AN IMPEDANCE RELAY DESIGN AND THE IMPACT OF ANTI-ALIASING FILTERS ON ITS PERFORMANCE

A Thesis

Submitted to the College of Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Department of Electrical Engineering

University of Saskatchewan

Saskatoon, Saskatchewan

Canada

By

AMITPAL BIMBHRA

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Master's degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying, publication, or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical Engineering,
University of Saskatchewan,
Saskatoon, Saskatchewan, Canada S7N 5A9

ABSTRACT

Faults occur in power systems because of the failure of insulation and structures. The faults almost always result in the flow of large quantities of currents. Protective relays are used, therefore, for detecting the faults and isolating the faulted section of the system before the damage spreads. Traditionally, electromechanical and static technologies have been used for designing and manufacturing relays. More recently, advancements in the VLSI technology have resulted in the development of microprocessor-based numerical relays. Several algorithms have been proposed in the past, which are suitable for using in numerical relays for detecting the occurrence of faults. The hardware of a numerical relay has several subsystems, such as, analog input subsystem, signal conditioning, digital input and output, digital processor and power supply subsystems.

Operating speed and accuracy of numerical relays are of major concern. Numerical relays use digital processors to implement protection algorithm. The limited power of the processors can limit the speed of operation of the relay. The speed of operation of the numerical relay also depends on the components of the analog input subsystem, such as, the characteristics of the anti-aliasing filters and the analog-to-digital converters (ADCs).

This thesis is concerned with the selection of the specialized digital processors for implementing the Discrete Fourier Transform (DFT) algorithm at different sampling frequencies, and the selection of suitable components for the analog input subsystem. The processors were selected based on the computational requirements of the DFT at different sampling frequencies. The filters were designed and simulation tests conducted to evaluate the group delay associated with them at different sampling frequencies. The ADC selection was based on the highest sampling rate it could support. Based on the selection of the digital processors and the ADC, suitable numerical relay hardware was purchased and a numerical distance relay was developed.

ACKNOWLEDGEMENTS

The author would like to express his gratitude and appreciation to Dr. M. S. Sachdev for his invaluable guidance and help throughout the course of this research work. The advice and constant encouragement given by Dr. T. S. Sidhu and David J. Callele is gratefully acknowledged.

The author is thankful to Mrs. Tamije Selvy Munian, Mr. Balamourougan, and Mr. Sandro Aquiles Perez for providing an excellent working environment in the power system protection lab. Their support was invaluable throughout this research work. Thanks are extended to the author's family for their selfless love, support and constant encouragement. Special thanks are due to Ms. Tiova Boardman and all of the friends for their moral support.

Financial support provided by the University of Saskatchewan and the National Sciences and Engineering Research Council (NSERC) of Canada is gratefully acknowledged.

Dedicated To My Beloved Parents

Dr. P. S. Bimbhra

Mrs. Surinder Kaur

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMNS

EMTDC	Electro Magnetic Transient Direct Current Analysis
PSCAD	Power Systems Computer Aided Design
CT	Current Transformer
VT	Voltage Transformer
RAM	Random Access Memory
ROM	Read-Only Memory
PROM	Programmable Read Only Memory
EEPROM	Electronically Erasable Programmable Read-Only Memory
S&H	Sample and Hold
SAL	Successive Approximation Decision Logic
DAC	Digital to Analog Converter
MSB	Most Significant Bit
LSB	Least Significant Bit
ADC	Analog to Digital Converter
FSR	Full Scale Resolution
DSP	Digital Signal Processors
DFT	Discrete Fourier Transform
EHV	Extra High Voltage
HV	High Voltage

MV	Medium Voltage
LV	Low Voltage
SDB	Sundance Digital Bus
FIFO	First In First Out
SDRAM	Synchronous Dynamic Random Access Memory
SBRAM	Synchronous Burst Random Access Memory
RTOS	Real Time Operating Software
UI	User Interface
CPU	Central Processing Unit
DMA	Direct Memory Access
EMIF	External Memory Interface
T.I.	Texas Instruments

CHAPTER 1

INTRODUCTION

1.1 Background

Electrical energy plays a significant role in modern society. The increasing dependence on electrical energy is so marked that even a short interruption in electrical supply can lead to major disruption of everyday life in major urban centers. Computer networks, medical life support equipment, process industry, banking machines, communications and airports need an uninterrupted electrical power supply.

Electrical energy is usually supplied by power systems at reasonable cost and with minimum interruptions. An electric power system can be divided into four subsystems: generation, transmission, sub-transmission and distribution systems [1]. Electric power is produced at generating stations where generators convert mechanical energy into electrical energy. The generator output voltages, which are in the range of 11 kV to 35 kV, are applied to transformers that step-up the generated voltage to higher levels for transmission. The voltage levels of a transmission system are typically in excess of 230 kV. A transmission system, which is a vital link in an integrated power system, transmits electrical energy from a substation at one location to another substation at another location. In transmission substations, the power transformers step-down the voltage to sub-transmission levels that range from 69 kV to 138 kV. The sub-transmission systems are connected to distribution systems that can be classified into primary and secondary distribution systems. The primary distribution systems provide energy to small industrial customers at voltages that typically range from 4 kV to 34.5 kV. The secondary distribution system feeds the residential and commercial users at 120/240 V.

Advancements in technology have made it possible to design and construct power systems, which span wide geographic areas. The systems are comprised of many different items of equipment that are very expensive and so, therefore, power systems represent major capital investments. To maximize the return on this outlay, the power system should be operated efficiently and protected from damage due to abnormal operating conditions and faults, which occur occasionally.

Discriminative power system protection plays an important part. Progress in this field is vital and prerequisite for the operation and continuing development of power systems. Figure 1.1 [1], shows the basic elements of a power system.

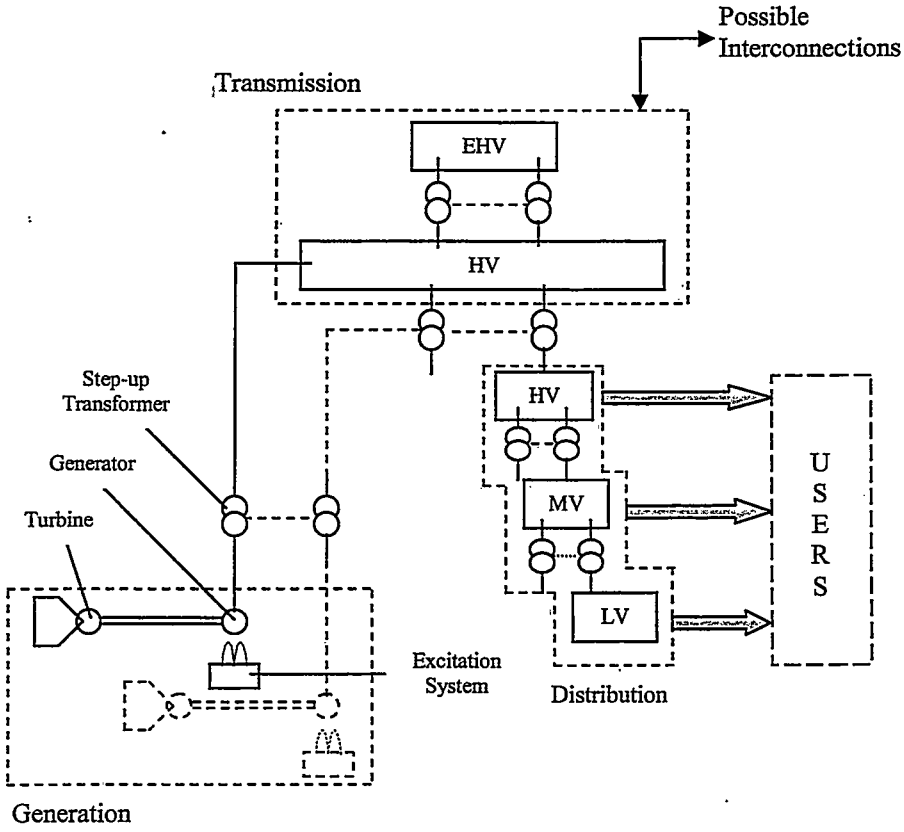


Figure 1.1 Basic elements of an electric power system

(EHV, HV, MV, LV denote extra-high, high, medium, and low voltage, respectively)

1.2 Power System Faults

A power system fault is an unhealthy or abnormal electrical condition in one of the components of the system; such a condition can potentially cause serious damage to the equipment and great loss of revenue. Faults should, therefore, be detected promptly and action should be taken to isolate the faulted element.

1.2.1 Causes of Power System Faults

Faults on a power system can occur due to:

- Lightning
- Punctured or broken insulators
- Birds and animals
- Aircraft or cars hitting power lines and structures
- Trees growing in to power lines
- Pollution (cement dust, moisture, soot, road salt)
- Ice and snow loading
- Wind
- Natural disasters (hurricanes, tornados, earthquakes)
- Failure of insulation due to moisture or aging
- Mechanical damage
- Flashover caused by over voltages due to transients
- Human errors.

1.2.2 Types of Power System Faults

The different types of faults [2] experienced on an electric power system are classified as follows:

1. Single phase to ground faults are experienced as a short circuit between any one of the phases and the ground. From 65% to 75% of all power system faults are single-phase to ground faults.

2. Two phase or phase-to-phase faults are encountered when any two phases are short-circuited. From 20% to 25% of all power system faults are phase-to-phase to faults.

3. Two phase to ground faults occur when any two phases of the power system are short circuited to the ground. From 10% to 15% of all power system faults are two-phase to ground faults.

4. Three phase and three phase-to-ground faults are relatively rare in power systems. From 3% to 5% of all power system faults are three phase-to-ground faults; and these are triggered when all the three phases of a power system are short circuited together.

1.2.3 Effects of Power System Faults

When a fault occurs in a power system, the following abnormal operating conditions are experienced:

- Increase of currents to several times their normal levels
- Decrease of voltage to substantially low levels
- Increase or decrease of the system frequency from its rated value
- Changes in the phase angles of voltages and currents

A power system could sustain serious and extended damage if abnormal conditions or faults were allowed to persist. To avoid the damage, some form of mechanism is needed to detect the occurrence of faults and to disconnect, as soon as possible, the faulted element of the power system. This function is provided by protective relays.

1.3 Protective Relays

Protective relays are devices that detect the presence of a fault and initiate the isolation of the faulted equipment from the remaining power system as quickly as possible. Quality of relaying depends upon the following basic principles [3].

