REALIZATION OF GENERALIZED-RADIX, ADDITIVE-NORMALIZATION DIVISION CIRCUITS

A MASTER'S THESIS

in

Electrical and Electronics Engineering
Gaziantep Engineering Faculty
Middle East Technical University

By
Nurdal TÜRKBEYLER

December 1984

Approval of the Graduate School of Natural and Applied Sciences.

Prof.Dr.Bilgin Kaftanoğlu

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Sciences Program of Electrical and Electronics Engineering Department.

Assoc.Prof.Dr.Muhammet Köksal

Chairman of the Department

We certify that we have read this thesis and in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Computer Sciences Program of Electrical and Electronics Engineering Department.

Assoc.Prof.Dr.Muhammet Köksal

Supervisor

Examining Committee in Charge:

Asst.Prof.Dr.Hüseyin Filiz (Chairman)

Assoc.Prof.Dr.Muhammet Köksal

Assoc.Prof.Dr.Ünsal Kalaycıoğlu

Assoc.Prof.Dr.Doğan Çalıkoğlu

Asst.Prof.Dr.Aref Gholinia

ABSTRACT

REALIZATION OF GENERALIZED-RADIX, ADDITIVE-NORMALIZATION DIVISION CIRCUITS

TÜRKBEYLER, Nurdal
M.S. in Electrical and Electronics Engineering
Supervisor: Assoc.Prof.Dr.Muhammet Köksal
December 1984, 58 pages

In this thesis, the theory of generalized-radix, additive-normalization division circuits is summarized and an application of this theory is realized. In these circuits the dividend and the divisor are normalized first, to a pre-determined domain. The quotient selector provides a quotient digit for each partial remainder and the divisor in the pre-determined domain. The quotient selector table is constructed by using one of the three existing algorithms. The quotient digits can be selected based on the crude estimates of the partial remainder and the divisor, since redundant quotient digit sets are used in the algorithms.In this work, the quotient selector unit is realized by using erasable programmable read-only-memories. During the division operation, the multiple of the divisor which depends upon the magnitude of the quotient digit is either added or subtracted from previous partial remainder and the result is obtained by means of an iterative procedure.

This work presents the realization of the two variations of the generalized-radix, additive-normalization division circuits. The control processor is designed based on the suggested algorithm. The data processor circuit is designed and combined with the control processor circuit. The speed of the division operation (time needed for each quotient digit) is determined experimentally as 360ns for the first variation, 340ns for the second variation.

Key words: redundancy, additive-normalization division, quotient selector.

ÖZET

GENEL TABANLI, TOPLAMALARLA BÖLME YAPAN DEVRELERİN GERÇEKLEŞTİRİLMESİ

TÜRKBEYLER, Nurdal
Yüksek Lisans Tezi, Elektrik ve Elektronik Müh.Bölümü
Tez Yöneticisi: Doç.Dr.Muhammet Köksal
Aralık 1984, 58 sahife

Bu tezde genel tabanlı, toplamalarla bölme yapan devrelerin kuramı özetlenmiş ve bu kuramın bir uygulaması gerçekleştirilmiştir. Bu devrelerde öncelikle bölünen ve bölen önceden belirlemiş bir alana kaydırılır. Bölüm seçici devresi önceden belirlenmiş alandaki her kısmi kalan ve bölen için bir bölüm basamağı üretir. Bölüm seçici tablo bilinen üç algoritmadan birinin yardımıyla oluşturulur. Algoritmalarda artık bölüm basamağı kümesi kullanıldığından, bölüm basamakları, kısmi kalanın ve bölenin kaba tahminlerine bağlı olarak seçilebilir. Bu çalışmada bölüm seçici devresi silinebilir, programlanabilir salt oku bellek kullanılarak gerçekleştirilmiştir. Bölme işlemi esnasında bölüm basamağının değerine bağlı olan bölenin katı, bir önceki kısmi kalana eklenir veya çıkarılır ve sonuç yinelemeli bir yöntem yoluyla elde edilir.

Bu çalışma genel tabanlı, toplamalarla bölme yapan devrelerin iki değişik şekilde gerçekleştirilmesini sunmaktadır. Denetim işleyici devresi önerilen algoritmaya göre tasarlanmıştır. Veri işleyici devresi tasarlanmış ve denetim işleyici devresiyle birleştirilmiştir. Bölme işleminin hızı (bir bölüm basamağı için gereken süre) deneysel olarak birinci devre için 360ns, ikinci devre için 340ns olarak belirlenmiştir.

Anahtar kelimeler: artıklık, toplamalarla bölme yapan devre, bölüm seçici.

ACKNOWLEDGEMENTS

The author wishes to express her sincerest gratitude to her supervisor, Assoc.Prof.Dr.V.Ü.Kalaycıoğlu for his guidance, encouragement, suggestions, his great help in the preparation of this work, and above all, for his friendship.

Thanks is also due to Assoc.Prof.Dr.M.Köksal who accepted to take formal supervision of the M.S. program when Assoc.Prof.Dr.V.Ü.Kalaycıoğlu was on leave.

The author wishes to thank Mr.H.Gerez for his excellent drawings.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

a,b	limit points of the domain of the divisor
d	the divisor
$\hat{d}$	the divisor estimate
d_1	input to the control processor
d_2	input to the control processor
h	the function realized by the radix-r^2 QS
f_j	the j^{th} quotient digit function
j	the quotient digit
m	the number of quotient digits in the quotient
N	the number of iterations
n	the magnitude of the largest quotient digit value
P_0	the dividend
p_k	the k^{th} partial remainder
p_m	the remainder
P_z	output of the test block
Q_i	the set of permissible values of quotient digit q_i
Q_{i+1}	the radix-r^2 quotient digit
q	the quotient
q_i	the i^{th} quotient digit
$q(n)$	the definition area of $q_i = n$ region in the P-D plot
r	the radix
rp_i	the shifted partial remainder
rp_{imin}	minimum value of comparison constant
rp_{imax}	maximum value of comparison constant
s_k	the k^{th} selection variable in a normalization
T_0	the output of control register
T_1	the output of control register
T_2	the output of control register
t_i	one iteration time of the QSi type structure
x	T_0
α	uncertainty limit in $\hat{d}$
α'	$\alpha/\Delta d$
β	uncertainty limit in $\hat{d}$
β	$\beta/\Delta d$

Δd	increments in d-axis
Δrp	increments in rp-axis
δ	the number of binary digits to the right of the radix point in $\hat{d}$
ε	The number of binary digits to the right of the radix point in $r\hat{p}$
γ	uncertainty limit in $r\hat{p}$
γ'	$\gamma/\Delta rp$
λ	uncertainty limit in $r\hat{p}$
λ'	$\lambda/\Delta rp$
ρ	the redundancy ratio

CHAPTER 1

INTRODUCTION

In a digital computer, the existance of an adder circuit is enough for implementing all other arithmetic operations by means of software. However, doing these operations by hardware instead of software increases the efficiency of the arithmetic unit of the digital computer. In designing the arithmetic unit of a computer, the most difficult basic operation to implement is division among the four basic operations. A trial-and-error procedure is used in division. For that reason, division takes more time than other three operations (addition, subtraction, multiplication). The salient point in a restoring division algorithm is that it requires an initial guess of the quotient digit followed by a subtraction, or at least a comparison, to determine whether the guess is correct. If it is not correct, the initial choice is modified and the process is repeated. It is the trial-and-error nature of the division operation, whether performed by man or machine, which complicates its execution.

But despite the complexity, the literature is replete with themes and variations for implementing digital division. The division techniques can be classified as additive normalization techniques and multiplicative normalization techniques. In a normalization technique, a variable of the procedure is directed to a known goal keeping the value of the function unchanged by a sequence of additive and multiplicative transformations, resulting in additive or multiplicative normalization, respectively. Division can be evaluated by using either of the techniques. An example of additive normalization technique is the classical division method that we employ in calculations by hand. One of the multiplicative normalization techniques in division is to multiply both the numerator and the denomerator by a sequence of constants to direct the denomerator to unity while the numerator approaches the quotient.

Additive normalization technique is commonly used in hardware implementation of the division operation. The theory and application of additive normalization techniques involving redundancy in the representation of the quotient start with SRT* division. This is a binary non-restoring technique with digit set 1, 0, -1. It was independently discovered by D.Sweeney of IBM, J.E.Robertson of the University of Illinois [1], and T.D.Tocher of Imperial College, London in the middle 1950's. By introducing redundancy in the representation of quotient digits, more freedom was gained in the selection of the quotient digits; thereby increasing the probability of a zero quotient digit, which allows to bypass the adder (or subtracter) and to form the next partial remainder just by shifting the partial remainder left one bit position.

In 1965, Robertson [2] extended the concepts inherent in the SRT division to higher radix division methods. Redundancy was still kept in these schemes. Since the number of different quotient digit values increases proportionally to the radix, if the measure of redundancy is kept constant, probability of a zero quotient digit is reduced so much that adder bypass has no more importance in increasing the speed of higher radix division. However, redundancy allowing freedom in the selection of quotient digits; permits selection of quotient digits based only on higher-order digits of partial remainder and divisor.

Atkins [3],[4] presented a detailed theory of higher radix division using the model division approach. He developed the graphical representation, called P-D plot, which was suggested by Freiman [5]: He developed expressions for the precision of the estimates (which are used as inputs to the model division for a given radix) of the shifted partial remainder and the divisor, for lower limit of the divisor, and finally for the amount of redundancy in the representation of quotient.

Atkins [6] has presented a general structural model of a quotient selector. The structure involves three approaches for the formation of the quotient digit: (i)Multiplying the estimate of the partial remainder with an estimated inverse

* (S:Sweeney, R:Robertson, T:Tocher)

of the divisor, the estimated inverse is found by a table lookup, (ii) Supplying the estimates of the partial remainder and the divisor to a table, which is designed using a graph called "P-D plot", that yields the quotient digit directly, (iii) Combining (i) and (ii). The P-D plot is a graphical representation of the quotient digit value regions as a function of the estimates of the partial remainder and the divisor.

Atkins, in reference [6], developed computer aided methods to construct the tables required by his quotient selector model. The first two approaches of selecting the quotient digit are implemented by using combinatorial circuits. Cost and performance comparison of approach (i) and approach (ii) above showed that approach (ii) is superior to the first one. The table used in the second approach is called "combinatorial quotient selector table" or simply "quotient selector table (QS table)"

Kalaycıoğlu [7] in his Ph.D.Thesis discovered alternate algorithms for the synthesis of the quotient selector table. Atkins algorithm is used to construct a QS table that will be completely implemented by using combinatorial logic. Kalaycıoğlu modified and extended this algorithm for columnwise implementation of the QS table. Columnwise implementation of the QS table gains more importance as the radix increases since the complexity of the QS table grows proportionally to the square of the radix. With the application of virtual memory-paging system, the size of the active memory in the QS is greatly reduced, thus increasing the performance of the quotient selector. A disadvantage of columnwise implementation, however, is the overhead time and cost in loading the active memory in the QS unit for each division operation unless the divisor is constant. He discovered new organizations which reduce the division cycle time by increasing concurrency between quotient digit selection and partial remainder calculation. One of these organizations utilizes the radix-r^2 QS table which permits concurrency in the computation of the quotient digit and the next partial remainder. Kalaycıoğlu in his Ph.D.Thesis, presented four structures, QS1, QS2, QS3, QS4; each of these structures has a speed greater than the preceding one, but at the same time requires more hardware.

In this work, the first two structures in [7] are realized and their speeds are compared. It is observed that the second method is faster than the first one as implied by the theoretical results. A control processor circuit is designed and combined with data processor circuit for each structure. Erasable Programmable Read-Only-Memory (EPROM) is used in the QS unit of the data processor circuit.

CHAPTER 2

THE THEORY OF GENERALIZED-RADIX ADDITIVE NORMALIZATION DIVISION SCHEMES

2.1. Redundancy

Positional notation is the weighted digit representation, in which the digits form a vector $(q_0, q_1, \ldots, q_k)$. Such a vector maps to the value

$$q = r^{\ell} \sum_{i=0}^{k} q_i r^{-i} \tag{2.1}$$

where

r is the radix of the representation,

$q_i \in Q_i$ = the set of permissible values of quotient digit q_i,

$\ell+1$ is the number of radix-r digits to the left of the radix point.

For a fixed base representation, let $Q=\{m, m+1, \ldots, m+n-1\}$. In conventional number systems for non-redundant representations $m=0$, $n=r$ [8].

A number system is redundant if a number in the system can be represented in many different ways. In conventional number systems the representation is redundant if $n>r$. Clearly, using more than ten different digit values in decimal representation or more than two digit values for binary numbers, results in redundancy.

2.2. Additive Normalization Division Algorithm*

B.G.DeLugish [9] suggested a derivation of one possible scheme for the positional notation of the quotient for radix-2 by using additive normalization of the divisor approaching the value of dividend as follows:

$$\alpha_{k+1} = p_0 - d\left(\sum_{i=0}^{k} m_i\right) \tag{2.2}$$

* Some of the symbols used in this section are explained after their first occurrences in the formulas and some are defined on page 7.

where α_k is the variable of normalization which approaches to zero as k increases and $m_i = s_i r^{-i}$. Then

$$p_0 - d\left(\sum_{i=0}^{k} s_i r^{-i}\right) \cong 0$$

or

$$\sum_{i=0}^{k} s_i r^{-i} \cong \frac{p_0}{d}$$

Hence each s_i turns out to be a radix-r digit of the quotient and the procedure in equation (2.2) can be reformulated in the form of positional notation as follows:

$$s_i = q_i \qquad \text{for every } i \in [0,k]$$

$$\begin{aligned} \alpha_{k+1} &= p_0 - d\left(\sum_{i=0}^{k} q_i r^{-i}\right) \\ &= p_0 - d\left(\sum_{i=0}^{k-1} q_i r^{-i} + q_k r^{-k}\right) \\ &= \alpha_k - dq_k r^{-k} \end{aligned} \tag{2.3}$$

Let $p_k = r^k \alpha_k$ for $k>1$, then equation (2.3) reduces to

$$\frac{1}{r} p_{k+1} = p_k - q_k d \tag{2.4}$$

in which p_k is the partial remainder.

There is another recursive relationship of additive normalization division schemes which can be derived as follows: Let

p_0 be the dividend,
d be the divisor,
q be the quotient.

We want the ratio of the dividend to the divisor to approach the quotient, i.e.,

$$\frac{p_0}{d} - q \rightarrow 0.$$

Let $Q_i = \sum_{j=0}^{i} r^{-j} q_j$, $Q_0 = q_0$, $P_0 = p_0$, where q_i is the i^{th} radix-r quotient digit. If the initial conditions are such that $q_0 = 0$, we define the remainder as

$$P_i = p_0 - dQ_i$$

$$= p_0 - dQ_{i-1} - r^{-i} q_i d$$

$$= p_{i-1} - r^{-i} q_i d. \qquad i \geq 1 \qquad (2.5)$$

Defining the shifted partial remainder as $p_i = r^i P_i$, Equation (2.5) becomes

$$r^{-i} p_i = r^{-(i-1)} p_{i-1} - r^{-i} q_i d$$

or

$$p_i = r p_{i-1} - q_i d$$

or

$$p_{i+1} = r p_i - q_{i+1} d \qquad 0 \leq i \leq m-1 \qquad (2.6)$$

Therefore, the partial remainder p_{i+1} is found by subtracting a multiple of the divisor from the previous partial remainder shifted left one digital position. Which multiple of the divisor is to be subtracted is determined by the quotient digit.

There exists some restrictions for using the recursive relationship (2.6) , where

i is the index of recursion,
m is the number of digits in the quotient,
p_i is the partial remainder,
p_0 is the dividend,
p_m is the remainder,
q_i is the i^{th} quotient digit in radix-r,
d is the divisor normalized to $a < d < b$
rp_i is the shifted partial remainder.

The radix point is to the right of q_0. The restrictions are as follows:

(1) $q_i \varepsilon \{0,\pm 1,\pm 2,\ldots,\pm n\}$ for $i \geqslant 0$, where

$$(1/2)(r-1) \leqslant n \leqslant r-1 \qquad (2.7)$$

$$q_0=0.$$

(2) The dividend, P_0, must be normalized to the domain defined by

$$|P_0| \leqslant \rho|d| \qquad (2.8)$$

where ρ is a measure of redundancy in the representation of the quotient, and it is called redundancy ratio. Redundancy ratio is given by

$$\rho = \frac{n}{(r-1)} \qquad (2.9)$$

(3) The divisor must be normalized to a given domain defined by quantities a and b as follows:

$$a \leqslant |d| < b \qquad (2.10)$$

(4) Every q_{i+1} must be selected such that Equation (2.6) yields p_{i+1} in the range

$$|p_{i+1}| \leqslant \rho|d| \quad \text{for} \quad i \geqslant 0 \qquad (2.11)$$

These restrictions are stated in [6]. The derivation of Equation (2.9) and the restrictions are given in reference [4].

2.3 The P-D Plot

By solving the recursive relationship (2.6) for rp_i we obtain,

$$rp_i = p_{i+1} + q_{i+1}d$$

For a fixed quotient digit, the upper limit of rp_i as a function of the divisor, d, occurs when p_{i+1} is maximum, i.e. when

$$p_{i+1} = \rho d$$

thus

$$rp_{i_{max}} = (\rho + q_{i+1})d \qquad (2.12)$$

likewise, the lower limit occurs with

$$p_{i+1} = -\rho d$$

thus

$$rp_{i_{min}} = (-\rho + q_{i+1})d \qquad (2.13)$$

These linear equations may be plotted as a function of d with q_{i+1} as a parameter ranging from -n to + n in steps of one. The area between $rp_{i_{max}}$ and $rp_{i_{min}}$ for a given $q_{i+1}=j$ will be denoted the q(j) area. The plot of partial remainder versus divisor is called as P-D plot.

