and $\varepsilon + 1$ to the right of the binary point. Let,

$$\Delta d = 2^{-\delta}, \text{ and} \tag{2.9}$$

$$\Delta rp = 2^{-\varepsilon}. \tag{2.10}$$

The set of $\hat{d}$-values are therefore integer multiples of Δd and the set of $\hat{rp}$ values are integer multiples of Δrp. A given value of $\hat{d}$ is representative of the range of full precision divisor values given by

$$\hat{d} - \alpha \leq d \leq \hat{d} + \beta, \tag{2.11}$$

where

$$\alpha = \alpha' \; \Delta d \tag{2.12}$$

$$\beta = \beta' \; \Delta d. \tag{2.13}$$

Similarly, $\hat{rp}$ is representative of the range of full precision shifted partial remainders in the range

$$\hat{rp} - \lambda \leq rp \leq \hat{rp} + \gamma, \tag{2.14}$$

where

$$\lambda = \lambda' \; \Delta rp, \text{ and} \tag{2.15}$$

$$\gamma = \gamma' \Delta rp. \tag{2.16}$$

The quantities α', β', λ', and γ' are in the range 0 to 2 and depend upon the truncation procedure and the form of representation of d and rp.

2.4 Physical Model of the Quotient Selection Mechanism

We now turn to the question of the physical realization of a quotient selection mechanism; the device which performs the operation $\hat{rp}/\hat{d}$ to produce the quotient, i, such that i belongs to the set of quotient digits, I, implied by (rp, d). Since the operation time of division relative to multiplication is limited by the model division, the requirements for a high performance arithmetic processor would demand the design of a high-speed model division. One way to achieve this would be to use a higher-speed class of logic in building the model division than in building the remainder of the arithmetic processor, but in this work we are assuming one given class of logic and are constraining the design problem such that speed advantages must be gained by organization.

Any valid division technique is a candidate for a model division. One aspect of this study was a survey of known division techniques suitable for implementation in a digital computer. References [14] through [32] are some of the works consulted. In evaluating possible candidates we should keep in mind the advantages of dealing with relatively short operands coupled with the potential requirement for low operating times.

Digital division schemes may be classified as additive, multiplicative, tabular or some combination of the three. Additive techniques are those such as restoring and non-restoring division in which addition and subtraction are the fundamental operations; the divisor remains invariant. Multiplicative schemes are those in which both the dividend and divisor are multiplied by factors in such a manner that the modified divisor converges to 1 and thus the modified dividend converges to the quotient. Tabular techniques are those based upon a combinational network: the quotient digit is produced by table-look-up. Note that neither of the two later techniques produces a remainder but that a remainder is not needed for a model division.

We have eliminated analog schemes and threshold logic from consideration in this study. We have also ruled out logarithmic techniques since, although the division is transformed to a subtraction, the equipment-time ratio suffers due to necessity for forming logs and antilogs.

We now propose a generalized structure into which will fit multiplicative and tabular techniques. These schemes appear to have a potential for higher operating speeds than the additive techniques. Since it is an additive (non-restoring) scheme which is controlled with the model division it seems intuitively justifiable to consider a higher performance class for the model. Emphasis on hardwired table look-up techniques is also justified by trends of technology towards LSI.

Figure 2 is the generalized structure. The parameters and blocks are as follows:

Divisor Estimate Formation - This block accepts a full precision divisor, d, in the range $a \leq d < b$, and from it produces an estimate of the divisor, $\hat{d}$, with maximum negative uncertainty, α, and maximum positive uncertainty, β. This box may also incorporate provisions for changing the form of representation of $\hat{d}$ from that of d. For example, if the model division structure accepts only positive quantities, but d is in both negative and positive range, this box could convert $\hat{d}$ to a sign and magnitude form. The magnitude would serve as input to the model. The sign would be used together with the sign of the partial remainder in determining the sign of the quotient digit. This block is part of the interface between the full precision structure and the model division.

In addition to this interfacing function, the divisor estimate formation box also serves as a selector. Note that the output of Multiplier 2

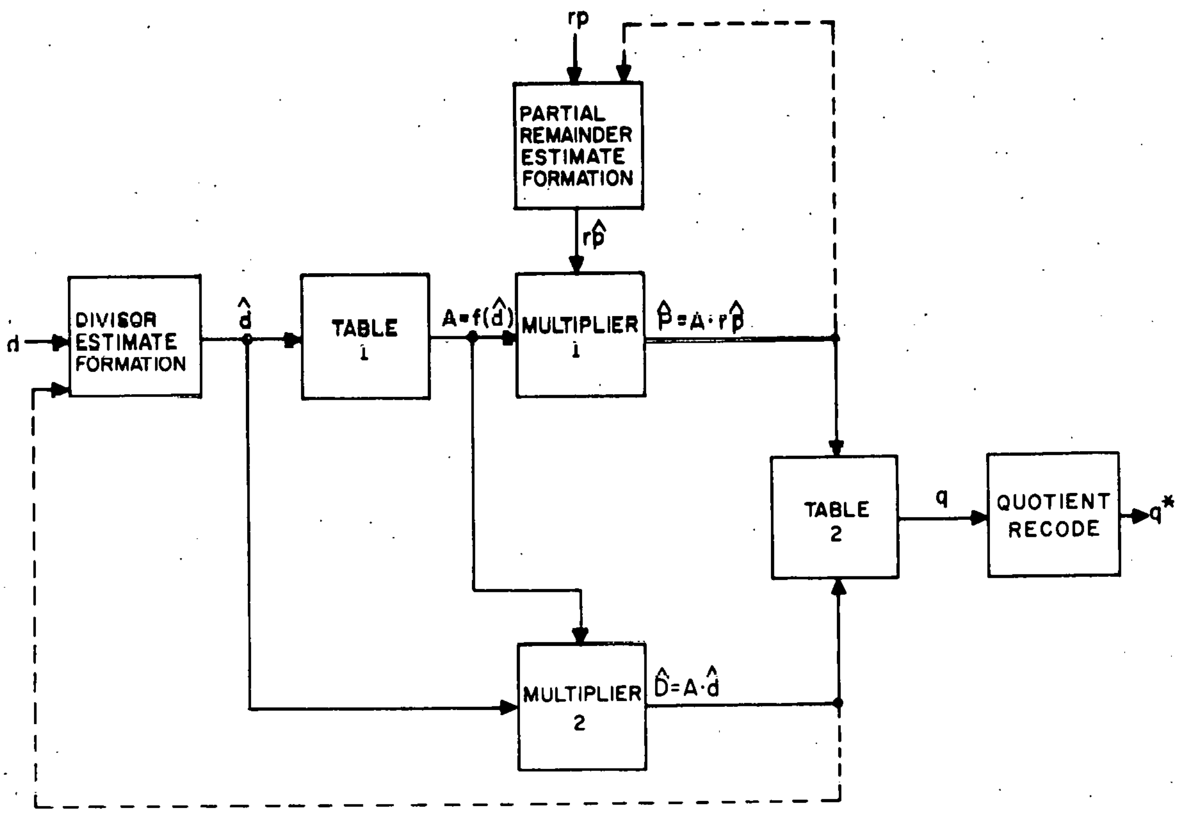


Figure 2. Generalized Structure of Model Division (Quotient Selector)

is coupled back into this box. This feedback loop together with the one from Multiplier 1 to the partial remainder estimate formation box admits iterative multiplicative schemes into this structure.

Table 1 - This block accepts $\hat{d}$ as the input and produces a value, A, as a function of $\hat{d}$, i.e. $A = f(\hat{d})$. The quantity A is a factor by which both $\hat{d}$ and $\hat{rp}$ are multiplied (the quotient is therefore not changed). It may be interpreted as a scale factor used to transform the range of $\hat{d}$ or as an estimate of the inverse of $\hat{d}$.

Partial Remainder Estimate Formation - This block accepts a full precision shifted partial remainder, rp, and from it produces an estimate of the shifted partial remainder, $\hat{rp}$, with maximum negative uncertainty of λ and maximum positive uncertainty, γ. As with divisor estimate formation, the estimate is in practice a truncated version of the full precision quantity. The block may also incorporate provisions for changing the form of the representation.

In actual implementations the full precision remainder may be the result of operations using an adder-subtracter which produces a redundant representation. The estimate of the remainder, $\hat{rp}$, however, is restricted to non-redundant representations. We have assumed, although not rigorously demonstrated the fact, that use of a redundantly represented value would unduly complicate the structure of the quotient selection mechanism. Merely determining the sign, for example, is of the same order of complexity as converting the value into a non-redundant form. It is important to realize, however, that the estimate consists of only the high order digits of the full precision remainder. In practice this estimate is sufficiently short to permit conversion to a non-redundant form using full-lookahead techniques.

The partial remainder estimate formation block also enables the output of Multiplier 1 to couple back into the input side. As with the divisor loop, this path is necessary for the inclusion of the iterative multiplicative division scheme.

Multipliers - Multiplier 1 and Multiplier 2 form, respectively, the quantities $\hat{P} = A\,\hat{rp}$ and $\hat{D} = A\,\hat{d}$. The outputs of both multipliers are the inputs to the second table look-up structure, Table 2. $\hat{P}$ may be thought of as a transformed version of $\hat{rp}$. The maximum negative uncertainty in P is λA_{max}; the maximum positive uncertainty is γA_{max}, where $A_{max} = f(b)$. If the product, Arp is truncated so that non-zero digits are lost, additional uncertainties λ_m and γ_m are introduced. In this case $\hat{P}$ represents transformed rp values in the range

$$\hat{P} - A\lambda - \lambda_m \leq Arp \leq \hat{P} + A\gamma + \gamma_m. \tag{2.17}$$

Similarly, the maximum uncertainties in D are A_{max}, βA_{max} with $A_{max} = f(b)$. If $\hat{D}$ is truncated with maximum truncation errors (α_m, β_m) then $\hat{D}$ is representative of transformed d values in the range

$$\hat{D} - A\alpha - \alpha_m \leq Ad \leq \hat{D} + A\beta + \beta_m \tag{2.18}$$

Table 2 - This structure is an implementation of the function which relates quotient digits, q, to the products $\hat{P}$ and $\hat{D}$, the scaled remainder and divisor, respectively, for the model division.

Quotient Recode - The quotient recode block represents the interface between the output of the model division and the full precision division. The output of Table 2, q, may require a recoding into a form directly usable by the shift gate complex which selects the next multiple of the divisor to be used in forming the subsequent partial remainder.

At this point we narrow the scope of the present research to exclude iterative multiplicative schemes: the feedback loops of Figure 2 will not be used. The remaining structure includes what might be considered two extremes or boundary cases. In the one structure, to be designated Type 1, Table 1 is defined such that the rounded, integer portion of the product $A\ \hat{rp}$ is the correct quotient digit for the division, rp/d. For a Type 1 structure neither Table 2 nor Multiplier 2 need be implemented. The other extreme occurs when $A = f(\hat{d}) = 1$. In this case, designated Type 2, Table 2 bears the full burden of quotient selection and neither Table 1 nor the multipliers are required.

But there are also intermediate, hybrid, structures in which neither Table 1 nor Table 2 is degenerate. In these structures Table 1 and the multipliers are used to transform $A\ \hat{d}$ into a range closer to 1 than was $\hat{d}$. The effect of this range transformation is to simplify Table 2. In the next chapter we shall examine the design of Table 1 and Table 2 independently and then make some observations about hybrid structures. The shift from a Type 1 structure to a Type 2 structure and accompanying trade-off between speed and hardware is but an example of the trade-offs available between sequential networks and their combinatorial equivalent.

3. DEFINITION OF COST AND PERFORMANCE

3.1 Preliminary Remarks

To this point in the thesis we have defined a division procedure which generates a quotient by successive calls to a lower precision, model division unit. A generalized structure of the model division was proposed and now we begin to consider the synthesis of such a unit.

Besides the definitive aspects of this work, a major goal is to derive useful estimates of minimal cost and performance as functions of the design parameters of the generalized structure in Figure 2. Design parameters include such quantities as radix, r; magnitude of maximum quotient digit, n; and the point of truncation of rp and d. In this section, the important boxes of Figure 2 are made sufficiently specific to allow a measure of minimal cost and performance to be proposed.

In finding a measure of cost or performance, the designer is faced with a trade-off between generality and accuracy. Determining absolute cost or absolute performance is very much dependent upon hardware and details of implementation; but restricting the study to a specific class of logic limits the significance of the work. Questions of minimization are further complicated by controversy as to what to minimize.

This work makes a compromise. Since much of the emphasis is on <u>comparison</u>, a <u>relative</u> measure of cost and performance is adequate. On the other hand, some <u>estimate</u> of <u>absolute</u> cost is desirable. The higher-radix, table look-up schemes offer potentially high performance but require a larger number of gates to construct. Whether, in fact, they are at all feasible for a real machine strongly depends upon the absolute cost.

3.2 Definition of Cost

3.2.1 Preliminary Remarks

For this study, the cost of a logic network is defined as the total number of literals required to implement the network in two-level, sum-of-products (AND-OR or equivalent) logic. The choice ignores fan-in and fan-out restrictions, but this shortcoming is outweighed by the following considerations.

1. The logical definitions of the networks are in a canonical form which can serve as an input to a specific minimization and/or design automation package.

2. The networks are realized in the theoretical minimum number of circuit delays and thus will be an upper bound on speed and cost.

3. The tables for higher-radix structures are candidates for LSI. In this case the number of literals is a measure of silicon area required and power dissipation requirements.

4. A very efficient computer program for sum-of-products minimization is available to the author.

The cost of implementing the structure shown in Figure 2 is the sum of the costs of implementing each sub-block. Symbolically,

$$C = C_{DEF} + C_{T1} + C_{M1} + C_{M2} + C_{PREF} + C_{T2} + C_R \tag{3.1}$$

where

C is the total cost,

C_{DEF} is the cost of the Divisor Estimate Formation block,

C_{T1} is the cost of Table 1,

C_{M1} is the cost of Multiplier 1,

C_{M2} is the cost of Multiplier 2,

C_{PREF} is the cost of the Partial Remainder Estimate Formation block,

C_{T2} is the cost of Table 2, and

C_R is the cost of the Quotient Recode block.

At this point, it is convenient to introduce intermediate variables, C_{TMM} and C_{DPQ}, and group the cost terms as follows:

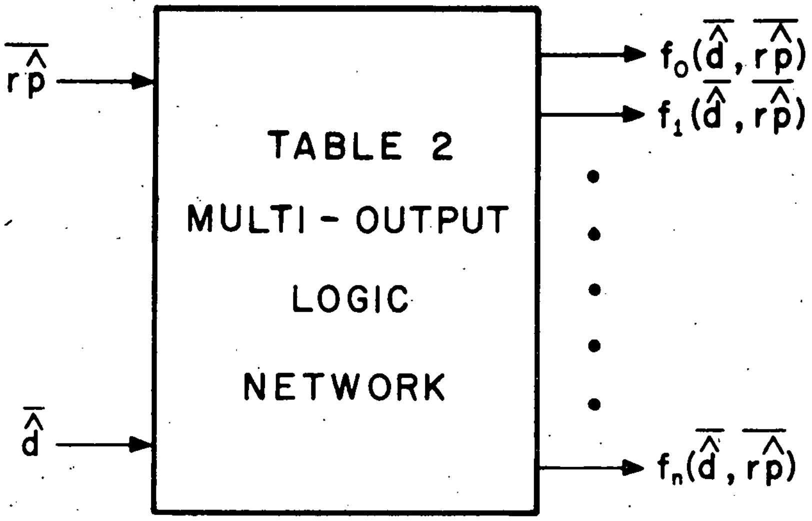
$$C_{TMM} = C_{T1} + C_{M1} + C_{M2} \tag{3.2}$$

$$C_{DPQ} = C_{DEF} + C_{PREF} + C_R \tag{3.3}$$

The cost terms C_{T2}, C_{TMM}, and C_{DPQ} are functionally related to the design parameters such as radix, maximum quotient digit, range of divisor, and uncertainty in the estimates of the divisor and remainders. The terms C_{T2} and C_{T1} in C_{TMM} are the most complex and will be studied by computer synthesis. Estimates of C_{DPQ} and the remaining terms of C_{TMM} will be obtained manually as required. In most cases, the term C_{DPQ} is dominated by $C_{T2}+C_{MM}$ and may be neglected.

3.2.2 Structure for Finding Cost of Table 2

Table 2 will be studied as a multiple-output logic network. It may be represented as shown in Figure 3. The functions, f_o through f_n are Boolean functions of the bit vectors corresponding to $\hat{d}$ and $\hat{rp}$. These vectors are denoted $\bar{\hat{d}}$ and $\bar{\hat{rp}}$, respectively.



$\overline{r\hat{p}}$
$\hat{\bar{d}}$
TABLE 2
MULTI - OUTPUT
LOGIC
NETWORK
$f_0(\bar{\hat{d}}, \overline{r\hat{p}})$
$f_1(\bar{\hat{d}}, \overline{r\hat{p}})$
$f_n(\bar{\hat{d}}, \overline{r\hat{p}})$

Figure 3. Network Definition of Table 2

In specifying the quotient selection criterion (Section 2.3), every pair $(\hat{d}, r\hat{p})$ has been associated with a set, I, of quotient digits which the quotient selection mechanism may generate when given inputs $(\hat{d}, r\hat{p})$. The functions, f_o, f_1, ..., f_n must be found such that for every ordered pair, $(\hat{d}, r\hat{p})$ with allowable quotient digit set, I,

$$f_i\ (\bar{\hat{d}}, \overline{r\hat{p}}) = 1 \quad \text{for one and only one } i \varepsilon I, \tag{3.4}$$

and $$f_k\ (\bar{\hat{d}}, \overline{r\hat{p}}) = 0 \text{ for all other values of } i. \tag{3.5}$$

In other words, every pair $(\hat{d}, r\hat{p})$ in the set $\hat{D} \times \hat{P}_o$ must cause one and only one of the outputs to be true, and this output must correspond to a correct quotient digit.

Due to the overlap of adjacent quotient regions produced by redundancy, many elements in $\hat{D} \times \hat{P}$ may have sets, I, containing more than one element, thus many sets of different functions are allowable for given design parameters. But our wish to compare minimal* costs imposes another constraint, namely, that the cost of the multiple output network (as defined in Section 3.2.1) is minimal. Symbolically stated: the requirement is that Cost ($f_o + f_2 + f_2 + \cdots + f_n$) be minimal.

In the general minimization of two-level, AND-OR realization of a multiple-output network, it is necessary to generate the prime implicants of each of the individual output functions, plus the prime implicants of the functions which are equal to all possible products of two output functions, three output functions, etc. Each product is a multiple-output prime implicant. McCluskey [33], states the following theorem of use here:

> Theorem: For any definition of networks cost such that the cost does not increase when a gate or gate input is removed, there exists at least one minimum-cost, two-stage network in which the corresponding expressions for the output functions, f_j, are all sums of multiple-output prime implicants. All the product terms which occur only in the expression for f_j are prime implicants of f_j; all the product terms which occur in both the expressions for f_j and f_k but in no other expressions are prime implicants of $f_i \cdot f_k$, etc.

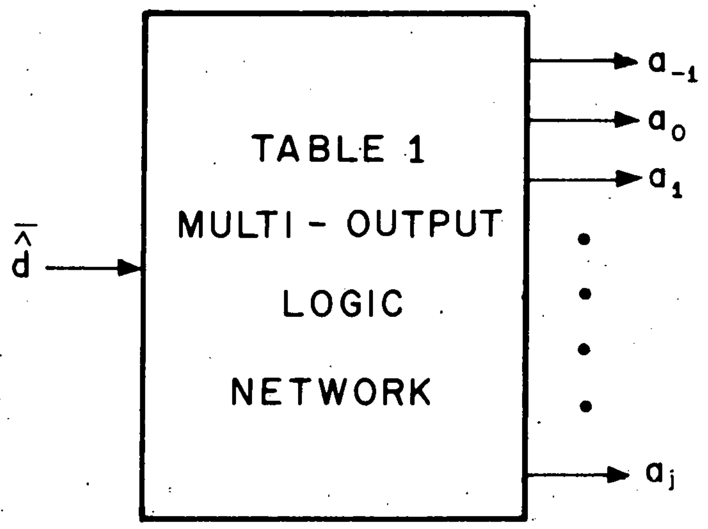
But in the present case, no two functions are ever simultaneously true and thus none of the prime implicants of f_j are contained in any other function, f_k, $k \neq j$. Thus, by the theorem stated above, there exists a minimum cost two stage network which may be found by minimizing each function independently of the rest, i.c.

$$\text{Min Cost } (f_o + f_1 + \cdots + f_n) = \text{Min Cost } (f_o) + \text{Min Cost } (f_1) + \cdots + \text{Min Cost } (f_n).$$

*The term minimal, implies that we wish to find any one of possibly more than one minimum cost implementations.

3.2.3 Structure for Finding Cost of Table 1 and the Multipliers

As with Table 2, Table 1 will be defined as a multiple-output logic network as shown in Figure 4. The input is $\bar{\hat{d}}$, the bit-vector representation of d. The outputs are the variables $a_{-1} = g_{-1}(\bar{\hat{d}})$, $a_o = g_o(\bar{\hat{d}})$, ... $a_j = g_j(\bar{\hat{d}})$, where g_i is a Boolean function. The bits, a_{-1} through a_j comprise the binary representation of inverse of $\hat{d}$, A. Unfortunately, in this case, we cannot constrain the problem so that none of the outputs are simultaneously true. For purpose of estimation, however, it will be assumed that the results obtained by minimizing each function independently will yield an adequate estimate of the minimum cost, i.e. C_{T1} = Min Cost g_o + Min Cost g_1 + ... + Min Cost g_j.



$$A = a_{-1}\, a_o \cdot a_1 \ldots a_j$$

Figure 4. Network Definition of Table 1

We now consider the cost of the multipliers. It is beyond the scope of this work to develop a cost-performance analysis for multiplication structures. The approach adopted here is to present a structure which experience has shown to be efficient and to approximate C_{M1} from the structure. More information about such a structure may be found in [8].

The multiplier is illustrated in Figure 5. It consists of a cascade of limit carry-borrow adder-subtracters together with shift-gates (S.G.) which form the necessary multiples of the multiplicand $(r\hat{p})$. Shift gate SG0, in conjunction with complementing circuits, form the multiples ± 1 and ± 2 times; SG1 forms ± 4, ± 8 times; and, in general, SGi, form multiples of $\pm 2^{2i+1}$ times the multiplicand. The multiples are selected by a recoding of a_o through a_j. Appended to the output of the last adder is hardware which converts the products from the redundant representation produced by the limited-carry or borrow device to a non-redundant format. The cost of Multiplier 1, C_{M1}, will be defined by

$$C_{M1} = jC_R + N_A N_B C_A + (N_A+1) N_B C_{SG} + N_B^2 C_c \tag{3.6}$$

where

C_R is the cost per input digit of the recoding logic,

N_A is the number of adders in the multiplier and is given by

N_A = Integer portion of $(j + 1)/2$,

N_B is the number of bit positions per adder and is given by

$N_B = \varepsilon + \log_2 r$,

C_A is the cost of one position of an adder,

C_{SG} is the cost of one position of a shift gate,

C_C is the cost of converting one digit from redundant to non-redundant form (assuming the use of look ahead techniques).

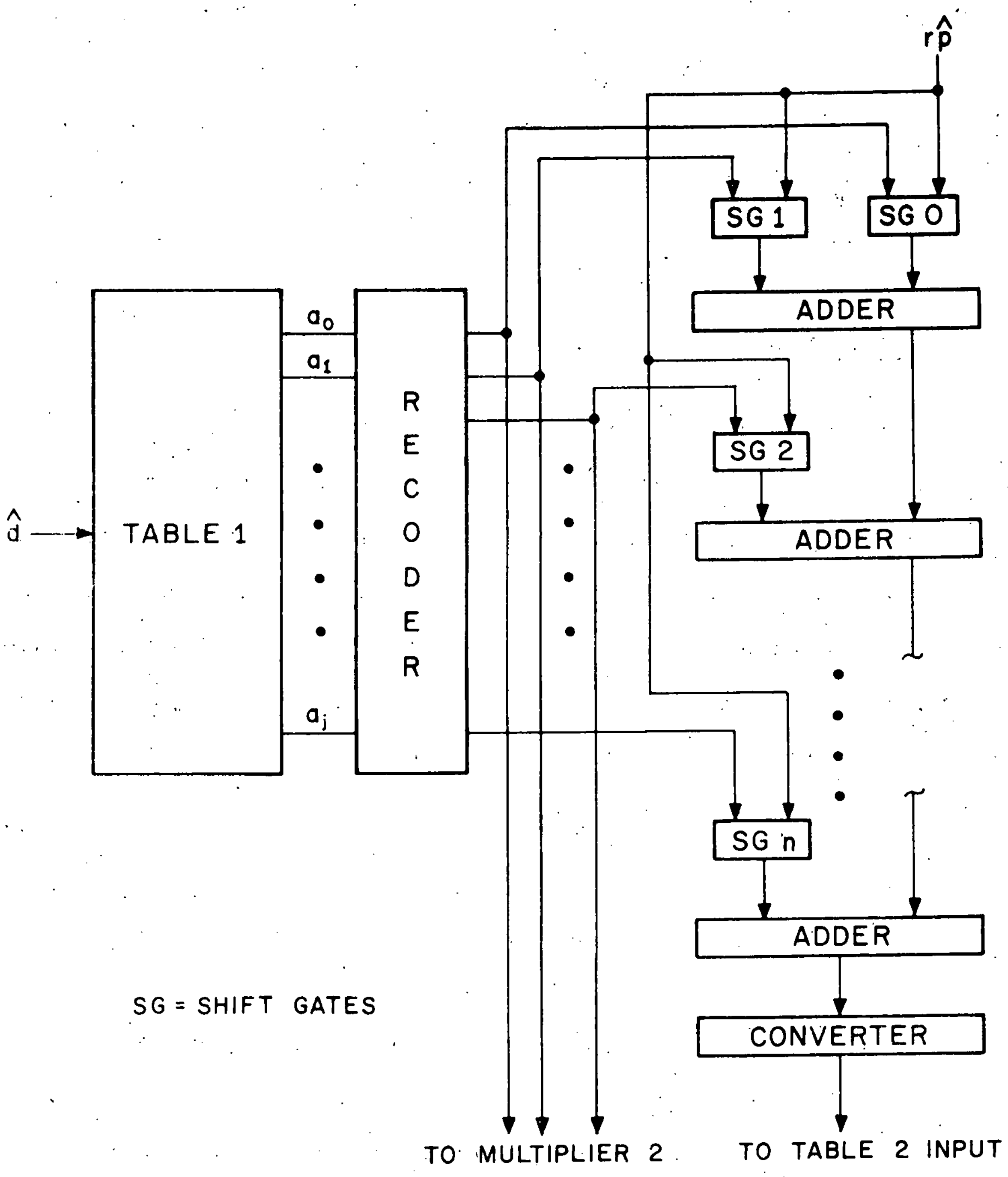


Figure 5. Structure of Multipliers

The quantity, j, is the index of the low-order bit of A, the approximation of d^{-1} ($A = a_o \, . \, a_1 a_2 \ldots a_j$), ε is the number of bits to the right of the radix point in $r\hat{p}$, and r is the radix of the model division. As the need arises, estimates of minimum values of C_R, C_A, C_{SG}, and C_C may be obtained. The cost of M_2, C_{M2}, is given by Equation 3.6 with ε replaced by ($\varepsilon + \log_2 r$) replaced by δ, the number of bits in $\hat{d}$.