1.3.1 Reliability

Reliability is the trustworthiness of a relay to operate correctly. This manifests in two forms known as dependability and security. Dependability is the certainty that the relay will operate on the occurrence of a fault and security is the ability to avoid incorrect operation during faults.

1.3.2 Speed

Speed is the minimum operating time required to initiate the opening of a circuit when a fault is experienced.

1.3.3 Selectivity

Selectivity is the ability of the protective relays to distinguish between conditions for which immediate action is required and the conditions for which no action is required.

1.3.4 Cost

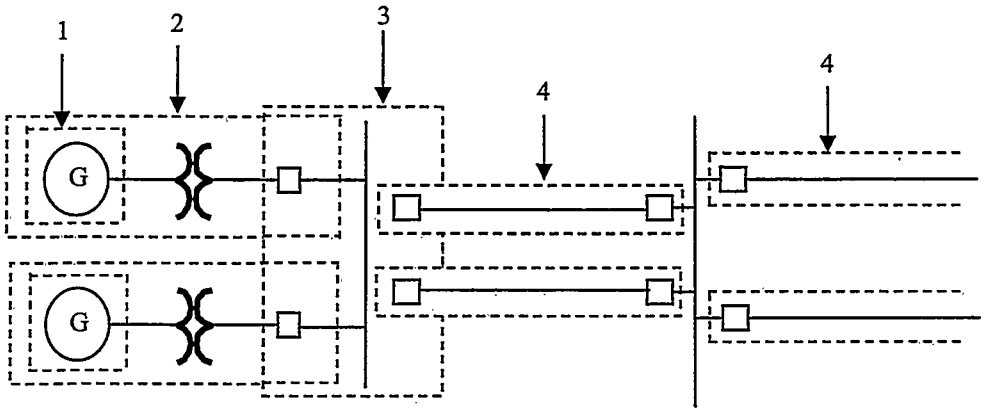
Protective relays should be able to provide maximum protection at the lowest cost.

1.4 Relays in Power Systems

Relays work in combination with circuit breakers by continuously monitoring the power system to ensure availability of maximum electrical supply with minimum damage to equipment, property and life. The basic principle of applying relays is to divide the power system into separate protection zones. Each protection zone has its own set of individual relays and circuit breakers. A typical power system is divided into protective zones for:

- generators
- transformers
- buses
- transmission and distribution systems
- motors

Figure 1.2 shows a basic power system and related protection zones. As seen from the figure, all the elements have their own protection zones. When a fault occurs in any component of the power system, the relays associated with that component detect it and send a signal to the circuit breakers to isolate the component. Adjoining zones overlap to ensure that no part of the power system is left unprotected. When a fault occurs in an overlapped region, relays of both zones operate and isolate both zones from the system.



1- Generator protection zone
2- Generator transformer protection zone
3- Bus bar protection zone
4- Transmission line protection zone

Figure 1.2 Power system with protective zones

Figure 1.3 shows a single line diagram of a typical relay [4]. The magnitudes of power system currents and voltages are very high and cannot be applied to relays directly. The current and voltage levels are reduced to typically 5 A and 110 V nominal values using current transformers (CTs) and voltage transformers (VTs).

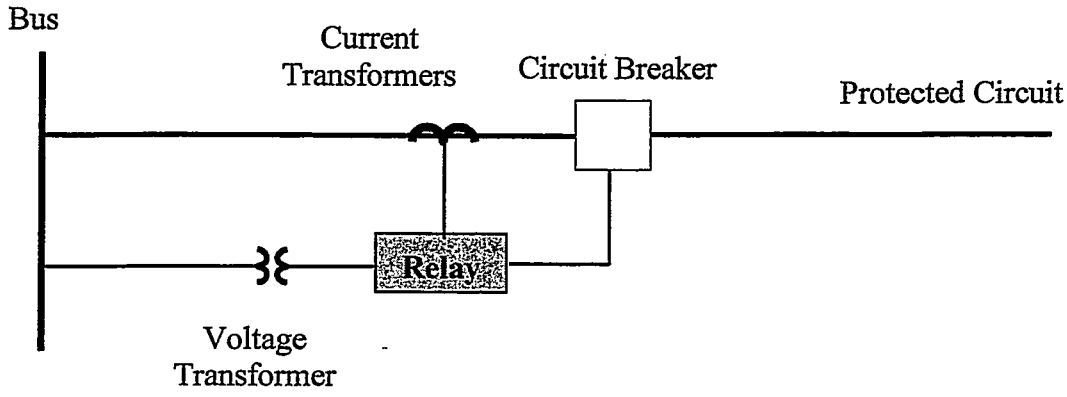


Figure 1.3 Typical arrangement of a relay

1.4.1 Evolution of Power System Relays

The earliest power system protection device used to isolate a faulted element was a fuse. Its major disadvantage is that it cannot distinguish a faulted zone from an unfaulted zone, and has to be replaced after each operation. Later, relays were developed using electromechanical and electromagnetic principles. The relays provided a considerable improvement in the protection of power systems. The electromechanical relays convert energy in the electrical inputs to mechanical force, which is translated to physical movement that closes one or more trip contacts [5]. These relays detect the occurrence of faults and energize the trip circuit of the circuit breakers provided to control the system element. With the increase in the complexity of power systems, advancements in protective devices continued to meet the system requirements.

Introduction of solid-state electronic relays began in the early 1960's. Their design was based on the use of analogue electronic devices to create the relay

characteristics. Early versions of electronic relays used discrete devices such as transistors and diodes in conjunction with resistors, capacitors, inductors; and had huge failure rates. Advances in electronics introduced Large Scale Integrated (LSI) technology, after which more reliable and accepted designs of electronic relays evolved [6]. As there are no moving parts in solid-state relays, they are also known as "static" relays. They provided faster operating speeds, versatile characteristics, low burden and incorporated several protective functions in one compact unit.

With the advent of digital technology, microprocessor-based relays were developed [6, 7, and 8]. Microprocessors and microcontrollers replaced analogue circuits used in static relays to implement relay functions. The first investigations in the use of digital technology for power system protection started in the 1960's. Last and Stalewsky [9] proposed the use of real time computations for system protection in 1966. In 1968, Rockefeller [10] studied the feasibility of using a single computer for the protecting all major components in a substation. This work inspired the development and installation of the first microprocessor based relay at the Tesla substation as a joint venture between the Westinghouse Electric Corporation and the Pacific Gas and Electric Company [11]. A more detailed review of microprocessor-based relays, also called numerical relays, is provided in Chapter 2.

1.5 Objective of the Thesis

The objective of the research presented in this thesis is to investigate the different components in the analog input subsystem, and to evaluate the speed of execution of a numerical relay algorithm at different sampling frequencies, using specialized digital signal processors. A numerical relay uses processors to implement the protection algorithm. The limited power of the processors can restrict the number of samples of inputs that can be processed. The speed of execution of the numerical relay algorithm and thus the speed of operation of the numerical relay is determined by the processors' computational ability. The speed of operation of the numerical relay, also depends on the

hardware components in the analog input subsystem, such as the anti-aliasing filters and the analog-to-digital converters.

1.6 Outline of the Thesis

This thesis is organized into eight chapters and four appendices. Chapter 1 introduces the background of power system protection, the focus of the research and outlines the organization of this work.

Chapter 2 describes the basic functional blocks of a typical numerical relay. An overview of the important features and the description of major functional components of a numerical relay are presented. This chapter also discusses the errors associated with the various hardware components that constitute a numerical relay.

The algorithms for numerical relays are presented in Chapter 3. The mathematical equations and the assumptions used in developing the algorithms are given. The advantages and disadvantages associated with the algorithms are also outlined.

Chapter 4 discusses the factors and the tradeoffs affecting the selection of the algorithms and the hardware for numerical relays. The procedures followed in the selection of the algorithm and the hardware used for this project are described. The importance, basic requirements and the design of anti-aliasing low-pass filters at the front end of numerical relays are presented. A description of the selected hardware for this project is also included.

Chapter 5 describes the implementation of the numerical relay algorithm on the selected processors using the real time operating software.

Chapter 6 describes the simulation studies executed to evaluate the performance of the algorithm and the impact low-pass anti-aliasing filters on the estimation of operating parameters. Results obtained from the studies are presented and discussed.

Finally, Chapter 7 includes a summary and conclusions drawn from the work reported in the thesis. A list of references is provided in Chapter 8.

There are five appendices included in this thesis. In Appendix A, an introduction to the EMTDC is given, and the parameters of the transmission line and other components of the power system used for generating the simulation data are presented. In Appendix B, some additional results are presented for different sampling frequencies. The list of various data acquisition boards considered, are tabulated in Appendix C. In Appendix D, the digital output of the analog-to-digital converter considered for this project is described. Appendix E gives a brief introduction to the digital processors.

1.7 Summary

In this chapter, a brief introduction to power system protection, various causes of faults, types of faults, and the impact of faults on the power system are discussed. Protective devices used to detect these faults and isolate the faulted elements from the power system are also described. The evolution of protective devices such as fuses, electromechanical, solid-state and numerical relays is reviewed. Finally, the objective of this research project and the outline of this thesis are presented.

CHAPTER 2

NUMERICAL RELAYS FOR POWER SYSTEM PROTECTION

2.1 Introduction

Typical features of numerical relays are summarized in this chapter. A functional description of the major blocks of a numerical relay is then provided. The sources of errors caused by limitations of the hardware used for numerical relays are then examined finally.

2.2 Features of a Numerical Relay

Initially, computers were used in power systems for off-line analysis such as load-flow studies, fault studies, transient and dynamic stability analysis, planning and load forecasting, etc. The use of computers for protection was not a viable proposition in the 1960's due to their inability to meet the need for performing numerically intensive calculations in a timely manner. With the significant advances in the development of large scale integrated technology, it became possible to design viable microprocessor-based relays that replaced the relays designed with electromechanical and solid-state technologies. The development of 16-bit and 32-bit microprocessors made high speed computer relaying technically achievable; these relays, called numerical relays, turned out to be superior to their predecessor electromechanical and static relays. Some of the benefits gained from numerical relays are as described in the following sections [3]:

2.2.1 Flexibility

Single general purpose hardware can be used for designing numerical relays that can perform a variety of protection and control functions. This feature reduces the need for keeping large inventories of spares for maintenance of relays. Numerical relays use software to implement appropriate relaying algorithms. The characteristics of a numerical

relay can be modified by altering the relay's software. The numerical relays are, therefore, more flexible than conventional electromechanical and static relays without substantial increase in cost.

