Hardware implementation of programmable coefficients second order recursive digital filter.

Kamel. Niksan

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/2271

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS REÇUE
HARDWARE IMPLEMENTATION OF PROGRAMMABLE COEFFICIENTS SECOND ORDER
RECURSIVE DIGITAL FILTER

by

Kamel Niksan

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

Windsor, Ontario, 1984

(C) Kamel Niksan, 1984
ABSTRACT

A microprocessor structure and a hardwired logic based structure of a programmable coefficient's digital filter are discussed in this thesis. These structures have a dynamic range of 8-bits with an internal arithmetic of 12 bits. The hardware complexity of each of the structures is minimized by employing fixed-point arithmetic (2's complement notation), mixed processing (parallel and serial) and multiplication tables stored in ROM arrays.

The structure proposed was simulated in software (MATLAB), and its frequency response was derived.

The microprocessor based structure was implemented using INTEL 6800, and its frequency response was derived by feeding various frequencies covering the passband, transition, and stopband regions to the filter.

The sampling frequency of the system was found to be limited to 600kHz. In order to achieve higher sampling rate, the second structure was implemented using hardwired logic. The frequency response of this filter was derived in a fashion similar to that carried out on the above filter.

The results obtained for these filters are discussed.
ACKNOWLEDGEMENTS

I am extremely grateful to my supervisor, Dr. M. A. Sid-Ahmed for his encouragement, support, valuable discussions and the constructive criticisms throughout the course of this thesis.

I would like to take this opportunity to thank Dr. G. A. Jullien, Dr. A. C. Miller and Dr. M. Amadzi for their valuable comments and suggestions.

I wish to thank Mr. Joe Nomasau for his sincere help and co-operation throughout this project. I am also greatly indebted to Mr. Naas N. Senek for his help and the direction.

My special thanks and gratitude are due to Vijay, who helped me a lot in typing and supported me morally.

Finally, I would like to thank Mr. Manesh and Mr. Narasimhan for helping me in typing the thesis, and saving me precious time.
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Fig. No.</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Four Common Forms Of Realization</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Software Realization (using IBM-370)</td>
<td>26</td>
</tr>
<tr>
<td>2.1</td>
<td>Block Diagram Of Hardwired Logic Based Structure</td>
<td>27</td>
</tr>
<tr>
<td>2.2</td>
<td>Block Diagram Of Microprocessor Based Structure</td>
<td>28</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Master Flowchart</td>
<td>35</td>
</tr>
<tr>
<td>3.5.2</td>
<td>I/O Configuration</td>
<td>36</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Module A; process module</td>
<td>40</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Module A, level A1, Subroutine Sum</td>
<td>41</td>
</tr>
<tr>
<td>3.5.5</td>
<td>Module B, request module</td>
<td>42</td>
</tr>
<tr>
<td>4.1</td>
<td>Multiplication Tables</td>
<td>48</td>
</tr>
<tr>
<td>4.2</td>
<td>Programmable Coefficient Register (RAM)</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Programmable Coefficient Register (using PIFC)</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>Circuit Diagram Of Arithmetic Section</td>
<td>56</td>
</tr>
</tbody>
</table>
4.5 Block Diagram of Circuit Which Generates S/H, S/C, CLACC Signals

4.6 Circuit Which Generates 'OE', And 'CKSb'

4.7(a) Timing Diagram of Control Signals (One Sample)

4.7(b) Timing Diagram of Control Signals

4.8 The 'CRCOUNT' Generator

4.9 Control Signal Generator

4.10 Circuit which Produces 'BOUND' and 'CKACC'

4.11 Programmable Address Selector

4.12 A/D Converter

4.13 D/A Converter

4.14 External Component of A/D Converter

5.1(a) Frequency Response of Simulated Filter (using 1CM-370)

5.1(b) Software Simulated Time-Domain Response of Filter (Passband region)

5.1(c) Software Simulated Time-Domain Response of Filter (Transition region)

5.1(d) Software Simulated Time-Domain Response
5.2.1(a) Actual Input/Output Of Micro-processor Based Structure (F=20 Hz) .................. 86
5.2.1(b) Actual Input/Output Of Micro-processor Based Structure (F=30 Hz) ................. 86
5.2.1(c) Actual Input/Output Of Micro-processor Based Structure (F=200 Hz) ............... 87
5.2.2 Frequency Response Of Micro-Based System .............................................. 89
5.2.3(a) Actual Input/Output Of Hardwired-Logic Based Structure (F=1 kHz) ................. 91
5.2.3(b) Actual Input/Output Of Hardwired-Logic Based Structure (F=3 kHz) ................. 91
5.2.3(c) Actual Input/Output Of Hardwired-Logic Based Structure (F=5 kHz) ................. 92
5.2.4 Frequency Response Of Hardwired Logic Based Digital Filter .......................... 93
5.2.5(a) Control Pulses For Five Cycle .......................................................... 94
5.2.6(a) Control Pulses For Three Samples .................................................... 94
5.2.6(a) Control Signals .......................................................... 95
5.2.6(b) Control Signals .......................................................... 95
5.2.7 Photograph Of Implemented Hardwired
5.2.8 Photograph Of Implemented Micro-

Based Structure
## CONTENTS

**Abstract** ........................................ ii

**Acknowledgements** ................................ iii

**List of Illustrations** ............................. iv

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>INTRODUCTION .................................. 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIGITAL FILTER IMPLEMENTATIONS .............. 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Forms of digital filter structures .......... 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Types of Arithmetic .......................... 9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Forms of Number Representation .............. 11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Arithmetic Devices .......................... 12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EFFECTS OF FINITE WORD LENGTH IN DIGITAL FILTERS 14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coefficient Quantization ..................... 14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input Quantization .......................... 15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Quantization ........................ 15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPARISON BETWEEN THE RECURSIVE AND NONRECURSIVE DESIGN 16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PREVIOUS WORK .............................. 17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OBJECTIVES OF THIS RESEARCH ................. 19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>THESIS ORGANIZATION ......................... 20</td>
<td></td>
</tr>
<tr>
<td>II.</td>
<td>FILTER ARCHITECTURE CONSIDERATION .......... 22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GENERAL ...................................... 22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IMPLEMENTATION CONSIDERATION ............... 23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Implementation of Filter .......... 23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware Realisation of the Filter .......... 24</td>
<td></td>
</tr>
<tr>
<td>III.</td>
<td>MICROPROCESSOR BASED STRUCTURE ............. 30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTRODUCTION .................................. 30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHOICE OF MICROPROCESSOR ................... 31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MICROPROCESSOR DEVELOPMENT SYSTEMS .......... 32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>THE INTERFACE MODULE ....................... 33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CREATION OF THE CIRCUIT .................... 34</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Phase of Testing ........................... 44</td>
<td></td>
</tr>
<tr>
<td>IV.</td>
<td>HARDWIRED LOGIC DESIGN ..................... 46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SECOND-ORDER FILTER SPECIFICS ............... 47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Storage ..................................... 47</td>
<td></td>
</tr>
</tbody>
</table>
Arithmetic: .............................................. 55
Input/Output: ........................................... 57
Control: .................................................. 57
S/H, SC and CLACC pulse generator: .................. 58
Output enable (OE) and Clock shift
register (CKSB) circuits: .............................. 60
Multiplexer controller circuit (CKCOMT
pulse generator) ........................................ 62
Control signals generator (Using Decoder
circuitry) ................................................ 66
Round signal and CLACC pulse generator ......... 69
Programmable address selector ......................... 71
A/D AND D/A CONVERSIONS ......................... 74
Analog to Digital converters ........................... 74
successive approximation converters ................. 74
Parallel (Flash) converters ............................. 75
Digital to Analogue conversion ........................ 78
External circuit for basic operation of A/D .......... 78

V. FILTER OPERATION AND TESTING ....................... 80

Results Obtained From Software Simulation ......... 80
Results Obtained From The Hardware realizations
Microprocessor Based Structure ......................... 85
Hardwired Logic Design Structure ..................... 90

VI. Rounding Effects .................................... 97

Rounding Of Binary Numbers ........................... 97
Propagation Of The Quantization Noise Through
The System .............................................. 99
Limit Cycles: ............................................ 101

VII. CONCLUSION AND COMMENTS ....................... 105

Conclusion .............................................. 105
Comments ............................................... 106

APPENDIX - A Interface Between NOVA-And-EPRCH Programmer .... 107
APPENDIX - B Listing of Software Simulation(using IBM) .................. 118
APPENDIX - C Operating Procedure For Micro-Based Structure ........ 133
APPENDIX - D Listing Of Software Routines(using INTEL-8085) ...... 134
APPENDIX - E Description Of Filter Control Signals .................. 142
APPENDIX - F Circuit Diagrams Of Micro-Based Structure .............. 144
APPENDIX - G Circuit Diagrams Of Hardwired Logic Design ............. 154

REFERENCES ............................................ 181
Chapter I
INTRODUCTION

In the same manner that digital computers have left analog computers far behind, the technique of digital filtering promises to advance far ahead of analog filtering. Digital filtering or digital signal processing is concerned with the representation of signals by sequences of numbers and the processing of these sequences [1]. The ease with which high speed digital computers are available has quickened the development of increasingly complex and sophisticated signal processing algorithms. Also recent advances in integrated circuit technology promises the economical implementation of complex digital signal processing schemes on a chip.

The implementation of a digital filter can be accomplished in two basic ways. It can be realized as software, such as a subroutine on a digital computer or as hardware, such as an integrated circuit system utilizing registers, multiplexers and adders. For a number of years, software implementation was the only mode possible for accomplishing digital filtering. Presently the software mode is still the common method of realizing a digital filter us-
ing either a general purpose or special purpose digital computer. The hardware realization of digital filters has become greatly simplified due to the rapid development of large scale integrated circuit technology and the development of microprocessors.

In many applications, digital signal processing functions should be performed in real time and the implementation must be cost effective and competitive with alternate analog implementations [2]. Since multiplication is relatively a time consuming operation which must be performed repeatedly in a signal processing environment, ROM look-up tables are used to implement such an operation.

1.1 DIGITAL FILTER IMPLEMENTATIONS

Various selections in all steps of the implementation, are available to the hardware designer of digital filters as follows:

i.) Select a filter structure.

ii.) Select between fixed-point and floating point arithmetic.

iii.) Select a number representation.

iv.) Select the arithmetic devices.
Numerous options are obviously open to the designer and as a consequence the implementation can assume a variety of forms.

1.1.1 Forms of digital filter structures

The transfer function of a digital filter may basically be realized in two different ways:

1. Using digital hardware (adders, delay elements, multiplexers).
2. Programming on the computer.

In both cases there are different ways to realize the discrete transfer function [3]. Four basic realizations will be considered here. The transfer function of a digital filter can be commonly described in terms of its Z-domain transfer function [4].

\[
H(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{N} a_i z^{-i} = \frac{\sum_{i=0}^{N} a_i z^{-i}}{1 + \sum_{i=1}^{N} b_i z^{-i}} \quad \text{(1.1)}
\]
\[ y(n) = \sum_{i=0}^{N} a_i x(n-i) - \sum_{i=1}^{N} b_i y(n-i) \]  \hspace{2cm} (1.2)

where \( x \) is the input sequence and \( y \) is the output sequence, \( a_i \) and \( b_i \) are the filter coefficients. For the second-order section of the filter \((N=2)\) the difference equation is as follows:

\[ y(n) = a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) - b_1 y(n-1) - b_2 y(n-2) \]  \hspace{2cm} (1.3)

A simple structure for realizing the difference equation \((1.3)\) is the 'Direct Form 1' as shown in the Fig. 1.a. This structure uses separate delays for both the input and output data terms. This structure is simple and it has direct relation to the \( Z \) transform of equation \((1.1)\). If equation \((1.1)\) is rewritten in a slightly different form, i.e.

\[ H(z) = \left( \frac{1}{1 + \sum_{i=1}^{N} b_i z^{-i}} \right) \left( \sum_{i=0}^{N} a_i z^{-i} \right) \]  \hspace{2cm} (1.4)

\[ = H_1(z) \cdot H_2(z) \]  \hspace{2cm} (1.5)
the pair of difference equations that will be obtained are:

\[ w(n) = x(n) - \sum_{i=1}^{N} b_i w(n-i) \]  \hspace{1cm} (1.6)

\[ y(n) = \sum_{i=0}^{N} a_i w(n-i) \]  \hspace{1cm} (1.7)
Fig. 1(a) Direct Form 1
Fig. 1(b) Direct Form 2

Fig. 1(c) Cascade Form

Fig. 1(d) Parallel Form

Fig. (1.1) Four Common Forms of Realization
This can be realized as shown in Fig. 1b. This structure is called 'Direct Form 2'. The third structure for realizing digital filters can be obtained by writing eq (1.1) in the form of:

\[ H(z) = \frac{Y(z)}{X(z)} = a_0 \cdot \prod_{i=1}^{K} H_i(z) \quad \text{(1.8)} \]

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

\[ H_i(z) = \frac{1 + a_{1i}z^{-1} + a_{2i}z^{-2}}{1 + b_{1i}z^{-1} + b_{2i}z^{-2}} \quad \text{(1.9)} \]

or a first-order section, i.e.

\[ H_i(z) = \frac{1 + a_{1i}z^{-1}}{1 + b_{1i}z^{-1}} \quad \text{(1.10)} \]
and \( k \) is the integer part of \((N+1)/2\). This is the 'Cascade Form' for the digital filter depicted in fig. 1.1.c. [4].

One general difficulty with the cascade structure is deciding the exact order in which to cascade the individual first and second order sections [4]. A further difficulty with the cascade structure is the necessity for having scaling multipliers between the individual sections in the cascade to prevent the filter variables from being too large or too small. The fourth structure is the parallel structure shown in fig 1.1.c, which is obtained by writing eq.1.1 in its partial fraction expansion as:

\[
H(z) = C + \sum_{i=1}^{K} H_i(z) \tag{1.11}
\]

where \( H_i(z) \) is either a second order section of the form:

\[
H_i(z) = \frac{a_{0i} + a_{1i}z^{-1}}{1 + b_{1i}z^{-1} + b_{2i}z^{-2}} \tag{1.12}
\]
A first order section of the form:

\[ H(z) = \frac{a_0}{1 + b_1 z^{-1}} \]  

\[ (1.13) \]

\( k \) is the integer part of \((N+1)/2\) and \( C = a_N/b_N \).

Since the Direct form-1 is simple to implement in terms of addition and multiplication, this structure will be used in the hardware implementation [4].

1.1.2 Types of Arithmetic

There are different types of arithmetic used in the implementation of digital systems. Among the most common are fixed-point and floating-point arithmetic.

In the fixed-point representation, it is assumed that the position of the binary point is fixed. The bits to the right of the binary point represent the fractional part of the number and those to the left represent the integer part. In fixed-point arithmetic, all the bits are used to represent the amplitude or 'mantissa' of the word. In
floating point some of the bits are used for 'mantissa' (fractional) part and the rest for the exponent [4].

Floating point addition is more complicated because it requires realignment of radix points (i.e., the exponent parts of the two numbers must be made equal before addition or subtraction). That is:

\[
A = f_a \times 10^{e_a} \quad \text{--------------------------(1.14)}
\]

And

\[
B = f_b \times 10^{e_b} \quad \text{--------------------------(1.15)}
\]

where \( f_a \) and \( f_b \) are fractional part of the numbers \( A \) and \( B \) respectively and \( e_a \) and \( e_b \) are their exponent parts. Then [5]:

\[
A + B = (f_a + f_b \times 10^{-d}) \times 10^{e_a} \quad \text{for} \quad e_a > e_b \quad \text{--------------------------(1.16)}
\]

\[
A + B = (f_a \times 10^{+d} + f_b) \times 10^{e_b} \quad \text{for} \quad e_a < e_b \quad \text{--------------------------(1.17)}
\]

where, \( d = e_a - e_b \).
Floating point arithmetic leads to increased dynamic range and improved accuracy of processing. Unfortunately, it also leads to increased cost of hardware and to reduced speed of processing, because the hardware is in a sense duplicated, since both the mantissa (fractional part) and the exponent have to be manipulated. For software implementation on a general purpose digital computer, floating-point arithmetic is preferred since neither the cost of hardware nor the speed of processing is a significant factor [2].