The division procedure is now determined: A given value of divisor, d and the i^{th} shifted partial remainder will specify a point in a q(j) area. The digit j will be the value of the next quotient digit q_{i+1} which in turn is used in forming the next partial remainder. In this representation the redundancy is manifested as the overlapping q(j) regions, i.e. some pairs of d and rp_i will specify a point for which either $q_{i+1} = j$ or $q_{i+1} = j-1$ is a valid choice. Two adjacent quotient digit value regions overlap when $\rho > 1/2$. In this case, stairstep boundaries can be drawn between the quotient digit value regions as shown in Figure 2.1. Then, the P-D plot can be partitioned into disjoint equal size rectangles, and a proper quotient digit value can be assigned to each rectangle. Hence, the quotient digits can be determined from the estimates of the divisor and the partial remainder. If the estimates of the divisor and the partial remainder are represented as $\hat{d}$ and $\hat{rp}_i$ and if they have δ and ε binary digits to the right of the radix point respectively, then increments in $\hat{d}$ and $\hat{rp}_i$ are:

$$\Delta d = 2^{-\delta} \qquad (2.13)$$

$$\Delta rp = 2^{-\varepsilon} \qquad (2.14)$$

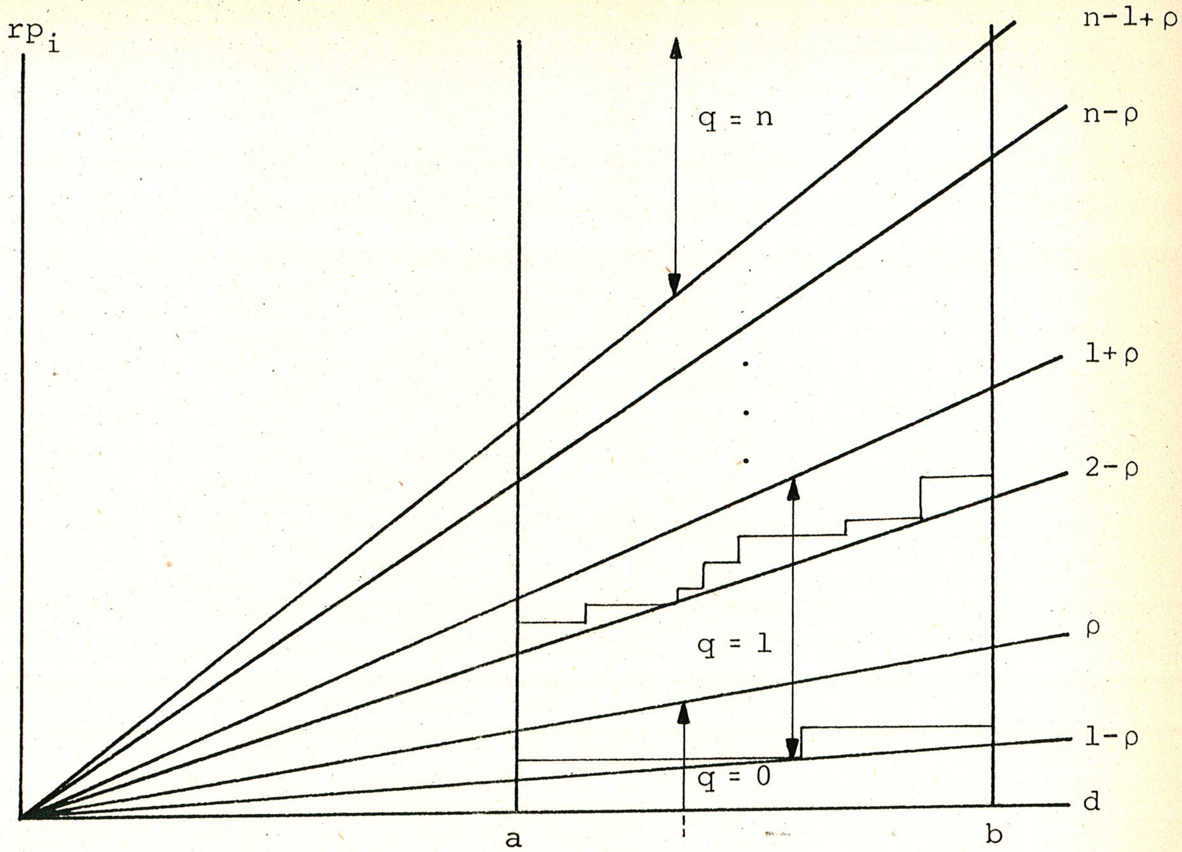


Figure 2.1. The P-D plot

respectively. Hence

$$\hat{d} \in \{ \ell\Delta d, \ell \text{ is an integer}\}$$

$$\hat{rp} \in \{ k\Delta rp, k \text{ is an integer}\}$$

A given value of $\hat{d}$ is representative of full-precision divisors, d, in the range

$$\hat{d}-\alpha \leqslant d \leqslant \hat{d}+\beta \qquad (2.15)$$

where

$$\alpha = \alpha'\Delta d \qquad (2.16)$$

$$\beta = \beta'\Delta d \qquad (2.17)$$

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

$$\hat{rp}-\lambda \leqslant rp \leqslant \hat{rp}+\gamma \qquad (2.18)$$

where

$$\lambda = \lambda'\Delta rp \qquad (2.19)$$

$$\gamma = \gamma' \Delta rp \tag{2.20}$$

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

A quotient digit value j, in the P-D plot, is represented by the area bounded by the lines $d = a$, $d = b$, and the stairstep boundaries in the overlap regions between $q = j+1$, $q = j$ and $q = j$, $q = j-1$. This area can be covered by a boolean function of the binary representations of $\hat{d}$ and $\hat{rp}$, f_j, called " j^{th} quotient digit function". The quotient digit functions are mutually exclusive, i.e. only one function is true for each access to the QS table.

The structure of additive normalization division schemes is composed of mainly three units:

(1) The quotient selector which is the implementation of quotient selector table,

(2) The divisor multiple formation unit which forms dq_{i+1},

(3) The partial remainder formation unit which forms the next partial remainder by using Equation (2.6).

2.4. Design of the Quotient Selector Table

The design parameters of the combinatorial quotient selector table are as follows: The radix of the quotient selector, r; the redundancy ratio, ρ; the limits of the domain of the divisor, a and b; and $\alpha', \beta', \lambda', \gamma', \Delta d$, Δrp. $\alpha', \beta', \lambda', \gamma'$ depend on the representation of the partial remainder and the divisor, on which quadrants the P-D plot are implemented, and the method of estimate formation (truncation with or without rounding).

In Kalaycıoğlu's thesis the values of parameters are determined from the cost performance figures, and according to this reasoning the domains for r, n, ρ are as follows:

Parameters	Domain
r	2^k, k is an integer
n	an integer value given by $1/2(r-1) \leq n \leq r-1$
ρ	$1/2 \leq \rho \leq 1$

The divisor, d is normalized to an arbitrary domain, commonly used domains are

$$1/2 \leq d < 1 \qquad (a = 1/2,\ b = 1) \text{ or}$$

$$3/4 \leq d < 9/8 \qquad (a = 3/4,\ b = 9/8)$$

The second domain yields less expensive QS table.

Since the divisor is given in non-redundant form, truncation without rounding results in

$$\alpha' = 0 \qquad \beta' = 1$$

The values for λ', γ' are given in Table 2.1.

Table 2.1. The smallest sufficient values of λ' and γ'

Representation of partial remainder	1's or 2's complement λ'	γ'	Sign-magnitude λ'	γ'	Sign-digit λ'	γ'
Non-redundant	0	1	0	1	-	-
Redundant	0	2	1	ℓ_1	ℓ_2	ℓ_2

$$\ell_1 = \begin{cases} 1 & \text{if 1 is added to the least significant position of the estimate of the partial remainder obtained in an adder when the sign of the operands are the same,} \\ 2 & \text{otherwise} \end{cases}$$

$$\ell_2 = \begin{cases} 1 & \text{if the partial remainder is in non-redundant form} \\ 2 & \text{if the partial remainder is in the specified redundant form.} \end{cases}$$

The values of Δd and Δrp are determined from worst case bounds. The powers of two that are less than the bounds can be chosen as values for Δd and Δrp. However, the largest such powers of two are chosen since the larger Δd and Δrp, the less expensive is the QS table.

Atkins [4] derived bounds to determine Δd and Δrp based upon a geometric argument. However, this approach will produce incorrect results (large Δd, Δrp) due to the failure of the method to account quantizing effects. In reference [6] he revised this approach and he derived equations for the analytic derivation of Δd, Δrp.

Kalaycıoğlu proposed another method in reference [7]. This proposition is given below:

Proposition

The largest sufficient value of Δd as a function of Δrp or the largest sufficient value of Δrp as a function of Δd can be found as follows.

(i) Depending on the representation of the divisor and partial remainder, substitute the proper values of $\alpha', \beta', \lambda', \gamma'$,

(ii) substitute the resultant identities in inequalities

$$\lambda+\gamma \leqslant (2\rho-1)(a-\alpha) \tag{2.21}$$

$$\alpha+\beta \leqslant \frac{(2\rho-1)(a-\alpha)-(\lambda+\gamma)}{(n-\rho)} \tag{2.22}$$

(iii) find the largest value of Δd as a function of Δrp, or the largest value of Δrp as a function of Δd that satisfies the inequalities simultaneously.

The proof of the proposition is given in Appendix B of reference [7].

2.5. Algorithms for the Construction of the QS Table

The algorithms are designed to produce boolean function definitions of the quotient digit functions for the implementation of the QS table by using combinatorial logic. In reference [6], Atkins suggested an algorithm to construct a QS table completely by using combinatorial logic (Fig.2.2).

As the radix of the QS table increases, the number of terms in the definition of the quotient digit functions increases greatly. Then, it becomes useful to partition the terms into "pages" when the table is implemented by using memories. Each page contains the terms that cover a column in the P-D plot. For a division operation, the page corresponding to column that the divisor falls in is brought to the memory and it is used during the division operation. Kalaycıoğlu in his work [7], modified and extended Atkins algorithm for columnwise implementation of the QS table. The block diagram of this algorithm is given in Figure2.3.

Again in[7],to construct QS Table a third algorithm which is called as columnwise construction algorithm is presented (Fig.2.4). This algorithm is composed of several subalgorithms. The first one is called "boundary selection sub-

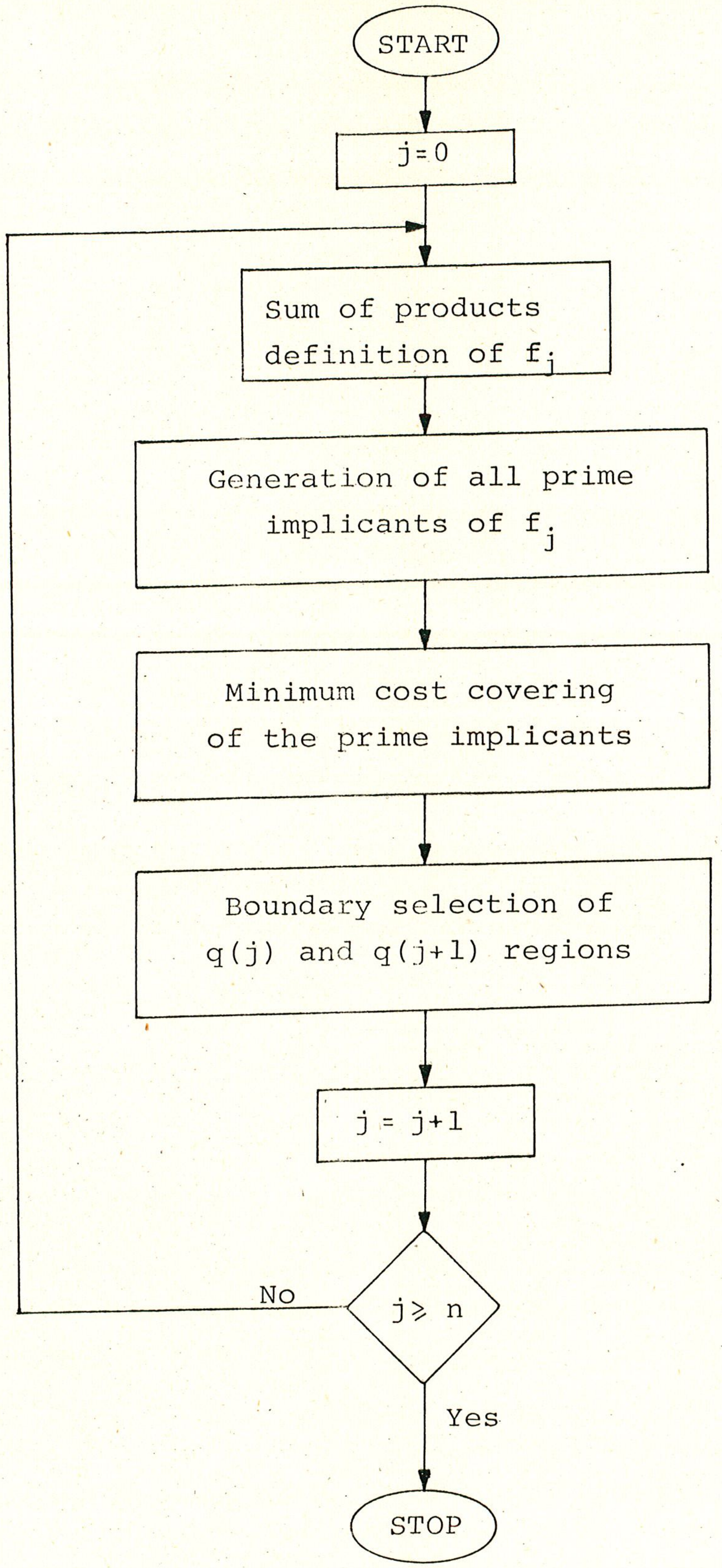


Figure 2.2. Atkins' algorithm

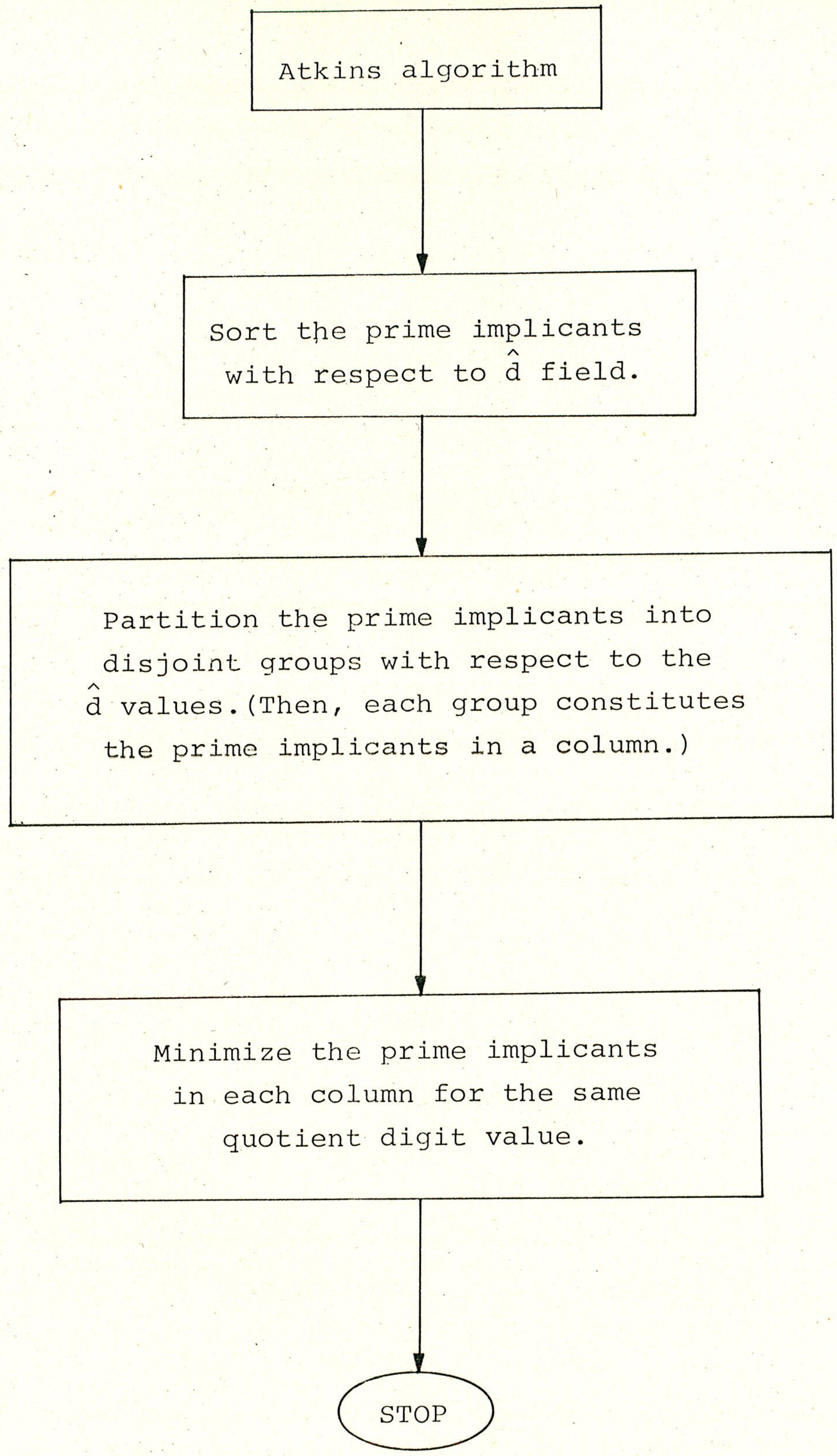


Figure 2.3. Modified Atkins' algorithm for columnwise implementation

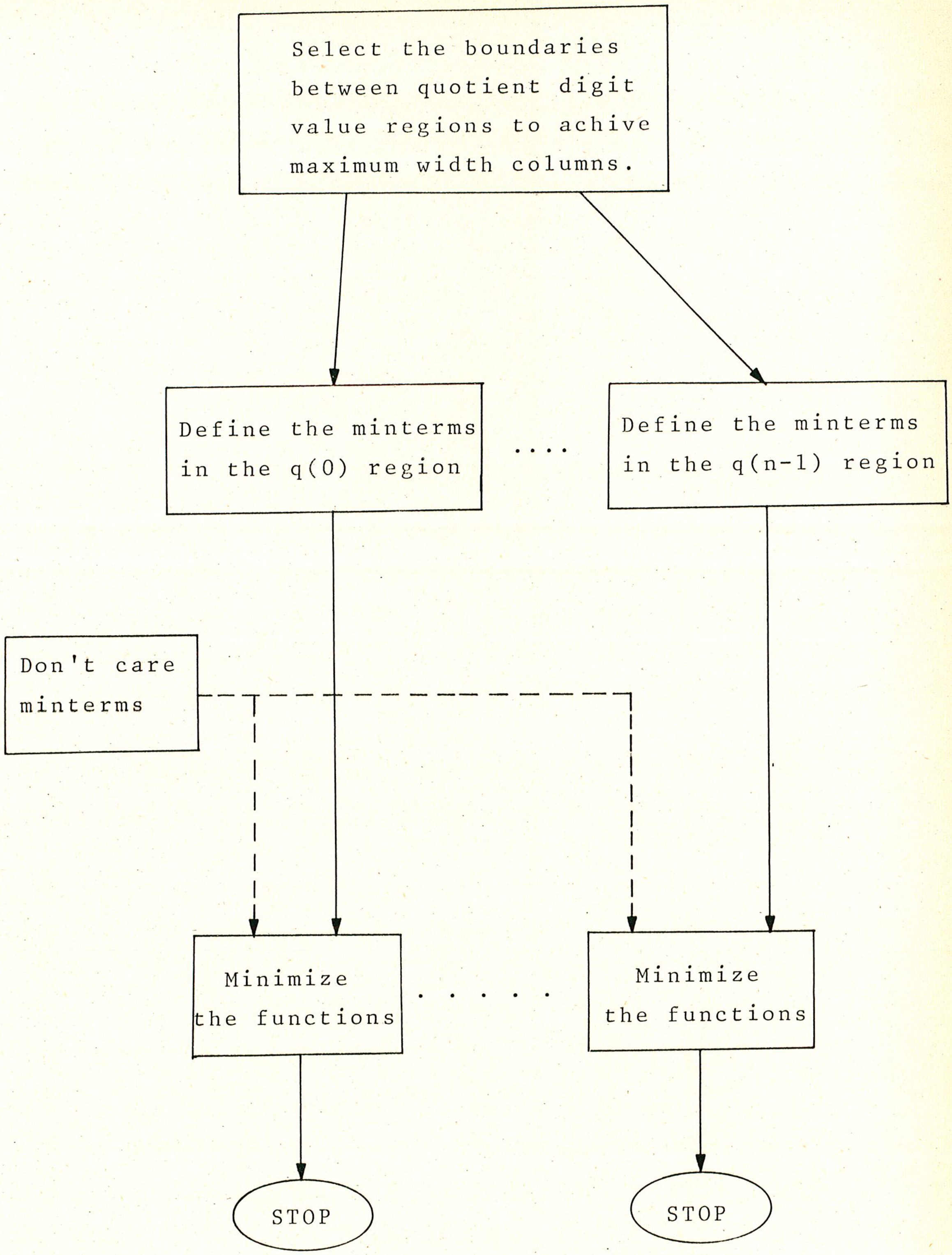


Figure 2.4. The columnwise construction algorithm

algorithm". It iteratively constructs the boundaries of the quotient digit value regions starting at $d = a$ and preceding towards $d = b$ in maximum possible width columns. The rp boundaries in in a column are also called "comparison constants" since the value of a quotient digit can be determined by comparing the estimate of the partial remainder with them.

