1984

Hardware realization of one & two dimensional recursive digital filters.

Anilkumar R. Shah
University of Windsor

HARDWARE REALIZATION OF ONE & TWO DIMENSIONAL RECURSIVE DIGITAL FILTERS

by

Anilkumar R. Shah

A thesis
presented to the University of Windsor
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in
Department of Electrical Engineering

Windsor, Ontario, 1983

ABSTRACT

This thesis discusses the realization of one and two dimensional high speed recursive digital filters using the residue number system.

Initially a one dimensional recursive digital filter, using the Peled-Liu bit slicing structure was realized in both the weightage number system (fixed point arithmetic) and the residue number system (RNS). This study revealed that

i) The throughput rate of the filter obtained using the RNS is considerably higher than its counterpart the weightage number system (3 : 1 difference).

ii) If the coefficients of the digital filter transfer function are suitably scaled to integers, the resulting performance characteristics of the filters realized using the RNS resembles those obtained using the weightage number system.

On the basis of these results, a residue coded two dimensional recursive digital filter was designed and compared with the conventional realizations for such factors as spectral characteristics stability and the speed of the resulting filter.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor, Dr. M.A. Sid-Ahmed for his valuable suggestions and guidance during the course of this research. The valuable advice and comments of Dr. G.A. Jullien and Dr. M. Shridhar during the course of this work are gratefully acknowledged. In addition, the help of Dr. Hari Nagpal and many of the graduate students is sincerely appreciated.

To my brother and parents, I extend my sincerest thanks, without whom this work would not have been accomplished.

TABLE OF CONTENTS

Appendix page

LIST OF FIGURES

Figure No. | | Page No.

Figure No.		Page No.

LIST OF TABLES

LIST OF ABBREVIATIONS

$x(n)$	=	Input Sequence
$x(n-j)$	=	Previous j^{th} Input Sequence
$X(z)$	=	z-Transform of $x(n)$.
$y(n)$	=	Output Sequence
$y(n-j)$	=	Previous j^{th} Output Sequence
$Y(z)$	=	z-Transform of $y(n)$.
I/O	=	Input/Output
N	=	Order of the Filter.
B	=	No. of bits used to represent the coefficients and the input data to the filter.
$\{x(n)^j\}$	=	j^{th} bit of $x(n)$.
$\{y(n)^j\}$	=	j^{th} bit of $y(n)$.
ROM	=	Read Only Memory
ACC	=	Accumulator
ALU	=	Arithmetic And Logic Unit
SE	=	Sign Extension Flag
RNS	=	Residue Number System
m_i	=	i^{th} Moduli of the RNS.
L	=	Number of Moduli used to form the RNS.
M	=	The total number range of the RNS.
$\|a\|_{m_i}$	=	a mod m_i
$\left\|\frac{1}{a}\right\|_{m_i}$	=	Multiplicative Inverse of a mod m_i.
S	=	Scale Factor

Chapter I

INTRODUCTION

1.1 PREVIEW

In this thesis an attempt has been made to realize digital filters which have the following features.

i) The filters are constructed from a small set of relatively simple circuits, primarily read only memories (ROM) and shift registers.

ii) The configuration of the digital circuits is highly modular in form and thus well suited for VLSI construction.

Normally a ROM oriented scheme based on a table look-up method would not be feasible in a weighted number system (with a fixed radix), for any realistic dynamic range of the filter due to the number of possible combinations. Alternatively the residue number system (RNS) is a carry free, modular number system [18]. The use of the RNS allows a number to be represented with respect to the number of the selected moduli and the operations of the various moduli can be carried out independent of each other. The resulting small dynamic range makes it possible to use ROM table look-up technique and independent parallel operations capability provides the basis for a very high throughput rate of the filter.

Thus the coefficients of the digital filter to be realized were coded in residues and the filter structures were implemented using the RNS for processing one and two dimensional signals.

1.2 REALIZATION OF DIGITAL FILTERS

The hardware realization of digital filters can assume various forms, depending upon the desired degree of dedication or specialization needed for the purpose. In a broad sense, these can be divided into two classes: recursive and nonrecursive. For a recursive realization also referred to as infinite impulse response filters, the functional relationship between the input sequence of the filter {x(n)} and the resulting output sequence of the filter {y(n)} can be described as

$$y(n) = F\left[y(n-1),\ y(n-2), --\ x(n),\ x(n-1),\ x(n-2)--\right] \qquad (1.1)$$

i.e. the current output sample y(n) is a function of past outputs as well as present and past input samples.

For a nonrecursive realization also referred to as finite impulse response filters the relation between the output and the input sequences becomes

$$y(n) = F\left[x(n),\ x(n-1),\ x(n-2),\ --\right] \qquad (1.2)$$

i.e. the output sample y(n) is a function of only present and past input samples.

Consider a z-domain transfer function of the digital filter H(z), given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{j=0}^{N} a_j z^{-j}}{1 + \sum_{j=1}^{N} b_j z^{-j}} \tag{1.3}$$

Here z^{-1} is the unit delay operator, a_j's and b_j's represent the coefficients of the filter transfer function and N is the order of the filter.

A difference equation relating the output and the input can be derived by cross multiplying the terms of eqn.(1.3) and taking the inverse z-transform to give

$$y(n) = \sum_{j=o}^{N} a_j x(n-j) - \sum_{j=1}^{N} b_j y(n-j) \tag{1.4}$$

The difference equation (1.4) forms the basis of hardware realization of one dimensional recursive digital filters.

Similarly the difference equation for a two dimensional recursive digital filter can be derived to give

$$\dot{y}(k,\ell) = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{N_2} a_{n_1 n_2} \cdot x(k-n_1,\ \ell-n_2)$$

$$- \sum_{m_1=0}^{M_1} \sum_{m_2=0}^{M_2} b_{m_1 m_2} \cdot y(k-m_1,\ \ell-m_2)$$

$$(m_1,\ m_2) \neq 0 \text{ simultaneously} \qquad (1.5)$$

1.3 ELEMENTS OF DIGITAL FILTERS

The study of equation (1.4) reveals that the basic elements of digital filters needed to realize it consists primarily of

i) Delay Elements

ii) Adder (Or Subtracter)

iii) Multiplier

1.3.1 Delay Elements

These consists primarily of shift registers or First-In-First-Out registers. The size of the shift register is equal to the dynamic range of the filter, and the number of shift registers required is equal to twice the order of the filter (for recursive digital filter implementation.).

1.3.2 Adder (Or Subtracter)

Commercial adders are available in binary or binary coded decimal (BCD) form, out of which binary adders are most suited for digital filter realization [9]. If the incoming data is in two's complement form subtraction, can also be performed using the same adder.

Quite often these adders are used in conjunction with the Look-Ahead carry generators so as to limit the total add time of the larger numbers (more than 4 bits.). to that of single four bit adder [4].

1.3.3 Multipliers

The multiplier is often the most complicated element of the digital filter, and some efforts has been made to discover efficient ways of realizing this function. Conventionally multiplication of two numbers is performed using Fast Array Multipliers .

Alternatively a multiplication table can be formulated and stored using the read only memory (ROM). Here the multiplicand, and and multiplier forms the address inputs to the ROM and its output is the product of the two. The technique introduces a quantization error owing to the limited wordlength of the ROM used. However for digital filter implementation purpose the exact precision output is often not necessary; hence the technique suffices an important option in realizing the function.

Multiplication of two numbers can also be performed using the ADD/SHIFT technique. The details of this along with the special purpose hardware needed for digital filtering is provided in references [4] and [8].

1.4 FORMS OF REALIZATION

There are a multitude of equivalent digital forms in which the transfer function (1.3) can be realized, but three canonical forms or variations thereof, are most commonly used. These are

i) The Direct Form

ii) The Cascade Form

iii) The Parallel Form

1.4.1 The Direct Form

The Direct Form shown in fig. (1.1) implies direct realization of eqn. (1.4) and is usually avoided for the implementation of higher order filters because of its coefficient sensitive nature for poles close to the unit circle [3].

1.4.2 The Cascade Form

The Cascade form shown in fig. (1.2) corresponds to the factorization of the numerator and denominator polynomials of eqn. (1.3) to produce H(z) of the form

$$H(z) = \prod_{i=1}^{K} H_i(z) \qquad (1.6)$$

where H (z) is either a second-order section, i.e.,

$$H_i(z) = \frac{a_{oi} + a_{1i}z + a_{2i}z}{1 + b_{1i}z + b_{2i}z} \qquad (1.7)$$

or a first-order section, i.e.,

$$H_i(z) = \frac{a_0 + a_{1i}z}{1 + b_{1i}z} \qquad (1.8)$$

and K is the integer part of (N+1)/2. The individual first or second order sections of fig.(1.2) may be realized using the direct form. One general difficulty with the cascade structure is that, we must decide which poles to pair with which zeroes in the exact order in which it is to be realized. In the limit of infinite bit precision, for the word-length of all variables, the questions of pairing and ordering are insignificant. In a practical situation, however they are quite important. A more complete discussion of this problem is given in [5].

1.4.3 The Parallel Form

The parallel form shown in fig. (1.3) results from the partial fraction expansion of eqn. (1.3) to give

$$H(z) = C + \sum_{i=1}^{K} H_i(z) \qquad (1.9)$$

where $C = a_N / b_N$. Again first and second order filter sections in direct form are used to realize individual sections connected together in parallel to realize $H(z)$.

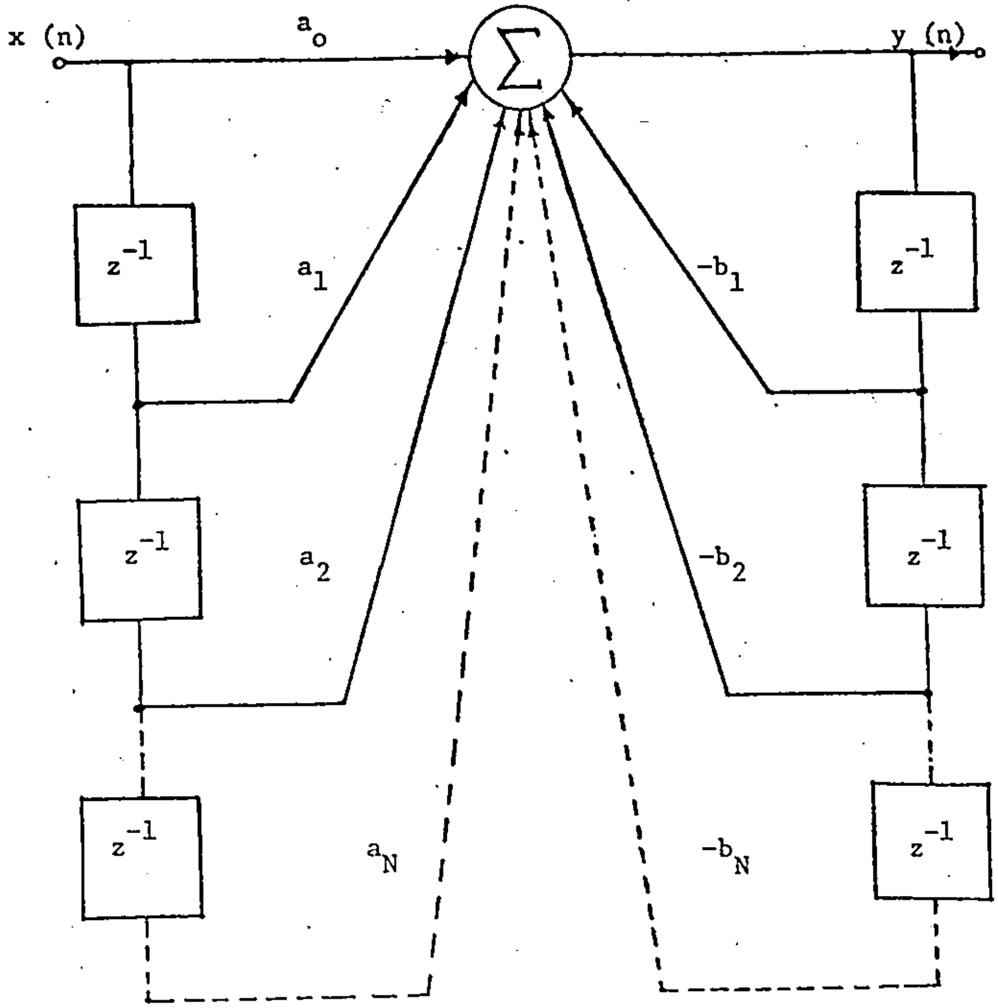


Fig. (1.1): Direct Form of Realization

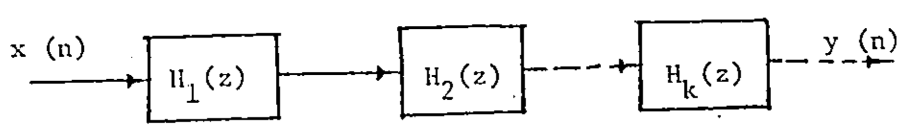


Fig. (1.2): Cascade Form of Realization

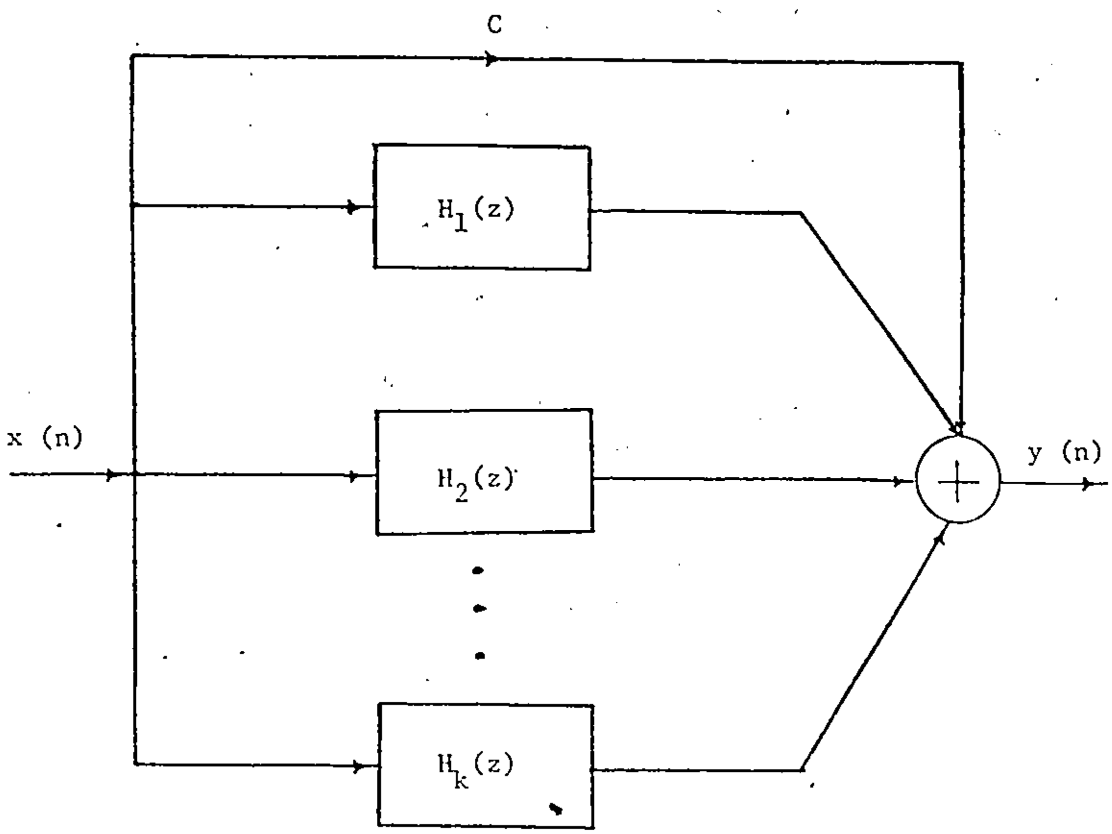


Fig. (1.3): Parallel Form of Realization

1.5 EFFECTS OF QUANTIZATION

In implementing a digital filter it is required to express the coefficients and the input data to the filter in terms of finite wordlengths. The different areas of quantization thus arising can be categorized as follows.

1.5.1 A/D Conversion Noise Or Data Quantization

With regard to data word quantization (x's and y's of eqn. (1.4)) the filter input sequence itself must be expressed by digital words of fixed length. If this sequence is obtained by encoding an analog signal, then an error noise of mean square value $\delta^2/12$ where δ is the amplitude corresponding to the least significant bit of the data word is incurred at the outset, this being reduced by 6 db with each additional bit used [9].

1.5.2 Coeffecient Quantization

Like the input data each of the coefficients of the digital filter must also be expressed in terms of a finite wordlength. As a consequence

i) The frequency response of the filter deviates from that obtained using infinite wordlength precision.

ii) A coefficient sensitive filter may become unstable depending upon the type or structure used for realization [2].

1.5.3 Round Off Error

Of the three types of quantization error, the round off error in arithmetic operations is usually the most serious. As long as no overflow occurs, addition does not lead to any inaccuracy in representing the sum; multiplication always requires quantization of the results and so this operation is the usual cause of noise in the filter structure.

The upper and lower bounds of error due to rounding and truncation after each arithmetic operation (or small groups of arithmetic operations) in fixed and floating point arithmetic is given in fig. (1.4), as a probability density function of the error [1].

1.5.4 Limit Cycles [1]

In the case of recursive digital filters there is an added problem of limit cycles mainly occuring due to the overflow of the dynamic range of the filter.

This effect can be eliminated by one of three ways.

i) Properly scaling the filter so that overflow is absolutely impossible. This technique has the disadvantages of decreasing the signal to noise ratio of the filter realized.

ii) Modifying the adders so that they saturate rather than 'wrap around' as in a time two's complement unit.

iii) Changing the structure of the filter section.

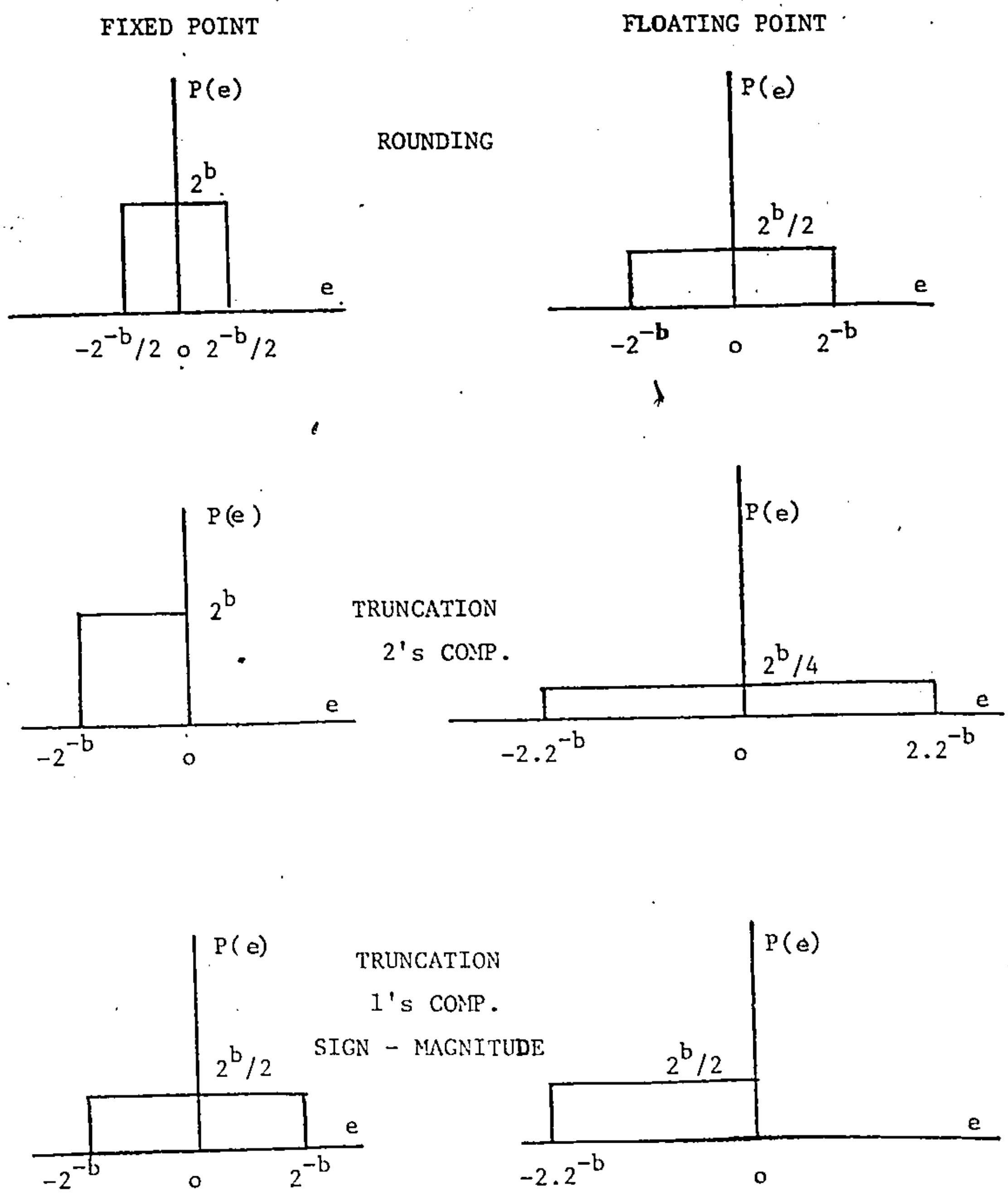


Fig. (1.4) Summary of Quantization Noise Expressed As Probability Density Fn. of Error

1.6 PREVIOUS WORK

1.6.1 For Realizing One Dimensional Digital Filters

In 1968 Jackson [7] suggested an approach to the implementation of digital filters. The paper outlines the various forms of realization along with the configuration of digital circuits that is highly modular and thus well suited for LSI construction. It also discusses the applicability of two's complement arithmetic in the implementation of digital filter.

In 1974 Peled and Liu [10] presented a novel idea for the implementation of digital filters. The realization calls for the storing of finite number of possible outcomes of an intermediate arithmetic operation, and using them to obtain the next output sample, through repeated addition and shifting operations. Thus in effect multiplication has been replaced by ADD/SHIFT operation, and bit slicing technique was used to obtain the next output sample.

The serial nature of the proposed structure, however accounts for the decrease in the operating speed of the filter with an increase in its dynamic range. This limits its usability to a class of general filters whose coefficients are not very sensitive to quantization.

Later in 1975 Agarwal and Burrus [11] proposed structure particularly suited for coefficient sensitive recursive digital filters. The method is based on transforming the coefficients of the z- plane to $\hat{z}$- plane whose origin is at z=1.

In 1978 Ahmad Shenoi and Allen [12] presented a filter structure based on the digital increment computers also referred to as the digital differential analyzers. The filter structure is shown to be suitable for realizing transfer functions having their poles near z = 1

Until recently, all filters were implemented in a fixed radix system. Despite the number of advantages of the weighted number system one of its major drawback is that, the arithmetic operations (addition subtraction and multiplication) of higher order bits of the same number cannot proceed, until carry from the lower order bits are generated. The problem can be solved by using the Look-Ahead carry generators or alternatively by using the number system which is free from any base viz. the RNS.

Lately Jullien and Jenkins [19] studied the applications of residue number system in the realization of digital filters. The carry free property of the residue number system was utilized and filter structures were proposed faclitating digital filtering in the range of 4-6 MHz. Also scaling algorithm was presented for the use of recursive digital filters.

1.6.2 For Realizing Two Dimensional Digital Filters

Very little work has been done in the hardware realization of Two dimensional recursive digital filters, primarily due to the difficulty of factorizing the transfer function into sections of 2x2 or 3x3 filters.

in 1976 Mitra et al [13] presented a paper on the hardware implementation of two dimensional recursive digital filters. The work was an extension of the Peled Liu bit slicing approach to suit two dimensional filtering

In 1978 Mitra and Chakrabarti [14] presented a general scheme for realizing two dimensional digital transfer functions. The method is based on partitioning two dimensional transfer function into that consisting of only X delays, only Y delays and a linking transfer function of both X and Y delays.

1.7 OBJECTIVES OF THE RESEARCH

Hardware realization of digital filters have ranged from generalized processors to dedicated large scale integrated circuits. One of the key issue in the implementation of digital filters is to achieve a high throughput rate, for any realistic dynamic range of the filter. This problem can be solved effectively by the use of residue number system in the implementation of digital filters.

Recently Jenkins [20] proposed a residue coded recursive digital filter using the general scaling technique to obtain the output. This thesis studies the feasibility and the problems associated with such a realization, mainly due to the

i) Limited number range of the filter.

ii) Quantization of the coefficients when scaled to integers.

These problems have been solved by selecting a higher moduli. This increases the total number range of the system and allows the use of a larger scale-factor to scale the coefficients without overflowing the number range of the filter. Also the technique of scaling through metric vector estimates proposed by Jullien [21] was used in lieu of the general scaling technique to decrease the quantization error in the output processed signal.

On a similar basis a filter structure has been proposed and simulated on the computer to process two dimensional signals recursively.

1.8 THESIS ORGANIZATION.

Chapter 2 presents the details of hardware realization of fixed point recursive digital filter using the Peled-Liu bit slicing technique.

In chapter 3 the mathematical concepts of the RNS is introduced through a discussion of its basic properties. It is shown that the RNS is much more suitable for realizing highly parallel digital filters than the conventional fixed radix number system (such as the binary number system), because arithmetic operations in RNS are fully independent digits.

Further the coefficients of the low-pass recursive digital filter used in chapter 2 were coded in RNS and the residue coded recursive digital filter structure proposed by Jenkins was simulated on the computer to study the feasibility of the structure for practical implementation.

In chapter 4 a hardware structure based on RCM implementation is proposed and tested to process two dimensional signals recursively using the RNS.

Finally chapter 5 presents the conclusions that can be obtained from the research work presented in this thesis.

Chapter II

REALIZATION OF 1-D RECURSIVE DIGITAL FILTER

The difference equation (1.4) can be realized in a variety of ways using the fixed point arithmetic. The Peled-Liu bit slicing technique[1] [10] was chosen for realization at the outset mainly due to the ease with which multiplication can be performed without the use of array multipliers.

2.1 THE PELED-LIU APPROACH

Consider the implementation of a second order section of eqn. (1.4) specified by the input/output relationship

$$y(n) = a_0 \cdot x(n) + a_1 \cdot x(n-1) + a_2 \cdot x(n-2) - b_1 \cdot y(n-1) - b_2 \cdot y(n-2) \qquad (2.1)$$

The input/output signals of the filter are bounded by ± 1 volt and B binary bits including the sign bit in 2's complementary code are used to represent the signal that is

---

[1] This technique was suggested earlier by Crosier [6].

$$S_i(n) = - S_i(n)^0 + \sum_{j=1}^{B-1} S_i(n)^j \cdot 2^{-j} \qquad (2.2)$$

where $S_i(n)^j$[2] is either x(n) or y(n). Eqn. (2.1) can now be rewritten as

$$\begin{aligned} y(n) = \; & a_o \cdot \{ \sum_{j=1}^{B-1} x(n)^j \cdot 2^{-j} - x(n)^0 \} \\ & + a_1 \cdot \{ \sum_{j=1}^{B-1} x(n)^j \cdot 2^{-j} - x(n-1)^0 \} \\ & + a_2 \cdot \{ \sum_{j=1}^{B-1} x(n-2)^j \cdot 2^{-j} - x(n-2)^0 \} \\ & - b_1 \cdot \{ \sum_{j=1}^{B-1} y(n-1)^j \cdot 2^{-j} - y(n-1)^0 \} \\ & - b_2 \cdot \{ \sum_{j=1}^{B-1} y(n-2)^j \cdot 2^{-j} - y(n-2)^0 \} \qquad (2.3) \end{aligned}$$

Define a function (·) with five arguments as follows:-

$$\begin{aligned} F(s^1\ s^2\ s^3\ s^4\ s^5) = \; & a_0 \cdot s^1 + a_1 \cdot s^2 + a_2 \cdot s^3 \\ & - b_1 \cdot s^4 - b_2 \cdot s^5 \qquad (2.4) \end{aligned}$$

---

[2] $S_i(n)^j$ represents the j^{th} bit of $S_i(n)$.

Rearranging the terms of eqn. (2.3), and by using eqn. (2.4) we get

$$y(n) = \sum_{j=1}^{B-1} 2^{-j} \cdot F\,[x(n)^j,\ x(n-1)^j,\ x(n-2)^j,\ y(n-1)^j,\ y(n-2)^j]$$

$$- F\,[x(n)^0,\ x(n-1)^0,\ x(n-2)^0,\ y(n-1)^0,\ y(n-2)^0] \quad (2.5)$$

Eqn. (2.5) forms the basis of Peled-Liu approach for the realization of the difference equation (2.1). Inspection of eqn. (2.5) reveals the following properties.

i) There are five arguments of the function F(.) forming $2^5 = 32$ possible combinations of the function F(.). The value of the function F(.) depends upon the binary vector (0 or 1) that forms its argument. Thus for a given difference equation these values can be precomputed and stored in ROM forming a Look Up table as illustrated in table 2.1 .

In practice the contents of ROM are scaled by a factor S and rounded to B bits before storing them in ROM. The selection of the scale factor S is on a minimax basis [16] .

ii) The arguments of function F(.) serves as address inputs to the ROM.

iii) Each output sample y(n) can be computed by resorting to data additions with Two's Complementary Right Shift and a single subtraction. The details of Two's Complement Multiplication performed using ADD/SHIFT technique are given in section 2.2.

TABLE 2.1

Contents Of ROM

LOC.	MEM. ADD.	ROM CONTENTS	LOC.	MEM. ADD.	ROM CONTENTS
1	00000	0	17	10000	a_0
2	00001	$-b_2$	18	10001	$(a_0 - b_2)$
3	00010	$-b_1$	19	10010	$(a_0 - b_1)$
4	00011	$-(b_1 + b_2)$	20	10011	$(a_0 - b_1 - b_2)$
5	00100	a_2	21	10100	$(a_0 + a_2)$
6	00101	$(a_2 - b_2)$	22	10101	$(a_0 + a_2 - b_2)$
7	00110	$(a_2 - b_1)$	23	10110	$(a_0 + a_2 - b_1)$
8	00111	$(a_2 - b_1 - b_2)$	24	10111	$(a_0 + a_2 - b_1 - b_2)$
9	01000	a_1	25	11000	$(a_0 + a_1)$
10	01001	$(a_1 - b_2)$	26	11001	$(a_0 + a_1 - b_2)$
11	01010	$(a_1 - b_1)$	27	11010	$(a_0 + a_1 - b_1)$
12	01011	$(a_1 - b_1 - b_2)$	28	11011	$(a_0 + a_1 - b_1 - b_2)$
13	01100	$(a_1 + a_2)$	29	11100	$(a_0 + a_1 + a_2)$
14	01101	$(a_1 + a_2 - b_2)$	30	11101	$(a_0 + a_1 + a_2 - b_2)$
15	01110	$(a_1 + a_2 - b_1)$	31	11110	$(a_0 + a_1 + a_2 - b_1)$
16	01111	$(a_1 + a_2 - b_1 - b_2)$	32	11111	$(a_0 + a_1 + a_2 - b_1 - b_2)$

2.2 TWO'S COMPLEMENT MULTIPLICATION

Consider the multiplication of two numbers n_1 and n_2 as shown below

$$y = n_1 \cdot n_2$$

$$= (-0.390625) \cdot (0.8125)$$

$$= -0.3173828125 \qquad (2.6)$$

Here n_1 forms the multiplicand (MD) n_2 forms the multiplier (MR) and Y is the product of two numbers.

Two's Complement Of	-0.390625	=	1.100111	(MD)
Binary Represantation of	0.8125	=	0.1101	(MR)

The '1' amd '0' on the left of the binary point of multiplicand and multiplier indicates its respective sign (refer table 2.2). To start with the accumulator is cleared. The accumulator is of 12 bits in order to get the exact output.

Since the least significant bit (LSB) of the MR is 1 the MD 1100111 is added to the accumulator, and a 2's complementary right shift is performed to ensure the sign of the shifted partial product.

The operation is continued till the most significant bit (MSB) of the MR is reached. Now since the 2's complement of the number is represented as

$$x = -x_k^0 + \sum_{j=1}^{B-1} x_k^j \cdot 2^{-j} \qquad (2.7)$$

the Md is multiplied with the ~~MSB~~ of the MR, 2's complement of it is taken and added to the partial product of the accumulator to give the final output. The details of the operation are shown in table 2.2 .

TABLE 2.2

The Details Of Two's Complement Multiplication

$$y = (-0.390625) \quad (0.8125)$$
$$= -0.3173828125$$

Two's Complement Of $-0.390625 = \overbrace{1.100111}^{\hat{m}}$ (MD)

x_0 x_4

Binary Representation of $0.8125 = 0.1101$ (MR)

OPERATION	ACCUMULATOR
CLEAR	0 0 0 0 0 0 0 0 0 0 0 0
ADD $x_4\hat{m}$ = 1 1 0 0 1 1 1	1 1 0 0 1 1 1 0 0 0 0 0
2's comp. Right Shift	1 1 1 0 0 1 1 1 0 0 0 0
ADD $x_3\hat{m}$ = 0 0 0 0 0 0 0	1 1 1 0 0 1 1 1 0 0 0 0
2's comp. Right Shift	1 1 1 1 0 0 1 1 1 0 0 0
ADD $x_2\hat{m}$ = 1 1 0 0 1 1 1	1 1 0 0 0 0 0 1 1 0 0 0
2's comp. Right Shift	1 1 1 0 0 0 0 0 1 1 0 0
ADD $x_1\hat{m}$ = 1 1 0 0 1 1 1	1 0 1 0 1 1 1 0 1 1 0 0
2's comp. Right Shift	1 1 0 1 0 1 1 1 0 1 1 0
SUB. $x_0\hat{m}$ = 0 0 0 0 0 0 0	1 1 0 1 0 1 1 1 0 1 1 0

1 1 0 1 0 1 1 1 0 1 1 0 = -0.3173828125

2.3 HARDWARE IMPLEMENTATION

The Peled-Liu approach discussed above was implemented in hardware. The hardware architecture may be divided into three sections.

i) The analog interfacing hardware consisting of analog to digital converter (ADC), digital to analog converter (DAC) and the code conversion circuitry.