2.2.2 Reliability and Self-Checking

A failure in a conventional relay becomes apparent only when the relay fails to operate or operates incorrectly when an abnormal operating condition is experienced in the power system. Numerical relays, however, are designed to regularly monitor themselves and most hardware failures in them are detected soon after they occur. The failures are reported automatically to the protection engineers who have an opportunity to repair the relay before a system disturbance is experienced. This eliminates the need for conducting routine maintenance of the relay. Automatic self supervision and reporting of failures improve the reliability of the protection system that consists of numerical relays.

2.2.3 Data Interface Access

A numeric relay is usually equipped with input and output ports through which data and control commands are exchanged. Sequence-of-events, which occur on the power system, are recorded and transmitted to a central computer through a communication link. This information is used in further investigations that would generally lead to improved system design, system operating practices and relay designs.

2.2.4 Adaptive Capability

This feature allows the relay settings to be changed automatically as the operating condition of the power system changes. This guarantees that relay settings are suitable for the real time operating state of the power system instead of the settings based on a critical system operating state.

2.2.5 Mathematical Capability

Designs of conventional relays are constrained by the characteristics and limitations of electro-mechanical or solid–state technologies. Numerical relays, on the other hand, can be programmed to provide characteristics of almost any shape.

2.2.6 Fault Locating Capability

Fault locating has become a standard feature in nearly all numerical relays [12]. Fault locating information reduces the time for finding the location where repairs are to be performed on a faulted line.

2.3 Functional Description of a Numerical Relay

Numerical relays are microprocessor based devices, which consists of two main components: software and hardware. Software is used to process quantized signals for implementing protection functions. The hardware is used for converting analog voltages and current waveforms to numerical form for use with the software in a microprocessor. Major aspects of the hardware are discussed in this chapter whereas the software is discussed in Chapter 3.

Power system currents and voltages are in the kiloampere (kA) and kilovolt (kV) range. It is not appropriate to bring the signals of these levels into numerical relays. The currents are reduced by using CTs to either 5 A or 1 A and the voltages are reduced by using VTs to 110 V or 120 V.

Numerical relays are made of subsystems with well defined functions. Figure 2.1 shows the main functional blocks of a numerical relay [7].

2.3.1 Analog Input Subsystem

The analog input subsystem consists of

- isolation and scaling systems,
- anti-aliasing filters,
- sample and hold devices,
- multiplexers, and
- analog-to-digital converters.

The isolation and scaling systems consist of auxiliary CTs, VTs and surge arrestors. The outputs of the main CTs and VTs are scaled down further by using auxiliary CTs and VTs. The electronic circuits of numerical relays are protected from power system transients by using metal oxide varisters.

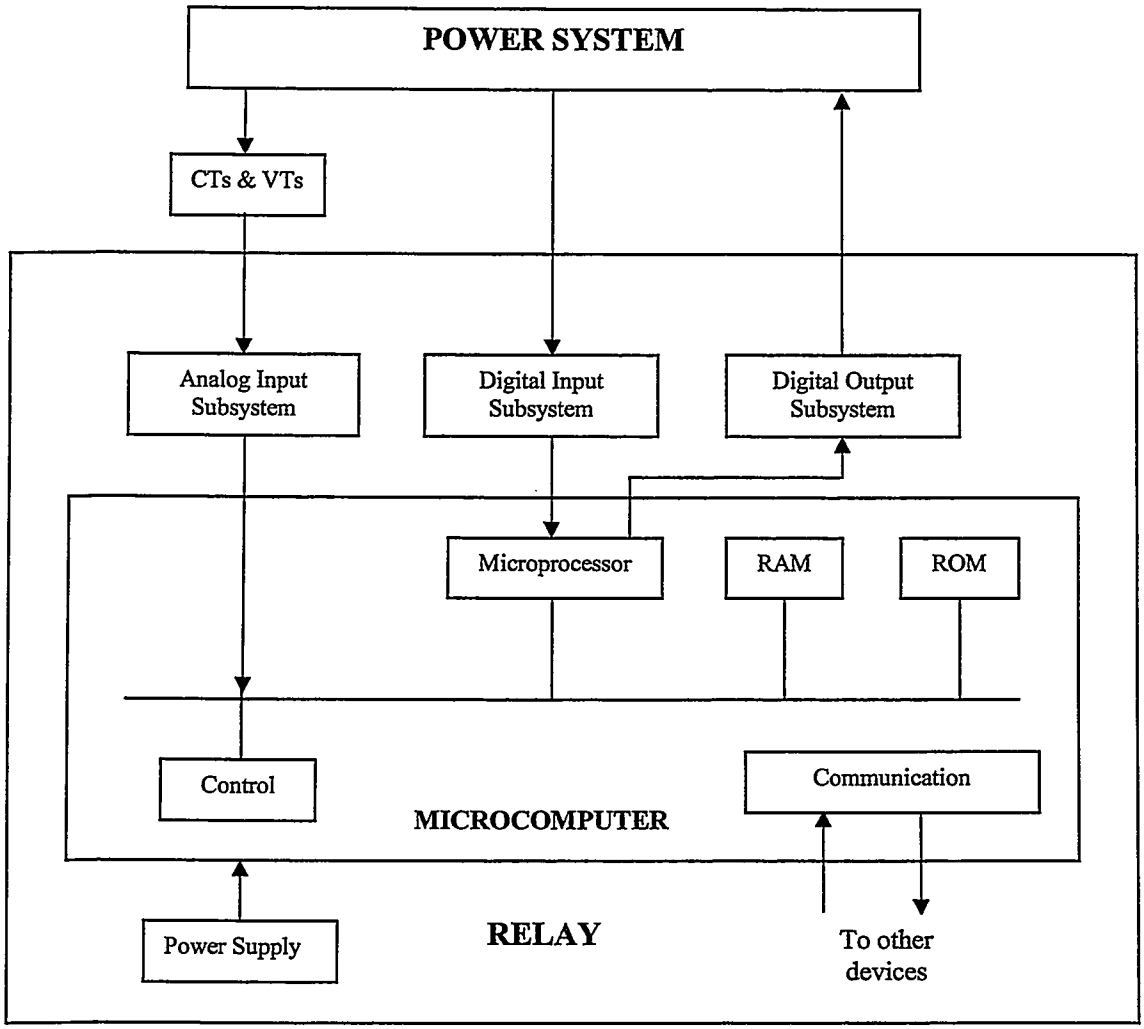


Figure 2.1 Block diagram of a typical numerical relay

Pre-filtering with low-pass filters is done to remove high frequency components. The scaled down and filtered voltages and currents are sampled using sample-and-hold circuits. As analog-to-digital converters accept only voltage signals as inputs, current

signals are converted into equivalent voltages. Analog-to-digital converters generate numerical values corresponding to the levels of the analog signals captured by the sample and hold circuits.

2.3.2 Digital Input Subsystem

The digital input subsystem provides the status of the circuit breaker(s) (ON/OFF), isolators and contact switches to the relay. Digital input wiring must be properly shielded to protect the relay from transient voltages that may be present on the wiring. The number of digital inputs can be five to ten [6].

2.3.3 Digital Output Subsystem

The digital output subsystem conveys the decisions of the relay to the power system. These outputs from the relay, generally, provide signals for tripping circuit breakers and turning on attenuators. A maximum of five to ten digital outputs are sufficient for most relaying applications [6].

2.3.4 Central Processing Unit

The central processing unit manipulates the data obtained from the input subsystems by implementing the algorithm stored in memory. The basic tasks performed by the microprocessor are as follows:

- Execution of relay programs
- Maintenance of various functions for sample-and-hold circuits, multiplexers and analog-to-digital converters
- Communicating with the peripherals connected to the relaying system.

2.3.5 Memory

Several types of memories are used; each of them serves a specific purpose. ROM (Read Only Memory) or PROM (Programmable Read Only Memory) is used to store relay programs permanently. EEPROM (Electronically-Erasable Programmable Read-Only Memory) is used for storing relay settings and other vital information. This

information must remain intact even if the power supply to the relay is interrupted. RAM (Random Access Memory) holds the numerical values of voltages and currents as they are acquired and processed. The processor communicates with these memories through its data, address and control buses.

2.3.6 Power Supply

A digital relay requires an uninterrupted supply of power which is usually supplied by a battery through a single DC input multiple DC output converter. A battery charger or AC to DC converter keeps the battery fully charged at all times. This ensures that the relay performs its intended function even when the station AC power supply is interrupted.

2.3.7 Other Components

Numerical relays are provided with communication ports to share information with other devices. Some of the tasks, such as reading fault records or changing the relay settings, can be and are performed remotely.

2.4 Errors in Numerical Relays

As discussed in Section 2.3, most subsystems of a numerical relay are digital in nature and have inherent errors associated with them. As a result, the outputs provided by a numerical relay will have limited accuracy.

The voltage and currents from a power system are continuous-time signals. There levels are quantized so that microprocessors can process them. This conversion process introduces errors that are discussed in this section.

Errors are caused by imperfect representation of the input signal by

- Anti-aliasing filters
- Sample and hold circuits
- Multiplexers

- Analog-to-digital converters
- Processors

2.4.1 Errors caused by Filters

To provide adequate filtering, the order of the filters is first determined. Higher order low-pass filters remove unwanted frequency components more efficiently than the lower order filters. However, higher order filters are associated with longer delays, which are not desirable in high speed relays. The dynamic response of the filter should also be taken into account [6]. The important aspects of the dynamic characteristics are:

- Rise Time - this gives an indication of how long the output of a low-pass filter takes to reach its final value.
- Overshoot - this gives an indication of how much the output of a filter will exceed its steady state value initially in response to a step input.
- Settling Time - this gives an indication of how long it takes the output of a filter to settle to its steady state value.

2.4.2 Errors caused by Sample-and-Hold circuits

The sample-and-hold (S&H) circuits have droop error, which causes their output voltage to drop to lower values than the actual sampled voltage. This droop should be within the tolerable limit of the analog-to-digital converter.

2.4.3 Errors caused by Analog-to-Digital Converters

Analog-to-digital converters (ADCs) operate on continuous time signals, v, and converts them to discrete-time binary-coded numerical values, Q (v). There are two purposes for performing these conversions:

1) to enable computer analysis of the signal, and
2) to enable digital transmission of the signal.

ADCs introduce errors, many of which are within acceptable limits for well designed converters. The unavoidable errors are the quantization errors and the saturation errors.

2.4.3.1 Quantization Errors

The principal feature of an ADC is its word length expressed in bits. Word length affects the ability of the ADC to represent the analog signal with a sufficient detailed digital representation. Ideally, an infinite number of bits may be required to accurately represent some analog inputs within a specified range. For actual implementations, these numbers are either truncated or rounded to fit into a selected word length.