The second subalgorithm (minterm definition subalgorithm) supplies the set of minterms that are within the boundaries determined by the first subalgorithm to the prime implicant generation subalgorithm for each quotient digit function separately. The minterms outside of the boundaries of quotient digit value regions -n through n are supplied as don't-cares to the prime implicant generation subalgorithm to produce prime implicants with a minimum number of literals. Finally, the minimum number of prime implicants are selected by the covering subalgorithm.

For larger radices, the number of operations increases and computer assistance is required for all algorithms.

The minimization subalgorithm is the most costly part of the procedure. In the computer aided designs, the amount of CPU time and space used is largely a function of the number of minterms supplied to the prime implicant generation program. The cost of the QS table, in terms of the total number of minterms, is proportional to $r(r-2)$ [7]. Hence the cost of the application of the algorithms should be approximately proportional to the square of the radix.

2.6. Radix-r^2 Quotient Selector

The iteration time in an additive normalization division scheme can be reduced by concurrently producing quotient digit and the partial remainder. This is accomplished with lookahead techniques which find the quotient digit one iteration earlier.

The quotient digits can be found one iteration earlier by using a quotient selector which consists of a radix-r^2 QS table and a subtracter as shown in Figure 2.5.

The radix-r^2 QS table produces a radix-r^2 quotient digit, Q_{i+1}, which has a value $rq_i + q_{i+1}$. The subtracter

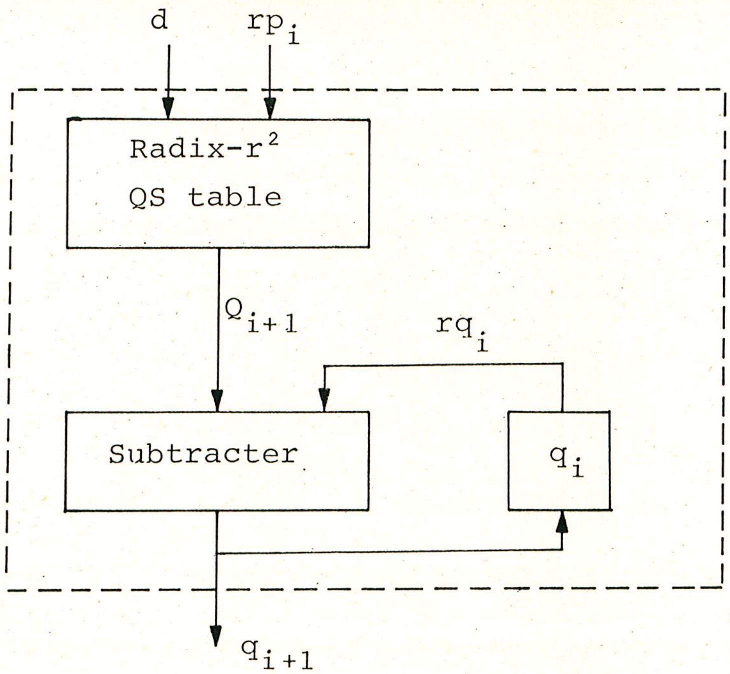


Figure 2.5. Radix-r^2 quotient selector

yields the value of q_{i+1} by subtracting the shifted previous quotient digit q_i from Q_{i+1}.

Let h be the function realized by the radix-r^2 quotient selector:

$$h(rp_i, d) = q_{i+1} \quad \text{for} \quad i \geq -1 \tag{2.23}$$

where

$$p_{-1} = \frac{1}{r} p_0 \tag{2.24}$$

Then in the i^{th} iteration q_{i+1} and p_{i+1} can be formed concurrently by the following operations:

(a) $q_{i+1} = h(rp_i, d)$

(b) $\frac{1}{r} p_{i+1} = p_i - q_i d$

The second term at (b) $(-q_i d)$ is formed in the divisor multiple formation block either in iteration i or i-1.

Another concurrent operation is also possible with a division structure which uses radix-r quotient selector table. This can be accomplished as follows. In the i^{th} iteration, the estimate of p_{i+1} is found from the estimates of p_i and $-dq_i$ by using an adder which is short and fast relative to the adder in the partial remainder formation unit. It then finds q_{i+1} from the estimates of p_{i+1} and d, while p_{i+1} is found from p_i and $q_i d$.

2.7. Alternate Division Algorithms

In reference [7] , four different type division structures (QS1, QS2, QS3, QS4) with four variations in the quotient selector are presented.

The aspects for the structure of the quotient selector are

(1) The choice of radix: r or r^2

(2) Method of forming estimate of the partial remainder: Truncation of full precision partial remainder or computation based on estimates of results in the previous iteration. Figure 2.6 shows four types of quotient selectors.

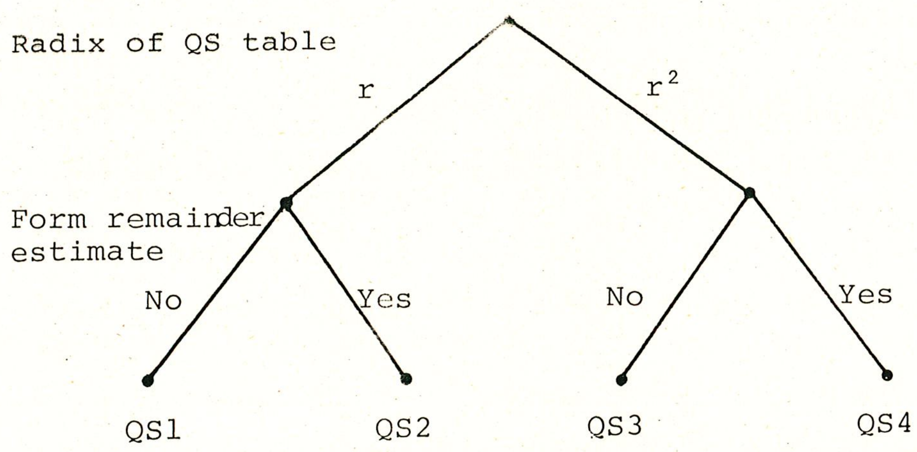


Figure 2.6. Variations in the quotient selector

The structures of the division schemes corresponding to the four types of quotient selectors and their relative timing diagrams are shown in Figures 2.7 through 2.14. The timing diagrams show the degree of concurrency in the operations of the three main units, namely the quotient selector, the partial remainder unit and the divisor multiple formation unit.

2.7.1. The QS1 Type Structure

In this structure, there is no concerrency in the operation of units. The structure uses Recursive Relationship (2.6). At the beginning of the i^{th} iteration, the estimates of the divisor and the partial remainder are supplied to the quotient selector to determine q_i. The multiple of the divisor, according to the value of q_i, is selected in the divisor multiple formation (DMF) unit. The multiple of the divisor and the i^{th} partial remainder are supplied to the partial remainder formation (PRF) unit and the next partial remainder computed and stored in a register. One iteration time in this structure is given by

$$t_1 = t_{QS1} + t_{DMF} + t_{PRF}$$

where t_1 is the delay time of the corresponding structure.

2.7.2. The QS2 Type Structure

This structure permits concurrency of operations in the PRF unit and the QS unit. Hence, the quotient digit is found one iteration earlier. In the i^{th} iteration the following two operations are performed.

(1) p_{i+1} is formed in the PRF unit from p_i and the multiple of the divisor, dq_i (which was stable at the output of the DMF unit at the end of the $(i-1)^{st}$ iteration); and it is stored in register p.

(2) In the QS2, the next estimate of the partial remainder $\hat{p}_{i+1}$ is formed from some estimates of dq_i and p_i in a partial remainder estimate formation (PREF) block.

The structure of the PREF block depends on the specific division scheme. When only truncation is applied, the PREF block is a data path which transfers the higher order bits of the divisor and partial remainder to the QS table. If the partial remainder is in the redundant form or if $r\hat{p}_{i+1}$ is obtained from the estimates of the previous partial remainder and a divisor multiple, the PREF block consists of an adder of length $\log_2 \frac{rb}{\Delta rp}$ bits [7]. The QS2 table produces q_{i+1} and stores it in register q. At the end of the iteration, the multiple of the divisor determined by q_{i+1} stabilizes at the output of the DMF unit. The iteration time for QS2 type structure is

$$t_2 = \max(t_{PRF}, t_{QS2} + t_{DMF})$$

2.7.3. The QS3 Type Structure

This structure has a radix-r^2 quotient selector table with a small and fast subtracter. The designer has the choice of executing the DMF unit concurrently either with the QS or with the PRF unit. Figure 2.11 shows the case for the second choice. This structure does not require the PREF block in QS2, hence the quotient digit calculation time is considerably reduced. One iteration time is given by

$$t_3 = \min\left[\max(t_{QS3} + t_{DMF}, t_{PRF}), (\max(t_{QS3}, t_{DMF} + t_{PRF})\right]$$

t_3 can be further reduced by partitioning the QS or the DMF unit into smaller blocks and placing these blocks in the concurrently operating branches. For example, the QS and the DMF unit can be partitioned as

QS = QS table + subtracter

DMF = decoder + selector

In the case of partitioning the DMF unit, for instance, the decoder can be operated in the QS branch and the selector in the selector in the PRF branch.

2.7.4. The QS4 Type Structure

Greater concurrency in the operation of units can be obtained by combining both of the techniques illustrated in the QS2 and QS3 type structures. In this case, the quotient selector is composed of the PREF block, the radix-r^2 quotient selector table and the subtracter. In QS4, the PREF block and the QS table are operated concurrently rather than serially as in QS2. Hence in the QS4 type structure the three units, namely the PRF unit, the QS table and the PREF block, operate concurrently. In this structure, the DMF unit can be concurrently executed either with QS unit and the subtracter or with PREF block. Figure 2.13 shows the case for the first choice. In this structure the PRF unit or QS unit (or both) can operate continuously without the intervening time gaps as shown in Figure 2.14. One iteration time is given as

$$t_4 = \min\left[\max(t_{QS4}, t_{DMF}+t_{PREF}, t_{PRF}), \max(t_{QS4}+t_{DMF}, t_{PREF}, t_{PRF})\right]$$