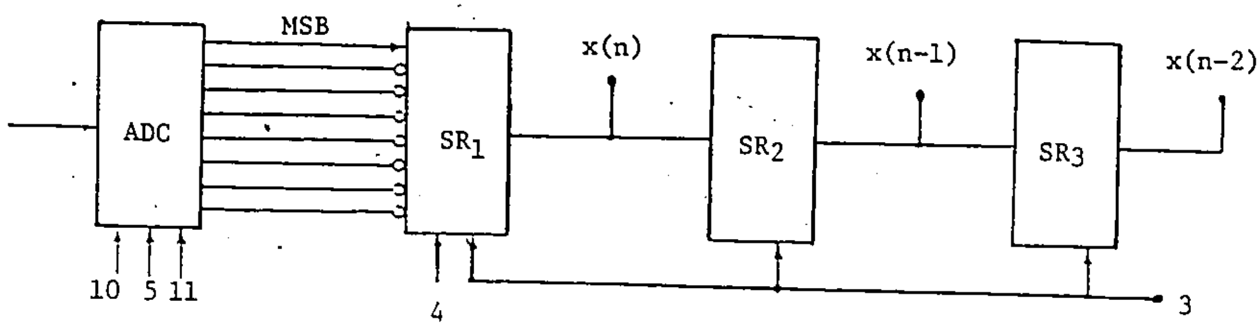
ii) The digital filter consisting of shift registers, arithmetic logic units (ALU), accumulator (ACC) and the ROM for storing the partial fractions of a given transfer function.

iii) The control circuit consisting of the circuitry to synchronize and provide timing pulses for the filter and the interfacing hardware.

Fig.(2.1) depicts the circuit diagram of the second order recursive digital filter with a dynamic range of 8 bits. Fig.(2.2) represents the corresponding timing diagram of the filter controls.

Analog input of +5 volts range enters the filter through the ADC. The ADC is level triggered by a Start Of Conversion pulse (SC) and the End Of Conversion line (EOC) will remain low until the conversion is completed. The details of the interfacing hardware is given in [17]

At the beginning of every sample cycle, the input data x(n) is synchronously parallel loaded into the shift



3	CKSR	10	CKADC
4	S/$\overline{LSR}$, Round	11	EOC
5	SC		

Fig. 2.1(a) Interface Section

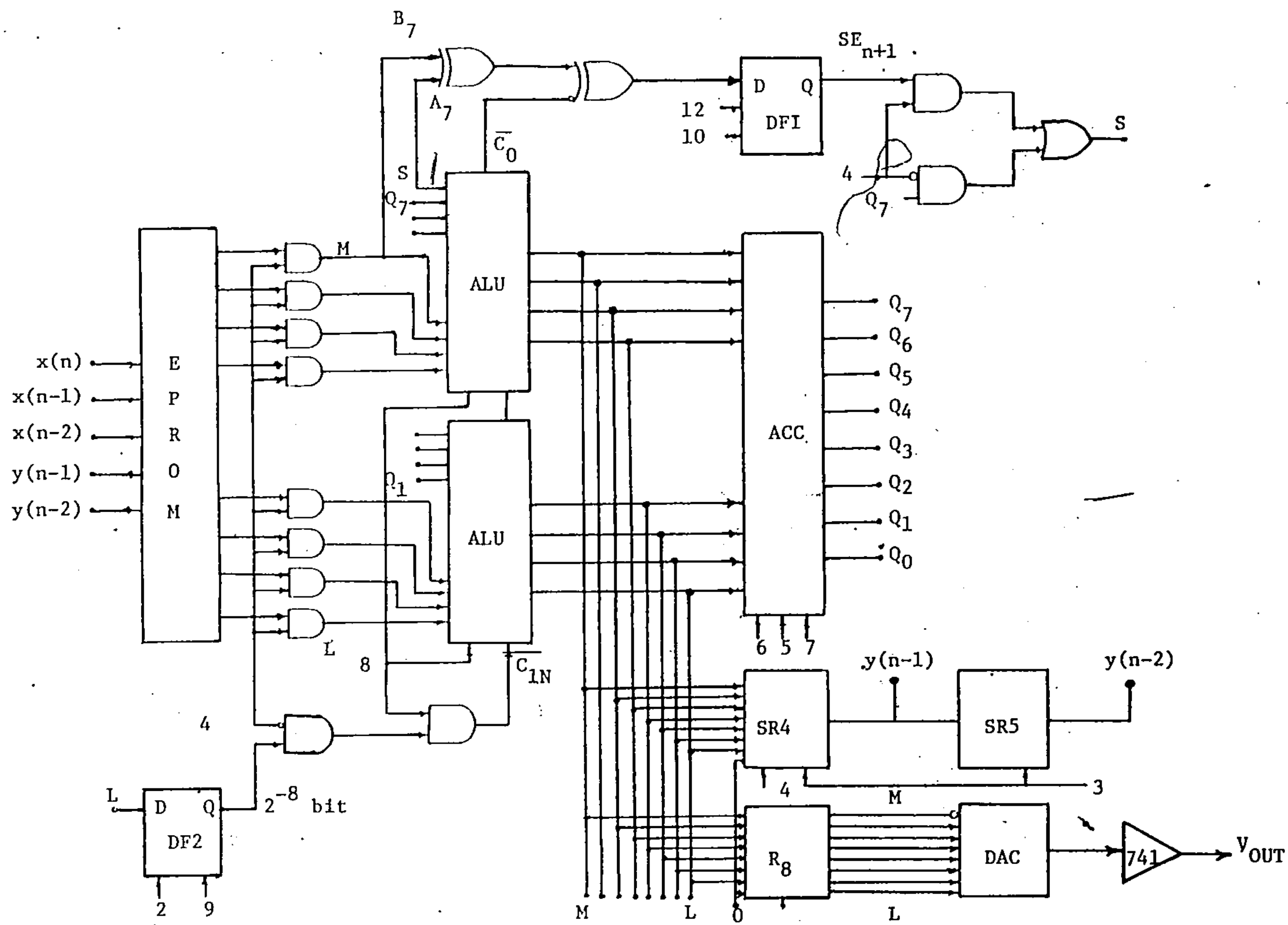
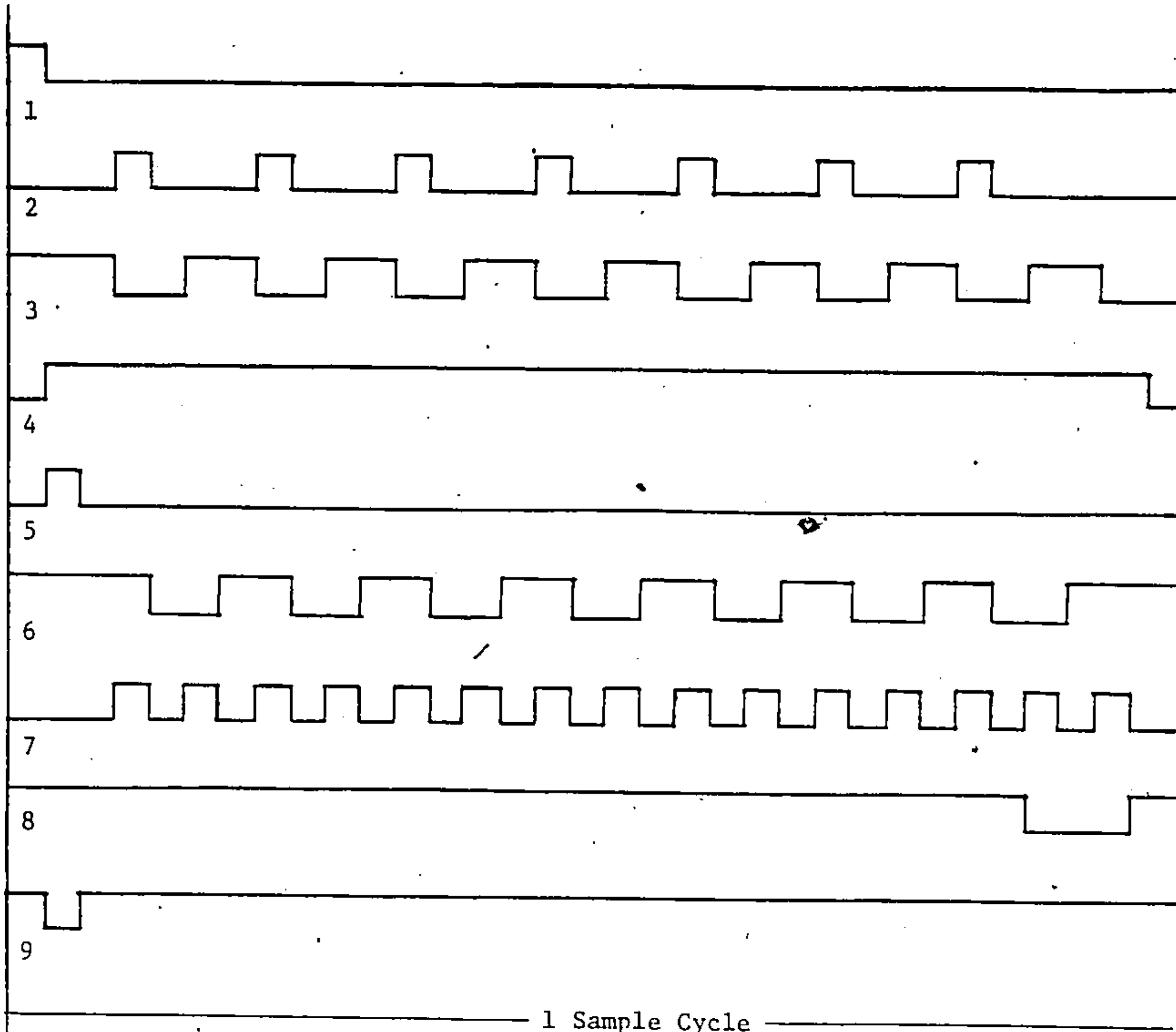


Fig. (2.1) Cct. Diagram of a Second Order Recursive Digital Filter Realized Using Peled-Liu Approach.



1 CKR8
2 CKFF
3 CKSR
4 S/$\overline{\text{LSR}}$, ROUND
5 SC,CLACC

6 L/$\overline{\text{SACC}}$
7 CKACC
8 ADD/$\overline{\text{SUB}}$
9 $\overline{\text{CLACC}}$

Fig. (2.2) Timing Diagram of the Filter Control Signals.

register SR1. This occurs at the negative edge of the clock CKSR (clock shift register) ; the line $S/\overline{LSR}$ being held low for the purpose. Similarly the previous output sample y(n-1) is also loaded broadside into SR4. However since the coefficient data in the RCM has been scaled down by a factor of S (S = 2 in the filter implemented), the output data y(n-1) must be scaled up by the same factor before being loaded into SR4. For S = 2 this scaling is accomplished by a hard-wired 1 bit left shift (skewed parallel connection). A '0' is wired to the LSB input of SR4.

Seven more successive clock edges of the clock CKSR are needed to shift all 8 bits through the registers. The output of the shift registers are tapped to the address pins of the EPROM (MC2708) which contains the 32 values of the partial fraction F(.). The value of F(.) selected are AND gated with the rounding control line 'ROUND' (which is active low during rounding cycle.) and becomes the B inputs of the ADDER/SUBTRACTOR.

The other input A of the ADDER/SUBTRACTOR has its seven inputs coming from the accumulator (74198). The MSB $A_{7\,n+1}$ of the input A to the adder is derived from the sign extension logic (SE) which performs the operation

$$SE_{n+1} = D\ (A_{7n} + B_{7n} + C_{0n}) \tag{2.8}$$

Here D(.) is the unit delay operator.

A_{7n} is the MSB of the input A to the ADD/SUB unit corresponding to the n-th interval of time.

B_{7n} is the MSB of the input B to the ADD/SUB unit corresponding to the n-th interval of time.

and C_{0n} is the final carry output of the ADD/SUB unit corresponding to the n-th interval of time.

Additional circuitry is required to accomodate the rounding cycle and hence

$$A_{7n+1} = SE_{n+1} \cdot ROUND + \overline{ROUND} \cdot Q_{7n} \quad (2.9)$$

where Q_{7n} is the MSB output of the accumulator [16].

At the beginning of every sample cycle the accumulator and the delay flipflops DF1 and DF2 are cleared by holding the line Clear Accumulator (CLACC) low. The output of the adder is then loaded into the accumulator and shifted one bit right on two sucessive positive edges of the clock CKACC. The sign extension flag is loaded on the positive edge of the clock CKFF.

The LSB output of the ADDER/SUBTRACTOR is also tied to the input of the delay flipflop DF2. This effectively holds the 2^{-8} bit of the result which is used during the rounding cycle.

After 7 partial sums has been computed, the ADD/SUB (add/Subtract) line of the ALU is pulled low for subtraction and the final result is loaded into the accumulator.

The ADD/SUB line is now held at logic level '1' and the ROUND line at '0' for rounding the output of the filter to 8 bits. This is accomplished by adding the contents of the accumulator and the delay flipflop DF2 (holding the 2^{-8} bit of the result) in the ADD/SUB unit.

The rounded output thus obtained is scaled up by the factor S and loaded into the shift register SR4 through a skewed parallel connection, and the filter is ready to process another sample.

The details of the control circuit required to synchronize and provide timing pulses for the filter and the interfacing hardware is given in [17]

2.4 FILTER OPERATIONS AND TESTING

The conventional Peled Liu approach was implemented in software (see appendix for the listing of the program.) as well as in hardware.

A second order lowpass butterworth filter with a cutoff frequency of 100 hz. was designed and used for the purpose. The corresponding difference equation is given by

$$y(n) = 0.019804 \cdot x(n) + 0.039608 \cdot x(n-1) + 0.019804 \cdot x(n-2) + 1.564376 \cdot y(n-1) - 0.6435943 \cdot y(n-2) \quad (2.10)$$

The ROM contents of the difference equation were derived using table (2.1), with a scale-factor of 2. A Fortran program to compute and scale the values of F(.) for a given second order IIR filter was developed and used for the purpose. The program is listed in the appendix.

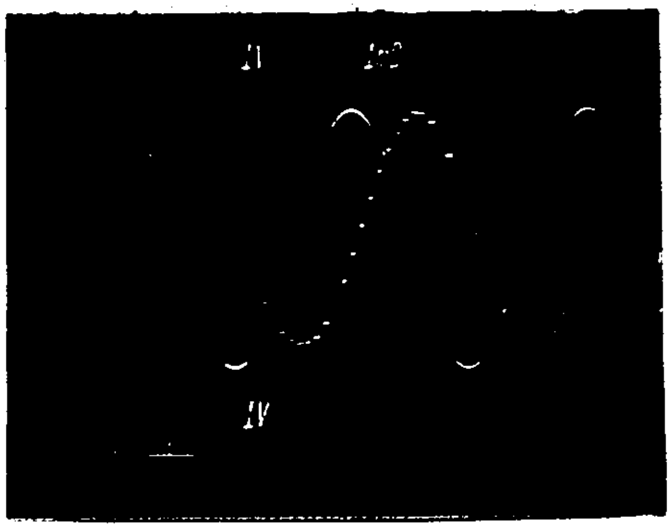
A ± 1 volt test sine wave of frequency ranging from 10 hz. to 1 KHz. was fed as an input to the filter. Fig. (2.3) represents the photographs of the actual input/output waveforms of the filter for three test frequencies.

i) f = 50 hz. (pass band)

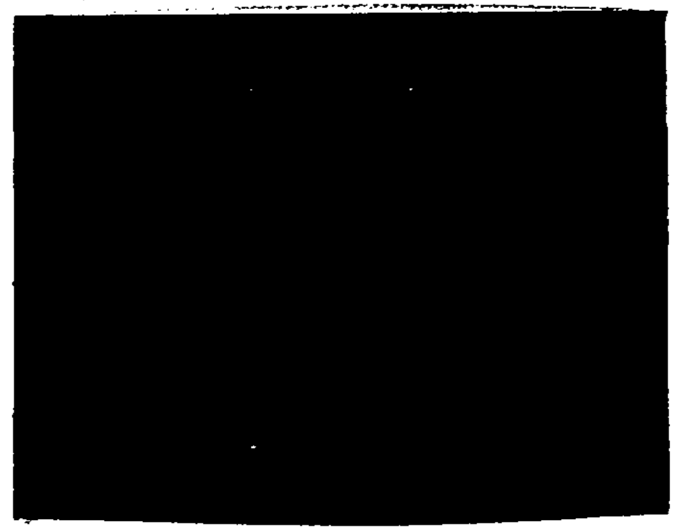
ii) f = 100 hz. (cutoff freq.)

iii) f = 400 hz. (stop band)

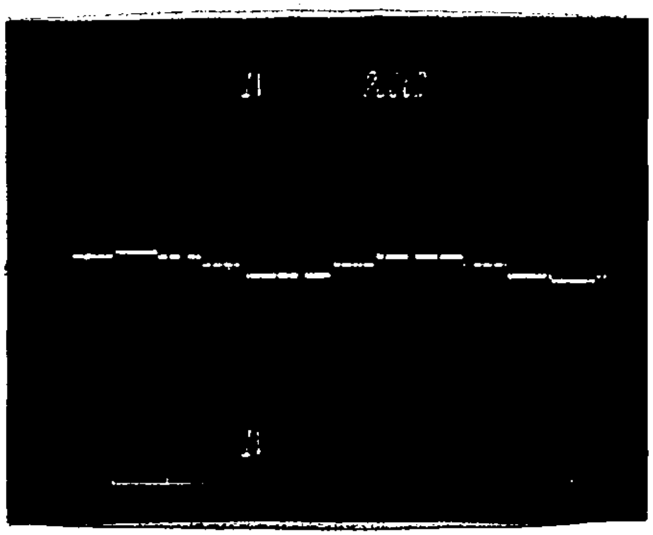
Fig. (2.4) depicts the actual and the ideal frequency response of the filter. The ideal frequency response is obtained by directly computing the difference equation (2.8) on IBM 370/3031 computer.



(a)



(b)



(c)

Input Waveform ————

Output Waveform — — — —

Fig. (2.3) Typical I/O Waveforms of a Low-Pass Filter Realized Using Peled-Liu Approach.

a) Frequency in the passband (f = 50 Hz)
b) Frequency at cutoff (f = 100 Hz)
c) Frequency in the stopband (f = 400 Hz)

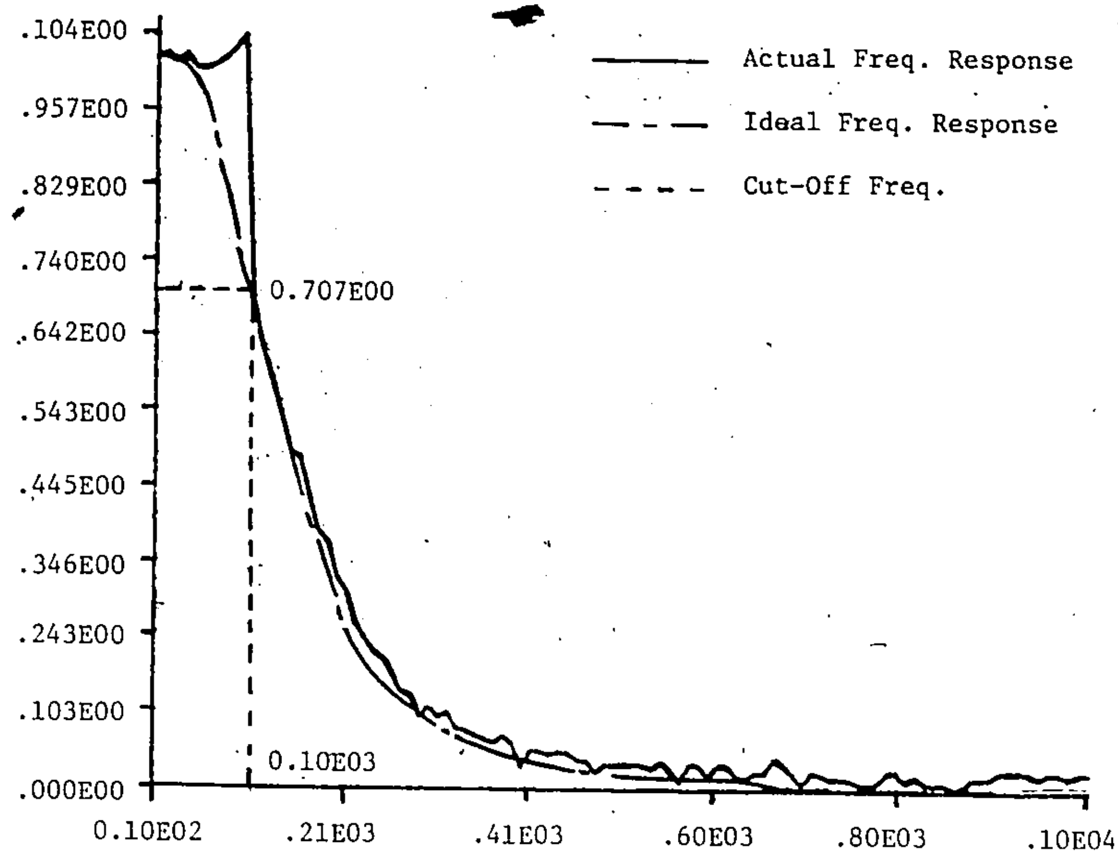


Fig. (2.4) Actual and Ideal Frequency Response of a Low-Pass Filter.

Chapter III

RESIDUE CODED COMBINATORIAL RECURSIVE DIGITAL FILTERS

Earlier in chapter 2, hardware implementation of a recursive digital filter transfer function using Peled-Liu bit slicing approach was discussed. The implementation was based on the use of fixed point arithmetic with negative numbers represented in two's complement code. It thus utilizes fixed radix number system. Despite numerous advantages of the weighted number system , its basic limitation is that truely parallel arithmetic, in which all digits are processed simultaneously, is not possible because of carry propogation. In recent years attempts have been made to circumvent this limitation. These approaches can be classified into two categories :

i) Methods to reduce carry propogation time by adding specialized Look Ahead carry circuitry.

ii) The use of number system with special carry characteristics i.e. The Residue Number System.

A brief introduction of RNS and its corresponding arithmetic is discussed in the next section; further details of which is provided in [18].

3.1 RESIDUE NUMBER ARITHMETIC

A residue number system is constructed from a set of pairwise relatively prime moduli $\{m_1, m_2, \ldots m_L\}$. The total number range of the system is defined as

$$M = \prod_{i=1}^{L} m_i \tag{3.1}$$

where L is the number of moduli used to formulate the number system.

The total number range is partitioned so as to accomodate negative numbers. If M is odd, the dynamic range of the residue representation is $[-(M-1)/2\ ,\ (M-1)/2]$. If M is even the dynamic range is $[-M/2\ ,\ M/2-1]$.

Each integer, x, within the dynamic range can be uniquely represented as a sequence of residue digits as shown below.

$$x = (x_1, x_2, \ldots, x_L) \tag{3.2}$$

where $x_i = |x|_{m_i} \qquad x \geq 0$

$\qquad m_i - |x|_{m_i} \quad x < 0 \quad i = 1, 2, \ldots, L$

and $|x|_{m_i}$ denotes the least positive residue of x with respect to the modulus m_i. Residue operations can be defined as follows :

$$(x_1, x_2, \ldots, x_L) * (y_1, y_2, \ldots, y_L) = (z_1, z_2, \ldots, z_L) \tag{3.3}$$

where $z_i = | x_i * y_i |_{m_i}$, $i = 1, \ldots, L$

and * denotes addition, subtraction or multiplication.

General division is a complicated process of RNS, and fortunately not required in the implementation of digital filters. However a restricted form of division in which the divisor is a fixed constant is often required in the implementation. This fixed constant division is called scaling.

Scaling is required in residue coded recursive digital filter implementation, because stable recursive difference equations generally have fractional or mixed fractional coefficients that cannot be represented in an integer number system. These fractional coefficients must be converted to integers by multiplying them with an appropriate scale-factor S and rounding them to the nearest integer as illustrated in eqn. (3.4) for a second order recursive filter section.

$$\begin{aligned} y_s(n) = \frac{1}{S} \{ & [S \cdot a_0]_r \cdot x(n) + [S \cdot a_1]_r \cdot x(n-1) \\ & + [S \cdot a_2]_r \cdot x(n-2) - [S \cdot b_1]_r \cdot y(n-1) \\ & - [S \cdot b_2]_r \cdot y(n-2) \} \end{aligned} \tag{3.4}$$

The notation $[S \cdot a_0]_T$ indicates that a_0 is multiplied by the scale-factor S and the result is rounded to the nearest integer for representing eqn. (3.4). After the expression within the brackets {.} of eqn.(3.4) has been computed the result must be divided by S and quantized to an integer value so that y(n) is available for further iteration.

There are two general techniques of scaling within the RNS. These techniques are based on a weighted magnitude conversion, and requires that the scale-factor be chosen as a product of some moduli. The two scaling techniques are the method of exact division [18] and the summation of metric vector estimates [22].

3.1.1 Exact Division scaling

This technique uses the multiplicative inverses to perform the division using the theorem

$$\left|\frac{b}{a}\right|_{m_i} = \left|\frac{1}{a} \cdot b\right|_{m_i} \qquad (3.5)$$

where b is the dividend, a is the divisor and the symbol $\left|\frac{1}{a}\right|_{m_i}$ represents the multiplicative inverse of a mod m_i. The proof of the theorem is given in [18]. The theorem is valid only if the dividend b is an exact multiple of the divisor a.

The use of the theorem can be extended for the purpose of scaling by selecting a divisor that is equal to one or product of some moduli. The details of the manipulations required for the purpose is illustrated below. Let X be the input to the scaling process, Y be the output, and K the scaling factor. Then

$$Y = \left[\frac{X}{K} \right] \qquad (3.6)$$

where [•] means the integer value of the number in the brackets. This can be written as :

$$X = YK + |X|_K \qquad (3.7)$$

which yields

$$Y = \frac{X - |X|_K}{K} \qquad (3.8)$$

The term $X - |X|_K$ in eqn. (3.8) is an exact multiple of K. Hence using eqn. (3.5), eqn. (3.8) can be expressed as

$$y_i = |Y|_{m_i} = \left| \, \left| X - |X|_K \right|_{m_i} \cdot \frac{1}{a} \right|_{m_i} \qquad (3.9)$$

If we consider scaling by the first S' moduli of the RNS base vector elements, then the following recursive form constitutes the scaling algorithm [21].

$$\left| \phi^{(k+1)} \right|_{m_i} = \left| \left| \phi^{(k)} - \phi_k \right|_{m_i} \cdot \left| \frac{1}{m_k} \right|_{m_i} \right|_{m_i} \qquad (3.10)$$

with

$$\phi^{(1)} = X, \ \phi^{(S'+1)} = \left[\frac{X}{\prod_{i=1}^{S'} m_i} \right] \text{ and } \phi_k = \left| \phi^{(k)} \right|_{m_k}$$

for $0 < k < S'-1$; $k < i < N-1$.

The remaining S' residues may be determined by the base extension method [18]. Hence the scaling algorithm actually consists of two operations namely, division remainder zero and extension of base.

3.1.2 Scaling Using Estimates

An alternate procedure to scaling by exact division, using multiplicative inverses, is to use scaled metric vector estimates.

From the Chinese Remainder Theorem (CRT), we can express the input x in terms of its residue representation [18].

$$x = \sum_{i=1}^{L} \hat{m}_i \left| \frac{x_i}{\hat{m}_i} \right|_{m_i} - A(x) \cdot M \tag{3.11}$$

where A(x) is an integer function of X and

$$\hat{m}_i = \frac{M}{m_i} \; ; \; \left| \frac{x_i}{\hat{m}_i} \right|_{m_i} = \left| x_i \cdot \left| \frac{1}{\hat{m}_i} \right|_{m_i} \right|_{m_i}$$

Eqn. (3.11) can be rewritten in terms of summation of metric vectors as

$$x = \sum_{i=1}^{L} x_i \cdot U_i - A(x) \cdot M \tag{3.12}$$

where the magnitude of the unit metric vector associated with the i^{th} modulus is given by

$$U_i = \hat{m}_i \left| \frac{1}{\hat{m}_i} \right|_{m_i} \tag{3.13}$$

From eqn. (3.12) we can find that

$$Y_E = \frac{X}{\frac{S'}{\prod_{j=1} m_j}} = \sum_{i=1}^{L} x_i \cdot \frac{U_i}{\frac{S'}{\prod_{j=1} m_j}} - A(x) \cdot \frac{M}{\frac{S'}{\prod_{j=1} m_j}} \qquad (3.14)$$

where Y_E is in the set of all real numbers and is not in general an integer. We can form a scaling function based on the summation of scaled metric vectors in eqn. (3.14).

$$\dot{Y}' = \sum_{i=1}^{L} \left[\frac{x_i \cdot U_i}{\frac{S'}{\prod_{j=1} m_j}} + \frac{1}{2} \right] = \sum_{i=1}^{L} \frac{x_i \cdot U_i}{\frac{S'}{\prod_{j=1} m_j}} + \varepsilon \qquad (3.15)$$

where ε is an error resulting from the summing estimates of the scaled metric vectors. An obvious upper bound on is given by $|\varepsilon| < L/2$.

From eqn. (3.13) and eqn. (3.15)

$$\left|Y'\right|_{m_i} = \left| \sum_{i=1}^{L} \left| \left[\frac{\prod_{j=s'+1}^{L} m_j \left| \frac{x_i}{\hat{m}_i} \right|_{m_i}}{m_i} + \frac{1}{2} \right] \right|_{m_i} \right|_{m_i} \quad (3.16)$$

We note that for $S'+1 < n < L$, the remainder of the rounding process is zero. The reconstruction of the first S residues can be done through the base extension method.

3.2 RESIDUE CODED COMBINATORIAL DIGITAL FILTERS

Residue number concepts and combinatorial techniques can be combined to yield very efficient hardware architecture for conventional cascade or parallel digital filters [20].

To accomplish this, the input/output sample D_k is encoded in hardware as a binary integer after multiplying it with a suitable scale-factor X_S to suit the integer representation as

$$D_k = \sum_{\ell=0}^{R-1} 2^{\ell} b_{k\ell} \quad (3.17)$$

Here R represents the dynamic range of the filter and $b_{k(R-1)} \ldots b_{ko}$ represents the binary integers of the input sample. Rewriting eqn.(2.1) and substituting eqn.(3.17) in eqn.(2.1) we get

$$y(n) = \sum_{k=1}^{N} C_k \cdot D_k$$

$$= \sum_{\ell=0}^{R-1} 2^{\ell} F(b_{1\ell}, \ldots, b_{N\ell}) \qquad (3.18)$$

where c_k represents the coefficients of the filter transfer function.

N represents the total number of the filter coefficients. (N=5 for a second order filter.)

and $F(b_{1\ell}, \ldots, b_{N\ell}) = \sum_{k=1}^{N} b_{k\ell} \cdot C_k$

Thus y(n) can be computed by addressing the stored function $F(b_{1\ell} \ldots b_{N\ell})$, shifting and adding. If the filter coefficents are required to have large number of bits then the arithmetic operations in the above equations will be executed with long wordlength operands, and the memory will need long wordlength to accomodate F(.). In such a case it may be advantageous to encode the operations into a residue system so that the operations can be executed with shorter residue operands in parallel subfilters. The computation then takes the form

$$\left|y(n)\right|_{m_i} = \left|\sum_{\ell=0}^{R_{i-1}} 2^{\ell}\, F_i(b_{1\ell}, \ldots, b_{N\ell})\right|_{m_i} \tag{3.19}$$

where $F_i\,(b_{1\ell} \cdots b_{N\ell}) = |F\,(b_{1\ell} \cdots b_{N\ell})|_{m_i}$. The combinatorial algorithm can be implemented by using more than a single bit from each data sample. When the residue number system consists of many small moduli e.g. when each moduli needs only 5 bits the computation can be arranged as follows :

$$\left|y(n)\right|_{m_i} = F_3\left[\left\{F_1\left(\left|D_2\right|_{m_i}, \left|D_3\right|_{m_i}\right) + F_2\left(\left|D_4\right|_{m_i}, \left|D_5\right|_{m_i}\right)\right\} + \left|D_1\right|_{m_i}\right] \tag{3.20}$$

where

$$F_1(|D_2|_{m_i}, |D_3|_{m_i}) = \left|\left|\sum_{\ell=0}^{4} 2^{\ell} \cdot x(n-1)\cdot s\cdot a_1\right|_{m_i} + \left|\sum_{\ell=0}^{4} 2^{\ell} \cdot x(n-2)\cdot s\cdot a_2\right|_{m_i}\right|_{m_i} \tag{3.21}$$

$$F_2(|D_4|_{m_i}, |D_5|_{m_i}) = \left|\left|\sum_{\ell=0}^{4} 2^{\ell}\, y(n-1)\cdot(-s\cdot b_1)\right|_{m_i} + \left|\sum_{\ell=0}^{4} 2^{\ell}\, y(n-2)\cdot(-s\cdot b_2)\right|_{m_i}\right|_{m_i} \tag{3.22}$$

$$\left|D_1\right|_{m_i} = \left| \sum_{\ell=0}^{4} 2^{\ell} \cdot x(n) \cdot s \cdot a_0 \right|_{m_i} \qquad (3.23)$$

A program to compute the functicr tables $F_1(\cdot)$, $F_2(\cdot)$, and $F_3(\cdot)$ for all four moduli, given the coefficients of the difference eqn. is listed in the appendix. The resulting architecture is shown in fig. (3.1) for the i^{th} subfilter of an RNS system with base vector { 32,31,29,27 }. The scaler block provided at the end of the subfilter computes y (n) from $|y(n)|_{m_i}$. Details of the scaler block are given in the next section.