3.3 Definition of Performance

3.3.1 Performance of the Model Division

Performance will be measured in units of number of bits of quotient generated per gate delay. For practical cases, the number of bits of quotient generated by the model division is $\log_2 r$. Since the divisor is constant for a given division operation, the operating time of the model division is limited by the paths driven by the remainder. The time, T_Q, in gate delays, required to produce a quotient digit, radix r, is given by

$$T_Q = T_{PREF} + T_{M1} + T_{T2} + T_R \tag{3.7}$$

where

T_{PREF} is the number of logic delays required in forming the estimate of the remainder,

T_{M1} is the number of logic delays required to form $A \, r\hat{p}$ in Multiplier 1,

T_{T2} is the number of logic delays to select a quotient digit in Table 2, and

T_R is the number of logic delays to recode the output of T2.

Performance of the quotient selector, P_Q, is therefore given by

$$P_Q = \frac{\log_2 r}{T_Q} \tag{3.8}$$

3.3.2 Performance of the Full Precision Division

The measure of primary interest is the performance of the full precision division. We shall assume a full precision multiplication structure similar to that shown in Figure 5. It consists of a cascade of K adder subtracters each of which is capable of retiring K' bits of the multiplier. The effective radix for multiplication is therefore $r_M = 2^{kk'}$.

Let,

M be the quotient length in bits,

T_D be the number of logic delays required for the iterative steps of division,

T_A be the number of logic delays required to add two full precision numbers,

T_C be the number of logic delays required for control after the quotient bits have been generated by the quotient selector, and

N_Q be the number of calls to the quotient selector.

Then,

$$T_D = \frac{M}{K'} T_A + N_Q (T_Q + T_C) \tag{3.9}$$

where, if r is the radix of the model division,

$$N_Q = \frac{M}{\log_2 r} \tag{3.10}$$

For this study, K' = 2, thus

$$T_D = M\left(\frac{T_A}{2} + \frac{T_Q + T_C}{\log_2 r}\right) \tag{3.11}$$

The performance of the full precision division is defined by

$$P_D = \frac{M}{T_D} = \left(\frac{2 \log_2 r}{T_A \log_2 r + 2(T_Q + T_C)}\right) \tag{3.12}$$

4. ALGORITHMS FOR SYNTHESIS AND ANALYSIS

4.1 Preliminary Remarks

The derivation of cost and performance functions by a direct, analytic approach is complicated by the discrete nature of these functions and by the large number of variables. An empirical, constructive approach was therefore adopted. The first phase of the experiment (the topic of this section) required the formulation of a systematic approach to the synthesis of a minimal cost, mathematically accurate, quotient selection mechanism for a given set of design parameter values. Although the synthesis routines in themselves would be of use in designing a quotient selection mechanism, in this study they are used as tools in studying the cost and performance functions. We are performing analysis by means of computer-aided synthesis.

In the second phase of the experiment, the programs developed in the first phase were run with various combinations of parameter values in order to estimate cost and performance. The results of each run might be thought of as determining a point on a cost versus performance curve. The hope is that only a few runs, relative to all possible parameter combinations, would be necessary in order to find approximations which would be useful for interpolation and extrapolation.

But this empirical approach is not without major practical problems. There are a huge number of possibilities for parameter values, and the minimization problems are very large and demanding of computer time. These problems were mitigated by restricting the values of parameters to those of practical importance and by concentrating on the effects of dominant parameters.

As discussed in Section 3, the dominant cost term for a Type 2 structure is C_{T2}, the cost of Table 2. For a Type 1 structure, although the cost of Table 1 (C_{T1}) may not dominate the cost of the multiplier, it is the least studied term. The following sub-sections comprise a description of algorithms which generate logic equations which define Table 1 and Table 2 for given values of design parameters. The algorithms do not produce a definition of the other blocks of Figure 2, but do place some constraints upon their structure.

4.2 <u>Deriving a Minimal Cost Design for Table 2</u>

Conceptually, Table 2 in Figure 2 is a direct implementation of a P-D plot. To implement a given P-D plot, a relation must be defined from the set D x P to a subset of D x P, $\hat{D}$ x $\hat{P}$, such that each element of D x P maps into an element of $\hat{D}$ x $\hat{P}$ and with error bounds for each element ($\hat{d}$, $\hat{rp}$) such that the quotient selection criterion is satisfied. Note that we have not required that the relation be a function, since, due to redundant representation, the same rp-value or d-value may map into different $r\hat{p}$ or $\hat{d}$ values; uniqueness is not guaranteed. For practical reasons the relation is restricted to those which may be defined by the successive operations of truncation and assimilation (conversion to a non-redundant form). Even within this restriction, however, there are many possible alternatives. The maximum amount of truncation error which may be tolerated for a given pair ($\hat{d}$, $\hat{rp}$) depends upon the location of the point. There is also trade-off between ε and δ, the points of truncation of rp and d, respectively.

The following is a list of the steps in the process of deriving a minimal cost design for Table 2.

1. Set the values for design parameters:

   $n, r, a, b, \alpha', \beta', \gamma', \lambda', \varepsilon, \delta$.*

2. Run the program QS3 (described in Section 4.2.1) to produce a sum-of-products (minterm) definition of each output function of Table 2.

3. Run the program, PI, with each set of minterms produced by QS3 as input. The program PI finds all prime implicants of the functions, identifies the essential prime implicants, and generates the constraints which must be satisfied in order to cover the function.

4. Run an Integer Linear Programming routine to find a minimal cost set of prime implicants which satisfy the constraints produced in step 3. The cost of a prime implicant is the number of literals. The combination of the prime implicants selected in this step, together with the essential prime implicants identified in step 3, define the Boolean function.

5. Tabulate the total number of literals required to define each output functions. The total of these values will be taken as the cost of implementing Table 2.

4.2.1 Defining the Output Functions

As described in Section 3.2.2, Table 2 is treated as a multiple output network. This section describes an algorithm for specifying these

*Initially, Table 2 is studied apart from T1, M1, and M2. $A = F(\hat{d}) = 1$.

functions as sums-of-products of minterms. The minterms are formed by concatenating bit vectors, $\overline{\hat{rp}}$, with bit vectors, $\overline{\hat{d}}$. A Fortran program called QS3 (Quotient Selection Program 3) was written to accept design parameters and to produce the minterm definitions of each of the output functions, $f_o(\overline{\hat{rp}}, \overline{\hat{d}})$, ..., $f_n(\overline{\hat{rp}}, \overline{\hat{d}})$.

The derivation will be restricted to the first quadrant (positive rp and d) of the P-D plot. The full P-D plot is symmetric about both axes and thus the cost of implementing one quadrant is a good estimate of the cost of implementing any other.

Figure 6 illustrates a portion of the first quadrant of a P-D plot. Three adjacent quotient regions, q (i+1), q(i), and q (i-1) are designated together with the horizonal line, $rp = \hat{rp} = m\Delta rp$. Every line of this form will be designated an "rp-line". The quantity, m, is an integer, and $\Delta rp = 2^{-\varepsilon}$. The task of defining the output functions for Table 2 may be reduced to that of assigning adjacent sections of every rp-line to one and only one q-region. For example, the segment of the rp-line between d = a and d = b must be subdivided into three segments: one in each q-region shown. The dividing line between adjacent line segments assigned to q(i) and q (i+1) will be called the "divisor transition value between q(i) and q(i+1)." A divisor transition value between q(i) and q(i+1) may be picked from a sub-range of the divisor between the intersections of the rp-line and the boundaries of the overlap region. The range in which the divisor transition value may be chosen is determined as follows.

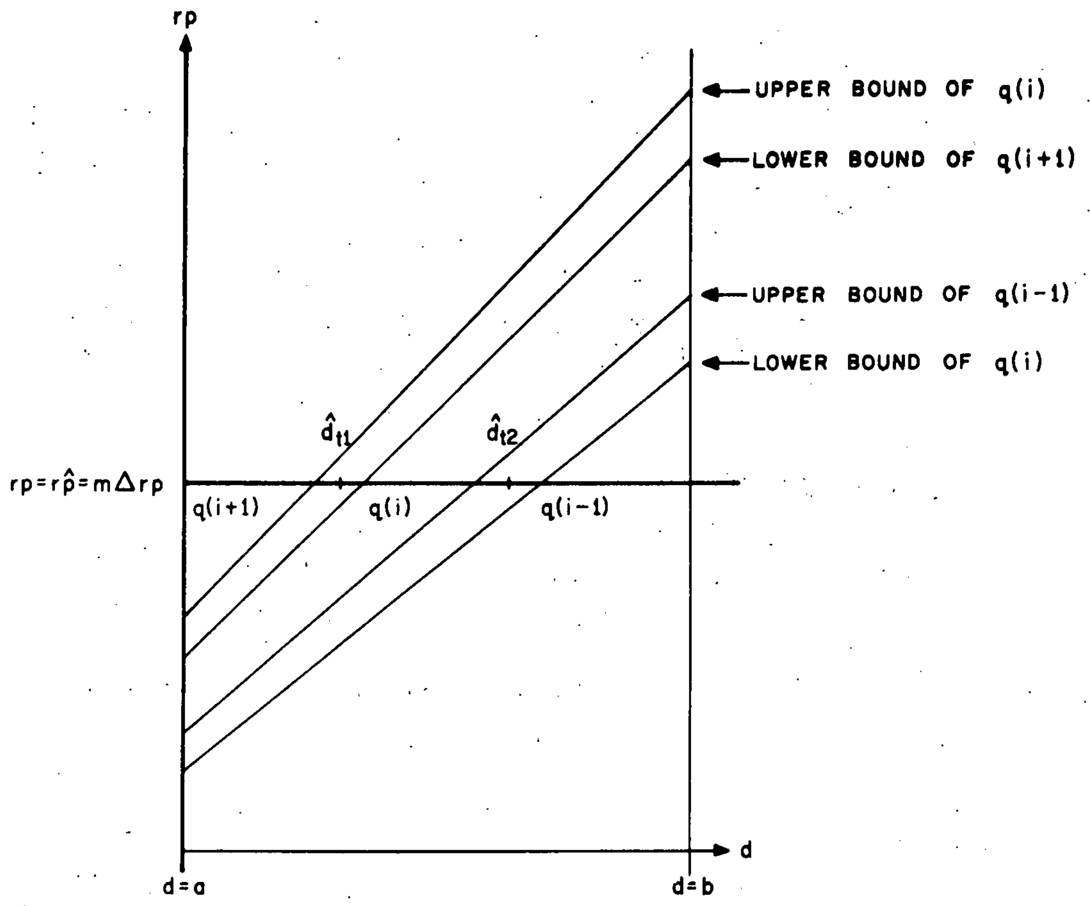


Figure 6. Portion of P-D Plot Illustrating Segmentation of rp-line

Let $\hat{d}_t$ be the divisor transition value for rp = $\hat{rp}$, between q(i) and q(i-1). Then the ordered pair ($\hat{rp}$, $\hat{d}_t$) will be representative of all (rp, d) in the rectangle shown in Figure 7. Since $\hat{d}_t$ is a transition value, ($\hat{d}_t$, $\hat{rp}$) implies a quotient digit of i-1 and ($\hat{d}_t$ - Δd, $\hat{rp}$) implies a quotient digit of i.

The rectangle corresponding to ($\hat{d}_t$, $\hat{rp}$) must be completely within the q(i-1) region. The strictest bound is therefore at the upper, lefthand corner of the rectangle in Figure 7, and thus the following must hold.

$$\hat{rp} + \gamma \leq (i - 1 + \rho)\ (d_t - \alpha) \qquad (4.1)$$

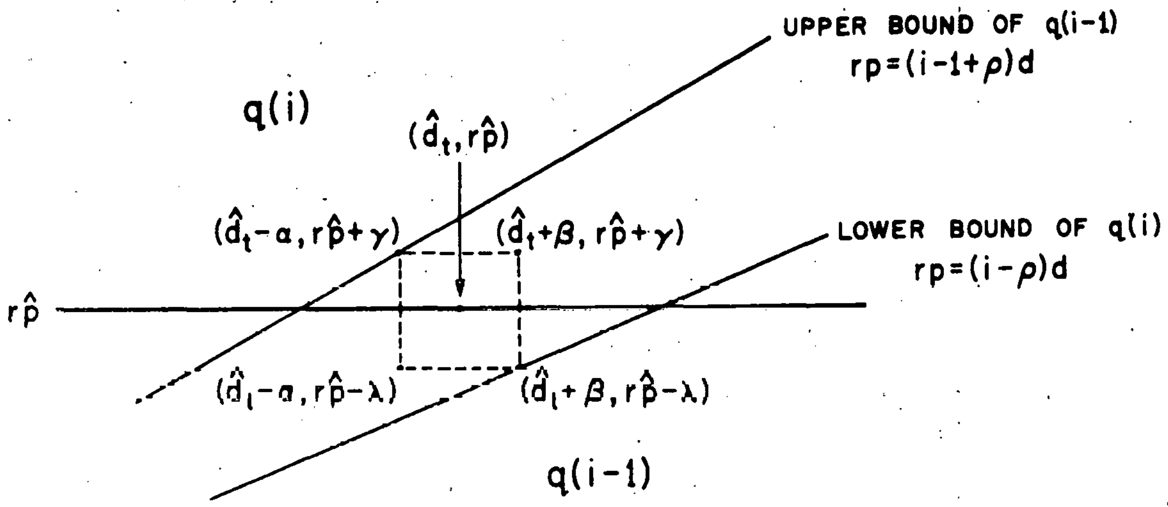

UPPER BOUND OF q(i-1)
rp=(i-1+ρ)d
q(i)
($\hat{d}_t$, r$\hat{p}$)
($\hat{d}_t$-α, r$\hat{p}$+γ)
($\hat{d}_t$+β, r$\hat{p}$+γ)
LOWER BOUND OF q(i)
rp=(i-ρ)d
r$\hat{p}$
($\hat{d}_t$-α, r$\hat{p}$-λ)
($\hat{d}_t$+β, r$\hat{p}$-λ)
q(i-1)

Figure 7. Portion a P-D Plot Illustrating Constraints in Finding Divisor Transition Interval

Similarly, the rectangle corresponding to $(d_t - \Delta d, \hat{rp})$ must be completely within the q(i) region. The strictest bound in this case is at the lower, righthand corner of the rectangle where the following must hold.

$$\hat{rp} - \lambda \geqq (i-\rho)\ (\hat{d}_t - \Delta d + \beta) \qquad (4.2)$$

In practical cases, to insure that all d values map into at least one d value, $\Delta d = \beta$ and thus (4.2) becomes

$$\hat{rp} - \lambda \geqq (i-\rho)\hat{d}_t \qquad (4.3)$$

Combining (4.1) and (4.3) yields a range restriction on $\hat{d}_t$, namely,

$$(\hat{rp} + \gamma)/(i-1+\rho) + \alpha \leqq \hat{d}_t \leqq (\hat{rp} - \lambda)/(i-\rho) \qquad (4.4)$$

Note that the strategy is to select the size of the rp-steps, Δrp, and to allow the algorithm to find the maximum size steps allowable for d. Theoretically, the program could be designed such that Δd would be specified and the precision requirements for the partial remainder would be determined. The former approach is taken due to the fact that control of Δrp is more critical. The precision of the estimate of the partial remainder (the number of bits) should be kept low in order to keep down the time required to convert from a redundant to a non-redundant form. The logic paths involving $\hat{rp}$ as opposed to those involving $\hat{d}$, are changing with each call to the model division. For this reason there is motivation to simplify the logic involving only $\hat{rp}$ at the expense of complicating the logic involving only $\hat{d}$. It should also be realized that the precision requirements on the estimate of the partial remainder are based upon worst case calculations. Although QS3 uses this worst case precision uniformly in generating the division precision requirements, the minimization routines will remove unneeded precision.

The quantity, d_t, may be any value in the range defined in Equation 4.4. Since the design goal is to minimize the total number of literals required to implement the table, $\hat{d}_t$ is picked to be a number which can be represented with the fewest bits. In other words, if all numbers in the range specified by (4.4) are represented as the ratio of two integers in the form $N/2^M$, the $\hat{d}_t$ selected is one satisfying (4.4) and with the minimum value of M.

Using the algorithm of selecting the simpliest binary number in the allowable divisor transition ranges, the rp-line in Figure 6 is divided into three segments, as follows:

Segment	Assigned to
$a \leq d < d_{t1}$	q(i+1)
$d_{t1} \leq d < d_{t2}$	q(i)
$d_{t2} \leq d < b$	q(i-1)

The segments are next defined as minterms and the minterms are assigned to the appropriate output function, f_{i+1}, f_i, f_{i-1}, etc.

The complete family of rp-lines is produced by stepping along the rp-axis (beginning at 0) in increments of Δrp. By segmenting the rp-lines at the boundaries of the P-D plot and the divisor transition values, each quotient region, q(i) for i=0 through n, is defined by a set of triplets of the form

$$(d_{l,m}, d_{r,m}, m\Delta rp)$$

where

$d_{l,m}$ is the left end of the segment of the mth rp-line in q(i);

$d_{r,m}$ is the right end of the segment of the mth rp-line in q(i); and

$m \cdot \Delta rp$ defines the values of $\hat{rp}$.

Rather than being stored as triplets, each segment is stored as a set of minterms.

Given the ordered pair, $(\hat{d}, \hat{rp})$, the minterm equivalent is $\overline{\hat{rp}} \, || \, \overline{\hat{d}}$ where || denotes bit string concatenation. The minterm may be represented as a bit string or as decimal integer equivalent of $\overline{\hat{rp}} \, || \, \overline{\hat{d}}$, treated as a binary integer. Each triplet, $(d_{l,m}, d_{r,m}, m\Delta rp)$ is transformed into a pair of minterms, $(MINTRM_l, MINTRM_r)$. Under this transformation, each segment of the rp-line is logically defined by $MINTRM_l \lor (MINTRM_l + 1) \lor \ldots \lor MINTRM_r$. The triplets are converted to minterms as follows.

The quantities $d_{l,m}$ and $d_{r,m}$ are all divisor transition values and are therefore of the form $N/2^{\delta}$. For a given q(i) region, find the largest δ, δ_{max}, required to represent $d_{l,m}$ or $d_{r,m}$. Then $2^{-\delta max}$ is the maximum precision required to represent d. Given $d_{l,m} = N_l / D_l$; $d_{r,m} = N_r/D_r$ (both fractions in reduced form) rp = mΔrp, and NBDL = the number of bits of the divisor to the right of the radix point, then

$$\text{MINTRM}_l = m2^{(\delta_{max} + \text{NBDL})} + (2^{\delta_{max}} \ N_l)/D_l \qquad (4.5)$$

$$\text{MINTRM}_r = (m2^{(\delta_{max} + \text{NBDL})} + (2^{\delta_{max}} \ N_r)/D_r) - 1. \qquad (4.6)$$

A useful estimate of the number of minterms required to define a given q (i) region may be derived. The QS3 algorithm will actually select the upper and lower boundary of each q (i) region which will be a stairstep in the transition region between q (i), q (i + 1) and q (i - 1). For purposes of this estimate, assume that the dividing line between q (i) and q′ (i + 1) is the average value between the upper boundary of q (i) and the lower boundary of q (i + 1). The boundary between q (i) and q (i + 1) is thus defined by rp = (i + 1/2) d. The area of each q (i) region will be defined as the area between the lines d = a, d = b, rp = (i + 1/2) d, and rp = (i - 1/2) d. Thus,

$$\text{Area } (q\ (i)\ ) = \int_a^b x \ dx = (b^2 - a^2)/2. \qquad (4.7)$$

The area is independent of the value of the quotient digit. Let ε be the number of bits to the right of the radix point in $\hat{rp}$ ($\Delta rp = 2^{-\varepsilon}$) and δ be the number of bits to the right of the radix point in $\hat{d}$. Note that the minimum value of δ may increase with i. If the worst case value of δ is applied uniformly in defining all quotient regions, the bits of excess precision will become don't care literals in the course of minimization. To reduce the minimization problem, δ may be treated as a function of i by defining δ (i) as the minimum number of bits required in $\hat{d}$ in order to correctly define the q (i) region for the given value of ε. The number of minterms for each q (i) region, M (i), is thus given by

$$M\ (i) = (b^2-a^2)\ 2^{(\varepsilon+\delta(i)\ -\ 1)}. \qquad (4.8)$$

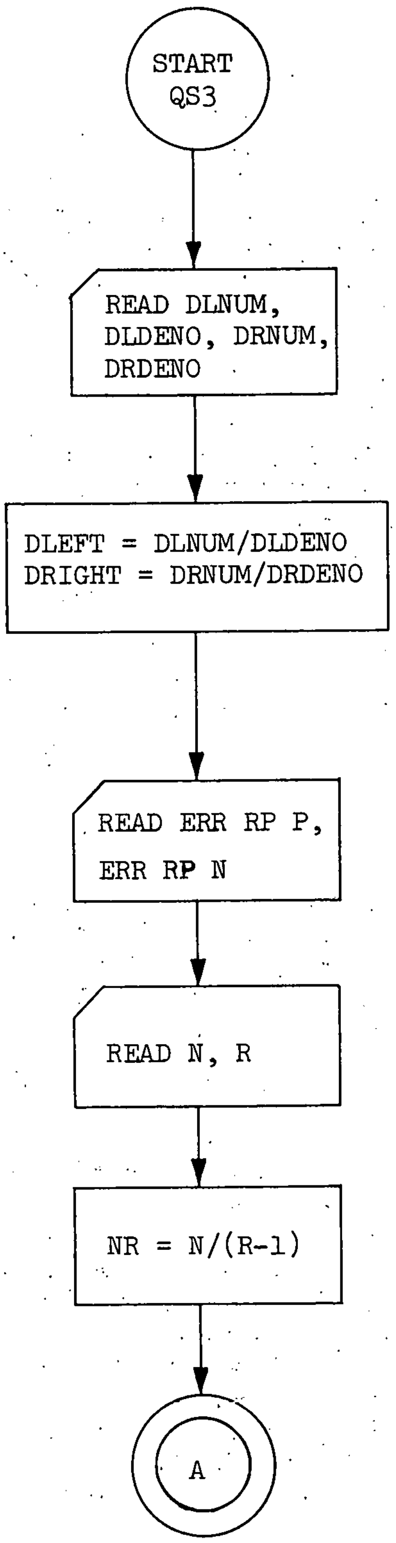
Figure 8 is an annotated flowchart of the program (QS3) which actually produces the definition of the output functions for Table 2. The following assumptions and conventions should be noted:

1. The program was written in Fortran and thus Fortran notation and variable names are used in the flowchart.

2. In most cases, the Fortran variable names differ from that used in Section 2. Included in the comments are statements which related the Fortran name to that used in the derivations. For example, DLEFT $\equiv$ a (2.1). The number in parentheses is the section number in which "a" is defined.

3. The divisor is restricted to positive values in a non-redundant representation and thus $\alpha = 0$ in Equation 4.4.

4. Single circles on the flowchart denote entrances; double concentric circles denote exits.



The endpoints of the divisor interval are read in a fractional form. DLNUM and DLDENO are the numerator and denominator, respectively, of the left end. DRNUM and DRDENO are the numerator and denominator, respectively, of the right end.

DLEFT ≡ a (2.1)
DRIGHT ≡ b (2.1)

ERR RP P is the maximum positive truncation error in rp; ERR RP N is the maximum negative truncation error in rp.

ERR RP P ≡ γ (2.3)

ERR RP N ≡ λ (2.3)

N is the maximum allowable quotient digit. R is the radix.

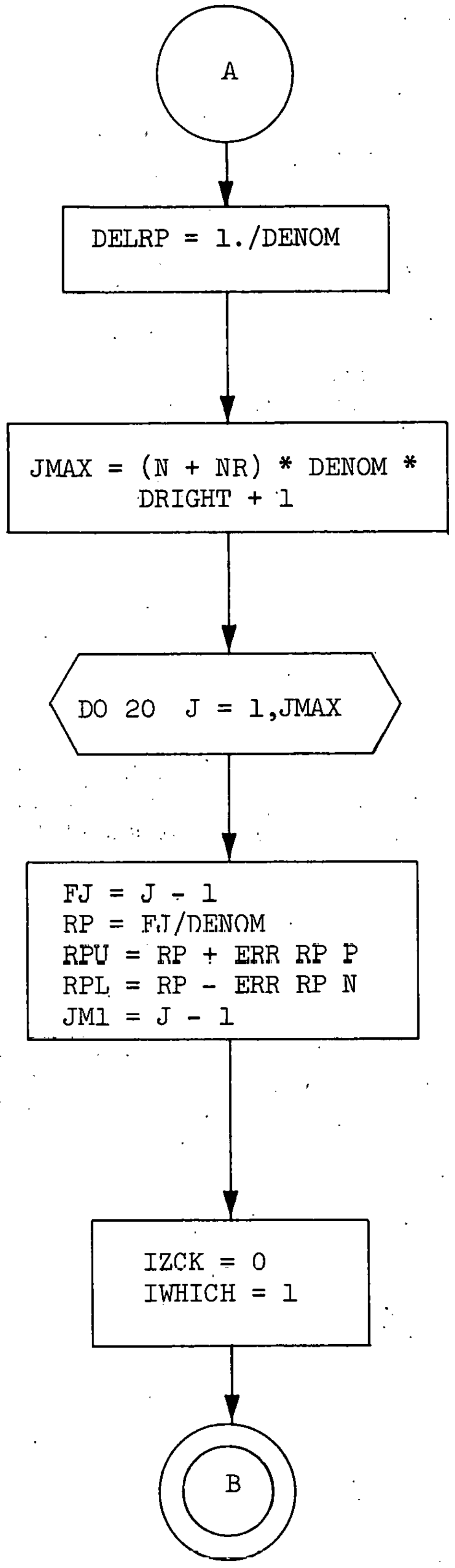
N ≡ n (2.1)

R ≡ r (2.1)

NR ≡ ρ (2.1)

Note: NR is REAL

Figure 8. Flowchart of QS3 Program



DELRP is the increment between successive values of $r\hat{p}$. DENOM is defined by an assignment statement prior to this step.