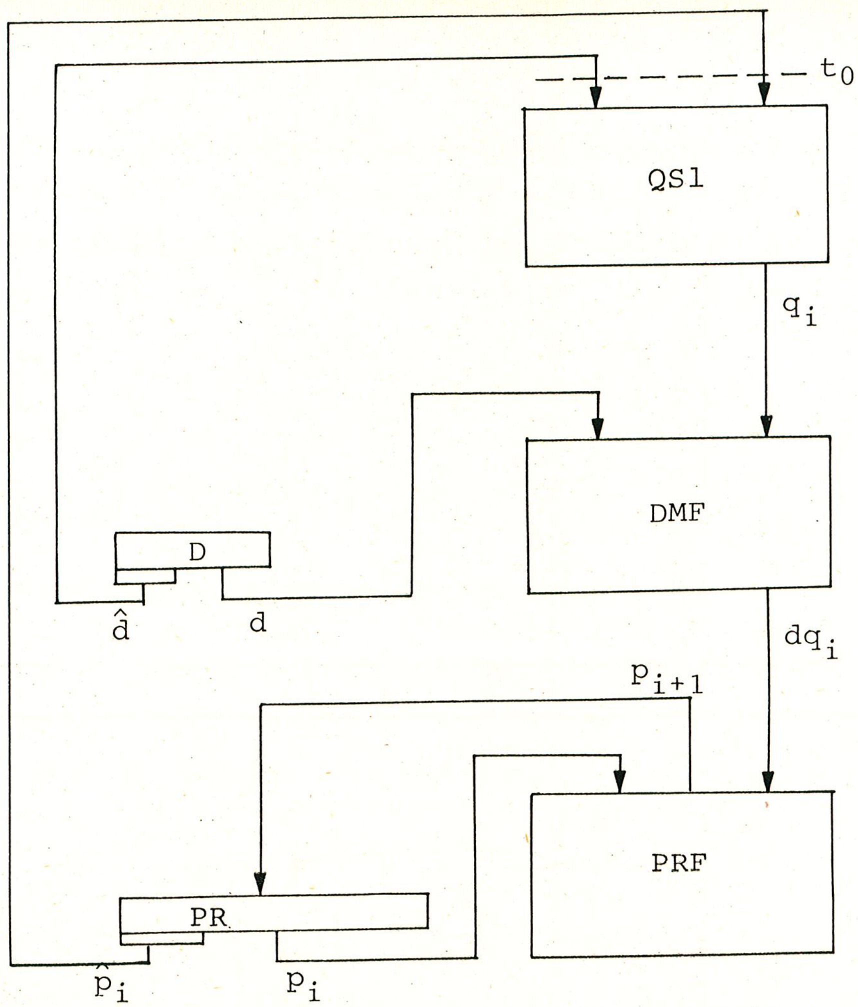


Figure 2.7. Block diagram of QS1 type structure

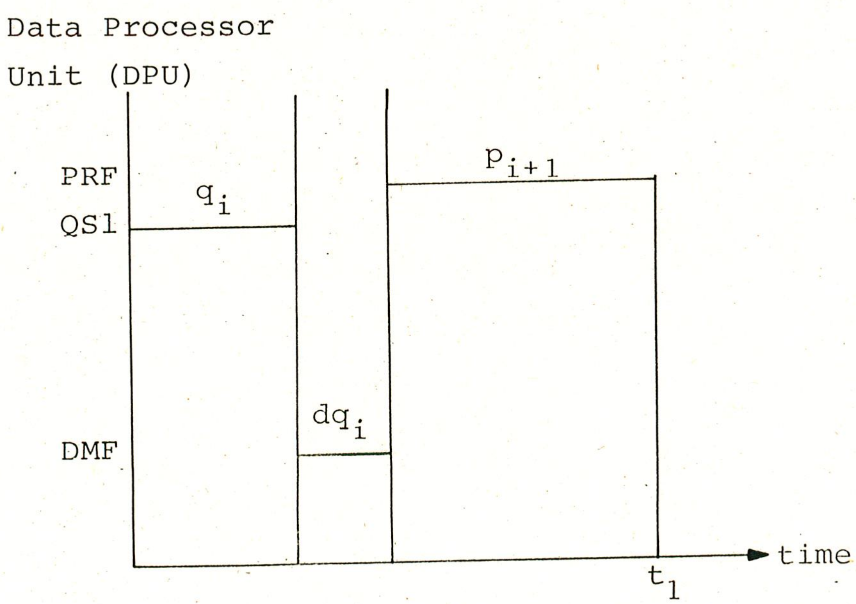


Figure 2.8. Timing diagram of QS1 type structure

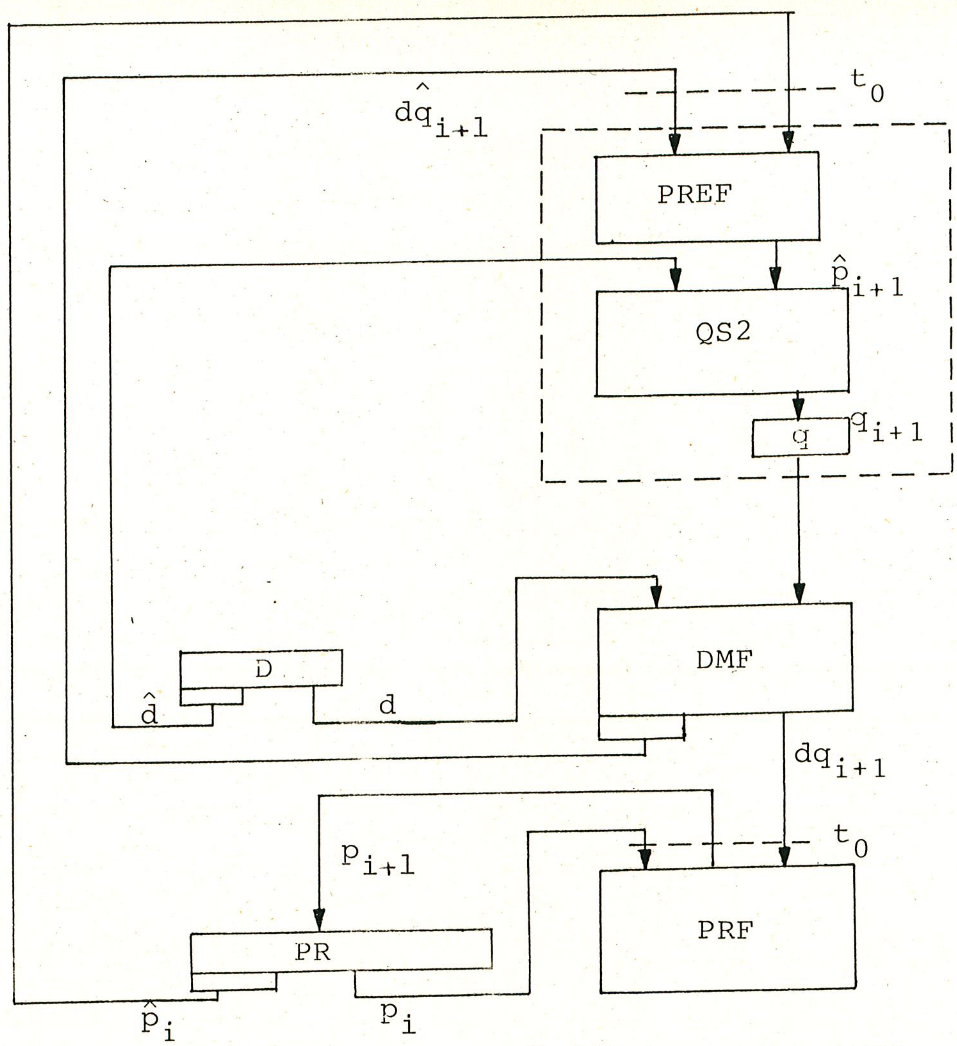


Figure 2.9. Block diagram of QS2 type structure

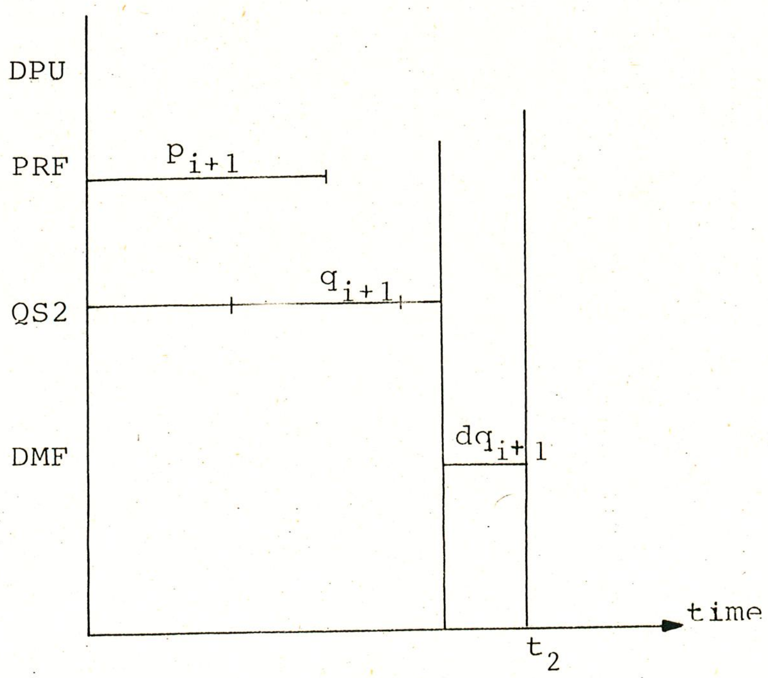


Figure 2.10. Timing diagram of QS2 type structure

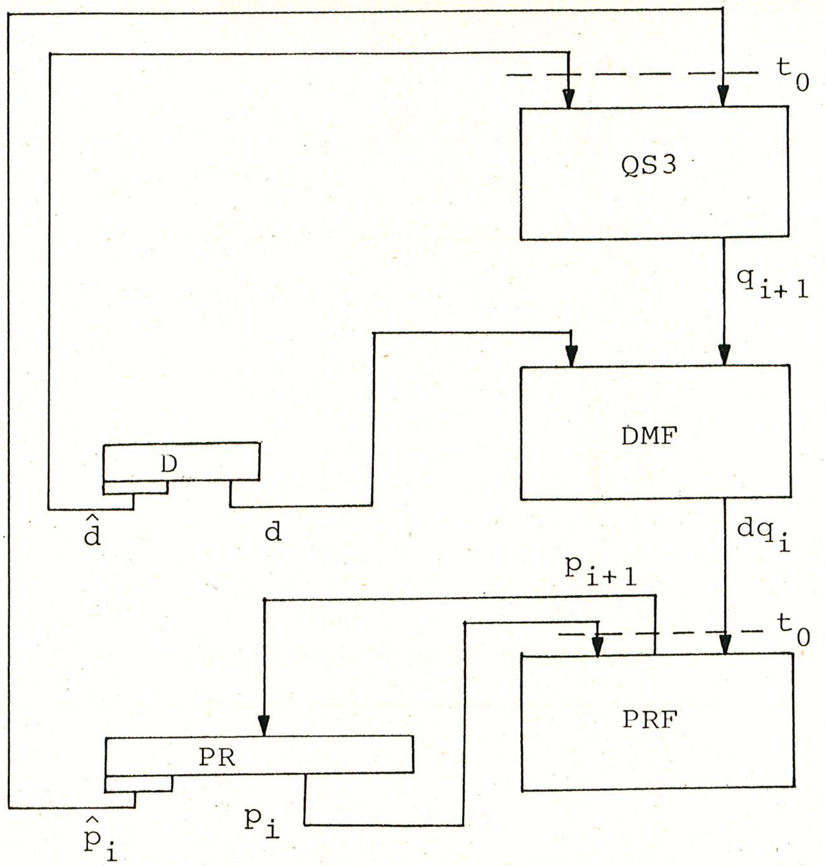


Figure 2.11. Block diagram of QS3 type structure

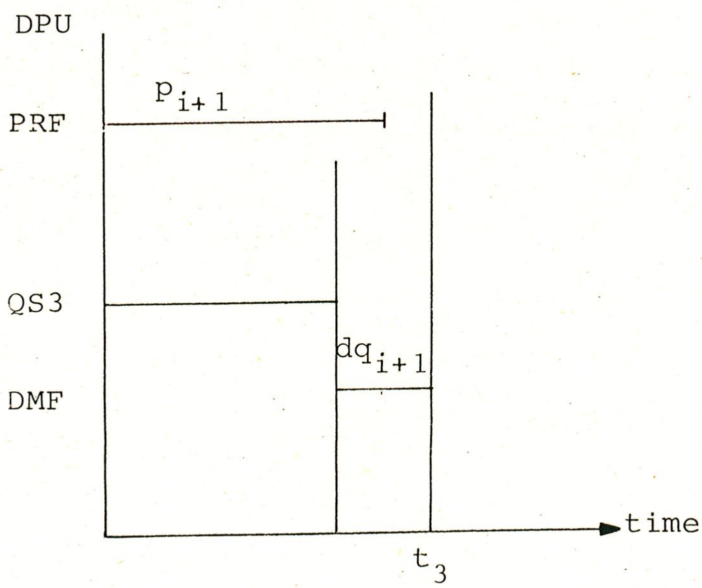


Figure 2.12. Timing diagram of QS3 type structure

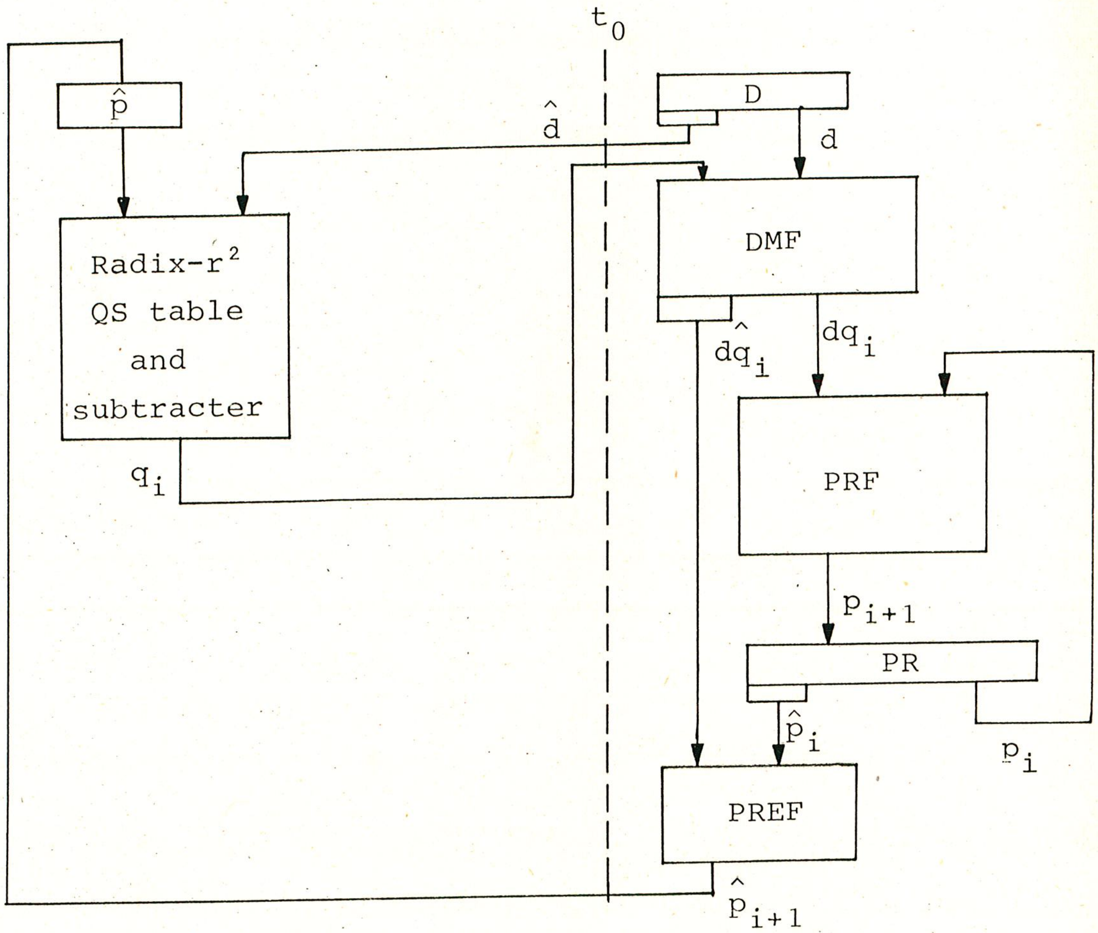


Figure 2.13. Block diagram of QS4 type structure

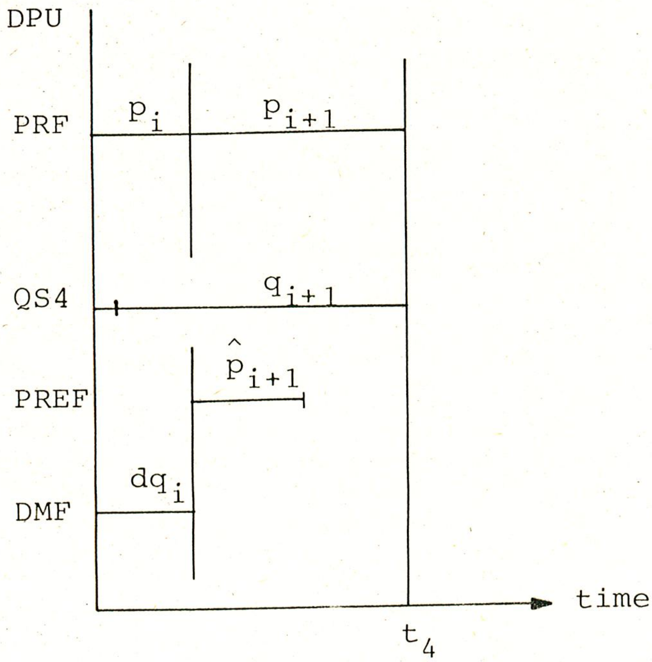


Figure 2.14. Timing diagram of QS4 type structure

CHAPTER 3

DESIGN OF THE QS1 AND QS2 TYPE STRUCTURES

The logic design of a digital system is a process for deriving the digital circuits that perform data processing and the digital circuits that provide control signals. The relationship between the control and the data processor in a digital system is shown in Figure 3.1.

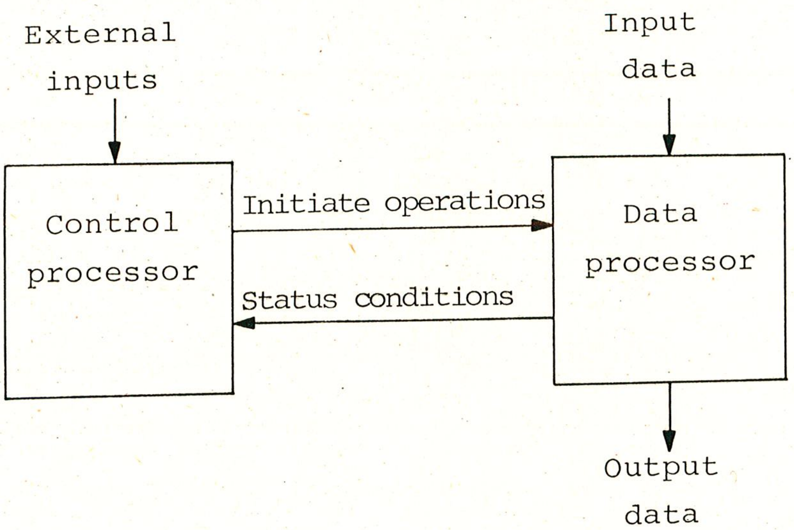


Figure 3.1. The block diagram of a digital system

The data processor part consists of registers and associated digital functions. The control processor initiates all operations in the data processor. The control processor is a sequential circuit whose internal states dictate the control functions for the system. At any given time, the state of the sequential control initiates a prescribed set of operations. Depending on the present state and status conditions or inputs the sequential control goes to the next state to initiate other operations. Thus, the digital circuit that acts as the control processor provides a time sequence of signals for initiating the operations in the data processor part of the system.

The control processor of the QS1 and QS2 type structures is designed by using one flip-flop per state method of hard-wired control [10]. The design parameters are selected as $r = 4$, $n = 3$, $\rho = 1$. Thefore, the quotient digits can take values -3,-2,-1,0,1,2,3.

3.1. Design of QS1 Type Structure

The initial equipment configuration is as shown in Figure 3.2.

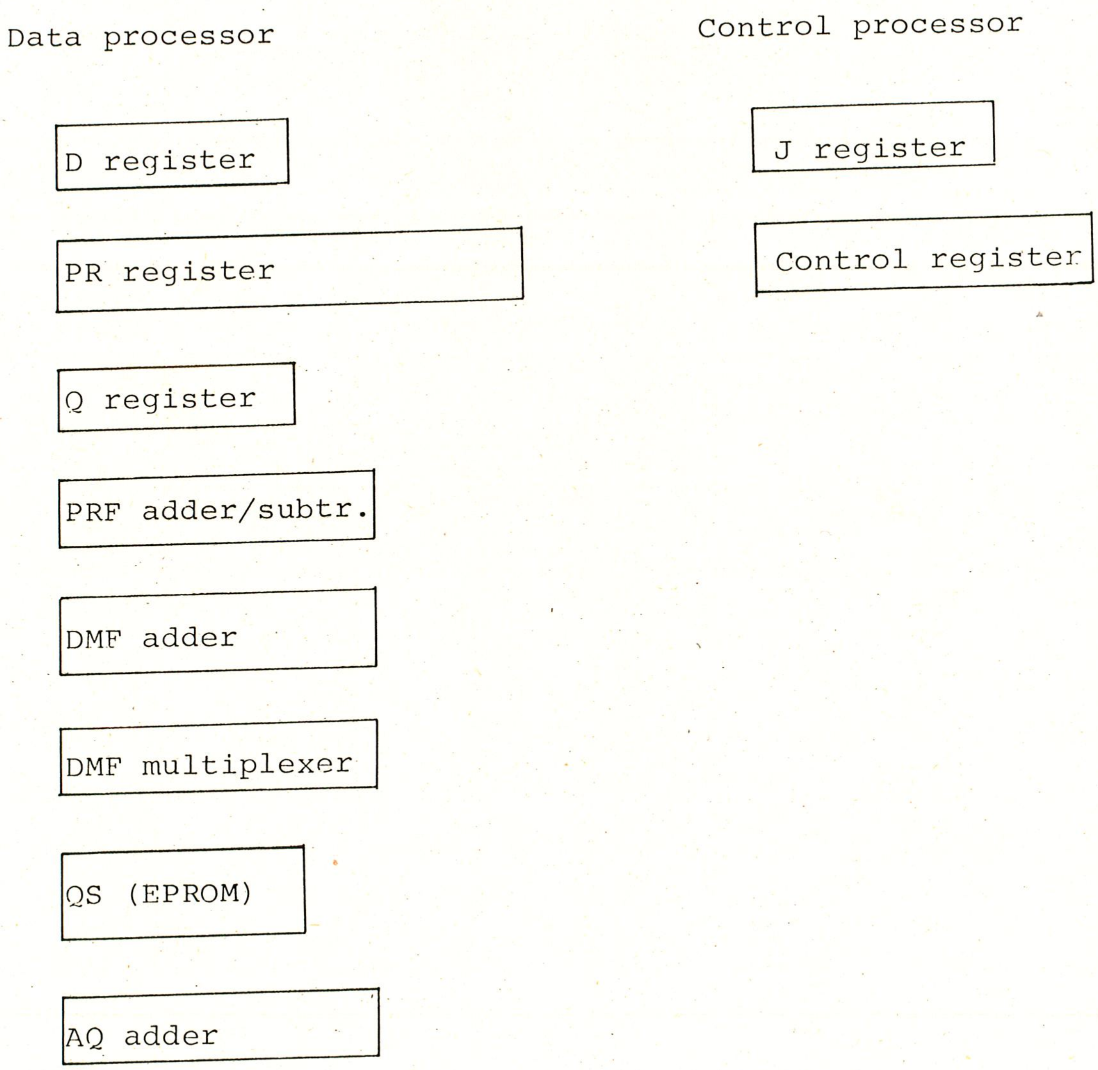


Figure 3.2. The equipment configuration of QS1 type structure

3.1.1. Control Processor of QS1 Type Structure

The flowchart of the algorithm is shown in Figure 3.3.

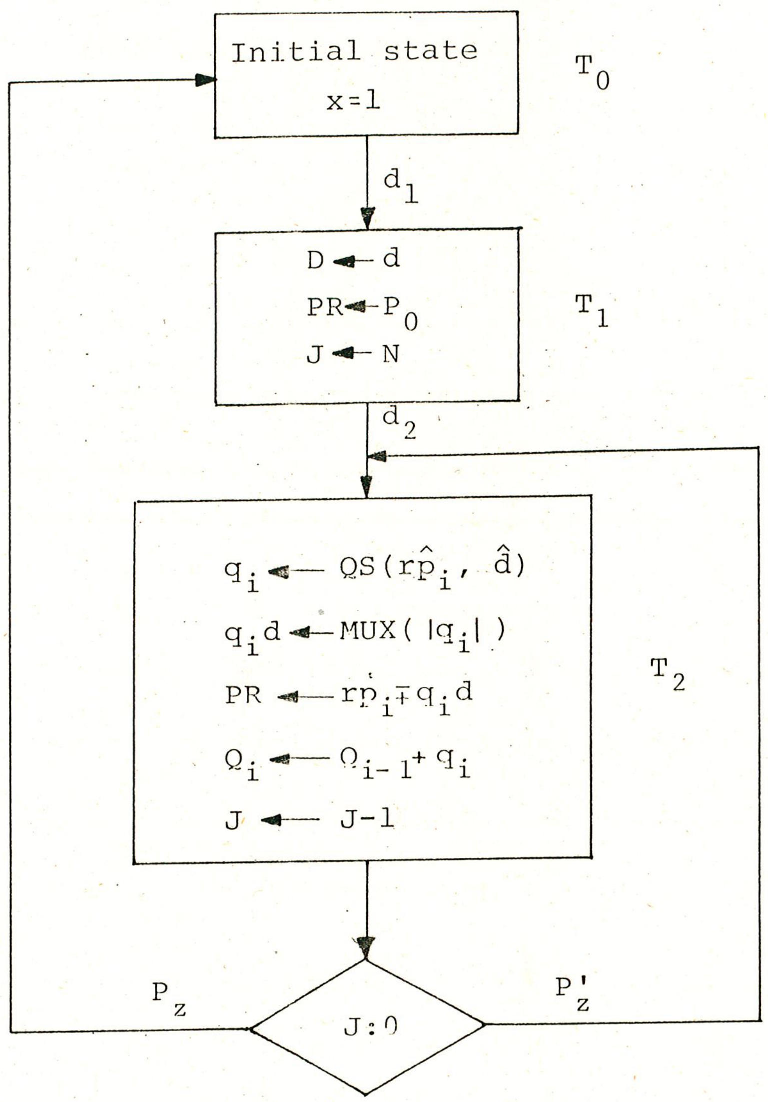


Figure 3.3. The flowchart of the algorithm for the QS1 type structure

T_0 is the initial state. x is the output of the control processor, x = 1 means that the circuit is ready for a new data, the previous operation is finished. The external input which starts the division operation is d_1. In this work, d_1 input is obtained from a pushbutton. At state T_1, the dividend P_0, is placed into register PR. The divisor d, is placed into register D. The number of quotient digits N, which will be found during the division operation, is placed into register

J (The value of N depends upon the length of PR and D registers). Another external input d_2 starts the T_2 state. The necessity of the existance of d_2 comes from the following fact. Since d_1 is obtained from a pushbutton, the division operation will probably be finished before d_1 becomes zero. Therefore, the division operation starts again with the same data and the correct result can not be obtained. This problem is solved by using another external input d_2.