The architecture provides the capability for obtaining high speed and high precision simultaneously. The system is approximately equivalent to a conventional 2's complement system with 20 bits wordlengths[3] where the computation are quantized to 10 bits by rounding the lower half word.

The total propogation delay before an output sample is reached can be categorized as below. The typical access time of the shift register 74LS195 (= 30 nanoseconds) and 1K x 8 schottky TTL PROM DM577181 (= 40 nanoseconds) is used as a reference in the calculation of the propogation delay.

---

[3] Let $2^A = 32 \cdot 31 \cdot 29 \cdot 27$ (where A is the equivalent no. of bits in the weighted number system.) Then A=20 bits.

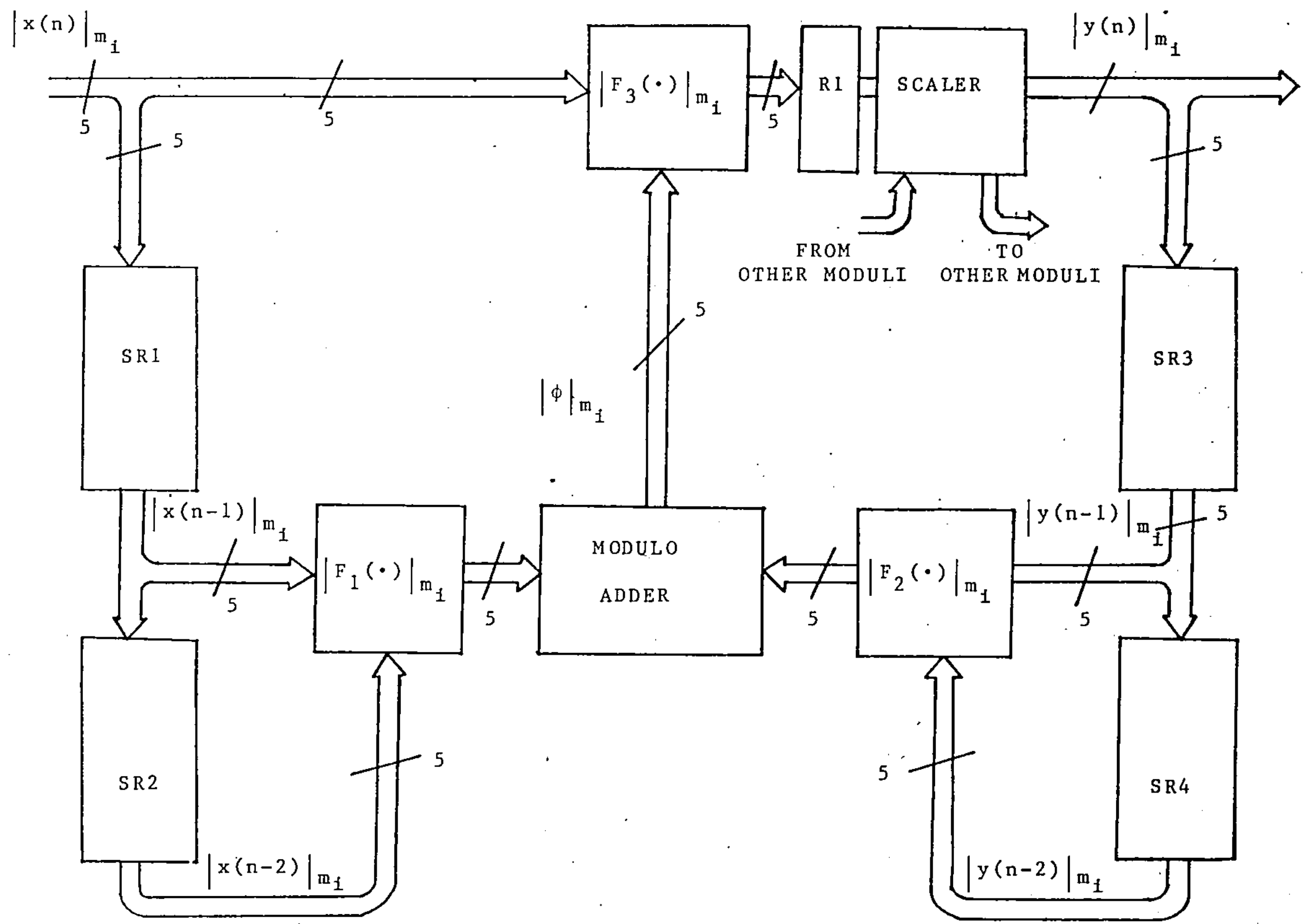


FIG.(3.1) An i^{th} Section of a Residue Coded Combinatorial 2nd Order Recursive Digital Filter.

(1) Total access time (t_a) from the interface buffer to the output of the function table $F_1(\cdot)$ or its equivalent.

$= t_a$ (SR1) + t_a $\{F_1(\cdot)\}$.

= 70 nanoseconds.

(2) Total access time (t_a) to formulate $|y(n)|_{m_i}$

$= t_a$ (Modulo Adder) + t_a $\{ F_3(\cdot) \}$.

= 80 nanoseconds.

(3) Total access time (t_a) to formulate $|y_s(n)|_{m_i}$ in the scaler block.

= 105 nanoseconds.

The total propogation delay; in computing one output sample, thus is the summation of (1), (2) and (3) which is equal to 255 nanoseconds enabling the filter to operate at a maximum speed of 4 MHz.

3.3 SCALER

The scaler block shown in fig. (3.1). is constructed using eqn. (3.16)

Let $S = m_1 \cdot m_2$ be the scale-factor of the four moduli number system used to implement the residue coded recursive digital filter. Then eqn. (3.16) can be rewritten as

$$\left| y_s(n) \right|_{m_k} = \left| \frac{y(n)}{s} \right|_{m_k} \tag{3.24}$$

$$= \left| \sum_{i=1}^{2} \left| \left[\frac{\prod_{j=3}^{4} m_j \left| \dfrac{x_i}{\hat{m}_i} \right|_{m_i}}{m_i} + \frac{1}{2} \right] \right|_{m_k} \right|_{m_k}$$

$$+ \left| \sum_{i=3}^{4} \left| \left[\frac{\prod_{j=3}^{4} m_j \left| \frac{x_i}{\hat{m}_i} \right|_{m_i}}{m_i} + \frac{1}{2} \right] \right|_{m_k} \right|_{m_k} \tag{3.25}$$

$$= F_1(y_1, y_2) + F_2(y_3, y_4) \tag{3.26}$$

Here $m_k = m_3 \cdot m_4$; the resultant scaled number system.

The functions $F_1(y_1, y_2)$ and $F_2(y_3, y_4)$ can be precomputed and stored in ROM for all possible combinations of $|y(n)|_{m_i}$, $i = 1, \ldots, 4$. The outputs $|y_s(n)|_{m_i}$ needed for all further iteration can simply be obtained as

$$|y_s(n)|_{m_i} = \left| |y_s(n)|_{m_k} \right|_{m_i} \tag{2.27}$$

$$= FM_i \qquad i = 1, \ldots, 4 \tag{3.28}$$

Thus similar to the functions $F_1(y_1, y_2)$ and $F_2(y_3, y_4)$ the functions FM_i can also be precomputed and stored in ROM for all possible combinations of $|y_s(n)|_{m_k}$.

The architecture for a four moduli scaler is shown in fig. (3.2). It requires 10 I/C packages and has 3 levels of propagation.

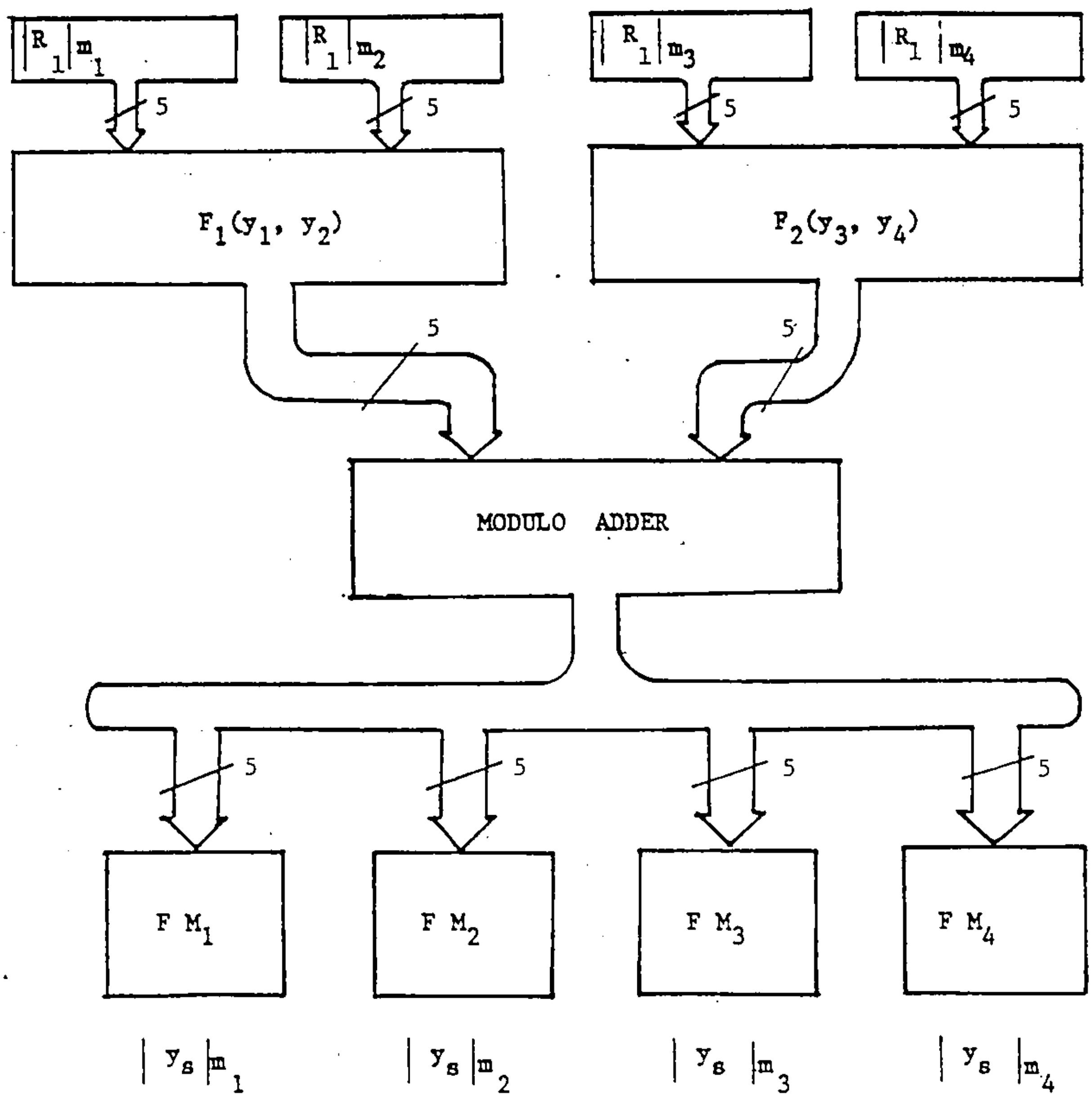


Fig. (3.2) Block Diagram of a Scaler

3.4 FILTER OPERATIONS AND TESTING

The filter structure of fig.(3.1) was simulated in software (the program is listed in the appendix.) using the second order difference egn.(2.10) for the purpose.

The difference equation was coded in RNS using the moduli {32,31,29,27} with a scale-factor of 32·31. The resulting coefficients along with the details of scaled and unscaled number range is listed below.

i) Total number range ; M = 776736

ii) Scale factor of the filter coefficients ; S = 992

iii) Scaled number range ; m_k = 783

For example the orignal coefficient

$$a_0 = 0.019804306$$

$$a_0 \cdot S = 0.019804306 \times 992$$

$$(a_0 \cdot S)_r = 20$$

Similarly the coefficients $(a_1 \cdot S)_r$,---,$(b_2 \cdot S)_r$ can be computed and coded in residues as shown in table 3.1

Fig.(3.3) shows the output of the filter when a delayed unit step function is fed as an input to the filter. A study of this waveform reveals that the output of the filter is stable and free from any limit-cycle oscillations.

---

[4] An examination of Figs.(3.4) to (3.7) reveals an apparent phase-shift between the input and the output waveforms of the RNS filter. This apparent phase-shift was caused by the exclusion of the transient portions of the output in the illustration.

TABLE 3.1

Coefficients Of The Filter In Mod Form

ORIGNAL COEFFICIENT	SCALED COEFFICIENT	m_1 (32)	m_2 (31)	m_3 (29)	m_4 (27)
a_0 = 0.019804	$(a_0 \cdot S)_r$ = 20	20	20	20	20
a_1 = 0.039608	$(a_1 \cdot S)_r$ = 39	7	8	10	12
a_2 = 0.019804	$(a_2 \cdot S)_r$ = 20	20	20	20	20
b_1 = −1.564376	$-(b_1 \cdot S)_r$ = 1552	16	2	15	13
b_2 = 0.643594	$-(b_2 \; S)_r$ = −638	2	13	0	10

Figs. (3.4) . to (3.7)* show the time domain outputs of the filter when a sine or square wave is fed as an input to the low-pass filter. The output waveform of the filter resembles the sinuscidal input to the filter in figs. (3.4) and (3.5). Also the integrating effects of the low-pass filter transfer function on the square wave inputs are clearly visible in figs. (3.6) and (3.7). This suggests that if the coefficients of the filter transfer function are suitably scaled to integers, the RNS preserves the characteristics of the filter transfer function in the limit of finite bit precision.

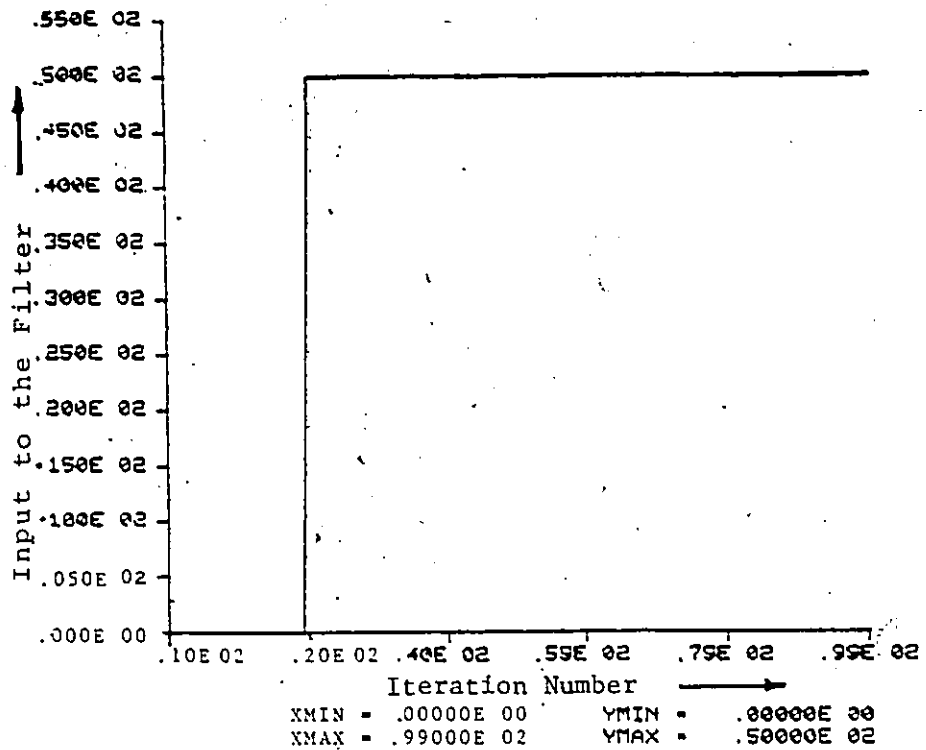
Fig. (3.8) shows the ideal and the actual frequency response of the low-pass filter transfer function. It is seen from the figure that both the frequency response resembles each other and that the 3-db cut-off frequency is preserved in the residue coded recursive digital filter. The ideal frequency response is computed by directly computing the difference equation (2.1) on IBM 370/3031 having 32 bits precision.

On a similar basis a second order high-pass recursive digital filter transfer function was also realized.

Figs. (3.9) to (3.12) show the time domain outputs of the filter when a sine or square wave is fed as an input to the high-pass filter.

Figs. (3.13) and (3.14) shows the ideal and the actual frequency response of the high-pass filter transfer function.

A study of these figures reveal that, like the low-pass filter transfer function {eqn. (2.10)}, the time domain and the frequency domain characteristics of the filter are also preserved here.



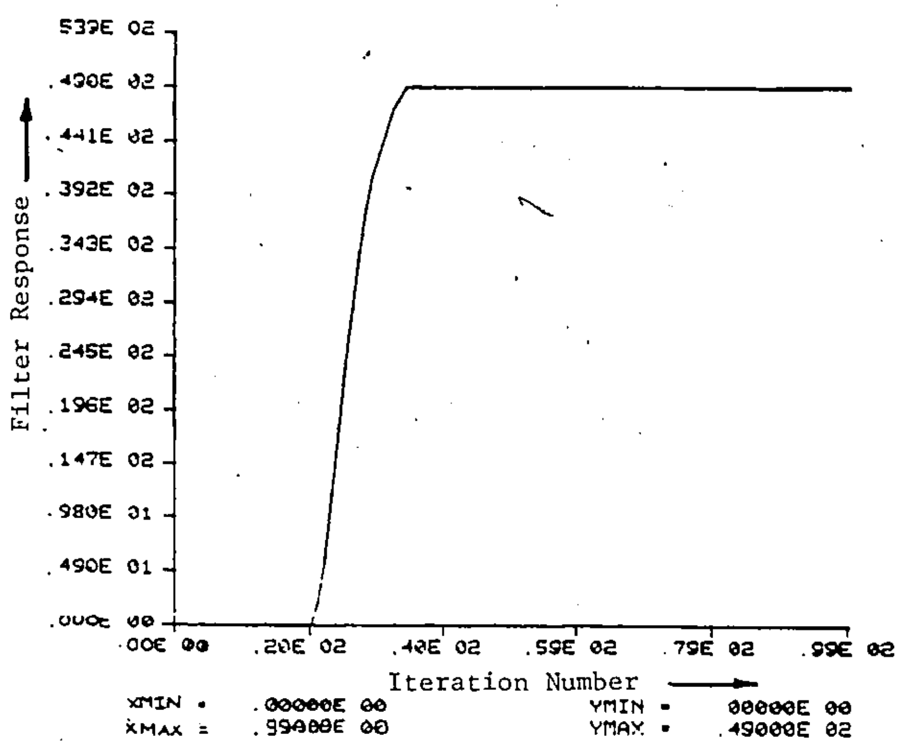


Fig. (3.3) Delayed Unit-Step Response of a Residue Coded Low Pass Filter.

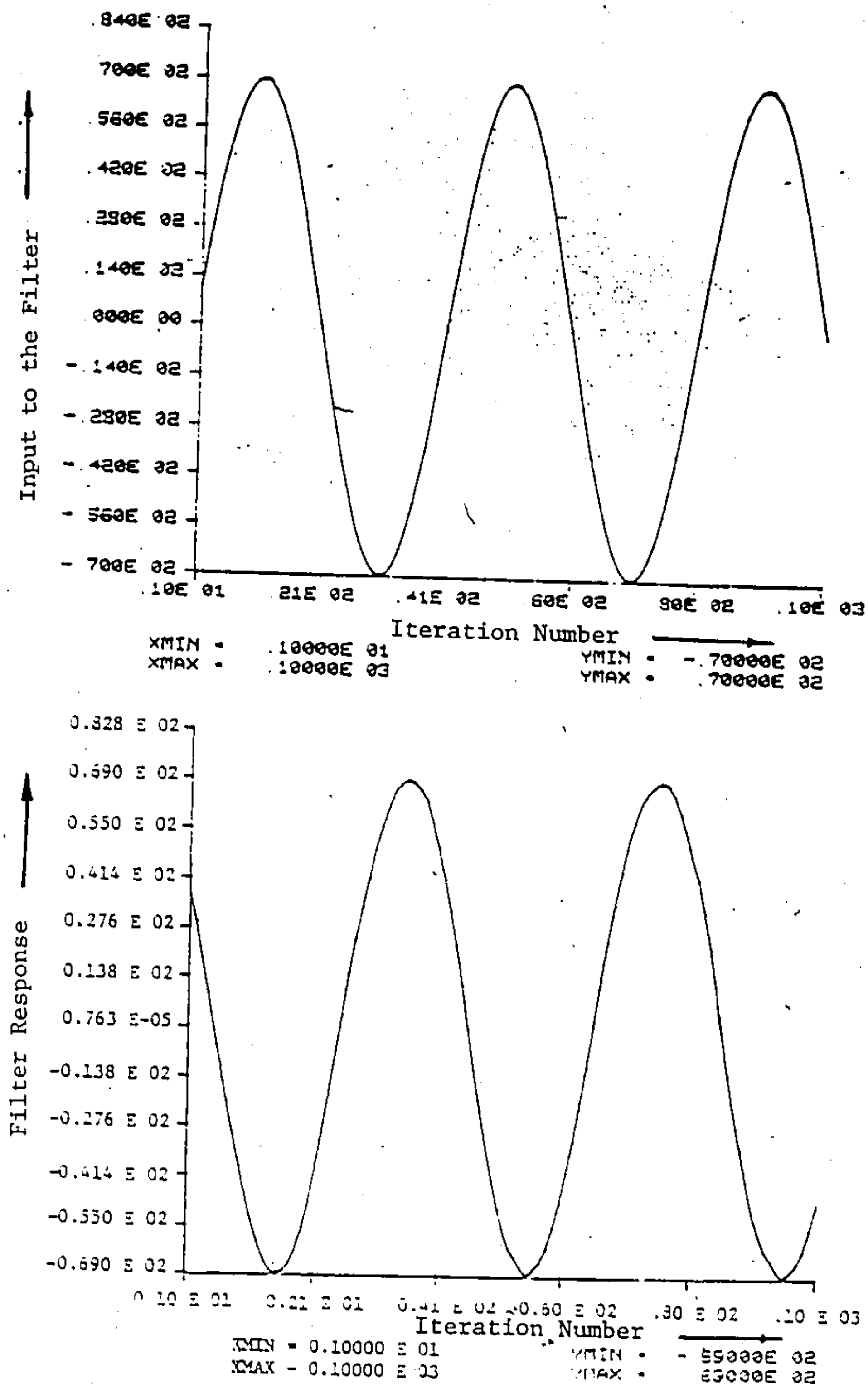
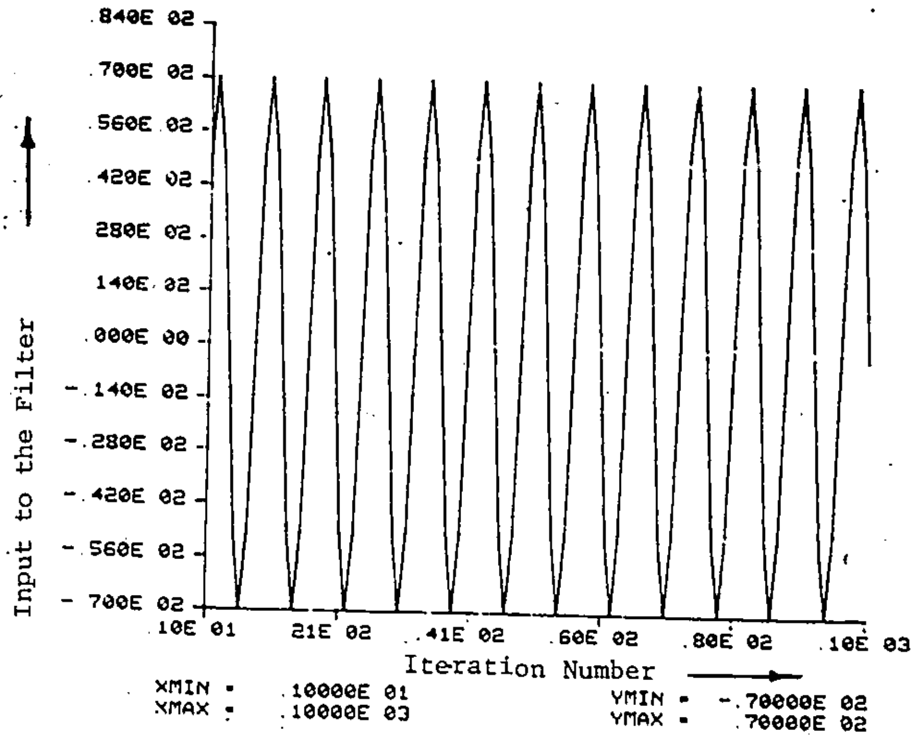


Fig. (3.4) Typical I/O Waveforms of a Residue Coded Low-Pass Filter
Test Input = Sine Wave at F = 50 Hz



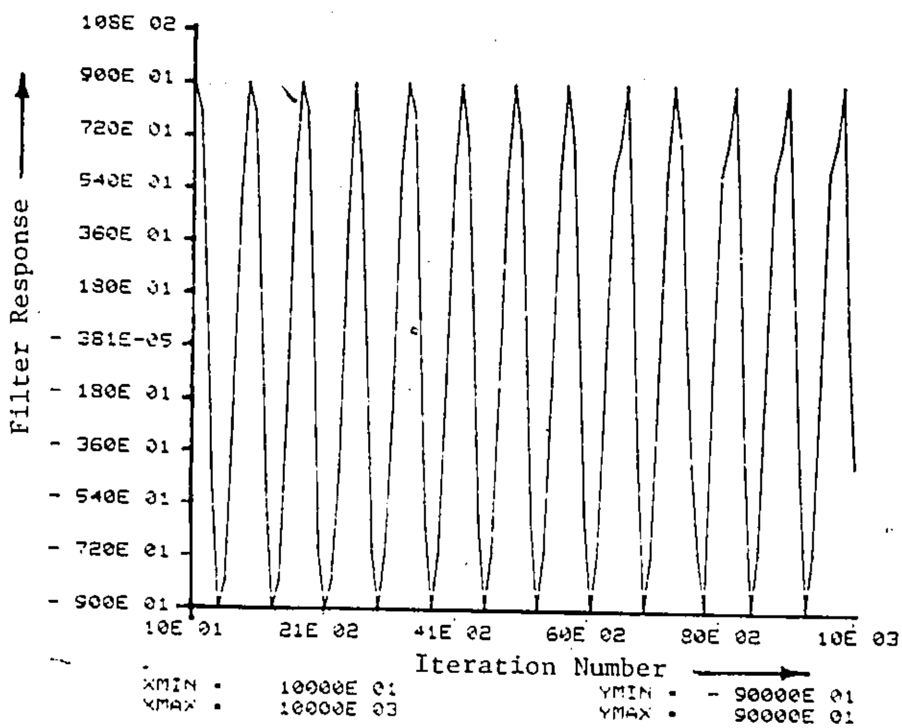


Fig. (3.5) Typical I/O Waveforms of a Residue Coded Low-Pass Filter.
Test Input = Sine Wave at F = 250 Hz.

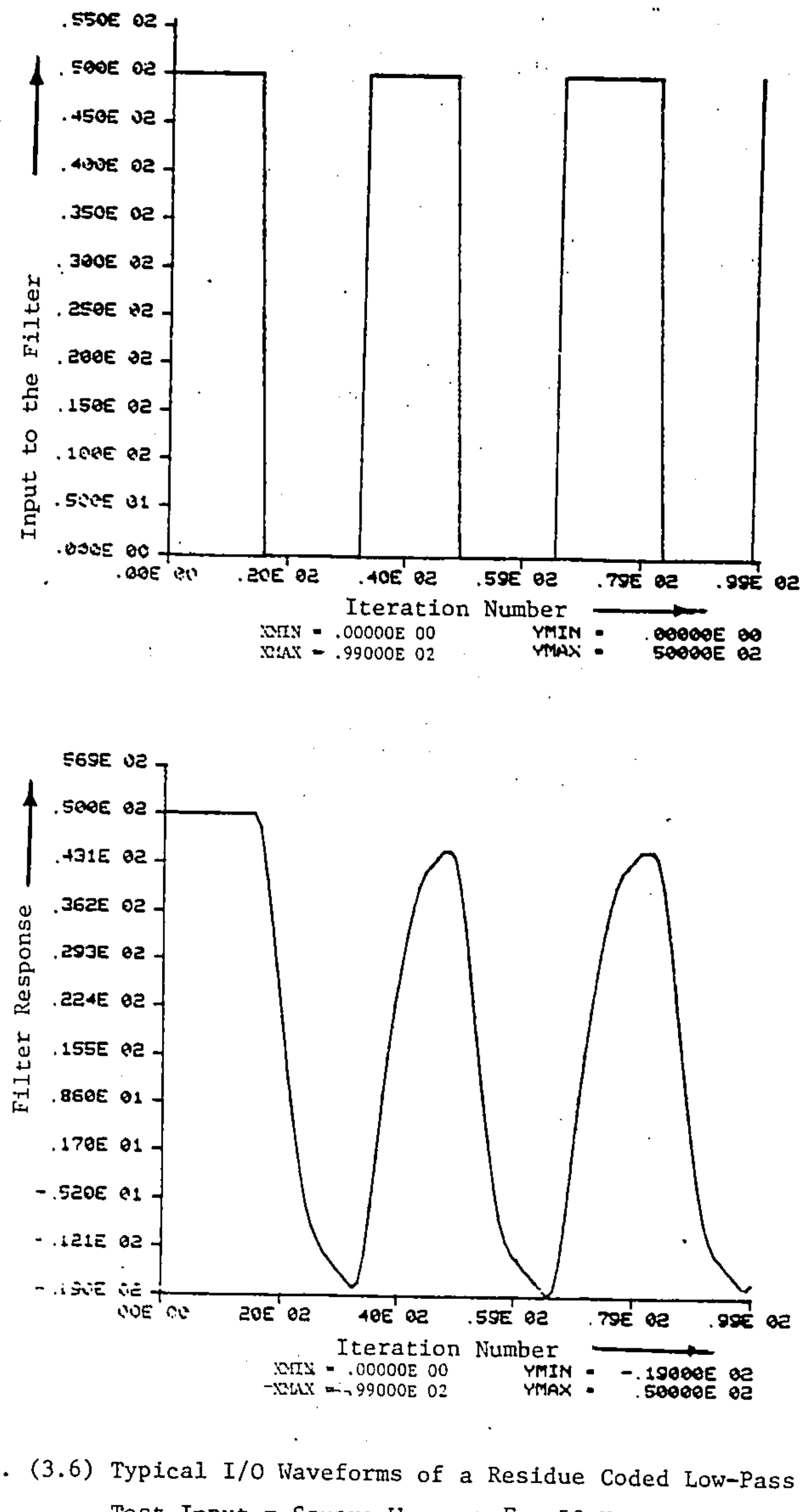
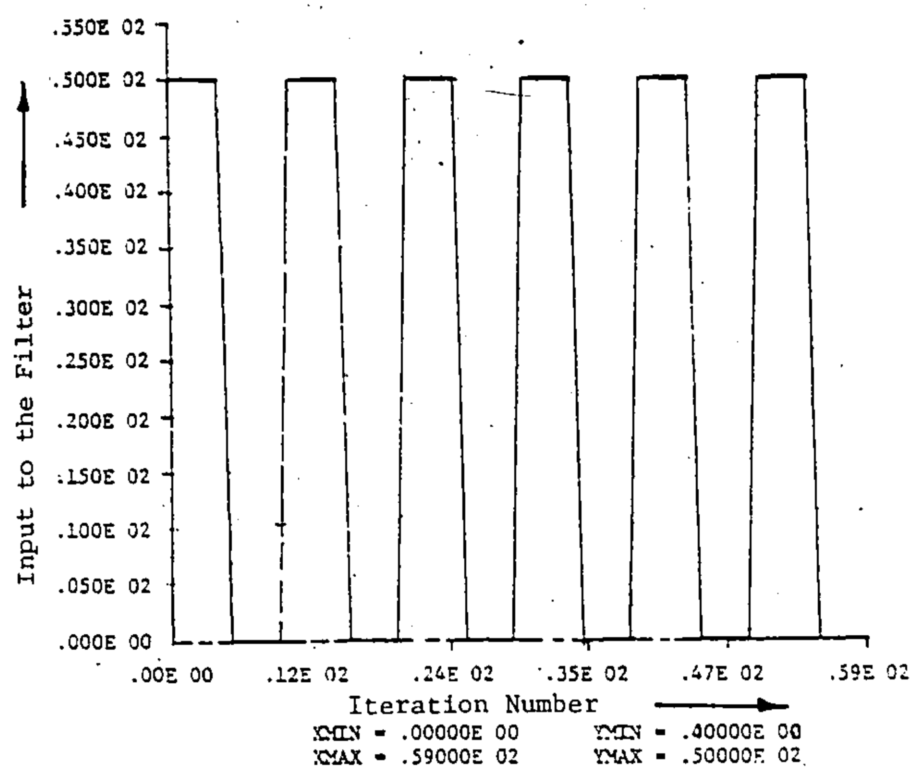


Fig. (3.6) Typical I/O Waveforms of a Residue Coded Low-Pass Filter.
Test Input = Square Wave at F = 50 Hz.