DELRP $\equiv \Delta rp$

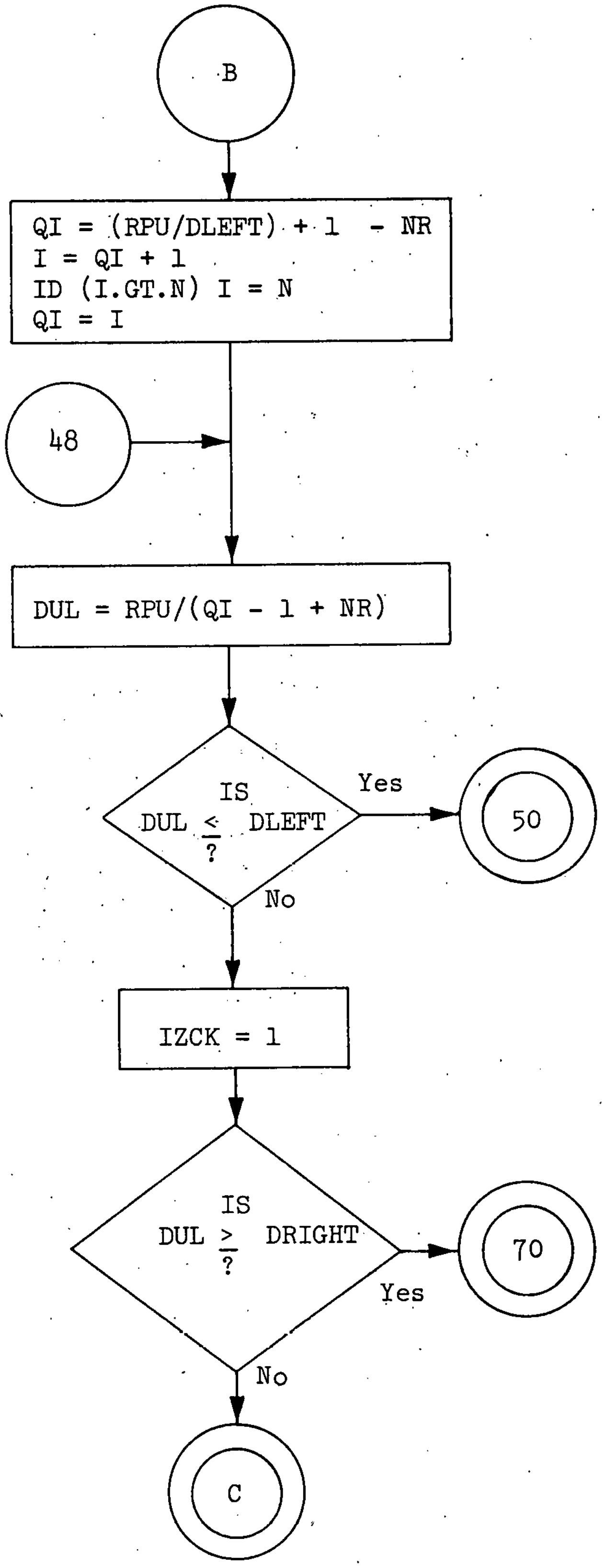
JMAX is the upper limit on the index use to step along the rp-axis.

This is the beginning of the outer loop which steps along the rp-axis.

Compute the present value of RP to be used as $r\hat{p}$ and also the upper (RPU) and lower (RPL) bounds of the rp values represented by $r\hat{p}$.

Initialize two control variables. If IZCK remains at 0 through the inner-loop, which varies the quotient digit, then no divisor transition intervals occur between (a,b). IWHICH = 1 indicates that we are looking for the first divisor transition interval for the present value of $r\hat{p}$. In this case, a $\equiv$DLEFT, will be used as the left end of the segment.

Figure 8 (continued). Flowchart of QS3 Program



QI, the quotient digit value, is initialized at the greatest value such that the part of the line segment formed by RP + FJ/DENOM and the end points of the divisor interval, (a,b), is in the QI-region.

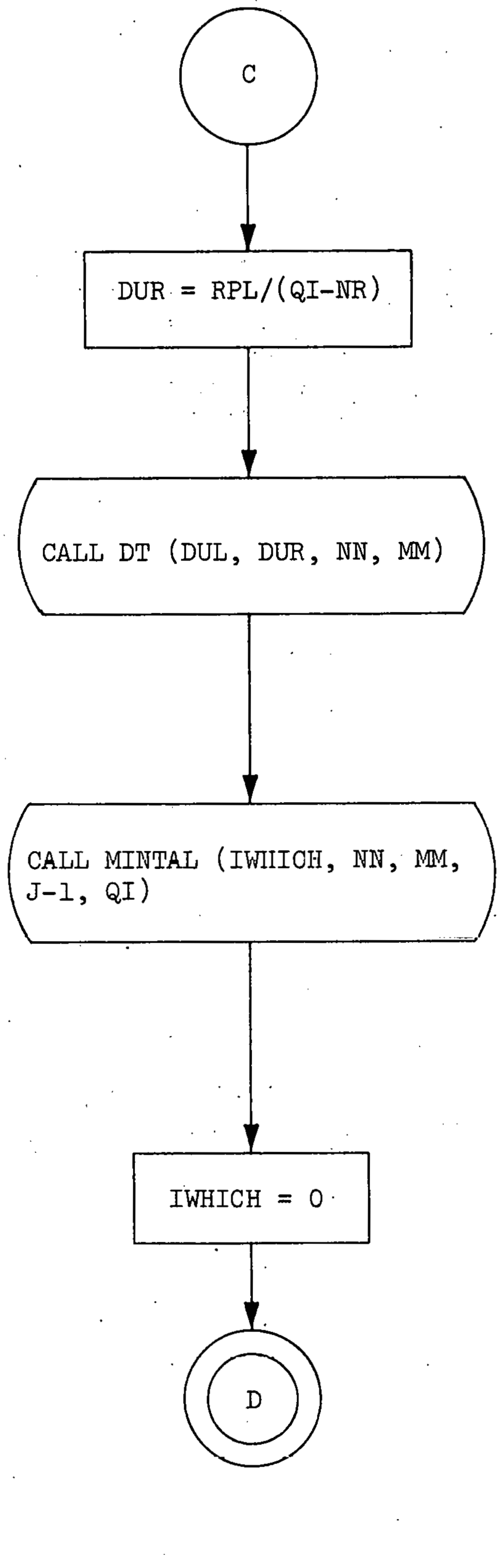
DUL is the left endpoint of the divisor transition interval between QI and QI - 1.

This tests whether or not the transition interval is to the left of the left boundary of the P-D plot. If so, QI is decremented.

A divisor transition interval within (a,b) has been found.

This tests whether or not the divisor transition interval is to the right boundary of the P-D plot. If so, continue with new RP-value.

Figure 8 (continued). Flowchart of QS3 Program


C
DUR = RPL/(QI-NR)
CALL DT (DUL, DUR, NN, MM)
CALL MINTAL (IWHICH, NN, MM, J-1, QI)
IWHICH = 0
D

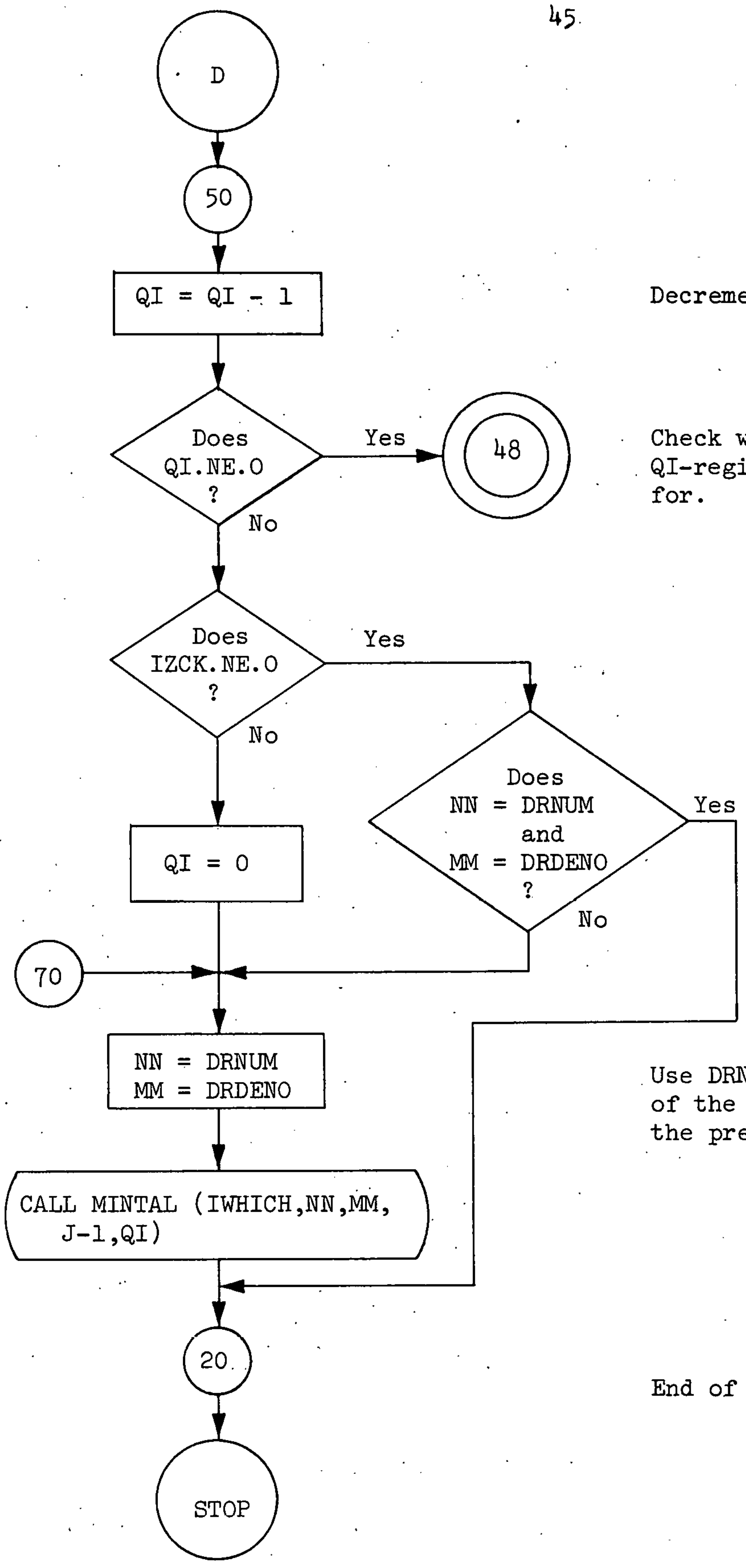
DUR is the right endpoint of the divisor transition interval between QI and QI - 1.

Subroutine DT selects the divisor transition value between DUL and DUR. The value selected is returned in a fractional form (NN/MM). $MM = 2^m$, where m is the smallest integer such that DUL $\leq$ NN/MM $\leq$ DUR.

Subroutine MINTAL creates the minterm definition of the rp-line segments. If IWHICH = 1, then DLNUM/DLDENO is the left end of the segment and NN/MM is the right end. If IWHICH = 0, then the value of NN/MM on the previous call to MINTAL is the left end and the present NN/MM is the right end. J-1 denotes the r$\hat{p}$-line and QI, the quotient region.

Set IWHICH = 0.

Figure 8 (continued). Flowchart of QS3 Program



Decrement QI

Check whether or not all QI-regions have been accounted for.

Use DRNUM/DRDENO as the right end of the last r$\hat{p}$-line segment for the present.

End of DO-Loop which increments rp.

Figure 8 (continued). Flowchart of QS3 Program

4.2.2 Minimizing the Output Functions

A discussion of the minimization of two level switching circuits is beyond the scope of this thesis. However, this section sketches the approach used in this work and references a detailed description of the algorithms. These algorithms are noteworthy due to the fact that they will minimize functions of many variables involving many minterms. In the present work they have been used to minimize functions of 19 variables with over 3100 minterms.

The program QS3 generates a sum-of-products (each product is a minterm) definition of each output function. For each function, the remaining tasks are: 1) to obtain all the prime implicants of the function; and 2) to select a minimal cover which consists of some subset of all prime implicants.

The program used to accomplish step 1 was recently developed by V. G. Tareski [34]. It is an extension of an algorithm developed by Carroll et. al. [35] in late 1968. Tareski has coded his improved version of the algorithm in both PL/1 and Fortran IV on the IBM 360/75.

The output from the program (PI for Prime Implicant) is a list of prime implicants, each in the form:

TTTTTT, where T is

1 if the corresponding variable appears in the true form;

0 if the corresponding variable appears in the complement form ; and

X if the corresponding variable is not present.

Each prime implicant is assigned an identification number. The PI program also partially solves the covering problem in that it identifies all <u>essential</u>

prime implicants. A prime implicant is essential (must be selected for the covering) if it covers a cell in the n-cube representation of the function which is not covered by any other prime implicant.

The program generates a set of constraint equations which must be simultaneously satisfied to guarantee covering. The constraints are specified by a set of equations, each of which is a Boolean sum of prime implicant identification numbers. The identification number is "true" if the prime implicant is selected; false otherwise. For example, two such equations might be

$$2 \vee 5 \vee 7 = '1'$$

$$5 \vee 9 \vee 11 = '1'$$

The set of constraint equations pose a covering problem, i.e. the problem of finding a set of prime implicants which satisfy every equation. The problem is further constrained by the requirement that the sum of the literals of the selected prime implicants be minimal. Fortunately, Liu [36] and Ibaraki et. al. [37] recently developed a very efficient algorithm and computer program which will solve this problem. The program accepts the constraint equations together with the number of the literals in each prime implicant, and produces a minimal cost covering. These prime implicants together with the essential prime implicants found by the PI program constitute the total function. An example is given in Appendix B.

It must be noted that the minimization program is not making explicit use of "don't care" minterms. If ε' is the total number of bits in $\hat{rp}$ and δ' is the total number of bits in $\hat{d}$, then the total number of minterms which can be formed by concatenating $\hat{rp}$ and $\hat{d}$ is $2^{\delta' + \varepsilon'}$. Many of these minterms may not correspond to area within the range of the P-D plot and therefore are don't cares in the sense that they may be arbitrarily added to or deleted from a function depending upon which yields the simplest function. In the cases actually designed, the number of don't cares far exceeds the number of true minterms. For example, with a divisor in the range 1/2 to 1, the number of minterms required to define a P-D plot with $\rho = 2/3$ and a uniform grid of $2^{-\delta} \times 2^{-\varepsilon}$ is $.25\ r\ 2^{\delta+\varepsilon}$, and the number of don't care minterms is $.75\ r\ 2^{\delta+\varepsilon}$. Since in cases studied $\delta+\varepsilon$ may be as great as 14, the don't care minterms would severely tax the minimization routines. They have, therefore, not been included explicitly. The potential effect of the don't cares can be approximated in specific cases considering the following observations:

1. For d in the range $1/2 \leq d < 1$, the don't care minterms corresponding to area of the P-D plot to the left of d = 1/2 would eliminate the $\hat{d}$ bit of weight 1/2 from all output functions of Table 2. The cost in literals, therefore, reduced by the number of prime implicants.

2. If the don't care minterms above the upper boundary of the q (n) region are combined with the true minterms defining q (n), the output function for q (n) is greatly minimized. The cost of q (n) will, therefore, be neglected in estimating the total cost of Table 2.

3. If the don't care minterms above the upper boundary of q (n) region are combined with the true minterms defining q (i), $i \neq n$, then some literals may drop out of the bit string corresponding to the

<u>integer</u> portion of $\hat{rp}$, but none are removed from the fractional part of $\hat{rp}$ or $\hat{d}$. This reduction may be approximated by studying the problem of minimizing a decoder of the integers 0 through n, each of bit length, $\log_2 r$. The minterms n + 1 through r - 1 should be treated as don't cares. It has been estimated that this effect will reduce the total cost of Table 2 by about 15%.

4.3 <u>Deriving a Minimal Cost Design for Table 1</u>

This section describes the algorithms used to synthesize a design for Table 1 of a Type 1 structure. The approach can yield only an estimate of minimal cost since the minimization algorithm is applied to each output function independent of the others. Furthermore it has not been demonstrated that the algorithm used to define the output function necessarily produces a minimal cost design. Despite these shortcomings, the algorithms appear to produce sufficiently accurate results for purposes of cost comparison and for studying trade-offs between the cost of Table 1 and Table 2.

The following is a list of the steps in the process of generating Table 1 and evaluating the cost:

1. Set the values for design parameters = $n, r, a, b, \alpha, \beta, \gamma, \lambda$.
2. Run the program QS4 (described in Section 4.3.1) to produce a sum-of-products (minterm) definition of each output function of Table 1.
3. Run the program PI (Section 4.2.2) with each set of minterms produced by QS4 as input.

4. Run an Integer Linear Programming routine to find a minimal cost set of prime implicants which satisfy the constraints produced in step 3.

5. Tabulate the total number of literals required to define each output function. The total of these values will be taken as the cost of implementing Table 1.

4.3.1 Defining the Output Functions

Generating a quotient digit using a Type 1 structure is accomplished as follows:

1. Given d, form an estimate of d, $\hat{d}$, and from $\hat{d}$ form an estimate of 1/d, A.

2. Form $y = rp \cdot A + 1/2$.

3. Take the integer portion of y as the quotient digit, i. e. $q = I\ (y)$.

The algorithm consists of two steps:

1. For a given Δrp, γ, λ, n, r, α, β, find a $\hat{D}$ such that the selection critereon is satisfied everywhere on the P-D, plot.

2. The $\hat{d}$-values are of the form $j\ \Delta d$, where j is an integer. Each $\hat{d}$ represents a divisor interval d to $d + \Delta d$. For every $\hat{d}$, we must find a value of the function $A\ (\hat{d})$ such that if $(\hat{d}, \hat{rp})$ implies $q = i$, then $I\ (A(\hat{d})\ \hat{rp} + 1/2) = i$.

2

The strictest bounds occur in the vicinity of the transitions between adjacent quotient regions. For a given $\hat{d}$ consider rp lines in the vicinity of the intersection of $\hat{d}$ and the upper boundary of q (i-1) and lower boundary of q (i). See Figure 9.

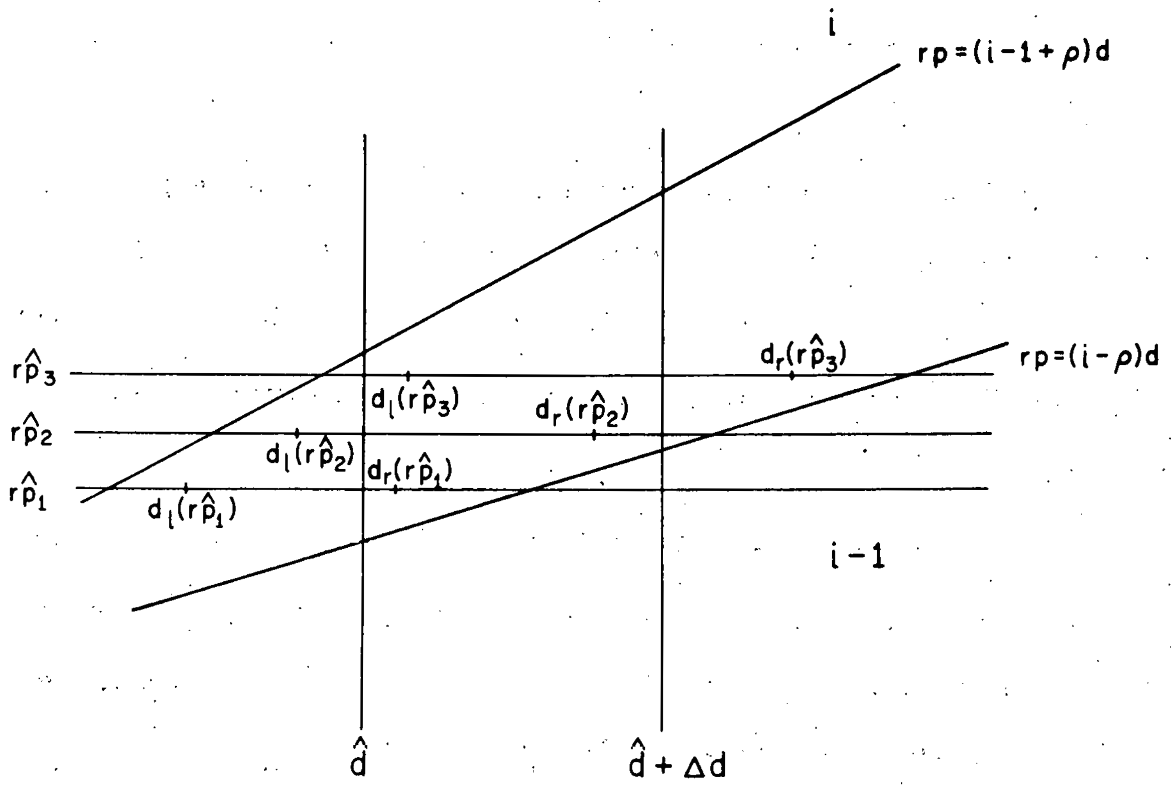


Figure 9. Portion of P-D Plot Illustrating Constraints in Finding $A(\hat{d})$.

Each rp-line has a division transition range between i and i-1 with left end given by

$$d_l(\hat{rp}) = ((\hat{rp}+\gamma)/(i-1-\rho)) + \alpha \tag{4.9}$$

and right end given by

$$d_r(\hat{rp}) = (\hat{rp}-\lambda)/(i-\rho) \tag{4.10}$$

This derivation is given in Section 4.2.1.

$$\text{If } \hat{d} < d_l(\hat{rp}) \tag{4.11}$$

then a quotient digit of i must be selected and thus a value of $A(\hat{d})$ must be found such that $(i-1/2)/\hat{rp} \leq A(\hat{d}) < (i+1/2)/\hat{rp}$. Similarly, if

$$\hat{d} + \Delta d > d_r(\hat{rp}) \tag{4.13}$$

then a quotient digit of i-1 must be selected and thus an estimate must be found such that

$$(i-3/2)/\hat{rp} \leq A(\hat{d}) < (i-1/2)/\hat{rp} \tag{4.14}$$

For a given value of i and $\hat{d}$, find the minimum value of $\hat{rp}$ such that Equation 4.11 is true. Denote this quantity rp_{top}. Also find the maximum value of $\hat{rp}$ such that Equation 4.13 is true. Denote this value $\hat{rp}_{bot}$.

Substituting these quantities into Equations 4.12 and 4.14, respectively, yields

$$(i-1/2)\ \hat{rp}_{top} \leq A(\hat{d}) < (i+1/2)/\hat{rp}_{top} \tag{4.15}$$

$$(i-3/2)\ \hat{rp}_{bot} \leq A(\hat{d}) < (i-1/2)/\hat{rp}_{bot} \tag{4.16}$$

A value of A (d) is needed which satisfies both Equations 4.15 and 4.16. Such a value must be within the range

$$(i-1/2)/\hat{rp}_{top} \leq A(\hat{d}) < (i-1/2)/\hat{rp}_{bot} \tag{4.17}$$

Denote the lower bound of this range, LB(i), and the upper bound, UB(i). Now for all i, find maximum value of LB(i) and designate it LB max. Find minimum UB(i) and designate it UB min. Then select $A(\hat{d})$ such that $LB_{max} = A(\hat{d}) = UB_{min}$ and A(d) is the simplest binary number in the range.

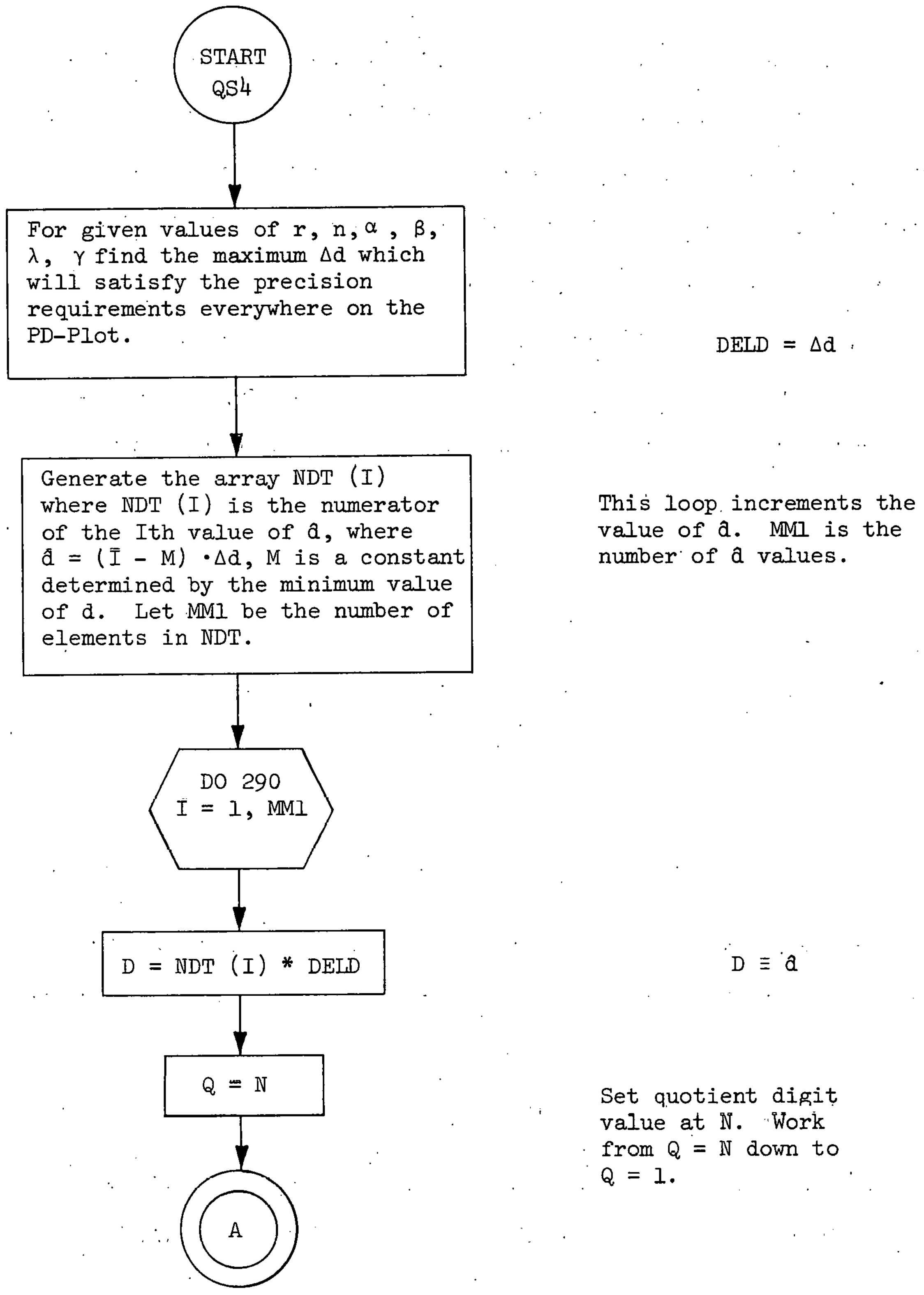
Every value of d is of the form $m\Delta d$ where m is an integer and d is a negative, integer power of 2. The index, m, is therefore a unique, minterm definition of $\hat{d}$. Let $a_{-1}\, a_o\, .\, a_1 \ldots . . a_j$ be a bit string representation of $A(\hat{d})$. Each bit corresponds to a Boolean function of d and thus a Boolean function of m.

$$a_{-1} = g_{-1}\, m$$
$$a_o = g_o\,(m)$$
$$a_1 = g_1\,(m)$$
$$\vdots \quad \vdots$$
$$a_j = g_j\,(m)$$

Each function, g_i, is defined as the OR of all $\hat{d}$-minterms for which a_i is 1 in the bit string version of $A(\hat{d})$. In other words, the set of minterms, M_i, corresponding to g_i is

$$M_i = \{m | a_i \text{ in } A\ (m\Delta d) \text{ is } 1\} .$$

Figure 10 is an annotated flowchart of the program (QS4) which actually produces the definitions of the output functions for Table 1.