At state T_2, the quotient digit, whose value depends upon the estimates of rp_i and d, is selected. The divisor multiple formation unit selects the $q_i d$ value according to the value of the quotient digit. At PRF unit, the output of DMF unit (which is a multiplexer) is either added or subtracted depending upon the sign of the quotient digit. Since the quotient selector is connected combinatorially with accumulate quotient (AQ) adder, the value of q_i is ready for accumulation at quotient (Q) register. The quotient digit (which is in sign-magnitude form) is converted to 2's complement representation and added to the accumulated quotient. The value at the J register (which is a count-down counter) is decremented by one. At the test block, P_z is obtained. $P_z = 1$ means that J register is equal to zero, if $P_z = 0$ then the next iteration starts, otherwise the division operation will be terminated. If $P_z = 1$ the control goes to initial state T_0.

The control state diagram is shown in Figure 3.4.

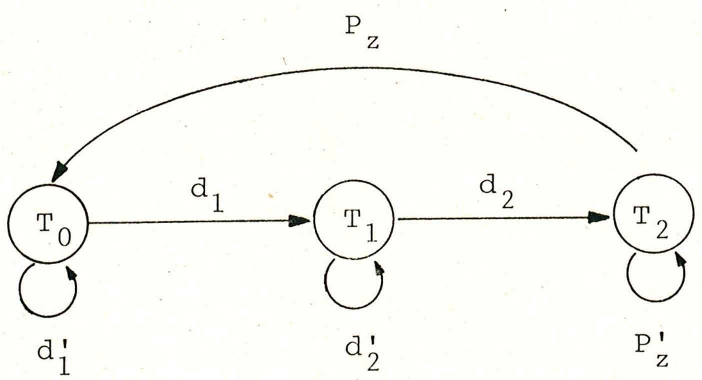


Figure 3.4. The state diagram of QS1 type structure

The control processor consists of three flip-flops and necessary gates for the flip-flop input functions. T_0, T_1 and T_2 are the outputs of these flip-flops. The following flip-flop input functions are obtained by using D flip-flops:

$$DT_0 = d_1' T_0 + P_z T_2$$

$$DT_1 = d_2' T_1 + d_1 T_0$$

$$DT_2 = P_z' T_2 + d_2 T_1$$

In this study, the length of the dividend is taken as 24 bits and the divisor 12 bits. Therefore, the quotient digit is also 12 bits long. At each iteration, two bits of quotient are found. Hence, six iterations are necessary and six is placed into register J. In eight clock pulse duration the division operation is terminated: One pulse for T_0, one for T_1 and six pulses for iterations.

The clock inputs for all registers are determined as follows:

$T_1 Cl \longrightarrow$ Clock input of D register

$(T_1 + T_2) Cl \rightarrow$ Clock input of PR register

$T_2 Cl \longrightarrow$ Clock input of J register

$T_2 Cl \longrightarrow$ Clock input of Q register

All registers are leading-edge-triggered type except the PR register. The timing diagrams are shown in Figure 3.5.

3.1.1.1. <u>The J Register</u>

A 74LS191 synchronous up-down counter is used as J register. The counter is fully programmable; that is, a desired binary value can be loaded to it (when the load input is low) by entering the desired data to the data inputs. At state T_1, six is placed to this counter. The load input is connected to T_1'. The content of the counter is decremented by one at each clock pulse at state T_2. It will be zero after six iterations. The output of each flip-flop is connected to the inputs of a four-input NOR gate. The output of the NOR gate (which is called as P_z) will be one when all outputs are zero. Then, the control state goes to the initial state T_0. This counter is triggered at the leading edge of the clock pulse. The wiring diagram of J register and all data processor units are shown in the Appendix.

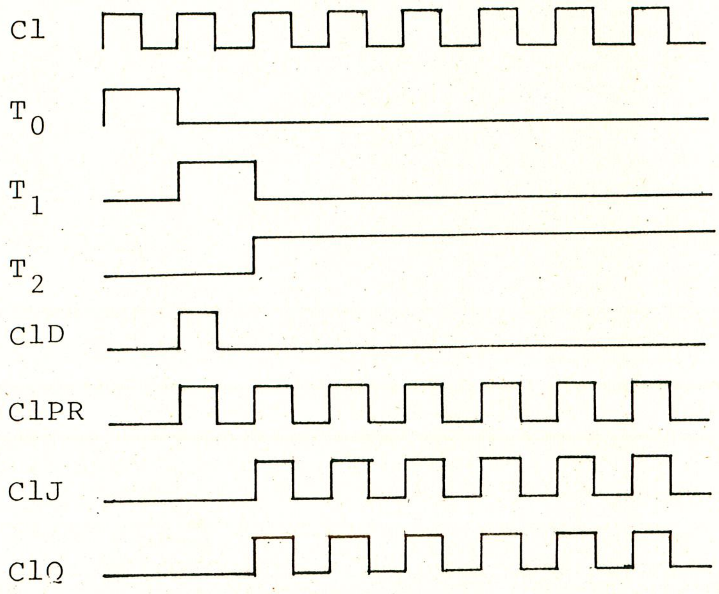


Figure 3.5. The timing diagram of the clock inputs for the QS1 type structure

3.1.2. Data Processor of QS1 Type Structure

The block diagram of the data processor unit is shown in Figure 3.6.

Initialization: At state T_1, the value of divisor (in two's complement representation, normalized into $1/2 \leqslant d < 1$ region) is placed into register D. The register D is composed of two hex-D type flip-flop integrated circuits (74LS174's). In order to obtain the multiple of divisors (2d,3d) three four-bit adder circuit (74LS83) is used. 2d is obtained simply by shifting the content of register D one bit to the left. 3d is obtained by adding d with 2d. Actually, it is not necessary to place 2d and 3d into registers; once d is placed, 2d and 3d are always available combinatorially. The initialization circuit consists of register D and an adder. Figure 3.7 shows the structure of the initialization circuit.

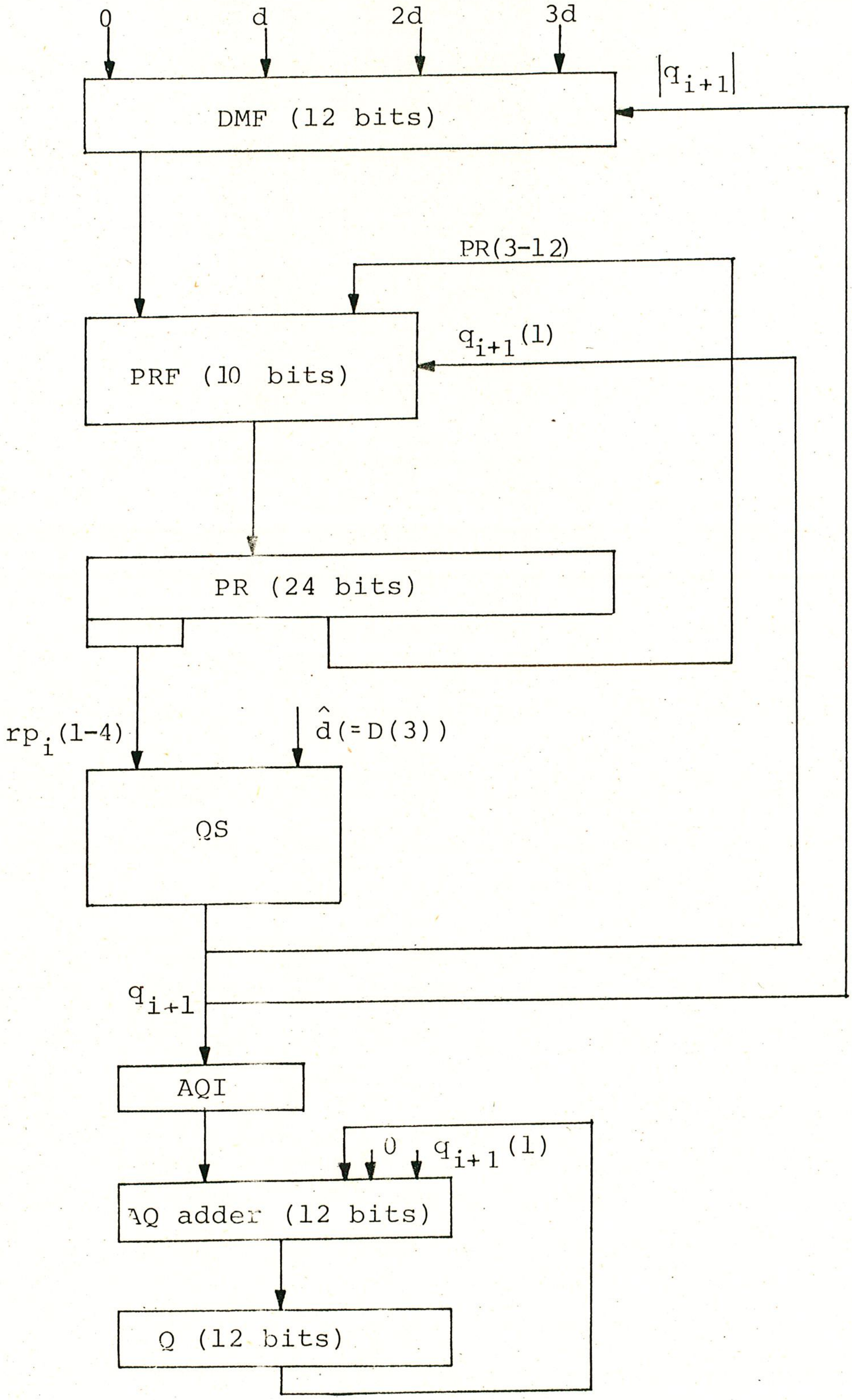


Figure 3.6. The block diagram of the data processor of the QSl type structure

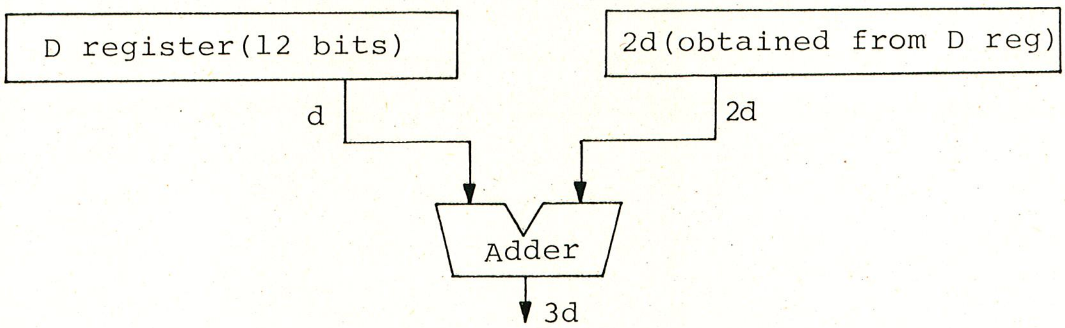


Figure 3.7. The structure of the initialization circuit

3.1.2.1. <u>The Divisor Multiple Formation Unit</u>

DMF unit is simply a multiplexer with inputs 0,d,2d, and 3d. Hence, the multiplexer is a four-input one each input being 12 bits long. Therefore, six 2-bit 4-to-1 line multiplexers (74LS153's) are used. Since the multiplexer is a four input one, two select inputs are available. The magnitude of the quotient digit is send to the select inputs of the multiplexer. If the select inputs are 00, then the output of the multiplexer are all zeros. If it is 01, d is selected. If it is 10, 2d is selected and if it is 11, 3d is selected.

3.1.2.2. <u>The Qutient Selector</u>

The quotient selector is composed of only the quotient selector table. The estimate of the shifted partial remainder is obtained from the high order bits of the PR register hence $\lambda'=0$, $\gamma'=1$. The largest sufficient bounds for Δrp and Δd can be obtained as 1/4 and 1/8 respectively, by the procedure of Proposition. Figure 3.8 shows the P-D plot and the selected boundaries using the procedures in the boundary selection subalgorithm. If the upper boundary of q = 3 region is selected as shown with the dashed lines, then it is possible to have $\Delta rp = 1/2$ and $\Delta d = 1/4$ [7].

The estimates of the partial remainder and the divisor is given as *$s\ p_1\ p_0.p_{-1}$ and $.\_\ d_{-2}$, respectively. The quotient digit is represented in sign-magnitude form as $s_q t_1 t_0$.

The P-D plot is divided into 32 rectangles with $\Delta d = 1/4$ and $\Delta rp = 1/2$. There is a distinct quotient digit .

* S:sign bit, p_1,p_0,(p_{-1}):bits before(after) the decimal point.

for each rectangle.

In the realization of the circuit, a 2 KByte erasable programmable read only memory (EPROM,2716Q) is used as quotient selector unit. The quotient digit values for QS1 type structure is placed between addresses 0-31. The EPROM acts as a combinatorial circuit that realizes quotient digit functions which are obtained with columnwise construction algorithm. Table 3.1 shows the P-D plot but in a different form. It shows the data at each address after programming the EPROM. MM 2716Q type EPROM has 11 address lines and 8 data lines. The first six address lines are connected to ground, following four bits comes from the most significant bits of PR register. The last bit comes from the third bit of the D register. The quotient digits are placed to the least significant three bits of the data word. The first five bits of the data word are all ones because an unprogrammed EPROM is filled with ones.

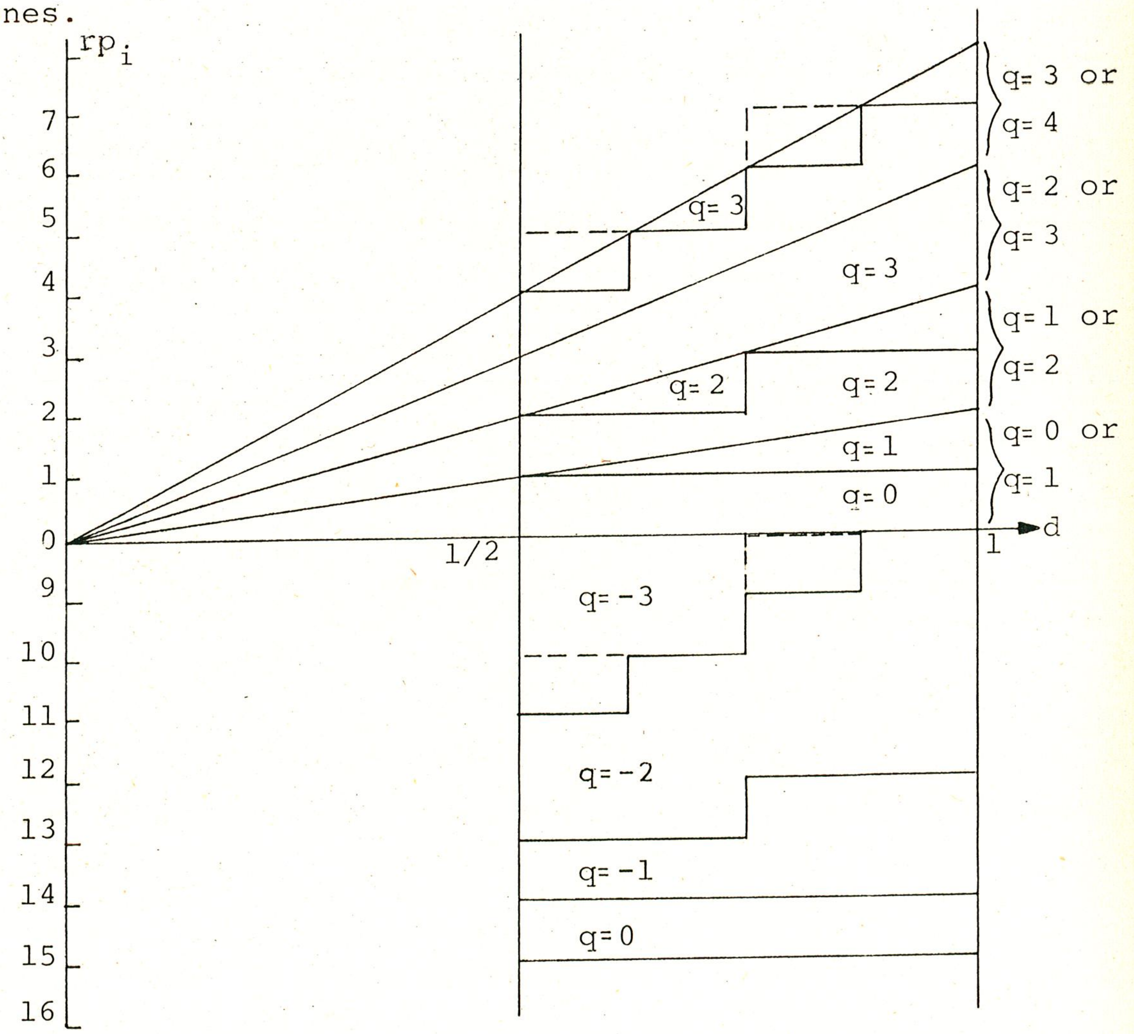


Figure 3.8. The boundaries selected in the QS table of the QS1 type structure

Table 3.1. The QS table for QS1 type structure

$$A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0 \; - \; D_7D_6D_5D_4D_3D_2D_1D_0$$

$$\underbrace{0 \;\; 0\,0\,0\,0\,0\, \underbrace{X\,X\,X\,X}_{\hat{rp}}\, \underbrace{X}_{\hat{d}}}_{\text{address lines}} \; - \; \underbrace{1\,1\,1\,1\,1\,X\,X\,X}_{\text{data lines}}$$

$A_4A_3A_2A_1A_0$	-	$D_2D_1D_0$
0 0 0 0 0	-	0 0 0
0 0 0 0 1	-	0 0 0
0 0 0 1 0	-	0 0 1
0 0 0 1 1	-	0 0 1
0 0 1 0 0	-	0 1 0
0 0 1 0 1	-	0 1 0
0 0 1 1 0	-	0 1 1
0 0 1 1 1	-	0 1 0
0 1 0 0 0	-	0 1 1
0 1 0 0 1	-	0 1 1
0 1 0 1 0	-	0 1 1
0 1 0 1 1	-	0 1 1
0 1 1 0 0	-	0 1 1
0 1 1 0 1	-	0 1 1
0 1 1 1 0	-	0 1 1
0 1 1 1 1	-	0 1 1
1 0 0 0 0	-	1 1 1
1 0 0 0 1	-	1 1 1
1 0 0 1 0	-	1 1 1
1 0 0 1 1	-	1 1 1
1 0 1 0 0	-	1 1 1
1 0 1 0 1	-	1 1 1
1 0 1 1 0	-	1 1 1
1 0 1 1 1	-	1 1 1
1 1 0 0 0	-	1 1 1
1 1 0 0 1	-	1 1 0
1 1 0 1 0	-	1 1 0
1 1 0 1 1	-	1 1 0
1 1 1 0 0	-	1 0 1
1 1 1 0 1	-	1 0 1
1 1 1 1 0	-	0 0 0
1 1 1 1 1	-	0 0 0

3.1.2.3. The PR Register

The PR register is constructed with 4-bit shift registers (7495's). At state T_1, the dividend which is in two's complement representation is placed into the PR register. At state T_2, the output of the PRF unit must be placed to the most significant 10 bits, therefore a two-input one-output, 10-bit multiplexer (MUX1) is used. In this way, at state T_1 the input data, at state T_2 the next partial remainder is loaded to the PR register. The remaining 14 bits must be shifted left two bits at T_2 and input data must be loaded at T_1, to accomplish this another multiplexer (MUX2) is used. The MUX2 is again two-input one-output but it is 14 bits long. The register which is used as the PR register has load, shift-right, shift-left capability. It shifts right internally with the trailing edge of the clock pulse, shifting-left requires external connections. In order to obtain two-bit shift-left operation, the output of each flip-flop is connected to the input of the corresponding flip-flop which is two bits on the left of it. The mode control input of the flip-flops must be high for load and shift-left operation.