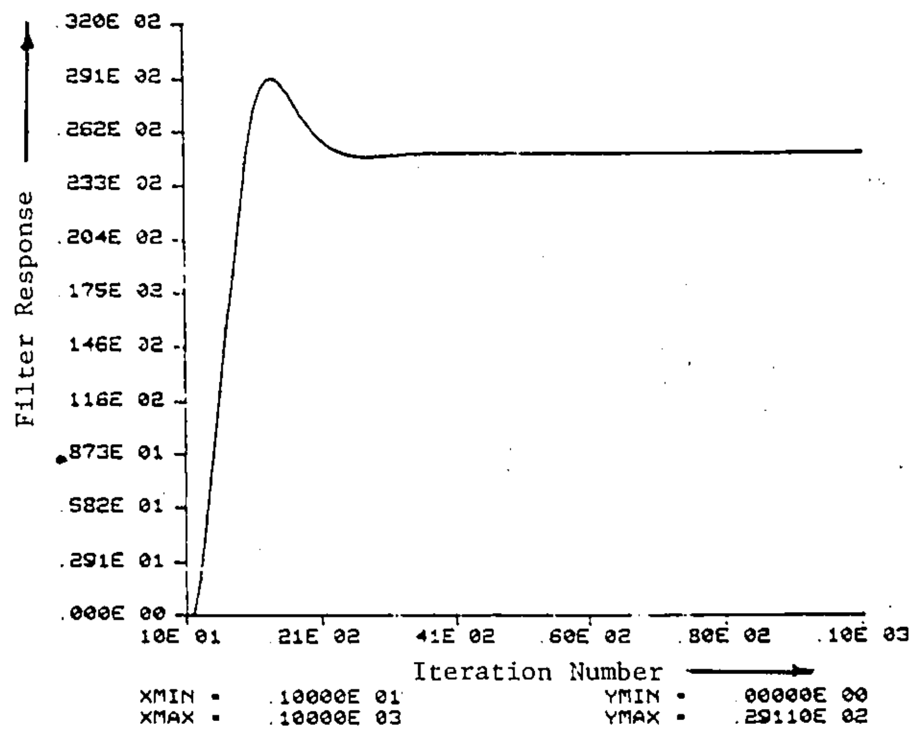


Fig. (3.7) Typical I/O Waveforms of a Residue Coded Low-Pass Filter.
Test Input = Square Wave at F = 150 Hz.

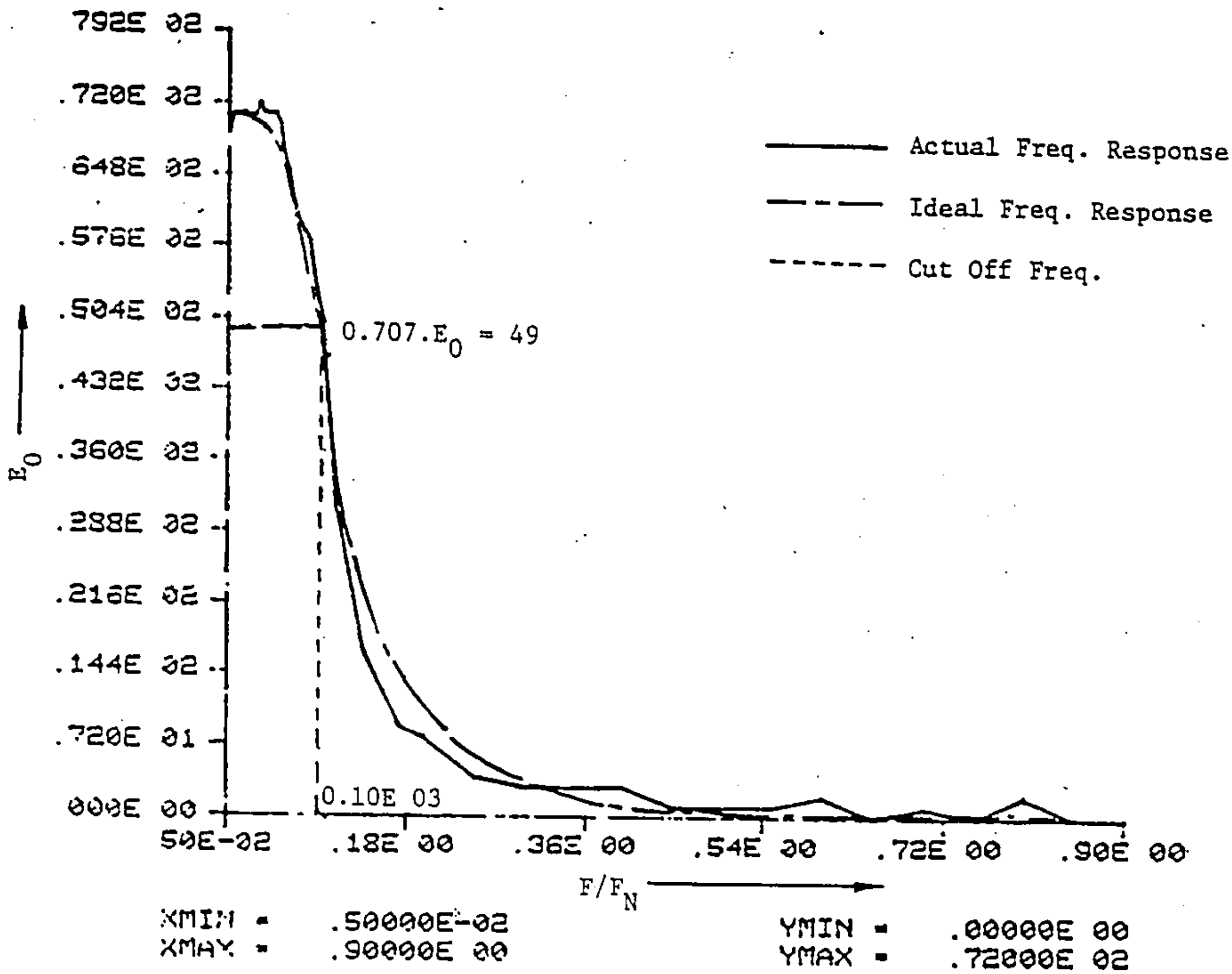


Fig. (3.8) Actual and Ideal Frequency Response of a Low Pass Filter Realized Using RNS.

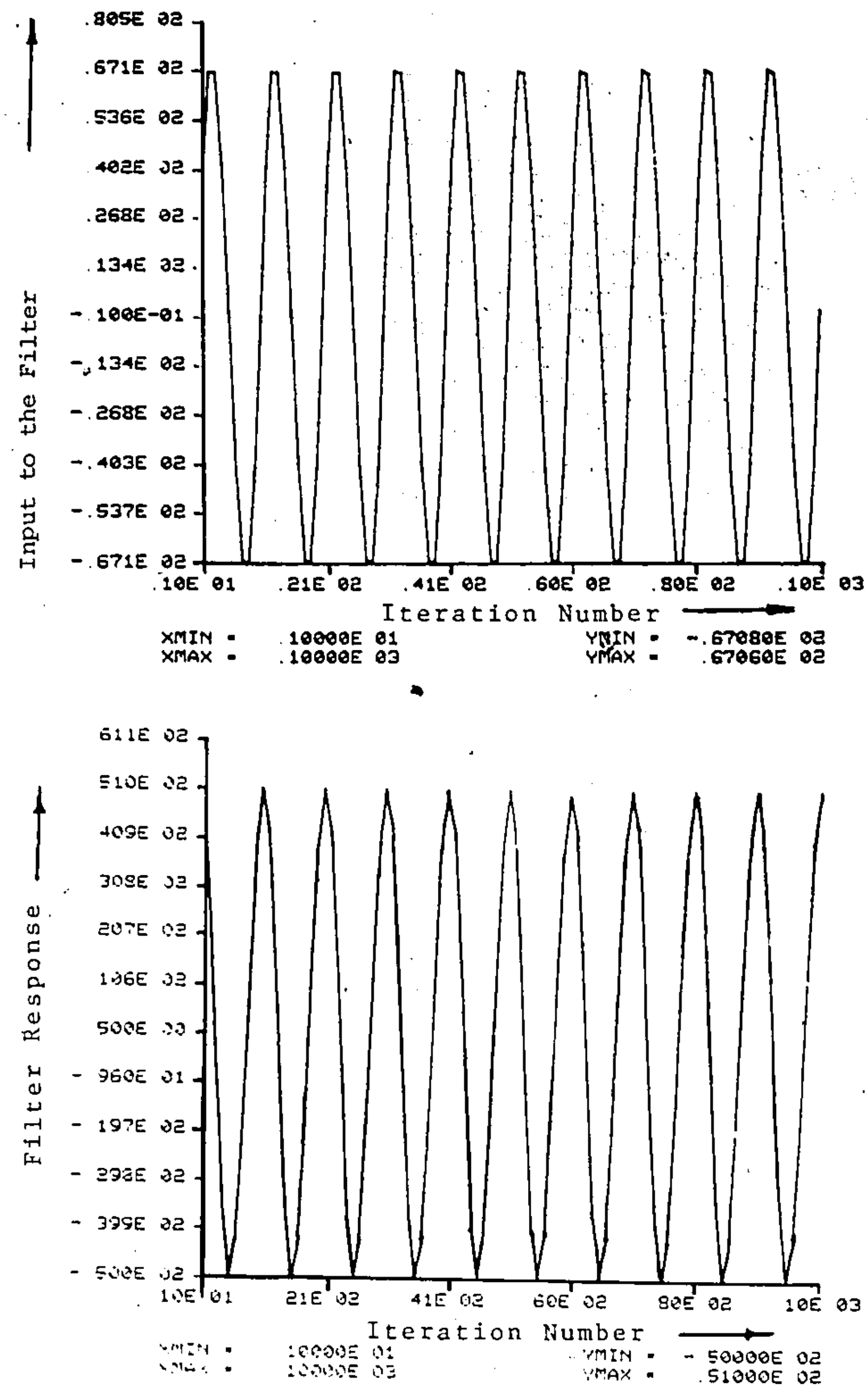
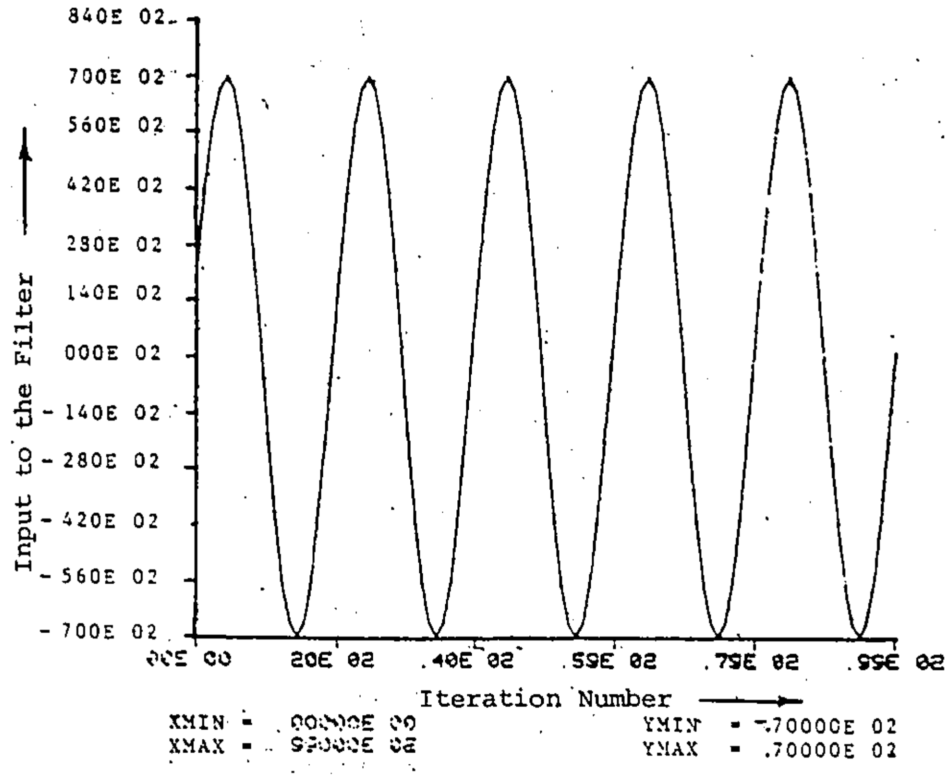


Fig. (3.9) Typical I/O Waveforms of a Residue Coded High-Pass Filter. Test Input = Sine Wave at F = 10 KHz.



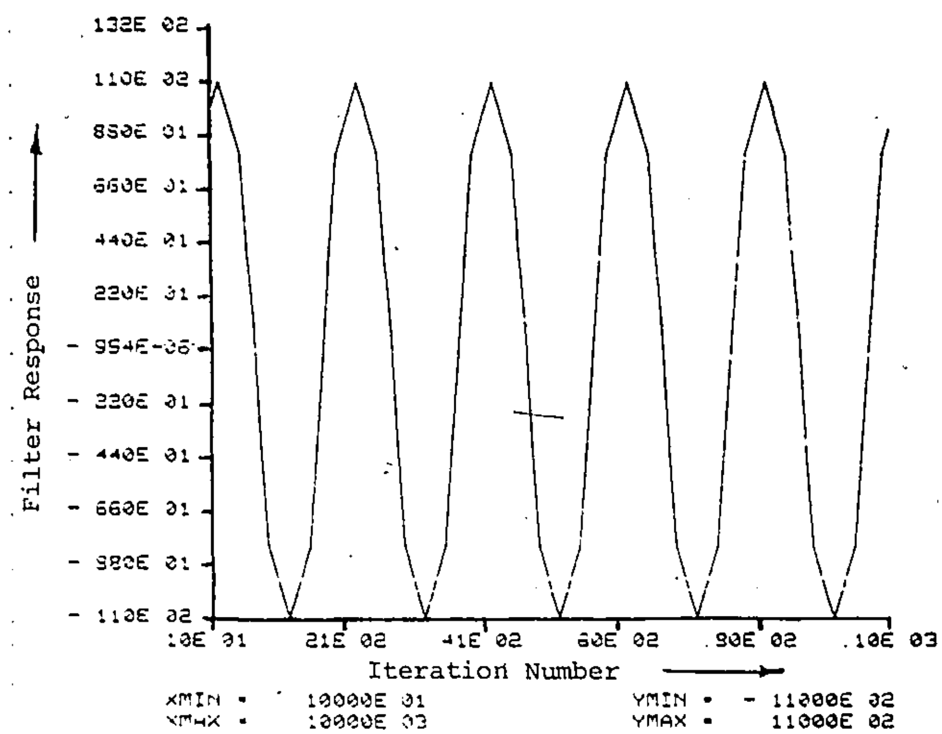
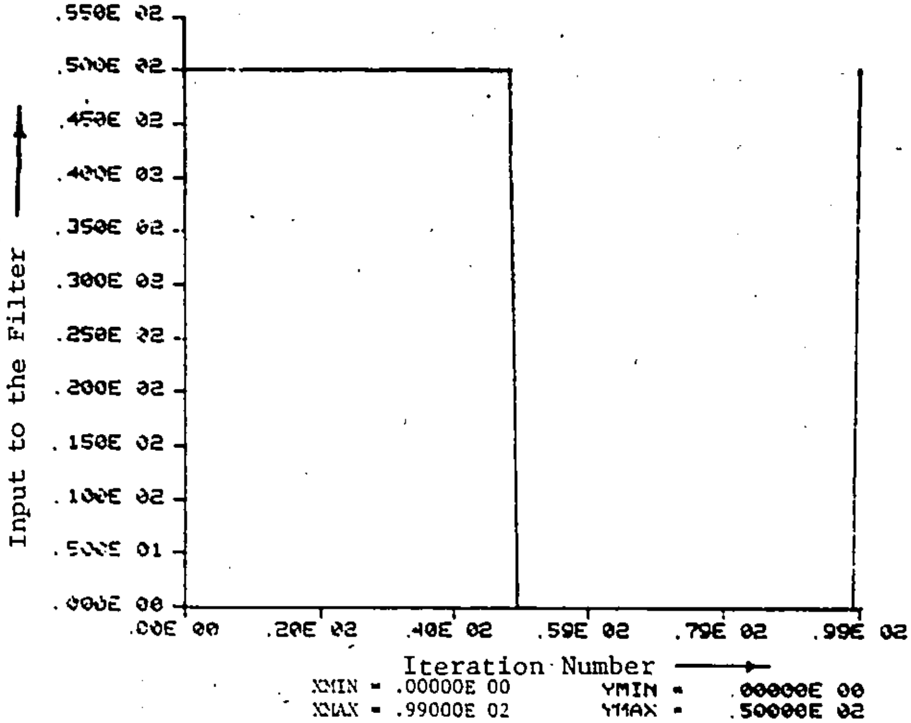


Fig. (3.10) Typical I/O Waveforms of a Residue Coded High-Pass Filter

Test Input = Sine Wave at F = 1 Khz



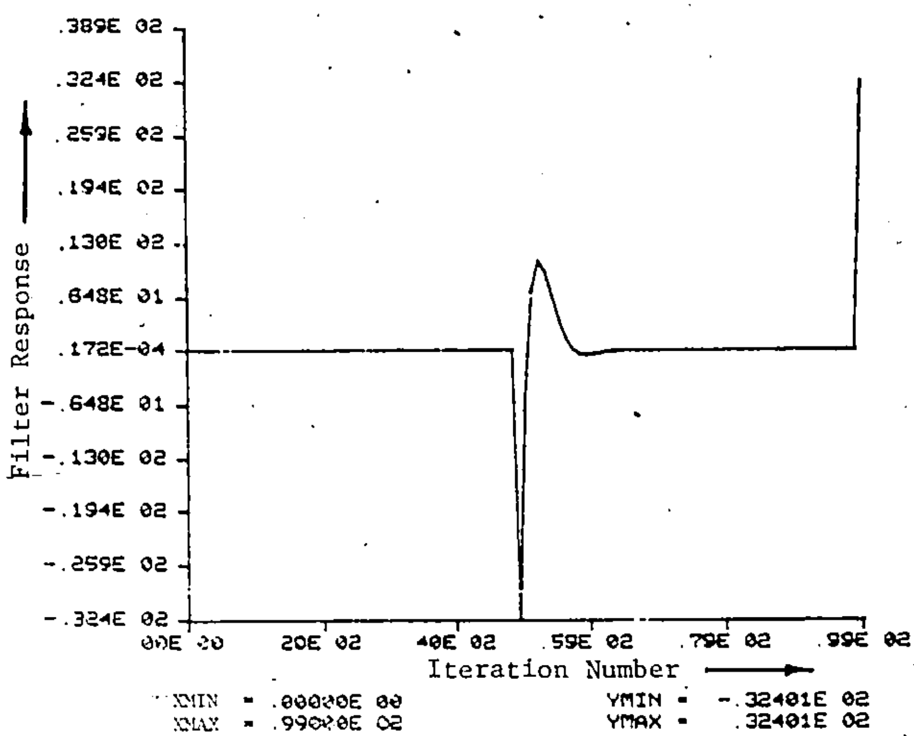
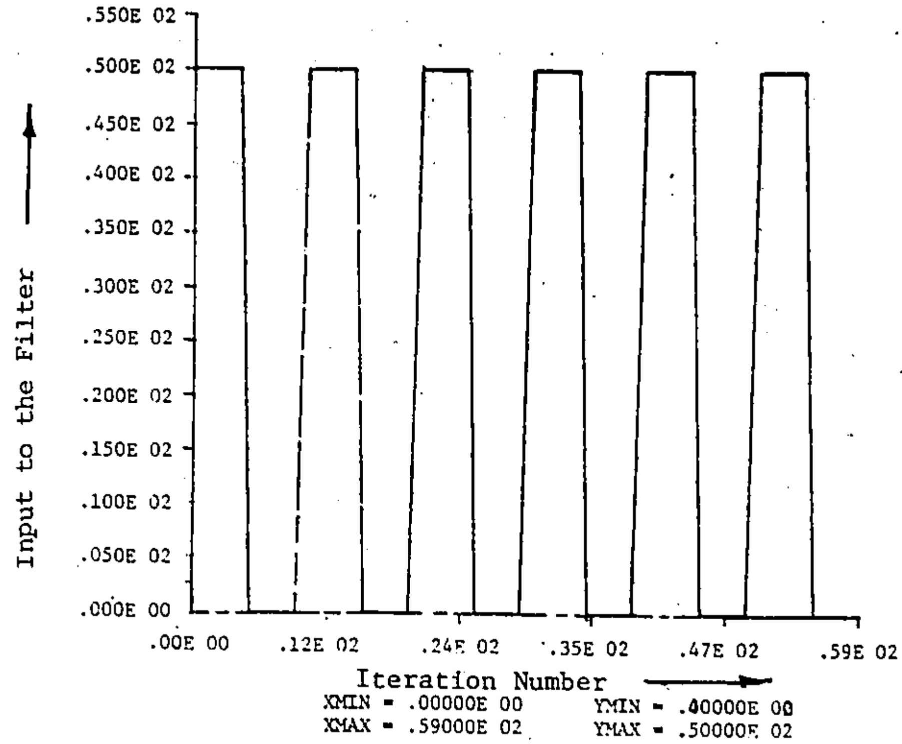


Fig. (3.11) Typical I/O Waveforms of a Residue Coded High-Pass Filter.
Test Input = Square Wave at F = 50 Hz.



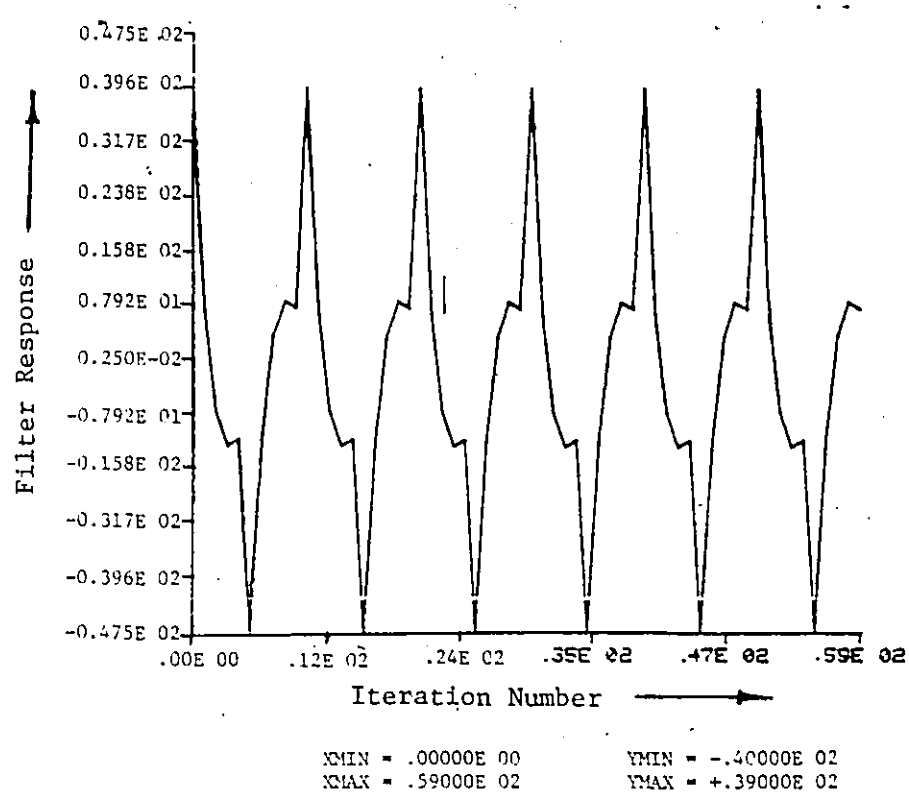


Fig. (3.12) Typical I/O Waveforms of a Residue Coded High-Pass Filter.

Test Input = Square Wave at F = 10 KHz.

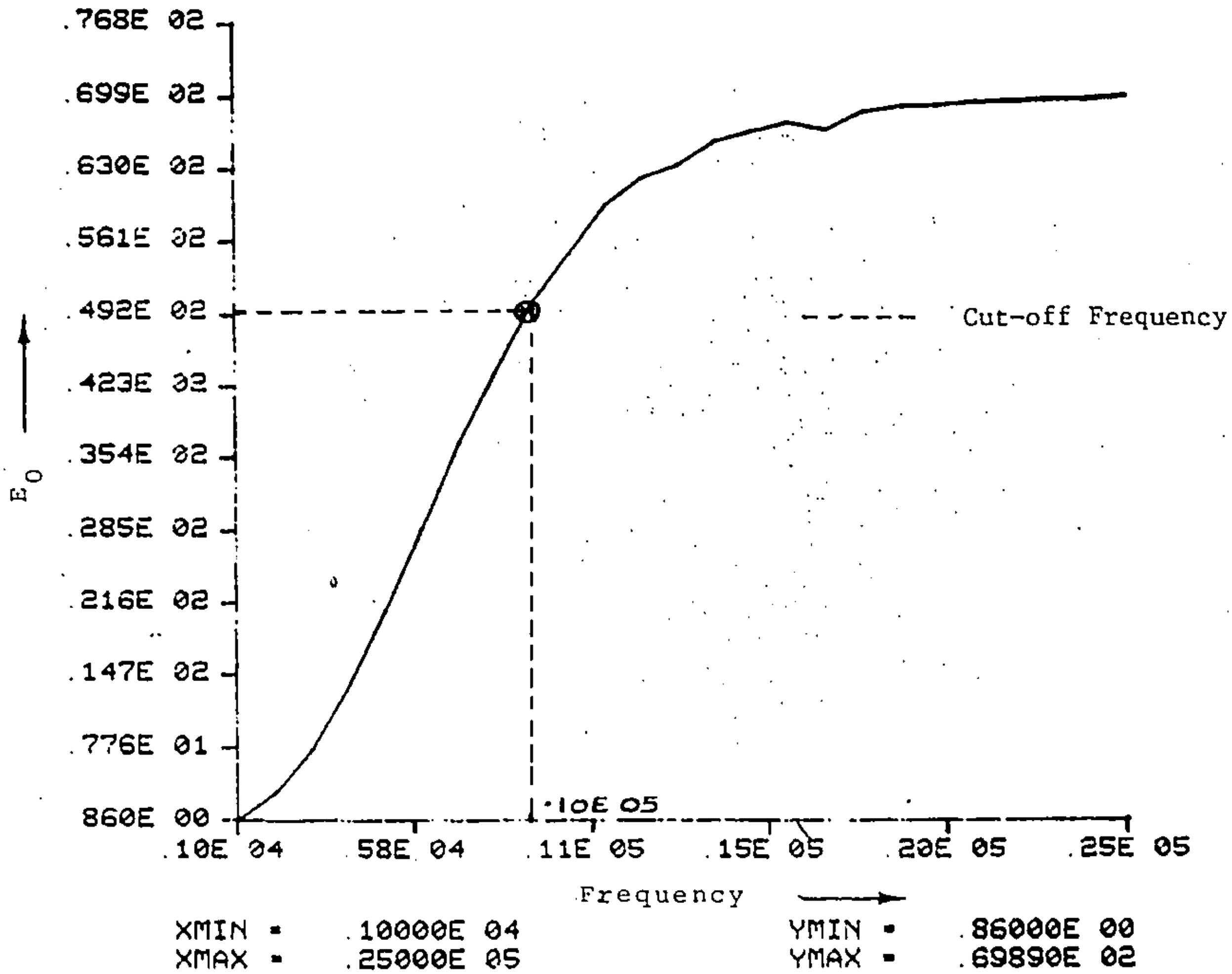


Fig. (3.13) Ideal Frequency Response of a Second Order High Pass Filter.

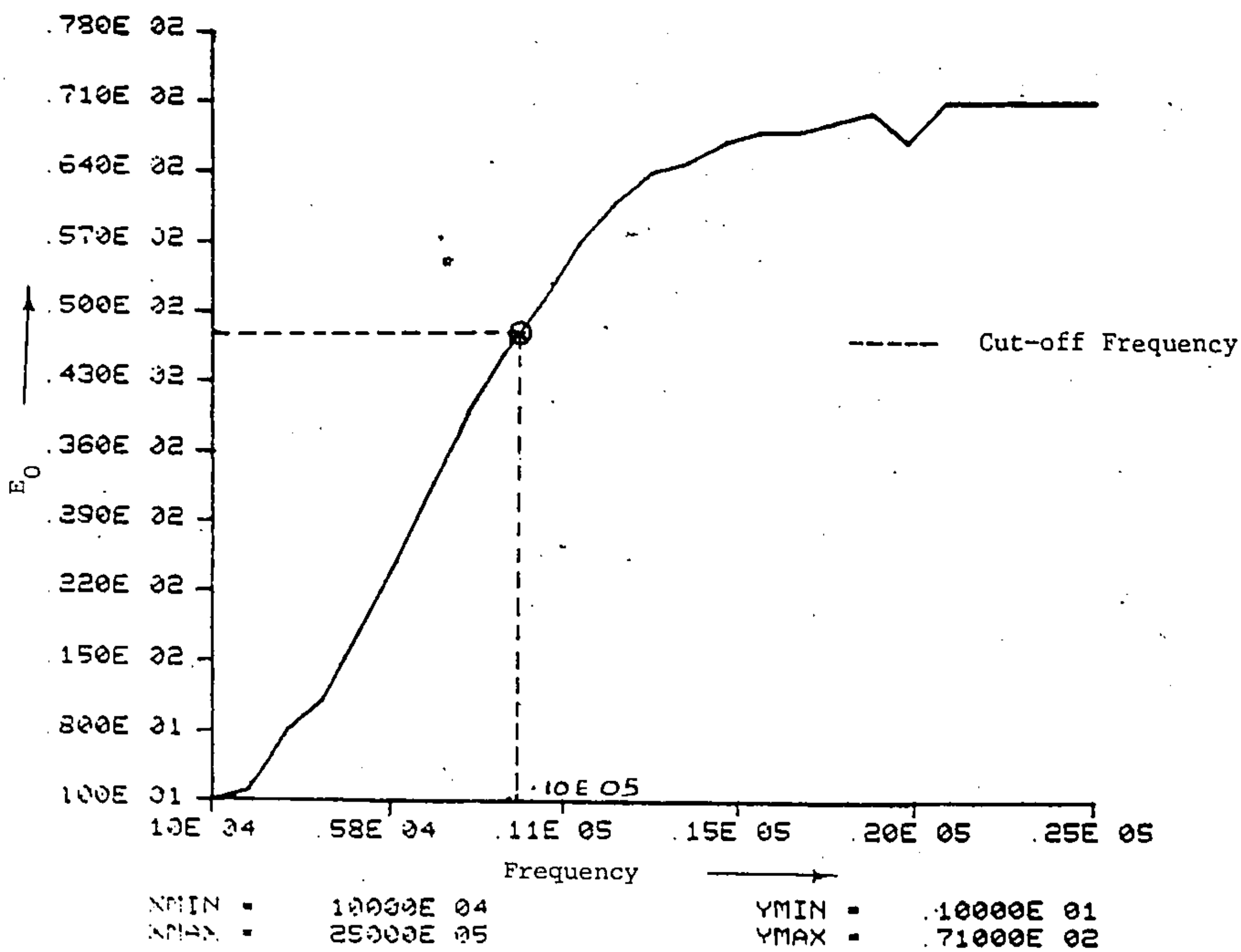


Fig. (3.14) Actual Frequency Response of a Second Order High Pass. Filter Realized Using RNS.

3.5 COMPARISON BETWEEN THE COMBINATORIAL RECURSIVE DIGITAL FILTER REALIZED USING THE FIXED POINT ARITHMETIC AND THE RESIDUE NUMBER ARITHMETIC

In this section a comparison is made between the Peled-Liu approach realized using the weightage number system (fixed point arithmetic) and the residue number system under the following titles.

i) Dynamic Range

ii) Quantization Error

iii) Scaling

iv) Hardware Complexity

3.5.1 Dynamic Range

The number of bits required to represent the coefficients and the input signal, depends upon the coefficient sensitiveness, type of filter structure chosen for realization, and the tolerance specifications in the design of the filter.

With the conventional Peled-Liu approach realized using fixed point arithmetic, it is possible to achieve an internal arithmetic of 2B bits for a B bit realization. One of the major drawbacks of this realization is that the operating speed of the filter decreases on increasing the dynamic range of the filter. [Refer to Table (3.2)]

The residue coded combinatorial recursive digital filter can be realized with shorter residue operands in parallel subfilters. This phenomena is particularly useful in realizing a coefficient sensitive filter; having a large dynamic range, at very high speed.

Tables 3.2 and 3.3 show the actual relationship between the dynamic range and the bandwidth of the filter realized using the weightage number system (fixed point arithmetic) and the RNS respectively. The tables are based on the following data.

i) The access time to generate one output sample is equal to B x 100 nano-secs. in the conventional Peled-Liu approach [17]* and about

ii) 255 nano-secs. in the residue coded combinatorial filter (refer page 45)

TABLE 3.2

Relationship Between The Dynamic Range And Bandwidth Of Digital Filter Realized Using The Fixed Point Arithmetic.