(a) Truncation Errors: The difference between a quantized value of an analog signal and its true level, Q (v)-v, is the truncation error. The largest error occurs when all the truncated bits are unity. For positive numbers, the error is zero or negative and its magnitude is less than the value of the least significant bit (LSB). Thus, for truncation of numbers:

$$0 \geq E_t \geq -2^{-N} \tag{2.1}$$

where,

E_t is the Truncation Error

N is the number of bits in the ADC

(b) Rounding Errors: Instead of truncating, the numbers are usually rounded to fit into a finite-length word of the ADC. This process consists of choosing the closest quantization level. Numbers lying exactly halfway between two quantization levels may either be rounded up or down. Assuming that a number falling exactly between two steps always rounds up, the rounding error (E_r) is:

$$-\left(\frac{1}{2}\right)\left(2^{-N}\right) < E_r \leq \left(\frac{1}{2}\right)\left(2^{-N}\right) \tag{2.2}$$

2.4.3.2 Saturation Errors

ADC's are designed to operate for specified maximum and minimum input voltages that are equal to the reference voltages of the ADC. Reference voltages for commercially available ADC's usually are ± 12 V, ± 10 V, ± 5 V, ± 1 V. If the level of an input signal is beyond the range of the reference voltage, the output of the ADC would saturate. Figure 2.2 shows an input of 1.2 V peak signal as acquired by an ADC designed for an input range of +1 V to -1 V. Table 2.1 summarizes the errors associated with various components of the analog input subsystem.

Table 2.1 Error summaries for analog input subsystem components

Component	Errors	Delays
Anti-aliasing Filter	Rise time Overshoot Settling time	Delay in input signal
Sample and Hold	Offset voltage Pedestal error Droop Aperture Delay Drift with time & temperature Input noise	Acquisition time Settling time
ADC	Quantization errors Saturation errors Offset voltage Differential & Linearity errors Gain errors Drift with time & temperature Missing codes	Conversion time

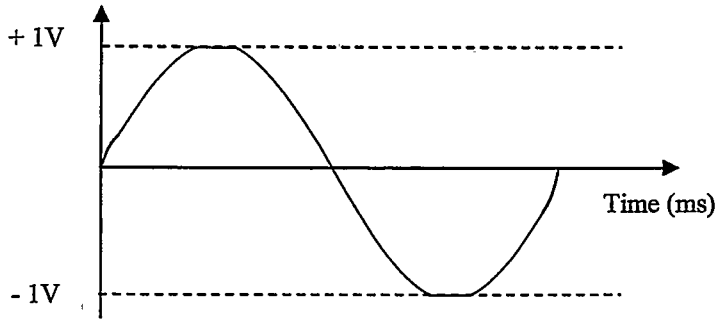


Figure 2.2 Representation of a signal from saturated ADC

2.4.4 Digital Signal Processor Errors

The quantized signal from ADC is sent to one or more processors for performing mathematical computations. Processors, like ADCs have a finite word size, which limits the storage of the results after computations, like additions, subtractions, multiplications and divisions. For example, when two numbers of (N+1) bits are multiplied, the product is a number that cannot be stored in a word of (N+1) bits. In such cases, truncation or rounding is needed. If the two numbers are added or subtracted and the result can be represented by word of (N+1) bits, there will be no overflow or underflow. When the result of addition or subtraction cannot be held in the word of the processor, overflow or underflow results. The results computed by numerical relays for such cases are erroneous. Proper care should be taken to avoid these conditions.

2.5 Summary

This chapter describes the features of a numerical relay, presented and discussed the major functional blocks of a typical numerical relay. Some sources of errors in the digital components of the numerical relays are identified and described. Chapter 3 presents commonly used algorithms for numerical relays.

CHAPTER 3

ALGORITHMS FOR COMPUTER RELAYING

3.1 Introduction

Numerical relays consist of two main components: hardware and software. Hardware configuration for numerical relays is described in Chapter 2 including all its subsystems. The algorithms used in numerical relays and the errors associated with those algorithms are examined in this Chapter.

In conventional electromechanical and static relays, the operating characteristics are fully defined by the design of the hardware and remain fixed for a relay. Numerical relays, however, are much more flexible and can be used to perform a number of different tasks. What makes the hardware of a numerical relay more adaptable is the software that determines its characteristic and functionality. An integral and important part of the software is the algorithm, which is a set of mathematical instructions used to process input currents and/or voltages. The objective is to estimate system parameters, such as the RMS values of inputs, impedance, fundamental frequency, differential currents etc. The calculated parameters are then used to decide whether the power system is experiencing a fault or not, and consequently, initiate the action necessary to isolate the faulted element(s) of the power system. The algorithms used to obtain these parameters use numerical data that represents the waveforms of the inputs.

3.2 Classification of Relaying Algorithms

Relaying algorithms can be classified into the following two basic types:

- Non-Recursive Algorithms
- Recursive Algorithms

Non-recursive algorithms use a finite number of data samples to estimate the needed parameters. The output, therefore, solely depends on the value of samples. The time period that contains these samples is called a data window. A data window contains a collection of samples and, when a new sample becomes available, the oldest sample is discarded. Recursive algorithms use the recent samples of the inputs and previous outputs to estimate the needed parameters. Recursive Kalman filtering and recursive Least Squares have been successfully used in numerical relays for power system protection. This section presents only the non-recursive algorithms used in numerical relays.

3.2.1 Non-Recursive Algorithms

Non-recursive algorithms can be classified by the number of samples in a data window as follows:

- Short window algorithms that use a window of less than one half-cycle of the fundamental frequency.

- Long window algorithms that use a window of more than one half-cycle of the fundamental frequency.

Generally, these algorithms are classified in the literature [2] based on the mathematical principles that are used to calculate the phasors. Some of the classifications are as follows:

- Trigonometric algorithms
- Correlation algorithms
- Least Error Square algorithms

3.3 Trigonometric Algorithms

The trigonometric algorithms assume that the waveform of the input voltage or current is a sinusoid of the nominal frequency and the frequency of the input signal is invariant. These assumptions are not generally valid, particularly during a fault and for some time after the fault is cleared. When these algorithms are used, the input signals are filtered, by using analog low-pass filters, to eliminate the components of the non-fundamental frequencies and noise. The outputs of analog filters that are sinusoidal waveforms of the fundamental frequency can be expressed as follows:

$$v = V_p \sin(\omega_0 t + \theta_v) \tag{3.1}$$

where,

v is the instantaneous value of the voltage,

V_p is the peak value of the sinusoidal voltage,

ω_0 is the nominal frequency of the power system,

t is the time in seconds, and

θ_v is the phase angle of voltage phasors at t = 0.

The first and the second derivatives of the voltage can be expressed by the following equations

$$v' = \frac{dv}{dt} = \omega_0 V_p \cos(\omega_0 t + \theta_v) \tag{3.2}$$

$$v'' = \frac{d^2 v}{dt^2} = -V_p \omega_0^2 \sin(\omega_0 t + \theta_v) \tag{3.3}$$

All trigonometric algorithms are designed to predict either the peak or the squared peak of the input waveforms and are based on the above equations. Some of the trigonometric algorithms are as follows:

1. Mikki and Mikano Algorithm
2. Mann and Morrison Algorithm
3. Rockefeller and Udren Algorithm

3.3.1 Mikki and Mikano Algorithm

The Mikki and Mikano [13] algorithm uses two samples of the signal to determine the peak value and the phase angle. Consider the voltage waveform represented by Equation 3.1, which is sampled at ΔT s intervals. Two samples taken at time t = 0 and t = - ΔT can be expressed as follows:

v_{-1} sample taken at time t = -ΔT s

v_0 sample taken at time t = 0

The sample taken at time t = -ΔT can be expressed as

$$v_{-1} = V_p \sin(-\omega\Delta T + \theta_v), \tag{3.4}$$

and the sample taken at time t = 0 can be expressed as

$$v_0 = V_p \sin(\theta_v) \tag{3.5}$$

Equation 3.4 can be expanded by using the trigonometric identity, *sin (x-y) =sin (x) cos (y)-cos (x) sin (y)*, as follows:

$$v_{-1} = V_p \sin(\theta_v)\cos(\omega_0\Delta T) - V_p \cos(\theta_v)\sin(\omega_0\Delta T) \tag{3.6}$$

Substituting for $V_p \sin(\theta)$ from Equation 3.5 provides

$$v_{-1} = v_0 \cos(\omega_0\Delta T) - V_p \cos(\theta_v)\sin(\omega_0\Delta T) \tag{3.7}$$

Rearranging this equation provides

$$V_p \cos(\theta_v) = \frac{v_0 \cos(\omega_0 \Delta T) - v_{-1}}{\sin(\omega_0 \Delta T)} \tag{3.8}$$

Equation 3.5 provides the imaginary part of the phasors and Equation 3.8 provides the real part of the phasor. Adding the squares of Equations 3.5 and 3.8, the peak value of the voltage is obtained as follows:

$$V_p^2 = (v_0)^2 + \left(\frac{v_0 \cos(\omega_0 \Delta T) - v_{-1}}{\sin(\omega_0 \Delta T)} \right)^2 \tag{3.9}$$

The phase angle of the phasors can be obtained by dividing Equation 3.5 by Equation 3.8 as follows:

$$\tan \theta_v = \frac{v_0 \sin(\omega_0 \Delta T)}{v_0 \cos(\omega_0 \Delta T) - v_{-1}} \tag{3.10}$$

In Equations 3.9 and 3.10, v_0 and v_{-1} are the measured values of the two samples, ω_0 the nominal frequency of the system and ΔT is the inverse of the selected sampling rate.

3.3.2 Mann and Morrison Algorithm

The Mann and Morrison [14] algorithm uses three samples and the first derivatives of the signal to determine the peak value and the phase angle.