Thus considering these factors the fixed-point representation will be used throughout the design mainly for its simplicity in handling numbers.

1.1.3 Forms Of Number Representation

Depending on the way negative numbers are represented, there are three different forms of fixed-point arithmetic representation. They are called: sign magnitude, 2's complement and 1's complement. The three number systems are best illustrated by an example [2].

The 4-bit representation (including sign) of +3 and -3 are:

<table>
<thead>
<tr>
<th></th>
<th>Sign Magnitude</th>
<th>2's Complement</th>
<th>1's Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>+3</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
</tr>
<tr>
<td>-3</td>
<td>1011</td>
<td>1101</td>
<td>1100</td>
</tr>
</tbody>
</table>
Sign magnitude addition is complicated by the fact that the inputs with like sign are added but those with unlike sign must be subtracted. Since it presents inherent problems in performing simple arithmetic such as addition, it is generally avoided in digital systems. Thus the 2's complement form which is simple to handle will be employed throughout this design.

1.1.4 Arithmetic Devices

A study of eq1.2 would reveal that the basic elements for the realization of digital filter consists primarily of
i.) Delay Elements: Consists of shift registers. The size of the registers is equal to the dynamic range of the filter and the number of registers required is equal to the order of the filter (for recursive filter implementations).

ii.) Adders: Binary adders are required for digital filter realization, if the incoming data is in 2's complement form, subtraction can also be performed using the same adders.

iii.) Multipliers: The multiplier is often the most complicated element of the digital filter. The efficient way of realizing this function is to formulate and store a multiplication table using ROM's (look-up table) where the multiplicand and the multiplier form 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 word length of the ROM used. However, for most digital filter implementation purposes, the exact output is often not necessary, and hence the look-up table technique satisfies an important option in realizing multiplication.
1.2 EFFECTS OF FINITE WORD LENGTH IN DIGITAL FILTERS

In the implementation of digital filter in software or hardware numbers are ultimately stored in finite length registers. Consequently coefficients and signal values must be quantized by rounding or truncation before they can be stored. Number quantization gives rise to three types of errors [11]:

1.) Coefficient quantization errors.
2.) Input quantization errors.
3.) Product quantization errors.

1.2.1 Coefficient Quantization

The transfer function coefficients are normally evaluated to a higher degree of accuracy during the approximation step [2]. If they are quantized the frequency response of the resulting filter may differ appreciably from the desired response and depending upon the type of structure used for realization, a coefficient sensitive filter may become unstable [11].
1.2.2 **Input Quantization**

Input quantization errors arise in applications where digital filters are used to process continuous time signals. These are the errors inherent in the A/D conversion [2]. By encoding an analog signal an error noise of mean square value \( \frac{\sigma^2}{12} \), where \( \sigma \) 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 [2].

1.2.3 **Product Quantization**

Product quantization errors arise at the outputs of multipliers. Each time a signal represented by \( b_1 \) digits is multiplied by a coefficient represented by \( b_2 \) digits a product having as many as \( b_1 + b_2 \) digits is generated. Since a uniform register length in practice is used throughout the filter, each multiplier output must be rounded or truncated before processing can continue. The quantization of the product is a usual cause of noise in filter structures [2].

It is generally thought to be better to realize an nth order filter as a collection of first and/or second order subfilters having real coefficients [7]. The round off error
resulting from this system decomposition is generally smaller than that obtained using higher order subfilters.

1.3 COMPARISON BETWEEN THE RECURSIVE AND NONRECURSIVE DESIGN

Digital filters can be partitioned into two distinct classes, the finite and infinite impulse response filters. A finite impulse response filter (FIR) possesses an output response to an impulse forcing function, which is of finite duration. An infinite impulse response filter (IIR) possesses an impulse response which may persist for all time [7].

The principal advantage of the FIR structure is its ability to exhibit the linear phase versus frequency behaviour. One of the negative features of the FIR is its inability to achieve sharp magnitude response in the frequency domain. In general a high order FIR must be used to achieve sharp frequency response. Therefore high Q filters are generally architectured as IIR filters. Unlike the FIR, the IIR filter does not exhibit the phase linearity of FIR but compensates for this shortcoming by providing an improved magnitude frequency response.

The low order IIR filters generally have a higher throughput than higher order FIR filters designed to meet the same magnitude and phase specifications [7]. In practice
the cost of a digital filter tends to increase and its speed tends to decrease as the order of the transfer function is increased. Hence for high selectivity applications the choice is expected to be recursive design [2].

1.4 PREVIOUS WORK

The concepts of earlier papers will be investigated below for the purpose of realizing 'Programmable Coefficients Digital Filters'.

Jackson & Kaiser [3] have proposed an approach to the implementation of digital filters that is well suited for large scale integrated construction (LSI). They also propose a very efficient serial multiplier that produces a rounded binary number and lends itself particularly well to multiplexed operation.

Abarbanel Peled & Berg Lin [5] have proposed an approach for the hardware implementation of fixed-point arithmetic operation, the next output sample is obtained through repeated addition and shifting operations and no multiplication is needed.

Stanley [6] has presented a tutorial summary of the basic considerations necessary for designing custom digital filter hardware. Emphasis is placed on fundamental principles and not on promotion of a specific design approach [6]. The
paper begins with a review of quantization (finite accuracy and arithmetic) effects. This is followed by a discussion of the elements of digital filters, and how they are constructed. The multipliers suitable for use in the digital filters are also discussed.

Ahmad I. [10] has proposed a filter structure based on a digital incremental computer, which has low sensitivity, good error characteristic and simple hardware implementation for pole locations close to \( |z| = 1 \). The second order digital filter implemented by this structure has two arbitrary gain constants which allow some freedom in the design, and can be used to reduce the roundoff errors and limit cycle oscillations.

Johnson, R. [11] has proposed an improved binary multiplication employing a combination of look-up multiplication tables and addition units [11]. An example of multiplying two 8-bit data words comprising four look-up tables, and two addresses is presented.

Schroder, H. [12] has proposed an addressing method for implementing the high word-rate digital filters. In this method the structure of the digital filter is comprised of coefficient memory, arithmetic unit, and filtering unit. The essentials for designing such a fast tunable digital filter
and a brief description of each unit is also presented. The main advantage of the addressing method is the small number of different sets of stored data.

Neuvo, J. [13] describes the design of microprogrammed digital filter which can be used to realize modular digital filter structures. A separate RAM address processor, and a separate hardware array multiplier are included in this design to minimize the processing time. The filter is completely microprogrammed to achieve the greatest possible operating speed.

Hughes, P. M. & Cheetam, S. M. G. [14] have considered the effects of structure on microprocessor based recursive digital filters, which employ a predetermined multiplication technique based on canonical signed digit code [14]. A set of design rules for choosing the best structure to implement a second order transfer function which has pairs of zeros located at +1 or -1 in the complex z plane has also been developed.

1.5 OBJECTIVES OF THIS RESEARCH

The work assigned can be stated as follows:
1.) Simulation of a programmable coefficients  
   2nd order recursive digital filter using IBM 370.
2.) Generation of look-up tables for product of data and  
   coefficients within the filter.
3.) Development or interface routine between the EPROM programmer  
   and NOVA 640 minicomputer to program EPROM's (2732)  
   used for look-up table purposes.
4.) Hardware realization of the Programmable Coefficient 2nd order  
   recursive Digital Filter using an INTEL 8085 microprocessor.
5.) Design and construction of the above filter using a  
   dedicated hardware structure.

1.6 **Thesis Organization**

This thesis presents the design and construction of  
programmable coefficient 2nd order recursive digital filter. Chapter I summarizes the review of the previous work  
done in the literature.

In Chapter II, the architecture of the filter is con-  
sidered.

Chapter III presents the selection of microprocessor  
for this particular application and the design and construc-  
tion of a microprocessor based Digital Filter. This chapter  
gives detailed discussion on both, hardware and software  
aspects of the design.
Chapter IV gives a detailed description of the design and implementation of the filter based on Hardwired Logic.

Chapter V provides the experimental results obtained from the software simulation (using 16K-370), and from both, the microprocessor and non-microprocessor realizations (Hardwired Logic).

Chapter VI discusses the roundoff effects in digital filter.

Finally, chapter VII presents the conclusions that can be obtained from the research work presented in this thesis.
Chapter II
FILTER ARCHITECTURE CONSIDERATION

2.1 GENERAL

Selection of a digital filter architecture is influenced by many factors which include speed, cost and performance. These factors interact and a satisfactory compromise must be reached. For instance, choice of basic word length used in an implementation depends partly on the minimum attenuation and allowable ripple specifications. The required sampling frequency determines the time frame with which computations needed to compute an output sample, must be performed [15].

Next, there are some basic architectural features that must be addressed when implementing digital filters. These are as follows:

1. STORAGE - These must be storage of input and output samples, constant coefficients and any intermediate results.

2. ARITHMETIC - Circuitry to perform the various arithmetic operations required by the digital filter algorithm is necessary. Speed compromise can be made by choosing between parallel and serial architectures.
3. **INPUT/OUTPUT (I/O)** - A/D and D/A conversions are needed to interface the digital filter with the analog signals.

4. **CONTROL** - Circuitry which coordinates the various portions of the filter is needed.

The general filter architecture is derived from the filter algorithm and then the actual selection of hardware is based on the merits of speed, cost, and performance. Choice between the major logic families (i.e., metal oxide semiconductor (MOS) vs bipolar) is based primarily on speed and power consumption restrictions.

### 2.2 IMPLEMENTATION CONSIDERATION

The implementation of a digital filter can assume two forms, software and hardware. In the first form, implementation involves the simulation of the filter network on a digital computer.

The second form involves the conversion of the filter network into a dedicated piece of hardware [11].

#### 2.2.1 Software Implementation of Filter

The filter structure shown in Fig. 2.7 was simulated on the IBM 370 to study the structure for practical implementation. Flowchart of Figure 2.1 shows the operation of the software. The results obtained for low-pass filter with cut-off frequency of 100 Hz is shown and explained in the testing part of the filter (Chapter V).
The programs are documented in great detail as can be seen from Appendix-B and are self explanatory.

2.2.2 **Hardware Realisation Of The Filter**

The difference equation of a second order recursive digital filter is given by equation (1.3) can be rewritten in the following form:

\[ y_n = \sum_{k=1}^{5} r_k c_k \]

where,

- \( c_k \) is the coefficient of the filter
- \( r_k \) is the input/output data

The filter was implemented in two ways: a) A Microprocessor Structure. b) Hardware Logic Design Structure.

These two implementations are dealt with in detail in chapters III & IV respectively. The layout of both structures to realize the filter using the above difference equation are shown in figs. 2.1 & 2.2. These structures are a combination of serial and parallel structures and have a dynamic range of 8 bits. Analog input of \( \pm 5 \) volts range enters the filter through A/D converter. The A/D is triggered by the start or conversion pulse (SC). The end of conversion
line(80C) remains low until the conversion is completed. The details of the interfacing hardware is explained later.

Shift registers SR1-SR5 represents the delayed version of I/O signals. These signals are multiplexed in Mux1-Mux8 before reaching the multiplication tables.
START

Read:-- Sampling Frequency.
Cutoff Frequency.

1. Compute & Scale Coefficients
2. Convert them to 2's Complement Form.
3. Store them in Coefficient Registers.

Generate Multiplication Tables

1. Sample the Input Data
2. Convert Data to 2's Complement Form

Cycle = 1

Determine Address of Multiplication Tables using corresponding coefficients & data

Evaluate
\[ r_{k,k'} = r_{k,k} - k - 2 \cdot r_{k,k'} \]

Next Cycle

Cycle = 5?

Y

All Samples Tested?

N

Next Sample

1. Output \( y_n \)
2. Plot Response

STOP

Fig. (2) Software Realization (using IBM 370)
To perform multiplication \( r_k c_k \) in equation 2.1, a table look-up technique is used to speed up the operations of the filter. To overcome the limitation in the size of available EPROMS, the coefficients are partitioned into two or four bits parts which address a multiplication table to formulate the product. Mathematically this can be stated as:

\[
c_k = c_{kh} + 2^{-3} c_{kl}
\]

where,

- \( c_{kl} \) is 4 MSB of the Coefficient
- \( c_{kh} \) is 4 LSB of the Coefficient

The sign of the coefficient selects the appropriate multiplication table (\( \text{mult2} \) or \( \text{mult3} \)), and the output of these tables are then added to formulate the product. The \( 2^{-3} \) is achieved by using 2's complement shifting (explained in the next chapter), giving rise to 12 bit internal arithmetic.
Chapter III
MICROPROCESSOR BASED STRUCTURE

3.1 INTRODUCTION

For a given task, a given tool is required. The microprocessor makes sense when cost and packing constraints are to be met. However, there are many cases where the microprocessor is not capable of achieving the needed speed and capacity demands of a signal processing application [12].

In many applications, microprocessors are used in signal processing designs. Often, the relatively slow speed of these processors limits their use in high-speed applications. Bit slice processors use high speed technologies to achieve high throughput rates. Since bit slice processors are microprogrammed, their performance is often related to the concept of microcycle. Bit slice microprocessors are giving way to byte slice microprocessors. These high speed flexible processors have many applications in digital signal processing. The byte slice microprogrammable chip is capable of managing external memory transactions, I/O operations to a peripheral multiplier chip, ALU operations, internal data manipulations and others.
3.2 CHOICE OF MICROPROCESSOR

The choice of which microprocessor to use is based on many variables. For example, in addition to basic performance, the specifications, the question of availability, up and down compatibility with other products, and software support influence the design decision. Considering the 16-bit microprocessor data in Table 3.2.1, for example, based on memory expansion and power supply considerations, the 8080 and 8086 are good design choices. For a digital recursive filter, the memory requirements are often secondary to arithmetic speed. When speed is the primary design consideration the High Speed Processor should be considered [12].

<table>
<thead>
<tr>
<th>Item</th>
<th>8008</th>
<th>8080</th>
<th>Z80</th>
<th>990</th>
<th>8086</th>
<th>Z8000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word length</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Memory addressing</td>
<td>16K</td>
<td>64K</td>
<td>64K</td>
<td>32K</td>
<td>1024K</td>
<td>8172K</td>
</tr>
<tr>
<td>Power</td>
<td>5, -9</td>
<td>12, ±5</td>
<td>5</td>
<td>12, ±3</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

In the selection of a microprocessor the user is faced with understanding and comparing many parameters such as
cost, availability, programming aids, program check-out, architecture considerations and physical structure. However, timing plays an important role in choosing a microprocessor. At the time when this project was started, there existed two microcomputer development systems, namely the INTEL 8085 and 8086 in the Electrical Engineering Department at the University of Windsor. Due to the familiarity of the author with the INTEL 8085, this microprocessor was chosen, even though it may seem inferior in speed and memory addressed as compared to INTEL 8086.

3.3 **MICROPROCESSOR DEVELOPMENT SYSTEMS**

When designing a new microprocessor based system for a simple application, it is often unwise to start building the system from scratch by wiring up all the chips. Therefore, microprocessor manufacturers developed single board computers which contain CPU, I/O and memory chips prewired and tested, and microcomputer systems in kit form which can be easily assembled [17]. One such kit is the SDK-85. These kits are usually programmed using hexadecimal codes and have very limited software debugging aids, such as single stepping or break points, and no hardware debugging tools. A microprocessor development system provides hardware and software debugging capabilities. It contains a number of hardware devices and software packages. The main hardware
components, are the CPU, main memory, mass storage (disk and tape), CRT console, in circuit emulator, and so on. The software elements included with the development system are an editor, assembler, debugger, high level language compilers, linkage editors and operating system software.