No. Of Bits	Internal Arithmetic -Bits	Word Rate -MHz.	Actual Bandwidth -KHz.
8	16	1.25	625
12	24	0.833	416.70
16	32	0.625	312.50

*The parallel form of the Peled-Liu approach is not considered for comparison here due to its complexity in realizing it.

TABLE 3.3

Relationship Between The Dynamic Range And Bandwidth Of Residue Coded Combinatorial Digital Filter Along With The Equivalent Dynamic Range Actually Achieved Corresponding To That Of Table 3.2

Moduli Used	No. of Bits	Equivalent No. Of Bits (Tab. 3.2)	Internal Arithmetic -Bits	Scale Factor	Word Rate -MHz	Actual B.W. -MHz
16,15,13,11	4	7.91	15.07	13 x 11	4.0	2.0
32,31,29,27	5	9.954	19.567	29 x 27	4.0	2.0
64,63,61,59	6	12.970	23.790	61 x 59	4.0	2.0
64,63,61,59	6	17.900	23.790	59	4.0	2.0

3.5.2 Quantization Error

The quantization error in the conventional Peled-Liu approach is due to the rounding of the partial fractions and the output of the filter to B bits (where B is the dynamic range of the filter.). It is governed by the inequality

$$- \frac{2^{-B}}{2} \leq y(n)_r - y(n) \leq \frac{2^{-B}}{2} \qquad (3.29)$$

where $y(n)_r$ is the rounded output of the digital filter

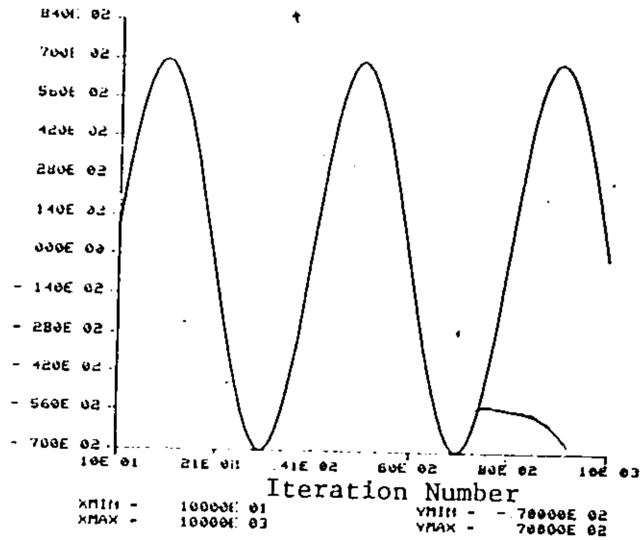
The principal source of error in a residue coded recursive digital filter is due to the quantization of the coefficients and the input signal to integers. However if the coefficients of the digital filter transfer function are suitably scaled to integers, the resulting performance characteristics of the filter realized using the RNS resembles those obtained using the weightage number system. To justify this statement a low-pass recursive digital filter (eqn.(2.10)) was realized in the weightage number system (fixed point arithmetic) and the RNS. The scaled dynamic range is equal to 8 bits in both the cases.

Fig.(3.15) shows simulated results of a sine waveform, (f = 50 Hz.) propogating through the low-pass filter. Fig.(3.15(b)) shows the simulated waveform, with full computer precision (32 bits). Fig.(3.15(c)) shows the simulated waveforms, for a 2's complement arithmetic (P-L approach) with 8 bits wordlength while fig.(3.15(d))[5] shows a residue simulation, of 7.91 bits of scaled dynamic range. It is apparent from these waveforms that both the residue and the 2's complement computations, resembles each other and track the exact response very closely.

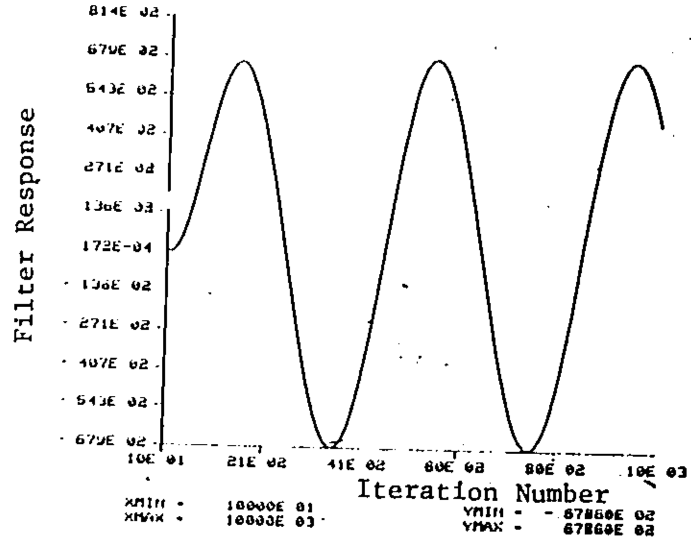
Similarly fig.(3.16) shows simulated results of a sine waveform, propogating through the low-pass filter at f = 250 Hz.

---

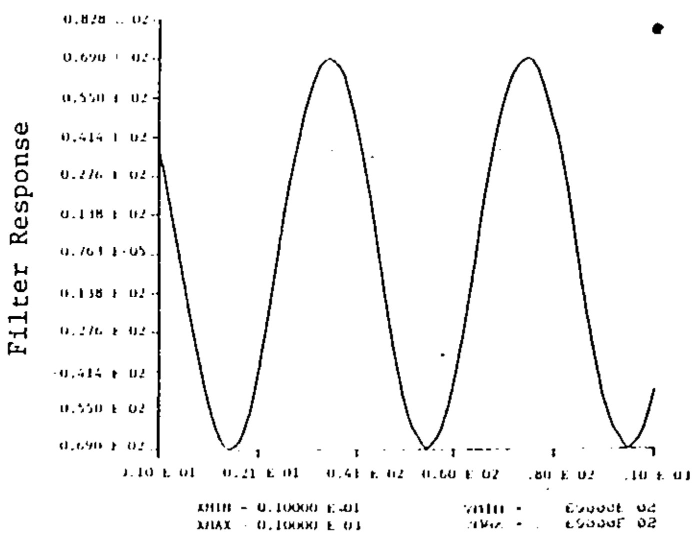
[5] An examination of Figs.(3.15-(c)), Figs.(3.15-(d)), Figs.(3.16-(c)) and (3.16-(d)) reveals an apparent phase-shift between the input and the output of the 2's complement and the RNS filters. This apparent phase-shift was caused by the exclusion of the transient portions of the output in the illustration.



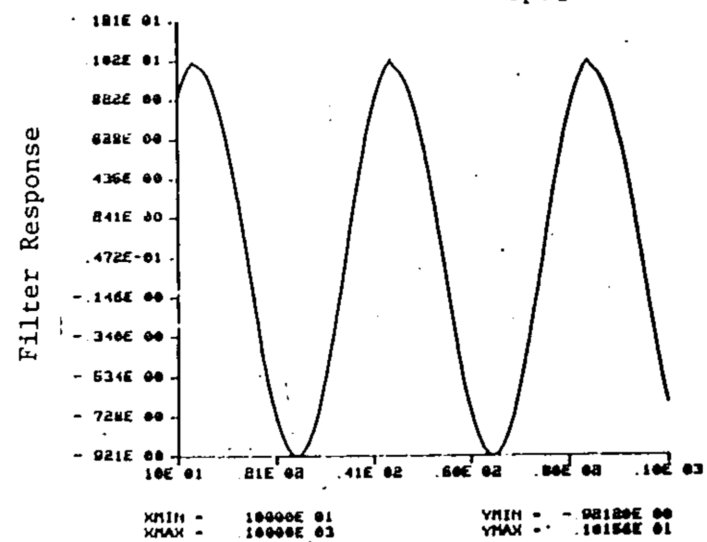
(a) Input to the Filter



(b) Full Precision Output

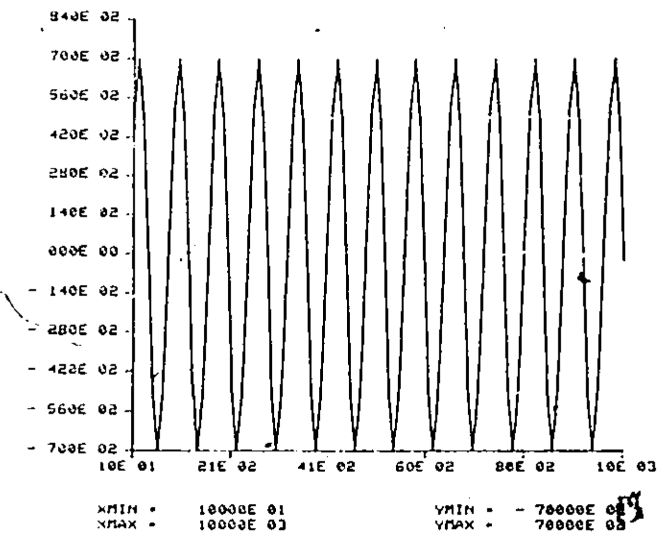


(c) Residue Coding (Rounding)

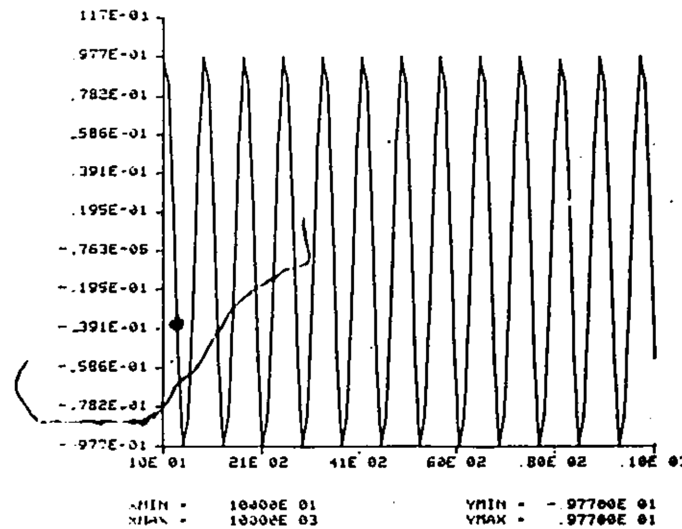


(d) 2's Complement Coding (Rounding)

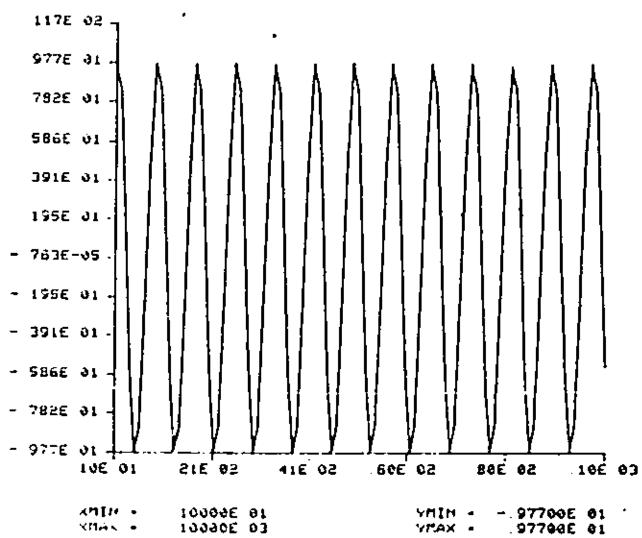
Fig. (3.15) Sine Wave Input tc a Low-Pass Filter at F = 50 Hz



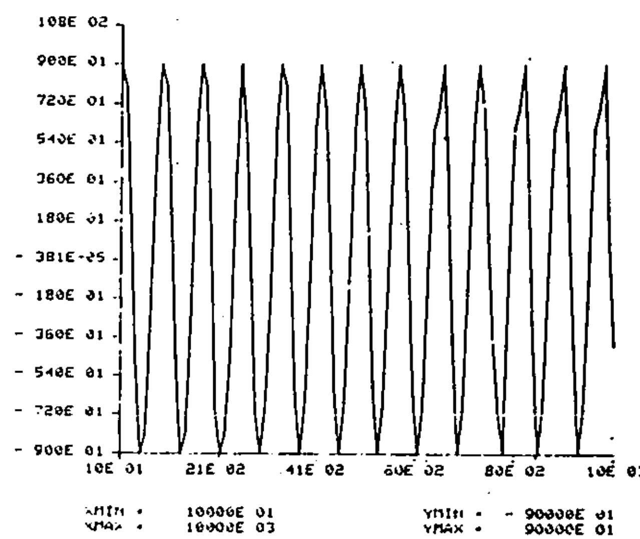
(a) Input to the Filter



(b) Full Precision Output



(c) Residue Coding (Rounding)



(d) 2's Complement Coding (Rounding)

Fig. (3.16) Sine Wave Input to a Low-Pass Filter at F = 400 Hz

3.5.3 Scaling

The scaling operation is necessary in the implementation of a recursive digital filter to prevent the overflow of the dynamic range of the system at the branching nodes.

In the conventional Peled-Liu approach realized using the weightage number system (fixed point arithmetic), division by the fixed radix (= 2 or multiples of 2 for the binary system) can be accomplished by shifting the bits (magnitude portion only) to the right. This shift can also be wired in the existing hardware, without the need of any extra hardware for the purpose of scaling operation. This is illustrated as follows :-

[0] 1 1 0 0	=	0.75	Division by 2 is accomplished by
[0] 0 1 1 0	=	0.375	shifting 1 bit left.

Scaling is required in residue coded recursive filters because stable recursive difference equations have fractional or mixed fractional coefficients that cannot be represented in an integer number system These fractional coefficients must be converted to integers by multiplying them with an appropriate scale-factor S and rounding them to the nearest integer value as illustrated in eqn.(3.30) for a second order recursive filter section.

$$y_s(n) = \frac{1}{S} \{ [S \cdot a_0]_r \cdot x(n) + [S \cdot a_1]_r \cdot x(n-1)$$

$$+ [S \cdot a_2]_r \cdot x(n-2) - [S \cdot b_1]_r \cdot y(n-1)$$

$$- [S \cdot b_2]_r \cdot y(n-2) \} \quad (3.30)$$

The notation $[Sa_0]_r$ indicates that a_0 is multiplied by the scale factor S, and the result is rounded to the nearest integer in preparation for representing (3.30) in an RNS. After the expression within the brackets {·} of (3.30) has been computed the result must be divided by S and quantized to an integer value so that the output y(n) is available for the next iteration. This is not possible in the RNS directly. A limited form of division in which the divisor is one or product of some moduli chosen for realization is however possible through the use of the Mixed-Radix Conversion Technique [18] or through the use of the Chinese Remainder Theorem [23]. The scaling process here, thus requires an extra amount of hardware to implement this algorithms.

3.5.4 Hardware Complexity

In the conventional Peled-Liu approach, due to the bit slicing technique, it is possible to simplify the structure, and achieve higher operating speeds as compared to other similar realizations. However some of the basic drawbacks of the structure are

i) It is a combination of synchronously and asynchronously controlled devices requiring a fairly complex control.

ii) The necessity of Sign-Extension logic to restore the sign of the accumulated output at each stage of iteration.

In contrast the residue coded combinatorial recursive digital filter can be synchronized easily as

i) The basic arithmetic is a modular arithmetic enabling pipelined structures.

ii) The entire filter is a ECM implementation.

iii) The sign of the number is included within the number system, so no extra logic is necessary to restore the sign.

Chapter IV

TWO DIMENSIONAL RECURSIVE DIGITAL FILTERS

In the preceding chapters a recursive digital filter was realized using the conventional weightage number system and the residue number system for computing the output. The results of this simulation reveals that

i) The throughput rate of the filter realized using the RNS is considerably higher than the weightage number system.

ii) If the coefficients of the digital filter transfer function are properly scaled to integers, the resulting performance characteristics of the filters realized using the RNS resembles those obtained using the weightage number system.

Since a video spectrum ranges from 0-4 MHz., RNS is a better choice of number system for video bandwidth filtering.

4.1 IMPLEMENTATION OF A 2-D QUARTER PLANE DIGITAL FILTER

Consider a general 3 x 3 two dimensional recursive digital filter characterized by the following difference equation.

$$y(k, \ell) = \sum_{n_1=0}^{2} \sum_{n_2=0}^{2} a_{n_1 n_2} \cdot x(k - n_1, \ell - n_2)$$

$$- \sum_{m_1=0}^{2} \sum_{m_2=0}^{2} b_{m_1 m_2} \cdot y(k - m_1, \ell - m_2) \quad (4.1)$$

$(m_1, m_2 \neq 0$ simultaneously)

where { $a_{n_1 n_2}$ } and { $b_{m_1 m_2}$ } are the sets of constant coefficients that charaterize the particular filter. and k, ℓ defines the position of the sample in the array filtered.

The coefficients of the filter $a_{n_1 n_2}$ and $b_{m_1 m_2}$ of a stable realizable difference equation in general, have fractional or mixed fractional coefficients that cannot be represented in an integer number system. These coefficients are converted to integers by multiplying them with an appropriate scale-factor S and rounding the result to the nearest integer as shown below.

$$y_s(k, \ell) = \frac{1}{s} \left[\sum_{n_1=0}^{2} \sum_{n_2=0}^{2} (s \cdot a_{n_1 n_2})_r \cdot x(k - n_1, \ell - n_2) - \sum_{m_1=0}^{2} \sum_{m_2=0}^{2} (s \cdot b_{m_1 m_2})_r \cdot y(k - m_1, \ell - m_2) \right]$$

$(m_1, m_2 \neq 0$ simultaneously)

(4.2)

After computing the expression within the brackets of eqn. (4.2), the result must be scaled by S and quantized to an integer value so as to obtain $y_s(k, \ell)$ for use in the next iteration. This is particularly important in the implementation of recursive digital filters to prevent growth of the output due to greater units gain.

The realization scheme of a stable 2-D RNS recursive digital filter should be such that the quantization error due to rounding of coefficients to integers does not produce an unstable filter. At the same time care should be exercized to see that the unscaled output of the difference equation (4.2) at any instant does not produce any overflow of the number system.

$$\sum_{n_1=0}^{2} \sum_{n_2=0}^{2} (s \cdot a_{n_1 n_2})_r \cdot x(k-n_1, \ell-n_2) -$$

$$\sum_{m_1=0}^{2} \sum_{m_2=0}^{2} (s \cdot b_{m_1 m_2})_r \cdot y(k-m_1, \ell-m_2) \leq \prod_{i=1}^{L} m_i \qquad (4.3)$$

The hardware implementation of the difference equation (4.2) is based on a set of five moduli {16,15,13,11,7}[6] with no common factor. The selection of moduli is such that each modulus can be represanted within four bits, facilitating the use of 256 x 4 prom's needed for all arithmetic table look-up operations. The scale-factor of the number system is chosen as the product of 13 x 11 x 7 and estimate scaling technique suggested by Jullien [21] is used for obtaining the output. The moduli set thus has an unscaled dynamic range of 17.8 bits and a scaled dynamic range of 7.9 bits.

The difference equation (4.2) for each modulus is splitted into subfunctions Fi(.)'s, i = 1,6. The resulting difference equation is expressed as below.

$$\left|y(k, \ell)\right|_{m_i} = \left|F_{2N}(\cdot) + F_{4n}(\cdot) + F_{6N}(\cdot) + F_{2D}(\cdot) + F_{4D}(\cdot) + F_{6D}\right|_{m_i} \qquad (4.4)$$

where

$$F_{2N}(\cdot) = \left|\left|a_{00} \cdot x(k, \ell)\right|_{m_i} + \left|a_{01} \cdot x(k, \ell-1)\right|_{m_i} + \left|a_{02} \cdot x(k, \ell-2)\right|_{m_i}\right|_{m_i}$$

$$= \left|F_{1N}(\cdot) + \left|a_{02} \cdot x(k, \ell-2)\right|_{m_i}\right|_{m_i} \qquad (4.5)$$

---

[6] A similar result can also be obtained using the moduli set {32,31,29,27}. The moduli set has a dynamic range of 20 bits and each number in the set can be represented by 5 bits.

$$F_{4N}(\cdot) = \Big|\, |a_{10} \cdot x(k-1,\ \ell)|_{m_i} + |a_{11} \cdot x(k-1,\ \ell-1)|_{m_i}$$

$$+ |a_{12} \cdot x(k-1,\ \ell-2)|_{m_i}\Big|_{m_i}$$

$$= F_{3N}(\cdot) + |a_{12} \cdot x(k-1,\ \ell-2)|_{m_i}\Big|_{m_i} \tag{4.6}$$

$$F_{6N}(\cdot) = \Big|\, |a_{20} \cdot x(k-2, \ell)|_{m_i} + |a_{21} \cdot x(k-2,\ \ell-1)|_{m_i}$$

$$+ |a_{22} \cdot x(k-2,\ \ell-2)|_{m_i}\Big|_{m_i}$$

$$= F_{5N}(\cdot) + |a_{22} \cdot x(k-2,\ \ell-2)|_{m_i} \tag{4.7}$$

Similarly the functions $F_{2D}(\cdot)$, ..., $F_{6D}(\cdot)$ can be derived by replacing numerator coefficients by dencminator and x's with y's.

The resulting structure of the filter for moduli m_i is shown in fig.(4.1).The figure does not include the interface of the filter. The basic block diagram of the interface section section is shown in fig.(4.2).It consists of a T.V. scanner, a buffer and a residue encoder.The details of the interface section is given in [15]. The implementation scheme used in the realization is a row-wise recursion of the input array requiring storage of rows of input and output viz $x(k-1,\ell)$, $x(k-2,\ell)$, $y(k-1,\ell)$ and $y(k-2,\ell)$.The size of such storage register depends upon the size of the input array to be filtered and the number of such registers is equal to twice the size of the filter [13].

Referring to fig.(4.1) the registers R1N,--,R6D represents four bit parallel access shift registers 74LS195 used either as buffers or as column delay elements. The access time of shift registers is equal to 30 nanoseconds. The row delays are provided by First In First Out (serial memory) shift registers 67401 organized as 64 words of 4 bits. The shift in shift out rate (SI/SO) of the FIFO is 100 nanoseconds and is guaranteed only if the FIFO is not full. This can be avoided by using a FIFO of larger size than the size of the input array to be filtered. The subfunctions Fi(.)'s are stored in schottky TTL Proms 93417 requiring an active

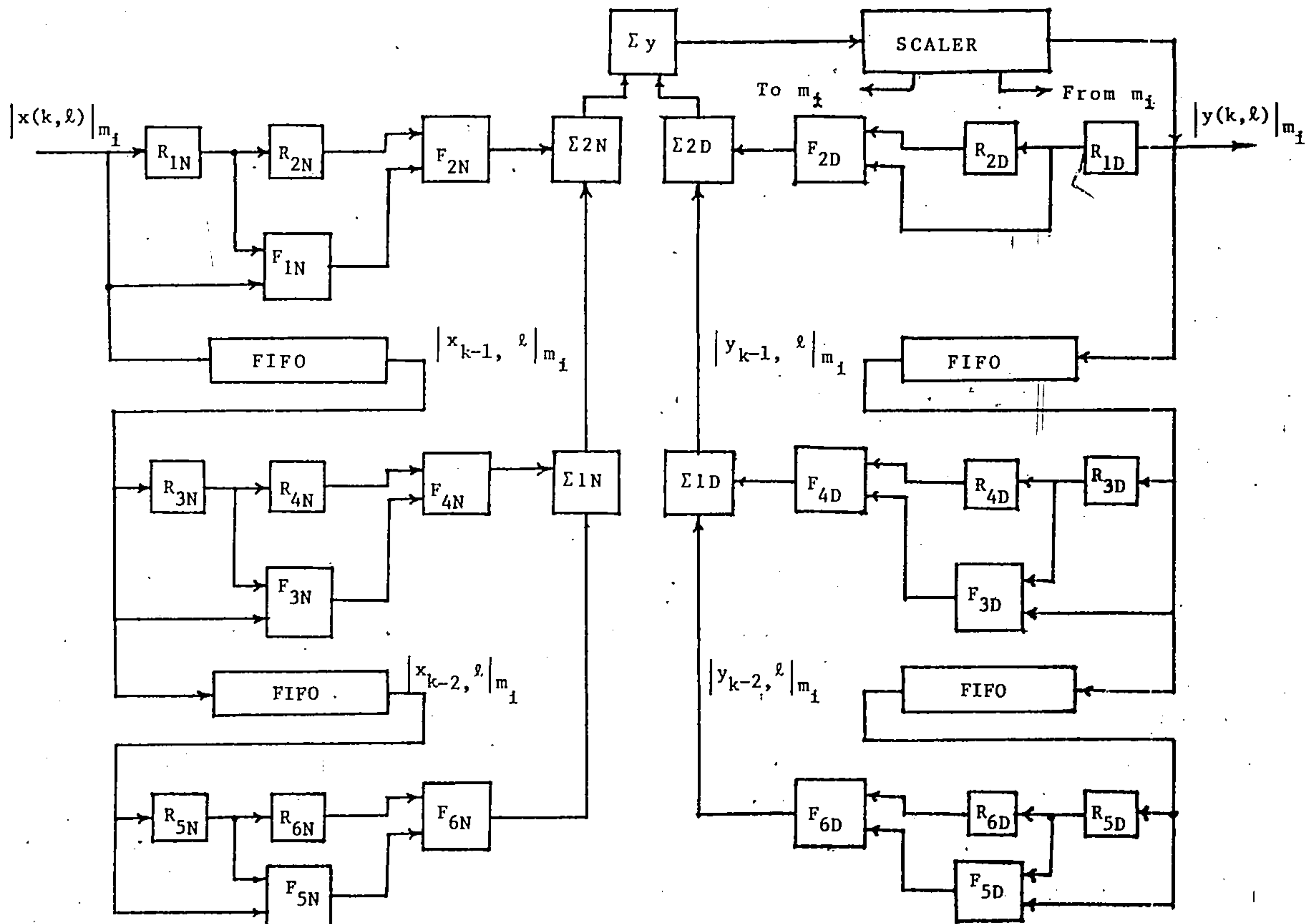


FIG. (4.1) An i^{th} Section of a 3 x 3 2-D Residue Coded Recursive Digital Filter.

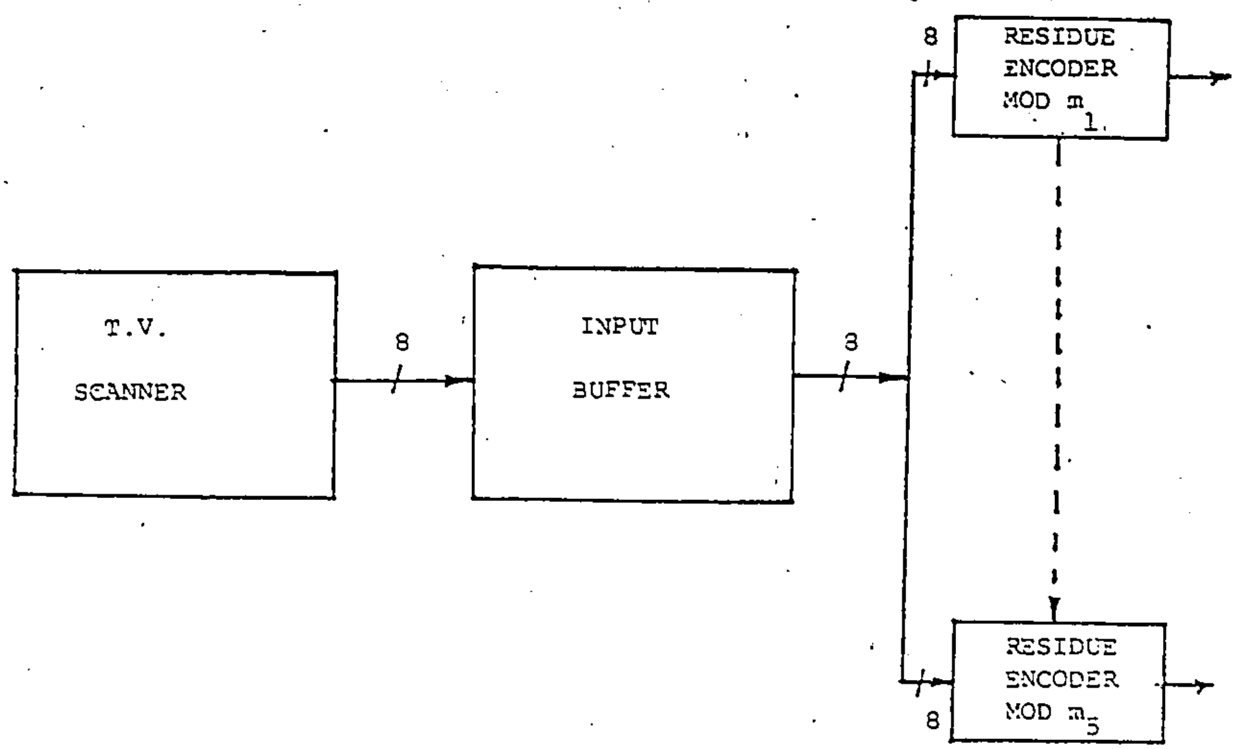


Fig. (4.2) Block Diagram of an Interface Section to the Filter.

pull up resistor for each output data line. The Proms has an access time of 25 nanoseconds.

The total propogation delay[7] before an output sample is reached can be categorized as below :-

(1) Total access time (t_a) from the interface buffer to the output of the function table F2N(.) or its equivalent.

$= t_a (R1N) + t_{max} \{t_a(R2N) \text{ or } t_a(F1N)\} + t_a(F2N)$

= 120 nanaseconds.

(2) Total access time to add subfunctions Fi(.)'s in three stages using modulo adder:

$= t_a(\Sigma 1) + t_a(\Sigma 2) + t_a(\Sigma 3)$

= 75 nanoseconds

(3) Total access time to formulate $|y_s(k, \ell)|$ in the scaler block.

= 105 nanoseeconds.

The total propogation delay is thus the summation of (1), (2), and (3) which is equal to 305 nanoseconds enabling the filter to operate at a maximum speed of 3 MHz.

---

[7] This propogation delay is calculated using the above mentioned componants in its implementation and is a rough estimate of the actual delay.

4.2 FILTER OPERATIONS AND TESTING

The proposed two dimensional residue coded recursive digital filter was simulated on the computer using a 3 x 3 order circularly symmetric low-pass filter transfer function with the following specifications.

$$Y(w_1, w_2) = 1.0,\ 1.0,\ 0.8,\ 0.44,\ 0.14,\ 0.03,\ 0.002,\ 0.001,\ 0.001,\ 0.001,\ 0.001$$

for $\sqrt{w_1^2 + w_2^2}\ /\pi = 0.0,\ 0.1, \text{----------}, 1.0$ respectively.

Figs. (4.3) and (4.4) show an impulse response and a delayed unit step response of the filter transfer function. A study of these figures reveal that the stability of the filter transfer function is preserved.

Fig. 4.5(a) show a 2-D circularly symmetric sinusoidal input within the passband region of the filter ($f/f_N = 0.05 \times \pi$). Figs. 4.5(b) and 4.5(c) depicts the full precision and the actual output of the filter realized using RNS.

Similarly Figs. (4.6) and (4.7) shows the input/output of the filter when a 2-D circularly symmetric sinusoidal signal is fed within the transition ($f/f_N = 0.2 \times \pi$) and stopband regions ($f/f_N = 0.4 \times \pi$) of the filter transfer function.

It is found from looking at these waveforms that the full precision and the actual output of the filter resembles each other. This suggests that a properly scaled 2-D recursive digital filter realized using RNS preserves the spectral characteristics of the filter transfer function.

Finally figs.(4.8) and (4.9) shows the ideal and the actual frequency response of the filter realized using RNS.

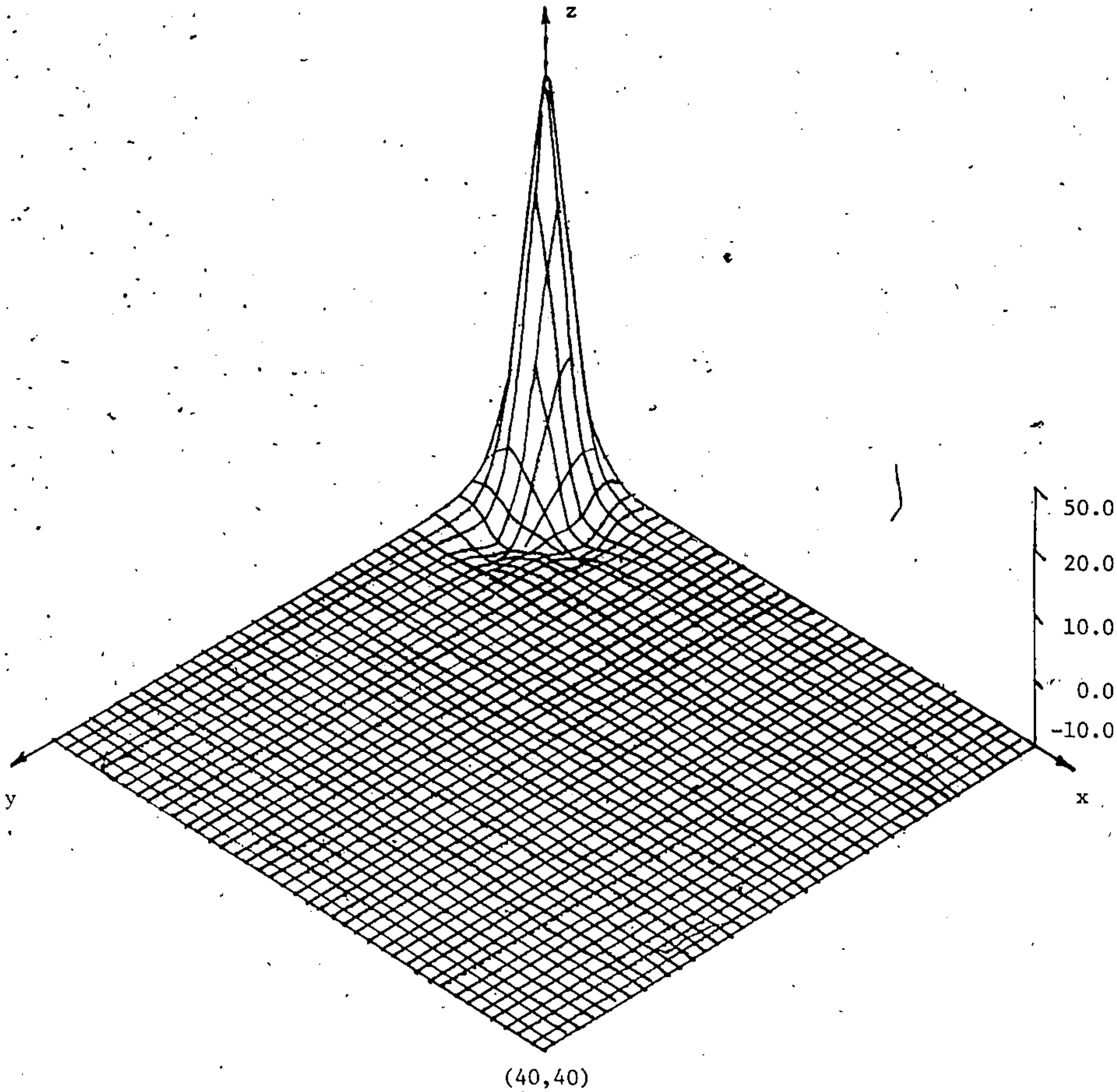


Fig. (4.3) The Impulse Response of a Low-Pass Filter.