START QS4
For given values of r, n, α , β, λ, γ find the maximum Δd which will satisfy the precision requirements everywhere on the PD-Plot.
DELD = Δd
Generate the array NDT (I) where NDT (I) is the numerator of the Ith value of đ, where đ = (Ī - M) ·Δd, M is a constant determined by the minimum value of d. Let MM1 be the number of elements in NDT.
This loop increments the value of đ. MM1 is the number of đ values.
DO 290 Ī = 1, MM1
D = NDT (I) * DELD
D ≡ đ
Q = N
Set quotient digit value at N. Work from Q = N down to Q = 1.
A

Figure 10. Flowchart of QS4 Algorithm

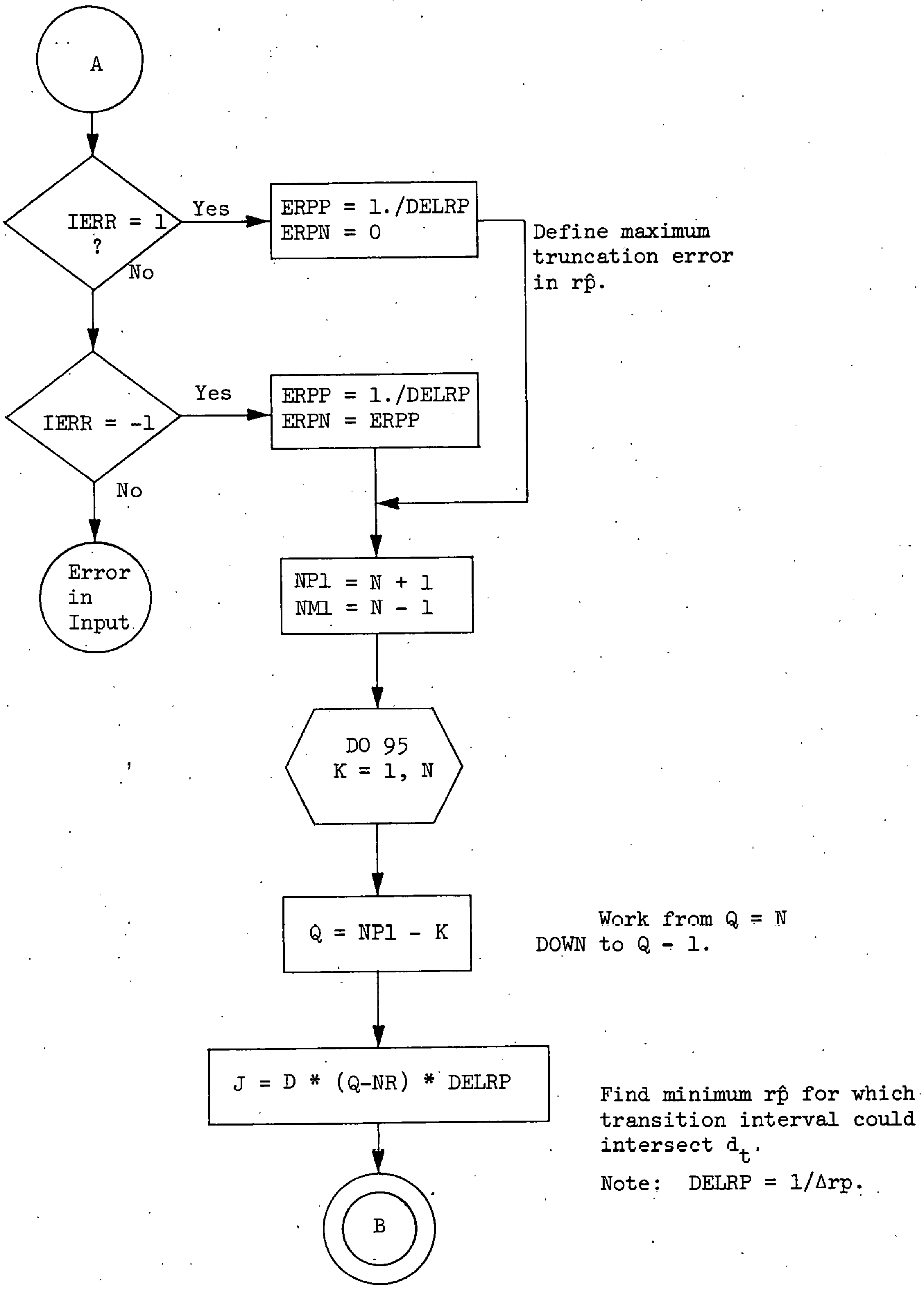
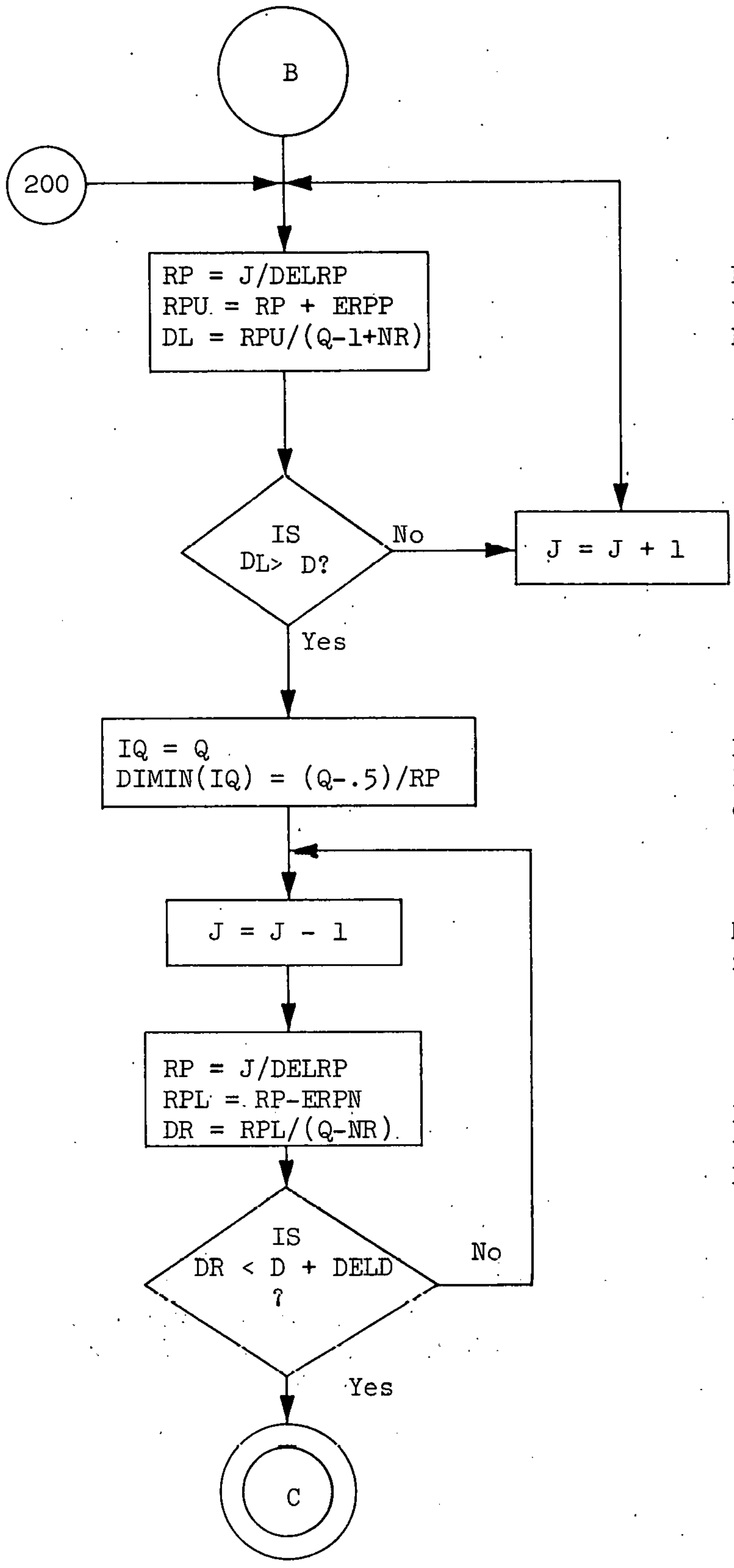

A
IERR = 1 ?
Yes
No
ERPP = 1./DELRP
ERPN = 0
Define maximum truncation error in $r\hat{p}$.
IERR = -1
Yes
No
ERPP = 1./DELRP
ERPN = ERPP
Error in Input
NP1 = N + 1
NM1 = N - 1
DO 95
K = 1, N
Q = NP1 - K
Work from Q = N DOWN to Q - 1.
J = D * (Q-NR) * DELRP
Find minimum $r\hat{p}$ for which transition interval could intersect d_t.
Note: DELRP = 1/Δrp.
B

Figure 10 (continued). Flowchart of QS4 Algorithm



Find left end, DL, of divisor transition interval for present RP.

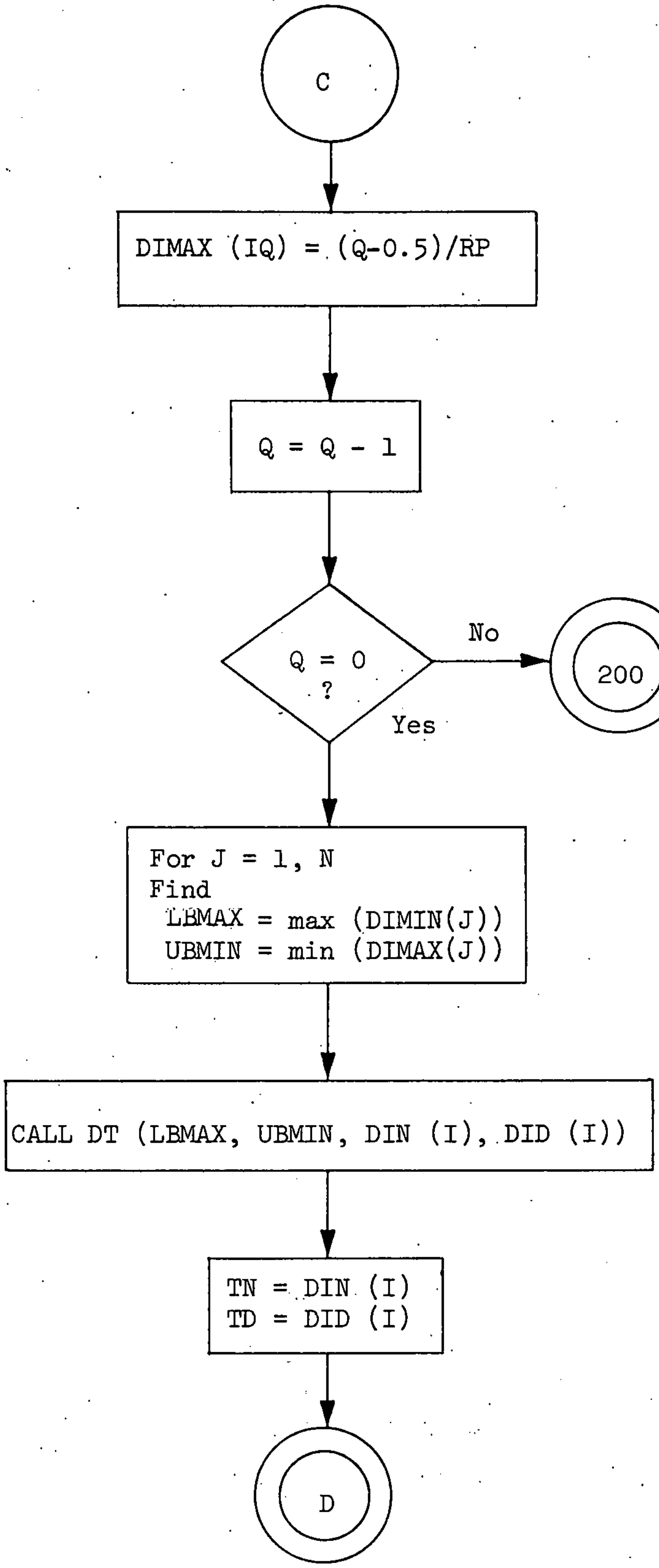
$ERPP = \gamma$
$DL = d_1$
$\alpha = 0$

RPTOP has been found. IQ is an integer version of Q.

Move down to next lower $\hat{rp}$.

Find right end, DR, of divisor transition interval for present RP.

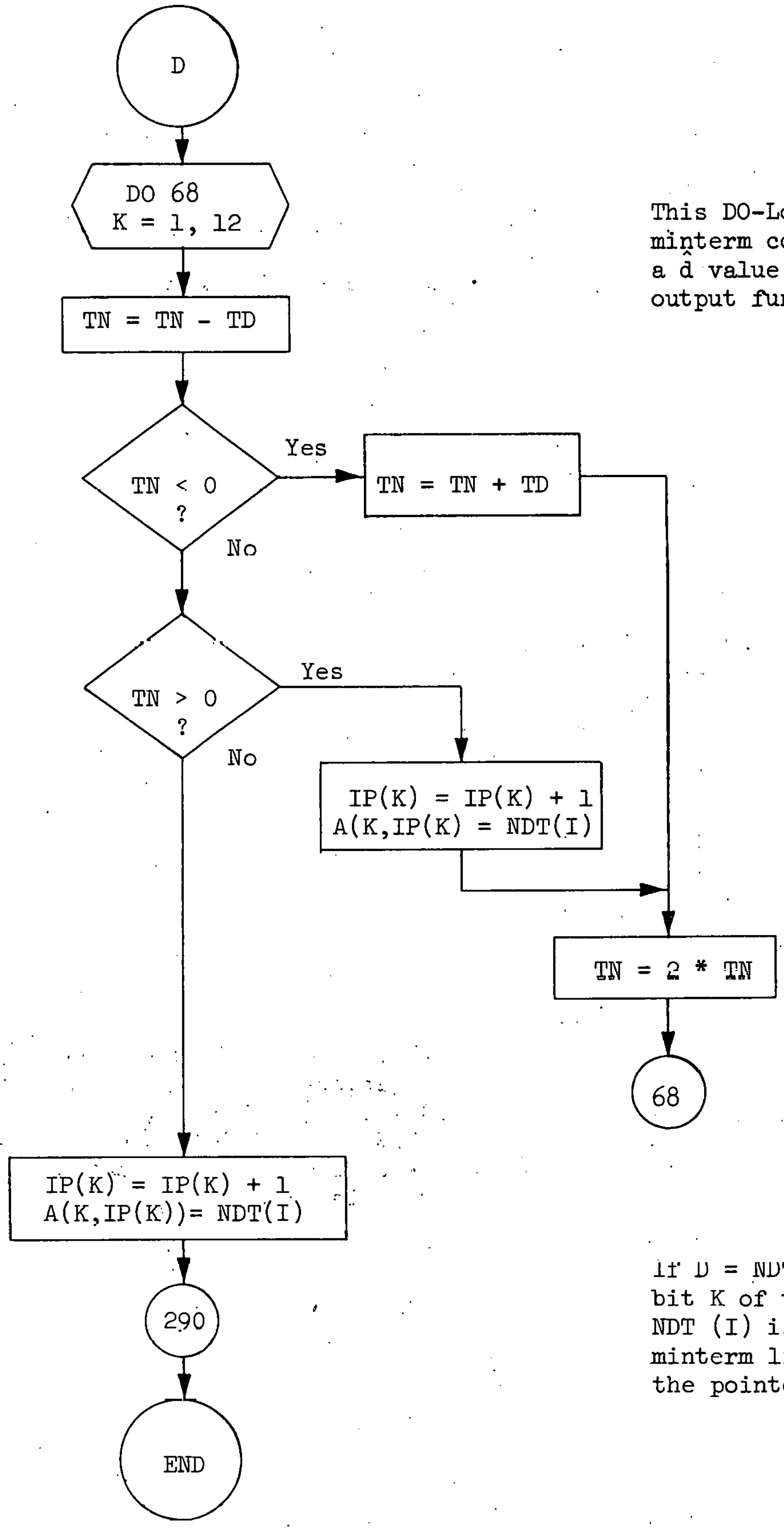
Figure 10 (continued). Flowchart of QS4 Algorithm



RPBOT has been found.

Subroutine DT finds a value for the inverse of D, DI, such that DI = DIN (I)/DID (I), LBMAX $\leq$ DI $\leq$ UBMIN, and DI is the simplest binary fraction in the interval.

Figure 10 (continued). Flowchart of QS4 Algorithm



This DO-Loop assigns each minterm corresponding to a $\hat{d}$ value to the appropriate output functions.

If D = NDT (I) * DELD implies bit K of the output is 1, the NDT (I) is added to the minterm list for A(K). IP(K) is the pointer for the Kth list.

Figure 10 (continued). Flowchart of QS4 Algorithm

4.3.2 Minimizing the Output Functions

The same techniquesused to minimize the output functions of Table 2 are used to minimize the output functions of Table 1. These were described in Section 4.2.2.

5. RESULTS FROM DESIGN PROGRAMS

5.1 Preliminary Remarks

The series of computer runs of the design and analysis routines described in the last chapter gave rise to four types of results. First, the algorithm produced numerical results for the cost of implementing Table 1 or Table 2 for various values of design parameters. But in retrospect it appears that the value of the computer was more insight than numbers. Studying the numerical results gave rise to some theoretical results with which to attack the problem of determining cost without actual design.

A third result was a discrepancy. For some parameter values the theoretical results and the results obtained from the computer-aided synthesis were in disagreement. Closer study revealed a weakness in the QS3 algorithm. The fourth and final result of the work to date was therefore an improved algorithm for designing Table 2.

5.2 Numerical Results from Design Programs

5.2.1 Cost of Table 2 for Type 2 Structure

Considering the large number of possible combinations of parameter values, even if restricted to practical cases, very few designs were actually generated in this present work. After generating the cost data for Table 2 with $r = 16$, $n = 10$, $a = 1/2$, $b = 1$, $\gamma = \lambda = 1/16$, $\alpha = 0$, and $\beta = 1/256$, sufficient insight was gained to propose an analytic expression for the cost of implementing each quotient region of the table. Two additional runs of the Table 2 routines with different parameter values tended to substantiate the predicted costs, but several points stood out as discrepancies. In attempting to reconcile the disagreement, a flaw in the QS3 algorithm was discovered: the selection of divisor transition values as the simplest binary number in the transition interval does <u>not</u> necessarily produce a minimum cost design. In view of this flaw, further runs of the algorithm were not justified. The major emphasis was shifted to that of developing a reasonable derivation of an analytic cost expression and to developing an algorithm which would in fact yield correct results which could be used to verify the expression. The parameter values selected correspond to practical cases. Let r, denote the radix and assume a multiplication structure in which the following multiples of the multipland are available: ± 1 or ± 2, ± 4 or ± 8, ± 16 or ± 32, ... , $\pm (r-2)$ or $\pm (r-1)$. Each of the groups such as ± 4 or ± 8, correspond to a two-way shift gate. Only one of the two multiples may be selected simultaneously. The magnitude of the maximum multiple which may be formed, n, is therefore $2 + 8 + 32 + \dots + (r - 1) = 2 (r - 1)/3$. Since the same structure is used for division, the maximum quotient digit is also n and therefore in the cases studied, $n = 2 (r - 1)/3$ and thus the redundancy ratio, ρ, is 2/3.

As mentioned earlier, the study was restricted to the first quadrant of the P-D plot. The divisor ranges considered were the binary normalized case in which $1/2 \leq d < 1$, and a second case for which $3/4 \leq d < 9/8$. This second case corresponds to a case in which a divisor in the range $1/2 \leq d < 1$ is multiplied by 3/2, if $d < 3/4$.

The maximum truncation errors in $r\hat{p}$, γ and λ, are initially set to the maximum value for which the criterion in Section 2.3 is satisfied, 1/16. Error was assumed in both directions so that the results would be applicable to symmetric adders or subtracters [10].

The divisor is strictly positive and non-redundantly represented thus $\alpha = 0$. The positive truncation error was the maximum necessary to satisfy the selection criterion (Section 2.3) everywhere on the P-D plot for the given value of γ and λ.

Table 2 summarizes the cost computations for a Table 2 structure with $r = 16$, $n = 10$, $a = 1/2$, $b = 1$, $\gamma = 1/16$, $\lambda = 1/16$, $\alpha = 0$, and $\beta = 1/256$. Radix 16 was selected as sufficiently large to be interesting but not so large as to demand great expense of computer time. Table 4 presents corresponding results for divisors in the range $3/4 \leq d < 9/8$. No cost values are given for the upper quotient region, q (n). These regions were not minimized since the results would be highly inaccurate without the ability to include don't care minterms. The upper boundary of q (n) need not be implemented since the range restrictions imposed by the division algorithm would prohibit $(\hat{d}, \hat{rp})$ values to occur above the q (n) region. All minterms corresponding to points above the line $rp = (n + \rho)\, d$ are therefore don't care minterms which sharply minimize the cost of implementing the adjacent q (n) region.

Note that the cost of a Table 2 structure for $r = 4$, $n = 2$ is also contained within Table 2. Neglecting the upper region q(2) the cost is the cost of q (0) + q (1) for radix 16 less 2 literals per required prime implicant.

Table 2. Summary of Cost Calculations for Table 2 with

$r = 16,\ n = 10,\ a = 1/2,\ b = 1,\ \gamma = 1/16,$

$\lambda = 1/16,\ \alpha = 0,\ \beta = 1/256.$

q-Region	No. of Bits in $r\hat{p}$	Min. No. of Bits Required in $\hat{d}$ to Define the Region	Min. No. of Min-terms to Define the Region		Min. No. of Prime Impli-cants to Define Region $M'(i)$	No. of Literals to Define Region		$C'(i)$	Average Fan-in $F'(i)$
			Est.	Act.		$r\hat{p}$	$\hat{d}$	Total	
0	8	2	12	12	4	25	6	31	7.75
1	8	4	96	99	13	82	27	109	8.38
2	8	5	192	195	21	138	62	200	9.52
3	8	6	384	384	36	236	129	365	10.14
4	8	6	384	385	45	296	190	486	10.80
5	8	7	768	765	60	389	269	658	10.96
6	8	7	768	774	72	464	334	798	11.08
7	8	7	768	764	84	541	424	965	11.49
8	8	7	768	771	96	627	507	1134	11.81
9	8	8	1536	1526	109	711	584	1295	11.88
				Totals	540	3509	2532	6041	

Table 3. Summary of Cost Calculations for Table 2

with $r = 16$, $n = 10$, $a = 3/4$, $b = 9/8$, $\gamma = 1/16$,

$\lambda = 1/16$, $\alpha = 0$, $\beta = 1/128$.

q-Region	No. of Bits in rp	Min. No. of Bits Required in $\hat{d}$ to Define the Region	Min. No. of Min-terms to Define the Region		Min. No. of Prime Implicants to Define Region $M'(i)$	No. of Literals to Define Region		$C'(i)$	Average Fan-in $F'(i)$
			Est.	Act.		$r\hat{p}$	$\hat{d}$	Total	
0	8	4	24	24	2	10	7	17	8.50
1	8	4	45	44	7	40	26	66	9.43
2	8	5	90	91	14	84	57	141	10.07
3	8	6	180	180	23	139	101	240	10.43
4	8	6	180	181	27	166	127	293	10.85
5	8	7	360	359	32	199	160	359	11.22
6	8	7	360	362	40	246	212	458	11.45
7	8	7	360	358	54	332	301	633	11.72
8	8	7	360	363	54	339	305	644	11.93
9	8	7	360	358	61	383	351	734	12.03
				Totals	314	1938	1647	3585	

5.2.2 Cost of Table 1 for Type 1 Structure

The design of Table 1 is considerably less complicated than that of Table 2 since it is a function of only one input rather than two. The costs for radix 4, 16, and 64 were generated and summarized in Table 4. The complexity of the table is adequate to produce a quotient digit in the leading bits of the product $A \cdot r\hat{p}$, where $A = f(\hat{d})$ and is of the form $a_{-1}\, a_0 \,.\, a_1 \ldots a_j$.

Table 4. Summary of Cost Calculations for Table 1 with

$a = 1/2,\ b = 1,\ \gamma = 1/16,\ \lambda = 1/16,\ \alpha = 0.$

Note: NPI = Minimum Number of Prime Implicants
NL = Minimum Number of Literals

Output Bit	$r = 4,\ n = 2$ $\beta = 1/16$		$r = 16,\ n = 10$ $\beta = 1/256$		$r = 64,\ n = 42$ $\beta = 1/1024$	
	NPI	NL	NPI	NL	NPI	NL
a_0	1	1	1	1	1	1
a_1	2	4	3	8	4	13
a_2	3	7	8	29	9	38
a_3	3	7	12	56	18	91
a_4			16	79	28	153
a_5			19	95	41	239
a_6			18	109	63	401
a_7					80	554
a_8					80	579
a_9					9	70
Totals	9	19	77	377	333	2139

5.3 Analytic Results Concerning Cost of Table 2

5.3.1 Preliminary Remarks

Figure 11 is a plot of cost in literals of implementing q(i) versus i for results given in Table 2. To a first approximation the cost varies linearly with i. This observation led to a comparison of the empirical results with the theoretical, indirect measure of the cost of selection of quotient digits suggested by Robertson [5]. This cost function also exhibits a similar behavior with i. In the following we will review aspects of Robertson's work, suggest extensions and then propose an expression for the cost of implementing Table 2 as a function of design parameters.