3.4 THE INTERFACE MODULE

The interface module is mainly made up of an Analog to Digital conversion section, delays, registers, multiplexers, multiplication tables, D/A conversion section and associated latches. A schematic diagram of the module is presented in figure 2.2, arrows indicate the direction of information transfer.

For transferring and receiving data and signals to and from the external circuit, five I/O ports are used. They are: Port A, B and C of 8155 and Port #00 and #01 of 8355 chip. The address of the ports are represented in hexadecimal codes. The lines A00 and A01 of 8355 are decoded to address ports A and B. The 8155 uses lines A00-A02, used to address I/O ports A, B, and C. Hence the ports are determined by this configuration are Ports 00,01 for 8355 and Ports 21,22,23 for 8155.
3.5  OPERATION OF THE CIRCUIT

The operation of the circuit is best described by the master flowchart shown in fig. (3.5.1). The I/O configuration for this particular design is shown in Fig. 3.5.2.

As it can be seen from the MASTER flowchart, in certain cases non-uniform sampling may occur which can cause jitter.
START

Initialize I/O Ports

Obtain Data from A/D

Select Appropriate Data from MUX 1 - MUX 8. Send Coefficients c_k to MULT 1 - MULT 3.

\[ y_n = \sum_{k=1}^{F_{rm}} r_k c_k \]

Next Cycle

Cycle=5 ?

Perform Rounding and send y_n

Next Filter

Interrupt ?

STOP

Fig.(3.5.7) Master Flowchart
Fig. (3.5.2) I/O Configuration
The coefficients of the filter will be scaled initially, and then stored in blocks of memory, each block containing the coefficients of one filter module. The memory block and the word within the blocks will be addressed directly via a software routine. Initially the number of filter modules, and the scaled coefficients will be stored externally in the data memory (RAM). The internal register will be used as a block counter to keep track of the filters used, and will be decremented later when the filter computation proceeds to the next module.

In the initialization part, the following steps will be performed via software:

1. Ports 'A' and 4 LSB of port 'C' will be made as an output and port '3' as an input. These ports will then be cleared.
2. Port #00 will be made as an output and will be cleared initially.
3. Bits 60 to 67 of port #01 will be made as an output, and bit 67 will be configured as an input line.
4. The A/D converter will be reset and its output will be enabled via line 'Output Enable (OE)'.
5. Registers Sa1-Sa5 will be cleared and all the latches and the multiplication tables will be disabled.
6. Latch 3 will be enabled and then disabled, so that, the output of D/A converter will be set to its initial value (zero voltage in this case).
Port 'A' will be made as an input so that it can be used later to fetch the 8 bit value of \texttt{ckkl} from the multiplication tables \texttt{MULT2} or \texttt{MULT3}. It will be made as an output only at the end of the 5th cycle in each sampling process, that is during the time when it is required to feed the 8 bit output to the D/A converter.

Once the corresponding ports have been initialized the next step will be to digitize the input signal. This is done by sending appropriate control signals to the A/D converter (via port #01).

Once the conversion is completed and the digitized data is ready, the microprocessor will sense this signal (End of Conversion signal 'EOC') via port #01. The processor will then issue an enable command signal from this port which will enable the A/D converter to feed the digitized data into the 2's complement circuit and hence to the register [\texttt{REG1}].

At this stage, only the register [\texttt{REG1}] contains the sampled input data, and the other registers, which are used as delay elements for input and output data, have all been cleared. These registers will be clocked via 'CKSR' signal at the start of each sampling process. Then next operation of the circuit, which is described by the process module (in
the flow diagram or fig (3.5.3), will be to form an output sample 'y_n' for corresponding digitized input 'x_n' in each sample.

Since there are five variables for the 2nd order section of the digital filter, the process has to be repeated for five cycles for each sample to obtain an output for each sample. Operation of the system which is described by flow chart or figures (3.5.3) and (3.5.5) will be achieved in the following manner:
Fig.(3.5.3) Module A: Process Module
ENTRY

1. Get $r_{k}c_{kh}$ from MULT1 and store it in 'D' & 'E'
2. Clear ACC1 & ACC2

Get $r_{k}c_{kl}$ from MULT2

\[ N \quad r_{k}c_{kl} = +ve \quad Y \]

Perform 2's Comp. Right Shift (Insert three ones between 1\textsuperscript{st} and 2\textsuperscript{nd} MSB of $r_{k}c_{kl}$)

Save the Shifted Result in 'D' & 'E' Registers

Perform $r_{k}c_{k} = r_{k}c_{kh} + 2^{-5}r_{k}c_{kl}$

Add its M.S. byte and L.S. byte to ACC1 and ACC2 respectively.

EXIT

Fig.(3.5.4) Module A ; Level A1 : Subroutine Sum
Fig. (3.5.5) Module B: Request Module
1. The appropriate data \( r_k \) at inputs of multiplexers MUX1-8 will be selected (via their control lines), which will form one of the inputs to the multiplication tables MULT1 to MULT3. To select such data at inputs of multiplexers, one of the internal registers of processor (REG 'B') will used as an up-counter which will be cleared initially. Its value in each cycle will determine which of the inputs (i.e. \( x(n) \) or \( x(n-1) \) etc.) has to be selected. Port #01 will be used for this purpose.

2. The corresponding partitioned coefficient (CKH & CKL) will be fed to other inputs of multiplication tables using port #00. These multiplication tables will then be enabled via port 'C'.

3. By enabling the latches at the outputs of these tables, the results obtained at the output of multiplication table MULT1 (\( r_k \cdot CKH \)) and multiplication table MULT2 or MULT3 (\( r_k \cdot CKL \)) will be fetched by the processor via port 'B' and 'port 'C' respectively. The processor will then disable the multiplication tables and latches.

4. Upon receiving the results of \( r_k \cdot CKH \) and \( r_k \cdot CKL \), the processor will perform the 12 bit addition to obtain the value of \( r_{k+1} \).

5. Steps (i) to (iv) is then repeated for five cycles, and each time the value of the counter used as
an address selector for multiplexers), is incremented to select the next I/O data. The value in the counter is checked to see if all the variables have been tested.

At the end of fifth cycle the 8 bit port 'A' is programmed as an output and the output sample $y_m$ is rounded to its closest integer (3 bits), and this value is then scaled up and fed to the D/A converter via this port. The above operations are repeated for that particular filter, until the processor is interrupted via switch SW1; once pressed, the processor will select the awaiting filter coefficients, and it will continue its operation for next samples whenever the switch is depressed. The operation comes to the halt position whenever the 'ESC' key on the development system is pressed by the operator.

3.6 PHASE OF TESTING

Testing of the microprocessor based controller falls into three phases: primary, intermediate and final. The primary phase was carried out utilizing the microprocessor development system. The control software was loaded into the RAM of the SDK-85 kit by mapping the appropriate memory areas of the microprocessor development system (used for control software) into the user's area of SDK - 85. The specifi-
cations are loaded through keyboards of the development system.

The intermediate phase included brainstorming the interface module and executing the software. The system was well defined upon completion of this phase.

The final phase was performed by integrating the interface module, the processor module, and the software to carry out the testing under all possible conditions. This stage was mainly provided for debugging the boards and wiring. After this phase, a unit, meeting all the requirements specified was produced and ready to operate.
Chapter IV  
HARDWIRED LOGIC DESIGN

Due to the inherent limitation of the Microprocessor Based Structure, (a max of 600Hz sampling rate was achieved with the previous structure using the INTEL 8085*), namely the slow speed of operation of the filter, another filter structure was realized using Hardwired Logic. This structure has the advantage of having a higher speed of operation and hence a greater bandwidth than the microprocessor based structure.

Referring back to the block diagram of this structure given in Fig. 2.1, the operation of this structure can be briefly explained as follows:

At the start of each cycle, data word is selected from the multiplexers and is multiplexed with the corresponding scaled coefficient in the multiplication table. The output of the multiplication tables are added to the contents of the accumulator which is initially cleared at the start of each sample. Since there are five variables, the process has to be repeated five cycles for each sample in order to obtain an output data word. In the 6th cycle, the contents of accumulator is rounded to 3-bits via the rounding circuit, upon receiving the rounding signal. Since the coefficient data has been

Though a sampling rate of 600Hz is very low, it, however, has many biomedical and control system application.
scaled down by a factor of \( S(S=2 \text{ in the filter implemented}) \), the output data \( y(n-1) \) must also be scaled up by the same factor before being loaded into \( S5 \). For \( S=2 \) this scaling is accomplished by a hard-wired one bit left shift (skewed parallel connection). A "0" is wired to the least significant bit (LSB) input of \( S5 \).

4.1 **SECOND ORDER FILTER SPECIFICS**

The filter is constructed using a mixture of standard Transistor-Transistor-Logic (TTL) integrated circuits available from many manufacturers.

The operation of the basic component of this structure is explained below:

4.1.1 **Storage**

1. The premultiplied values of \( (rk \cdot ck\#) \) and \( (rk \cdot ck1) \) are stored in the multiplication tables utilising ROM arrays. Fig. 4.1 shows the structure in which these look-up tables have been constructed.
Fig.(4.7): Multiplication Tables
multiplication table, "MULT1" will hold the result of \( (r_k \cdot c_kH) \) and multiplication tables MULT2 and MULT3 will hold the value of \( (r_k \cdot c_kL) \) depending on the sign of coefficient \( c_k \), i.e., for negative coefficient \( c_k \), the multiplication table "MULT3" will be selected.

2. The programmable coefficients of the individual filters are scaled by a factor 'S' and stored in blocks of programmable coefficient memory whose address lines are controlled via programmable address selector. Fig.4.2 shows the hardware structure of coefficient storage. An alternative method for storing these coefficients utilising parallel-in-parallel-out (PIPO) registers shown in figure(4.3). The operation of these circuits are as follows:

- **Programmable coefficient storage using BAM:**

  The scaled coefficients of each filter are stored in blocks of coefficient memory (each block containing five consecutive coefficients for corresponding filters). Multiplexers MUX1-MUX4 determine the selection of the address lines during read and write mode. The address lines will be controlled via programmable address controller. The circuit diagram shown in figure(4-2) will operate in the following two steps:

  **Step 1 WRITE MODE**
In order to store the scaled coefficients initially, the following steps must be taken:

1. Initially the 'SELECT/F' is cleared, thus selecting the 'A' inputs lines of MUX1-MUX4 only. Also, since Q2 is at logic '0', the h/\overline{w} is at logic '0' and the coefficient memory will be in its 'Write' mode.

2. By keeping the switch 'Enable/Disable' open, the \overline{E} line of 'coefficient memory' will be at logic high thus disabling this chip. Note that the Enable/Disable pulse at input 'A' of 'AND' gate is at logic
Fig. (4.2) Programmable Coefficients Register using RAM
high initially and thus the control of changing the EE line is given to the push button switch 'Enable/Disable'. By pressing it, the coefficient memory will be disabled.

3. Set the desired address locations via "loc switch" and enter the corresponding coefficient value via data switches D₀ - D₇.

4. Enable the coefficient memory by pushing the 'Enable/Disable' switch, so that corresponding coefficients can be written into the desired location of the coefficient memory.

5. Repeat steps 2-4 for reading coefficients.

6. At the end of these operations, make sure that the 'Enable/Disable' switch is open.

step 2 read mode:

1. Once the positive edge of "ckcont" pulse arrives, the output of 'select/F' goes to logic '1' and remains at '1' for the entire operation (since D' input is tied to +5 V permanently), thus pulling the A/W line to logic '1' and keeping the coefficient memory in its "read" mode. Also, the 'SELECT' lines of MUX1 - MUX4 will be held at logic '1', thus giving the control of address lines or 'coefficient memory' to the lines coming from "programmable address selector".

2. Set the address location of the 1st coefficient (a₀) via switches SW1-SW7, shown in figure (4-11).
3. Once the "E/D coeff memory" pulse goes low, it pulls the NE line to logic '0' (since switch SW1 is open) and hence enables the coefficient memory. Thus the outputs of this memory will be the corresponding stored coefficients.
4. From now on the address lines of coefficient memory will be automatically selected via programmable address selector.

ii-b. **Programmable coefficient storage using PIPE systems:**

Five parallel-in-parallel-out registers SAR1-SAR5 shown in figure(4.3) form the storage for the scaled coefficients a₀ – b₇. The multiplexers MUX1-MUX8 isolate the input lines of SAR1 during read/write operations.

The operation of this circuit during write and read modes is as follows:-

step a. **Loading the scaled coefficients:**

1. Initially "SELECT" lines of MUX1-MUX8 are at logic '0' (since "SELECT/" is cleared initially), and hence the 'A' input lines of these multiplexers are selected. Make sure that the '5/4' switch is open.
2. Set positions of data switches D₀ – D₇ to the value of coefficient a₀.
3. Clock these registers via push button '5/4'.
4. Repeat steps 2-3 for remaining coefficients, namely a₁, a₂, ..., a₇, a₈.
Fig. 4.3 Programmable Coefficients Registers using 'PIFC'
5. At the end of 5th clocking, the coefficient \( a_0 \) will appear at the output of \( S\bar{H}5 \) and at the 'B' input lines of \( \bar{M}UX1-\bar{M}UX3 \).

**Step 2: Reading the Coefficients:**

1. Once the operation of the digital filter is started, upon receiving the first positive edge of "CKM1", the "SELECF/P" will be clocked. This will keep the "SELECT" lines of the multiplexers at logic '1' (since the 'D' input of "SELECF/P" is permanently tied to \(+5 \, V\) line), and thus, only the 'B' input lines of \( \bar{M}UX1-\bar{M}UX3 \) will be selected.

2. From now on, the \( S\bar{H}1-S\bar{H}5 \) will be clocked via \( \bar{N}7 \) line at the end of each cycle, thus outputting the corresponding stored coefficients.

**4.1.2 Arithmetic:**

The adders, accumulator and sign determination logic are shown in figure (4-4). The adders (\( F\bar{A}1-F\bar{A}3 \)) will perform the 12-bit arithmetic operation of \( (r\cdot c\bar{H}3 + 2^{-3} \cdot r\cdot c\bar{X}1) \), and the data from these adders is summed with the wired outputs of the accumulator, using three 4-bit adders, \( F\bar{A}4, 5 \) and 6 and latches 0, 9 and 10.
Fig. (4.4) Circuit Diagram of Arithmetic Section
The positive edge of the "CKACC" loads the accumulator. Five partial sums are computed using \( \text{rk.ck} \). The accumulator is cleared at the start of each sample via CKACC pulse.

At the end of the fifth cycle, upon receiving the "round" signal, the \( 2^{-9} \) th bit is added to the contents of FA5-FA6 via the rounding circuit, and the results are latched in latches 4,586. By clocking the accumulator at this stage, the rounded 8-bit output is then fed to the D/A converter and to SR5 (i.e., \( y_n-1 \) register). The 2's complement right shift to provide \( 2^{-3}(\text{rk.ckl}) \) is achieved by ANDing the MS3 of \( \text{rk.ckl} \) with logic '1'.

4.1.3 Input/Output:

To process an analog signal by digital techniques the signal must first be digitized. Although in principle there are many approaches to accomplish this, the practical method is to sample the analog signal periodically and to quantize and code each sample. The digital signal can then be processed by a digital filter.

4.1.4 Control:

Control circuitry is needed to coordinate all the filter operations to obtain valid outputs. The particular control circuitry used, is determined by the actual hardware elements chosen to satisfy the storage, arithmetic and I/C requirements of the filter. Basically, the control circuit-
try must generate clock edges of proper polarity for the sequential elements, and maintain correct logic levels for the combinational elements. These clock edges and logic levels are developed to provide the correct timing and sequence of operation as dictated by the various propagation delays, hold times, memory access time, etc., of all the filter components. The operation of the circuits which generate all the control signals are described below.

4.1.4.1 S/H, SC and CLACC pulse generator:

Fig. 4.5 shows the circuit diagram which generates these signals. The digital filter circuit will be initiated via the "start" pulse using RESET/START switch. Due to the bouncing effects of mechanical switches, the debouncing circuitry is used to eliminate this effect.