The first derivative defined by Equation 3.2 at time t = 0 is as follows.

$$\left. \frac{dv}{dt} \right|_{t=0} = \omega_0 V_p \cos(\theta_v) \tag{3.11}$$

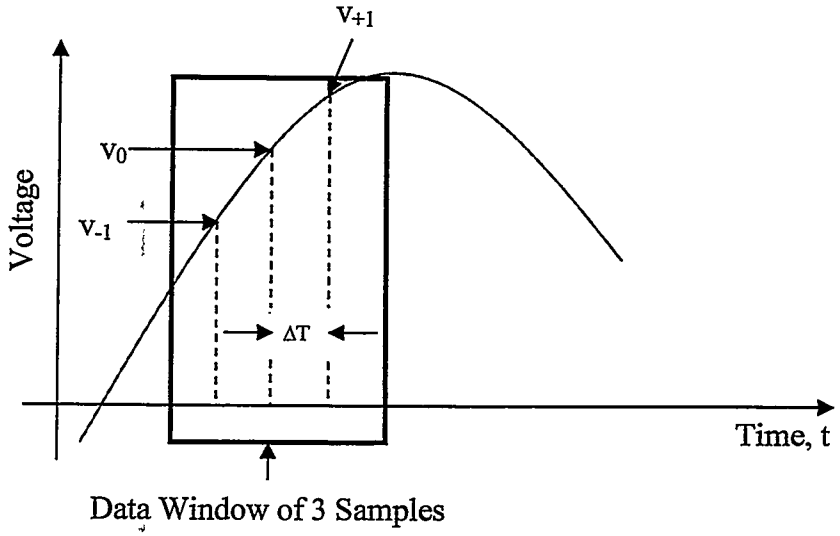


Figure 3.1 Data window of three samples

Consider the three samples of an input waveform, shown in Figure 3.1. The first derivative is approximately given by the following equation

$$\left.\frac{dv}{dt}\right|_{t=0} = \frac{v_{+1} - v_{-1}}{2\Delta T} \tag{3.12}$$

Equating the right hand side of Equations 3.11 and 3.12 and re-arranging provides the real component of the phasor as follows:

$$V_p \cos\theta_v = \frac{v_{+1} - v_{-1}}{2\omega_0 \Delta T} \tag{3.13}$$

The imaginary part of the phasor is given by Equation 3.5, which is as follows:

$$v_0 = V_p \sin(\theta_v)$$

Adding the squares of Equations 3.5 and 3.13, the square of the peak voltage is obtained as follows:

$$V_p^2 = (v_0)^2 + \left(\frac{v_{+1} - v_{-1}}{2\omega_0 \Delta T} \right) \tag{3.14}$$

The phase angle of the phasors can be obtained by dividing Equation 3.5 by Equation 3.13 as follows:

$$\tan\theta_v = \frac{v_0 (2\omega_0 \Delta T)}{v_{+1} - v_{-1}} \tag{3.15}$$

3.3.3 Rockefeller and Udren Algorithm

The Rockefeller and Udren algorithm [9] uses three samples and the first and the second derivatives of the input signal. Rearranging Equations 3.2 and 3.3 and substituting t = 0 provides the following equations

$$V_p \cos(\theta_v) = \frac{v'}{\omega_0} \tag{3.16}$$

$$V_p \sin(\theta_v) = -\frac{v''}{\omega_0} \tag{3.17}$$

The first derivative is given by Equation 3.11 when three consecutive samples are used. It can be shown that the second derivative at t = 0 is approximately given by the following equation

$$v'' = \frac{v_{+1} - 2v_0 + v_{-1}}{(\Delta T)^2} \tag{3.18}$$

As previously shown, the real component of the phasor is given by Equation 3.14 which is as follows:

$$V_p \cos(\theta_v) = v' = \frac{v_{+1} - v_{-1}}{\omega_o \Delta T} \tag{3.19}$$

Substituting Equation 3.17 in Equation 3.18 provides the imaginary component of the phasors as follows:

$$V_p \sin(\theta_v) = v'' = \frac{v_{+1} - 2v_0 + v_{-1}}{(\Delta T)^2} \tag{3.20}$$

Adding the squares of Equations 3.19 and 3.20 the square of the peak voltage is obtained as follows:

$$V_p^2 = \left(\frac{v_{+1} - v_{-1}}{\omega_0 \Delta T} \right)^2 + \left(\frac{v_{+1} - 2v_0 + v_{-1}}{\omega_0^2 (\Delta T)^2} \right)^2 \tag{3.21}$$

The phase angle of the phasor can be obtained by dividing Equation 3.20 by Equation 3.19 as follows:

$$\tan\theta_v = -2\left(\frac{v_{+1} - 2v_0 + v_{-1}}{(v_{+1} - v_{-1})(\omega_0 \Delta T)} \right) \tag{3.22}$$

A review of Equations 3.13 and 3.19 shows; that the presence of a constant DC component has no effect on the estimates of the phasors because it is eliminated by the subtractions.

3.3.4 Comments about Trigonometric Algorithms

1. All algorithms discussed in this section use two or three samples. They, therefore, respond rapidly to changes in the inputs.
2. Implementation of trigonometric algorithms requires few computations.
3. Presence of decaying DC adversely affects the outputs of the Mikki and Mikano and Mann and Morrison algorithms.
4. Trigonometric algorithms amplify noise.

3.3.5 Analysis with Mann and Morrison Algorithm

Consider that the sampling frequency is 720 Hz and the fundamental frequency is 60 Hz. The real part of the filter is calculated by using Equation 3.13, which is as follows:

$$\begin{aligned} V_p \cos\theta_v &= \frac{v_{+1} - v_{-1}}{2\omega_0 \Delta T} \\ &= \frac{v_{+1} - v_{-1}}{1.0472} \\ &= 0.9549 v_{+1} - 0.9549 v_{-1} \end{aligned} \tag{3.23}$$

The coefficients of the real filter, therefore, are 0.9549, 0 and -0.9549.

The frequency response is determined by first writing this equation in terms of z-transform as follows

$$H_{real}(z) = 0.9549(z^{+1} - z^{-1}) \tag{3.24}$$

and then replacing z with $e^{j\omega\Delta T}$ in this equation. This provides the following transfer function in the frequency domain.

$$H_{real}(\omega) = 0.9549(j2\sin\omega\Delta T) \tag{3.25}$$

The imaginary part of the filter is given by Equation 3.5 which is as follows.

$$v_0 = V_p \sin(\theta_v)$$

The coefficient of the function is 1.

$$H_{imag}(\omega) = 1 \tag{3.26}$$

The frequency responses of the transfer functions are calculated by assigning different values to the frequency and determining the value of each transfer function. These are shown in Figures 3.2 and 3.3.

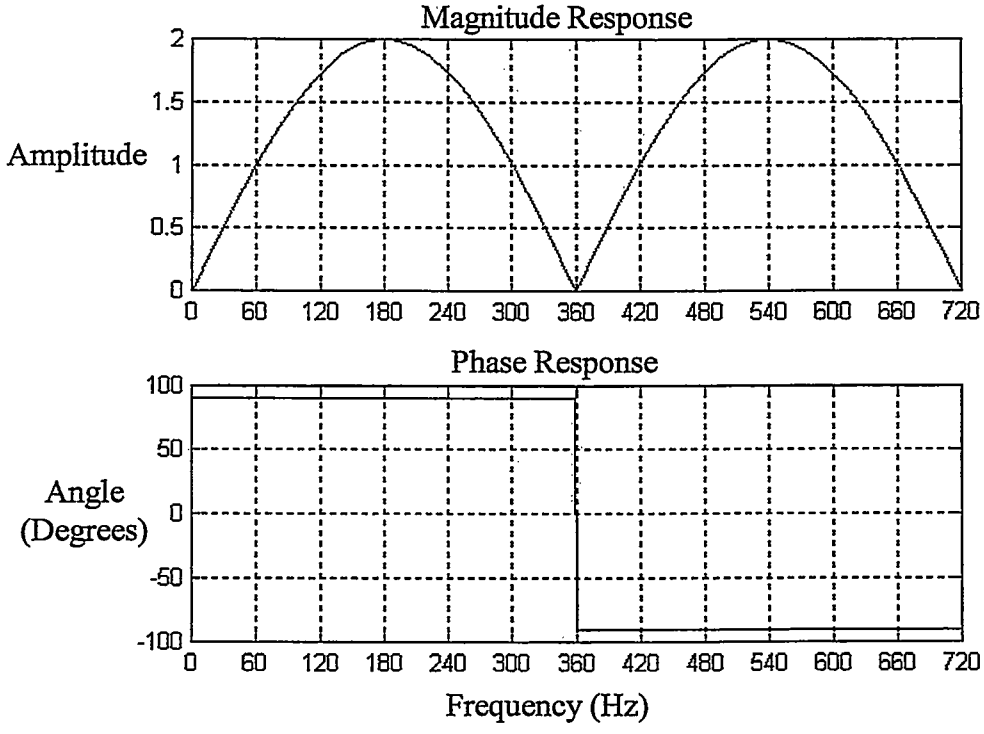


Figure 3.2 Frequency response of Mann and Morrison real part filter

Figure 3.2 shows the frequency response of the Mann and Morrison algorithm that calculates the real part of the phasor. The following observations can be made from this response:

- Passes 60 Hz component without attenuation,
- Amplifies the 2nd, 3rd and 4th harmonics, and
- Phase shifts all frequencies by 90°.

Figure 3.3 shows the frequency response of the Mann and Morrison algorithm that calculates the imaginary part of the phasor.

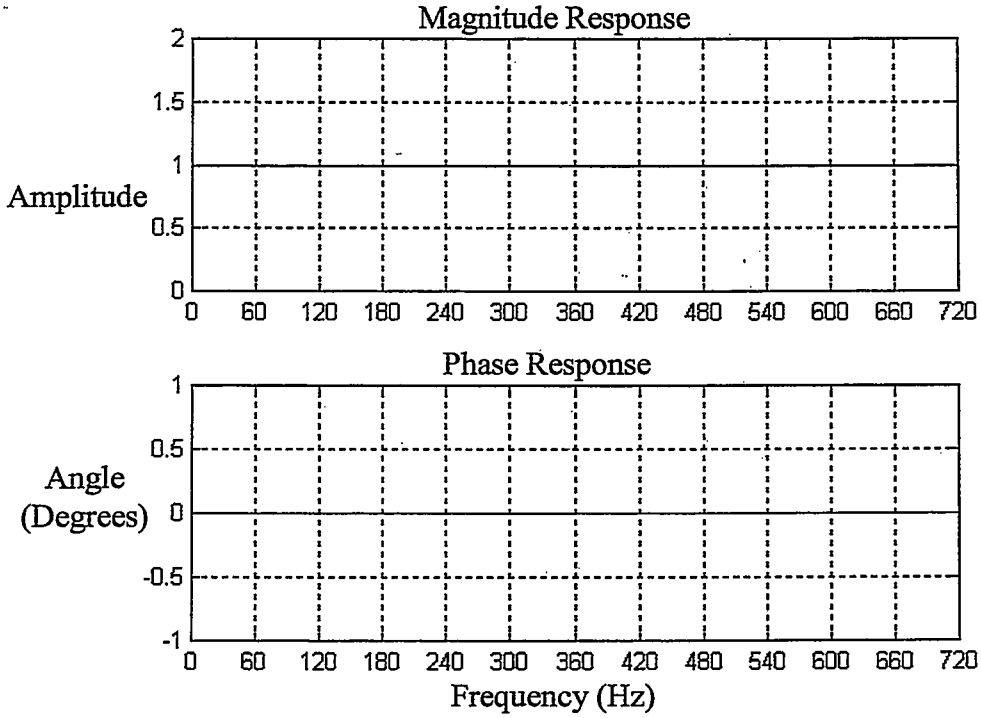


Figure 3.3 Frequency response of Mann and Morrison imaginary part filter

Figure 3.3 shows, that the imaginary component of the Mann and Morrison algorithm passes all the frequencies without attenuating, amplifying or phase-shifting them.