3.1.2.4. The PRF Adder/Subtracter

The PRF unit is constructed with three four-bit arithmetic logic units (74LS181's). This integrated circuit has the capability of performing 16 different arithmetic and logic functions. Therefore, it has four function select-lines and one mode-select line. When the mode-select input is high, it performs logic functions, if it is low it performes arithmetic functions. Since, in this work, only addition and subtraction will be done, the mode-select input is connected to ground. The arithmetic logic unit makes addition when the function-select lines are 1001, it subtracts when the function-select lines are 0110. The carry input must be one while doing subtraction. Hence, the outer two bits of the function-select lines are connected to the sign bit of the quotient digit. The inner two bits are connected to the complement of the sign bit. Also, the carry-in to the least significant bit comes from the complement of the sign bit.

3.1.2.5. The AQ Adder

The quotient digit found at each iteration is converted to two's complement representation and added to the accumulated sum in the Q register. Actually, the AQI coder converts the quotient digit into one's complement representation. If the quotient digit is represented as (s_q, t_1, t_2), it is converted to one's complement representation as follows:

$a_i = s_q$ for $i = 1$ to 10

$a_{11} = s_q \oplus t_1$

$a_{12} = s_q \oplus t_0$

(a_i is the i^{th} bit of the quotient.)

The accumulated quotient is multiplied with radix (this is done by shifting two bits to the left and putting two zeros to the least significant two bits). But instead of putting the second zero, if the sign bit of the quotient digit is sent, the quotient is obtained in two's complement form (One's complement representation + 1 = two's complement representation). Both the shifted accumulated quotient and quotient in one's complement form are supplied to the AQ adder and the next accumulated quotient is obtained at the output.

3.1.2.6. The Q Register

The output of AQ adder is placed into Q register at each iteration. This register is also 12 bit register which is constructed with two 74LS174's.

3.2. Design of QS2 Type Structure

The equipment configuration for this structure is the same as QS1-type structure except for the PREF block and the q register.

3.2.1. Control Processor of QS1 Type Structure

The flowchart of the algorithm is as shown in Figure 3.9.

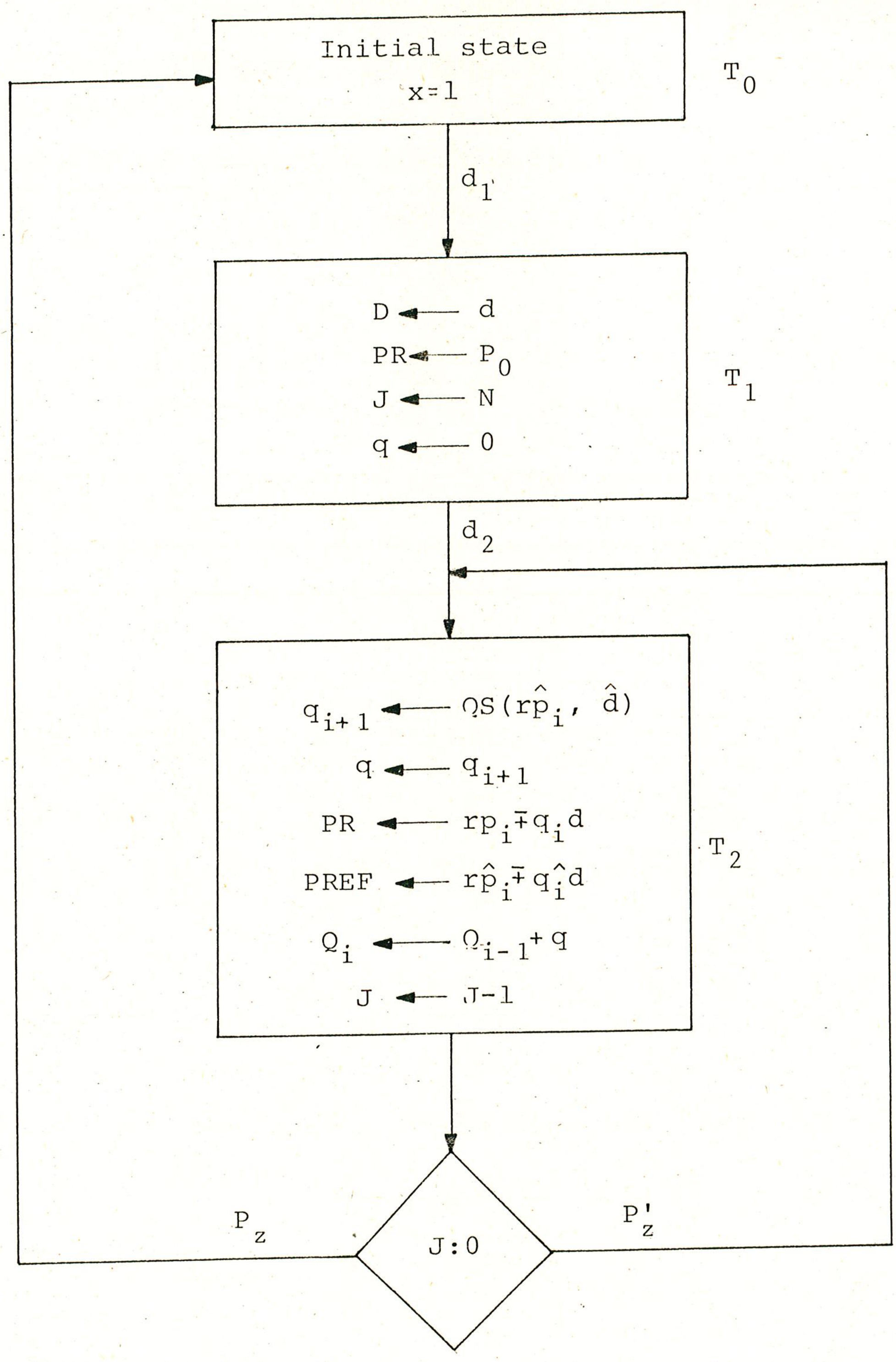


Figure 3.9. The flowchart of the algorithm for the QS2 type structure

T_0 is the initial state, with the application of the external input d_1, the control state goes to state T_1 and the initial values are placed into the registers. At state T_2, the estimate of p_{i+1} is found from the estimates of rp_i and the multiple of the divisor. The estimate of p_{i+1} is five bits long, hence the PREF block is a five bit adder/subtracter. The output of the PREF block and the third and fourth bits of the divisor are supplied to the QS and the quotient digit of the next of the next iteration is found. The quotient digit is placed into register q. While these operations are being performed, the exact result for p_{i+1} is determined at the PRF unit and placed into register PR. The counter is decremented by one and the quotient digit is added to the accumulated quotient. The timing diagram of the structure is given in Figure 3.10. The number of iterations are the same for QS1 and QS2 type structures. But, the iteration cycle time is reduced. The same control state diagram also applies for QS2. The control state diagram (Figure 3.11) and the flip-flop input functions are repeated here for convenience.

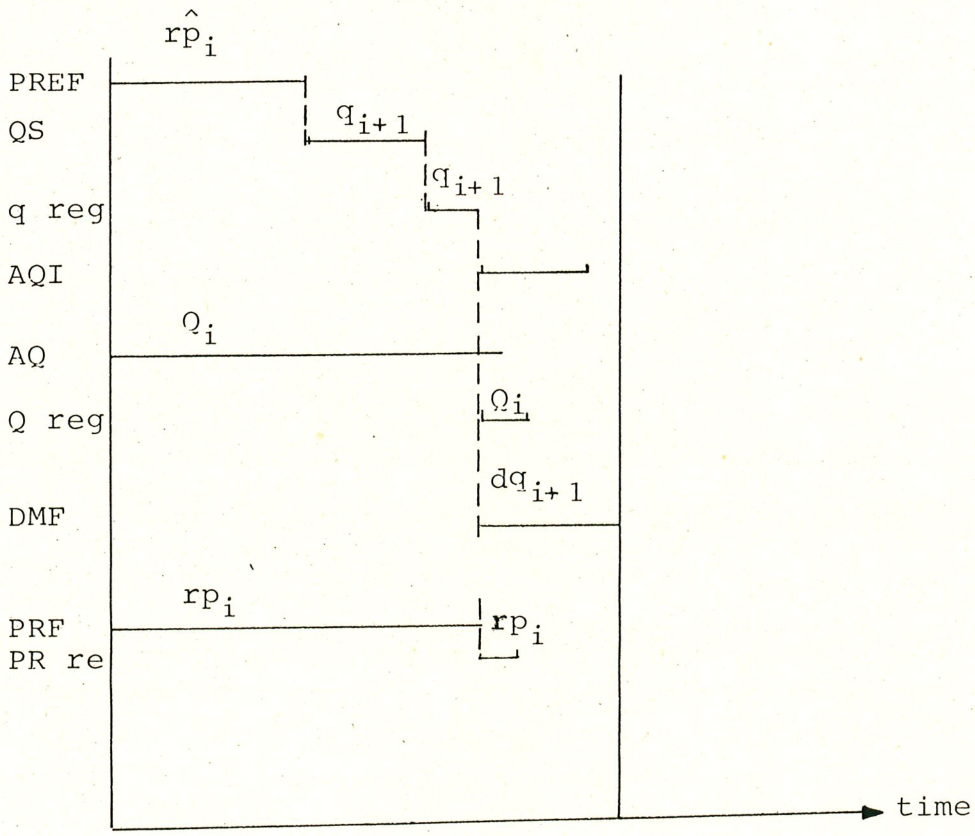


Figure 3.10. The timing diagram for the QS2 type structure

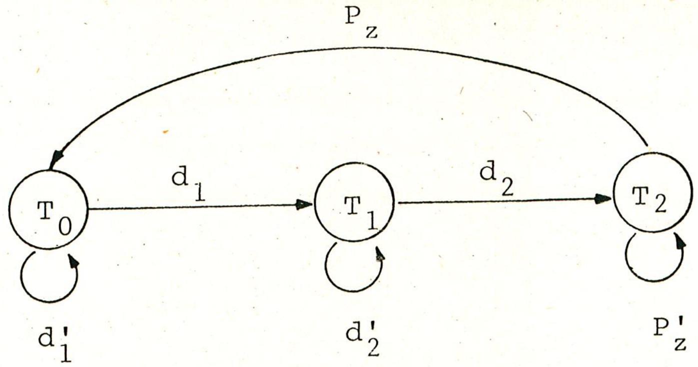


Figure 3.11. The state diagram of QS2 type structure

The flip-flop input functions are

$$DT_0 = d'_1T_0 + P'_zT_2$$

$$DT_1 = d'_2T_1 + d_1T_0$$

$$DT_2 = P'_zT_2 + d_2T_1$$

The clock inputs for the registers are determined as follows:

$T_1Cl \longrightarrow$ Clock input of D register

$(T_1+T_2)Cl \rightarrow$ Clock input of PR register

$(T_2Cl)' \longrightarrow$ Clock input of Q register

$(T_1+T_2)Cl \rightarrow$ Clock input of q register

$(T_2Cl)' \longrightarrow$ Clock input of J register

The timing diagrams are shown in Figure 3.12.

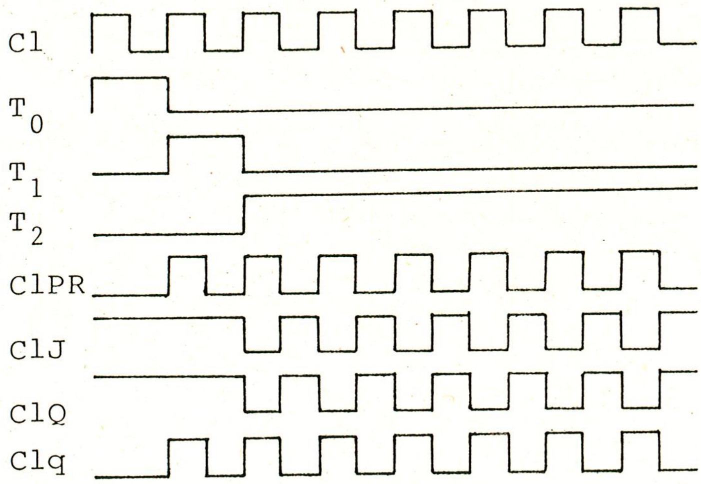


Figure 3.12. The timing diagram of the clock inputs for the QS2 type structure

3.2.2. Data Processor of QS2 Type Structure

As mentioned before, the additional hardware for QS2 type structure is the PREF unit and the q register. Since other data processor units are the same as in QS1, only PREF unit and q register will be presented here. The block diagram of the data processor of QS2 type structure is shown in Figure 3.13.

3.2.2.1. The Quotient Selector

The quotient selector is composed of a small adder (PREF), the quotient selector table and the quotient digit register. The estimate of the partial remainder is obtained from the high order bits of the output of the DMF unit and the PR register. Hence, $\lambda' = 0$, $\gamma' = 2$. By the procedure of Proposition the largest sufficient values of Δrp and Δd can be obtained as 1/8 and 1/8. Figure 3.14 shows the P-D plot and the selected boundaries of the quotient digit value regions. As it is seen in the figure, it is possible to select Δrp and Δd as 1/4 and 1/8, respectively. Hence $\hat{r}p$ has 5 bits and d has 2 bits long. The QS table of this structure is placed between the addresses 128 and 255. The address bus consists of 7 lines, 5 lines come from PREF block the other two lines come from the third and fourth bits of the divisor, respectively. The QS table is shown in Table 3.2.

3.2.2.2. The q register

This register is three bits long. A two-input three bit multiplexer provides loading zero at state T_1, and loading quotient digit at state T_2. The output of this register goes to the AQ adder, the DMF unit, the PRF unit and the PREF unit.

3.2.2.3. The PREF unit

The PREF unit is composed af a 5-bit adder/subtracter. 74LS181's are used for this block. The first five bits of the partial remainder constitute one of the inputs of the PREF unit. The other input is the first five bits of the divisor. These two inputs are either added or subtracted depending upon the sign of the quotient digit. The output of the PREF unit is connected to the QS block.