Pushing the RESET/START switch causes a rising edge triggered pulse at the input of the monostable latch '7' in its disable mode initially, thus holding the input 'A' of the AND gate at logic '1' and its inverted delayed version ('SC' pulse) is used to start the A/D converter. This signal is ANDed with 'MASTR AND' signal and is then used to clear the accumulator at the start of any sample. The next 'S/H' and 'SC' pulses (for the continuous sampling process) are produced after each consecutive 'E/D' operation of Latch '7', at the end of each sampling process.
Fig. (4.5) Block Diagram of the circuit which generates S/H, S/C and CLACC signals.
4.1.4.2 Output enable (OE) and Clock shift register (CKSR) circuits:

As can be seen from the circuit diagram of figure (4-6), the "OE" pulse is obtained using timing capacitor and resistor, $C_3$ and $R_3$, together with an AND gate upon detecting the "EOC" pulse from the A/D converter. This pulse is then used in conjunction with the +ve edge triggered monostable to trigger the 2nd -ve edge triggered monostable to obtain the desired CKSR pulse.
Fig. (4.6) Circuit which generates 'CE' and 'CKSR.' signals.
4.1.4.3 Multiplexer controller circuit (CKCONT pulse generator)

As seen from the timing diagram of figure (4.7a) and figure (4.7b), it is required to generate the 'CKCONT' pulse after every falling edge of 'CKSR' pulse and at the start of each cycle. This pulse is used to control the cycle time so that appropriate data and coefficients can be selected in each cycle. Other control circuits use this pulse as a reference along with the main system clock pulse.

The circuit which generates this pulse is shown in figure (4.8).
Fig. (4.7a) Timing Diagram of Control Signals
For One Sample Operation
Fig. (4.7b) Timing Diagram of Control Signals
Fig. (4.6) The 'CKCCNT' Signal Generator
Upon detecting the negative edge of "CKSÈ" pulse the single pulse generator produces a pulse which is "OR"ed with c/f of "END CYCLE DETECT" circuit to provide the first "CKCONT" pulse, at start of each sample. The +ve edge of "CKSÈ" pulse also clocks the CKCON/F flip flop, thus taking its output to logic high (since its D-input is tied to +5 V permanently) and allowing the system clock ("SYSCK") to start the "cycle/sample cont"; initially cleared by its RST line. From now on, the 'Cycle/Sample Cont' in conjunction with 'END CYCLE DETECT' circuitry takes control of producing the 'CKCONT' pulse. The successive 'CKCONT' pulses are then obtained till the "ROUND" signal is detected which resets the corresponding control circuit. Similar operations are then resumed for the other incoming samples.

4.1.4.4 Control signals generator (using Decoder circuitry)

The circuit diagram shown in figure(4.9) is used to enable/disable the Multiplexers(MUX1-MUX8), Multiplication Tables(MULT1-3), Latches, and to provide the appropriate signals in conjunction with the other control signals.

The "SYSCK" at the input of "NAND DECO" in the circuit, is blocked via "DECO/F/F" flip flop, until the first rising edge of "CKCONT" appears at its "CK" input. Once the positive edge of this pulse arrives, the "SYSCK" pulses take control of clocking the "DECOCONTÈ" and hence operate the decoder circuit.
The desired signals are then obtained by use of this circuit and the latching circuits.
Fig. (4.9) Control Signals Generator
At the end of each cycle, the falling edge of \( \overline{N} \) line signal resets this circuit and blocks further incoming "SYSCK" pulses. The operation of the circuit is then resumed for the next cycle upon receiving the next "CKCNT" pulse.

4.1.4.5 Round signal and CKACC pulse generator

To provide the "ROUND" signal at the end of the 5th cycle in each sample (for rounding off purpose), and to clock the accumulator during this time and at the end of each cycle, the circuit diagram of figure (4.10) is used. This circuit is also used to provide the Enable/Disable pulse for 'latch 7' so as to feed the output result to the D/A converter.
Fig. (4.10) Circuit Diagram of circuit which produces the 'ROUND' and 'CKACC' signals.
The "CYCLE CONT", which determines the end of each sampling process, is initiated via \( R_s \) and \( C_s \), and is brought to its normal operation by rising edge of \( \overline{\text{Round.RESET}} \) pulse at the start of each sample. The "CYCLE CONT" runs via the positive edge of the "delayed \( N_2 \)" line pulse, thus indicating the end of each cycle. At the end of the 5th cycle, the "MOD5" signal is brought to logic High, thus triggering the "one shot" and producing the appropriate 'round' signal. The inverted version of the "MOD5" signal is used to reset the 'MUX1-MUX8 address controller' circuit.

The falling edge of the 'round' signal triggers the second 'one shot' which is used with the \( N_4 \) line and \( \text{ANDed} \) with the logic '1' to provide the appropriate clock pulse for clocking the accumulator, and for enabling/disabling the latch '7' respectively.

4.1.4.6 Programmable address selector

To extract the appropriate coefficient from the memory in each cycle, the circuit diagram shown in figure (4.11) is used.

The starting address of each block of coefficient memory is referenced by the positions of "loc switches" \( S_1-S_{18} \) during the read mode. The "COEFFCONT" circuitry which controls the address lines of the coefficient memory, is preset to its input values (i.e., starting address of each block) at the start of the first cycle in each sample, and
then brought back to its counting mode in the consecutive cycles upon receiving the negative edge of the delayed $N$, line pulse.
Fig. (4.11) Programmable Address Selector
The output states of "COEFFCONT" in each cycle, points to the corresponding location of the coefficients in the coefficient memory. The reset line is used to bring the counter to its preset mode at the start of the first sample. For the next sample operation, the positive edge of the 'round' signal is used for this purpose.

4.2 A/D AND D/A CONVERSIONS

4.2.1 Analog to Digital converters

A/D converters handle the continuous signals produced by various types of sensors and transducers. There are many methods of performing an analogue to digital conversion. Four common types of A/D's are as follows:-

4.2.1.1 successive approximation converters

Successive approximation A/D conversion progressively resolve the input voltage bit by bit to the required bit resolution. The MSB of an internal DAC is first set to '1' and the output of the DAC is compared to the analogue input. If it is greater than the input the MSB is reset to '0', otherwise it is left at '1'. This procedure is repeated for every bit down to the LSB, and the final input code to the DAC is the output code of the ADC.
4.2.1.2 Parallel (Flash) converters

Parallel converters are very fast and may perform in excess of 10 million conversions per second. However, due to the large number of comparators required (255 for an 8-bit converter) they are expensive to produce.

Most of these converters work in the range of 1-20 MHz. But at these speeds, resolution can be a problem. Fortunately 8 bits and lower most often suffice in ultrafast A/D conversions. Such a converter is essential to diverse applications like video digitizing, radar signal processing, high-speed data conversions and storage oscilloscope.

The A/D converter is constructed using the ZN427E. The 8-bit ZN427 features fast, 15 micro seconds conversion time, and tri-state outputs to permit busing on common data lines with no missing codes over the full operating and temperature range. The logic diagram of ZN427E is shown in figure (4.12).

The conversion is started by 'START CONVERT' (SC) negative going pulse, it is applied synchronously with respect to the ZN427 clock. The negative going edge of 'SC' pulse must become active at least 1.5 micro seconds before the first active negative going edge of the clock to allow for most significant bit (MSB) setting. Also the trailing edge of the 'SC' pulse, has to occur outside the limit + 200 nano seconds of any negative going edge of the clock. Thus, in order to meet the timing requirements explained above, the
D-type flip flop is used in conjunction with the SC pulse (ref. to Appx-D).
Conversion takes nine clock periods. An 'END OF CONVERSION'
EOC pulse (Active high) is provided when conversion is
completed.

The three-state data outputs are 'off' (high impedance)
when 'OUTPUT ENABLE' (OE) is an '0' and are enabled when the
(OE) input is taken to logic '1'. A/D is clocked by clock 1
generator which generates a clock signal of 800 KHz.

4.2.2 Digital to Analogue conversion

The digital to analogue converter (D/A) is based upon
the Ferranti ZN425E, and the circuit diagram is shown in
figure (4.13). The input is a offset binary code which has
a settling time of 1 micro seconds. The Two's Complement
value of scaled \( V_n \), is converted back to offset binary code
by complementing its MSB, before feeding it to D/A conver-
ter.

A buffer amplifier is needed in order to remove the
offset voltage and to calibrate the converter. The 741 oper-
ational amplifier in figure (4.13) acts as this buffer ampli-
fier.

4.3 EXTERNAL CIRCUIT FOR BASIC OPERATION OF A/D

The external components for the basic operation of
ZN427 A/D is shown in figure (4.14). A resistor \( R_{ref} \) is
connected between pins 3 and 10 to supply a nominal refer-
ence current of about 6.4 mA i.e.:

For \( R_{ref} = 390 \), Current = \( \frac{(V_{cc} - V_{ref})}{R_{ref}} \)

= \( \frac{(5-2.5)}{390} \)

= 6.4 mA
Capacitor $C_{ref} = 1$ microfarads is connected across pins 8 and 9 for stability/decoupling purpose.

The nominal value of $I_{ext}$ is 65 microamps and suitable value for $R$ is given by:

$$R_{ext} = \frac{1}{V} \times 15K$$

so, choosing $V$ to be $-5V$ then $R_{ext} = 82K$. For optimum stability with temperature, the analogue input ($A_{in}$) should be applied through a source resistance of $R_{in} = 4K$ to match the ladder network.

Since for $A_{in} = -5V$ the required value of $V_{in}$ is $0V$ and since for $A_{in} = +5V$, the required value of $V_{in}$ is $2.5V$. So component values shown in Figure(5) were chosen to match the ladder resistance.

$$R_1 \parallel R_2 \parallel R_3 = A_{in} = 4K.$$ 

Since outputs of this A/D is offset binary code so, in order to change this digital data to its 2's complement form, only the MS3 or output of A/D is inverted to achieve this.
Chapter V
FILTER OPERATION AND TESTING

The filter structure discussed was implemented in software (refer to Appendix B, for listing of the program) and in hardware.

A Second-Order Low-Pass Butterworth Filter with a cut-off frequency of 100 Hz, was designed and used for this purpose. The corresponding difference equation is given by:

\[ y(n) = -0.019804x(n) + 0.039608x(n-1) + 0.019804x(n-2) 
- 1.564376y(n-1) + 0.6435943y(n-2) \quad (5.1) \]

5.1 RESULTS OBTAINED FROM SOFTWARE SIMULATION:

Figure (5.1-a), depicts the frequency response of the filter.
Fig. 5.1.a Frequency response of the filter obtained using the IBM-370.
The frequency response of the filter is obtained by feeding sinusoidal signals of frequencies ranging from 0 Hz to 2.5 kHz to the simulated software structure.

Fig. 5.1.b to 5.1.d show the time domain outputs of the filter, when a sine wave is fed as an input to the low-pass filter.

A study of these figures reveals that the output waveform of the filter resembles the sinusoidal input to the filter.
Fig. 5.1.b  **Software Simulated Time-Domain Response of the Filter.** (Passband Region)

Fig. 5.1.c  **Software Simulated Time-Domain Response of the Filter.** (Transition Region)
Fig. 5.1.d  Software Simulated Time-Domain Response of the Filter Structure. (Stopband Region)
5.2 RESULTS OBTAINED FROM THE HARDWARE REALIZATIONS:

5.2.1 Microprocessor Based Structure:

A ±5v test sine wave of frequency ranging from 10 Hz to 300 Hz, is fed to the filter. Figure 5.2.1 represents the photographs of the actual input/output waveforms of the filter for three test frequencies:

1. F=20 Hz (passband region)
2. F=30 Hz (cut-off frequency)
3. F=200 Hz (stopband region)
Fig. 5.2.1.a  Actual Input/Output waveforms of the Microprocessor-based Structure at f=20 Hz. (Passband Region)

Fig. 5.2.1.b  Actual Input/Output waveforms of the Microprocessor-based Structure at f=30 Hz. (Cutoff)
Fig. 5.2.1.c  Actual Input/Output waveforms of the
Micronprocessor-based Structure at f= 200 Hz.
(Stopband Region)
Fig. 5.2.2 represents the frequency response of the filter realized by this structure.
Figure 5.2.2: Frequency Response of Microprocessor-based Digital Filter
5.2.2 **Hardwired Logic Design Structure**

A 55v test sine wave of frequency ranging from 0 to 10 KHz is fed into the filter realized by this structure. Fig. 5.2.3 represents the photograph of the actual input/output waveforms of the filter for three frequencies:

1. \( f=1 \text{ KHz} \) (passband region)
2. \( f=5 \text{ KHz} \) (cut-off frequency)
3. \( f=8 \text{ KHz} \) (stopband region)

Fig. 5.2.4 represents the frequency response of the filter obtained by plotting the amplitude variation of the output waveforms versus the test frequency ranges.

Fig. 5.2.5 and 5.2.6 show the required control pulses observed on the Logic Analyzer. The obtained pulses resets the required control pulses shown in figures 4.7a & 4.7b. The photographs of the implemented structures are shown in figures 5.2.7 and 5.2.8.
Fig. 5.2.3.a  Actual Input/Output waveforms of the Hardwired-Logic designed Filter at f=1 KHz. (Passband Region)

Fig. 5.2.3.b  Actual Input/Output Waveforms of the Hardwired-Logic designed Filter at f=3 KHz. (Cutoff Region)
Fig. 5.2.3.c  Actual Input/Output Waveforms of the Hardwired-Logic Designed Filter at f=5 KHz. (Stopband Region)

______________________________________________________________

The V/cm knob of the output waveform is selected lower than the input. This was necessary in order to synchronize the two waveforms together so as to obtain a stable display.
FIG 124: Frequency Response of Hardwired-Logic based Digital Filter
Fig. 5.2.5a: Control Pulses Obtained For A Single Cycle (One Sample) Resembling Required Pulses Of Fig. 4.7a

Fig. 5.2.5b: Control Pulses Obtained For Three Samples Resembling Required Pulses Of Fig. 4.7a
Fig. 5.5.5.a Control Pulses obtained for One Cycle Operation resembling Required Pulses of Fig. 4.7.b

Fig. 5.5.5.b Control Pulses obtained for Five Cycles (one sample) resembling Required Pulses of Fig. 4.7.b
Fig. 5.2.7  Photograph of the Implemented Hardwired-Logic Designed Filter.

Fig. 5.2.8  Photograph of the Implemented Microprocessor Based Filter.
Chapter VI

ROUNDING EFFECTS

The errors caused by finite word length and their influence on the behavior of digital filters are investigated in this chapter.

Using an eight bit word length, the designer enters the fields of finite word length arithmetic and sampled signals. During the implementation various, arithmetic phenomena, such as, limit cycle, will be encountered.

6.1 Rounding of Binary Numbers

Properly scaling of data, avoids overflow during A/D conversion and during arithmetic operations. However, there is a limit on the resolution because the width of the resolution is the value of the least significant bit \(2^{-C}\), where the word length chosen to represent a number in binary form is \(2^{C+1}\).

Rounding-off binary numbers to 3-bits is accomplished by choosing the number in the 3-bit closest to the unrounded quantity.
For fixed point arithmetic, the error made by rounding is the same for all three types of number representations.

The error is:

$$\varepsilon_R = Q_R [x] - x$$  \hspace{1cm} (6.1)

$$\frac{-2^{-B}}{2} \leq \varepsilon_R \leq \frac{2^{-B}}{2}$$  \hspace{1cm} (6.2)

For an 8-bit number (B=8),

$$-2^{-9} \leq \varepsilon_R \leq 2^{-9}$$  \hspace{1cm} (6.3)

As explained in Chapter 1, there are three numerical sources of errors: errors in A/D converters, multiplication errors, and coefficients storage errors.