The Mann and Morrison algorithm provides accurate results when the input signal is of the nominal frequency only. However, when an input signal contains harmonics or a DC component, the output becomes inaccurate.

3.4 Correlation Algorithms

Correlation algorithms correlate the input waveform with a set of orthogonal functions to extract components of the selected frequencies [15]. A pair of functions is orthogonal if the integration of their product over one time period is zero. This can be mathematically expressed as follows.

$$\int_{t=0}^{T} f(\omega t)g(\omega t)dt = 0 \tag{3.27}$$

where,

T is the time period for one cycle.

Commonly used orthogonal functions in numerical relays are the sine and cosine waveforms. Early designs of numerical relays used even-and-odd rectangular waves.

3.4.1 Sine and Cosine Waveforms

Let $f(t) = \sin(\omega_0 t)$, (3.28)

$$g(t) = \cos(\omega_0 t) \tag{3.29}$$

$$T = \frac{2\pi}{\omega_0} \tag{3.30}$$

Equation 3.27 can be used to check if the sine and cosine waveforms defined by Equations 3.28 and 3.29 are orthogonal. Substituting Equation 3.28 and 3.29 in Equation 3.27 provides the following equation.

$$\int_0^{2\pi}\sin(\omega_0 t)\cos(\omega_0 t)d(\omega_0 t)=\frac{1}{4}\int_0^{2\pi}\sin(2\omega_0 t)d(2\omega_0 t)$$

$$=\frac{1}{4}\left[-\cos(2\omega_0 t)\right]_0^{2\pi}=0 \qquad (3.31)$$

The sine and cosine functions are, therefore, orthogonal and can be used to correlate with input waveforms. Basic steps for correlating the input with sine and cosine functions are as follows.

Consider that the input voltage is $v=V_p\sin(\omega_0 t+\theta_v)$. Correlating the input waveform with $\sin(\omega_0 t)$ provides the following equation.

$$\int_0^{2\pi}V_p\sin(\omega_0 t+\theta_v)\sin(\omega_0 t)d(\omega_0 t)$$

$$=V_p\int_0^{2\pi}\sin(\omega_0 t)\cos(\theta_v)d(\omega_0 t)+V_p\int_0^{2\pi}\sin(\theta_v)\cos(\omega_0 t)d(\omega_0 t)$$

$$=V_p\cos(\theta_v)\int_0^{2\pi}\sin^2(\omega_0 t)d(\omega_0 t)+V_p\left(\frac{\sin(\theta_v)}{2}\right)\int_0^{2\pi}\sin(2\omega_0 t)d(2\omega_0 t) \qquad (3.32)$$

Because $\int_0^{2\pi}\sin(2\omega_0 t)d(2\omega_0 t)$ is zero, Equation 3.33 gives the real part of the phasor as shown by following equation

$$V_p\cos(\theta_v)=\frac{1}{\pi}\int V_p\sin(\omega_0 t+\theta_v)\sin(\omega_0 t)d(\omega_0 t) \qquad (3.3)3$$

Similarly correlating the input waveform with $\cos(\omega_0 t)$ gives the imaginary part of the phasor as follows:

$$V_p \sin(\theta_v) = \frac{1}{\pi} \int V_p \sin(\omega_0 t + \theta_v) \cos(\omega_0 t) d(\omega_0 t) \tag{3.34}$$

In numerical relays, the input signals are available at discrete instants of time, and therefore, the numerical techniques are used to evaluate the integrations. The reference sine and cosine waveforms sampled at the selected frequency are as follows:

$$\sin\left(\frac{2\pi n}{m}\right) \tag{3.35}$$

$$\cos\left(\frac{2\pi n}{m}\right) \tag{3.36}$$

where,

m is the total number of samples in one cycle, and

n varies from 0 to (m-1).

The real and imaginary parts of the phasors can be found by correlating the sampled reference waveforms with the input signal in the form of Equations 3.37 and 3.38.

$$V_r(k) = \frac{2}{m} \sum_{n=0}^{m-1} V_{k+n-m+1} \sin\left(\frac{2\pi n}{m}\right) \tag{3.37}$$

$$V_i(k) = \frac{2}{m} \sum V_{k+n-m+1} \cos\left(\frac{2\pi n}{m}\right) \tag{3.38}$$

where,

k is the reference sample

The peak value and phase angle of the phasors can be determined as follows.

$$V_p = \sqrt{V_r^2 + V_i^2} \tag{3.39}$$

$$\theta_v = \arctan\left(\frac{V_i}{V_r}\right) \tag{3.40}$$

The procedure described above is referred as the Fourier algorithm and its implementation is shown in Figure 3.4. Some of the algorithms that use the Fourier type technique are as follows:

1. Discrete Fourier Transform (DFT) Algorithm
2. Half-Cycle DFT Algorithm
3. Cosine Filters

The DFT algorithm is described in brief to show how correlation algorithms work. Consider that the Sampling Frequency is 720 Hz and the fundamental frequency is 60 Hz. The reference sine and cosine wave coefficients are given in Table 3.1. The reference coefficients are taken starting from 15^0. This gives 12 samples per cycle.

Table 3.1 Reference Sine and Cosine coefficients for DFT

Reference Sine Coefficients	Reference Cosine Coefficients
0.2588	0.9659
0.7071	0.7071
0.9659	0.2588
0.9659	-0.2588
0.7071	-0.7071
0.2588	-0.9659
-0.2588	-0.9659
-0.7071	-0.7071
-0.9659	0.2588
-0.9659	0.2588
-0.7071	0.7071
-0.2588	0.9659

The sine reference coefficients are correlated with the input signal to obtain the real parts of the phasors, while the cosine reference coefficients are correlated with the input signal to obtain the imaginary parts of the phasors. The part which computes the real part of phasors, $V_p \cos(\theta_v)$, is known as the cosine filter and the part which computes the imaginary part of phasors, $V_p \sin(\theta_v)$, is known as the sine filter. Figures 3.5 and 3.6; shows the frequency response of the DFT sine and cosine filters respectively. These figures show that both the sine and cosine filters:

- Pass 60 Hz component without any attenuation,
- Remove the non-decaying DC component, and
- Attenuate all the harmonics

The disadvantages of the DFT algorithm are as follows:

1. Transient response is slower when compared to short window algorithms
2. Decaying part of DC component effects accuracy
3. Number of computations required is higher when compared to the short window algorithms

3.4.2 Rectangular Waveform Algorithms

Another group of functions that can be used in the correlation algorithms are even and odd rectangular waves [15]. These waves are shown in Figure 3.7 and can be mathematically expressed as follows:

$$W_r(t) = signum(\sin(\omega_0 t)) \tag{3.41}$$

$$W_i(t) = signum(\cos(\omega_0 t)) \tag{3.42}$$

where,

$signum = -1$ for x<0,

$= 0$ for x=0,

$= +1$ for x>0,

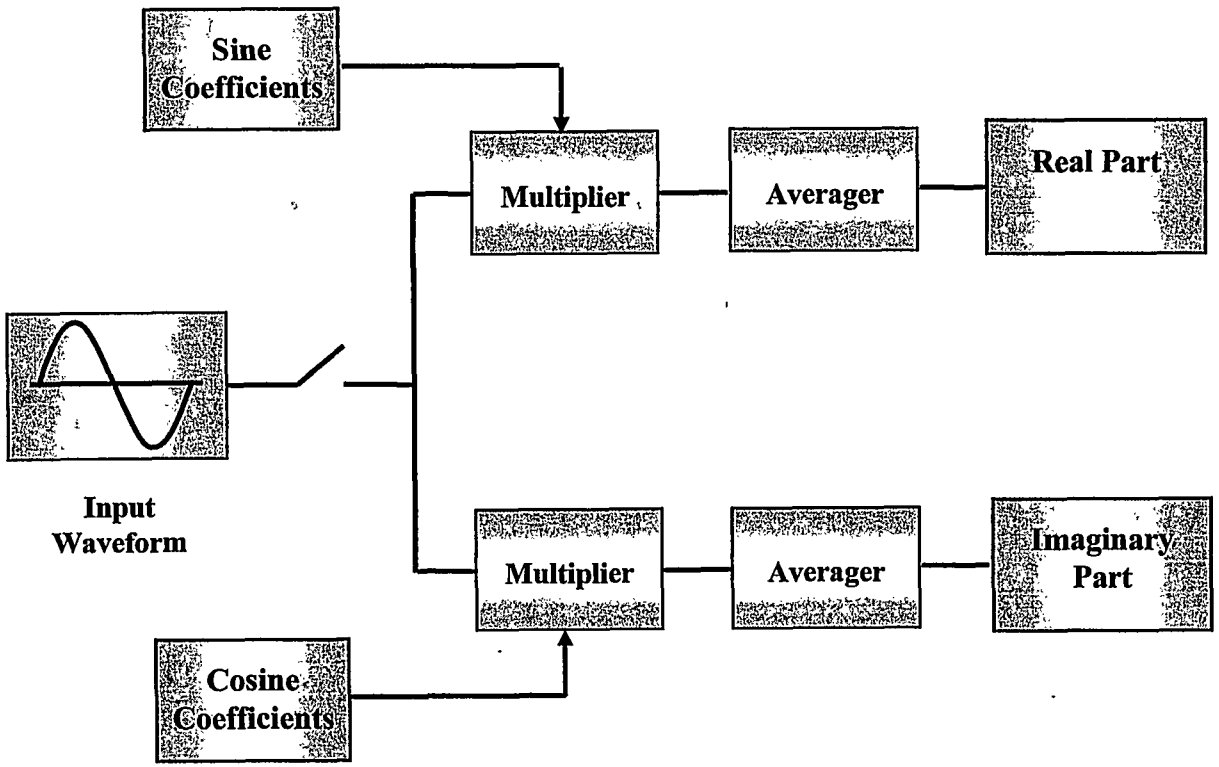


Figure 3.4 Schematic representation of Fourier Algorithm

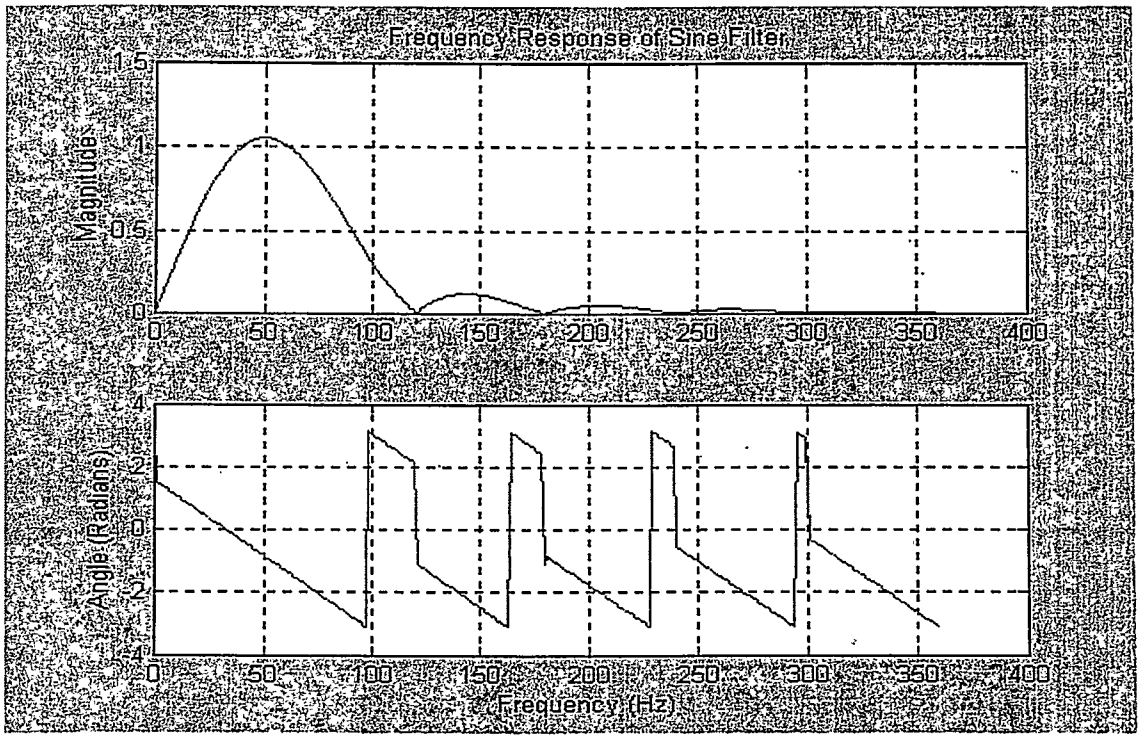


Figure 3.5 Frequency response of DFT sine filter

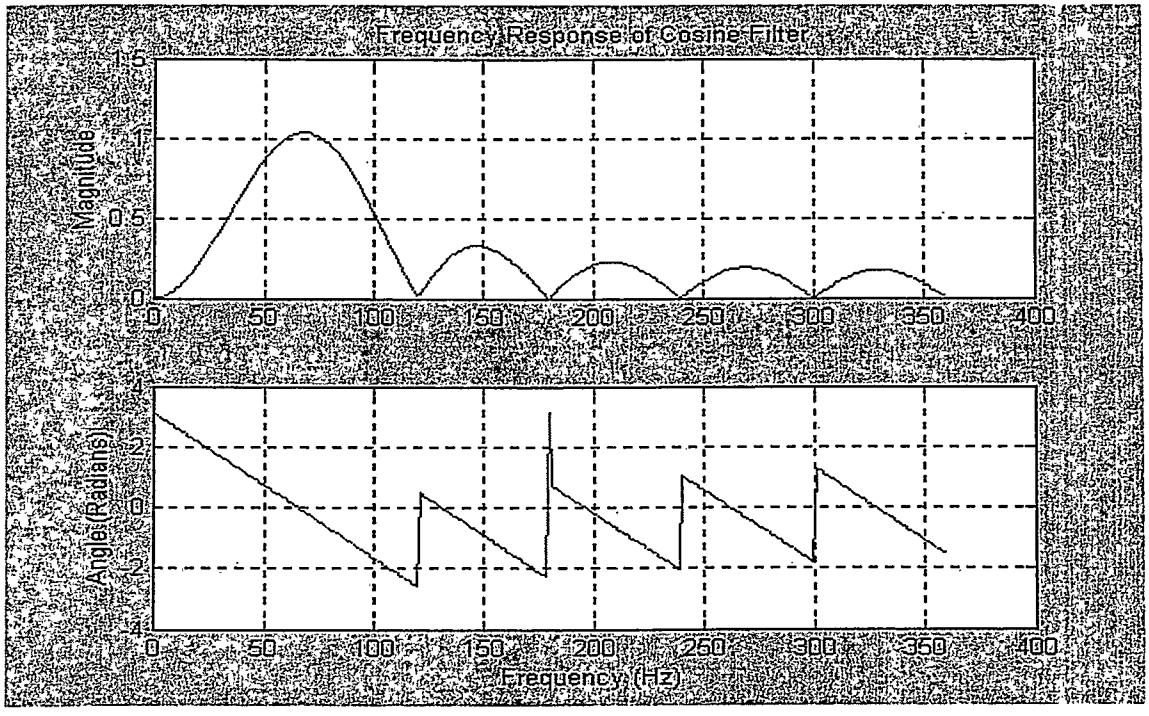


Figure 3.6 Frequency response of DFT cosine filter

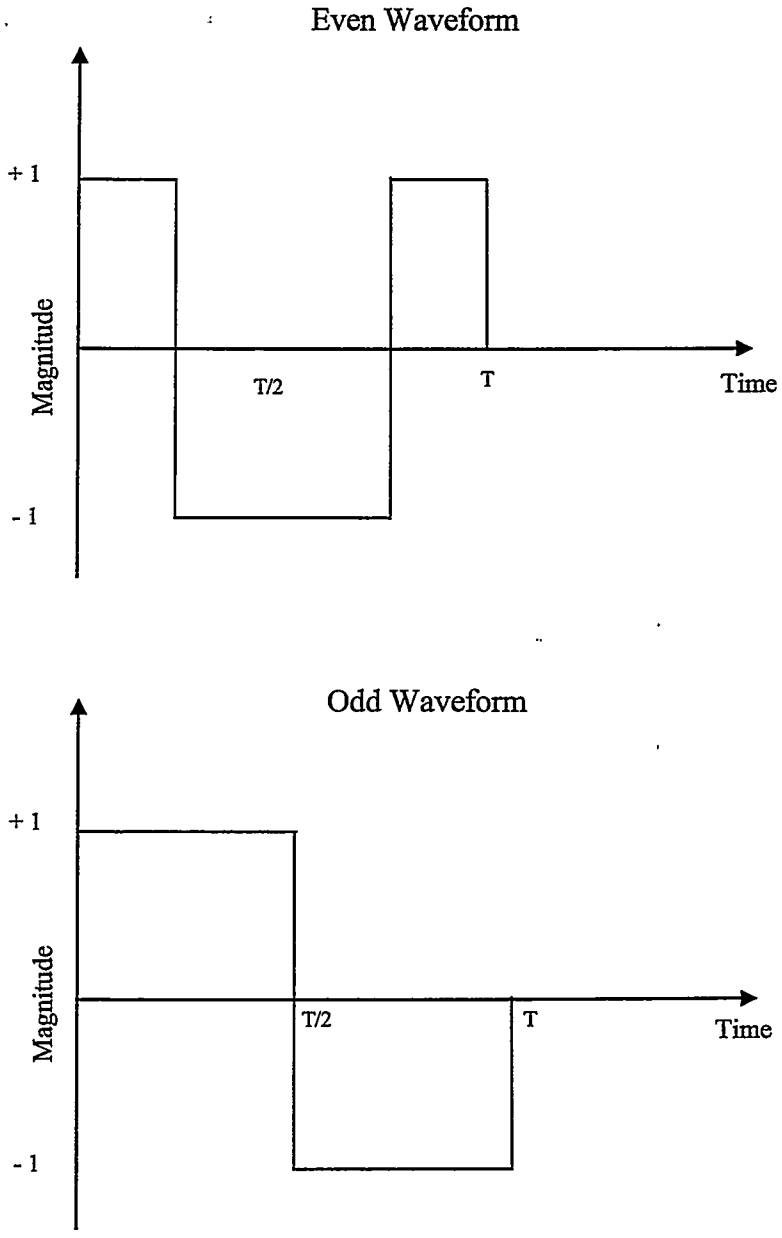


Figure 3.7: Even and odd rectangular waves

$W_r(t)$ is the odd rectangular wave, and

$W_i(t)$ is the even rectangular wave

The real and imaginary parts of phasors of a frequency can be calculated by using the following equations

$$V_r = \left(\frac{1}{A}\right)\sum_{n=0}^{m-1} V_{k+n-m+1}\, signum\left(\sin\left(\frac{2\pi n}{m}\right)\right) \tag{3.43}$$

$$V_i = \left(\frac{1}{A}\right)\sum_{n=0}^{m-1} V_{k+n-m+1}\, signum\left(\cos\left(\frac{2\pi n}{m}\right)\right) \tag{3.44}$$

where,

A is the scaling factor which must be calculated.

Example: Consider that the sampling frequency is 720 Hz and the fundamental frequency is 60 Hz.

The coefficients of the algorithm, W_r, for calculating the real part of the phasors are as follows.

1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1

The coefficients of the algorithm, W_i, for calculating the imaginary part of the phasors are as follows.

1, 1, 1, -1, -1, -1, -1, -1, -1, 1, 1, 1

- Consider that a sine wave of 1.0 peak value is sampled at 720 Hz and there are twelve samples in a window. The samples are

  0, 0.5, 0.866, 1.0, 0.866, 0.5, 0, -0.5, -0.866, -1.0, -0.866, -0.5

- The real part of the voltage phasor, V_r, obtained by multiplying the samples with the W_r coefficients and summing the results is

$$V_r = 0 + 0.5 + 0.866 + 1.0 + 0.866 + 0.5 - 0 + 0.5 + 0.866 + 1.0 + 0.866 + 0.5 = 7.464$$

- The imaginary of the voltage phasor, V_i, obtained by multiplying these samples with the W_i coefficients and summing the results is

$$V_i = 0 + 0.5 + 0.866 - 1.0 - 0.866 - 0.5 - 0 + 0.5 + 0.866 - 1.0 - 0.866 - 0.5 = -2.0$$

- $A = \sqrt{(V_R^2 + V_I^2)} = 7.727$

Numerical relays used rectangular waves originally to reduce the computational burden on the processors by avoiding multiplications.