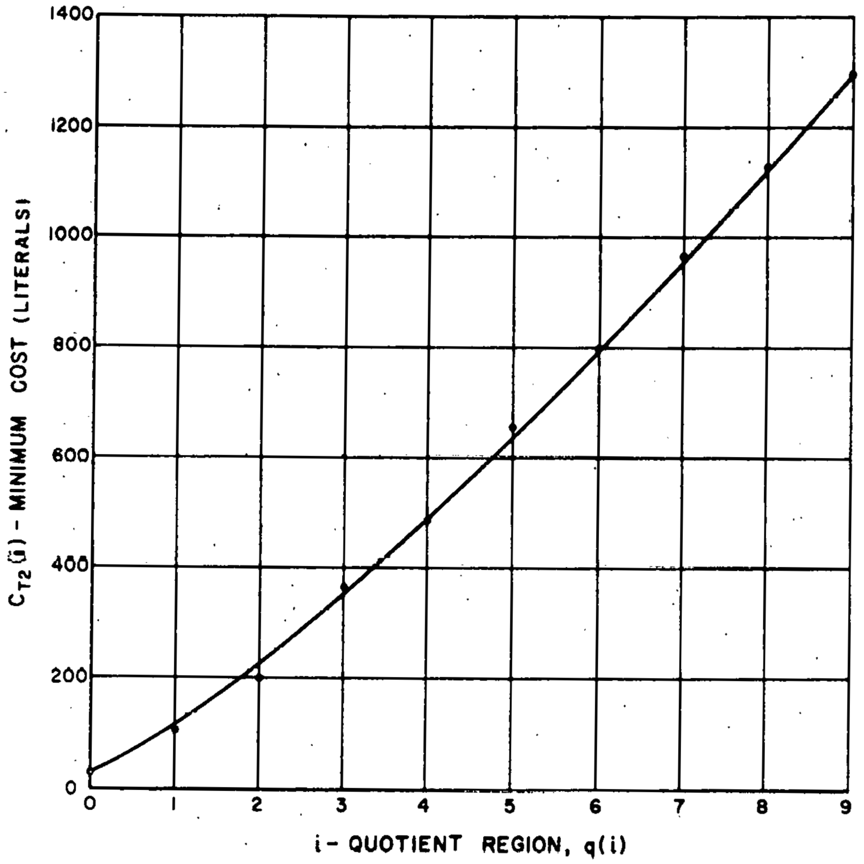


Figure 11. Cost of Implementing q(i)-Region vs. i for Data in Table 2.

5.3.2 Definition of s_i, s_i', and s_i''

In Robertson's work the design problem is presented as that of choosing comparison constants against which rp_j is compared and of determining the divisor range for which each comparison constant is valid. The proposed measure of cost of selecting between q(i) and q(i-1) is the minimum number of comparison constants required to cover the given range of the divisor.

The selection ratio, σ_i, is first defined. It is the ratio of the slope of the line defining the lower boundary of q(i) to the slope of the line defining the upper boundary of q(i-1), i.e.,

$$\sigma_i = \frac{i - \rho}{i - 1 + \rho}$$

The selection ratio is a relative measure of the width of the divisor interval for which a single comparison constant is valid. The minimum number of divisor intervals required to correctly distinguish between q = i and q = i-1 corresponds to the number of treads in the staircase between the upper boundary of q(i-1) and lower boundary of q(i).

Let s_i denote the minimum number of steps required to span the overlap region between q(i) and q(i-1) for the divisor range a to be as shown in Figure 12. The slope of the upper boundary is $v = i-1+\rho$ and the slope of the lower boundary is $w = i-\rho$. Let Δ_1 be the width of the rightmost tread, Δ_2 be the width of the second tread (moving from right to left), etc. The quantity, h, is the height of the riser between tread 1 and tread 2.

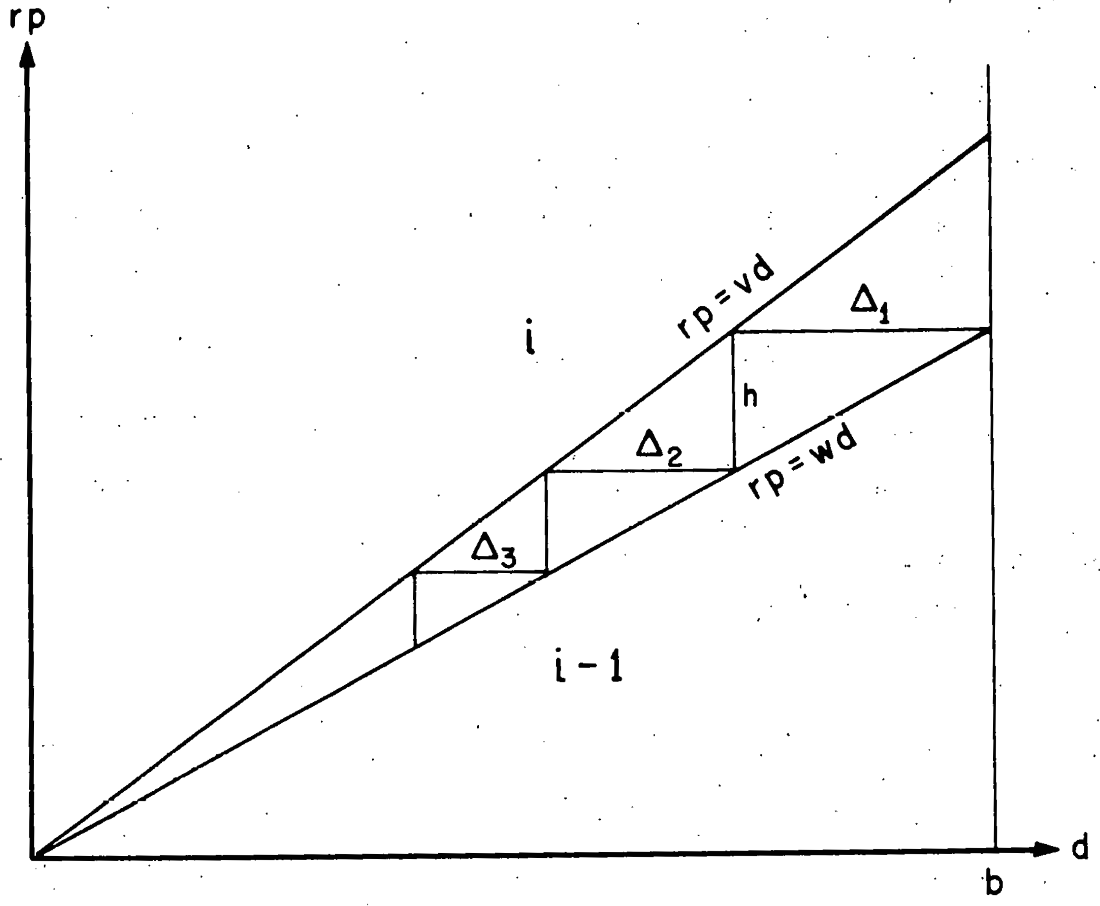
By definition

$$v = h/\Delta_2 \,, \tag{5.1}$$

$$w = h/\Delta_1 \,, \tag{5.2}$$

and thus,

$$\Delta_2 = \sigma_i \, \Delta_1 . \tag{5.3}$$



$v = i - 1 + \rho$ $w = i - \rho$

Figure 12. Graphical Interpretation of s_i.

In general,

$$\Delta_k = \sigma_i \Delta_k = \sigma_i^{(k-1)} \Delta_1 . \tag{5.4}$$

By definition

$$\sum_{k=1}^{s_i} \sigma_i^{(k-1)} \Delta_1 = b - a . \tag{5.5}$$

The left side of Equation 5.5 is the sum of a geometric series and thus

$$\Delta_1 \frac{\sigma_i^{s_i} - 1}{\sigma_i - 1} = b - a . \tag{5.6}$$

Since $\Delta_1 = b(1-\sigma_i)$, s_i is the smallest integer that satisfies

$$\sigma_i^{s_i} \leq a/b \; . \tag{5.7}$$

For present purposes, consider s_i to be a continuous variable, rather than an integer. Then,

$$s_i = \log(a/b) \,/\, \log \sigma_i \; . \tag{5.8}$$

We will now change the expression for s_i into a form which makes apparent the linear behavior with i. By the properties of logarithms

$$\begin{aligned} \log(\sigma_i) &= \log(i-\rho) - \log(i-1+\rho) \\ &= \log(1+x) - \log(1-x) \\ \text{where} \quad x &= (1-2\rho) \,/\, (2i-1). \end{aligned} \tag{5.9}$$

With ρ restricted to the range $1/2 \leq \rho < 1$, then $-1 < x < 1$ and thus a series form of $\log(1+x) - \log(1-x)$ may be used. Therefore,

$$\begin{aligned} \log \sigma_i &= 2\left[x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots + \frac{x^{2m-1}}{2m-1} + \cdots \right] \\ &= 2x + \text{h.o.t.}, \end{aligned} \tag{5.10}$$

and thus,

$$s_i \simeq \log(b/a) \;\; (i-1/2) \,/\, (2\rho-1). \tag{5.11}$$

The quantity, s_i, as defined so far is based upon the assumption of full precision in the representation of the divisor and partial remainder. The expression for s_i will now be modified to yield the minimum number of steps required to transerve the transition region between q(i) and q(i-1) when only estimates of rp and d are available, $\hat{rp}$ and $\hat{d}$ respectively. Assume as before that $\hat{rp}$ is representative of rp-values in a range given by $\hat{rp} - \lambda \leq rp \leq \hat{rp}+\gamma$, and that $\hat{d}$ is representative of d-values in the range $\hat{d} - \alpha \leq d \leq \hat{d} + \beta$. For the time being, assume that $\hat{rp}$ and $\hat{d}$ may assume any value, not merely discrete values.

If we consider the staircase to be the upper boundary of the $(\hat{d}, \hat{rp})$ values defining the q(i-1) region, then for all pairs, $(\hat{d}_{i-1}, \hat{rp}_{i-1})$, defining the risers and treads, the restriction

$$\hat{rp}_{i-1} + \gamma \leq v(\hat{d}_{i-1} - \alpha) \tag{5.12}$$

must hold. Thinking of the staircase as the lower boundary of values defining the q(i) region, then for all pairs $(\hat{d}_i, \hat{rp}_i)$ defining the risers and treads, the restriction

$$\hat{rp}_i - \lambda \geq w(\hat{d}_i + \beta) \tag{5.13}$$

must hold.

Since adjacent values of $\hat{rp}$ are separated by Δrp and adjacent values of $\hat{d}$ are separated by Δd,

$$\hat{d}_i = \hat{d}_{i-1} - \Delta d, \text{ and} \tag{5.14}$$

$$\hat{rp}_i = \hat{rp}_{i-1} + \Delta rp. \tag{5.15}$$

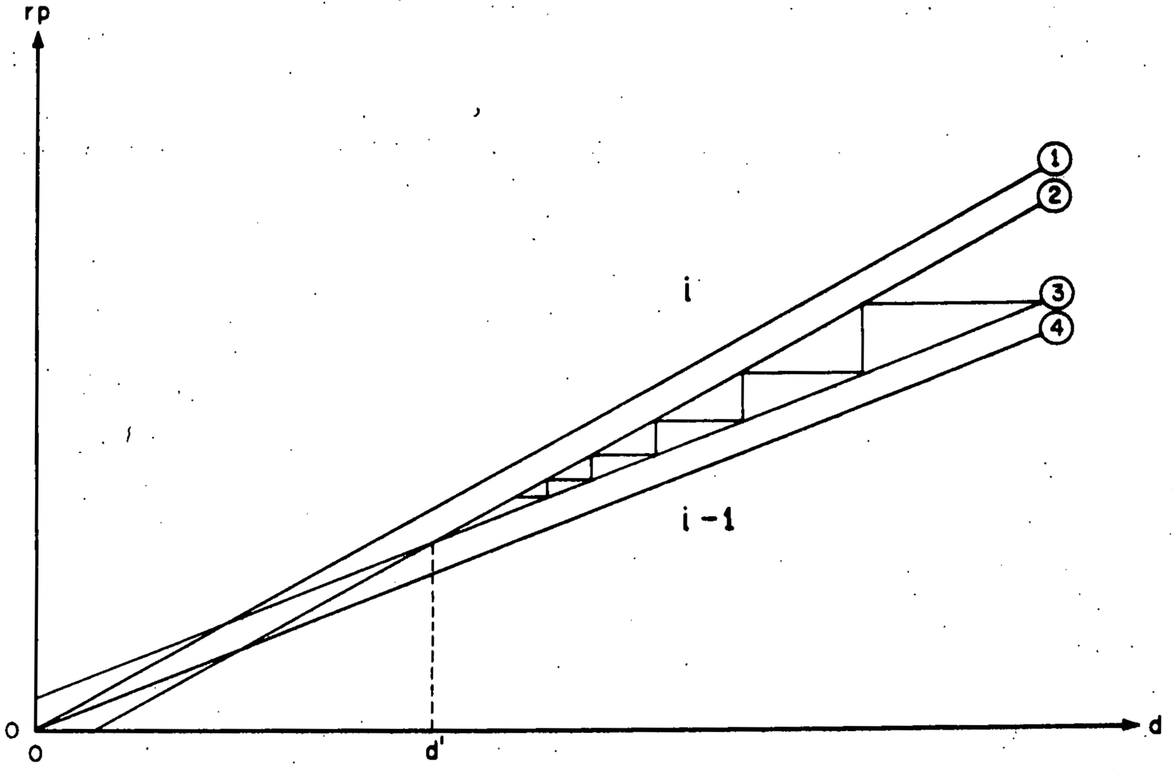
The staircase must satisfy both restrictions 5.12 and 5.13 subject to equations 5.14 and 5.15. Substituting Equations 5.14 and 5.15 into 5.13 yields another restriction in terms of $\hat{rp}_{i-1}$ and d_{i-1}, namely

$$\hat{rp}_{i-1} + \Delta rp - \lambda \geq w(\hat{d}_{i-1} - \Delta d + \beta). \tag{5.16}$$

For a given value of rp_{i-1} the maximum tread width is therefore the distance between the intersection of the line $rp = rp_{i-1}$ and the lines

$$rp = w(d - \Delta d + \beta) + \lambda - \Delta rp, \text{ and} \tag{5.17}$$

$$rp = v(d - \alpha) - \gamma. \tag{5.18}$$



$v = i - 1 + \rho \qquad w = i - \rho$

① $rp = vd$ ③ $rp = w(d - \Delta d + \beta) + \lambda - \Delta rp$

② $rp = v(d - \alpha) - \gamma$ ④ $rp = wd$

Figure 13. Graphical Interpretation of s_i'.

Figure 13 is a graphical interpretation of the minimum step boundary between q(i) and q(i-1) for this non-precise case.

The effect of the imprecision on s_i may be thought of as shifting the divisor range of the P-D plot by an amount, d' given by

$$d' = \frac{\lambda + \gamma - \Delta rp + v\alpha + w(\beta - \Delta d)}{2\rho - 1} . \tag{5.19}$$

The value of s_i in this case, denoted s_i', is given by

$$s_i' = \frac{\log(\ (a - d') / (b - d')\ )}{\log \sigma_i} . \tag{5.20}$$

Note that this equation is equivalent to replacing a by a-d' and b by b-d' in Equation 5.8. This may be verified by replacing Δ_1 in Equation 5.6 by the appropriate expression in the present case, namely by

$$\Delta_1 = b - \frac{w(b + \beta - \Delta d) - (\gamma + \lambda - \Delta rp) + \alpha}{v} . \tag{5.21}$$

Geometrically, d' is the value of d at the intersection of the lines defined by Equations 5.17 and 5.18.

Equation 5.19 implies that it is not merely the imprecision but rather the redundancy in the representation of $\hat{rp}$ and $\hat{d}$ which increases the number of treads in the boundary staircase. First, note that to insure covering, i.e. that every value of rp and d map into at least one $\hat{rp}$ and $\hat{d}$, respectively, the inequalities

$$\lambda + \gamma - \Delta rp \geqq 0, \text{ and} \tag{5.22}$$

$$\alpha + \beta - \Delta d \geqq 0 \tag{5.23}$$

must hold. This restriction forbids d' from being negative and thus s_i' being less than s_i. If $\lambda + \gamma - \Delta rp = 0$, $\alpha = 0$, and $\beta - \Delta d = 0$, then $s_i' = s_i$. This corresponds to the case in which every rp and d value map into one and only one $\hat{rp}$ and $\hat{d}$, respectively.

In terms of the P-D plot this means that there is no overlap between the area represented by the pairs $(\hat{d}, \hat{rp})$. In this case, even though $\hat{rp} \neq rp$, and $\hat{d} \neq d$, the selection is theoretically no more complicated than in the full precision case.

For the cases treated in this study $\lambda = \lambda'\Delta rp$, $\gamma = \gamma'\Delta rp$, $\alpha = 0$, $\beta = \Delta d$, and thus

$$d' = \frac{\Delta rp\,(\lambda' + \gamma' - 1)}{2\rho - 1} . \tag{5.24}$$

The analysis so far has allowed for an error in representing d and rp but has not restricted the value of $\hat{d}$ and $\hat{rp}$. In practice these are formed by truncation and therefore are restricted to integral multiples of $\Delta d = 2^{-\delta}$ and $\Delta rp = 2^{-\varepsilon}$ where δ and ε are the number of bits to the right of the binary point in the representation of $\hat{d}$ and $\hat{rp}$ respectively. The location of the treads and risers of the actual staircase which can be implemented may therefore simultaneously differ by as much as Δd and Δrp, respectively. The maximum number of steps (taking into account both error and discrete effects) required to define the boundary between q(i) and q(i-1) may therefore be given by

$$s_i'' = \frac{\log(\ (a - d'') \ / \ (b - d'')\ )}{\log \sigma_i} \tag{5.25}$$

where

$$d'' = \frac{\lambda + \gamma - \Delta rp + 2^{-\varepsilon} + v(\alpha + 2^{-\delta}) + w(\beta - \Delta d)}{2\rho - 1} \quad . \tag{5.26}$$

The actual number of steps required, $s_{i\ act}$ is therefore bounded by

$$s_i' \leq s_{i\ act} \leq s_i'' \quad . \tag{5.27}$$

Equation 5.26 may be used to determine the minimum values of ε and δ required for a given P-D plot. The quantity, s_i'', and thus the cost, will tend to infinity as d" approaches a. To insure that every region of the P-D plot may be correctly defined for given values of λ, γ, α, β, the quantities ε and δ therefore must be selected such that $d'' < a$.

5.3.3 An Estimate of Cost as a Function of s_i'

In this section we will hypothesize an expression for the cost of implementing the q(i) region of a given P-D plot. Consider the region to be defined by a set of minterms corresponding to the set of ordered pairs $(\hat{d}, \hat{rp})$

for which q = i. Let Δd for the region be $2^{-\delta(i)}$ and Δrp for the region be $2^{-\varepsilon(i)}$. The number of minterms to define the region will be

$$M(i) = (b^2 - a^2)\ 2^{\varepsilon(i) + \delta(i) - 1} . \tag{5.28}$$

The fan-in to each minterm, F(i) is given by

$$F(i) = \varepsilon' + \delta' \tag{5.29}$$

where

$$\varepsilon' = \log_2 r + \varepsilon(i), \text{ and} \tag{5.30}$$

$$\delta' = I(\log_2 (b - 2^{-\delta(i)}) + 1) + \delta(i). \tag{5.31}$$

The term $I\ (\log_2(b-2^{-\delta}) + 1)$ is merely the number of bits of the divisor to the left of the radix point. Recall that I(x) has been defined as the integer portion of x.

The cost before minimization is given by

$$C_T(i) = M(i)\ F(i) + M(i) \tag{5.32}$$

The term MF is the number of literals in the AND gates, the term M is the number of literals in the OR gate.

After minimization

$$C_T'(i) = M'(i)\ (F'(i) + 1) \tag{5.33}$$

where M'(i) is the number of prime implicants and F'(i) is the average fanin to each prime implicant.

In order to obtain approximations of M'(i) and F'(i), we now approximate the effects of minimization by the following algorithm.

Figure 14 illustrates a portion of a quotient region. Note that it may be defined by a set of adjacent rectangles (denoted by heavy lines) each of which is defined by a set of minterms (denoted by small squares). Consider one of the rectangles of width W and height H. Assume that minimization

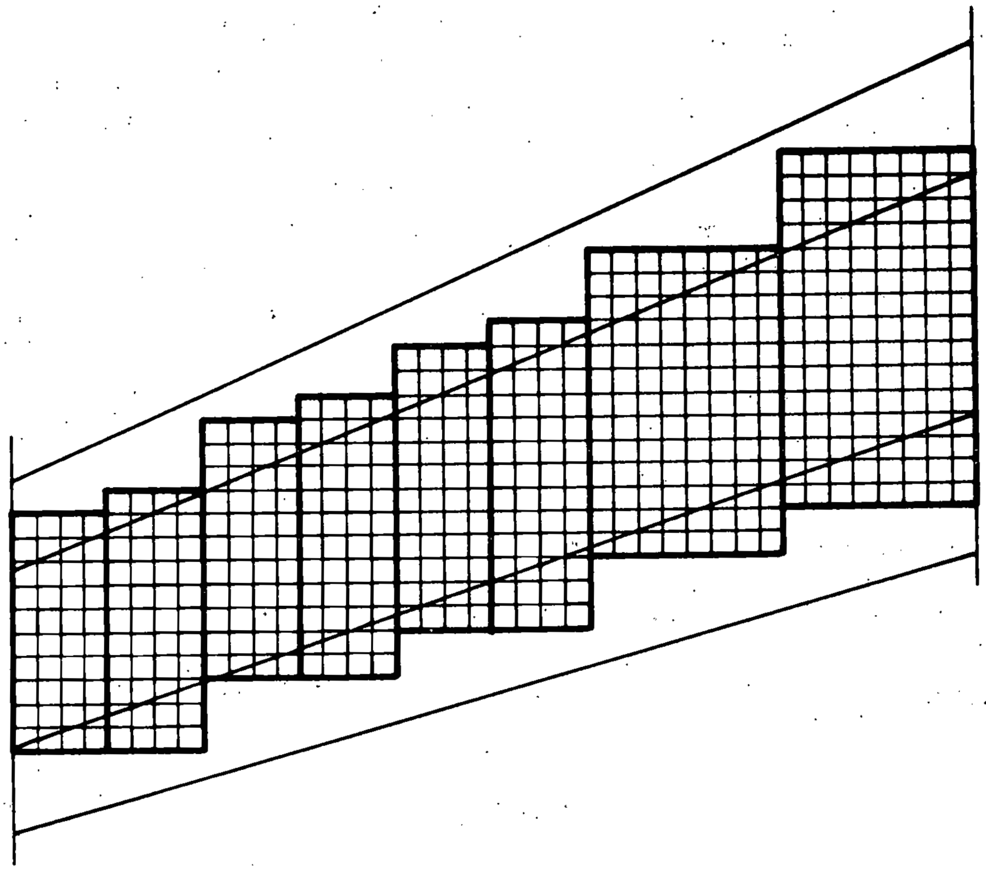


Figure 14. Model of the q(i) Region Used in Approximating Effects of Minimization

procedes first in the d-direction by combining adjacent minterms which differ by only the low order bit. If there were initially M minterms in the rectangle, after the first step there are M/2 implicants. Next, the implicants which differ only in the next to low-order position may combine to produce M/4 implicants, etc. The minimization in the d-direction continues for $k_d = I\ (\log_2 W)$ steps to form $M/2^{k_d}$ implicants. Similarly, combinations take place in the rp-direction, further reducing the number implicants to $M/2^{k_d + k_{rp}}$ where $k_{rp} = I\ (\log_2 H)$.

The minimization of the quotient region will be characterized by an average rectangle of dimensions W H. The width is defined by

$$W = 2^{\delta}(b - a) \,/\, \overline{s_i'} \tag{5.34}$$

where,

$$\overline{s_i'} = (s_i' + s_{i+1}') \,/\, 2. \tag{5.35}$$

The quantity, W, is therefore the average width of the minimum-number treads defining the upper and lower boundary of q(i). The height is defined by

$$H = 2^{\varepsilon}(b+a) \,/\, 4 \tag{5.36}$$

which is the average value of the distance between rp = (i + 1/2) d (nominal upper boundary) and rp = (i - 1/2) d (nominal lower boundary).

The preceeding argument suggest a cost expression of the following form:*

$$C'(i) = \phi_1 \frac{M(i)}{2^k} (F(i) - k + \phi_2) \tag{5.37}$$

where

M and F are defined by Equations 5.28 and 5.29, respectively, and k is defined by

$$\begin{aligned} k &= k_d + k_{rp} \\ &= \log_2 WH \end{aligned} \tag{5.38}$$

The factors ϕ_1 and ϕ_2 are constants which will be determined empirically. Equation 5.37 may be rewritten as

$$C'(i) = M'(i)\ F'(i) \tag{5.39}$$

where

$$M'(i) = 2\ \phi_1\ \overline{s_i'}\ \text{, and} \tag{5.40}$$

*Note that C'(i) is the number of literals in the AND gates; $C_T'(i) = C'(i) + M'(i)$ is total number of literals for the region.

$$F'(i)=\log_2 \frac{4\,\phi_2\, r\, \overline{s_i'}}{b^2-a^2} + I\,(\log_2 (b - 2^{-\delta}) + 1)\,. \qquad (5.41)$$

$M'(i)$ is the minimal number of prime implicants required to implement the Boolean function for $q(i)$ and $F'(i)$ is the average fanin to each prime implicant.

We now use numerical results from Tables 2 and 3 to find values for ϕ_1 and ϕ_2 and to test the predictive worth of Equation 5.39. The value of ϕ_1 is obtained by a least squares fit of the actual values of $M'(i)$ to Equation 5.40. The value of ϕ_2 is obtained by a least squares fit of the actual values of $F'(i)$ to Equation 5.41. Values of $\phi_1 = 2.12$ and $\phi_2 = 1.68$ were obtained.

Table 5 summarizes the results of the fit. Figures 15, 16, and 17 display the results graphically with $\overline{s_i'}$ as the independent variable. The heavy line denotes the predicted values; the circles denote actual values.

Note that Equations 5.40 and 5.41 do not explicitly account for the discrete effects resulting from the fact that the treads and risers of the q-region boundaries are restricted to integer multiples of $2^{-\varepsilon}$ and $2^{-\delta}$, respectively. The effect is included empirically in the choice of ϕ_1 and ϕ_2. There are indications that a more explicit cost function of both s_i' and s_i'', which does include discrete effects, might be found. For present purposes, however, the estimates given by Equations 5.40 and 5.41 were judged to be adequate.

Table 5. Results of Least Squares Fit of M'(i), F'(i), and C'(i) for Data from Table 2.