1. Input quantisation: Quantisation of the input signal takes place in the A/D converter. Here the input analog signal is sampled periodically and each sample is coded into a digital word of N-bits. For a full dynamic range of -1 to +1, the step size q is $2^{-B+1}$ and

$$x[n] = \tilde{x}_n + \varepsilon_n$$

where $\varepsilon_n$ is the error due to quantisation. It has a zero mean and its mean squared value (variance) is given by:

$$\text{Var}(\varepsilon) = q^2 / 12 = 2^{-2B} / 3$$

since $q = 2^{-B+1}$. Therefore it can be seen that the actual input to the digital filter is $\tilde{x}_n$ rather than $x[n].$

2. Coefficient quantisation: For 8-bit representation of the coefficients the actual coefficients are given by:

$$[a_k]_3 = a_k + \alpha_k$$  \hspace{1cm} (6.4a)

$$[b_k]_3 = b_k + \beta_k$$  \hspace{1cm} (6.4b)
where \( \alpha \) and \( \beta \) are the error due to quantizing. The actual transfer function of the filter is thus given by (6.5),

\[
[h(z)]_B = \frac{\sum_{k=0}^{N} [a_k]_B z^{-k}}{1 + \sum_{k=1}^{M} [b_k]_B z^{-k}}
\]

6.2 **PROPAGATION OF THE QUANTIZATION NOISE THROUGH THE SYSTEM**

The propagation of the quantization noise, \( \varepsilon \), depends on the transfer function between the source point \( \varepsilon \) (\( \varepsilon_a \) or \( \varepsilon_m \)) and between the output. Fig. 6.1 shows the numerical errors for first-order system. Given the statistics of the input noise (i.e., mean and var of \( \varepsilon_a \) and \( \varepsilon_m \)), the statistic of the noise in the output (i.e., \( \varepsilon_u \)) is of interest.
For a stable system, characterized by its transfer function $H(z)$, and for an input signal $\varepsilon(i)$, characterized by its mean $\bar{\varepsilon}$, and its variance $\sigma^2$, the mean of the output signal $\bar{u}$ is given by:

$$\bar{u}(z) = H(z) \cdot \bar{\varepsilon}(z)$$  \hspace{1cm} \text{(6.4)}

and since $\bar{\varepsilon} = \text{constant}$, thus:

$$\bar{\mu} = \bar{\varepsilon} \lim_{z \to 1} H(z)$$  \hspace{1cm} \text{(6.5)}

Consider:

$$\frac{U(s)}{E(s)} = \frac{1}{S + b} \frac{U(z)}{E(z)} = \frac{1}{1 - \beta z^{-1}}$$

where $\beta = e^{-2\pi T}$, and noise propagation for this first-order system, using Parseval theorem [3] is:

$$\sigma^2_{\mu} = \sigma^2_{\varepsilon} \left( \frac{1}{1 - \beta^2} \right)$$  \hspace{1cm} \text{(6.7)}

Assuming $b=1$ and $T \ll 1$, then, $\beta = 1-b T$, i.e.

<table>
<thead>
<tr>
<th>T</th>
<th>$\beta$</th>
<th>$\left( \frac{1}{1 - \beta^2} \right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.100</td>
<td>0.900</td>
<td>5</td>
</tr>
<tr>
<td>0.010</td>
<td>0.990</td>
<td>50</td>
</tr>
<tr>
<td>0.001</td>
<td>0.999</td>
<td>500</td>
</tr>
</tbody>
</table>

Thus it can be seen that due to fast sampling rate the difference between two successive numbers is small and only the least significant bits are changing.
The propagation and amplification of the quantization noise depends on the structure of the algorithm. There is no simple theory to estimate the noise which will be generated for a particular transfer function as realized in a particular structure. Some conclusions which may be derived from quantization errors are:

1. Fast varying roundoff errors ($E_{n}$, $E_{m}$) cause noisy outputs but do not influence stability.
2. Changes in coefficients influence dynamics and stability.
3. The influence of $E_{a}$ on the output does not depend on the system structure; it depends on the transfer function only.
4. The generation of multiplication errors and their propagation depend on the realization.

6.3 LIMIT CYCLES:

Another source of error is slowly varying quantization error. Essentially, a recursive filter with finite word length arithmetic, generates non-linear effects such as limit cycles [3]. This occurs due to the overflow of the dynamic range of the filter.

For special types of inputs, the roundoff noise is highly dependent upon the signal, making it necessary to adopt a different approach to analyze the effects. To de-
 demonstrate this, consider a first-order filter whose equation given by:

\[ y_n = x_n - 0.5 y_{n-1} \]  \hspace{1cm} (6.8)

with \( x = 0 \) (i.e. no input applied to the system) and \( y = 0.5 \).

For 3-bits (\( m=3 \)) plus a sign bit, the output \( \{ Y, Y, Y, \ldots \} \)
becomes in this case, \( \{ 0.5, -0.25, 0.125, -0.125, 0.125, \ldots \} \), an oscillating sequence. As seen from the table 6.1, it oscillates between 0.125 and -0.125. So is the sequence of roundoff errors \( \{ \varepsilon \} \). Also, for 4-bit data register (\( m=4 \)) plus a sign bit, the output oscillates between 0.0625 and -0.0625 as seen from the table 6.1. Thus, if \( m \)-bits plus a sign bit are used, the output would oscillate between \( (0.5)^m \) and \( -(0.5)^m \).

Therefore, the oscillation can be reduced by using more bits.
<table>
<thead>
<tr>
<th>i</th>
<th>Xn</th>
<th>Yn</th>
<th>2's Compl.</th>
<th>Rounded value</th>
<th>Rounded value cf.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of Yn (Y'n1)</td>
<td>Yn (Y'n2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(for m=3 plus)</td>
<td>4 plus sign bit</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>+0.5</td>
<td>0.1000000</td>
<td>0.100=0.5</td>
<td>0.100=0.5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-0.25</td>
<td>1.1100000</td>
<td>1.110=-0.25</td>
<td>1.110=-0.25</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>+0.125</td>
<td>0.0010000</td>
<td>0.001=0.125</td>
<td>0.001=0.125</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-0.0625</td>
<td>1.1110000</td>
<td>1.111=-0.125</td>
<td>1.111=-0.0625</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>+0.03125</td>
<td>0.0001000</td>
<td>0.001=0.125</td>
<td>0.001=0.0625</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-0.015625</td>
<td>1.1111000</td>
<td>1.111=-0.125</td>
<td>1.111=-0.0625</td>
</tr>
</tbody>
</table>

Table 6.1
This type of non-zero output in the absence of input signal is called zero input limit cycle. It is therefore important to use enough bits in data registers so that these oscillations, when they occur, are kept small enough [16]. Limit cycle oscillation can be eliminated by one of three ways:

1. Properly scaling the filter so that overflow is absolutely impossible.

2. Modifying the adders.

3. Changing the structure of the filter section.
Chapter VII

CONCLUSION AND COMMENTS

7.1 CONCLUSION

An approach to the implementation of Programmable Coefficients Digital Filter is described in this thesis. By using Parallel and Serial Processing, Fixed-Point Arithmetic (2's complement notation) and Look-Up tables utilizing BCM arrays, the hardware complexity of the structure is greatly minimized.

The Microprocessor Implementation has demonstrated its ease and flexibility in the filter design. This structure is flexible in a sense that the complexity of the control circuitry is reduced using software routines. By simply changing the software, the timing waveforms can be modified to achieve any desired control signal.

In its present form, the maximum sampling frequency of the Microprocessor Based Structure is 600 Hz, restricting its application to filtering of biomedical, control system, or other low-frequency signals.

The second structure implemented, namely via Hardwired Logic, is slightly more complex in terms of designing of hardware circuitry, but has the advantage of a higher throughput rate than the Microprocessor version.
The processing rate in the Hardwired Logic Design is limited only by the speed of the basic digital circuits speed \(=1/(5*tp)\), \(tp\) being the propagation delay of the IC's. However, due to the limitation of the A/D converter \(\text{used}\), and hand wiring, this structure could be tested only up to 20kHz.

In order to program the EPROM's, which are used for the Multiplication Tables, a transmission routine between the NCMVA-Minicomputer, and the EPROM-Programmer (model 811), was developed. (ref. APPX-A).

7.2 Comments

Parallel Processing can extend the speed of operation of the proposed digital filter. By the use of multiple processors, the signal processing can be distributed among several microprocessors which results in the capability of handling incoming signal of much higher frequencies.

Possible modification to the microprocessor structure might include use of a faster processor: Bit slice Microprocessors using bipolar technology are approaching execution speeds in the nano-second region.

Wire-wrapping technique and use of faster A/D converter and other high speed components of the control circuitries could raise the throughput rate of the Hardwired Logic structure to \(MHz\).
APPENDIX A

Interface between NOVA minicomputer and EPROM programmer (811).

In order to generate the multiplication tables (MULT1 - MULT3) the EPROMS (2732) used for this purpose, had to be programmed on an EPROM PROGRAMMER (model 811), using NOVA minicomputer. Since the serial interfacing was not available, therefore the software package was developed to link the NOVA minicomputer and the EPROM programmer together so that the data generated in the NOVA minicomputer could be transferred to the EPROM programmer and hence to the EPROMS.

The serial format in which the 'Data record' (i.e., address bytes, data bytes etc.,) and the 'End record' had to be transmitted is shown in figure A-1(a-b).

Start character (IFILD(1)): 3A: (colon - ASCII coded).

Byte count (IFILD(4)): Number of data bytes in a record (Maximum of 128 data bytes can be transmitted in a record). Each byte is two ASCII coded hexadecimal characters.

Load Address (IFILD(5) and IFILD(6)): Location to put first data byte in a record. A3 is arbitrary for model 811.

IFILD(5) - Higher byte of address.

IFILD(6) - Lower byte of address.
Fig. A (a) 'DATA RECORD' FORMAT

(b) 'END RECORD' FORMAT

Model 811 ignores address in end record when transmitting

the address 0000 is sent giving a check sum of FF₁₆
Check sum: Two's complement of 8-bit sum of 8 bytes of check sum field.

Control: Model 811 transmits: CR; NULL; LF; 5 NULLs.

Thus: \text{ICONT}(1) = \text{CR}(\text{'13'})

\text{ICONT}(2) - \text{ICONT}(8) = 7 \text{ NULLs} (\text{LF was transmitted as a NULL}).
Model 811 ignores address in end record. When transmitting, the address 0000 is sent giving a check sum of FF, a.
These characters have to be converted to their equivalent ASCII characters before being transmitted. The NOVA minicomputer uses 2 bytes for storing integer values, thus each byte of these records had to be transferred to their MS byte position before being transmitted, since only one byte (i.e., MS byte) can be transmitted in each transmission process. To match the speed of the transmitter (in this case NOVA minicomputer) and receiver (Model 811), the delay routine was introduced.

The program listed below achieves all the transmission requirements and proved to work successfully. This program is commented and is self explanatory.

INTERFACE CONFIGURATION:

The interface switch is accessed by removing the clear polycarbonate cover on the left hand side of the Model 811. The up, or "off", position of the switches correspond to logic '1'.

Three parameters must be established to permit the Model 811 to communicate. Switches 1, 2, 3 & 4 establish the data rates. Since NOVA minicomputer operates on BAUD rate of 9600, therefore the model 811 had to be set to that BAUD rate. This was achieved by setting the switch 1 to logic '0' (down position) and switches 2, 3 and 4 to logic '1' (up position).
Switches 5 & 6 set the word format. For 8 bit data these are set to logic '0'. Switches 7, 8, 9, and 10 establish the specific protocol. For INTEL, they are set to logic '0', '1', '0' and '0' respectively.

Note that to give the command of transmitting operation, the TEXTRONIX graphic terminal acted as a console and for this reason the two switches 40A and 40B on the NOVA minicomputer had to be flipped over, and RS232 cable had to be connected to model 811 rather than to the NOVA terminal.

**OPERATING PROCEDURE:**

After loading the EPROM programmer and after creating the desired data file, press 'load' and 'SER' keys in the EPROM programmer. Type in the input file name in the console, and then press 'CR'. Type in data file name followed by 'CR'. The data will then be transmitted to the EPROM programmer and once the transmission is ended, the 'VBR' light will be illuminated to verify the correct transmission. The "end of transmission" message will appear on the console screen followed by "stop" and "a".

To verify for the correct data entry, press 'CLEAR', 'KEY' and 'SET' keys, then press the required address keys on the keyboard or the model 811. The required data will be appeared on the right hand portion of the display (in HEX form). Place the EPROM on the circuit provided and then lock it by pressing down the handle on the socket. Set the starting and the ending address by pressing the "start" and
"end" keys, and by entering the corresponding address locations via keyboard. The 'start' and 'end' address will be displayed on the display. Press "clear", 'key' and 'prog' keys in order to dump the data from the memory of the EPROM programmer to the EPROM. Wait until the verify("VER") light comes on, this ensures the end of correct programming operation. Otherwise the 'FAIL' will appear on the display. To verify that the data have been correctly entered into the EPROM, press, "LOAD" key. Once again the 'VER' light will come on to verify the correct operation.
C***** FILE NAME:- NIK
C***** THIS PROGRAM TRANSMITS THE DATA FROM NOVA COMPUTER
C***** TO THE EPROM PROGRAMMER.
C***** IFILE: INPUT FILE NAME
C***** IDATA: INPUT DATA
C***** IFLD: CONTROL WORD FIELD
C***** IADRSS: TWO BYTE ADDRESS
C***** KCONT: BYTE COUNT
C***** NDATA: NO. OF BYTE TO BE SENT TO EPROM
C***** ICHECK: FLAG TO DETERMINE THE END OF LOCATION
C***** IF IFLAG=1, THEN END OF DATA.
C*****************************************************************
INTEGER IFILE(5), IDATA(256), IFLD(6), ICONT(8)
OPEN 0, "$TI1"
OPEN 1, "$T01"
OPEN 10, "$T02"
TYPE "ENTER INPUT FILE NAME"
READ(11,1) (IFILE(I),I=1,5)
1 FORMAT(5A2)
OPEN 2, IFILE
IADRSS=0
C TO READ 256 BYTE OF DATA
NDATA=128
S KCONT=0
ICHECK=0
DO 2 I=1,NDATA
READ(2,END=7) IDATA(I)
GO TO 5
2 ICHECK=1
GO TO 3
5 KCONT=KCONT+1
CONTINUE
C TO CHECK THE PRESENCE OF DATA
S IF(KCONT.EQ.0) GO TO 80
C GENERATE THE ASCII CODE OF 1ST FIELD, AND
C SEND IT TO EPROM PROGRAMMER
C IP=":`
BYTE(I9,1)=BYTE(IP,1)
CALL WRSEO(1,1,1,IER)
CALL DELAY
C TO GENERATE CORRESPONDING DATA BYTES OF IFLD(2) TO IFLD(5):
C IFLD(2)=BYTE(KCONT,2)
IFILD(3)=BYTE(IADRSS,1)
IFILD(4)=BYTE(IADRSS,2)
ZERO=0
IFILD(5)=BYTE(ZERO,2)
C TO GENERATE ASCII CODES OF THESE FILELD'S USING (SUB-CONVRT),
AND TO TRANSMIT THEM USING (SUN-WRSEQ):

DO 20 I=2,5
CALL CONVRT(IFILD(I),I1,I2)
CALL WRSEQ(1,I1,1,IER)
CALL DELAY
CALL WRSEQ(1,I2,1,IER)
CALL DELAY
CONTINUE

TO CONVERT IDATA(I) TO THEIR ASCII CODE AND TO TRANSMIT THEM TO EPROM PROGRAMMER.