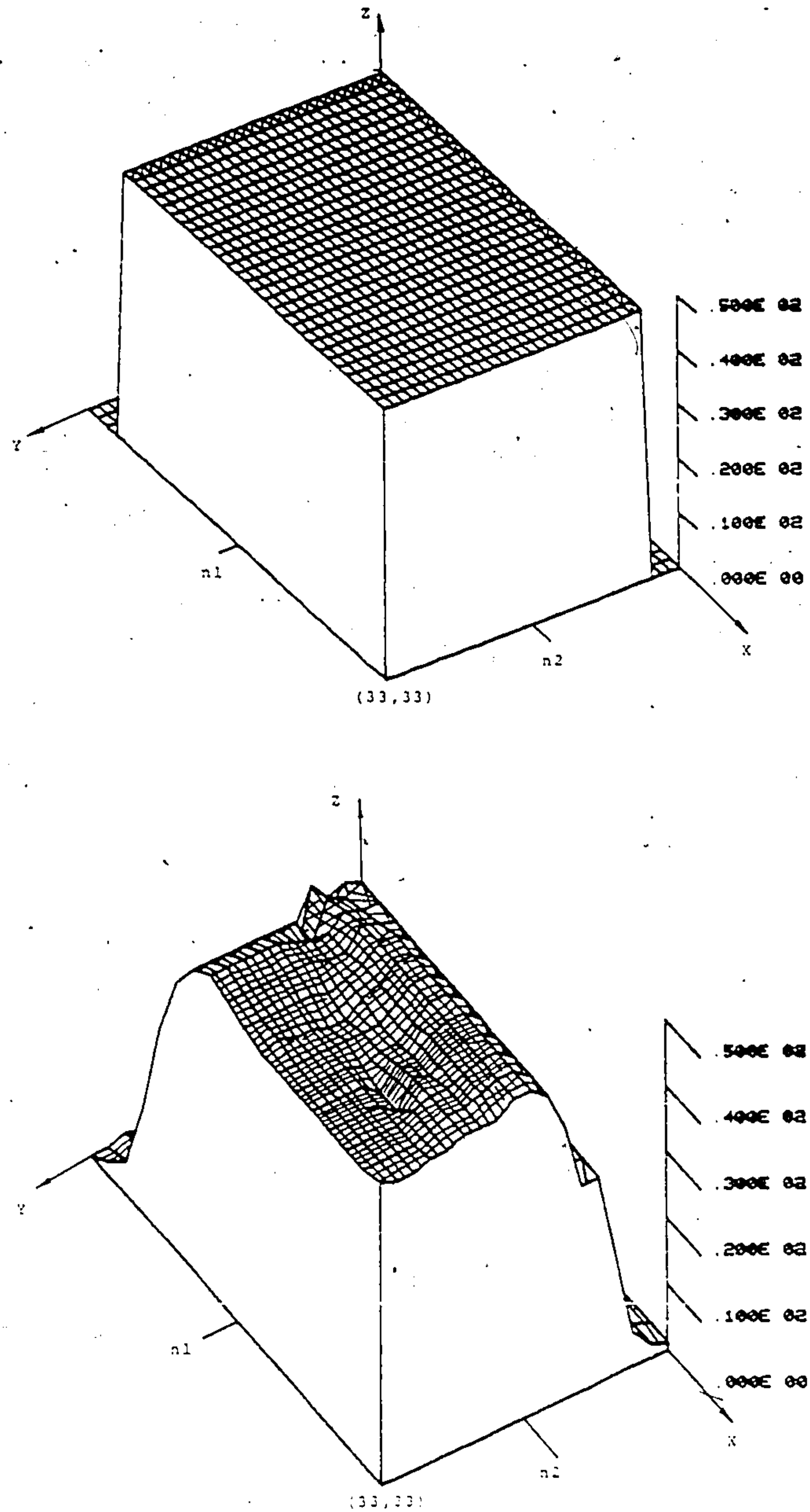


Fig. (4.4) Unit-Step Response of a Low-Pass Filter.

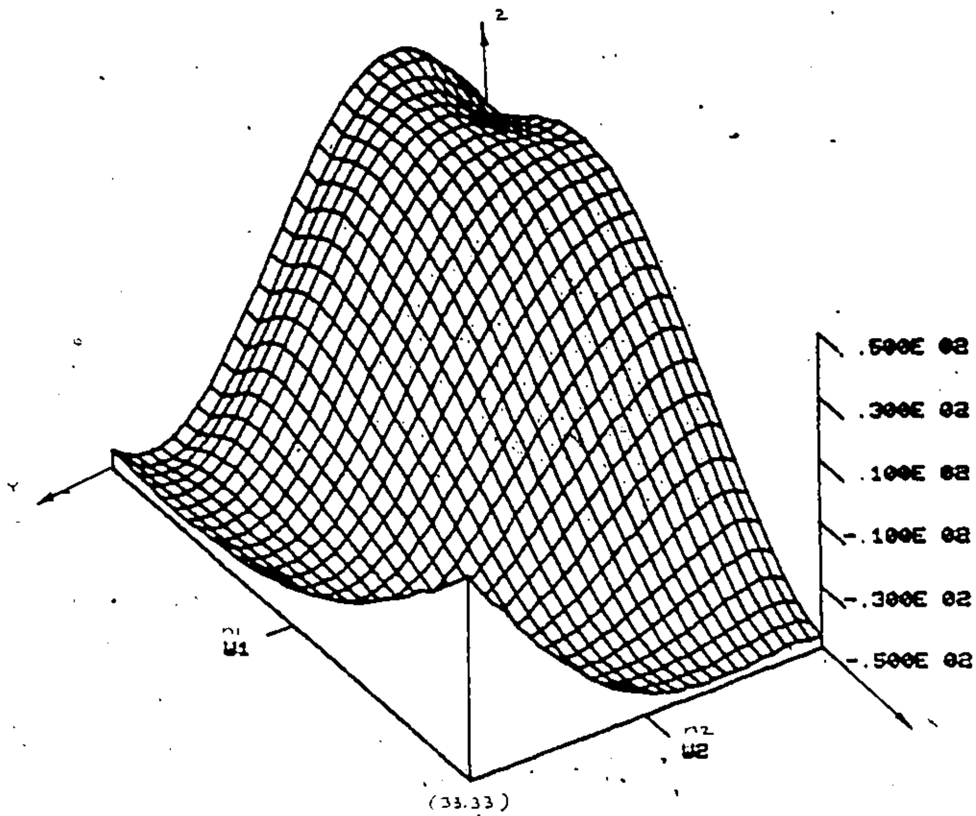


Fig. (4.5) (a) A 2-D Circularly Symmetric Sinusoidal Input to a Low-Pass Filter. ($f/f_N = 0.05\cdot\pi$)

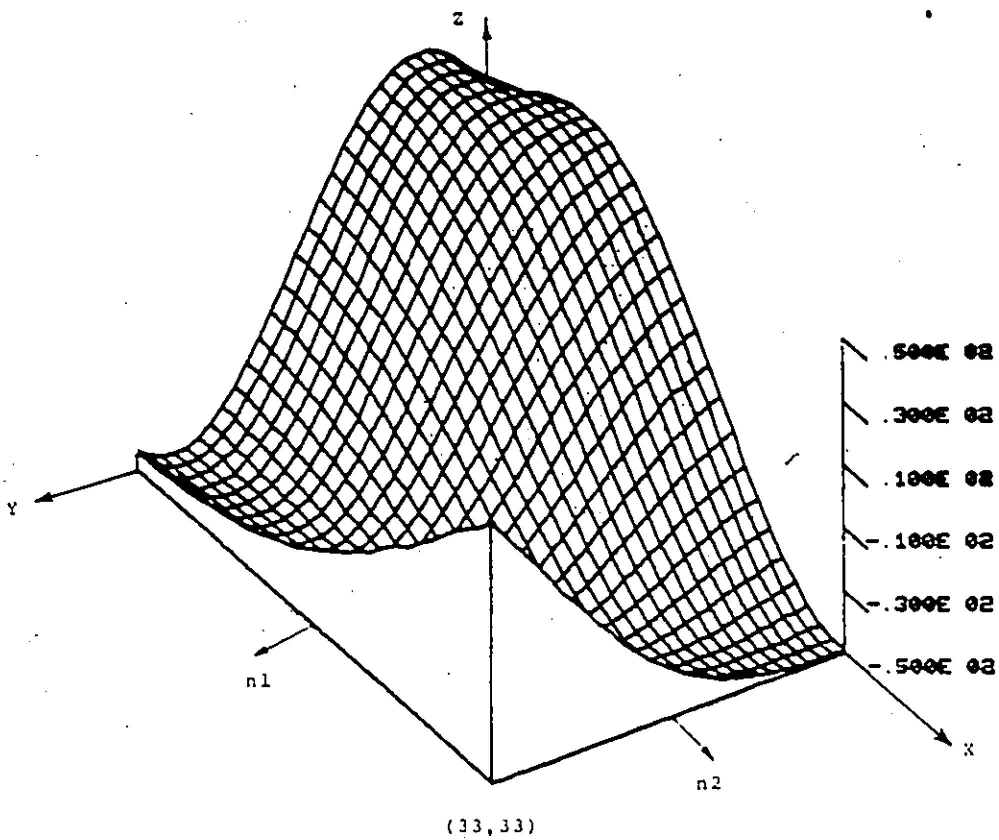


Fig. (4.5) (b) Full Precision Output.

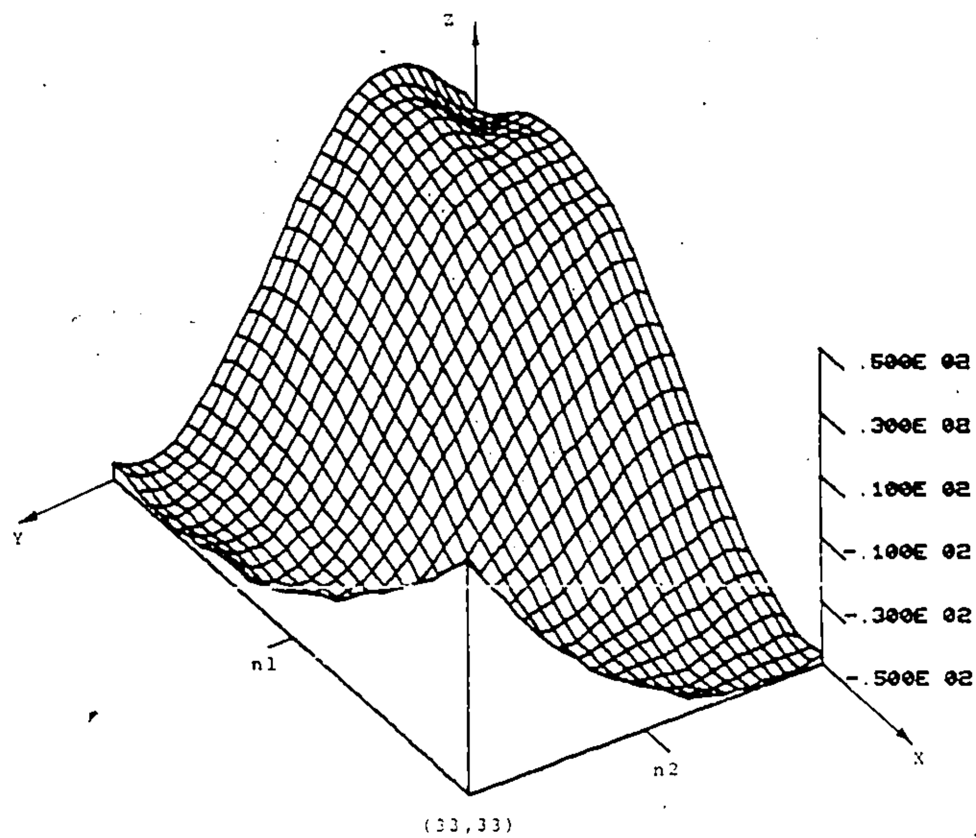


Fig. (4.5) RNS Output.

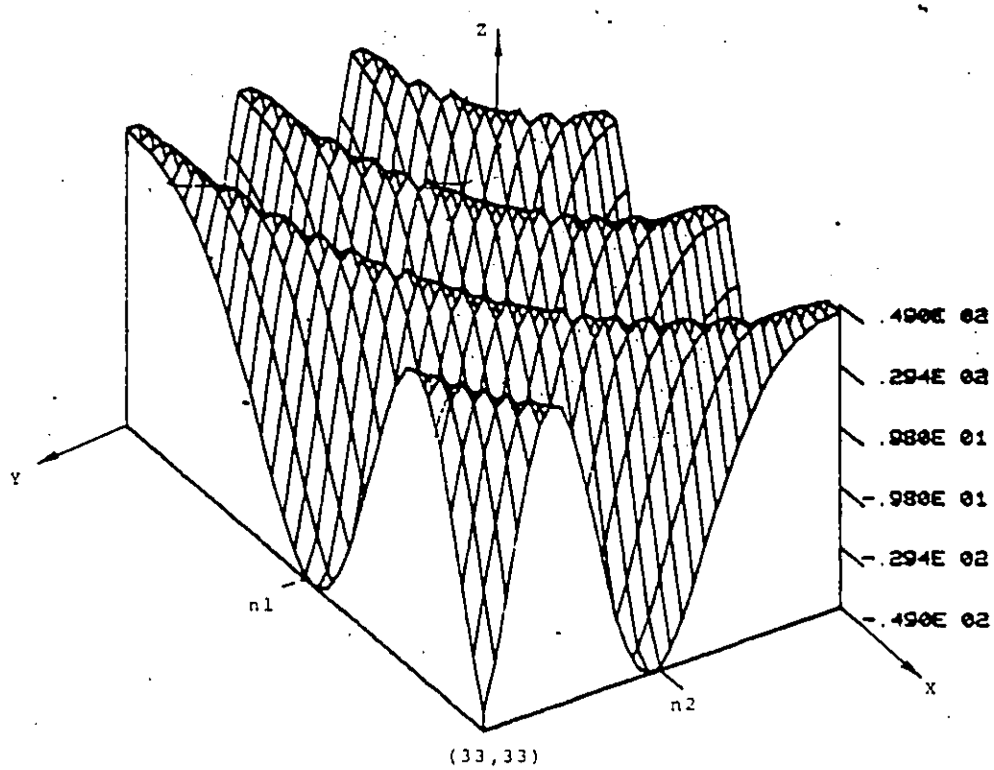


Fig. (4.6) (a) A 2-D Circularly Symmetric Sinusoidal Input to a Low-Pass Filter. $(f/f_N = 0.1 \cdot \pi)$

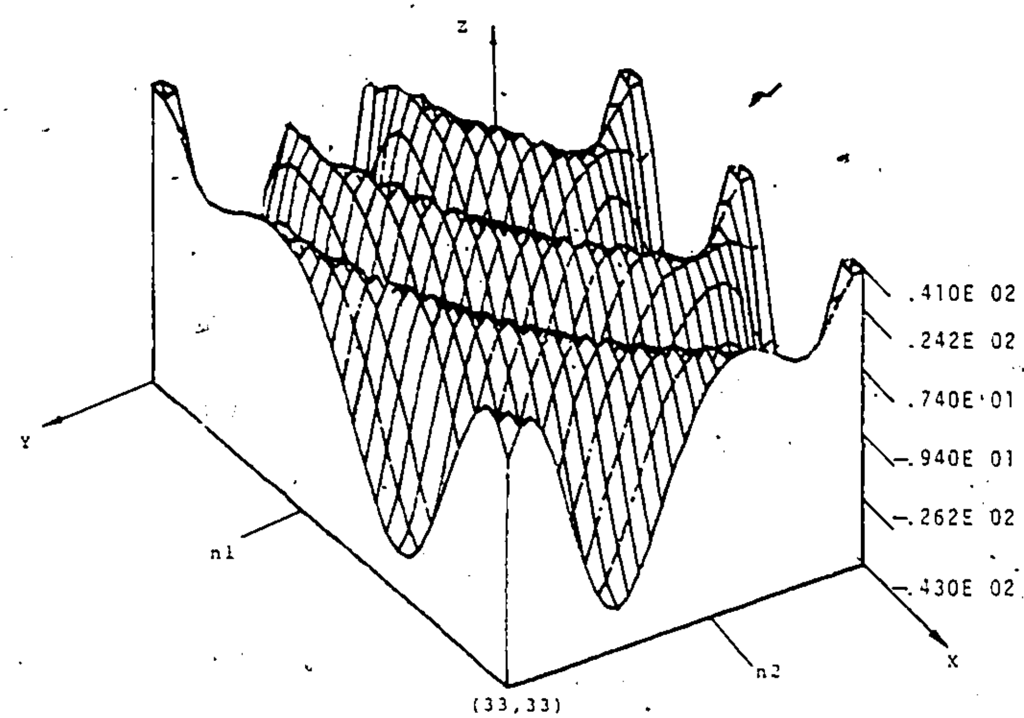


Fig. (4.6) (b) Full Precision Output.

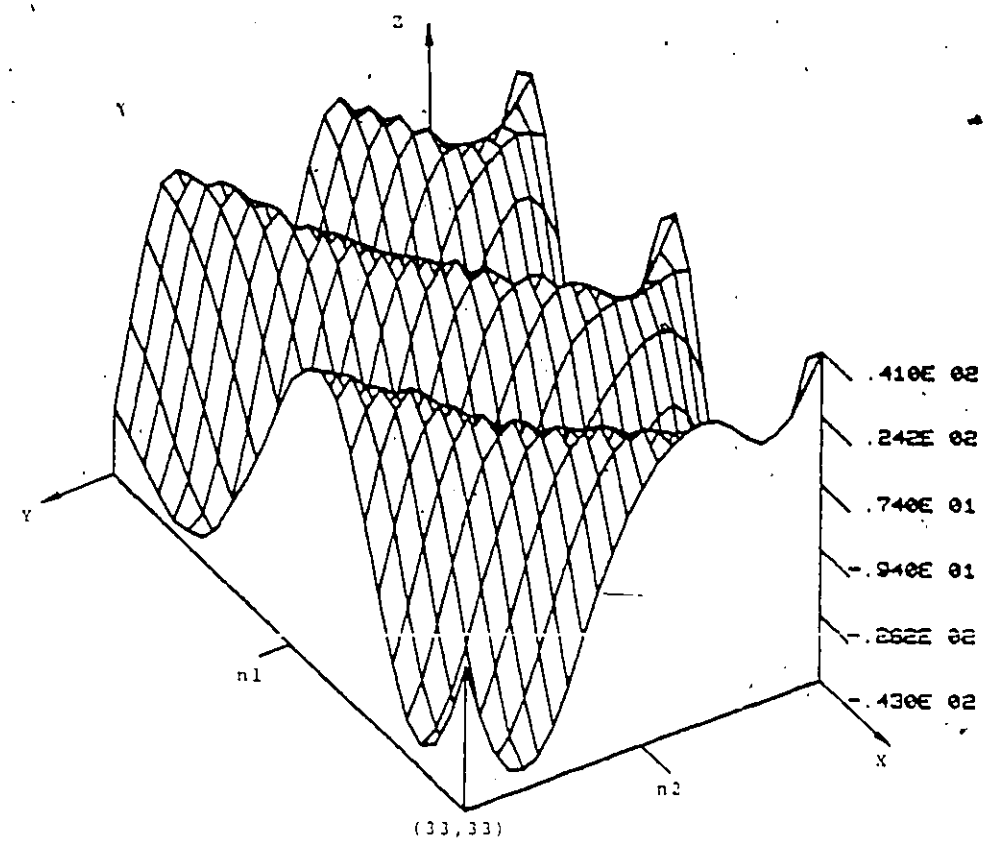


Fig. (4.6) (c) RNS Output.

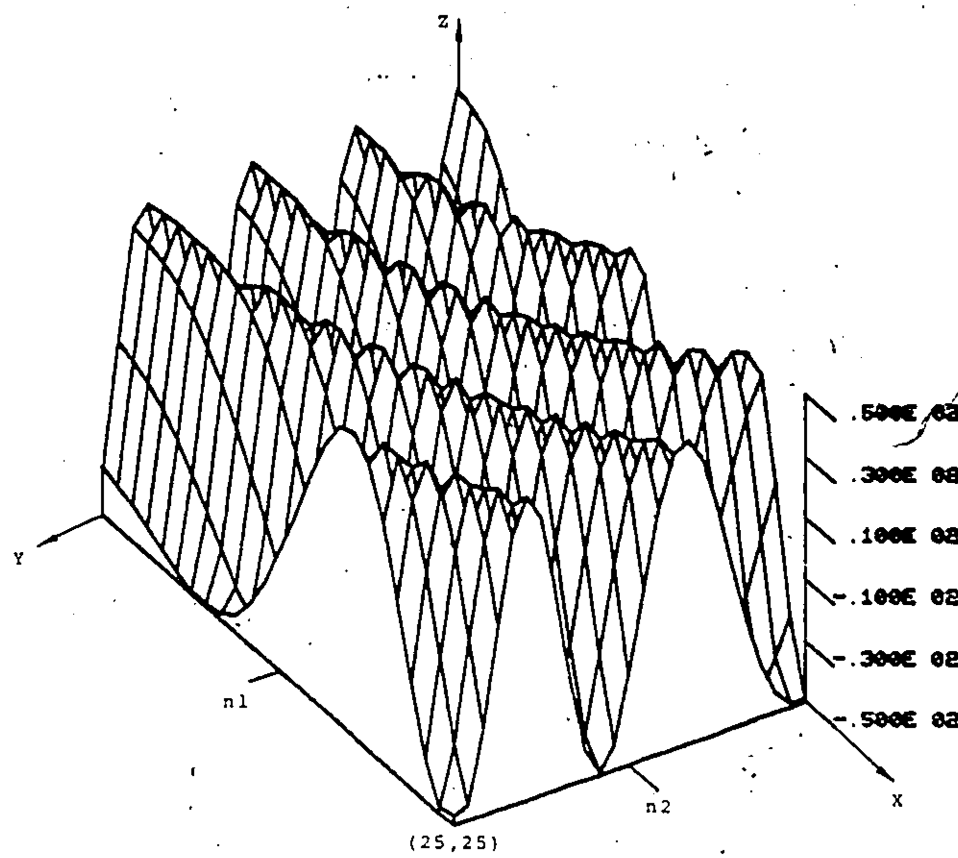


Fig. (4.7) (a) A 2-D Circularly Symmetric Sinusoidal Input to a Low-Pass Filter. $(f/f_N = 0.4 \cdot \pi)$

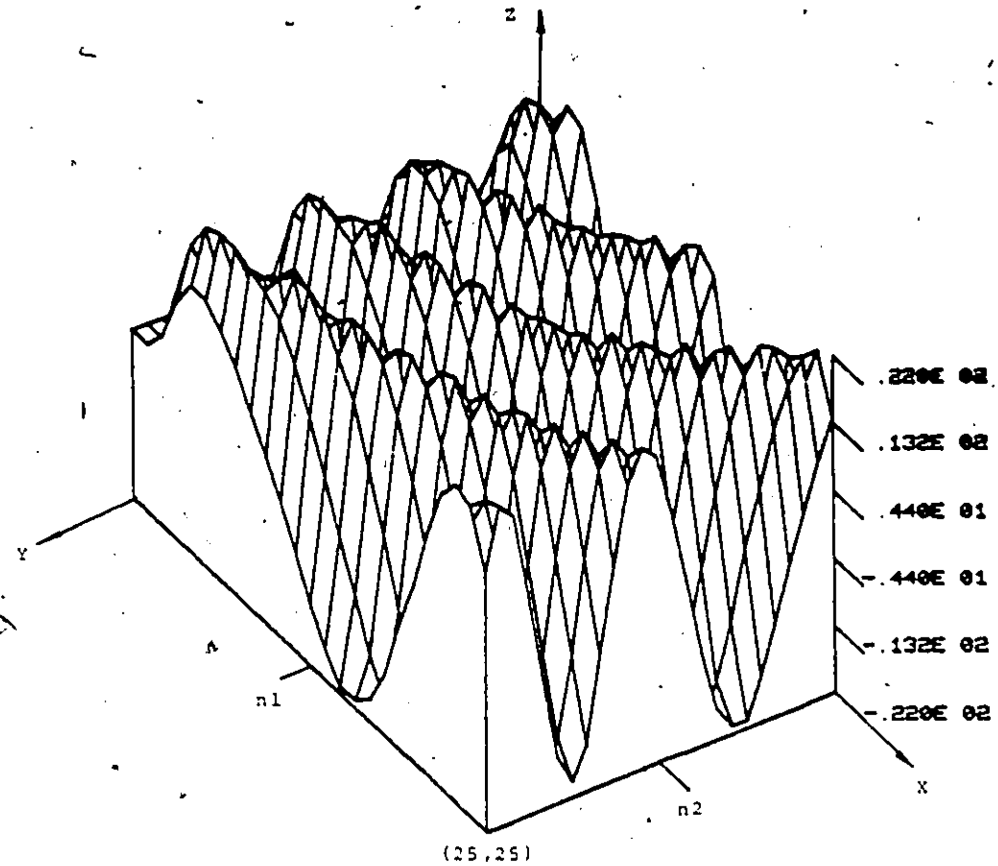


Fig. (4.7) (b) Full Precision Output.

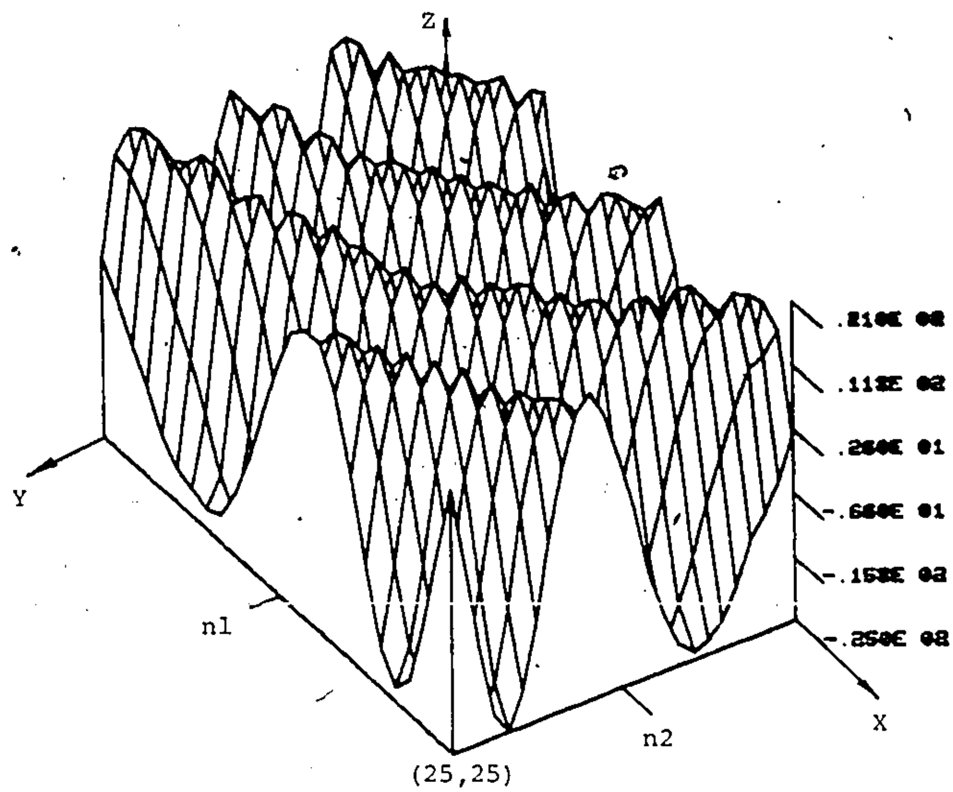


Fig. (4.7) (c) RNS Output.

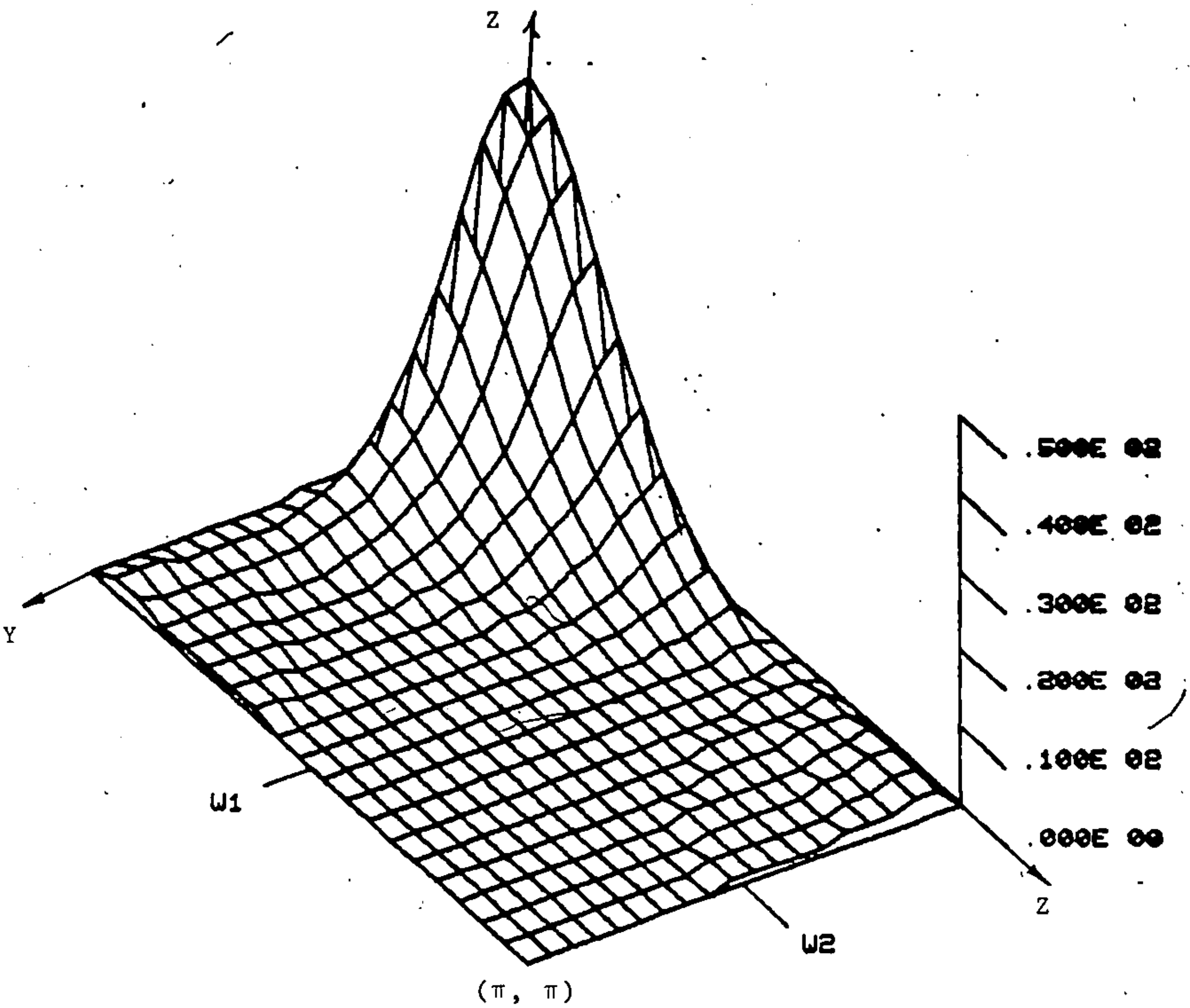


Fig. (4.8) Ideal Frequency Response of a 2-D Quarter Plane 3 x 3 Low Pass Filter

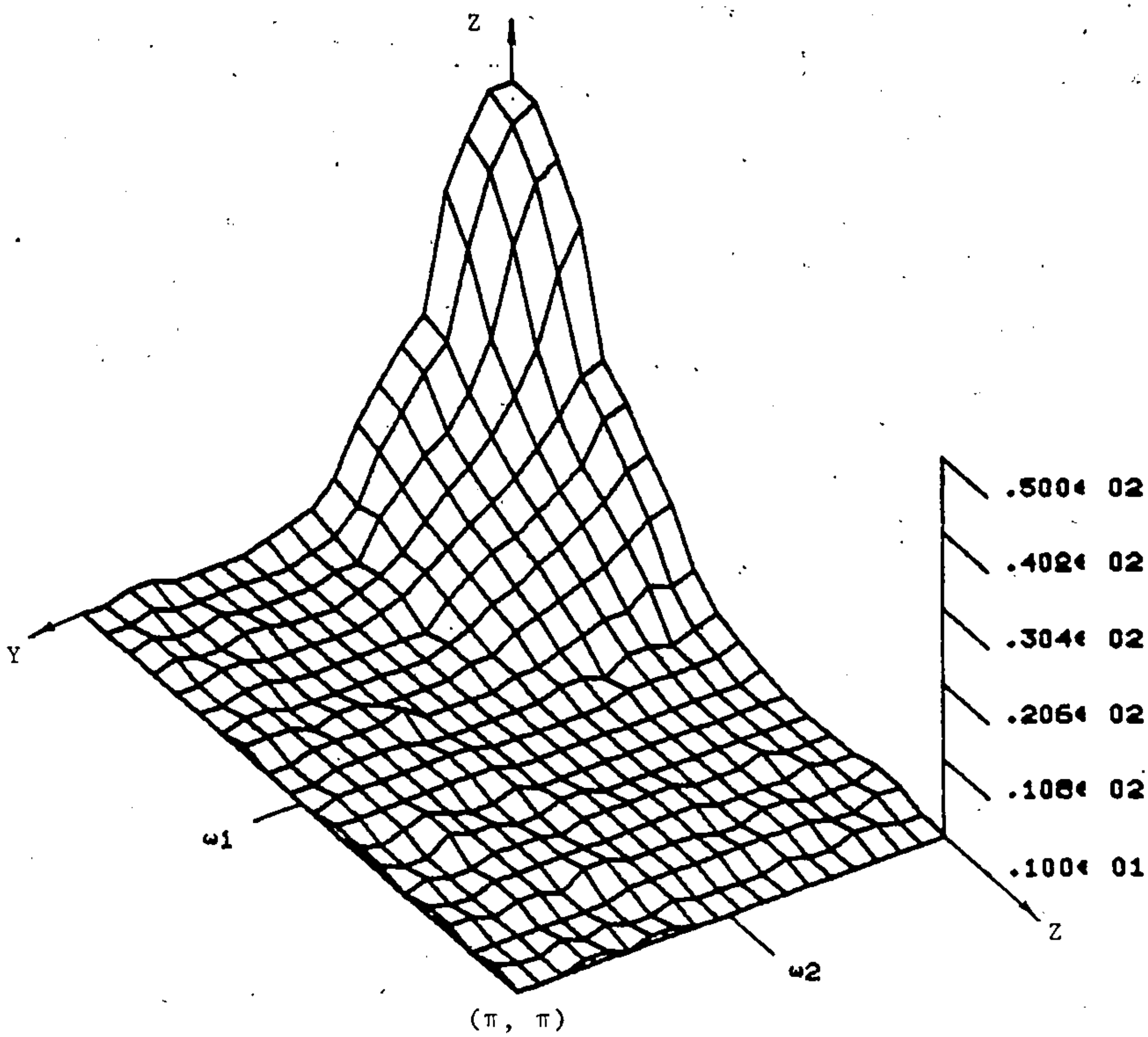


Fig. (4.9) Actual Frequency Response of a 2-D Quarter Plane 3 x 3 Low Pass Filter Realized Using RNS.