$a = 1/2$, $b = 1$

i	$\overline{s'_i}$	M'(i)		F'(i)		C'(i)	
		Equation	QS3	Equation	QS3	Equation	QS3
0	1.38	5	4	7.6	7.7	44	31
1	2.83	12	13	8.6	8.4	103	109
2	5.72	24	21	9.6	9.5	234	200
3	8.59	36	36	10.2	10.1	373	365
4	11.46	48	45	10.6	10.8	519	486
5	14.33	60	60	11.0	11.0	668	658
6	17.20	72	72	11.2	11.0	821	798
7	20.06	85	84	11.4	11.5	977	965
8	22.93	97	96	11.6	11.8	1135	1134
9	25.80	109	109	11.8	11.8	1296	1295

$a = 3/4$, $b = 9/8$

0	0.74	3	2	7.8	8.5	24	17
1	1.51	6	7	8.9	9.4	56	66
2	3.06	13	14	9.9	10.0	127	141
3	4.60	19	23	10.5	10.4	203	240
4	6.13	26	27	10.9	10.9	282	293
5	7.66	32	32	11.2	11.2	363	359
6	9.19	39	40	11.5	11.5	446	458
7	10.72	45	54	11.7	11.7	531	633
8	12.26	52	54	11.9	11.9	617	644
9	13.79	58	61	12.0	12.0	704	734

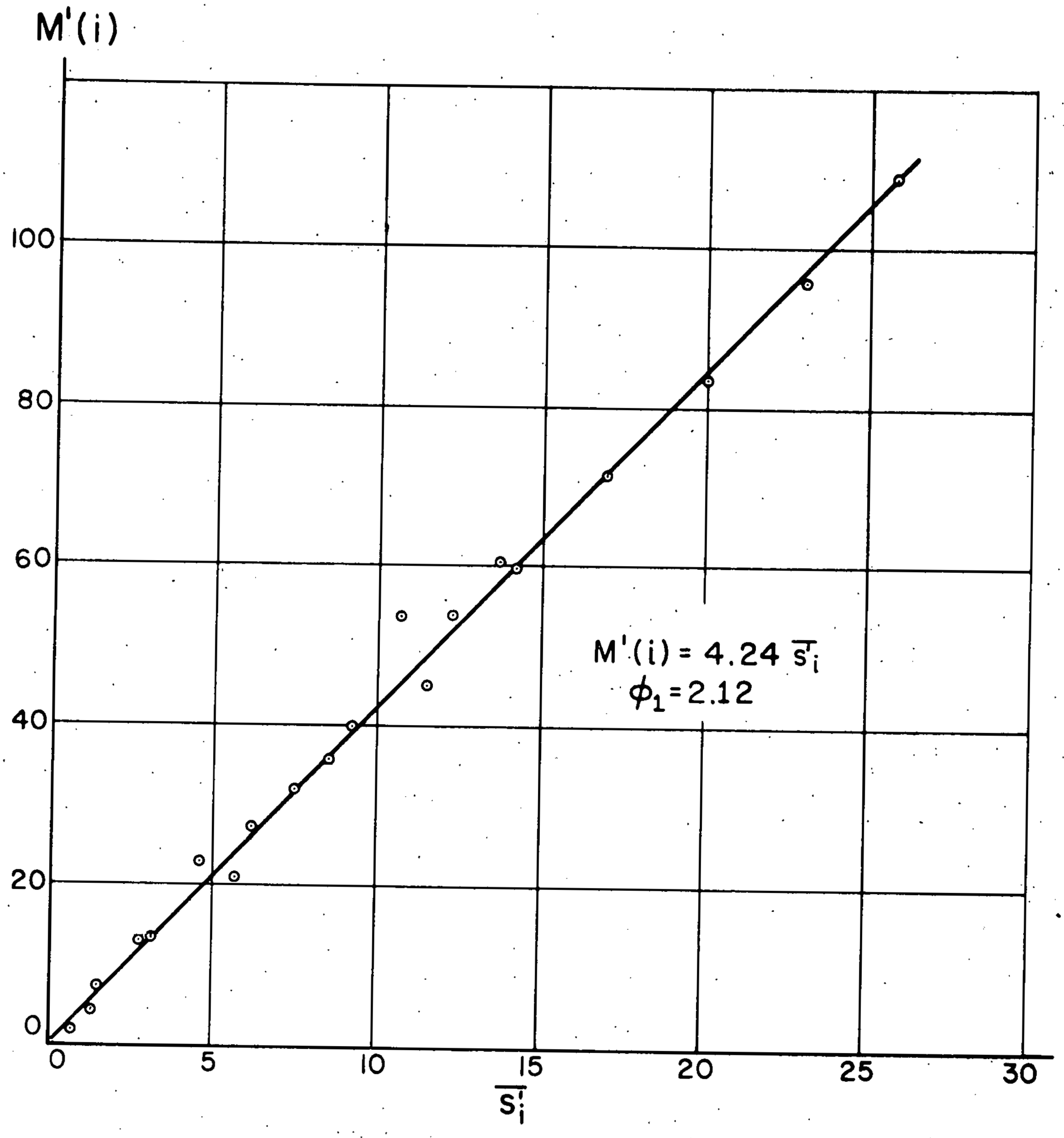


Figure 15. M'(i) versus $\overline{s_i'}$

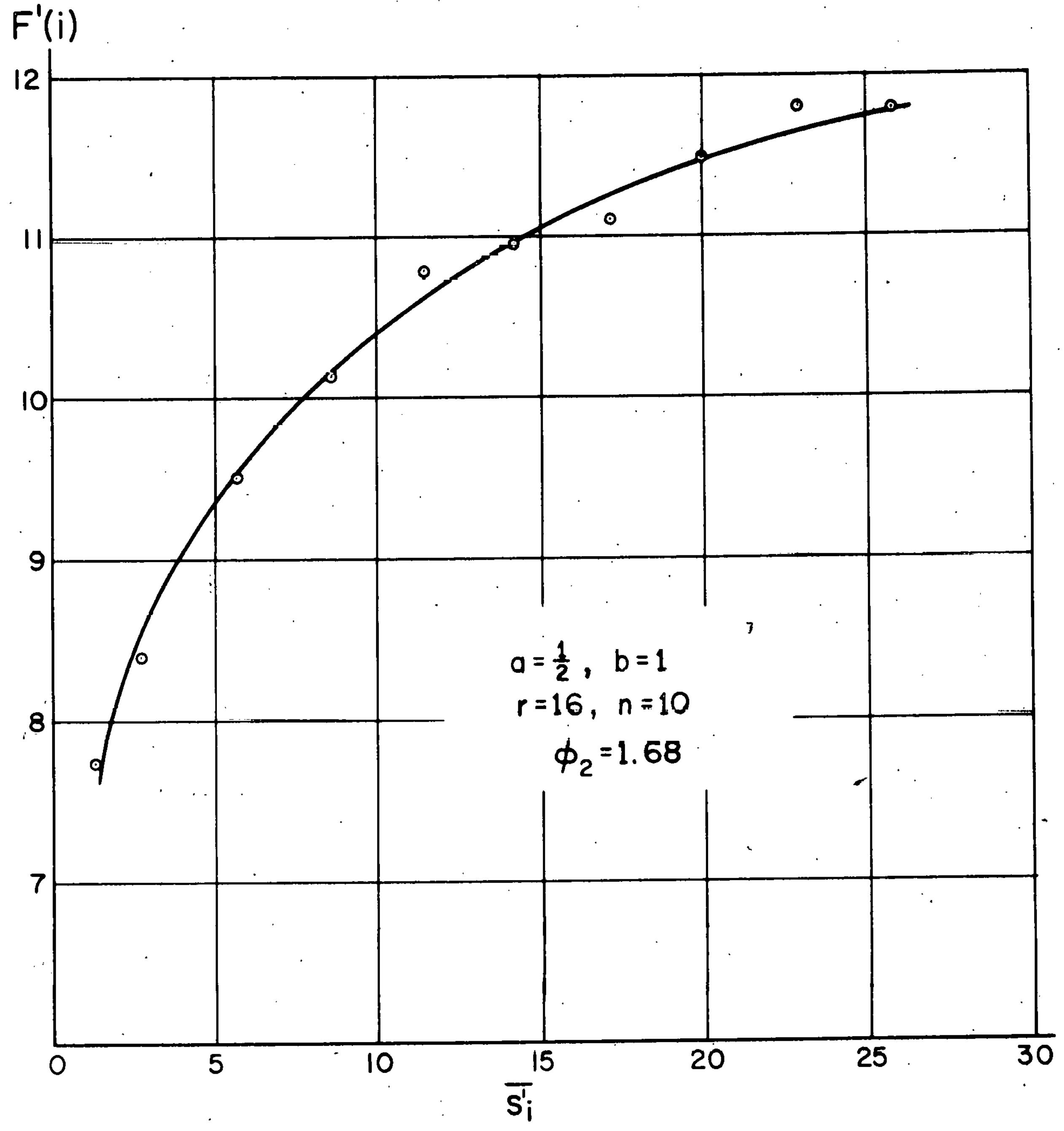


Figure 16a. F'(i) versus $\overline{s_i'}$

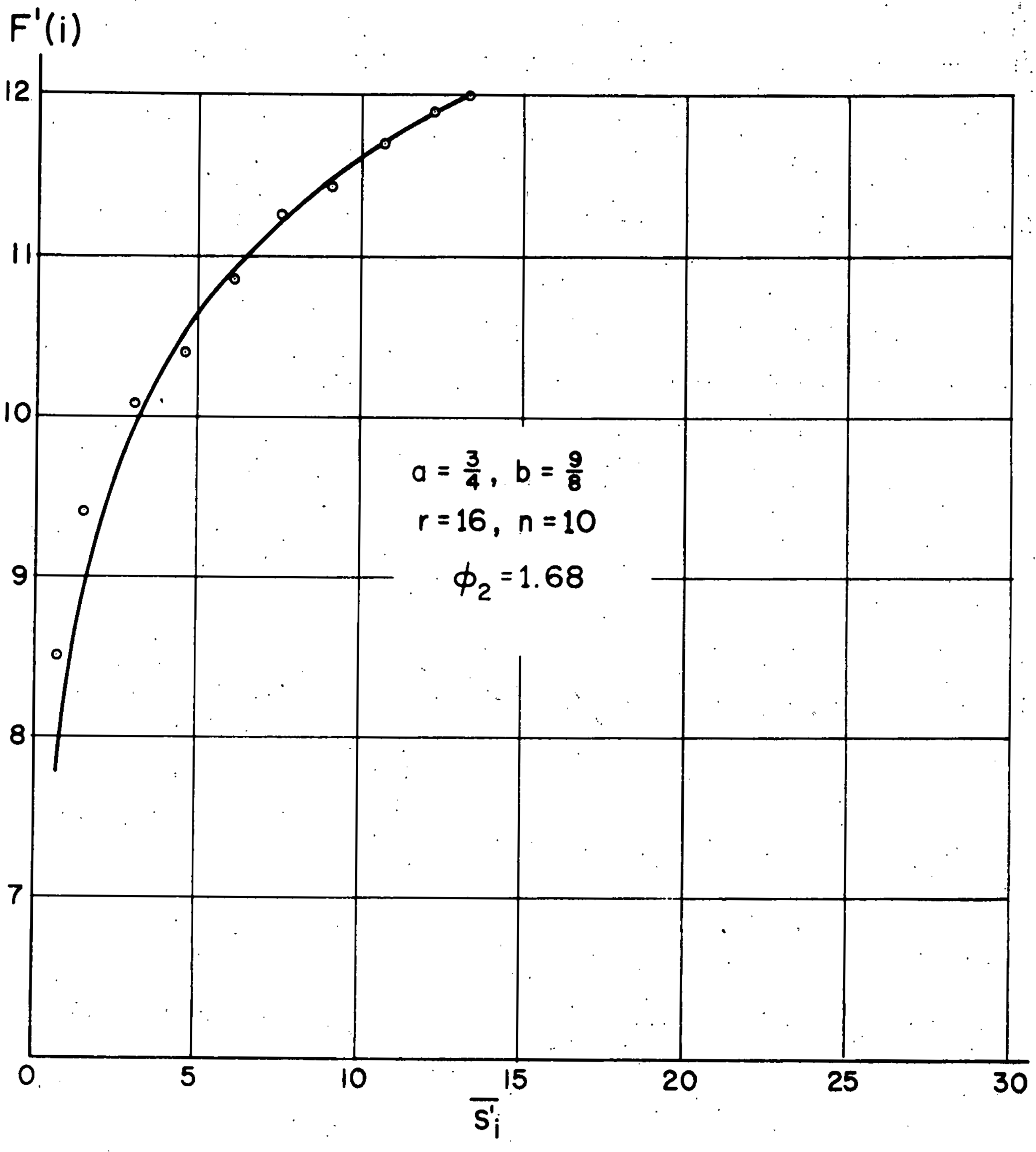


Figure 16b. F'(i) versus $\overline{s'_i}$

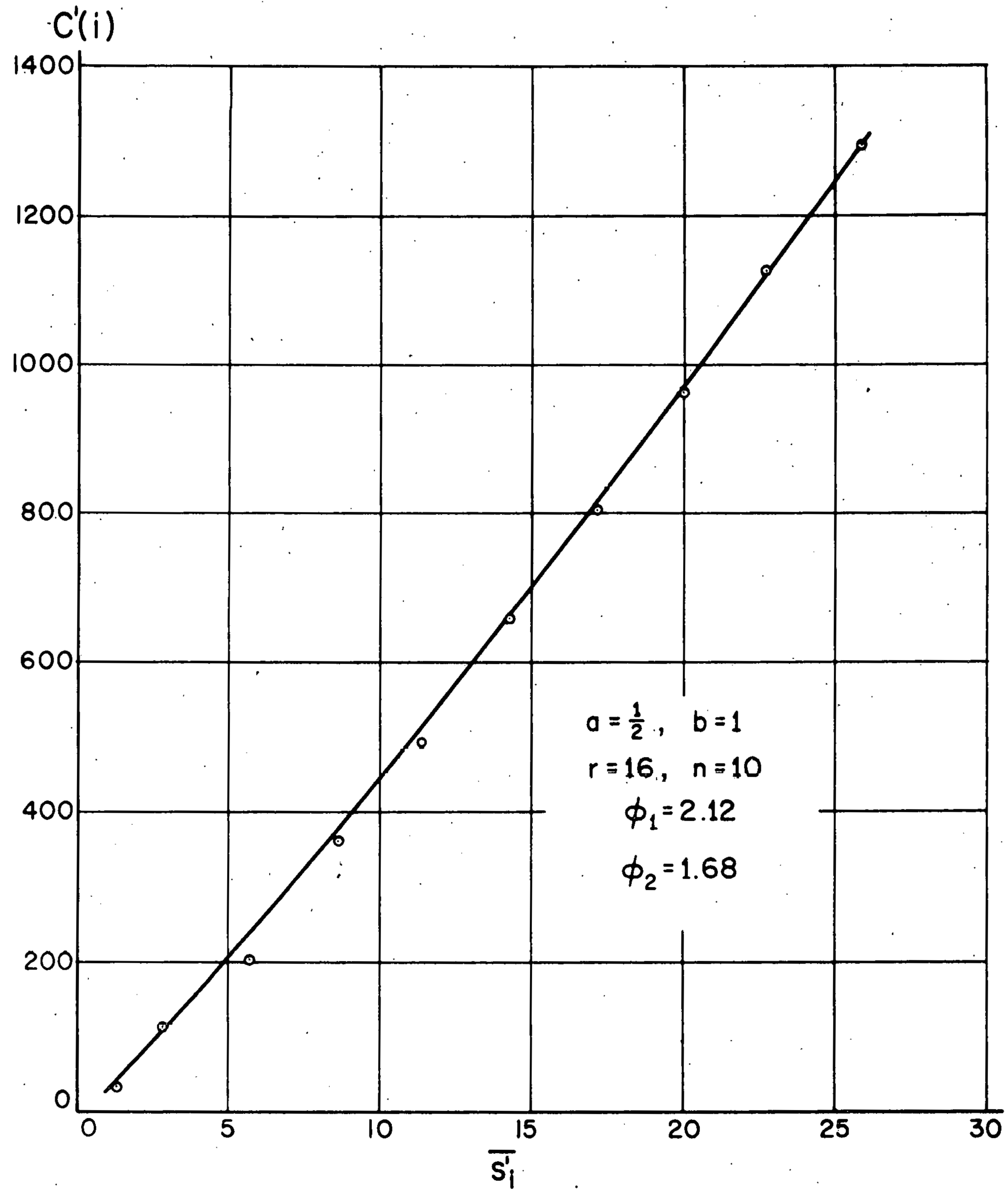


Figure 17 a. C'(i) versus $\overline{s'_i}$

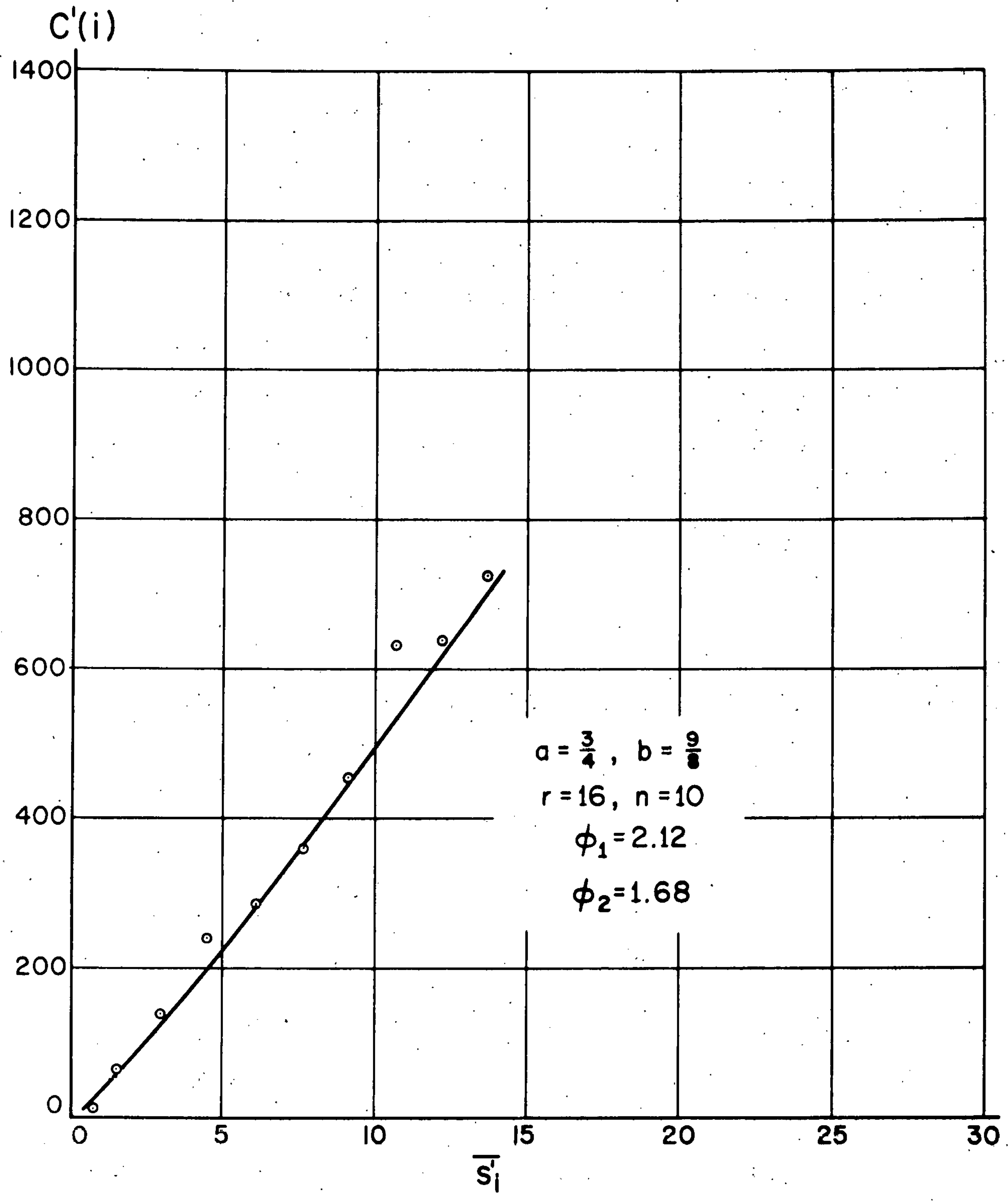


Figure 17b. $C'(i)$ versus $\overline{s'_i}$

5.3.4 Discrepancies

The two cases for which numerical results were presented in Section 5.2.1 differ only in the range of the divisor. We should also consider the effect of varying the precision in the estimates of the operands. The program, QS3, was therefore also run for the same parameter values as listed in Table 2 (Section 2.5.1) except that Δrp, γ, and λ were decreased from 1/16 to 1/32. The minimized results are shown in Table 6. Numbers under the heading 'Equation' are from the evaluation of Equation 5.39; numbers under the heading 'QS3' are from the QS3 and minimization programs.

Table 6. Comparison of Results from Estimating Equation and the QS3 Program for $\Delta rp = 1/32$.

i	$\overline{s'_i}$	M'(i)		F'(i)		C'(i)	
		Equation	QS3	Equation	QS3	Equation	QS3
0	1.16	5	3	7.37	7.66	36	23
1	2.38	10	10	8.41	8.20	84	82
2	4.80	20	20	9.43	9.65	191	193
3	7.21	31	34	10.01	10.02	306	346
4	9.62	41	44	10.43	10.8	425	476
5	12.03	51	62	10.75	10.9	548	679
6	14.44	61	67	11.02	11.4	674	749
7	16.85	71	84	11.24	11.5	802	970
8	19.25	82	90	11.43	11.9	933	1067
9	21.66	92	110	11.60	11.8	1065	1303

In Figure 18, the data from the C'(i)-QS3 column of Table 6 have been added (denoted by X's) to Figure 17(a). Note that these X-points start near the predicted values (solid line) but increasingly fall above the expected values.

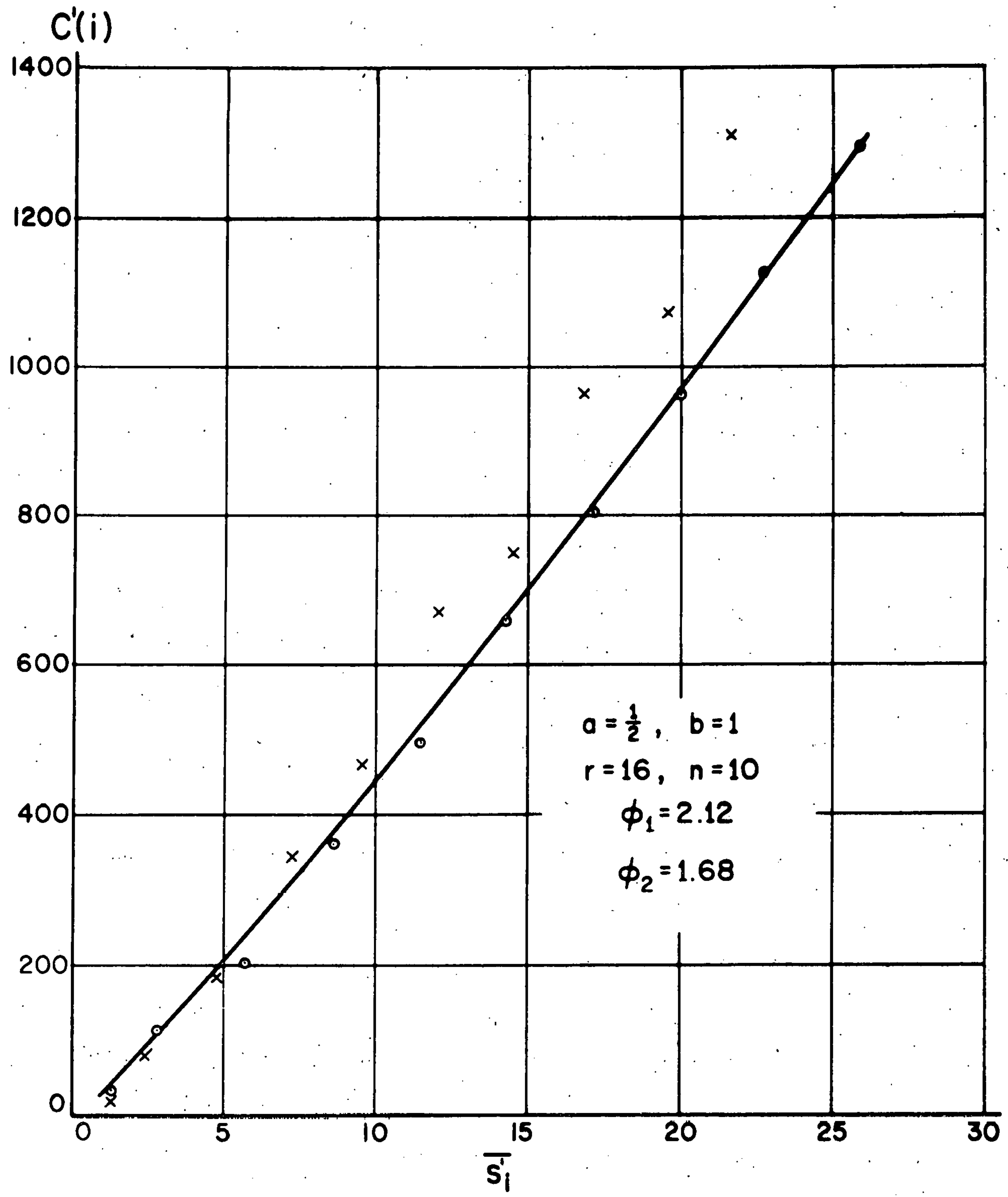


Figure 18. C'(i) versus $\overline{s'_1}$ for $\Delta rp = 1/16$ and $\Delta rp = 1/32$.

The source of this discrepancy turns out not to be the predictive equations, as might be first suspected, but rather the QS3 algorithm; specifically the decision to pick divisor transition values as the simplest binary fraction in the allowable interval. This choice was made in the early stages of the research when other measures of cost were being used and in changing to the minterm approach it was not evaluated critically. Fortunately, as will be explained, it was possible to salvage the numerical results produced by QS3. A correct algorithm has also been found and is described in the Appendix.

The essence of the problem is the failure to fully appreciate the two-dimensional nature of the minimization problem. For several of the q-regions which produced doubtful results, the areas corresponding to the prime implicants of the reduced function were drawn on a P-D plot. The upper and lower stairstep boundaries were therefore made apparent.

By close inspection of the boundaries, it could be seen that the decision to force the location of risers to the simplest binary fraction sometimes over-constrainted the location of the tread. In other words, in some cases for which a divisor interval would have been spanned with one tread, the algorithm generated two treads. Furthermore, each of these extra treads required an extra prime implicant to define it. Thus, although the output function was minimal for the given definition of the q-region, the given definition of the q-region was unduly complicated and therefore not truly minimal. By manually revising the boundary to eliminate the superfluous prime implicants, it was found that the cost was reduced to close agreement with the predicted values.