DO 30 I=1,KCONT
IP=BYTE(IDATA(I),2)
CALL CONVRT(IP,I1,I2)
CALL WRSEQ(1,I1,1,IER)
CALL DELAY
CALL WRSEQ(1,I2,1,IER)
CALL DELAY
CONTINUE

TO GENERATE THE 'CHECK-SUM':

IS=KCONT+IFILD(3)+IFILD(4)
DO 40 I=1,KCONT
IS=IS+IDATA(I)
CONTINUE
IS1=MOD(IS,256)

TO FIND ITS 2'S COMPLEMENT:

IVALE=65535
ICHKS=IEOR(IS1,IVALE)
ICHKS=ICHKS+1

THE CHECK-SUM

ICHKSM=BYTE(ICHKS,2)

CONVERT IT TO ITS ASCII CODE:

CALL CONVRT(ICHKSM,I1,1)
CALL WRSEQ(1,I1,1,IER)
CALL DELAY
CALL WRSEQ(1,I2,1,IER)
CALL DELAY

TO GENERATE ASCII CODE OF 'CR', 'LF', 'NULL':
N.B. SEND 0'S FOR 'LF' & 'NULL'

ICR=13
BYTE(IX+1)=BYTE(ICR,2)
CALL WRSEQ(1,IX,1,IER)
CALL DELAY
DO 60 I=2,8
   ICONT(I)=0
DO 70 I=2,8
   IP=BYTE(ICONT(I),2)
   CALL CONVRT(IP,II,12)
   CALL WRSEQ(1,II,1,IER)
   CALL DELAY
   CALL WRSEQ(1,12,1,IER)
   CALL DELAY
70 CONTINUE
C C TO UPDATE THE ADDRESS LOCATION FOR NEXT DATA TRANSFER.
C IADRSS=IADRSS+KCONT
IF(ICHECK.EQ.1) GO TO 80
   GO TO 9
C C TO SEND THE END OF TRANSMISSION CODE
C 80 IP=":"
   BYTE(IX,1)=BYTE(IP,1)
   CALL WRSEQ(1,IX,1,IER)
   CALL DELAY
   DO 90 J=2,4
90 IFILD(J)=BYTE(ZERO,2)
   IONE=1
   IFILD(5)=BYTE(IONE,2)
   ICHKS=255
   IFILD(6)=BYTE(ICHKS,2)
   DO 100 I=2,6
100 CALL CONVRT(IFILD(I),II,12)
   CALL WRSEQ(1,II,1,IER)
   CALL DELAY
   CALL WRSEQ(1,12,1,IER)
   CALL DELAY
   CONTINUE
   TYPE "END OF TRANSMISSION"
   CLOSE 0
   CLOSE 1
   CLOSE 10
   STOP
END
FILE NAME: CONVRT

THIS SUBROUTINE CONVERTS THE GIVEN ONE BYTE INTEGER INTO TWO ASCII CODES.

INPUT VARIABLE IS 'IP' AND OUTPUT VARIABLES ARE 'I1' AND 'I2'.

SUBROUTINE CONVRT(IP,I1,I2)
INTEGER IP,I1,I2
IP1=IP
IT=IP1/16
IF(IT.GT.9) IX=IT+55
IF(IT.LE.9) IX=IT+48
IP1=IP1-(IT*16)
IF(IP1.GT.9) IY=IP1+55
IF(IP1.LE.9) IY=IP1+48
BYTE(I1,1)=IX
BYTE(I2,1)=IY
RETURN
END
SUBROUTINE DELAY
  I=100
111  I=I-1
  IF (I.EQ.0) GO TO 222
    GO TO 111
222  RETURN
END
APPENDIX - B

LISTING OF SOFTWARE SIMULATION (USING IBM-370)
**THIS PROGRAM SHOWS THE TIME AND FREQUENCY RESPONSES**
**OF A GIVEN FILTER TRANSFER FUNCTION CORRESPONDING**
**TO A SINE WAVE TEST INPUT SIGNAL.**
**
**
**ABBREVIATIONS USED ARE AS FOLLOWS:**
**
**
**TYPFLT = TYPE OF FILTER TRANSFER FUNCTION**
**= 0 FOR LOW PASS FILTER**
**= 1 FOR HIGH PASS FILTER**
**PL = LOWER CUTOFF FREQUENCY**
**PH = HIGHER CUTOFF FREQUENCY**
**DCMAIN = 0 FOR TIME DOMAIN RESPONSE**
**= 1 FOR FREQUENCY DOMAIN RESPONSE**
**REGION = 0 FOR PASSBAND REGION INFORMATION**
**= 1 FOR TRANSITION REGION INFORMATION**
**= 2 FOR STOPBAND REGION INFORMATION**
**
**
**************************************************************************************

INTEGER TYPFLT, DOMAIN, REGION
INTEGER DR, ADR, PDE, APDE, XLMS, PXLMS, PXLM1, CLM, PCLM, PCLM1
INTEGER ALT1, ALT2, B1, CA, SF, SC, FPTS, TFP
INTEGER X0(8), X1(8), X2(8), X1(8), T2(8), XOUT(8), IFX(100),
      SX(100, 8), SY(100, 8), MUX(5, 8), PIPO(5, 8), CFB(8), IFYPE(100)
REAL*8 XLSB, CLSB1, CLSB2, OMAP1, OMAP2, OMAP3, OPADR, SR
REAL*8 XMT1(4096), XMT2(4096), XMT3(4096)
DIMENSION PFX(100), YP(100), T1(100), F(100), YFP(100), COEF(5)
DIMENSION RF(100)
LOGICAL*1 B(8)
COMMON/B1/ XLSB, CLSB1, CLSB2, XLMS, PXLMS, PXLM1, CLM, PCLM, PCLM1
DATA E/ .FALSE., .FALSE., .FALSE., .FALSE., .FALSE., .FALSE., .FALSE., .FALSE.,
      1. TRUE. /

READ, TYPFLT
READ, PL, PH
READ, FPASS, FTRAN, FSTOP
READ, DOMAIN
READ, REGION
READ, SF

THP=0.00002
PYE=2Z.0/7.0
TFP=1C0
T=0.0002
IDONE=0
IF(TYPFLT .EQ. 0) GO TO 11
NF=1/(2.*THP)

11  NF=1/(2.*T)
DE=8
IDR=DE
ADR=7
PDR=4
APDB1=3
APDE=3
XLM=256
CLM=16
PXLM=128
PXLM7=129
PCLM=8
PCLM7=9
XLSE=2.* ([-7])
CLSE1=2.* (-3)
CLSE2=2.* (-4)
LMT=4096
NS=100

C

IF (TYEFLT .EQ. 0) GO TO 111

C

HIGH-PASS FILTER COEFFICIENTS GENERATION :-

C

W0=2.*PYF*FH
VAR=2.826
CCNST=4.0/(THP**2)
BB0=(W0**2)+VAR*W0/THP +CONST
BB1=(2.*(W0**2)-2.*CONST)/BB0
COEF(4)=BB1
BB2=(W0**2)-VAR*W0/THP +CONST)/BB0
COEF(5)=BB2

C

A0=CCNST/BB0
COEF(1)=A0
A1=2.*CONST/BB0
COEF(2)=A1
A2=A0
COEF(3)=A2

C

PRINT 222

222 FORMAT('1',4X,'TYPE OF FILTER :-HIGH PASS FILTER')
GO TO 333

C

LOW-PASS FILTER COEFFICIENTS GENERATION :-

C

111

W0=2.*PYE*FL
VAR=2.826
CCNST=4.0/(T**2)
BB0=(W0**2)+VAR*W0/T +CONST
BB1=(2.*(W0**2)-2.*CONST)/BB0
COEF(4)=BB1
BB2=(W0**2)-VAR*W0/T +CCNST)/BB0
C

A0 = \( \frac{w0**2}{bb0} \)
C
C

A1 = + \( 2.0 \cdot (w0**2) \)/bb0
C
C

A2 = a0
C
C

C0EF(3) = A2
C
C

PRINT 41
C

FORMAT(\'1\',4X,\'TYPE OF FILTER : - LOW PASS FILTER\')
C
C

TO SCALE THE COEFFICIENTS :
C
C

333 DO 1 I=1,5
C
CNF=C0EF(I)/SF
C
C

TO CHANGE THE COEFFICIENTS TO THEIR 2'S COMP. FORM :
C
C

CALL DECTCP(CNF,CFB,B)
C
C

TO STORE THE SCALED COEFFS IN THE 'PIPO' REGS :
C
C

DO 2 J=1,DR
C
2 PIPO(I,J)=CFB(J)
1 C CONTINUE
C

PRINT 34
C

FORMAT(\'11X, \', 'COEF(I) IN\\',7X, \', 'COEF(I) IN\\',/.10X, \\
1\'DEC. FRACTION\\',5X, \', '2-S COMP. FORM\\',//
C
DO 32 I=1,5
C
32 PRINT 33; COEF(I), (PIPO(I,J),J=1,DR)
33 FORMAT(10X,F10.7,4X,8(I1,1X))
C

PRINT 40, SF, DR
C

FORMAT(4X,\'SCALE FACTOR OF THE FILTER COEF. = \', I2, //, 4X, \\
1\'DYNAMIC RANGE OF THE FILTER = \', I2, //)
C

SC=0
C
LT=1
C
CALL XMTELE(LMT,XMT1,LT,SC)
C
LT=0
C
CALL XMTELE(LMT,XMT2,LT,SC)
C
SC=1
C
CALL XMTELE(LMT,XMT3,LT,SC)
C
C
IF 'DCMAIN=0' THEN THE TIME DOMAIN RESPONSES OF
C
REQUIRED REGIONS (I.E. PASSBAND, TRANSITION OR
C
STOPBAND REGION) WILL BE PLOTTED. OTHERWISE,
C
THE FREQUENCY RESPONSE WILL BE PLOTTED.
C

IF(DCMAIN=EQ. 1) GO TO 444
FPTS=1
YMAX=0.
IF(REGION .EQ. 2) GO TO 555
IF(REGION .EQ. 1) GO TO 666
RF(FFTS)=FPASS/NF
GO TO 777
666 RF(FFTS)=FTRAN/NF
GO TO 777
555 RF(FFTS)=FSTOP/NF
777 DO 888 K=1,NS
   T1(K)=K
   XF(K)=SIN(PYE*RF(FFTS)*K)
P1=IF(K)
C
C TO CHANGE THE INPUT DATA TO ITS 2'S COMP FORM :
C
CALL LECTCP(P1,XOUT,B)
DO 885 I=1,DR
   SX(K,I)=XOUT(I)
888 CONTINUE
IDONE=1
C
444 DO 9 FFTS=1,TFP
   IF(IDONE .EQ. 1) GO TO 999
C
   FFTS=1
   YMAX=C.
   IF(TYFLT .EQ. 0) GO TO 445
   F(FFTS)=(FFTS*10.)*25.
   GO TO 446
445 F(FFTS)=FFTS*25.
446 DC 3 K=1,NS
   T1(K)=K*T
   XF(K)=SIN(2.*PYE*F(FFTS)*T1(K))
P1=IF(K)
   CALL LECTCP(P1,XOUT,B)
   DO 4 I=1,DR
   4 SX(K,I)=XOUT(I)
   3 CONTINUE
999 DO 5 I=1,DR
   SY(1,I)=SX(1,I)
5 SY(2,I)=SX(2,I)
   YP(1)=XF(1)
   YP(2)=XF(2)
   RC=3
   DC 6 I=1,DR
   X0(I)=SX(3,I)
   X1(I)=SX(2,I)
   X2(I)=SX(1,I)
   Y1(I)=SY(2,I)
   Y2(I)=SY(1,I)
14 IT=1
SB=0
DO 7 I=1,DR
  MUX(1,1)=X0(I)
  MUX(2,1)=X1(I)
  MUX(3,1)=X2(I)
  MUX(4,1)=Y1(I)
  MUX(5,1)=Y2(I)
7  KGLL=1
CALL ADLM(T,IDR,CA,MUX,IT,DR,KGLL)
CA=CA+16
CALL ADLM(T,IDR,LAT1,PIPO,IT,PDR,KGLL)
ALT1=CA+LAT1+1
OPM1=XMT1(ALT1)
KGLL=5
CALL ADLM(T,IDR,LAT2,PIPO,IT,PDR,KGLL)
SC=PIEO(IT,1)
IF(SC .EQ. 1) GO TO 10
ALT2=CA+LAT2+1
OPM2=XMT2(ALT2)*CLSB1
OPM3=0
10  GO TO 12
ALT3=CA+LAT2+1
OPM3=XMT3(ALT3)*CLSB1
OPM2=0.
12  QPADB=OPM1+OPM2+OPM3
SR=SB+QPADB
IT=IT+1
IF(IT .LE. 5) GO TO 20
YP(E0)=SNGL(SR)*SF
SPSR=YP(RO)
IF(YMAX .LT. SPSR) YMAX=SPSR
CALL LECTCP(SPSC,YOUT,B)
DO 15 I=1,DR
  SY(E0,I)=YOUT(I)
15  RO=EO+1
IF(EO .GT. NS) GO TO 995
DC 13 I=1,DR
  Y2(I)=Y1(I)
  Y1(I)=YOUT(I)
  X2(I)=X1(I)
  X1(I)=X0(I)
13  X0(I)=SX(RO,I)
GO TO 14

TO FlaT THE Freq-RESPONS (DOMAIn=1) AND TIME-DOmAIn
RESPOSS (DOMAIn=0).

995 IF(DCMAIn .EQ. 1) GO TO 77
CALL FLOT3(T1,IF,NS)
PRINT 42,F(FPTS)
42 FORMAT(52X,'INPUT TO THE FILTER :: F = ',F5.0,'HZ.\')
   CALL FLOT3(T1,IP,NS)
   PRINT 43,F(FPTS)
43 FORMAT(47X,'OUTPUT OF THE FILTER :: F = ',F5.0,'HZ.\')
   GC TO 1000
77 YFP(FPTS)=YMAX
9 CONTINUE
   CALL FLOT3(F,YFP,TFP)
C
1000 DO 100 I=1,FPTS
    IFX(I)=IFIX(F(I))
    IFYP(I)=IFIX(YFP(I)*10000)
    PRINT 99,IFX(I),IFYP(I)
99 FORMAT(1X,I5,I5)
100 CONTINUE
C
200 STOP
END
SUBROUTINE IMTLBE(ITL,CON,LT,SC)
INTEGER XLM,PILM,PILM1,CLM,PCLM,PCLM1,AL,SC
REAL*8 XI1,XJ1,XLSB,CLSB1,CLSB2
REAL*8 CON(ITL)
COMMON/B1/ XLSB,CLSB1,CLSB2,XLM,PILM,PILM1,CLM,PCLM,PCLM1
AL=1
DO 1 I=1,XLM
IF(I .GT. PILM) GO TO 3
XI1=(I-1)*XLSB
GO TO 4
3 XI1=(I-1)*XLSB - 2
IF(I .EQ. PILM1) XI1=0
4 IF(LT .EQ. 1) GO TO 5
IF(SC .EQ. 1) GO TO 6
DO 2 J=1,CLM
XJ1=(J-1)*CLSB2
CON(AL)=XI1*XJ1
2 AL=AL+1
GO TO 1
6 DO 7 J=1,CLM
J2=J+CLM
XJ1=(J2-1)*CLSB2 - 2
IF(J .EQ. 1) XJ1=0
CON(AL)=XI1*XJ1
7 AL=AL+1
GO TO 1
5 DO 8 J=1,CLM
IF(J .GT. PCLM) GO TO 9
XJ1=(J-1)*CLSB1
GO TO 10
8 AL=AL+1
1 CONTINUE
RETURN
END
SUBROUTINE ADLMT(IDR,ADL,IPR,IT,LL,KGLL)
INTEGER BI,ADL
INTEGER IPR(5,IDR)
ADL=0
IFL=1
BI=0
DO 1 I=IFL,LL
I1=LL-I+KGLL
ADL=ADL+IPR(IT,I1)*2**BI
1 BI=BI+1
RETURN
END
**SUBROUTINE DECTCP (X, XOUT, B)**

*********************************************************************

THIS IS A SUBROUTINE TO CONVERT A DECIMAL FRACTION INTO BINARY USING TWO'S COMPLEMENT'S FORM.