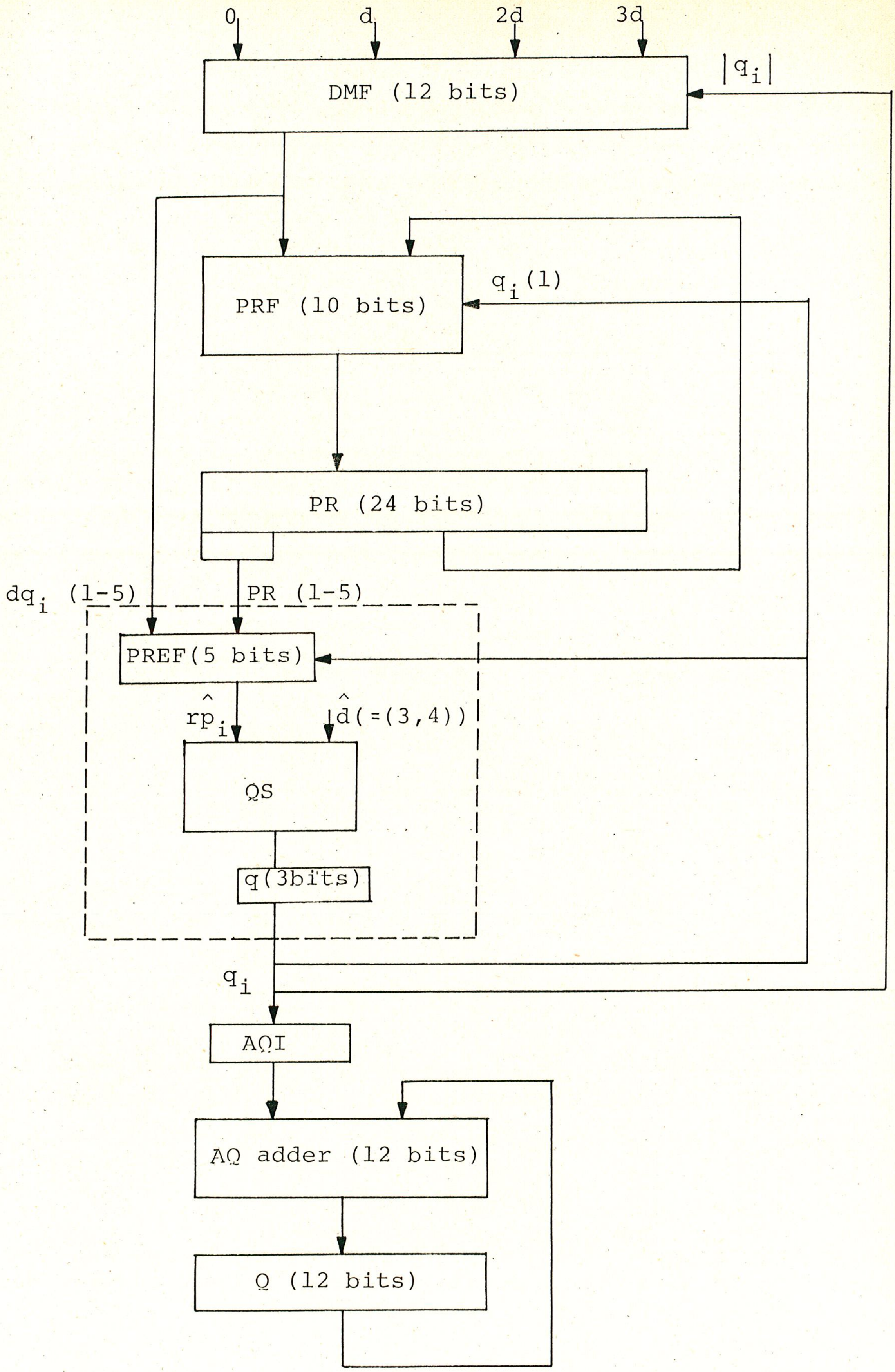


Figure 3.13. The block diagram of the data processor of the QS2 type structure

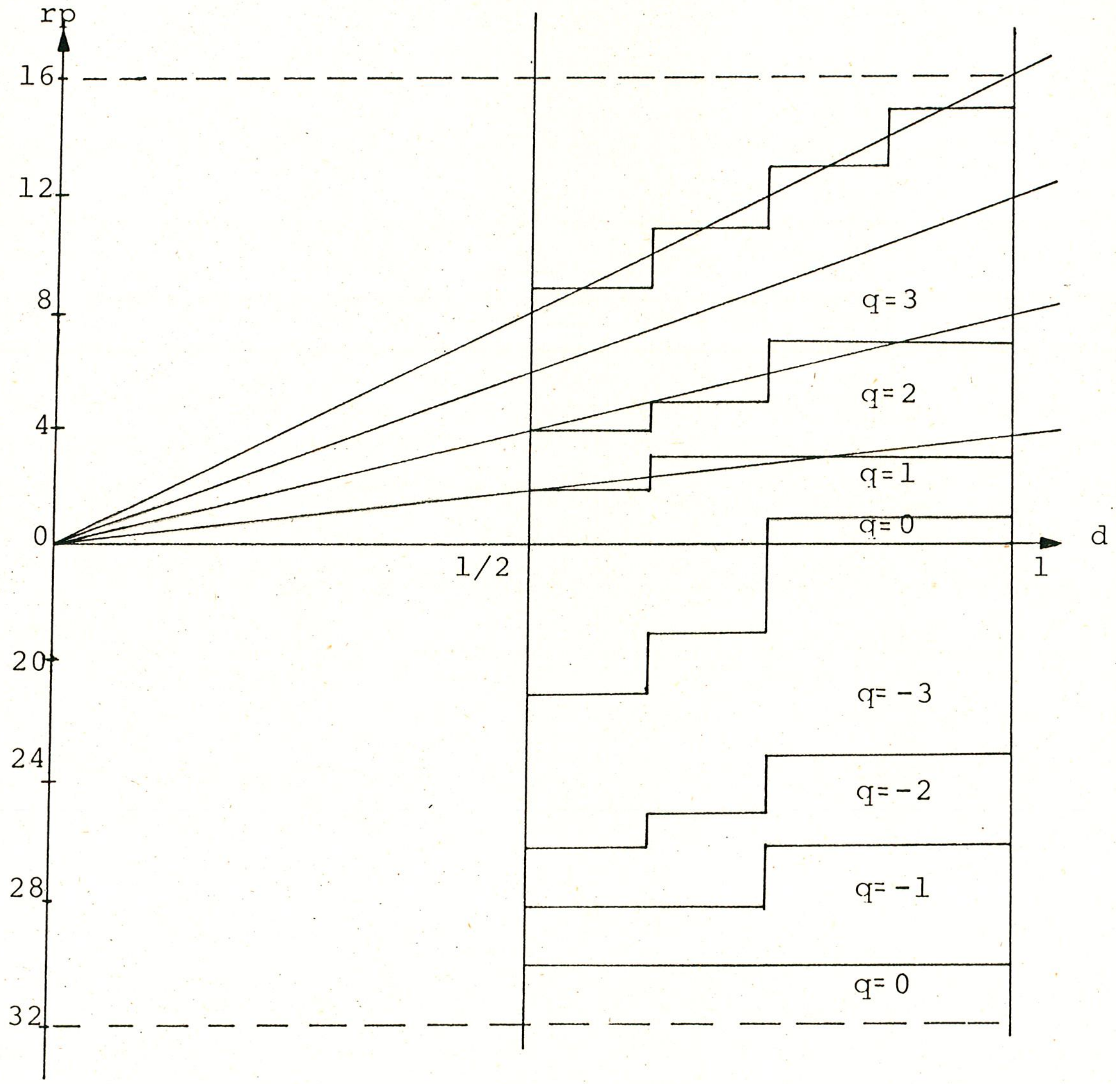


Figure 3.14. The boundaries selected in the QS table of the QS2 type structure

Table 3.2. The QS table for QS2 type structure

$$A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0 \; - \; D_7D_6D_5D_4D_3D_2D_1D_0$$

$$\underbrace{0\ \ 0\ 0\ 1\ \underbrace{X\ X\ X\ X\ X}_{\hat{r}p}\ \underbrace{X\ X}_{\hat{d}}}_{\text{address lines}} \; - \; \underbrace{1\ 1\ 1\ 1\ 1\ X\ X\ X}_{\text{data lines}}$$

$A_6A_5A_4A_3A_2A_1A_0$	$D_2D_1D_0$	$A_6A_5A_4A_3A_2A_1A_0$	$D_2D_1D_0$
0 0 0 0 0 0 0	0 0 0	0 0 1 1 1 1 1	0 1 0
0 0 0 0 0 0 1	0 0 0	0 1 0 0 0 0 0	0 1 1
0 0 0 0 0 1 0	0 0 0	0 1 0 0 0 0 1	0 1 1
0 0 0 0 0 1 1	0 0 0	0 1 0 0 0 1 0	0 1 1
0 0 0 0 1 0 0	0 0 1	0 1 0 0 0 1 1	0 1 1
0 0 0 0 1 0 1	0 0 1	0 1 0 0 1 0 0	0 1 1
0 0 0 0 1 1 0	0 0 0	0 1 0 0 1 0 1	0 1 1
0 0 0 0 1 1 1	0 0 0	0 1 0 0 1 1 0	0 1 1
0 0 0 1 0 0 0	0 0 1	0 1 0 0 1 1 1	0 1 1
0 0 0 1 0 0 1	0 0 1	0 1 0 1 0 0 0	0 1 1
0 0 0 1 0 1 0	0 0 1	0 1 0 1 0 0 1	0 1 1
0 0 0 1 0 1 1	0 0 1	0 1 0 1 0 1 0	0 1 1
0 0 0 1 1 0 0	0 1 0	0 1 0 1 0 1 1	0 1 1
0 0 0 1 1 0 1	0 0 1	0 1 0 1 1 0 0	0 1 1
0 0 0 1 1 1 0	0 0 1	0 1 0 1 1 0 1	0 1 1
0 0 0 1 1 1 1	0 0 1	0 1 0 1 1 1 0	0 1 1
0 0 1 0 0 0 0	0 1 0	0 1 0 1 1 1 1	0 1 1
0 0 1 0 0 0 1	0 1 0	0 1 1 0 0 0 0	0 1 1
0 0 1 0 0 1 0	0 1 0	0 1 1 0 0 0 1	0 1 1
0 0 1 0 0 1 1	0 1 0	0 1 1 0 0 1 0	0 1 1
0 0 1 0 1 0 0	0 1 1	0 1 1 0 0 0 0	0 1 1
0 0 1 0 1 0 1	0 1 0	0 1 1 0 0 0 1	0 1 1
0 0 1 0 1 1 0	0 1 0	0 1 1 0 0 1 0	0 1 1
0 0 1 0 1 1 1	0 1 0	0 1 1 0 0 1 1	0 1 1
0 0 1 1 0 0 0	0 1 1	0 1 1 0 1 0 0	0 1 1
0 0 1 1 0 0 1	0 1 1	0 1 1 0 1 0 1	0 1 1
0 0 1 1 0 1 0	0 1 0	0 1 1 0 1 1 0	0 1 1
0 0 1 1 0 1 1	0 1 0	0 1 1 0 1 1 1	0 1 1
0 0 1 1 1 0 0	0 1 1	0 1 1 1 0 0 0	0 1 1
0 0 1 1 1 0 1	0 1 1	0 1 1 1 0 0 1	0 1 1
0 0 1 1 1 1 0	0 1 0	0 1 1 1 0 1 0	0 1 1

Table 3.2. (cont'd)

$A_6A_5A_4A_3A_2A_1A_0$	$D_2D_1D_0$
0 1 1 1 0 1 1	0 1 1
0 1 1 1 1 0 0	0 1 1
0 1 1 1 1 0 1	0 1 1
0 1 1 1 1 1 0	0 1 1
0 1 1 1 1 1 1	0 1 1
1 0 0 0 0 0 0	1 1 1
1 0 0 0 0 0 1	1 1 1
1 0 0 0 0 1 0	1 1 1
1 0 0 0 0 1 1	1 1 1
1 0 0 0 1 0 0	1 1 1
1 0 0 0 1 0 1	1 1 1
1 0 0 0 1 1 0	1 1 1
1 0 0 0 1 1 1	1 1 1
1 0 0 1 0 0 0	1 1 1
1 0 0 1 0 0 1	1 1 1
1 0 0 1 0 1 0	1 1 1
1 0 0 1 0 1 1	1 1 1
1 0 0 1 1 0 0	1 1 1
1 0 0 1 1 0 1	1 1 1
1 0 0 1 1 1 0	1 1 1
1 0 0 1 1 1 1	1 1 1
1 0 1 0 0 0 0	1 1 1
1 0 1 0 0 0 1	1 1 1
1 0 1 0 0 1 0	1 1 1
1 0 1 0 0 1 1	1 1 1
1 0 1 0 1 0 0	1 1 1
1 0 1 0 1 0 1	1 1 1
1 0 1 0 1 1 0	1 1 1
1 0 1 0 1 1 1	1 1 1
1 0 1 1 0 0 0	1 1 1
1 0 1 1 0 0 1	1 1 1
1 0 1 1 0 1 0	1 1 1
1 0 1 1 0 1 1	1 1 1
1 0 1 1 1 0 0	1 1 1
1 0 1 1 1 0 1	1 1 1

$A_6A_5A_4A_3A_2A_1A_0$	$D_2D_1D_0$
1 0 1 1 1 1 0	1 1 0
1 0 1 1 1 1 1	1 1 0
1 1 0 0 0 0 0	1 1 1
1 1 0 0 0 0 1	1 1 1
1 1 0 0 0 1 0	1 1 0
1 1 0 0 0 1 1	1 1 0
1 1 0 0 1 0 0	1 1 1
1 1 0 0 1 0 1	1 1 0
1 1 0 0 1 1 0	1 1 0
1 1 0 0 1 1 1	1 1 0
1 1 0 1 0 0 0	1 1 0
1 1 0 1 0 0 1	1 1 0
1 1 0 1 0 1 0	1 0 1
1 1 0 1 0 1 1	1 0 1
1 1 0 1 1 0 0	1 1 0
1 1 0 1 1 0 1	1 1 0
1 1 0 1 1 1 0	1 0 1
1 1 0 1 1 1 1	1 0 1
1 1 1 0 0 0 0	1 0 1
1 1 1 0 0 0 1	1 0 1
1 1 1 0 0 1 0	1 0 1
1 1 1 0 0 1 1	1 0 1
1 1 1 0 1 0 0	1 0 1
1 1 1 0 1 0 1	1 0 1
1 1 1 0 1 1 0	1 0 1
1 1 1 0 1 1 1	1 0 1
1 1 1 1 0 0 0	0 0 0
1 1 1 1 0 0 1	0 0 0
1 1 1 1 0 1 0	0 0 0
1 1 1 1 0 1 1	0 0 0
1 1 1 1 1 0 0	0 0 0
1 1 1 1 1 0 1	0 0 0
1 1 1 1 1 1 0	0 0 0
1 1 1 1 1 1 1	0 0 0

CHAPTER 4

RESULTS AND CONCLUSIONS

The register contents for two exemplified division operations (one for QS1, one for QS2) are given below.

Division example for QS1:

Dividend in two's complement binary form (s.--..-):

1.10101100110011001100110

Divisor in two's complement binary form (s.--..-):

0.10000000000

Dividend in base 10:

-.325

Divisor in base 10:

.5

PR register	Quotient	Q register
110101100110011001100110		000000000000
110110011001100110011000	110	111111111110
111001100110011001100000	110	111111110110
110110011001100110000000	101	111111010111
111001100110011000000000	110	111101011010
110110011001100000000000	101	110101100111
111001100110000000000000	110	010110011010

Quotient in base 10:

-.65

Division example for QS2:

Dividend in two's complement binary form (s.--...-):

0.0111101111100101011101 0

Divisor in two's complement binary form (s.--...-):

0.1110010001

Dividend in base 10:

.48397

Divisor in base 10:

.946

PR register	PREF	Quotient	Q register
0011110111111001010111010	00111	000	000000000000
0001011010101011110100000	00000	010	000000000010
0101101010101111010000000	00010	000	000000001000
1000011001111010000000000	11100	001	000000100001
1111111000101000000000000	11111	101	000010000011
1111100010100000000000000	11111	000	001000001100
1110001010000000000000000		000	100000110000

Quotient in base 10:

.51159

The fundamental goal of this work is the realization and determination of speed of generalized-radix, additive normalization, non-restoring division circuits. The theory of the algorithms was investigated first. After the investigation, the remaining work involves the design of the data and control processor circuits and the experimental procedure.

The division cycle time was measured as 360ns and 340ns for QS1 and QS2 type structures, respectively. The slowness of the circuits is due to the memory access time in the QS unit. The maximum access time (address valid to output valid) of MM2716Q type EPROM is 350ns [11]. The division cycle time for QS1 type structure is the sum of delay times of QS1, DMF and PRF units. The delay times of DMF and PRF units are 14ns and 72ns, respectively. If the QS1 unit is constructed from gates, the delay time of the circuit will be 45ns (3 gate delays). Then, the division cycle time will be 131ns for QS1 type structure. But, the increase in speed means addition of 18 gates to the hardware of the circuit. In Kalaycıoğlu's thesis the division cycle times are determined theoretically as 104ns and 83ns for QS1 and QS2 type structures, respectively.

The QS1 and QS2 type division circuits were realized on a breadboard. The circuits were not realized on a printed circuit board, since it is outside the scope of this work. The stray capacitances and other parasitic elements affected the speed of the circuit. It will be possible to obtain better results if the circuits are realized on a printed circuit board.

The power consumption of the circuits is 3630mW and 3835mW, respectively. Theoretically, the power consumption must be 2948mW and 3152mW for QS1 and QS2 type circuits. The QS2 type circuit uses 45 integrated circuits. In order to use the circuit in a real computer, the circuit must be available as a single chip integrated circuit. Of course, the circuits for the normalization of the operands bust be added. Obviously, it will draw less power and it will be faster in a single chip form.

As a comparison with today's computers, the division time of a data word (8 bytes) is 8220ns (divide long instruction) for IBM 4341 Processor Model. This corresponds to a $8220/32 \approx 257$ns division cycle time, since in the designed circuits the number of division cycles for a data of the same length is 32. Obviously, the division cycle times of the designed circuits are greater than 257ns. But, as mentioned above the reason for the slowness is the used quotient selector unit.

Further research might be done in the following areas:

(i) design of QS3 and QS4 type structures,

(ii) a study of the classes of multiplicative normalization division schemes,

(iii) seek to apply the techniques of non-restoring additive normalization division schemes to other functions (logarithm, exponential, trigonometric, square-root).

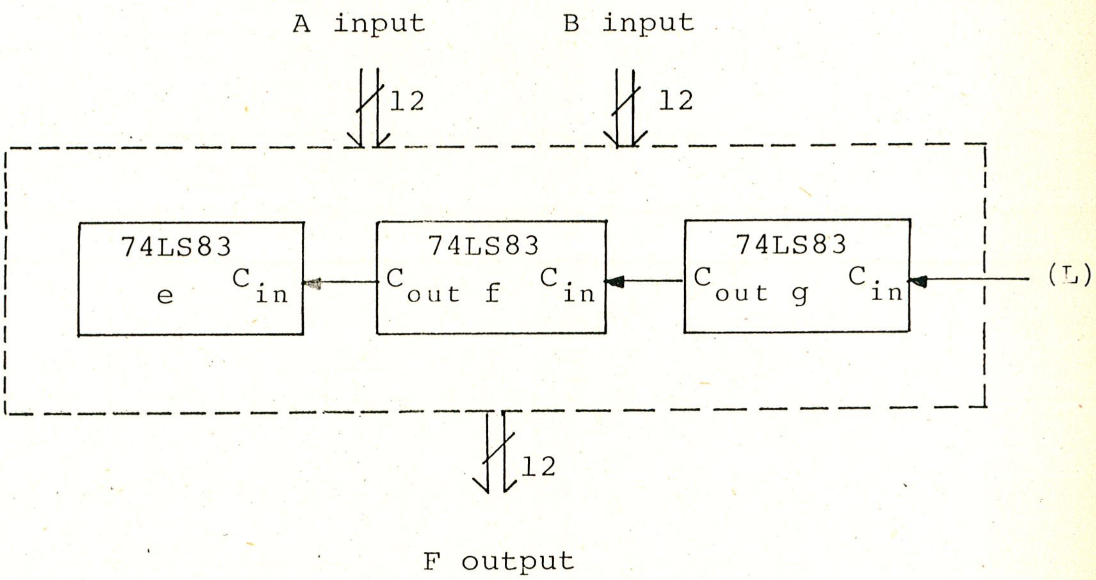
LIST OF REFERENCES

1. Robertson,J.E.,1958,"A New Class of Digital Methods", IRE Transactions on Electronic Computers,Vol.EC-7,pp.218-222.
2. Robertson,J.E.,1965,"Methods of Selection of Quotient Digits During Digital Division",File No.663,Department of Computer Science, University of Illinois.
3. Atkins, D.E., 1967,"The Theory of Implementation of SRT Division", Report No.230, Department of Computer Science, University of Illinois.
4. Atkins, D.E., 1968, "Higher Radix Division Using Estimates of the Divisor and Partial Remainder", IEEE Transactions on Computers, Vol.C-17, No.10, pp.925-934.
5. Freiman, C.V., 1961, "Statistical Analysis of Certain Binary Division Algorithms", Proceedings of IRE, Vol.49, pp.91-103.
6. Atkins, D.E., 1970, "A Study of Methods for Selection of Quotient Digits During Digital Division", Ph.D.Dissertation, Department of Computer Science, University of Illinois.
7. Kalaycıoğlu, V.Ü., 1975, "Analysis and Synthesis of Generalized-Radix Additive Normalization Division Techniques", Ph.D.Thesis, Department of Electrical and Computer Engineeering, The University of Michigan.
8. Atkins, D.E., 1972,"Redundancy in Computer Arithmetic", Proceedings of Sixth Annual IEEE Computer Society Internatial Conference,San Fransisco, pp.85-88.
9. DeLugish, B.G., 1970, "A Class of Algorithms for Automatic Evaluation of Certain Elemantary Functions in a Binary Computer", Report No.399, Department of Computer Science, University of Illinois.
10. Mano, M.M.,1979, "Digital logic and Computer Design", Prentice Hall, Inc., Englewood Cliffs, New Jersey.
11. "48-Series Microprocessors Handbook", National Semiconductor, pp.A-85, 1980

APPENDIX

The wiring diagrams of all data and control processor units are shown below.