But the constants in the equation for estimating cost, $\emptyset_1$ and $\emptyset_2$, were specified based upon results from the QS3 program. Why should they be trusted? The answer to this question is found in the following argument.

If we think of the transition region between q(i) and q(i-1) as being defined by a grid of vertical spacing, Δrp, and horizontal spacing, Δd, then the set of all boundaries between q(i) and q(i-1) is all stairsteps which can be drawn along these grids and still remain inside the transition region. As Δd and Δrp are decreased the number of different boundaries increases exponentially. The problem is to pick boundaries that will minimize the number of literals in the Boolean function defining the area enclosed by the boundaries. (Such an algorithm is described in the Appendix.) Fortunately for the parameter values used to derive the constants $\emptyset_1$ and $\emptyset_2$, there was very little choice in selecting the boundaries due to the dimensions of the transition regions. It is, therefore, asserted that the boundary produced by the QS3 algorithm and a correct algorithm would be very nearly the same. A graphical spot check of several of the boundaries confirmed this assertion. When however, Δrp was reduced from 1/16 to 1/32 the number of possible boundaries increased and thus the discrepancy became apparent.

There is one other case for which a discrepancy is apparent. In Table 5 for a = 3/4, b = 9/8, and i = 7, notice that M'(i) from QS3 is 54 while the predicted value is 45. This difference accounts for the high points at $\overline{s_i'} = 10.72$ in Figures 15 and 17(b). The prime implicant covering for this case (q(7)) was drawn and it was thus discovered that six extra prime implicants had been generated. In this case, although Δrp is also 1/16, the shifting of the divisor range to the right increases the width of the transition region to the extent that the QS3 algorithm may fail badly for d values near the upper limit, b. Fortunately, it did not except in the one region.

5.4 Analytic Results Concerning Cost of Table 1

5.4.1 Preliminary Remarks

The program, QS4, produces a cost estimate of Table 1 for a Type 1 structure for which the precision of $A\hat{d}$ is such that the rounded, integer portion of $A\hat{d}$ is a correct quotient digit. As mentioned in Section 2.4, we are also interested in hybrid structures in which Table 1 and the multiples are used to transform the divisor and remainders before they are applied to Table 2. In the following sections we consider the effect of the transformation on the design parameters for Table 2 and then propose an expression to estimate the cost of implementing Table 1 for given precision in A and $\hat{d}$.

5.4.2 Worst Case Bounds on Transformed Parameters

As in Section 2.2, assume that we are given $\hat{d}$ which is representative of divisor values in the range $\hat{d} - \alpha \leq \hat{d} \leq \hat{d} + \beta$ and are given $r\hat{p}$ which is representative of remainders in the range $r\hat{p} - \lambda \leq r\hat{p} \leq r\hat{p} + \gamma$. Let $A = F(\hat{d})$ be generated by Table 1. The range of the transformed divisor, d^T, now represented is given by

$$A\hat{d} - A\alpha \leq d^T \leq A\hat{d} + A\beta \tag{5.42}$$

and the range of the transformed remainder rp^T is given by

$$Ar\hat{p} - A\lambda \leq rp^T \leq Ar\hat{p} + A\gamma \tag{5.43}$$

The divisor range which must be accommodated by Table 2 is (a^T, b^T), where

$$a^T = (A\hat{d})_{min} - A_{max}\cdot\alpha, \text{ and} \tag{5.44}$$

$$b^T = (A\hat{d})_{max} + A_{max}\ \beta. \tag{5.45}$$

The worst-case transformed values of α, β, λ, and γ are merely $A_{max}\alpha$, $A_{max}\beta$, $A_{max}\lambda$, and $A_{max}\gamma$. If 2^{-j} is the weight of the low order bit in A, then

$$\Delta d^T = \Delta d\ 2^{-j}, \text{ and} \tag{5.46}$$

$$\Delta rp^T = \Delta rp\ 2^{-j}. \tag{5.47}$$

Assuming that $A_{max} = 2$, then d' (Equation 5.19) becomes

$$d' = \frac{2\lambda + 2\gamma - 2^{-(\varepsilon+j)} + 2v\alpha + w(2\beta - 2^{-(\delta+j)})}{2\rho - 1}. \tag{5.48}$$

Assume that $\alpha = 0$ and that j is sufficiently large to permit the terms $2^{-(\varepsilon+j)}$ and $2^{-(\delta+j)}$ to be neglected relative to λ, γ, α, and β; then

$$d' \approx \frac{2\ \Delta rp\ (\lambda' + \gamma') + 2w\beta}{2\rho - 1}. \tag{5.49}$$

This value of d' for given λ', γ', Δrp, and β is greater than d' as defined in Equation 5.24. Furthermore, d' increases with i due to the $2w\beta$ term. This comparison indicates that although the transformation reduces cost by narrowing the divisor range for Table 2, it increases cost by increasing restrictions on the q-region boundaries.

The most difficult terms to evaluate in this analysis are $(A\hat{d})_{min}$ in Equation 5.44 and $(A\hat{d})_{max}$ in Equation 5.45. This is the subject of the remainder of this section.

The design problem for Table 1 may be viewed as that of implementing an estimate of the function $f(d) = d^{-1}$. In the following analysis we shall treat divisors in the range $1/2 \leq d < 1$. The approach adopted here is to specify the precision in A, the estimates of d^{-1}, and then to determine the precision in $\hat{d}$ required to guarantee that $\hat{d}A$ is within a certain interval in the vicinity of one. The precision of A is selected as the independent variable since it determines the number of additions required in forming the product $\hat{d}A$. The number of additions is the dominant factor in determining the operating time of the T1, M1, M2 part of the quotient selector.

Let the set of discrete values of the output of Table 1 be defined by

$$A = \{ m\tau \} \tag{5.50}$$

where $\tau = 2^{-j}$ for some positive integer, j, and m is an integer ranging from $1/\tau$ through $2/\tau$. The tick marks on the ordinate of Figure 19 designate such a set for $\tau = 2^{-4}$.

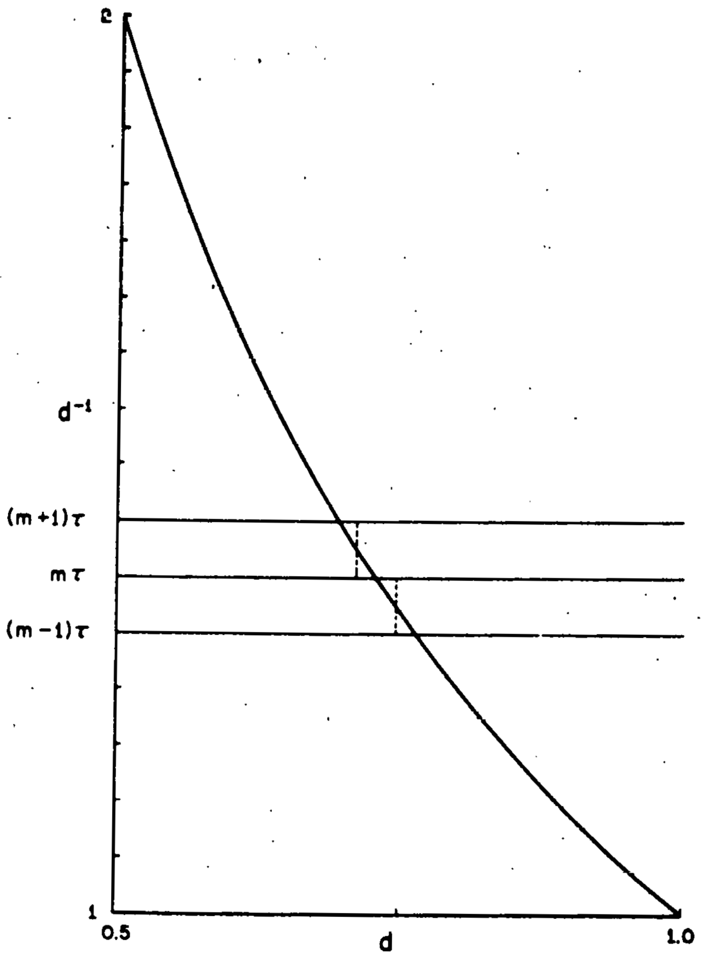


Figure 19. Geometry for Derivation of Estimates of d^{-1}

For every element of A we must define a divisor interval for which $m\tau$ is used as the estimate of the reciprocal of divisor values in the interval. Interpreted graphically, the elements of A determine the location of the treads of a stairstep approximation to d^{-1}. The remaining task is to specify the location of the risers (the dotted lines in Figure 19).

Let $d_{l,m}$ and $d_{r,m}$ denote the left and right ends respectively of the divisor interval for which $A = m\tau$ is taken as the inverse of divisor values in the range $d_{l,m} \leq d < d_{r,m}$. It may be shown that the optimum values for $d_{l,m}$ and $d_{r,m}$ in the sense of minimizing the maximum value of $|1-\hat{d}A|$ are

$$d_{l,m} = \frac{2}{\tau\ (2m+1)}\text{, and} \tag{5.51}$$

$$d_{r,m} = \frac{2}{\tau\ (2m-1)} \tag{5.52}$$

These equations correspond to the reciprocal of the average value of τm and $\tau(m+1)$, and τm and $\tau(m-1)$. For divisor values, d, in the range $d_{l,m} \leq d < d_{r,m}$, the range of dA is given by

$$1 - \varepsilon^{-}(m) < dA < 1 + \varepsilon^{+}(m) \tag{5.53}$$

where

$$\varepsilon^{+}(m) = 1 / (2m - 1) \tag{5.54}$$

$$\varepsilon^{-}(m) = 1 / (2m + 1) . \tag{5.55}$$

The negative error is maximum for $m = m_{min} = 1/\tau$, but since $1/2 \leq d < 1$, the positive error, $\varepsilon^{+}(m)$ is maximum at $m = m_{min} + 1$.

In practice $d_{l,m}$ and $d_{r,m}$ are also discrete values and thus in general, cannot be placed precisely as specified by Equations 5.51 and 5.52. In this case the determination of the error bounds on the product $\hat{d}A$ is more complicated.

If $d_{l,m}$ and $d_{r,m}$ are represented to δ places to the right of the radix point then the actual end points can be within $2^{-(\delta+1)}$ of the theoretically optimal point. Let $\Delta = 2^{-(\delta+1)}$ for the worst case, replace $d_{l,m}$ by $d_{l,m} - \Delta$ and replace $d_{r,m}$ by $d_{r,m} + \Delta$.

Now,

$$1 - \varepsilon(m) \leq \hat{d}A \leq 1 + \varepsilon^{+}(m) \tag{5.56}$$

where

$$\varepsilon^{+}(m) = m\Delta\tau + 1 / (2m - 1) \tag{5.57}$$

$$\varepsilon^{-}(m) = m\Delta\tau + 1 / (2m + 1) \tag{5.58}$$

Note that due to the range restriction of d,

$$\varepsilon^{+}(m_{min}) \doteq m_{min}\Delta\tau \tag{5.59}$$

and

$$\varepsilon^{-}(m_{max}) = m_{max}\Delta\tau \quad . \tag{5.60}$$

Since we require

$$2^{-\delta} \leq \frac{1}{\tau}\left(\frac{1}{m} - \frac{1}{m+1}\right) \tag{5.61}$$

for all allowable m, the maximum value of $2^{-\delta}$ should be less than or equal to $\tau/4$, and δ should be less than or equal $j + 2$.

For given values of τ and Δ

$$(A\hat{d})_{min} = 1 - \varepsilon^{-}(m)_{max} \tag{5.62}$$

$$(A\hat{d})_{max} = 1 + \varepsilon^{+}(m)_{max} \tag{5.63}$$

taken over all m in the range $1/\tau$ to $2/\tau$.

5.4.3 An Estimate of the Cost of Table 1

We now derive an expression with which to estimate the minimum cost in literals of Table 1 when structured as specified in Section 3.2.3. Let the outputs of value A be of the form

$$A = a_{-2}\, a_{-1} \cdot a_1\, a_2 \cdots a_j$$

and considered the d axis of Figure 19 to be equally divided in units of $2^{-\delta}$. After all values of $d_{1,m}$ and $d_{r,m}$ are specified, each bit of A may be defined by a sum-of-products of minterms of the form $k\,2^{-\delta}$.

Let $A = a_{-2}a_{-1}.a_1a_2\ldots a_j$. We will now derive an estimate of the cost of implementing $a_i = f_i(d)$. In the range $1 \leq A \leq 2$, each bit, a_i, is 1 in 2^{i-1} intervals, each of length 2^{-i}. Let $y'_{i,k}$ be the value of the bottom of the kth interval along the d^{-1} axis for bit a_i and let $y''_{i,k}$ be the top of the interval.

Thus,

$$y'_{i,k} = 1 + (2k - 1)\, 2^{-i} \tag{5.64}$$

$$y''_{i,k} = 1 + 2k2^{-i} \tag{5.65}$$

for $i = 1, 2, \ldots, j$ and $k = 1, 2, \ldots, 2^{(i-1)}$.

Let $X_{i,k}$ be the width of the corresponding interval along the d-axis, thus

$$X_{i,k} = \frac{2^{-i}}{(4k^2 - 2k)\, 2^{-2i} + (4k - 1)\, 2^{-i} + 1} \tag{5.66}$$

Let each interval of width $2^{-\delta}$ along the d-axis correspond to a minterm, each with a fan-in of δ. The number of minterms required to define $X_{i,k}$ is

$$M_{i,k} = 2^{\delta} X_{i,k} , \qquad (5.67)$$

the number of literals is

$$C_{i,k} = \delta M_{i,k} . \qquad (5.68)$$

Using the same approximation to the minimization algorithm as described in Section 5.3.3, the cost in literals, after minimization for implementing the $X_{i,k}$ interval is

$$C_{i,k} = M'_{i,k} F'_{i,k} \qquad (5.69)$$

where, with $\mu = I(\log_2 M_{i,k})$

$$M'_{i,k} = M_{i,k} / 2^{\mu}$$

is an approximation to the number of prime implicants required and

$$F'_{i,k} = \delta - \mu \qquad (5.70)$$

is an approximation to the average fan-in.

The cost of implementing $a_i = f_i(d)$ is therefore

$$C'_i = \sum_{k=1}^{2^{(i-1)}} C_{i,k} . \qquad (5.71)$$

The total number of prime implicants required is

$$M'_i = \sum_{k=1}^{2^{(i-1)}} M'_{i,k} . \qquad (5.72)$$

The cost for the entire table is therefore

$$C_{T1} = \sum_{i=1}^{j} C'_i + M'_i \qquad (5.73)$$

6. ESTIMATES OF COST AND PERFORMANCE

6.1 Preliminary Remarks

In this section we use the analytic tools developed in Section 5 together with the definitions in Section 3 to tabulate samples of expected cost and performance. Results are given for Type 2 structures, Type 1 structures, and finally for a family of hybrid structures. Since the radix of the model division is the primary determinant of performance, for each structure we first consider cost versus radix, then performance versus radix, and finally cost versus performance.

Some of the results depend upon assignment of numerical values to quantities used in the definitions of Section 3. The values selected are based upon experience in arithmetic unit design. A different set of realistic values would only shift the location of the cost-performances curves and not materially alter the shape of the curve. General conclusions inferred from them would not change.

6.2 Type 2 Structures

6.2.1 Cost versus Radix

The cost of Table 2, C_{T2}, is given by

$$C_{T2} = \sum_{i=0}^{n-1} (C'(i) + M'(i)) \qquad (6.1)$$

where $C'(i)$ is defined by Equation 5.39 and $M'(i)$ is defined by Equation 5.40.

Tables 7a and 7b summarize cost versus radix for several values of Δrp. Table 7a is for a divisor in the range 1/2 to 1 and Table 7b is for a divisor in the range 3/4 to 9/8. In all cases, $\rho = 2/3$, $\gamma' = \lambda' = 1$, $\beta' = 1$, and $\alpha' = 0$. The quantity Δd is $2^{-\delta}$ where δ is given for each entry in the tables.

The limiting cases (4 and 8) are based upon the assumption that the precision in $r\hat{p}$ and $\hat{d}$ is increased such that $s_i' = s_i$. A near minimal cost should lie between Cases 1 and 4 for the first division range or between Cases 5 and 8 for the second division range. The cost entries are given in the following form:

18	(Prime Implicants)
111	(Literals in AND Gates)
129	(Total Cost)

Table 7a. Cost of Table 2 versus Radix

r	δ	Case 1 Δrp=1/16	δ	Case 2 Δrp=1/32	δ	Case 3 Δrp=1/64	δ	Case 4 Δrp=0
4	5	18	5	15	3	14	∞	13
		111		90		81		74
		129		105		95		87
16	8	552	7	464	7	430	∞	400
		6170		5064		4646		4291
		6722		5528		5076		4691
64	9	10470	9	8792	9	8148	∞	7595
		160526		132578		121971		112928
		170996		141370		130119		120523
256	11	174597	11	146610	11	135871	∞	126656
		3381283		2802307		2582126		2394169
		3555880		2948987		2717997		2520825

Table 7b. Cost of Table 2 versus Radix

r	δ	Case 5 Δrp=1/16	δ	Case 6 Δrp=1/32	δ	Case 7 Δrp=1/64	δ	Case 8 Δrp=0
4	5	10	4	8	3	8	∞	7
		61		52		49		46
		71		60		57		53
16	7	296	6	261	6	247	∞	234
		3353		2920		2742		2583
		3649		3181		2989		2817
64	8	5597	8	4953	8	4684	∞	4443
		86870		75988		71481		67470
		92467		80941		76165		71913
256	10	93341	10	82590	10	78097	∞	74090
		1825332		1600477		1505408		1424130
		1918673		1683067		1583505		1498220

6.2.2 Performance versus Radix

The following equations from Section 3 are relevant to the calculations in this section.

Operating Time of Model Division:

$$T_Q = T_{PREF} + T_{M1} + T_{T2} + T_R. \qquad (3.7)$$

Performance of Model Division:

$$P_Q = \frac{\log_2 r}{T_Q}. \qquad (3.8)$$

Operating Time of Full Precision Division:

$$T_D = M \quad \frac{T_A}{2} + \frac{T_Q + T_C}{\log_2 r} \ . \tag{3.11}$$

Performance of Full Precision Division:

$$P_D = \frac{2 \log_2 r}{T_A \log_2 r + 2(T_Q + T_C)} . \tag{3.12}$$

Table 8 is a summary of P_Q and P_D for several radices with $T_{PREF}=3$, $T_{M1} = 0$, $T_{T2}= 2$, $T_R = 1$, $T_A = 3$, $T_C = 4$. For these values $T_Q = 6$. Note that we have actually computed a best-case for performance since we have assumed that Table 2, even for the higher radices, can be implemented in two delays ($T_{T2} = 2$).

Table 8. Performance of Type 2 Structure versus Radix

r	P_Q (bits/delay)	P_D (bits/delay)
4	.33	.15
16	.67	.25
64	1.00	.32
256	1.33	.36

6.2.3 Cost versus Performance

Neglecting the cost terms C_{PREF}, C_{DEF}, and C_R, the cost of implementing a Type 2 structure is C_{T2}. Table 9 summaries the bounds on C_{T2} versus performance of the full precision division. The actual cost should lie between the lower bound (LB) and the least upper bound (LUB) corresponding to Case 1 in Table 7a and Case 5 in Table 7b. These results are plotted and discussed further in the summary and conclusions (Section 7).

Table 9. Cost Bounds versus Performance for Type 2 Model Division

P_D (bits/delay)	C_{T2} (literals)					
		a = 1/2, b = 1			a = 3/4, b = 9/8	
	Times Increase	LB	LUB	Times Increase	LB	LUB
.15	1.00	87	129	1	53	71
.25	1.67	4,691	6,722	54	2,817	3,649
.32	2.13	120,523	170,996	1385	71,913	92,467
.36	2.40	2,520,825	3,555,880	28975	1,498,220	1,918,673

6.3 Type 1 Structures

6.3.1 Cost versus Radix

Neglecting the cost terms C_{PREF}, C_{DEF} and C_R, the cost of implementing a Type 1 model division is the sum of C_{T1} and C_{M1}. Values for C_{T1} are taken from the results given in Table 4. The term C_{M1} is computed from Equation 3.6, namely,

$$C_{M1} = j\ C_R + N_A\ N_B\ C_A + (N_A + 1)\ N_B\ C_{SG} + (N_B)^2\ C_C.$$

The following values are assumed:

$$C_R = 10,\ C_A = 50,\ C_{SG} = 6,\ C_C = 4,\ N_B = 8.$$

Table 10 summarizes the results.

Table 10. Cost of Type 1 Structure versus Radix

r	C_{T1}	j	N_A	C_{M1}	C_{TOT}
4	28	3	2	1230	1258
16	454	6	3	1708	2162
64	2472	9	5	2634	5106

6.3.2 Performance versus Radix

In computing the operating time for a Type 1 structure we assume that $T_{PREF} = 3$, $T_{T2} = 0$, $T_R = 1$, $T_A = 3$, $T_C = 4$, and $T_{M1} = 3\ N_A$, and therefore from Equation 3.7,

$$T_Q = 3\ N_A + 4.$$

Table 11 presents P_Q (Equation 3.8) and P_D (Equation 3.12) for the cases which were described in Table 4.

Table 11. Performance of Type 1 Structure versus Radix

r	P_Q (bits/delay)	P_D (bits/delay)
4	.20	.12
16	.31	.17
64	.32	.19

6.3.3 Cost versus Performance

Table 12 merges the computations of the previous sections.

Table 12. Cost versus Performance for Type 1 Model Division

P_D (bits/delay)	Times Increase	C (literals)	Times Increase
.12	1.00	1258	1.00
.17	1.42	2162	1.72
.19	1.58	5106	4.06

6.4 Hybrid Structures

6.4.1 Cost versus Radix and Number of Adders in Multiplier 1

For hybrid structures the cost is computed in several stages. First, C_{T1} and the worst-case bounds on the transformed divisor range (a^T, b^T) are computed for the cases of 1, 2, 3, and 4 adders in Multiplier 1. The number of adders, N_A, is the dominant factor in the performance of the model division and furthermore specifies the cost of Table 1 under the assumptions presented in Section 5.4.2. Recall that the maximum uncertainty in A, τ, is 2^{-j} where where $j = 2\ N_A$; that the maximum uncertainty in d, δ, is 2^{-j+2}; and that τ and δ determine C_{T1}.

Next the transformed parameters are computed for each of the four designs. The cost equation for Table 2 is evaluated for each set of transformed parameters, each for four different radices, to yield a total of sixteen designs. The total cost for each hybrid structure is taken to be $C_{T1} + C_{M1} + C_{M2} + C_{T2}$.

Table 13 summarizes the costs for the sixteen cases. The quantities a^T and b^T are defined by Equations 5.62 and 5.63, respectively, and C_{T1} is defined by Equation 5.73. The terms C_{M1} and C_{M2} are computed from Equation 3.6 with $C_A = 50$, $C_R = 10$, $C_{SG} = 6$, $C_C = 4$, and $\varepsilon = 5$. The cost term, C_{T2}, is computed from Equations 5.33, 5.39, and 5.40 with the transformed parameters specified as follows: $A_{max} = 2$, $\Delta rp^T = 2^{-j-5}$, $\Delta d = 2^{-j-\delta}$, $\gamma^T = 1/16$, $\lambda^T = 1/16$, $\alpha^T = 0$, $\beta^T = 2^{-\delta+1}$, $\rho = 2/3$.

Table 13. Cost Computations for Hybrid Structures

Table 1 Parameters	Case No.	r	C_{T1}	C_{M1}	C_{M2}		C_{T2}	C_{Total}
$N_A=1$, $j=2$, $\delta=4$,	1	4	17	512	332	332	10	928
$a^T = 27/32$	2	16	17	648	332	295	3233	4525
$b^T = 41/32$	3	64	17	784	332	5597	84615	91345
	4	256	17	920	332	93341	178713	1,882,323
$N_A=2$, $j=4$, $\delta=6$,	5	4	126	972	892	2	14	2006
$a^T = 123/128$	6	16	126	1220	892	76	843	3157
$b^T = 137/128$	7	64	126	1468	892	1449	22059	25994
	8	256	126	1716	892	24169	465627	492530
$N_A=3$, $j=6$, $\delta=8$	9	4	688	1444	1708	1	3	3844
$a^T = 507/512$	10	16	688	1824	1708	19	212	4451
$b^T = 521/512$	11	64	688	2184	1708	367	5583	10530
	12	256	688	2544	1708	6121	118070	129131
$N_A=4$, $j=8$, $\delta=10$	13	4	3469	1988	2780	0	0	8237
$a^T = 2043/2048$	14	16	3469	2460	2780	5	45	8759
	15	64	3469	2932	2780	85	1286	10552
$b^T = 2056/2048$	16	256	3469	3404	2780	1426	27463	38542

6.4.2 Performance versus Radix and Number of Adders in Multiplier 1

In computing the operating time for the hybrid structures we assume that $T_{PREF} = 3$, $T_{T2} = 2$, $T_R = 1$, $T_A = 3$, $T_C = 4$ and $T_{M1} = 3\ N_A$, and therefore from Equation 3.7

$$T_Q = 3\ N_A + 6.$$

Table 14 presents P_Q (Equation 3.8) and P_D (Equation 3.12) for the cases in Table 13.

Table 14. Performance Calculations for Hybrid Structures

Case No.	P_Q (bits/delay)	P_D (bits/delay)
1	.22	.13
2	.45	.21
3	.67	.27
4	.89	.32
5	.17	.11
6	.33	.18
7	.50	.24
8	.67	.29
9	.13	.09
10	.27	.16
11	.40	.21
12	.53	.26
13	.11	.08
14	.22	.14
15	.33	.19
16	.44	.24

6.4.3 Cost versus Performance

Table 15 merges the cost and performance (P_D) data for the hybrid structures. These results are plotted and discussed further in the next section.

Table 15. Cost versus Performance for Hybrid Model Division Structures

Case No.	P_D (bits/delay)	Times Increase	C (literals)	Times Increase
1	.13	1.00	928	1
2	.21	1.62	4525	5
3	.27	2.08	91345	98
4	.32	2.46	1882323	2028
5	.11	1.00	2006	1.0
6	.18	1.63	3157	1.6
7	.24	2.18	25994	13
8	.29	2.63	492530	245
9	.09	1.00	3844	1.0
10	.16	1.78	4451	1.2
11	.21	2.33	10530	2.7
12	.26	2.88	129131	34
13	.08	1.00	8237	1.0
14	.14	1.75	8759	1.1
15	.19	2.37	10552	1.3
16	.24	3.00	38542	4.7

7. SUMMARY AND CONCLUSIONS

7.1 General Summary

In the summary and conclusions it is convenient to distinguish between the definitive, synthetic, and analytic aspects of this study. Sections 2 and 3 are definitive. Section 2 defines the class of division techniques to be studied and Section 3 defines the measure of cost and performance to be applied. It is noted that an advantage of the model division approach is congruity with commonly used multiplication structures including the capacity to form the partial remainders using non-propagating adders or subtractors. The attendant disadvantages are the necessity to store two bits per quotient digit and the requirement for a terminal step to convert the redundant to non-redundant form. The fact that for division, unlike multiplication, the selection of the jth quotient digit cannot be straightforwardly overlapped with the formation of the jth partial remainder, prompts consideration of high-speed division techniques for the model. Furthermore, the overhead required to "call" and "return" from the model division prompts study of higher radix structures which produce several bits per call. A variable radix block structure of a class of model division schemes is proposed for study.