**SUBROUTINE DECTCP (X, XOUT, B)**
INTEGER XOUT(8)
LOGICAL A(8), B(8)
C1F=0.
DC 16 I=1, 8
16 A(I)= .FALSE.
H=ABS(X)
P=H
BINV=0.5
DC1 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(F .GE. 0.9921875) GO TO 21
CALL ADDN(A, B)
C1F=C1F + 0.00390625
21 IF(X .LT. 0.) GO TO 4
GO TO 6
4 DC 5 I=1, 8
5 A(I)= .NOT. A(I)
CALL ADDN(A, B)
6 DC 15 I=1, 8
XOUT(I)=0
15 IF(A(I)) XOUT(I)=1
RETURN
END
**SUBROUTINE BINCON(X,XOUT,B,IFLAG)**

**TO CONVERT DECIMAL FRACTION TO BIN 2'S COMPLEMENT (FOR MULT. 2 & 3) AND TO 1'S COMPLEMENT (FOR MULT. 1)**

INTEGER XOUT(8)
LOGICAL*1 A(8),B(8)
C1F=0.
DO 16 I=1,8
   A(I)=.FALSE.
   E=ABS(I)
   EINV=.5
   DO 1 I=2,8
      IF(E.GE.BINV) GO TO -2
      GO TO 3
2   A(I)=.TRUE.
   C1F=C1F+BINV
   H=H-BINV
   EINV=BINV/2.
   CONTINUE
1   IF(X.LT.0.) C1F=-C1F
   IF(E.GE.0.9921875) GO TO 20
5   DO 4 I=1,8
   A(I)=.NOT.A(I)
4   GO TO 4
   IF(X.LT.0.) GO TO 4
5   A(I)=.NOT.A(I)
   CONTINUE
   IF(X.LT.0.) C1F=-C1F
   IF(E.GE.0.9921875) GO TO 21
   CALL ADDN(A,B)
   C1F=C1F+0.00390625
21   IF(X.LT.0.) GO TO 4
   GO TO 6
   DO 5 I=1,8
   A(I)=.NOT.A(I)
5   IF(I .EQ. 0) GO TO 6
   CALL ADDN(A,B)
   DO 15 I=1,8
   XOUT(I)=0
15   IF(A(I)) XOUT(I)=1
   RETURN
   END
SUBROUTINE ADDN(A, B)

LOGICAL*1 A(8), B(8), G(8), P(8), S(8), C(9), P1, Q1, R1,

EXOR, TEIXOR

EXOR (P1, Q1) = P1 .AND. .NOT. Q1 .OR. Q1 .AND. .NOT. P1

TEIXOR (P1, Q1, R1) = R1 .AND. .NOT. EXOR (P1, Q1) .OR. R1

C(9) = .FALSE.,

DO 7 I = 1, 8

N = 9 - I

G(N) = A(N) .AND. B(N)

F(N) = EXOR (A(N), B(N))

S(N) = TEIXOR (A(N), B(N), C(N+1))

C(N) = G(N) .OR. C(N+1) .AND. P(N)

7 CONTINUE

DO 11 I = 1, 8

A(I) = S(I)

RETURN

END
** THIS PROGRAM GENERATES CONTENTS OF MULT-TABLE1**
** THAT IS : ALL POSSIBLE COMBINATION OF rk*ckh ; **
** rk=INPUT/OUTPUT DATA **
** ckh= 4 MSB OF COEFFICIENTS **

**************************************************************************************

DIMENSION XMT(4096),DATA(4096)
INTEGER XOUT(8)
LOGICAL A(8), B(8)
DATA .FALSE., .FALSE., .FALSE., .FALSE., .FALSE., .FALSE., .FALSE., .TRUE./
AI=1
IFLAG=0
DC I=1, 1, 256
IF (I.GT.128) GO TO 3
RK=(I-1)*(2**(-7))
GC TO 4
3 RK=(I-1)*(2**(-7)) - 2
IF (I.EQ.129) RK=0
4 DC J=1, 1, 16
IF (J.GT.8) GO TO 9
CKH=(J-1)*(2**(-3))
GC TO 10
9 CKH=(J-1)*(2**(-3)) - 1.875
IF (J.EQ.9) CKH=-0.875
10 XMT(I)=RK*CKH
8 AI=AI+1
1 CONTINUE

C TO CHANGE VALUES TO 1'S COMP.

DC I=1, 4096
I=XMT(I)
CALL BINCON(I, XOUT, 5, IFLAG)
DATA(I)=XOUT(8)+2*XOUT(7)+4*XOUT(6)+8*XOUT(5)+16*XOUT(4)
1 +32*XOUT(3)+64*XOUT(2)+128*XOUT(1)
100 CONTINUE
K=0
DC I=1, 256
PRINT 99, (DATA(J+K), J=1, 16)
99 FORMAT(2X, 16(F5.0))
199 K=K+16
STCF
END
DIMENSION XMT2(4096), IDATA(4096)
INTEGER XOUT(8)
LOGICAL*1 A(8), B(8)
DATA B//.FALSE..FALSE..FALSE..FALSE..FALSE..FALSE..FALSE.//
     .FALSE..TRUE.//
IFLAG=1
AL=1
DC 1 I=1,256
IF (I.GT.128) GO TO 3
RK=(I-1)*(2.**(-7))
GO TO 4
3
RK=(I-1)*(2.**(-7))-2
IF (I.EQ.129) RK=0
DC 2 J=1,16
CLK=(J-1)*(2.**(-4))
XMT2(AL)=RK*CLK
AL=AL+1
CONTINUE

TO CHANGE VALUES TO 2'S COMP.

DC 100 I=1,4096
X=XMT2(I)
CALL BINCON(X,XOUT,E,IFLAG)
IFDATA(I)=XOUT(8)+2*XOUT(7)+4*XOUT(6)+8*XOUT(5)+16*XOUT(4)
     +32*XOUT(3)+64*XOUT(2)+128*XOUT(1)
CONTINUE
K=0
DC 199 I=1,256
PRINT 99,(IDATA(J+K),J=1,16)
99 FORMAT(16(I5))
199 K=K+16
STOP
END
C
* THIS PROGRAM GENERATES CONTENTS OF MULT-TABLE3
* THAT IS : ALL POSSIBLE COMBINATION OF 'rk*ckl' FOR
* ALL NEGATIVE VALUES OF COEFFICIENTS.
* rk=INPUT/OUTPUT DATA
* ck1=4LSB OF COEFFICIENTS
******************************************************************************

DIMENSION XMT3(4096),IDATA(4096)
INTEGER XOUT(8)
LOGICAL*1 A(8),B(8)
DATA B/=FALSE.,FALSE.,FALSE.,FALSE.,FALSE.,FALSE.,FALSE.,
1 .FALSE.,.TRUE./
AL=1
IFLAG=1
DC 1 I=1,256
IF(I.GT.128) GO TO 3
RK=(I-1)*(2.*(-7))
GO TO 4
3 RK=(I-1)*(2.*(-7))-2
IF(I.EQ.129) RK=0
4 DO 7 J=1,16
   J2=J+16
   CKL=(J2-1)*(2.*(-4))-2
   IF(J.EQ.1) CKL=0.
   XMT3(AL)=RK*CKL
7   AL=AL+1
1 CONTINUE
C

$HASP170 PRINTER2 BACKSPACED
APPENDIX - C

OPERATING PROCEDURE FOR MICRO-BASED STRUCTURE

1. Power up the microprocessor based control unit, teletype, SDK-85, and associated equipments.

2. Insert the disk, and press the 'RESET' button on the front panel of the micro-development system.

3. Load in the program, and locate the object code. Select the IC8-85 mode, and map the memory of the development system to IC8-85 user's area, as shown below:

   IC8C5 MAP 4000 LENGTH 32K=INTELLEC 7000 MAP FC00 TC
   FFFF=INTELLEC FC00 MAP IO 00 , TO FF=USER RESET
   HARDWARE LOAD FILTER

4. Type in number of coefficients, their values.

5. Type: GO FROM .START TILL .EX EXECUTED - the filter will now be ready to operate once the 'CR' is pressed.

6. To stop the operation, press the 'ESC' key. Then type in 'EXIT' in order to come out of IC8-85 mode.
APPENDIX - D

LISTING OF SOFTWARE ROUTINES (USING INTELL 8085)
PROGRAMMABLE COEFFICIENT DIGITAL FILTER

:----------------------------------------:

:**************************************************************************

: NUMBER OF SCALED COEFFICIENTS AND
: THEIR VALUES WILL BE ENTERED INTO
: THE SPECIFIED MEMORY LOCATION
: VIA KEYBORD.

: THE FILTER WILL RUN UNTIL THE
: ESC KEY IS PRESSED WHICH WILL
: TERMINATE THE PROCESS.

: TO PROCESS THE NEXT FILTER
: THESE COEFFICIENTS ARE STORED THE
: "YES" KEY MUST BE PRESSED & RELEASED.

:**************************************************************************

: CPU REGS EACH: LOCATION OF NO OF COEFF

: CPU REGS EACH: LOCATION OF FIRST COEFF

: CPU REGS EACH: UP TO 50364 ARE LOC OF
: OTHER COEFFS...

: CPU REGS EACH: ADD TO REGISTER LOCATION OF LSF BYTE OF
: ACC.

: CPU REGS EACH: ADD TO REGISTER LOCATION OF LSF BYTE OF
: ACC.

: CPU REGS:

: X = 0

: START: LUI SP,7FFFH: INITIALIZE THE STACK.

: TO CLEAR THE ACCUMULATOR.

: LUI A

: A = A,ACUM1

: ADD A,CLEAR LSF BYTE OF ACC.

: JSR 3,0:: "CLS"

: TO SAVE LOCATION OF 1ST COEFF IN SP:

: LUI A

: A,ACUM1

: JSR 3,0:: "CLS"

: CPU REGS EACH: NEC DELAY

: LUI: IN CH/FEED THE BLOCK PULSE
LOC OBJ LINE SOURCE STATEMENT
4013 07 49 :LOC = 'EUC'=1?
4019 21648 50 JNC LIN1 IF YES GO A1.
4021 47 51 ADD PCL,FLR, C30H.
4022 30 52 NC DELAY=15 SEC DELAY.
4023 3 53 AND (INQ,SET, A30H) AND (Q1, Q'1, INQ, SET 30H).
4024 1 54 ADD PCL,FLR, C30H.
4025 3 55 CALL DELAY=15 SEC DELAY.
4026 3 56 NC DELAY=15 SEC DELAY.
4027 30 57 CALL DELAY=15 SEC DELAY.
4028 3 58 ADD PCL,FLR, C30H.
4030 3 59 CALL DELAY=15 SEC DELAY.
4031 45 60 JTC 98:
4032 65 61 JTC 98:
4033 65 62 JTC 98:
4034 79 63 JTC 98:
4035 37 64 JTC 98:
4036 3F 65 JTC 98:
4037 17 66 JTC 98:
4038 F5 67 JTC 98:
4039 0301 68 JTC 98:
403A 75 69 JTC 98:
403E 5C 70 JNC LIN1 IF YES GO A1.
403F 3A 71 ADD PCL,FLR, C30H.
4040 3 72 CALL DELAY=15 SEC DELAY.
4041 3E 73 CALL DELAY=15 SEC DELAY.
4043 41 74 CALL DELAY=15 SEC DELAY.
4044 00 75 CALL DELAY=15 SEC DELAY.
4045 00 76 CALL DELAY=15 SEC DELAY.
4046 00 77 CALL DELAY=15 SEC DELAY.
4047 00 78 CALL DELAY=15 SEC DELAY.
4048 00 79 CALL DELAY=15 SEC DELAY.
4049 00 80 CALL DELAY=15 SEC DELAY.
404A 00 81 CALL DELAY=15 SEC DELAY.
404B 00 82 CALL DELAY=15 SEC DELAY.
404C 00 83 CALL DELAY=15 SEC DELAY.
404D 00 84 CALL DELAY=15 SEC DELAY.
404E 00 85 CALL DELAY=15 SEC DELAY.
404F 00 86 CALL DELAY=15 SEC DELAY.
4050 00 87 CALL DELAY=15 SEC DELAY.
4051 00 88 CALL DELAY=15 SEC DELAY.
4052 00 89 CALL DELAY=15 SEC DELAY.
4053 00 90 CALL DELAY=15 SEC DELAY.
4054 00 91 CALL DELAY=15 SEC DELAY.
4055 00 92 CALL DELAY=15 SEC DELAY.
4056 00 93 CALL DELAY=15 SEC DELAY.
4057 00 94 CALL DELAY=15 SEC DELAY.
4058 00 95 CALL DELAY=15 SEC DELAY.
4059 00 96 CALL DELAY=15 SEC DELAY.
405A 00 97 CALL DELAY=15 SEC DELAY.
405B 00 98 CALL DELAY=15 SEC DELAY.
405C 00 99 CALL DELAY=15 SEC DELAY.
405D 00 A0 CALL DELAY=15 SEC DELAY.
405E 00 A1 CALL DELAY=15 SEC DELAY.
405F 00 A2 CALL DELAY=15 SEC DELAY.
4060 00 A3 CALL DELAY=15 SEC DELAY.
4061 00 A4 CALL DELAY=15 SEC DELAY.
4062 00 A5 CALL DELAY=15 SEC DELAY.
4063 00 A6 CALL DELAY=15 SEC DELAY.
4064 00 A7 CALL DELAY=15 SEC DELAY.
4065 00 A8 CALL DELAY=15 SEC DELAY.
4066 00 A9 CALL DELAY=15 SEC DELAY.
4067 00 AA CALL DELAY=15 SEC DELAY.
4068 00 AB CALL DELAY=15 SEC DELAY.
4069 00 AC CALL DELAY=15 SEC DELAY.
406A 00 AD CALL DELAY=15 SEC DELAY.
406B 00 AE CALL DELAY=15 SEC DELAY.
406C 00 AF CALL DELAY=15 SEC DELAY.
406D 00 B0 CALL DELAY=15 SEC DELAY.
406E 00 B1 CALL DELAY=15 SEC DELAY.
406F 00 B2 CALL DELAY=15 SEC DELAY.
4070 00 B3 CALL DELAY=15 SEC DELAY.
4071 00 B4 CALL DELAY=15 SEC DELAY.
4072 00 B5 CALL DELAY=15 SEC DELAY.
4073 00 B6 CALL DELAY=15 SEC DELAY.
4074 00 B7 CALL DELAY=15 SEC DELAY.
4075 00 B8 CALL DELAY=15 SEC DELAY.
4076 00 B9 CALL DELAY=15 SEC DELAY.
4077 00 BA CALL DELAY=15 SEC DELAY.
4078 00 BB CALL DELAY=15 SEC DELAY.
4079 00 BC CALL DELAY=15 SEC DELAY.
407A 00 BD CALL DELAY=15 SEC DELAY.
407B 00 BE CALL DELAY=15 SEC DELAY.
407C 00 BF CALL DELAY=15 SEC DELAY.
407D 00 C0 CALL DELAY=15 SEC DELAY.
407E 00 C1 CALL DELAY=15 SEC DELAY.
407F 00 C2 CALL DELAY=15 SEC DELAY.
4080 00 C3 CALL DELAY=15 SEC DELAY.
4081 00 C4 CALL DELAY=15 SEC DELAY.
4082 00 C5 CALL DELAY=15 SEC DELAY.
4083 00 C6 CALL DELAY=15 SEC DELAY.
4084 00 C7 CALL DELAY=15 SEC DELAY.
4085 00 C8 CALL DELAY=15 SEC DELAY.
4086 00 C9 CALL DELAY=15 SEC DELAY.
4087 00 CA CALL DELAY=15 SEC DELAY.
4088 00 CB CALL DELAY=15 SEC DELAY.
4089 00 CC CALL DELAY=15 SEC DELAY.
408A 00 CD CALL DELAY=15 SEC DELAY.
408B 00 CE CALL DELAY=15 SEC DELAY.
408C 00 CF CALL DELAY=15 SEC DELAY.
408D 00 D0 CALL DELAY=15 SEC DELAY.
408E 00 D1 CALL DELAY=15 SEC DELAY.
408F 00 D2 CALL DELAY=15 SEC DELAY.
4090 00 D3 CALL DELAY=15 SEC DELAY.
4091 00 D4 CALL DELAY=15 SEC DELAY.
4092 00 D5 CALL DELAY=15 SEC DELAY.
4093 00 D6 CALL DELAY=15 SEC DELAY.
4094 00 D7 CALL DELAY=15 SEC DELAY.
4095 00 D8 CALL DELAY=15 SEC DELAY.
4096 00 D9 CALL DELAY=15 SEC DELAY.
4097 00 DA CALL DELAY=15 SEC DELAY.
4098 00 DB CALL DELAY=15 SEC DELAY.
4099 00 DC CALL DELAY=15 SEC DELAY.
409A 00 DD CALL DELAY=15 SEC DELAY.
LUC 36A LINE 154 SOURCE STATEMENT