Chapter V

SUMMARY AND CONCLUSION

5.0.1 Summary

The throughput rate of the filter often becomes the focal-point in the implementation of a fast digital processor or a dedicated piece of hardware for realizing digital filters.

This thesis addresses the realization of an efficient structure, for implementing one and two dimensional recursive digital filters, capable of operating at very high speeds to handle image processing problems at video rates.

In addition to the throughput rate, the structure chosen for realization must

i) Maintain the spectral characteristics of the digital filter.

ii) Preserve the stability of the filter.

iii) Have a reasonable cost/speed factor.

Based on an exhaustive literature survey it was found that the Peled-Liu bit slicing approach in the weightage number system satisfies much of these constraints while still maintaining a throughput rate of 0.5-1 MHz. depending upon the number of bits chosen for realization. Thus the technique is best suited for a class of general filters, requiring 8-10 bits dynamic range.

However certain signal processing applications require filters having a dynamic range of 12-20 bits and a throughput rate of 1-3 MHz. In such cases it would be advantageous to implement the filters using the residue number system (RNS). To prove the concept the coefficients of the same recursive digital filter (used to realize the Peled-Liu structure) were coded in residues and the residue coded combinatorial recursive digital filter proposed by Jenkins was used in the realization of digital filters.

On a similar basis a two dimensional recursive digital filter structure was designed using ROM's, shift registers and FIFO's as the building blocks of the system. The filter structure was implemented in the RNS enabling a parallel pipelined modular architecture. (the characteristic feature of the number system)

The filter structure was tested out for the case of a two dimensional quarter plane low-pass transfer function and the performance of the filter was found satisfactory i.e. the spectral characteristics of the filter were preserved.

Finally the impulse response and the unit step response of the filter showed that the stability of the filter was also preserved.

5.0.2 Conclusion

The conventional Peled-Liu approach (weightage number system) is shown to offer the best trade-off between cost and speed for implementing a class of recursive digital filters whose coefficients can be quantized within 8-10 bits and whose actual bandwidth requirements are limited upto 500 KHz.

The RNS implementation is particularly advantageous for realizing a coefficient sensitive recursive digital filter which can operate at a speed of 4-5 MHz. This desirable feature is mainly due to the modular nature of the RNS providing the ability to add subtract or multiply in one step regardless of the length of the number involved and without recourse to intermediate carry digits or internal delays.

The proposed design of a two dimensional residue coded recursive digital filter structure is also shown to preserve the spectral characteristics and the stability of the given filter transfer function.