Section 4 describes algorithms with which to synthesize the most complicated sub-blocks of the family of proposed quotient selectors: a combinatorial network to produce an estimate of the reciprocal of the divisor (Table 1), and a combinatorial network to generate a quotient digit when given $\hat{d}$ and $\hat{rp}$ (Table 2). Although these synthesis routines generate a logic equation definition of the structure, the intent in this study is merely to determine the cost; essentially the number of literals in the logic equations. After the cost vs. performance behavior is sufficiently understood to permit specification of parameters of a practicable model, the synthesis routines may be applied as a first step in implementation.

Section 5 includes the bulk of the analytic work. The section opens with a tabulation of costs for several cases synthesized by the previously defined algorithms. But since there exists many variants of the model division and since even computer synthesis in this case is expensive, the numerical results and insight are applied to hypothesize formulas rather than algorithms with which to estimate cost. The formulas take account of the ten variables of the model division.

Although one of the formulas is normalized with two empirically defined quantities, it is assumed that these quantities are sufficiently constant to permit meaningful prediction of cost for cases other than those used in the normalization. In Section 6, the formulas for both cost and performance are applied to tabulate expected values of cost and performance.

The present section is an attempt to summarize the work in the previous sections, to reach some conclusions about the feasibility of the investigated quotient selection schemes, and to suggest areas for further investigation. The section is subdivided into consideration of numerical cost and performance results, analytic results, and concludes with additional remarks about areas for further research.

7.2 Cost and Performance

Figure 20 is a graphical summary of the cost versus performance estimates tabulated in Section 6. The necessity for a five cycle semi-log plot emphasizes the extreme range of costs and disappointing cost-performance

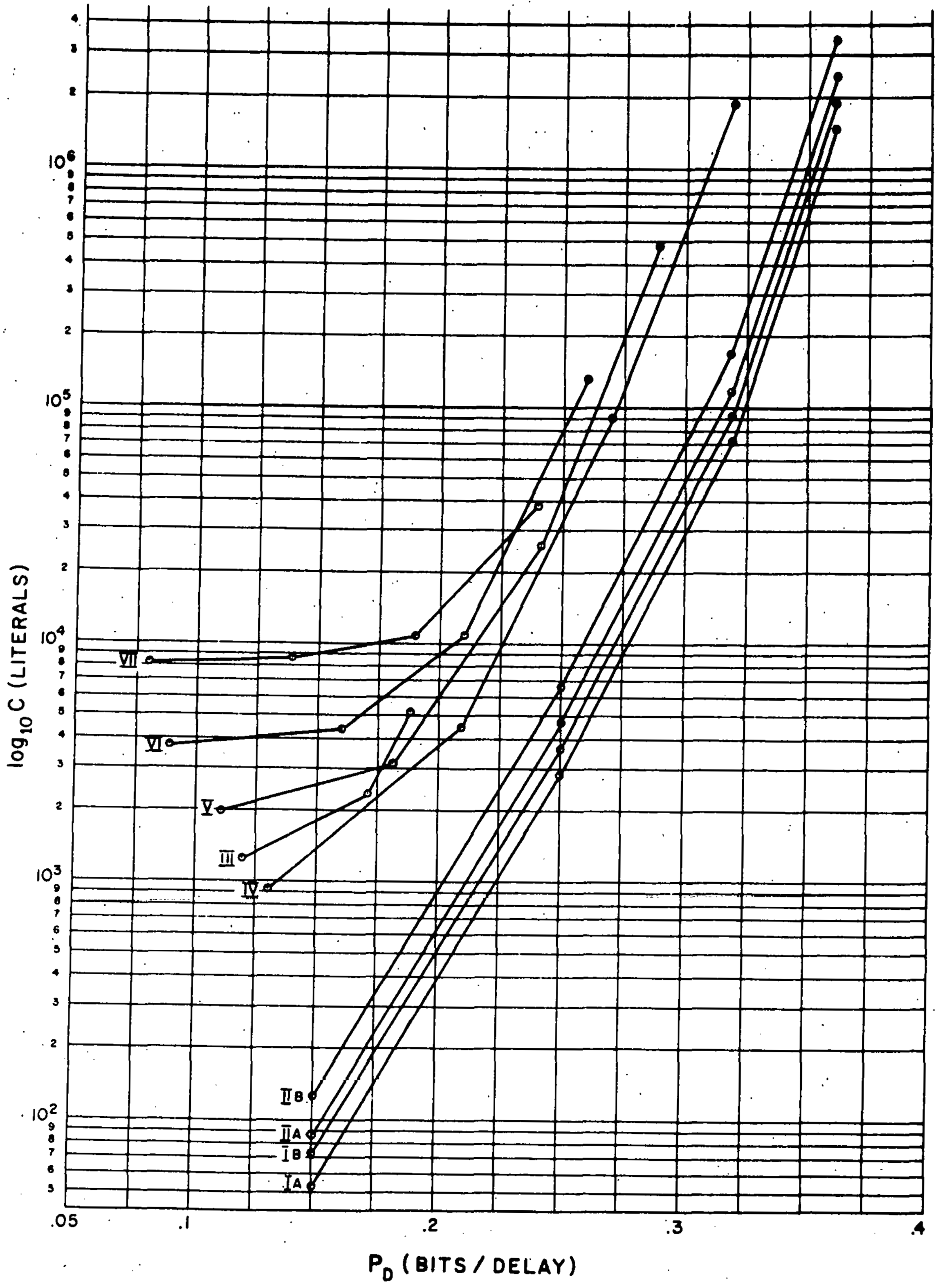


Figure 20. Cost versus Performance for Samples of Model Division Structures

behavior. It is apparent that many of the results are negative; they indicate what not to attempt to implement. The points on the graph are taken from Tables 9, 12, and 15. Points corresponding to the same type structure but differing in radix are connected by straight-line segments. Each of these "curves" is labeled with a Roman numeral.

Curves Ia and Ib, with points from Table 7b, are the lower and upper bounds on the cost of a Type 2 structure (direct table look-up) for divisors in the range (3/4, 9/8). Curves IIa and IIb, with points from Table 7a, are the lower and upper bounds for a similar structure with divisors in the range (1/2, 1). To a first approximation all four curves ($\log_{10}C$) vary linearly with performance and thus

$$\text{Cost} \propto 10^{kP_D}$$

where k is about 18. This exponential behavior is not surprising considering that performance varies as log r (see Equation 3.12) and that cost varies as $r^2 \log r$. This latter statement is derived from Equations 5.39, 5.40, and 5.41.

The radix 4 Type 2 structure is quite practicable, requiring about ten,10-input gates to yield performance of .15 bits per logic delay. Assuming 10 ns. logic, the scheme would generate 60 bits of quotient in about 4 μs. A radix 16 Type 2 structure theoretically increases performance by 5/3, consequently reducing divide time, under the same assumptions, to 2.4 μs. The cost, however, increases over 50 times.

Statements about the radix 16 structure must be qualified by the observation that due to fan-in and fan-out restrictions, the table cannot actually be implemented in two levels of logic. Since the divisor is constant, the $\hat{d}$ portion of each prime implicant can be formed in a cascade of many logic levels without degradation of performance. But going to additional levels to form functions of $\hat{rp}$, although cost may be reduced, will decrease performance below the ideal value assumed in Figure 20. Justification for a radix 16 Type 2 structure is discussed further in connection with a "quotient lookahead" scheme mentioned in Section 7.5. Type 2 structures beyond radix 16 are too expensive to consider further.

Based upon Figure 20, curve III, it appears that a Type 1 structure is never preferable to a Type 2 structure. Although this is probably true, the Type 1 structures might be studied further with the following points in mind:

1. The structures studied here employ a rather conventional multiplier requiring one cascaded adder per two bits of multiplier. Perhaps faster multipliers may be found. It is doubtful, however, that they would be less expensive.
2. For all structures studied the estimate of the partial remainders have been converted to a conventional form. For structures requiring a transformation of $\hat{rp}$, the assimilation is performed after the multiplication. The conversion to conventional form has been required as a concession to reducing the cost of Table 2. For Type 1 structures, Table 2 is not required and thus perhaps the redundantly represented result could be used directly by the shift gates in the full precision arithmetic unit. The elimination of the conversion is roughly equivalent to eliminating one adder from the multiplier structure.

The cost versus performance of the hybrid structures are shown in curves IV-VII, corresponding to 1 through 4 adders in the multipliers, M1 and M2. The curves initially rise slowly relative to the Type II curves but soon become steep as the cost of Table 2 for the higher radices dominates.

The $r^2 \log r$ behavior of C_{T2} is not easy to suppress. Again, based upon results shown in Figure 20, it appears that hybrid structures should not be chosen over a Type 2 structure.

It is apparent from Equation 3.12 that P_D as a function of r has an upper limit of $T_A/2$. This limit is the theoretical upper bound on the performance of the iterative steps of multiplication. With $T_A = 3$, the theoretical ratio of performance of division to performance of multiplication for cases in Figure 20 ranges from 0.09 to 0.53. For practicable cases, the range is 0.225 to 0.375.

7.3 Analytic Results

Only a few of the cases studied appear to be feasible. But negative results are valuable, and furthermore it should be kept in mind that the main purpose of this thesis is not to present an exhaustive enumeration of quotient selection schemes, but rather to develop general techniques for analysis.

It is important to appreciate the generality of the extension of Robertson's cost measurement (s_i) to the imprecise cases (s_i' and s_i''). Although the estimate of cost as a function of s_i' is not rigorous and includes empirically defined constants, the derivation of s_i' is rigorous. The analysis developed in Section 5.3.2 leads to a succinct statement of worst-case precision requirements in $\hat{rp}$ and $\hat{d}$, $(d'' < a)$ and to insight into the effect of the parameters of the model division on the cost of quotient selection.

The s_i' cost measurement is applicable to structures other than those fitting within the structure of the model division shown in Figure 2. For example, as mentioned earlier, the treads of the staircase boundaries between quotient regions may be viewed as comparison constants against which rp is compared to determine in which quotient region it belongs. The divisor range is partitioned into intervals such that for each interval there is a single comparison constant between each quotient region. The comparison constants could be stored in a read only memory. A given divisor value would determine a column of comparison constants which would be read out to become one input to a set of comparators; the other input to the comparators would be $\hat{rp}$. If c_i is the comparison constant between q(i) and q(i-1) then q=k, where k is the greatest such that $rp \geq c_k$. The number of sets of comparison constants has a lower bound of s_n' and upper bound of s_n''. The number of comparison constants in each set is n (assuming implementation of only the first quadrant of the P-D plot).

Among others, the analytic results prompt the following observations:

1. There are minimum requirements for the precision in the estimates of $\hat{rp}$ and $\hat{d}$.
2. For given precision above the minimum required, there is a limit, s_i', to the minimum number of comparison constants required between q(i) and q(i-1).
3. The actual number of steps, $s_{i\ act}$, is greater than s_i' due to discrete effects, i.e. due to the fact that the locations of treads and risers are restricted to discrete values.
4. The upper bound on $s_{i\ act}$, including the discrete effects, is s_i''.
5. Increasing precision in $\hat{d}$ and $\hat{rp}$ moves s_i closer to s_i and $s_{i\ act}$ closer to s_i', but by a decreasing amount.

7.4 Suggestions for Further Investigation

The following topics for further investigation have emerged in the course of this study. The order of listing does not imply any priority.

1. Compare the cost and performance of the model division approach to other division algorithms such as the Wallace algorithm [32] as implemented in the IBM 360/91[14], and division schemes in other large machines such as the CDC 7600.

2. Consider the use of a radix 4, Type 2 structure in a pipeline arithmetic unit. Assuming that the divisors and quotients may be streamed along with the partial remainders, it appears that a set of the inexpensive radix 4, Type 2 model division structures may be used to effectively pipeline the division operation. Multiplication and division could be intermixed in the same pipeline, however, assuming synchronous control, the clock frequency is limited by the quotient selection time and thus the multiply time is degraded.

3. Consider a "quotient lookahead" scheme. Assume that each adder in a cascade of adders is capable of performing a multiplication radix 2^k. Then the shift gates for each adder may be controlled by a model division of the same radix. If the radix of the model is greater than 2^k then more quotient digits are formed than can be used in forming the present partial remainder. It is conceivable, however, that as soon as they are formed they could be used to set shift gates to form the next partial remainder thus overlapping control time. For example, if k=2 but the model division is radix 16, control signals for the shift gates of two successive adders

are generated simultaneously. If a radix 16 quotient selector is coupled to the output of every adder in the cascade, then for each addition/subtraction four bits are formed, two of which overlap with the previously formed bits. The formation of the jth partial remainder may therefore be overlapped with formation of the j+1, radix 4 quotient digit. After startup, the effective control time per addition would be the quotient selection time minus the add time. If the times were equal, then division could proceed at multiply speed.

4. Study the variation in cost of the entire arithmetic unit as a function of ρ, the redundancy ratio. Recall that ρ is one variable in the equation for s_i'. In all numerical work produced in this study $\rho = n/(r-1) = 2/3$. The decision to keep ρ constant excluded the explicit study of radix 8, 32, and 128 for which there is no integer, n such that $\rho = 2/3$.

5. Study a model division structure based upon simultaneous comparisons of $\hat{rp}$ with comparison constants selected by the value of the divisor.

6. Consider the engineering details of a radix 16, Type 2 structure.

7. Program the correct algorithm (Appendix A) for producing the minimal cost definition of a Table 2 structure. Reference [34] defines the minimization algorithm. Compare the results with those produced by the QS3 algorithm (Section 4).

APPENDIX A

Algorithm for Generating Minimum Cost Sum-of-Products Definitions of the q-Regions of Table 2

1. Consider the P-D plot to be covered by a uniform grid with spacing of Δd along the d-axis and with spacing Δrp along the rp-axis. The intersection of each grid line is defined by the order pair $(\hat{d}, \hat{rp})$ where $\hat{d}$ is an integer multiple of Δd and $\hat{rp}$ is an integer multiple of Δrp. Every pair, $(\hat{d}, \hat{rp})$ is representative of full precision quantities in the ranges defined by Equations 2.11 and 2.14. Sufficient condition for the choice of λ, γ, α, β, Δrp, Δd, δ, and ε is that d'' (Equation 5.26) be greater than a, the lower bound of the divisor range. If Δd and/or Δrp are smaller than necessary, the excess precision is removed by minimization. However, the smaller Δd and Δrp, the closer the boundaries between the q-regions may approach the theoretical limit, i.e. the smaller will be the discrete effects.
2. Every pair, $(\hat{d}, \hat{rp})$ corresponds to a minterm, $\overline{\hat{rp}}||\overline{\hat{d}}$. (See page 38 for definition of the notation.)
3. Let R_i be the set of minterms which are required to define q(i), i.e. which must be assigned to the output function f_i. Thus,

$R_i = \{\overline{\hat{rp}}||\overline{\hat{d}}$ | all or any part of the area corresponding to $(\hat{d}, \hat{rp})$ is completely within the area defined by the lines $rp=(i+1-\rho)d$, $rp=(i-1+\rho)d$, $d=a$, and $a=b$.$\}$

Let T_i be the set of minterms which lie completely within the overlap region between q(i) and q(i+1). Thus,

$T_i = \{\overline{\hat{rp}} || \overline{\hat{d}}$ | the area corresponding to $(\hat{d}, \hat{rp})$ is completely within the area defined by the lines $rp=(i+\rho)d$, $rp=(i+1-\rho)d$, $d=a$, and $d=b$.$\}$

Let D be the set of all minterms which correspond to $(\hat{d}, \hat{rp})$ which do not represent area within the boundaries of the P-D plot, i.e. area not within any q-region.

4. Assume a minimization algorithm such as described in Section 4.2.2 which will accept both true minterms, Θ, and a set of don't care minterms, Δ, of a given function. The result of the minimization process is a minimal set of prime implicants, Π. Let Ω be the set of minterms implied by Π, i.e. all minterms for which the function defined by the OR of the elements of Π is true.

5. The following is the proposed algorithm for defining the output functions, f_i, for i=0, 1,..., n.

   a) Let $\Theta = R_0$, $\Delta = T_0 \cup D$.

   b) Execute the minimization algorithms to produce $P_0 = \Pi$, and construct $M_0 = \Omega$. Output function, f_0, is the OR of the elements of P_0.

   c) For i=1,2,...,n do the following:

      Let $\Theta = R_i \cup (T_{i-1} - (T_{i-1} \cap M_{i-1}))$, and $\Delta = T_i \cup D$. Execute the minimization algorithms to produce $P_i = \Pi$ and construct $M_i = \Omega$. Output function f_i is the OR of the elements of P_i.

APPENDIX B

Example of Results of QS4 and Minimization Program.

Note:

$r=4$, $n=2$, $a=1/2$, $b=1$

$\overline{\hat{rp}}=P_1\ P_2\ .\ P_3\ P_4\ P_5\ P_6$

$\overline{\hat{d}}=.d_1\ d_2\ d_3\ d_4$

In the following '1' implies that the variable is present in true form; '0' implies that variable is present in complement form; 'x' implies that variable is absent. Variable d_1 is deleted by inspection.

Minimal cost prime implicants for q(0):

P_1	P_2	P_3	P_4	P_5	P_6	d_1	d_2
0	0	0	x	0	0	x	x
0	0	0	0	x	x	x	x
0	0	0	x	x	0	x	1
0	0	0	x	0	x	x	1

Minimal cost prime implicants for q(1):

P_1	P_2	P_3	P_4	P_5	P_6	d_1	d_2	d_3	d_4
0	0	0	1	x	1	x	0	x	x
0	0	0	1	1	x	x	0	x	x
0	0	x	1	1	1	x	1	x	x
0	0	1	0	x	x	x	x	x	x
0	0	1	x	x	0	x	x	1	x
0	0	1	x	0	x	x	x	x	1
0	0	1	x	x	x	x	x	1	x
0	0	1	x	x	x	x	1	x	x
0	1	0	0	0	0	x	x	1	1
0	1	0	0	x	x	x	1	x	x
0	1	0	x	x	0	x	1	1	x
0	1	0	x	0	x	x	1	1	x

REFERENCES

[1] J. E. Robertson, "A new class of digital division methods," IRE Transactions on Electronic Computers, vol. EC-7, pp. 218-222, September 1958.

[2] T. D. Tocher, "Techniques of multiplication and division for automatic binary computers," Quart. Jour. Mech. Appl. Math., vol. 11, Part 3, pp. 364-384, 1958.

[3] C. V. Freiman, "Statistical analysis of certain binary division algorithms," Proceedings of the IRE, vol. 49, pp. 91-103, January 1961.

[4] M. Nadler, "A high speed electronic arithmetic unit for automatic computing machines," Acta Technica, no. 6, pp. 464-478, 1956.

[5] J. E. Robertson, "Methods of selection of quotient digits during digital division," File No. 663, Department of Computer Science, University of Illinois, Urbana, Illinois, June 1965.

[6] D. E. Atkins, "The theory and implementation of SRT division," Report No. 230, Department of Computer Science, University of Illinois, Urbana, Illinois, June 1967.

[7] D. E. Atkins, "Higher radix division using estimates of the divisor and partial remainders," IEEE Transactions on Computers, vol. C-17, no. 10, pp. 925-934, October 1968.

[8] D. E. Atkins, "Design of the arithmetic units of Illiac III: Use of redundancy and higher radix methods," IEEE Transactions on Computers, (to appear) August 1970.

[9] D. E. Atkins, "Illiac III computer system manual: Arithmetic units, vol. I," Report No. 366, Department of Computer Science, University of Illinois, Urbana, December 1969.

[10] J. E. Robertson, "A deterministic process for the design of carry-save adders and borrow-save subtractors," Report No. 235, Department of Computer Science, University of Illinois, Urbana, July 1967.

[11] R. T. Borovec, "The logical design of a class of limited carry-borrow propagation adders," Report No. 275, Department of Computer Science, University of Illinois, Urbana, Illinois, August 1968.

[12] F. A. Rohatsch, "A study of transformations applicable to the development of limited carry-borrow propagation adders," Report No. 226, Department of Computer Science, University of Illinois, Urbana, June 1967.

[13] J. E. Robertson, "The correspondence between methods of digital division and multiplier recoding procedures," Department of Computer Science Report No. 252, University of Illinois, Urbana, Illinois, December 1967.

[14] S. F. Anderson, J. G. Earle, R. E. Goldschmidt, D. M. Powers, "The IBM System/360 Model 91; Floating-point execution unit," IBM Journal of Research and Development, vol. 11, no. 1, pp. 34-53, January 1967.

[15] A. Avizienis, "Binary-compatible signed-digit arithmetic," AFIPS, Fall Joint Computer Conference, vol. 26, pp. 663-672, 1964.

[16] V. S. Burtsev, "Accelerating multiplication and division operations in high-speed digital computers," in report by The Institute of Exact Mechanics and Computing Technique, The Academy of Sciences of the USSR, Moscow, 1958.

[17] M. Combet, H. van Zonneveld, and L. Verbeek, "Computation of the base two logarithm of binary numbers," IEEE Transactions on Electronic Computers, vol. EC-14, no. 6, pp. 863-867, December 1965.

[18] K. J. Dean, "A precision code converter for reciprocals of binary numbers," The Computer Bulletin, vol. 12, no. 2, pp. 55-58, June 1968.

[19] D. Ferrari, "A division method using a parallel multiplier," IEEE Transactions on Electronic Computers, vol. EC-16, no. 2, pp. 224-226, April 1967.

[20] R. E. Gilman, "A mathematical procedure for machine division," Communications of the ACM, vol. 2, no. 4, pp. 10-12, April 1959.

[21] R. E. Goldschmidt, "Applications of division by convergence," M.S. Thesis, MIT, June 1964.

[22] Ernest F. Hall, David D. Lynch, Richard E. Young, "Generation of products and quotients using approximate binary logarithms for digital filtering applications," IEEE Transactions on Computers Repository, no. R-68-164, 1968.

[23] Jiri Klir, "A note on Svoboda's algorithm for division," Stroje Na Zpracovani Informaci (Information Processing Machines), no. 9, pp. 35-39, 1963.

[24] E. V. Krishnamurthy, "On range-transformation techniques for division," IEEE Transactions on Computers, vol. C-19, no. 2, pp. 157-160, February 1970.

[25] John N. Mitchell, Jr., "Computer multiplication and division using binary logarithms," IRE Transactions on Electronic Computers, EC-11, no. 4, pp. 512-518, August 1962.

[26] Ray G. Saltman, "Reducing computing time for synchronous binary division," IRE Transactions on Electronic Computers, vol. EC-10, no. 2, pp. 169-174, June 1961.

[27] A. Soceneantu, "Binary iterative division," (Report in Progress), Department of Computer Science, University of Illinois, Urbana, Illinois, 1970.

[28] R. Stefanelli, "A suggestion for an high-speed parallel binary divider," IEEE Transactions on Computers Repository, no. R-69-3, October 1968.

[29] A. Svoboda, "An algorithm for division," Stroje Na Zpracovani Informaci (Information Processing Magazine), no. 9, pp. 25-32, 1963.

[30] C. Tung, "A division algorithm for signed-digit arithmetic," IEEE Transactions on Computers, vol. C-17, no. 9, pp. 887-889, September 1968.

[31] R. M. Wade, "A carry-independent quarternary division scheme," IEEE Transactions on Computers Repository, no. R-68-52, November 1967.

[32] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Transactions on Electronic Computers, vol. EC-13, pp. 14-17, February 1964.

[33] E. J. McCluskey, Introduction to the Theory of Switching Circuits, McGraw-Hill, New York, 1965, pp. 135-136.

[34] V. G. Tareski, "Minimization of two level switching circuits involving many variables," Ph.D Thesis in preparation, Department of Computer Science, University of Illinois, Urbana, Illinois.

[35] Chester C. Carroll and George E. Jordan, "A fast algorithm for Boolean function minimization," Auburn University Report No. AD 680 305, December 1968.

[36] Tso-Kai Liu, "A code for zero-one integer linear programming by implicit enumeration (A Programming Manual for ILLIP,)" Department of Computer Science, Report No. 302, December 1968.

[37] T. Ibaraki, et al, "An implicit enumeration program for zero-one integer programming," Department of Computer Science, Report No. 305, January 1969.

VITA

Daniel Ewell Atkins, III was born in Jacksonville, Florida on [redacted]. He received the B.S. degree in Electrical Engineering from Bucknell University, Lewisburg, Pa., in 1965; the M.S. degree in Electrical Engineering from the University of Illinois, Urbana, in 1967; and the Ph.D. in Computer Science from the University of Illinois in 1970.

Between 1963 and 1967 he held summer positions with the Freas-Rooke Computing Center, Bucknell University, and the U.S. Naval Ordnance Laboratory, White Oaks, Md. While attending the University of Illinois he was employed as a research assistant in the Department of Computer Science. He designed the floating point arithmetic units for the Illinois Pattern Recognition Computer (Illiac III) under direction of Professor Bruce H. McCormick, and conducted research in the area of computer arithmetic under the direction of Professor James E. Robertson. Mr. Atkins has published papers evolving from this work in the IEEE Transactions on Computers of October 1968 and August 1970.

Mr. Atkins is a member of Tau Beta Pi, Sigma Xi, Pi Mu Epsilon, Pi Delta Epsilon, the Association for Computing Machinery, the Institute of Electrical and Electronic Engineers, and the American Association of University Professors.

3

Form AEC–427
(6/68)
AECM 3201

U.S. ATOMIC ENERGY COMMISSION
UNIVERSITY–TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO. Report No. 397 COO-1018-1204	2. TITLE A STUDY OF METHODS FOR SELECTION OF QUOTIENT DIGITS DURING DIGITAL DIVISION

3. TYPE OF DOCUMENT (Check one):

- [X] a. Scientific and technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference
  - Date of conference
  - Exact location of conference
  - Sponsoring organization
- [ ] c. Other (Specify)

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures may be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- [ ] c. Make no announcement or distrubution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

Daniel E. Atkins, Research Assistant

Organization

Department of Computer Science
University of Illinois

Signature [illegible]	Date May 28, 1970

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC patent group for clearance.
- [ ] c. Patent clearance not required.