0451 3E05 154 MOV A,PH:
0453 B9 155 MOV C:IS COUNTER 5?
0454 C6 156 JZ L3:IF SO GO TO L3
0455 F4 157 INX B:INCREMENT THE COUNTER
0456 23 158 INX,4:POINT TO NEXT COEFFICIENT.
0458 5D 159 MOV E,L:
045A 54 160 MOV D,H:
045B 18 161 PUSH D;D,E=H,L.
045C C33240 162 JMP L2:GO BACK TO L2

-----------------------------
: TO PERFORM ROUNDING :
:-----------------------------

045E 2125A 163 L3: LXI H,ACUML2
0460 7E 164 MOV A,M:GET VALUE OF ACC2
0463 87 165 KLC:CHECK ITS MSR
0466 1A6E40 166 JC L6:IF YES=1,GO L6 AND DO ROUNING
0467 2125A 167 ;OTHERWISE,PRESENT ACC1 IS THE YN V

0469 2E 168 LVI H,ACUML1
046F 4E 169 MOV C,M:STORE ACC1 IN REG-C.(=YN)
0471 C37540 170 JMP L7
0476 D12F5A 171 L6: LXI H,ACUML1
0479 7F 172 MOV A,M:GET ACC1 VALUE
047D 05 173 ADD 1H:
047A 48 174 MOV C,M:STORE IT IN REG-C.
047D 0A11 175 TO CLEAN ACC12 FOR NEXT
047F 6F 176; SAMPLING OPERATIONS:
0481 6F 177 L7: XRA A
0482 2125A 178 LXI H,ACUML1
0484 77 179 MOV A,H
0488 23 180 INX H
048B 77 181 MOV A,H
048D 3900 182 NIK1: MOV A,PUSH:
048F 0220 183 OUT 20H:MAKE PORT 'A' AS O/P
0491 77 184 ANA A:CLEAR CARRY
0495 79 185 MOV A,C:
0499 147 186 TO SCALE-UP THE O/P
049D 148 187 (I.E.,O/P=YN*2)
049F 149 188;
04A3 47 150 ANA A:CLEAR CARRY
04A3 17 151 XLI :O/P=YN*2
04A4 18 152 ;
04A8 0321 153 OUT 21H:SEND YN TO D/A
04A9 3C08 154 MOV A,PH:
This subroutine configures the I/O ports, sets & resets the particular signals and cleans the ports.

At end of this routine, 'SC' = 0,

UF1 = 1 (A/D disabled),
CL= 0,

(MX1 = R (disabled)),

LATCH2 disabled,

LATCH 1 disabled,

MULTI-TALES disabled.)

;*************************************************************************

;*************************************************************************

OUT 21 ; CLEAR PORTA
OUT 22;
MOV X, D;
OUT 24; ; CLEAR PORTB
UPD 4:
OUT 31 ; SET 'SC' TO LOW
MOV A, 004:
OUT 11 ; SET 'OE' TO DISABLE A/D DISCONNECT

;*************************************************************************

OUT 23;
MOV A, 74;
OUT 17 ; END OF PORT 'C' TO 41

;*************************************************************************

OUT 23 ; DISABLE LATCHES & MULTI-TABLES.
ANL 064:
OUT 23 ; ENABLE LATCH 1.

;*************************************************************************

CALL DELAYS: 15 SEC DELAY

;*************************************************************************

OUT 74:
OUT 23 ; DISABLE LATCH 3.

;*************************************************************************

LDI 0:

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************

;*************************************************************************
SUBROUTINE DELAY2:

---

THIS ROUTINE CAUSES DELAY OF 1SEC.
THIS IS DONE BY EXECUTING THE
LOOP BY 62500 TIMES.

; CALLS: NOTHING

---

DELAY2: NOP
LXI D,3A12H:
LD2: DEX D; DECREMENT THE COUNTER.
MOV A,D:
JMP E0A4H;
INT 0D:
REI

---

--------------------------
SUBROUTINE SUM:
--------------------------

-------------------------------
IN 22H:LOAD KCKL:
MOV F,B;SAVE IT IN F-NEG.
IN 21H:LOAD KCKL:
MOV L,A;SAVE IT TEMPORARILY IN L-NEG.

-------------------------------
SAVE CLEO OF KCKL.

-------------------------------

SUM:

-------------------------------
IN 22H:LOAD KCKL:
MOV F,B;SAVE IT IN F-NEG.
IN 21H:LOAD KCKL:
MOV L,A;SAVE IT TEMPORARILY IN L-NEG.

-------------------------------
SAVE CLEO OF KCKL.

-------------------------------

L1:

-------------------------------

L2:

-------------------------------

L3:

-------------------------------

L4:

-------------------------------

L5:

-------------------------------

L6:

-------------------------------

L7:

-------------------------------

L8:

-------------------------------

L9:

-------------------------------

L10:

-------------------------------

L11:

-------------------------------

L12:

-------------------------------

L13:

-------------------------------

L14:

-------------------------------

L15:

-------------------------------

L16:

-------------------------------

L17:

-------------------------------

L18:

-------------------------------

L19:

-------------------------------

L20:

-------------------------------

L21:

-------------------------------

L22:

-------------------------------

L23:

-------------------------------

L24:

-------------------------------

L25:

-------------------------------

L26:

-------------------------------

L27:

-------------------------------

L28:

-------------------------------

L29:

-------------------------------

L30:

-------------------------------

L31:

-------------------------------

L32:

-------------------------------

L33:

-------------------------------

L34:

-------------------------------

L35:

-------------------------------

L36:

-------------------------------

L37:
ACUM1 A 55E
ACUM2 A 5C2C
LET A 4F32
L3 A 4E35
LE A 49AA
LR A 413F
LD1 A 40F9
LD2 A 4117
LI A 4170
NK2 A 40C3
TI A 412C

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

END START
APPENDIX - E

DESCRIPTION OF FILTER CONTROL SIGNALS —
<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCK</td>
<td>general purpose clock signal, used for rising edge activated devices.</td>
</tr>
<tr>
<td>CKA</td>
<td>clock signal for 'DECOCONTR'</td>
</tr>
<tr>
<td>CK1</td>
<td>inverted version of CKA.</td>
</tr>
<tr>
<td>CLACC</td>
<td>The accumulator is cleared when this signal goes low.</td>
</tr>
<tr>
<td>CKACC</td>
<td>The accumulator is clocked once this signal goes high.</td>
</tr>
<tr>
<td>N1 &amp; N6</td>
<td>Used to produce E/D MUX1-8.</td>
</tr>
<tr>
<td>N2 &amp; N6</td>
<td>Used for E/D LATCH 1.</td>
</tr>
<tr>
<td>N3 &amp; N9</td>
<td>Used for E/D MULTI-3.</td>
</tr>
<tr>
<td>N4</td>
<td>Generates E/D LATCH 2,3.</td>
</tr>
<tr>
<td>N7</td>
<td>Used for detecting the end of each cycle.</td>
</tr>
<tr>
<td>DELAY N7</td>
<td>Delay version of N7.</td>
</tr>
<tr>
<td>RESET</td>
<td>Master reset signal for system.</td>
</tr>
<tr>
<td>CKSN</td>
<td>Signal used for clocking the registers SR1-SR5</td>
</tr>
<tr>
<td>LATCH 1-7.</td>
<td></td>
</tr>
<tr>
<td>RCOND</td>
<td>Round signal produced during the 5th cycle for rounding purpose.</td>
</tr>
<tr>
<td>OE</td>
<td>Output Enable signal; used in A/D CONV.</td>
</tr>
<tr>
<td>EOC</td>
<td>End Of Conversion signal.</td>
</tr>
<tr>
<td>SC</td>
<td>Start Conversion signal.</td>
</tr>
<tr>
<td>CKCONT</td>
<td>Used as a cycle count; keeps track of number of cycles used in each sampling operation.</td>
</tr>
<tr>
<td>MCD-5</td>
<td>Signal produced from the MOD-5 CONTR, at the end of 5th cycle operation.</td>
</tr>
</tbody>
</table>
APPENDIX - F

CIRCUIT DIAGRAMS OF MICROPROCESSOR BASED STRUCTURE
MICROPROCESSOR BASED STRUCTURE

<table>
<thead>
<tr>
<th>IC NAME GIVEN</th>
<th>IC VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW1-SW5</td>
<td>SN74198</td>
</tr>
<tr>
<td>MUX1-8</td>
<td>SN74151</td>
</tr>
<tr>
<td>PORT#00,01</td>
<td>INTEL 8355</td>
</tr>
<tr>
<td>PORT#21H,22H,23H</td>
<td>INTEL 8155</td>
</tr>
<tr>
<td>MULT1-3</td>
<td>INTEL 2732</td>
</tr>
<tr>
<td>INV1</td>
<td>SN7404</td>
</tr>
<tr>
<td>A/D CONV</td>
<td>2N427E</td>
</tr>
<tr>
<td>D/A CON</td>
<td>2N425E</td>
</tr>
<tr>
<td>44 CIRCUIT</td>
<td>SN7490</td>
</tr>
<tr>
<td>MUX-9</td>
<td>1/4SN74157</td>
</tr>
<tr>
<td>A/D F/F</td>
<td>SN74LS74</td>
</tr>
<tr>
<td>BUFFER 1 &amp; 2</td>
<td>SN7437</td>
</tr>
<tr>
<td>LATCH1-3</td>
<td>SN74116</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMPONENT NAME</th>
<th>COMPONENT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1N14001</td>
</tr>
<tr>
<td>SW1</td>
<td>PUSH-BUTTON</td>
</tr>
</tbody>
</table>
(5) 'SC Pulse'
(5) Pin4 MUX9
(1) CLESR
(5) 'OE'

S2 of MUX1-8
S1 of MUX1-8
S0 of MUX1-8

(6) CKSR

Pin5-Mult1
- 6 -
- 7 -
- 8 -

Pin5-Mult2
- 6 -
- 7 -
- 8 -

Page 2 of 8

I/O Port 8355
And
J3 Connector
(5) Pin 1-of NJX9
(4) Pin20-Mult1
(4) Pin2-Latch1
(7) Pin2-Latch3
  Pin25-Latch1
  Pin21
  Pin19
  Pin17
  Pin11
(4) Pin9
(4) Pin7
(4) Pin5

(6) Pin11-Buftr2
  Pin8
  Pin6
  Pin3
  Pin11-Buftr1
  Pin8
  Pin6
  Pin3

Page 8 of 8
I/O Port 8155
And
J4 Connector
APPENDIX - G

CIRCUIT DIAGRAMS OF HARDWIRED LOGIC STRUCTURE
# HARDWIRED LOGIC DESIGN STRUCTURE

## IC NAME GIVEN | IC NUMBER
--- | ---
CKCONTR | SN7493
SELEC F/F, CEFF F/F, F/F 1-3, A/D F/F, ROUN F/F | SN74LS74
SH1-SR5 | SN74198
MUX 1-8 | SN74151
MUX1-4, MUX5-8, ADDR MUX | SN74157
CEFF MEMORY | SN7489 (16*4 RAM)
ADDR-CONTR | SN74193
NAND-5-25 | SN7400
NAND-25 | SN7440 (4 INPUT)
HC01-4, CKSR-GEN | SN74123
LATCH1-3, 6-10 | SN74116
LATCH4, 5, 6 | SN74100
SYSCK-GEN | SN74124
ADD1-6 | SN7483
BUFFER-1-4 | SN7437
AND1, MAST RESET AND1-2 | SN7406
ACC1 | SN74273
ACC2 | SN74175
CEFF-REG | INTEL 2708
BUFFERX | SN7407
MULT1-3 | INTEL 2732
INH1-5 | SN7404
OR1-3 | SN7432
CEFF-CONTR | SN74161
ANALOG SWITCH | 4066
DECODA | SN74155
DECO-CONTR | SN7493
MUX-CONTR | SN74193

<table>
<thead>
<tr>
<th>COMPONENT NAME</th>
<th>COMPONENT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 &amp; R2</td>
<td>10K</td>
</tr>
<tr>
<td>R2 &amp; R5</td>
<td>50K</td>
</tr>
<tr>
<td>C1 &amp; Co</td>
<td>70PF</td>
</tr>
<tr>
<td>C2 &amp; C5</td>
<td>140PF</td>
</tr>
<tr>
<td>Cref</td>
<td>1UF</td>
</tr>
<tr>
<td>D1-D3</td>
<td>1N14001</td>
</tr>
<tr>
<td>SI/HS SWITCH</td>
<td>Double Pole-Double Throw Type</td>
</tr>
<tr>
<td>LCC SWITCH</td>
<td></td>
</tr>
<tr>
<td>E/D CEFF-MEMO SWITCH</td>
<td></td>
</tr>
</tbody>
</table>

---
ENABLE/ DISABLE LATCH & RAMS

(17)
From Pin 15 of ACC2
From 'ACC' (21)
From Pin 15 of ACC1

[Diagram of electronic circuit]

RESET (16)

Page 12 of 21

D/A CONVERTOR AND ITS ASSOCIATED CIRCUIT
ROUNDING CIRCUIT
AND
LATCH4- LATCH6
CIRCUIT
REFERENCES

1. KAWORSKI, RICHARD. J. 'Implement Digital Filters Efficiently with a Specially Configured Computer.' Electronic Design, September 1984


5. 'Texas Instruments Application Report on Floating Point Arithmetic with the TMS32010.' February 1984

6. STANLEY L. FABRY 'Special Purpose Hardware For Digital Filtering.' Proc. IEEE, Vol. 63, No. 4, Apr. 1975

7. FAED J. TAYLOR 'Digital Filter Design Handbook.'


12. HALHUT SCHRADER 'High Word Rate Digital Filters with Programmable Table Look-Up.' IEEE Trans. on Circuits and Systems, May 19

13. NEUMO Y. 'A Fast Microprogrammed Digital Filter Design'


17. RAFIQUZZAMAN "Microcomputer Theory and Applications with the INTEL 8085," John Wiley and Sons Inc., 1982


16. ABRHAIM PELED BEDELIN 'Digital Signal Processing Theory, Design and Implementation,' John Wiley and Sons Inc., 19% 

17. RAFIUZZAMAN 'Microcomputer Theory and Applications with the INTEL 8085,' John Wiley and Sons Inc., 1982

18. ALAN V. OPPENHEIM AND R. SCHAFER 'Digital Signal Processing,' Bell Telephone Laboratories Inc., 1975
VITA AUCTORIS

KAMBIE NIKSAN

May 1956  Born in Tabriz, IRAN.

June 1974  Completed High School education at Mansor High School, Tabriz, IRAN.


June 1982  Joined the Masters Program at the Electrical Engineering Department, University of Windsor, Windsor, Ont., Canada. N9B 3P4.

Aug 1984  Candidate for the M.A.Sc. Degree in Electrical Engineering, Department of Electrical Engineering, University of Windsor, Windsor, Ont.
Canada, N9B 3Z4.