Appendix A

LISTING OF COMPUTER PROGRAMS

```
C      ****************************************************
C
C      THE PROGRAM CALCULATES THE PARTIAL FRACTIONS TO
C      FORMULATES THE LOOKUP TABLE REQUIRED FOR THE PELED LIU
C      FIXED FORM REALIZATION.
C
C      THE DATA INPUTS REQUIRED FOR THE PURPOSE ARE
C
C      THE COEFFECIENTS OF THE FILTER :-A0,A1,A2,B1,B2
C      THE SCALE FACTOR :- S
C
C      ****************************************************
C
       DIMENSION DF(32)
       INTEGER AOUT(32,8),ADS(32)
       LOGICAL*1 B(8)
       COMMON/B4/K
       DATA B/.FALSE.,.FALSE.,.FALSE.,.FALSE.,.FALSE.,
      $.FALSE.,.FALSE.,.TRUE./
       READ,A0,A1,A2,B1,B2
       READ,S
       DO 40 K=1,32
       ADS(K)=K
       GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
      $18,19,20,21,22,23,24,25,26,27,28,29,30,31,32),K
   1   DF(1)=0.
       GO TO 34
   2   DF(2)=B2/S
       GO TO 34
   3   DF(3)=B1/S
       GO TO 34
   4   DF(4)=(B1+B2)/S
       GO TO 34
   5   DF(5)=A2/S
       GO TO 34
   6   DF(6)=(A2+B2)/S
       GO TO 34
   7   DF(7)=(A2+B1)/S
       GO TO 34
   8   DF(8)=(A2+B1+B2)/S
       GO TO 34
   9   DF(9)=A1/S
       GO TO 34
  10   DF(10)=(A1+B2)/S
       GO TO 34
  11   DF(11)=(A1+B1)/S
       GO TO 34
  12   DF(12)=(A1+B1+B2)/S
       GO TO 34
  13   DF(13)=(A1+A2)/S
       GO TO 34
  14   DF(14)=(A1+A2+B2)/S
       GO TO 34
  15   DF(15)=(A1+A2+B1)/S
```

```
      GO TO 34
16    DF(16)=(A1+A2+B1+B2)/S
      GO TO 34
17    DF(17)=A0/S
      GO TO 34
18    DF(18)=(A0+B2)/S
      GO TO 34
19    DF(19)=(A0+B1)/S
      GO TO 34
20    DF(20)=(A0+B1+B2)/S
      GO TO 34
21    DF(21)=(A0+A2)/S
      GO TO 34
22    DF(22)=(A0+A2+B2)/S
      GO TO 34
23    DF(23)=(A0+A2+B1)/S
      GO TO 34
24    DF(24)=(A0+A2+B1+B2)/S
      GO TO 34
25    DF(25)=(A0+A1)/S
      GO TO 34
26    DF(26)=(A0+A1+B2)/S
      GO TO 34
27    DF(27)=(A0+A1+B1)/S
      GO TO 34
28    DF(28)=(A0+A1+B1+B2)/S
      GO TO 34
29    DF(29)=(A0+A1+A2)/S
      GO TO 34
30    DF(30)=(A0+A1+A2+B2)/S
      GO TO 34
31    DF(31)=(A0+A1+A2+B1)/S
      GO TO 34
32    DF(32)=(A0+A1+A2+B1+B2)/S
34    CALL DECTCP(DF,AOUT,B)
40    CONTINUE
      PRINT 41
41    FORMAT('1',22X,'CONTENTS OF ROM',/,23X,15('-'),//)
      PRINT 42
42    FORMAT(7X,'LOC.',8X,'CONTENTS',9X,'LOC.',8X,'CONTENTS',//)
      DO 43 I=1,16
      M=I+16
43    PRINT 44,ADS(I),(AOUT(I,J),J=1,8),ADS(M),(AOUT(M,J),J=1,8)
44    FORMAT(8X,I2,5X,8(I2),6X,I2,5X,8(I2),/)
      STOP
      END
```

```
C       ************************************
C       ***                              ***
C       ***  SUBROUTINE DECTCP(DF,AOUT,B) ***
C       ***                              ***
C       ************************************
C
C       THE SUBROUTINE CONVERTS THE PARTIAL FRACTION
C       IN DECIMAL SYSTEM TO BINARY SYSTEM USING
C       TWO'S COMPLEMENT NOTATION FOR REPRESANTING
C       NEGATIVE NUMBERS.
C
        SUBROUTINE DECTCP(DF,AOUT,B)
        DIMENSION DF(32)
        INTEGER AOUT(32,8)
        COMMON/B4/K
        LOGICAL*1 A(8),B(8)
        DO 16 I=1,8
  16    A(I)=.FALSE.
        H=ABS(DF(K))
        P=H
        BINV=0.5
        DO 1 I=2,8
        IF(H .GE. BINV) GO TO 2
        GO TO 3
   2    A(I)= .TRUE.
        H=H-BINV
   3    BINV=BINV/2.
   1    CONTINUE
        IF(H .GE. 0.00390625) GO TO 20
        GO TO 21
  20    IF(P .GE. 0.9921875) GO TO 21
        CALL ADDN(A,B)
  21    IF(DF(K) .LT. 0.) GO TO 4
        GO TO 6
   4    DO 5 I=1,8
   5    A(I)= .NOT. A(I)
        CALL ADDN(A,B)
   6    DO 15 I=1,8
        AOUT(K,I)=0
  15    IF(A(I)) AOUT(K,I)=1
        RETURN
        END
```

```
C      ****************************
C      ***                      ***
C      *** SUBROUTINE ADDN(A,B) ***
C      ***                      ***
C      ****************************
C
C      THE SUBROUTINE ROUNDS OFF THE DATA TO 8 BITS
C      BY ADDING 2 * * (-9) BIT TO THE RESULT AND THEN
C      TRUNCATING THE DATA TO 8 BITS.
C
       SUBROUTINE ADDN(A,B)
       LOGICAL*1 A(8), B(8), G(8), P(8),S(8),C(9)
      $          ,P1,Q1,R1,EXOR,TEXOR
       EXOR(P1,Q1)=P1 .AND. .NOT. Q1 .OR. Q1 .AND. .NOT. P1
       TEXOR(P1,Q1,R1)=R1 .AND. .NOT. EXOR(P1,Q1) .OR.
      $                EXOR(P1,Q1) .AND. .NOT. R1
       C(9)= .FALSE.
       DO 7 I=1,8
       N=9-I
       G(N)=A(N) .AND. B(N)
       P(N)=EXOR(A(N),B(N))
       S(N) = TEXOR(A(N),B(N),C(N+1))
       C(N)=G(N) .OR. C(N+1) .AND. P(N)
  7    CONTINUE
 10    DO 11 I=1,8
 11    A(I)=S(I)
       RETURN
       END
```

```
C      **************************************************
C
C      THIS IS A PROGRAM OF HARDWARE SIMULATION OF A GIVEN
C      1-D SECOND ORDER DIGITAL FILTER T/F FN. USING
C      BIT SLICING TECHNIQUE SUGGESTED BY PELED AND LIU.
C
C      FOLLOWING TERMINOLOGY IS USED IN THE PROGRAM:-
C
C      N  = NO. OF BITS USED TO REPRESANT THE INPUT SIGNAL
C      RO = THE SAMPLE NO. OF THE ARRAY BEING PROCESSED
C      JS = NO. OF SAMPLES PER ARRAY
C      BJ = ADDRESS OF THE LOOK UP TABLE(ROM)
C      LT = LOOK UP TABLES STORING THE PARTIAL PRODUCTS
C           OF THE INPUT ARRAY.
C
C      ***************************************************
C
C      MAIN PROGRAM
C      ------------
C
       REAL LT(32),I1(100)
       DIMENSION XF(100),T1(100),YF(100),CF(100)
       DIMENSION XMAG(100),F(100)
       INTEGER BJ,RO,SX(100,8),SY(100,8)
       INTEGER X0(8),X1(8),X2(8),Y1(8),Y2(8)
       INTEGER XOUT(8),R1(5)
       DOUBLE PRECISION SR, R3, Y(100)
       COMMON/B1/C1F
       LOGICAL*1 B(8)
       DATA B/.FALSE.,.FALSE.,.FALSE.,.FALSE.,.FALSE.,
      $.FALSE.,.FALSE.,.TRUE./
       DO 39 I=1,32
 39    READ 38,LT(I)
 38    FORMAT(E14.7)
       PRINT 87
 87    FORMAT(52X,'CONTENTS OF LOOK UP TABLES',//)
       PRINT 89,(LT(I),I=1,32)
 89    FORMAT(4(2X,E14.7),/)
       PYE=22./7.
       N=8
       N2=2*N
       N1=N-1
       EM=1.
       T=0.00001
       R3=0.0039625D0
       DO 67 MPTS=1,100
       QMAX=0.
       F(MPTS)=500*MPTS
       F1=F(MPTS)
       RO=3
       JS=100
       W1=2. *PYE*F1
       DO 2 K=1,100
       I1(K)=FLOAT(K)
```

```
      T1(K)=K*T
      XF(K)=EM*SIN(W1*T1(K))
      P1=XF(K)
C
C     THE DECTCP SUBROUTINE REPRESENTS A NEGATIVE FRACTION
C     IN TWO'S COMPLEMENT FORM.
C
      CALL DECTCP(P1,XOUT,B)
      DO 1 I=1,8
    1 SX(K,I)=XOUT(I)
      CF(K)=C1F
    2 CONTINUE
      DO 85 I=1,8
      SY(1,I)=0
   85 SY(2,I)=0
      YP(1)=0.
      YP(2)=0.
      Y(1)=0.
      Y(2)=0.
   17 IT=1
      DO 55 I=1,N
   55 XO(I)=SX(RO,I)
      IF(RO .EQ. 3) GO TO 4
      GO TO 6
    4 DO 60 I=1,N
      X1(I)=SX(2,I)
   60 X2(I)=SX(1,I)
      DO 50 I=1,N
      Y1(I)=0
   50 Y2(I)=0
    6 R1(1)=XO(N)
      R1(2)=X1(N)
      R1(3)=X2(N)
      R1(4)=Y1(N)
      R1(5)=Y2(N)
C
C     THE LSB OF EACH REGISTER TOGETHER FORM A BINARY
C     VECTOR DEFINING A PARTICULAR POSITION OF ROM.
C
      BJ=R1(1)*16 + R1(2)*8 + R1(3)*4 + R1(4)*2 + R1(5)*1
      DO 7 I=1,N1
      XO(N+1-I)=XO(N-I)
      X1(N+1-I)=X1(N-I)
      X2(N+1-I)=X2(N-I)
      Y1(N+1-I)=Y1(N-I)
    7 Y2(N+1-I)=Y2(N-I)
      XO(1)=0
      X1(1)=R1(1)
      X2(1)=R1(2)
      Y1(1)=0
      Y2(1)=R1(4)
    9 R2=LT(BJ+1)
      IF(IT .EQ.1) GO TO 10
      GO TO 11
```

```
 10     SR=0.D0
 11     IF(IT .EQ. 8) GO TO 26
        SR=SR+R2
        GO TO 27
 26     SR=SR-R2
 27     IF(IT .LE. 7) GO TO 19
        GO TO 20
 19     SR=SR/2.
 20     IT=IT+1
        IF(IT .GT. N) GO TO 14
        GO TO 6
C
C       AS THE COEFFECIENTS OF ROM ARE SCALED DOWN BY
C       A FACTOR OF TWO. THE OUTPUT OF THE FILTER OBTAINED
C       IS A SCALED VERSION.
C
 14     IF(IT .EQ. N+1) SR=2. * SR + R3
        Y(RO)=SR
        YP(RO) = SNGL(SR)
        Q=YP(RO)
        IF (Q .GT. QMAX) QMAX=Q
        CALL DECTCP(Q,Y1,B)
        DO 59 J=1,8
 59     SY(RO,J)=Y1(J)
        RO=RO+1
        IF(RO .LE. JS) GO TO 17
        XMAG(MPTS)=QMAX
        CALL PLOT3(I1,CF,JS)
        CALL PLOT3(I1,YP,JS)
        PRINT 76,(CF(I),I=1,JS)
        PRINT 77
        PRINT 76,(YP(I),I=1,JS)
 67     CONTINUE
        CALL PLOT3(F,XMAG,50)
        DO 45 I=1,100
 45     PRINT 46,F(I),XMAG(I)
 46     FORMAT(25X,F7.1,10X,F7.4,/)
        PRINT 75,(F(I),I=1,100)
 75     FORMAT(10(F6.1,2X),/)
        PRINT 76,(XMAG(I),I=1,100)
 76     FORMAT(10(F7.4,1X),/)
 77     FORMAT(///)
        STOP
        END
```

```
C      **********************************
C      ***                            ***
C      *** SUBROUTINE DECTCP(X,XOUT,B) ***
C      ***                            ***
C      **********************************
C
C      THIS SUBROUTINE CONVERTS  A DECIMAL FRACTION
C      INTO BINARY, USING TWO'S COMPLEMENT'S FORM TO
C      TO REPRESENT NEGATIVE NUMBERS.
C
       SUBROUTINE DECTCP(X,XOUT,B)
       COMMON/B1/C1F
       INTEGER XOUT(8)
       LOGICAL*1 A(8),B(8)
       C1F=0.
       DO 16 I=1,8
  16   A(I)=.FALSE.
       H=ABS(X)
       P=H
       BINV=0.5
       DO1 I=2,8
       IF(H .GE. BINV) GO TO 2
       GO TO 3
   2   A(I)= .TRUE.
       C1F=C1F + BINV
       H=H-BINV
   3   BINV=BINV/2.
   1   CONTINUE
       IF(X .LT. 0.) C1F=-C1F
       IF(H .GE. 0.00390625) GO TO 20
       GO TO 21
  20   IF(P .GE. 0.9921875) GO TO 21
       CALL ADDN(A,B)
       C1F=C1F + 0.0039625
  21   IF(X .LT. 0.) GO TO 4
       GO TO 6
   4   DO 5 I=1,8
   5   A(I)= .NOT. A(I)
       CALL ADDN(A,B)
   6   DO 15 I=1,8
       XOUT(I)=0
  15   IF(A(I)) XOUT(I)=1
       RETURN
       END
```

```
C     ***************************
C     ***                     ***
C     *** SUBROUTINE ADDN(A,B) ***
C     ***                     ***
C     ***************************
C
      SUBROUTINE ADDN(A,B)
C
C     HERE ADDITION IS PERFORMED USING LOOK AHEAD CARRY
C     TECHNIQUE.
C
      LOGICAL*1 A(8), B(8), G(8), P(8),S(8),C(9),P1,Q1,R1,
     $          EXOR,TEXOR
      EXOR(P1,Q1)=P1 .AND. .NOT. Q1 .OR. Q1 .AND. .NOT. P1
      TEXOR(P1,Q1,R1)=R1 .AND. .NOT. EXOR(P1,Q1) .OR.
     $                EXOR(P1,Q1) .AND. .NOT. R1
      C(9)= .FALSE.
      DO 7 I=1,8
      N=9-I
      G(N)=A(N) .AND. B(N)
      P(N)=EXOR(A(N),B(N))
      S(N) = TEXOR(A(N),B(N),C(N+1))
      C(N)=G(N) .OR. C(N+1) .AND. P(N)
    7 CONTINUE
   10 DO 11 I=1,8
   11 A(I)=S(I)
      RETURN
      END
```

```
C      ******************************************************
C
C       THE PROGRAM SIMULATES ONE DIMENSIONAL COMBINATORIAL
C       SECOND ORDER RECURSIVE DIGITAL FILTER USING
C       RESIDUE NUMBER ARITHMETIC.
C
C       THE INPUT DATA REQUIRED FOR THE PURPOSE ARE AS
C       FOLLOWS :-
C
C       TYPE OF FILTER :- TYPFLT
C       COEFFECIENTS OF THE FILTER :- A0,A1,A2,-B1,-B2
C       NO. OF MODULI USED = NM
C       NO. OF BITS USED TO REPRESANT MODULI = N2
C       MODULI USED FOR IMPLEMENTATION :- M(I)
C       SCALE FACTOR FOR THE COEFFECIENTS :- XS
C
C       GIVEN THE  DATA, THE PROGRAM CALCULATES THE LOOK UP
C       TABLES REQUIRED FOR ROM IMPLEMENTATION.
C
C       IT ALSO COMPUTES THE FREQUENCY RESPONSE OF THE FILTER
C       TRANSFER FUNCTION USING SCALED COEFFECIENTS.
C
C      ******************************************************
C
C      MAIN PROGRAM
C      ------------
C
       INTEGER CM,C,PDR,SPDR,CMB,RSCF,FMAG,FI,XL,YL,
     $         SEM,S,XS,XI,YSN
       INTEGER A0M(4),A1M(4),A2M(4),B1M(4),B2M(4),
     $         RX(4),M(4),MIN(4),MB(4),RMIN(4)
       INTEGER FY1Y2(32,31),FY3Y4(29,27),
     $         F1M(4,32,32),F2M(4,32,32),F3M(4,32,32),
     $         XR(500),XM(4,500),X1M(4),X2M(4),
     $         YSM(4,500),Y(4),Y1M(4),Y2M(4),
     $         PH1(4),PH2(4),PH3(4),DC(4)
       REAL NF
       DIMENSION X(500),Y0(500),T1(500),COEF(5),FMG(50),FP(50)
       COMMON/B2/N2
       COMMON/B3/CM,CMB,XS
       READ 100, TYPFLT
 100   FORMAT(A9)
       READ,(COEF(I),I=1,5)
       READ,NM,N2
       READ,(M(I),I=1,NM)
       PRINT 200
 200   FORMAT(4X,A9,' FILTER',//)
       PRINT 201
 201   FORMAT(4X,'COEFFECIENTS OF THE FILTER = ',/,4X,29('-'),/)
       PRINT 202,(COEF(I),I=1,5)
 202   FORMAT(4X,'A0 = ',F10.7,/,4X,'A1 = ',F10.7,/,4X,'A2 = ',
     $F10.7,/,4X,'B1 = ',F10.7,/,4X,'B2 = ',F10.7,/)
       PRINT 203,NM,N2
 203   FORMAT('1',4X,'NO. OF MODULI = ',I2,/,5X,'NO. OF BITS',
```

```
     $/ USED TO REP. MODULI = ',I2,//)
      CM=1
      PYE=22./7.
      IFPTS=36
      FI=50
      TI=0.00005
      NF=1/(2.*TI)
      SEM=70
C
C     MODULO CAPITAL M=M1*M2*M3*M4 IS CALCULATED HERE.
C
      DO 5 I=1,NM
    5 CM=CM*M(I)
C
C     XS=SCALE FACTOR
C
      XS=CM/(M(3)*M(4))
      CMB=CM/XS
C
C     THE PROGRAM FINDS WHETHER THE VALUE OF CAPITAL MODULO M
C     IS AN ODD OR EVEN FUNCTION AND ACCORDINGLY ASSIGNS
C     DIFFERENT VALUES OF POSITIVE DYNAMIC RANGE.
C
      C=MOD(CM,2)
      IF(C.EQ.0.) GO TO 13
      PDR=(CM-1)/2
      SPDR=(CMB-1)/2
      GO TO 41
   13 PDR=CM/2*1
      SPDR=CMB/2-1
   41 PRINT 204,CM,XS,CMB,PDR,SPDR
  204 FORMAT(5X,'CAPITOL MODULLO M (TOTAL NO. RANGE) = ',I8,
     $/,5X,'SCALE FACTOR OF THE FILTER COEFFECIENTS = ',I6,
     $/,5X,'SCALED NO. RANGE = ',I5,
     $/,5X,'POSITIVE DYNAMIC RANGE = ',I7,
     $/,5X,'SCALED POSITIVE DYNAMIC RANGE = ',I6,///)
C
C     THE VALUE OF MBAR OF ALL THE MODULLI IS CALCULATED HERE
C
      DO 12 I=1,NM
   12 MB(I)=CM/M(I)
C
C     THE MULTIPLICATIVE INVERSE OF THE GIVEN MODULI
C     IS CALCULATED HERE.
C
      DO 61 I=1,NM
      MIN(I)=MOD(MB(I),M(I))
      TEMP=MIN(I)
      DO 62 J=1,100
      MULTIN=J
      MIN(I)=TEMP*MULTIN
      IF(MIN(I) .LT. M(I)) GO TO 62
      DIFF=MOD(MIN(I),M(I))
      IF(DIFF .NE. 1) GO TO 62
```

```
      MIN(I)=MULTIN
      GO TO 61
 62   CONTINUE
      PRINT 205
205   FORMAT( 4X, 'MULTIPLICATIVE INVERSE OF THE GIVEN'
     1,' MODULI DOES NOT EXIST')
      GO TO 300
 61   CONTINUE
      PRINT  206
206   FORMAT(5X,'MODULLI AND ITS MULTIPLICATIVE INVERSE',
     $/,5X,38('='),//)
      PRINT 207,(I,M(I),I,MIN(I),I=1,NM)
207   FORMAT(5X,'M(',I1,') = ',I2,5X,'MIN(',I1,') = ',I2,/)
      PRINT 208
208   FORMAT(5X,'M-HUT',/,5X,5('='),//)
      PRINT 209,(I,MB(I),I=1,NM)
209   FORMAT(5X,'MB(',I1,') = ',I6,/)
      DO 9 I=1,5
      SCF=COEF(I)*XS
      IF(SCF .LT. 0.) GO TO 7
      RSCF=SCF+0.5
      GO TO 8
  7   RSCF=SCF-0.5
  8   CALL DTRNS(M,NM,RSCF,RX)
      DO 9 K=1,NM
C
C     THE COEFFICIENTS ARE MULTIPLIED WITH THE SCALE FACTOR,
C     ROUNDED TO INTEGERS AND CODED IN RESIDUES HERE.
C
      RX(K)=RX(K)-1
      GO TO(19,20,18,21,16),I
 19   A0M(K)=RX(K)
      GO TO 9
 20   A1M(K)=RX(K)
      GO TO 9
 18   A2M(K)=RX(K)
      GO TO 9
 21   B1M(K)=RX(K)
      GO TO 9
 16   B2M(K)=RX(K)
  9   CONTINUE
      PRINT 210
210   FORMAT(///,5X,'ZEROES OF THE FILTER',/,5X,20('='),//)
      PRINT 211,(I,A0M(I),I,A1M(I),I,A2M(I),I=1,NM)
211   FORMAT(5X,'A0M(',I1,') = ',I2,5X,'A1M(',I1,') = ',I2,
     $5X,'A2M(',I1,') = ',I2,/)
      PRINT 212
212   FORMAT(///,5X,'POLES OF THE FILTER',/,5X,19('='),//)
      PRINT 213,(I,B1M(I),I,B2M(I),I=1,NM)
213   FORMAT(5X,'B1M(',I1,') = ',I2,5X,'B2M(',I1,') = ',I2,/)
      CALL FUNTBL(M,NM,F1M,A1M,A2M)
      CALL FUNTBL(M,NM,F2M,B1M,B2M)
      DO 31 I=1,NM
 31   DC(I)=1
```

```
      CALL FUNTBL(M,NM,F3M,AOM,DC)
      CALL GSCALR(NM,M,MIN,MB,FY1Y2,FY3Y4)
      DO 75 NFP=1,IFPTS
      FMAG=0.
      IF(NFP .LE. 20) GO TO 49
      IF(NFP .GT. 20) GO TO 51
49    NFP1=NFP*FI
      GO TO 57
51    NFP1=(NFP-19)*1000
57    F=NFP1/NF
      FP(NFP)=F
      DO 4 K=1,500
      T1(K)=K
      W1=PYE*F
      X(K)=FLOAT(SEM)*SIN(PYE*F*K)
82    IF(X(K).LT.0.) GO TO 10
      XR(K)=X(K)+0.5
      GO TO 11
10    XR(K)=X(K)-0.5
11    XI=XR(K)
      CALL DTRNS(M,NM,XI,RX)
      DO 22 I=1,NM
22    XM(I,K)=RX(I)
 4    CONTINUE
      IT=3
      DO 56 I=1,2
56    YO(I)=0.
      DO 55 I=1,NM
      Y1M(I)=1
      Y2M(I)=1
      X1M(I)=1
55    X2M(I)=1
17    DO 36 I=1,NM
      PH1(I)=F1M(I,X1M(I),X2M(I))
      PH2(I)=F2M(I,Y1M(I),Y2M(I))
      PH3(I)=PH1(I)+PH2(I)
      PH3(I)=MOD(PH3(I),M(I))+1
36    Y(I)=F3M(I,XM(I,IT),PH3(I))+1
      YSN=FY1Y2(Y(1),Y(2))+FY3Y4(Y(3),Y(4))
      YSN=MOD(YSN,CMB)
      IF(YSN.LE.SPDR) GO TO 69
      YSN=-(CMB-YSN)
69    YO(IT)=YSN
      IF(IT .GT. 50) GO TO 47
      GO TO 99
47    IF(FMAG .LT. YSN) FMAG=FLOAT(YSN)
99    CALL DTRNS(M,NM,YSN,RX)
      DO 27 I=1,NM
27    YSM(I,IT)=RX(I)
      DO 80 I=1,NM
```

```
      Y2M(I)=Y1M(I)
      Y1M(I)=YSM(I,IT)
      X2M(I)=X1M(I)
   80 X1M(I)=XM(I,IT)
      IT=IT+1
      IF (IT.LE.500) GO TO 17
      FMG(NFP)=FMAG
      PRINT,NFP1,FMAG
      PRINT 214,(YO(I),I=400,500)
   75 CONTINUE
      CALL PLOT3(FP,FMG,IFPTS)
      PRINT 215,(FP(I),I=1,IFPTS)
      PRINT 216
      PRINT 217,(FMG(I),I=1,IFPTS)
  214 FORMAT(10(F6.0,2X))
  216 FORMAT(///)
  215 FORMAT(10(F6.4,2X))
  217 FORMAT(10(F6.1,2X))
  300 STOP
      END
```

```
C
C      ************************************
C      ****                            ****
C      **** SUBROUTINE DTRNS(M,NM,X,RX) ****
C      ****                            ****
C      ************************************
C
       SUBROUTINE DTRNS(M,NM,X,RX)
C
C      THIS SUBROUTINE REPRESENTS A NUMBER IN MOD FORM.
C
       IMPLICIT INTEGER (C-H,O-Z)
       INTEGER M(4),RX(4)
       P=X
       DO 7 J=1,NM
       S1=M(J)
 23    IF(X .LT. 0) GO TO 22
       GO TO 25
 22    X=X+S1
       GO TO 23
 25    X=MOD(X,S1)+1
       GO TO(12,13,14,15),J
 12    RX(1)=X
       GO TO 7
 13    RX(2)=X
       GO TO 7
 14    RX(3)=X
       GO TO 7
 15    RX(4)=X
       GO TO 7
  7    X=P
       RETURN
       END
```

```
C
C     ****************************************
C     ****                                ****
C     **** SUBROUTINE FUNTBL(M,NM,FM,C1M,C2M) ****
C     ****                                ****
C     ****************************************
C
      SUBROUTINE FUNTBL(M,NM,FM,C1M,C2M)
      IMPLICIT INTEGER (A-H,O-Z)
      INTEGER M(4),FM(4,32,32),C1M(4),C2M(4)
      COMMON/B2/N2
      DO 5 K=1,NM
C
C     THE K LOOP GENERATES THE LOOKUP TABLE FOR ALL 4 MODULLI
C     THE J LOOP GENERATES ALL POSSIBLE COMBINATIONS OF X2.
C     THE I LOOP GENERATES ALL POSSIBLE COMBINATIONS OF X1.
C
      DO 1 I=1,N2
      DO 2 J=1,N2
      X1=I-1
      X2=J-1
      F1=C1M(K)*X1 + C2M(K)*X2
    3 IF(F1.LT.M(K)) GO TO 6
      F1=F1-M(K)
      GO TO 3
    6 FM(K,I,J)=F1
    2 CONTINUE
    1 CONTINUE
    5 CONTINUE
      RETURN
      END
```

```
C
C      **********************************************
C      ****                                     ****
C      **** SUBROUTINE GSCALR(NM,M,MIN,FY1Y2,FY3Y4) ****
C      ****                                     ****
C      **********************************************
C
       SUBROUTINE GSCALR(NM,M,MIN,MB,FY1Y2,FY3Y4)
       IMPLICIT INTEGER(A-H,O-Z)
       REAL FPT1,FPT2,FPT3,FPT4
       INTEGER M(4),MIN(4),RMIN(4),Y(4)
       INTEGER FY1Y2(32,31),FY3Y4(29,27)
       INTEGER MB(NM)
       COMMON/B3/CM,CMB,XS
C
C      THE FUNCTION TABLE FY1Y2 IS CALCULATED HERE.
C
       M1=M(1)
       M2=M(2)
       M3=M(3)
       M4=M(4)
       DO 30 YM1=1,M1
       Y(1)=YM1-1
       DO 30 YM2=1,M2
       Y(2)=YM2-1
       DO 40 I=1,2
       RMIN(I)=Y(I)*MIN(I)
 40    RMIN(I)=MOD(RMIN(I),M(I))
       FPT1=MB(1)*RMIN(1)/XS + 0.5
       FPT2=MB(2)*RMIN(2)/XS +0.5
       FT1=FPT1+FPT2
 30    FY1Y2(YM1,YM2)=MOD(FT1,CMB)
C
C      THE FUNCTION FY3Y4 IS CALCULATED HERE.
C
       DO 60 YM3=1,M3
       Y(3)=YM3-1
       DO 60 YM4=1,M4
       Y(4)=YM4-1
       DO 70 I=3,4
       RMIN(I)=Y(I)*MIN(I)
 70    RMIN(I)=MOD(RMIN(I),M(I))
       FPT3=MB(3)*RMIN(3)/XS
       FPT4=MB(4)*RMIN(4)/XS
       FT2=FPT3+FPT4
 60    FY3Y4(YM3,YM4)=MOD(FT2,CMB)
       RETURN
       END
```

```
C       ***************************************************
C
C       THE PROGRAM SHOWS THE TIME DOMAIN RESPONSE OF A GIVEN
C       FILTER TRANSFER FUNCTION CORRESPONDING TO A SINE OR
C       SQUARE WAVE INPUT SIGNAL.
C
C       ABBREVIATIONS USED IN THE PROGRAM ARE AS FOLLOWS :-
C
C       TYPFLT = TYPE OF FILTER TRANSFER FUNCTION
C
C              = 0 FOR LOW PASS FILTER
C              = 1 FOR HIGH PASS FILTER
C
C       MODIMP = MODE OF IMPLEMENTATION
C
C              = 0 FOR REPRESENTING THE COEFFICIENTS
C                  IN MIXED FORM
C              = 1 FOR REPRESENTING THE COEFFICIENTS
C                  IN INTEGER FORM
C
C       TYPINP = TYPE OF INPUT TEST SIGNAL USED
C
C              = 0 FOR SINE WAVE
C              = 1 FOR SQUARE WAVE
C
C       LCF    = LOWER CUTOFF FREQUENCY
C       UCF    = UPPER CUTOFF FREQUENCY
C       SCF    = SCALE FACTOR USED FOR REPRESENTING
C                THE COEFFEICIENTS IN INTEGERS
C       TF     = TEST FREQUENCIES VIZ TF1,TF2,TF3
C
C       ***************************************************
C
        DIMENSION X(100),Y(100),TT(100)
        REAL NF,LCF
        INTEGER TYPFLT,TYPINP,SCF
        INTEGER IX(100),IY(100)
        READ,TYPFLT,MODIMP
        READ,TYPINP
        READ,LCF,UCF
        READ,TF1,TF2,TF3
        PYE=22./7.
        SEM=70..
        SCF=32*31
        NS=100
        TI=0.00005
        CN=4.0/TI/TI
        NF=1/(2.*TI)
        IF(TYPFLT .EQ. 0) GO TO 57
        WO=2. *PYE * UCF
```

```
C
C     COEFFICIENTS OF THE HIGH PASS FILTER
C
      WOSQ=WO * WO
      BO=WOSQ +2.828*WO/TI +CN
      B1=(2. *WOSQ - 2.*CN)/BO
      B2=(WOSQ - 2.828*WO/TI + CN)/BO
      AO=CN/BO
      A1=-2. * CN/BO
      A2=AO
      PRINT 203,UCF,AO,A1,A2,B1,B2
      GO TO 58
 57   WO=2. *PYE * LCF
C
C     COEFFICIENTS OF THE LOW PASS FILTER
C
      WOSQ=WO * WO
      BO=WOSQ +2.828*WO/TI +CN
      B1=(2. *WOSQ - 2.*CN)/BO
      B2=(WOSQ - 2.828*WO/TI + CN)/BO
      AO=WOSQ/BO
      A1=2.*WOSQ/BO
      A2=AO
      PRINT 202,LCF,AO,A1,A2,B1,B2
 58   IF(MODIMP .EQ. 0) GO TO 53
C
C     THE COEFFICIENTS ARE CONVERTED TO INTEGERS
C     AFTER MULTIPLYING THEM WITH A SUITABLE
C     SCALE FACTOR SCF AND ROUNDING THEM TO
C     INTEGERS
C
      IA0=A0*SCF+0.5
      IA1=A1*SCF-0.5
      IA2=A2*SCF+0.5
      IB1=B1*SCF-0.5
      IB2=B2*SCF+0.5
      PRINT 204,SCF,SEM
 53   DO 5 K=1,3
      GO TO (6,7,8),K
  6   F=TF1
      GO TO 9
  7   F=TF2
      GO TO 9
  8   F=TF3
  9   IF(TYPINP .EQ. 0) GO TO 71
      PRINT 206
      T=1/F
      TB2=T/2.
      DO 1 I=1,NS
      P=I
      TT(I)=P
      ST=TI*P
      ST=MOD(ST,T)
      IF(ST .GE. TB2) GO TO 81
      X(I)=50.0
```

```
      GO TO 1
 81   X(I)=0.0
  1   CONTINUE
      GO TO 82
 71   PRINT 205
      DO 2 I=1,NS
      TT(I)=FLOAT(I)
      IF(MODIMP .EQ. 1) GO TO 51
      X(I)=SIN((F/NF)*PYE*I)
      GO TO 2
 51   IX(I)=SEM*SIN((F/NF)*PYE*I)
      X(I)=FLOAT(IX(I))
  2   CONTINUE
 82   Y(1)=0.0
      Y(2)=0.0
      IY(1)=0
      IY(2)=0
  3   IF(MODIMP .EQ. 1) GO TO 63
      DO 4 I=3,NS
  4   Y(I)=A0*X(I)+A1*X(I-1)+A2*X(I-2)
     $-B1*Y(I-1)-B2*Y(I-2)
      GO TO 61
 63   DO 62 I=3,NS
      IY(I)=(IA0*IX(I)+IA1*IX(I-1)+IA2*IX(I-2)
     $-IB1*IY(I-1)-IB2*IY(I-2))/SCF
 62   Y(I)=FLOAT(IY(I))
 61   CALL PLOT3(TT,X,NS)
      PRINT 200,F
200   FORMAT(52X,'INPUT TO THE FILTER -F=',F4.0,'HZ.')
      CALL PLOT3(TT,Y,NS)
      PRINT 201,F
201   FORMAT(47X,'OUTPUT OF THE FILTER AT F=',F6.0,'HZ.',//)
  5   CONTINUE
202   FORMAT('1',4X,'TYPE OF FILTER  :- LOW PASS FILTER',//,
     $4x,'CUT OFF FREQ.    =   ',F7.1,1X,'HZ.',/,4X,'COEF. OF',
     $' FILTER = ',/,4X,'A0 =   ',F11.8,/,4X,'A1 =   ',F11.8,/,
     $4X,'A2 = ',F11.8,/,4X,'B1 =   ',F11.8,/,4X,
     $'B2 = ',F11.8,//)
203   FORMAT('1',4X,'TYPE OF FILTER :- HIGH PASS',//,4X,
     $'CUT OFF FREQ.    =   ',F7.1,1X,'HZ.',/,4X,'COEF. OF',
     $'FILTER = ',/,4X,'A0 =   ',F11.8,/,4X,'A1 =   ',F11.8,/,
     $4X,'A2 = ',F11.8,/,4X,'B1 =   ',F11.8,/,4X,
     $'B2 = ',F11.8,//)
204   FORMAT(4X,'MODE OF IMPLEMENTATION = INTEGER REP.',/,
     $4X,'SCALE FACTOR OF THE COEF. = ',F5.1,/,4X,
     $'SIZE OF THE INPUT SIGNAL = ',F5.1,/)
205   FORMAT(4X,'INPUT TEST SIGNAL = SINE WAVE',/////)
206   FORMAT(4X,'INPUT TEST SIGNAL = SQUARE WAVE',/////)
      STOP
      END
```

```
C       *************************************************************
C
C       MAIN PROGRAM
C       ------------
C
C       THE PROGRAM IS A HARDWARE SIMULATION OF A TWO DIMENSIONAL
C       RECURSIVE DIGITAL FILTER. THE SIZE OF THE FILTER USED FOR
C       SIMULATION IS 3*3.THE FILTER OPERATES IN THE RESIDUE NO.
C       SYSTEM.
C       THE VARIOUS NOMENCLATURE USED IN THE PROGRAM ARE LISTED
C       BELOW:-
C
C       X(I,J)     = INPUT ARRAY TO BE FILTERED.
C       YO(I,J)    = OUTPUT ARRAY.
C       A(I,J)     = NUM. COEF. OF THE FILTER (IN FLOATING PT.).
C       B(I,J)     = DENOM. COEF. OF THE FILTER (IN FLOATING PT.).
C       AM(I,J,K)  = NUM. COEF. OF THE FILTER IN MOD FORM.
C       BM(I,J,K)  = DENOM. COEF. OF THE FILTER IN MOD FORM.
C       XMN1(K)    = X(M,N-1)
C
C       SIMILARLY ALL THE OTHER NOTATIONS CAN BE INTERPRETED
C       VIZ. XM1N(K),------,XM2N2(K).
C
C       THE DATA REQUIRED TO RUN THE PROGRAM ARE CATEGORIZED
C       AS BELOW:
C
C       IFT        :-  TYPE OF FILTER USED IN SIMULATION
C                      VIZ. LOW PASS,HIGH PASS.........
C
C       NRO*NCOL :-  SIZE OF THE I/O ARRAY.
C       COEFFICIENTS OF THE FILTER TRANSFER FUNCTION
C       NO. OF MODULI USED :- NM
C       MODULI USED FOR IMPLEMENTATION :- M(I)
C       SCALE FACTOR FOR THE COEFFICIENTS :- X(S)
C
C       *************************************************************
C
        IMPLICIT INTEGER (A-H,O-Z)
        REAL X(33,33),YO(33,33),A(3,3),B(3,3)
        REAL PYE,EM,SEM,VARXMN,IX1,JX1,GF,IW1,JW2,W
       $     SNCOF,SRNCOF,SDCOF,SRDCOF
        INTEGER AM(3,3,5),BM(3,3,5)
        INTEGER M(5),MIN(5),MB(5),RMIN(5)
        INTEGER XR(33,33),XAMN(33,33,5),
       $        XMN(5),XMN1(5),XMN2(5),XM1N(5),XM1N1(5),XM1N2(5),
       $        XM2N(5),XM2N1(5),XM2N2(5),XM(5)
        INTEGER XDM1N(31,5),XDM2N(31,5)
        INTEGER YMN(5),YMN1(5),YMN2(5),YM1N(5),YM1N1(5),YM1N2(5),
       $        YM2N(5),YM2N1(5),YM2N2(5),YSMN(5)
        INTEGER YDM1N(31,5),YDM2N(31,5)
        INTEGER AD2(5),AD4(5),SIMN(5)
        INTEGER FP(20,20),FP1(20,20)
        COMMON/B1/M
```

```
C
C     BLOCK B1 IS COMMON WITH THE SUBROUTINE NDFUNC.
C
      COMMON/B2/ITRX,ITRY
      COMMON/B3/XS,CM,PDR,SPDR,CMB
C
C     BLOCK B2,B3 IS COMMON WITH THE SUBROUTINE GSCALR.
C
      DATA IP,OP/5,6/
      READ(IP,100) IFT
100   FORMAT(A4)
      READ(IP,101) NRO,NCOL
101   FORMAT(2(I3,3X))
      READ(IP,102) M1,N1,NM,S
102   FORMAT(4(I3,3X))
      READ(IP,103) (M(I),I=1,NM),(MIN(I),I=1,NM)
103   FORMAT(10(I3,3X))
      DO 1 I=1,M1
1     READ(IP,104) (A(I,J),J=1,N1)
      DO 2 I=1,M1
2     READ(IP,104) (B(I,J),J=1,N1)
104   FORMAT(3(E14.7,3X))
      WRITE(OP,200) IFT
200   FORMAT('1',3X,'TYPE OF FILTER = ',A4,/,4X,14('*'),///)
      WRITE(OP,260)
260   FORMAT(4X,'SPECIFICATIONS :-',/,4X,14('='),//)
      WRITE(OP,261)M1,N1
261   FORMAT(4X,'TYPE OF SYMMETRY   :- CIRCULAR',/
     $,4X,'SIZE OF THE FILTER :- ',I1,'*',I1,/
     $,4X,'H(W1,W2) = 1.0        SQRT(W1**2 + W2**2)<0.4*PYE'
     $,//)
      WRITE(OP,201)
201   FORMAT(' ',3X,'NUMERATOR COEFFECIENTS A(M,N)',/,
     $4X,29('='),//)
      DO 3 I=1,M1
3     WRITE(OP,202) (A(I,J),J=1,N1)
202   FORMAT(4X,4(E14.7,3X),/)
      WRITE(OP,203)
203   FORMAT(//,4X,'DENOMINATOR COEFFECIENTS B(M,N)',/,
     $4X,31('='),//)
      DO 4 I=1,M1
4     WRITE(OP,202) (B(I,J),J=1,N1)
      WRITE(OP,205) NM,S
205   FORMAT(//,4X,'NO. OF MODULI = ',I2,4X,
     $'MULTIPLICATION FACTOR(OF INPUT SIGNAL) = ',I3,//)
      WRITE(OP,206)
206   FORMAT(4X,'MODULI AND ITS MULTIPLICATIVE INVERSE',/
     $,4X,37('='),//)
      WRITE(OP,207) (I,M(I),I,MIN(I),I=1,NM)
207   FORMAT(4X,'M(',I1,')=',I3,5X,'MIN(',I1,')=',I3,/)
      CM=1
      NHFP=7
      NVFP=10
      NCOL2=NCOL-2
```

```
      GF=0.035935
      PYE=3.141592
      EM=1.0
      SEM=FLOAT(S)*EM
C
C     MODULO CAPITAL M =M1*M2*M3*M4*M5 IS CALCULATED HERE.
C
      DO 5 I=1,NM
    5 CM=CM*M(I)
      XS=CM/M(1)/M(2)
      CMB=CM/XS
C
C     THE VALUE OF MBAR OF ALL THE MODULI IS CALCULATED HERE.
C
      DO 6 I=1,NM
    6 MB(I)=CM/M(I)
C
C     THE PROGRAM FINDS WHETHER THE VALUE OF CAPITOL MODULO
C     M IS AN ODD FUNCTION OR EVEN FUNCTION AND ACCORDINGLY
C     ASSIGNS DIFFERENT VALUES OF THE POSITIVE DYNAMIC
C     RANGE.
C
      C=MOD(CM,2)
      IF(C.EQ.0) GO TO 10
      PDR=(CM-1)/2
      SPDR=(CMB-1)/2
      GO TO 11
   10 PDR=CM/2-1
      SPDR=CMB/2-1
   11 WRITE(OP,208) CM,XS,CMB,PDR,SPDR
  208 FORMAT(//,4X,'CAPITOL MODULO M(TOTAL NO. RANGE) = ',I7,/
     $,4X,'SCALE FACTOR OF THE FILTER COEFFECIENTS = ',I4,/
     $,4X,'SCALED NO. RANGE = ',I4,/,4X,'POSITIVE DYNAMIC'
     $,' RANGE = ',I8,/,4X,'SCALED POSITIVE DYNAMIC RANGE = ',
     $I4,//)
      WRITE(OP,209)
  209 FORMAT(4X,'M-HUT',/,4X,5('='),//)
      WRITE(OP,210) (I,MB(I),I=1,NM)
  210 FORMAT(4X,'MB(',I1,')=',I5,/)
C
C     THE COEFFICIENTS ARE MULTIPLIED WITH THE SCALE FACTOR,
C     ROUNDED TO INTEGERS AND CODED IN RESIDUES HERE.
C
      DO 12 IC=1,M1
      DO 12 JC=1,N1
      SNCOF=A(IC,JC)*FLOAT(XS)*GF
      IF(SNCOF .LT. 0.) GO TO 54
      SRNCOF=SNCOF+0.5
      GO TO 55
   54 SRNCOF=SNCOF-0.5
   55 NCOF=SRNCOF
      CALL DTRNS(NM,M,NCOF,XM)
      DO 29 K=1,NM
   29 AM(IC,JC,K)=XM(K)
```

```
      SDCOF=-(B(IC,JC))*XS
      IF(SDCOF .LT. 0.) GO TO 56
      SRDCOF=SDCOF + 0.5
      GO TO 57
 56   SRDCOF=SDCOF - 0.5
 57   DCOF=SRDCOF
      CALL DTRNS(NM,M,DCOF,XM)
      DO 31 K=1,NM
 31   BM(IC,JC,K)=XM(K)
 12   CONTINUE
      WRITE(OP,211)
211   FORMAT(//,4X,'NUMERATOR COEFFECIENTS - A(M,N)',
     $' (IN MOD FORM)',/,4X,46('='),//)
      WRITE(OP,212)
212   FORMAT(8X,'M(1)',4X,'M(2)',4X,'M(3)',4X,'M(4)',4X,
     $'M(5)',//)
      DO 13 I=1,M1
      I1=I-1
      DO 13 J=1,N1
      J1=J-1
 13   WRITE(OP,213) I1,J1,(AM(I,J,K),K=1,NM)
213   FORMAT(4X,'A',2(I1),'=',5(I3,5X),/)
      WRITE(OP,214)
214   FORMAT(//,4X,'DENOMINATOR COEFFECIENTS B(M,N)',
     $' (IN MOD FORM)',/,4X,46('='),//)
      WRITE(OP,212)
      DO 14 I=1,M1
      I1=I-1
      DO 14 J=1,N1
      J1=J-1
 14   WRITE(OP,215) I1,J1,(BM(I,J,K),K=1,NM)
215   FORMAT(4X,'B',2(I1),'=',5(I3,5X),/)
      DO 95 IW=1,NHFP
      IW1=FLOAT(IW)*0.05
      DO 96 JW=1,NVFP
      JW2=FLOAT(JW)*0.05
      W=SQRT(IW1*IW1+JW2*JW2)
C
C     THE TWO DIMENSIONAL CIRCULARLY SYMMETRIC SINUSOIDAL
C     SIGNAL REQUIRED FOR TESTING OF THE RNS FILTER IS
C     GENERATED HERE.
C
      DO 15 IX=1,NRO
      IX1=FLOAT(IX)
      DO 15 JX=1,NCOL
      JX1=FLOAT(JX)
      VARXMN=SQRT(IX1*IX1+JX1*JX1)
      X(IX,JX)=SIN(W * PYE *VARXMN) * SEM
      IF(X(IX,JX).LT.0) GO TO 33
      XR(IX,JX)=X(IX,JX)+0.5
      GO TO 24
 33   XR(IX,JX)=X(IX,JX)-0.5
 24   XI=XR(IX,JX)
      CALL DTRNS(NM,M,XI,XM)
```

```
      DO 34 K=1,NM
 34   XAMN(IX,JX,K)=XM(K)
 15   CONTINUE
      WRITE(OP,217)
217   FORMAT('1',3X,'INPUT SIGNAL TO THE FILTER',/,4X,
     $26('='),//)
      DO 43 I=1,NRO
      WRITE(OP,218) (X(I,J),J=1,NCOL)
 43   WRITE(OP,221)
218   FORMAT(11(3X,F7.3),/)
221   FORMAT(//)
      WRITE(OP,219)
219   FORMAT('1',3X,'ROUNDED INPUT SIGNAL TO THE FILTER',/,
     $4X,34('='),//)
      DO 45 I=1,NRO
      WRITE(OP,220) (XR(I,J),J=1,NCOL)
 45   WRITE(OP,221)
220   FORMAT(11(3X,I4),/)
C
C     THE FIFO'S PROVIDING ROW DELAYS ARE INITIALIZED AT THE
C     BEGINNING OF THE COMPUTATION HERE.
C
      DO 16 K=1,NM
      DO 58 J=1,NCOL2
      J1=J+2
      P=NCOL-J1+3
      XDM2N(J,K)=0
      XDM1N(J,K)=0
      YDM2N(J,K)=0
 58   YDM1N(J,K)=0
 16   CONTINUE
      QMAX=0
      QMIN=0
      DO 37 ITRX=3,NRO
C
C     EACH COLUMN DELAY ELEMENT IS INITIALIZED AFTER THE
C     COMPUTATION OF EACH ROW OF THE GIVEN MATRIX HERE.
C
      DO 39 K=1,NM
      XMN1(K)=0
      XMN2(K)=0
      XM1N1(K)=0
      XM1N2(K)=0
      XM2N1(K)=0
      XM2N2(K)=0
      YMN1(K)=0
      YMN2(K)=0
      YM1N1(K)=0
      YM1N2(K)=0
      YM2N1(K)=0
 39   YM2N2(K)=0
      DO 20 ITRY=3,NCOL
      DO 40 K=1,NM
 40   XMN(K)=XAMN(ITRX,ITRY,K)
```

```
      CALL NDFUNC(M1,N1,NM,NCOL2,AM,XMN1,XMN2,XDM1N,XM1N1,XM1N2
     $              ,XDM2N,XM2N1,XM2N2,AD2)
      CALL NDFUNC(M1,N1,NM,NCOL2,BM,YMN1,YMN2,YDM1N,YM1N1,YM1N2
     $              ,YDM2N,YM2N1,YM2N2,AD4)
      DO 21 K=1,NM
      SIMN(K)=AD2(K)+AD4(K)
   21 SIMN(K)=MOD(SIMN(K),M(K))
      DO 25 K=1,NM
      YMN(K)=SIMN(K)+AM(1,1,K)*XMN(K)
   25 YMN(K)=MOD(YMN(K),M(K))
      CALL GSCALR(NRO,NCOL,NM,M,MIN,YMN,YSN,YSMN,YO,RMIN,MB)
      IF(YSN .GT. QMAX) QMAX=YSN
      IF(YSN .LT. QMIN) QMIN=YSN
      CALL SHIFT(NM,NCOL2,YSMN,YMN1,YMN2,YM1N1,YM1N2,
     $             YM2N1,YM2N2,YDM2N,YDM1N)
      CALL SHIFT(NM,NCOL2,XMN,XMN1,XMN2,XM1N1,XM1N2,
     $             XM2N1,XM2N2,XDM2N,XDM1N)
   20 CONTINUE
   37 CONTINUE
      WRITE(OP,249)
  249 FORMAT('1',4X,'RESULTING OUTPUT ARRAY OF THE FILTER :-',/
     $,5X,37('='),//)
      DO 53 I=3,NRO
      WRITE(OP,250) (YO(I,J),J=3,NCOL)
   53 WRITE(OP,221)
      FP(IW,JW)=QMAX
      FP1(IW,JW)=IABS(QMIN)
   96 CONTINUE
   95 CONTINUE
      WRITE(OP,266)
  266 FORMAT('1',3X,'MAGNITUDE RESPONSE',/,4X,18('='),//)
      DO 97 I=1,NHFP
   97 WRITE(OP,220) (FP(I,J),J=1,NVFP)
      WRITE(OP,221)
      DO 98 I=1,NHFP
   98 WRITE(OP,220) (FP1(I,J),J=1,NVFP)
  250 FORMAT(11(3X,F4.0),/)
      STOP
      END
```

```
C     ****************************************************
C     ***                                              ***
C     ***  SUBROUTINE NDFUNC(M1,N1,NM,NCOL2,COFM,DMN,  ***
C     ***  DMN1,DMN2,DM1N1,DM1N2,RDM2N,DM2N1,DM2N2,ADS2) ***
C     ***                                              ***
C     ****************************************************
C
      SUBROUTINE NDFUNC(M1,N1,NM,NCOL2,COFM,DMN1,DMN2,RDM1N,
     $                 DM1N1,DM1N2,RDM2N,DM2N1,DM2N2,ADS2)
      IMPLICIT INTEGER(A-H,O-Z)
      INTEGER COFM(M1,N1,NM)
      INTEGER DMN1(NM),DMN2(NM),DM1N1(NM),
     $        DM1N2(NM),DM2N1(NM),DM2N2(NM)
      INTEGER RDM1N(NCOL2,NM),RDM2N(NCOL2,NM)
      INTEGER M(5),ADS1(5),ADS2(5)
      INTEGER F1(5),F2(5),F3(5),F4(5),F5(5),F6(5)
      COMMON/B1/M
C
C     THIS SUBROUTINE CALCULATES THE NUMERATOR AND DENOMINATOR
C     FUNCTION TABLES.
C
      DO 18 K=1,NM
      F2(K)=COFM(1,2,K)*DMN1(K)+COFM(1,3,K)*DMN2(K)
      F2(K)=MOD(F2(K),M(K))
      F3(K)=COFM(2,1,K)*RDM1N(NCOL2,K)+COFM(2,2,K)*DM1N1(K)
      F3(K)=MOD(F3(K),M(K))
      F4(K)=F3(K)+COFM(2,3,K)*DM1N2(K)
      F4(K)=MOD(F4(K),M(K))
      F5(K)=COFM(3,1,K)*RDM2N(NCOL2,K)+COFM(3,2,K)*DM2N1(K)
      F5(K)=MOD(F5(K),M(K))
      F6(K)=F5(K)+COFM(3,3,K)*DM2N2(K)
      F6(K)=MOD(F6(K),M(K))
      ADS1(K)=F4(K)+F6(K)
      ADS1(K)=MOD(ADS1(K),M(K))
      ADS2(K)=ADS1(K)+F2(K)
   18 ADS2(K)=MOD(ADS2(K),M(K))
      RETURN
      END
```

```
C     ************************************************
C     ***                                          ***
C     ***  SUBROUTINE SHIFT(NM,DMN,DMN1,DMN2,DM1N1, ***
C     ***                  DM1N2,DM2N1,DM2N2)      ***
C     ***                                          ***
C     ************************************************
C
      SUBROUTINE SHIFT(NM,NCOL2,DMN,DMN1,DMN2,DM1N1,DM1N2,
     $                 DM2N1,DM2N2,RDM2N,RDM1N)
      IMPLICIT INTEGER(A-H,O-Z)
      INTEGER DMN(NM),DMN1(NM),DMN2(NM),DM1N1(NM),
     $        DM1N2(NM),DM2N1(NM),DM2N2(NM)
      INTEGER RDM2N(NCOL2,NM),RDM1N(NCOL2,NM)
C
C     THIS SUBROUTINE SHIFTS THE DATA COLUMN WISE
C     AND ROW WISE.
C
      DO 22 K=1,NM
      DM2N2(K)=DM2N1(K)
 22   DM2N1(K)=RDM2N(NCOL2,K)
      CALL RODEL(NM,NCOL2,RDM2N)
      DO 23 K=1,NM
 23   RDM2N(1,K)=RDM1N(NCOL2,K)
      DO 24 K=1,NM
      DM1N2(K)=DM1N1(K)
 24   DM1N1(K)=RDM1N(NCOL2,K)
      CALL RODEL(NM,NCOL2,RDM1N)
      DO 25 K=1,NM
 25   RDM1N(1,K)=DMN(K)
      DO 26 K=1,NM
      DMN2(K)=DMN1(K)
 26   DMN1(K)=DMN(K)
      RETURN
      END
```

```
C     ******************************************
C     ***                                    ***
C     ***  SUBROUTINE RODEL(NM,NCOL,RDELAY,DMN) ***
C     ***                                    ***
C     ******************************************
C
      SUBROUTINE RODEL(NM,NCOL2,RDELAY)
C
C     THIS SUBROUTINE IN CONJUNCTION WITH THE SHIFT
C     SUBROUTINE PROVIDES THE ROW DELAYS.
C
      IMPLICIT INTEGER(A-H,O-Z)
      INTEGER RDELAY(NCOL2,NM)
      DO 26 J=2,NCOL2
      J2=NCOL2-(J-2)
      J1=J2-1
      DO 26 K=1,NM
 26   RDELAY(J2,K)=RDELAY(J1,K)
      RETURN
      END
```

REFERENCES

1. L.R. Rabiner and B. Gold 'Theory And Application Of Digital Signal Processing' Englewood Cliffs, N.J., Prentice Hall, 1975.

2. Antoniou 'Digital Filters Analysis And Design' McGraw Hill Book Company, 1979.

3. A.V. Oppenheim and R.W. Shafer 'Digital Signal Processing' Englewood Cliffs, N.J., Prentice Hall, 1975.

4. A. Peled and B. Liu 'Digital Signal Processing' John Wiley & Sons. New York, 1976.

5. L.B. Jackson 'Roundoff-Noise Analysis For Fixed Point Digital Filters Realized In Cascade Or Parallel Form.' IEEE Trans. on AE, vol. 18, No. 2, June 1970.

6. A.Crosier, et al. 'U.S. Patent No. 3,777, 130 Dec. 4, 1983.' Also available in Digital Signal Computers and Processors IEEE Press, 1977.

7. Jackson Kaiser & McDonald 'An Approach To The Implementation Of Digital Filters' IEEE Trans. on ASSP, vol. 22, No. 3, pp. 413-421, September 1968.

8. J. Allen 'Computer Architecture For Signal Processing' Proc. of IEEE, vol. 63, No. 4, pp. 624-633, April 1975.

9. S.L. Freeny 'Special Purpose Hardware For Digital Filtering' Proc. of IEEE, vol. 63, No. 4, pp. 633-648, April 1975.

10. A. Peled And B. Liu 'A New Hardware Realization Of Digital filters' IEEE Trans. on ASSP, vol. 22, No. 6, pp. 456/462, December 1974.

11. Agarwal and Burrus 'New Recursive Digital Filter Structures Having Low Sensitvity And Round-Off Noise' IEEE Trans. on Circuits and Systems, vol. 22, No. 12, pp. 921-927, December 1975.

12. Abu-el-Haija, Shenoi and Peterson 'Digital Filter Structures Having Low Errors And Simple Hardware Implementation' IEEE Trans. on Circuits and Systems, vol. 25, No. 8, pp. 593-599, August 1978.

13. Gnansekaran, Mondal and Mitra 'Hardware Implementation Of Two Dimensional Digital Filters Using ROM' IEEE Conf. on ASSP

14. Mitra and Chakrabarti 'A New Realization Method For Two Dimensional Digital Transfer functions' IEEE Trans. on circuits and Systems, vol. 26, No. 6, pp. 544-550, December 1978.

15. Huang Peterson and Others 'Implementation Of A Fast Digital Processor Using Residue Number System' IEEE Trans. on Circuits and Systems, vol. 28, No. 1, January 1981.

16. Schmalzel, Hein and Ahmed 'Some Pedagogical Considerations of Digital Filter Hardware Considerations' IEEE Circuits and Systems Magazine, vol. 2, No. 1, pp. 4-13, 1980.

17. Lim Chen 'Undergraduate Project Report' Dept. of Electricl Engg., University of Windsor, Canada, March 1980.

18. N.S. Szabo and R.I. Tanaka 'Residue Arithmetic And Its Application To Computer Technology' McGraw Hill Book Company, New York, 1967.

19. G.A. Jullien and W.K. Jenkins 'The Applications Of Residue Number Systems To Digital Signal Processing' A report

20. W.K Jenkins 'Recent Advances In Residue Number Techniques For Recursive Digital Filtering' IEEE Trans. on ASSP, vol. 27, No. 1, pp. 19-30, February 1979.

21. G.A. Jullien 'Residue Number Scaling And Other Operations Using ROM Arrays' IEEE Trans. on Computers, vol. 27, No. 4, pp. 325-336, April 1978.

22. A.Svoboda and M.Valach 'Rational Numerical System Of Residual Classes Storje Na Zpracovani Informaci ', pp.1-29, Sbornik V, 1957.

VITA AUCTORIS

1953 Born on March 21 , in Dar-Es-Salaam, Tanzania.

1969 Completed High School education at Baroda High School, Baroda-India.

1976 Graduated from M.S.University, Baroda-India.

1976 Worked as a Research and Development Engineer at Jyoti Ltd.(Power Electronics division), Baroda-India.

1981 Worked as an Electronics Engineer at Canadian Instrumentation & Research Ltd., Toronto-Canada.

1983 Candidate for the Master Of Applied Science in Electrical Engineering at The University Of Windsor, Windsor, Ontario-Canada.