Initialization (DMF adder)

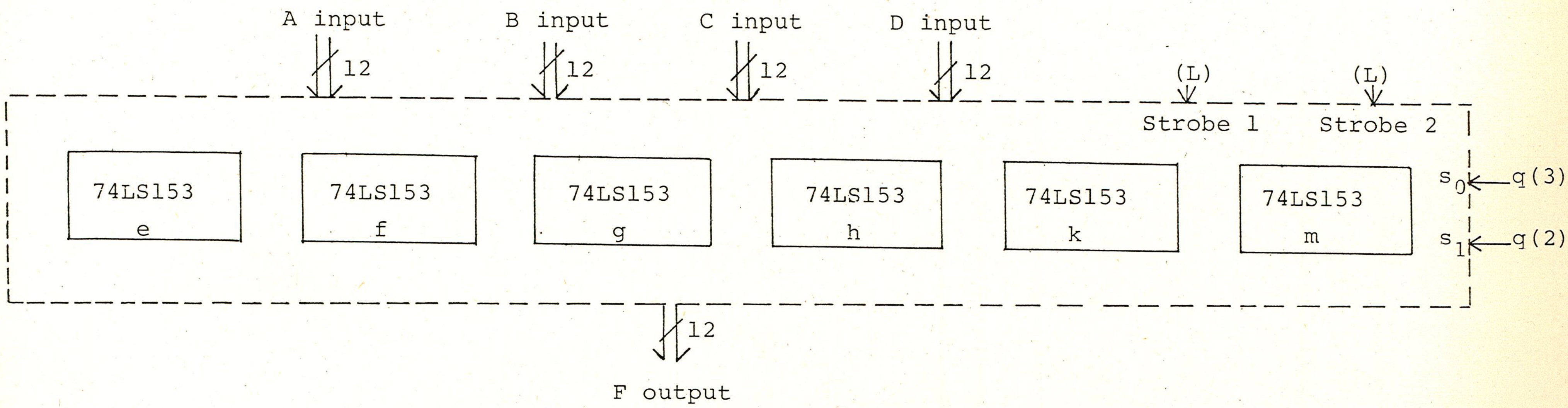


A input from D register (to the pins e1,e3,e8,e10,f1,f3, f8,f10,g1,g3,g8,g10)

B input from D register shifted left one bit (to the pins e16,e4,e7,e11,f16,f4,f7,f11,g16,g4,g7,g11)

F output to DMF unit (from pins e15,e2,e6,e9,f15,f2,f6,f9, g15,g2,g6,g9)

DMF unit



A input (all zeros, to pins e6,e10,f6,f10,g6,g10,h6,h10,k6,k10,m6,m10)

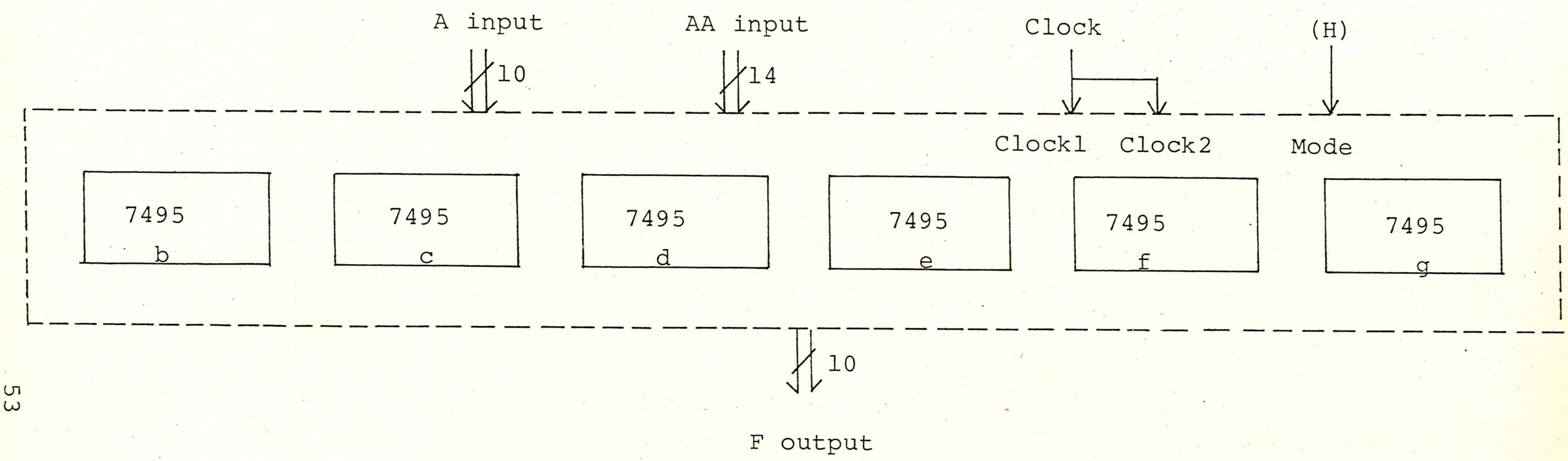
B input from D register (to the pins e5,e11,f5,f11,g5,g11,h5,h11,k5,k11,m5,m11)

C input from D register shifted left one bit (to the pins e4,e12,f4,f12,g4,g12,h4,h12, k4,k12,m4,m12)

D input from AQ adder (to the pins e3,e13,f3,f13,g3,g13,h3,h13,k3,k13,m3,m13)

F output to the PRF unit (from pins e7,e9,f7,f9,g7,g9,h7,h9,k7,k9,m7,m9)

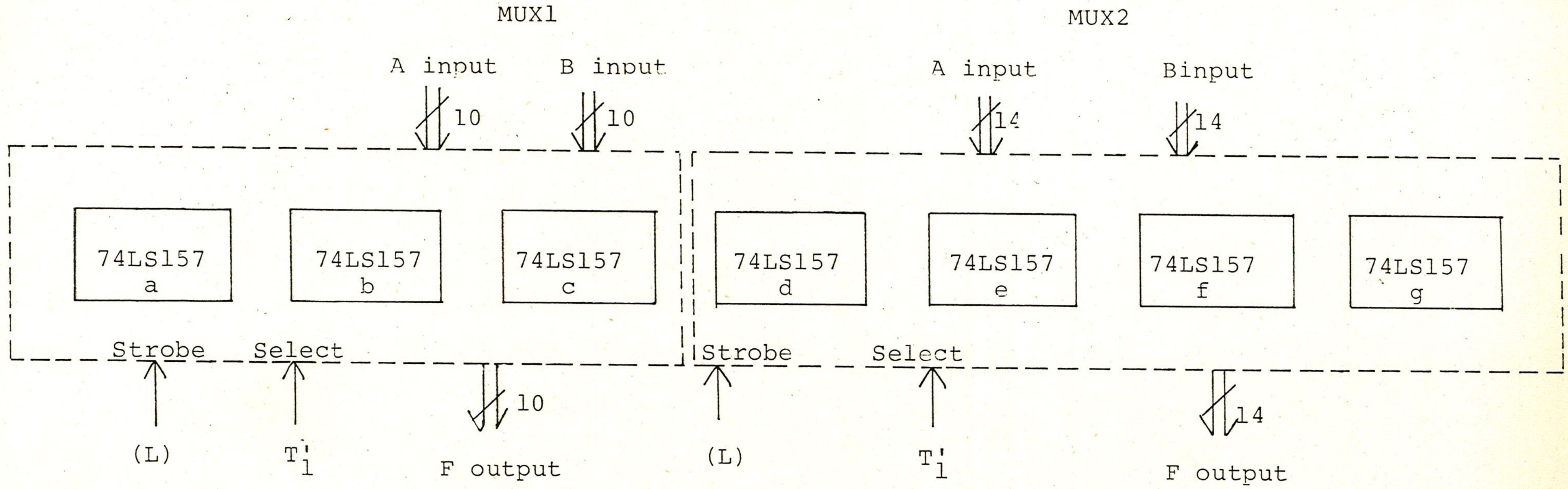
PR register



A input from MUX1 (to the pins b2,b3,b4,b5,c2,c3,c4,c5,d2,d3)

AA input from MUX2 (to the pins d4,d5,e2,e3,e4,e5,f2,f3,f4,f5,g2,g3,g4,g5)

F output to the PRF unit (from pins b11,b10,c13,c12,c11,c10,d13,d12,d11,d10)



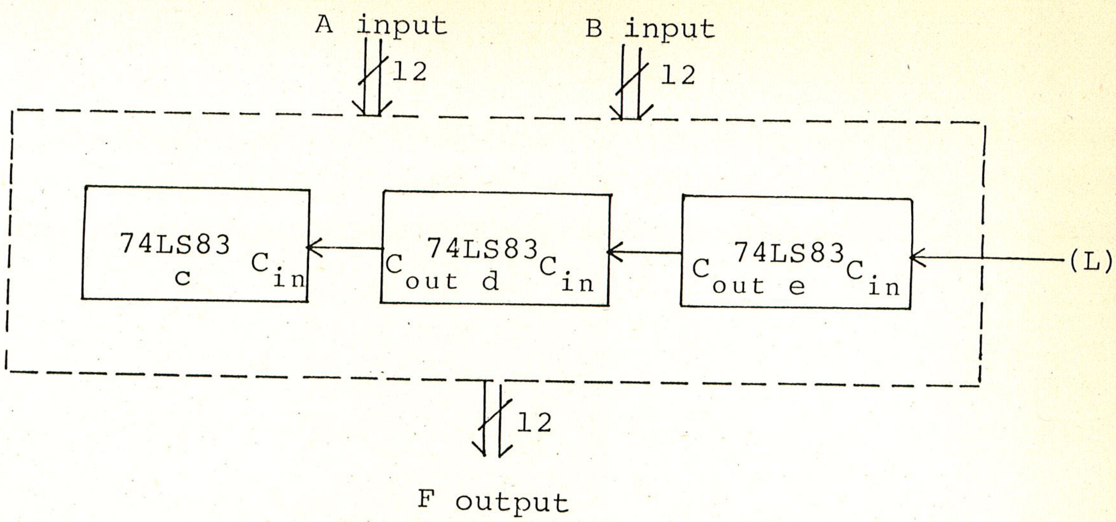
Dividend is put in as A input (pins a11,a14,b2,b5,b11,b14,c2,c5,c11,c14,d2,d5,d11,d14,e2,e5, e11,e14,f2,f5,f11,f14,g2,g5,g11,g14)

PRF output is put in as B input (pins a10,a13,b3,b6,b10,b13,c3,c6,c10,c13)

PR register output is put in as B input (pins d3,d6,d10,d13,e3,e6,e10,e13,f3,f6,f10,f13,g3,g6, g10,g13)

F output to the PR register (from pins a4,a7,a9,a12,b4,b7,b9,b12,c4,c7,c9,c12,d4,d7,d9,d12,e4,e7, e9,e12,f4,f7,f9,f12,g4,g7,g9,g12)

AQ adder

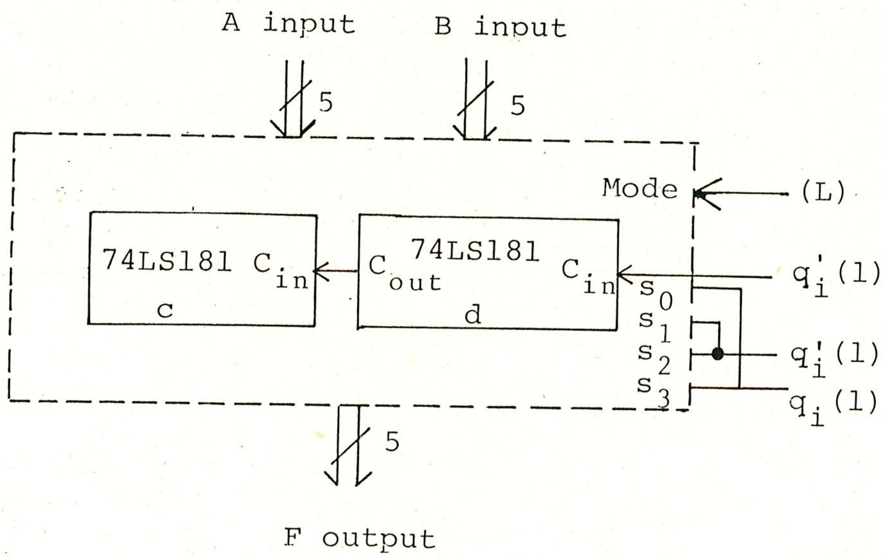


A input from AQI coder (to the pins c1,c3,c8,c10,d1,d3, d8,d10,e1,e3,e8,e10)

B input from Q register (to the pins c16,c4,c7,c11,d16,d4, d7,d11,e16,e4,e7,e11)

F output to the Q register (from pins c15,c2,c6,c9,d15,d2, d6,d9,e15,e2,e6,e9)

PREF unit

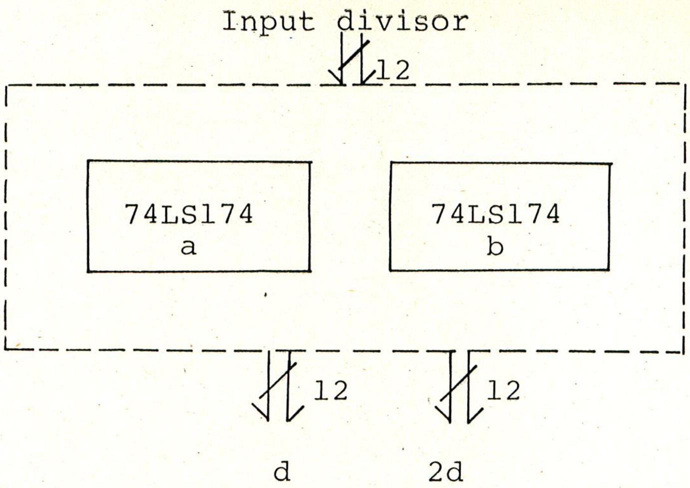


A input from PR register (to the pins c19,c21,c23,c2,d19)

B input from DMF unit (to the pins c18,c20,c22,c1,d18)

F output to the QS (from pins c13,c11,c10,c9,d13)

D register

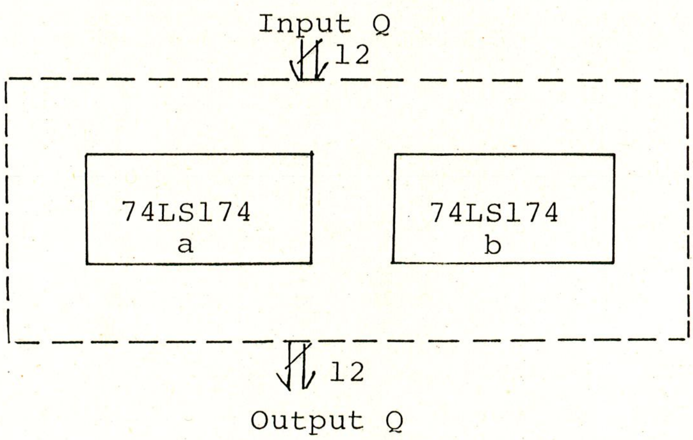


d is available at pins a15,a12,a10,a7,a5,a2,b15,b12,b10,
b7,b5,b2.

2d is available at pins a12,a10,a7,a5,a2,b15,b12,b10,b7
b5,b2,(L).

d is put in to the pins a14,a13,a11,a6,a4,a3,b14,b13,
b11,b6,b4,b3.

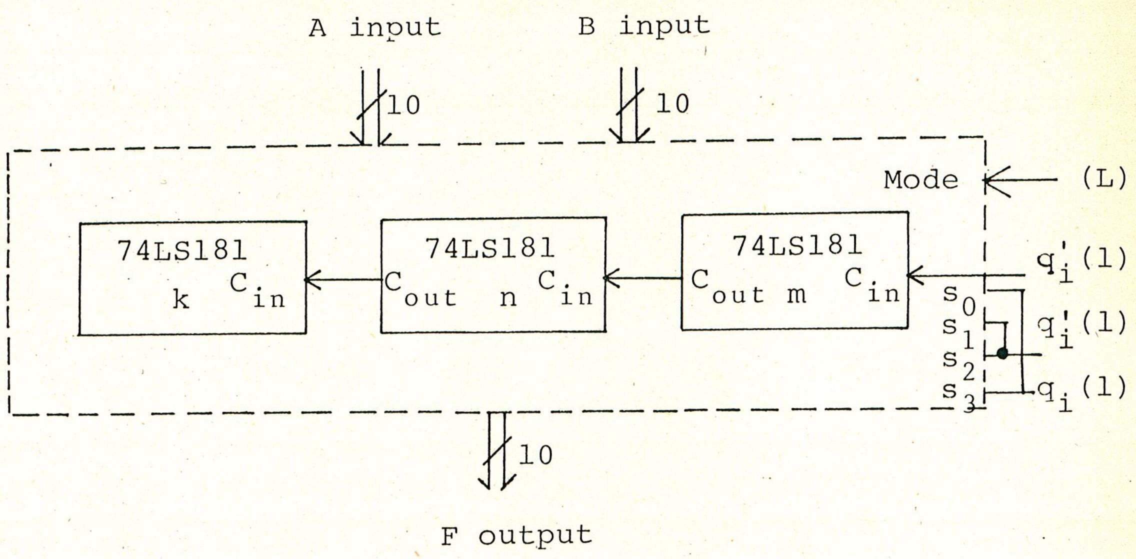
Q register



Q is put in to the pins a14,a13,a11,a6,a4,a3,b14,b13,
b11,b6,b4,b3.

Q is available at pins a15,a12,a10,a7,a5,a2,b15,b12,b10,
b7,b5,b2.

PRF Adder/Subtracter



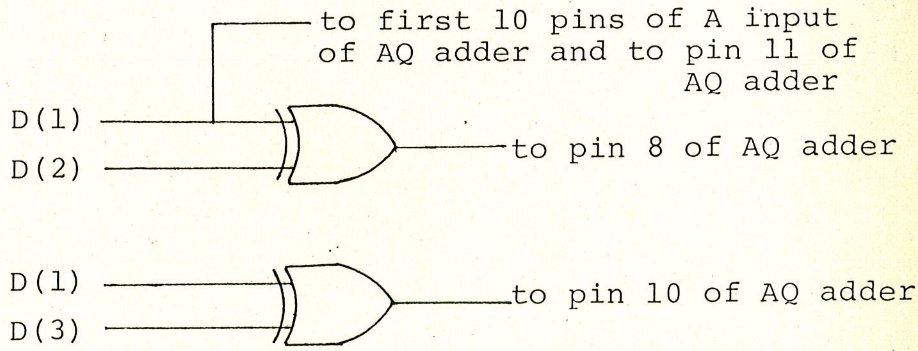
A input from PR register (to the pins k23,k2,n19,n21,n23,n2, m19,m21,m23,m2)

B input from DMF unit (to the pins k22,k1,n18,n20,n22,n1,m18, m20,m22,m1)

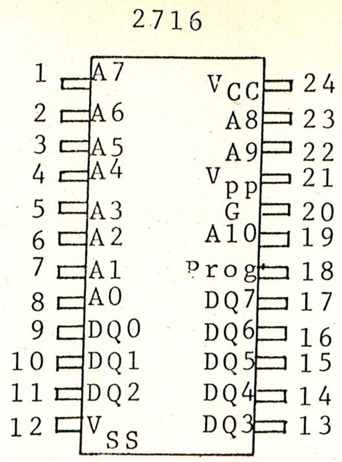
F output to the MUX1(from pins k10,k9,n13,n11,n10,n9,m13,m11, m10,m9)

F output to the PREF unit (from pins k13,k11,k10,k9,n13)

AQI Coder



QS unit



Pins 19,22,23,1,2,3 are connected to ground for QS1.

Pins 4,5,6,7 comes from pins b13,b12,b11,b10 of PR register for QS1.

Pin 8 comes from pin a10 of D register for QS1.

Pins 19,22,23 are connected to ground and pin 1 connected to 5 volts for QS2.

Pins 2,3,4,5,6 comes from PREF unit.

Pins 7,8 comes from pins a10,a7 of D register for QS2.

J register