Some of the algorithms that use rectangular waveforms are as follows

1. Even and Odd Functions Algorithm
2. Walsh Functions Algorithm

3.5 Least Error Squares Algorithm

The Least Error Squares algorithm fits the faulted current and voltage waveforms to a model of the inputs that may consist of the sinusoidal waveforms of the fundamental and harmonic frequencies and a decaying DC component. The basic steps to implement this algorithm are as follows [16].

1. Select a suitable model for representing a signal,
2. Linearize the model,

3. Select a sampling rate and data window size,
4. Express the process in matrix form,
5. Determine the left pseudo inverse of the coefficient matrix, and
6. Calculate the components of the phasors

Assume that the input signal, a voltage waveform $v(t)$, consists of a fundamental component, an exponentially decaying DC component, and (N-1) harmonics. The voltage signal can be expressed by a mathematical model of the form of Equation 3.45.

$$v(t) = K_0 e^{-t/\tau} + \sum_{m=1}^{N} K_m \sin(m\omega_0 t + \theta_m) \qquad (3.45)$$

where,

K_0 is the magnitude of the decaying DC component at t=0,

τ is the time constant of the decaying DC component,

K_m is the magnitude of the m^{th} harmonic component,

θ_m is the phase angle of the m^{th} harmonic component,

ω_0 is the angular frequency of the fundamental frequency component, and

N is the highest harmonic considered.

Assuming that the voltage contains a decaying DC component, the fundamental frequency and the third harmonic components only, the Equation 3.45 can be written as follows:

$$v(t) = K_0 e^{-t/\tau} + K_1 \sin(\omega_0 t + \theta_1) + K_3 \sin(3\omega_0 t + \theta_3) \qquad (3.46)$$

The exponential term $e^{-t/\tau}$ can be expanded by using the Taylor series. A reasonable approximation of the exponential is obtained by using the first two terms of the series. This approximation provides the following representation

$$e^{-t/\tau} = 1 - \frac{t}{\tau} \tag{3.47}$$

The fundamental and third harmonic components can be expanded using the trigonometric identity, $\sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y)$, as follows:

$$K_1 \sin(\omega_0 t + \theta_1) = K_1 \sin(\omega_0 t)\cos(\theta_1) + K_1 \cos(\omega_0 t)\sin(\theta_1)$$

$$K_3 \sin(3\omega_0 t + \theta_3) = K_3 \sin(3\omega_0 t)\cos(\theta_3) + K_3 \cos(3\omega_0 t)\sin(\theta_3)$$

Making these substitutions in Equation 3.46 provides the following equation

$$v(t) = K_0 - \left(\frac{K_0}{\tau}\right)t + (K_1 \cos\theta_1)\sin(\omega_0 t) + (K_1 \sin\theta_1)\cos(\omega_0 t)$$

$$+ (K_3 \cos\theta_3)\sin(3\omega_0 t) + (K_3 \sin\theta_3)\cos(3\omega_0 t) \tag{3.48}$$

At time $t = t_1$, Equation 3.48 can be written as follows:

$$v(t_1) = K_0 - \left(\frac{K_0}{\tau}\right)t_1 + (K_1 \cos\theta_1)\sin(\omega_0 t_1) + (K_1 \sin\theta_1)\cos(\omega_0 t_1)$$

$$+ (K_3 \cos\theta_3)\sin(3\omega_0 t_1) + (K_3 \sin\theta_3)\cos(3\omega_0 t_1) \tag{3.49}$$

This equation has six unknowns, namely, the constant part of the DC, the decaying part of the DC and the real and imaginary part of the fundamental frequency and third harmonic frequency phasors. Because there are six unknowns, at least seven samples are needed to find the least squares solution. Equation 3.49 can be written in a more convenient form as follows:

$$v(t) = a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + a_{15}x_5 + a_{16}x_6 \tag{3.50}$$

where,

x_1	$=K_0$		a_{11}	$=1$
x_2	$=-K_0/\tau$		a_{12}	$=t_1$
x_3	$=K_1\cos(\theta_1)$		a_{13}	$=\sin(\omega_0 t_1)$
x_4	$=K_1\sin(\theta_1)$		a_{14}	$=\cos(\omega_0 t_1)$
x_5	$=K_3\cos(\theta_3)$		a_{15}	$=\sin(3\omega_0 t_1)$
x_6	$=K_3\sin(\theta_3)$		a_{16}	$=\cos(3\omega_0 t_1)$

The 'a' coefficients of Equation 3.50 are functions of time at which samples are taken. If t_1 is taken as a time reference and voltage is sampled at a pre-selected sampling rate, the values of the 'a' coefficients can be specified and computed off-line.

Assume that the voltage is sampled at intervals of ΔT seconds, the next sample received at time $t_2 = (t_1+\Delta T)$ can be described by the following equation.

$$v(t) = v(t_1 + \Delta T) = a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + a_{25}x_5 + a_{26}x_6 \tag{3.51}$$

Proceeding in this manner, m equations can be generated for m samples of voltages. These equations can be represented in matrix form as follows.

$$\underset{m\times 7}{[A]}\underset{7\times 1}{[X]} = \underset{m\times 1}{[V]} \tag{3.52}$$

The solution for Equation 3.52 can be found by the concept of pseudo-inverse. This involves multiplying both sides of Equation 3.52 by the left pseudo-inverse of A, which gives the following equation.

$$\underset{7\times 1}{[X]} = \underset{7\times m}{[A]^{+}}\underset{m\times 1}{[V]} \tag{3.53}$$

where,

$[A]^+$ is the left pseudo inverse of [A]

The left pseudo-inverse is defined as $\underset{7\times m}{[A]} = \underset{7\times m\ m\times 7}{[(A^T)A]^{-1}}\ \underset{7\times m}{[A]}$

The elements of the [A] matrix and its left pseudo-inverse can be calculated in the off-line mode and then the left-pseudo inverse of [A] can be used in the on-line mode. This reduces substantially the on-line calculations necessary for estimating the real and imaginary components of the fundamental and harmonic frequency phasors.

The advantages of using the Least Squares algorithm are as follows:

1. It suppresses noise
2. The calculations of the fundamental frequency phasors are not affected by the presence of harmonics
3. The calculations of the fundamental frequency phasors are not affected by the presence of a decaying DC component

The disadvantages of this algorithm are as follows:

1. The transient response is slow
2. Compared with short window algorithms and some Fourier algorithms, it requires more computations

3.6 Summary

This chapter has provided an overview of numerical relay algorithms. The mathematical equations describing the Trigonometric, Correlation and the Least Error Squares algorithms have been presented. The assumptions made in each algorithm have been outlined. The use of algorithms in numerical relays has been described and the advantages and disadvantages of the algorithms have been outlined.

CHAPTER 4

NUMERICAL RELAY DESIGN CONSIDERATIONS

4.1 Introduction

The major functional blocks of a numerical relay are presented in Chapter 2 and numerical relay algorithms are described in Chapter 3. Factors which govern the selection of relaying algorithms and the hardware for the analog input and analog interface subsystems used for this project are discussed in this chapter. These factors have an impact on the performance of algorithms that convert sampled and quantized data to phasors. The factors considered and taken into consideration in this project are as follows:

1) The procedure followed in the selection of an algorithm.
2) The procedure followed in the selection of the hardware.

These procedures and the manner in which they are implemented are shown in Figure 4.1.

4.2 Procedure followed for Selecting an Algorithm

Some relaying algorithms use short-data windows of two to five consecutive samples. These algorithms, therefore, require very few computations; but they do not provide accurate estimates of the phasors. Long-data window algorithms use eight to thirty-two samples. They require more computations but provide accurate estimates of the phasors. Speed and accuracy of relaying algorithms are interdependent; the algorithms that present more accurate results use larger data windows and, hence, take more time for calculating the phasors. A speed and accuracy tradeoff, therefore, impacts these factors; for this reason, these factors are examined when an algorithm is selected for an application [14].

The algorithm selected for this project was the Discrete Fourier Transform (DFT) and it was decided to verify its response when different sampling frequencies are used.

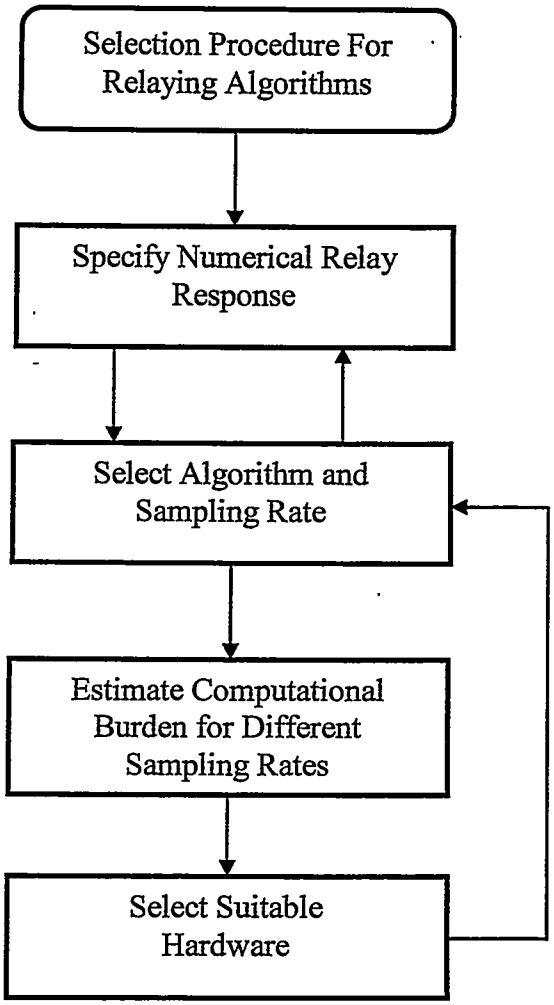


Figure 4.1 Selection procedures for relay algorithms

4.2.1 Computational Burden

The computational burden, which depends on the number of samples it takes to estimate the phasors and to make decisions from those estimates, associated with the DFT algorithm was estimated for various sampling rates.

The DFT algorithm is implemented by using the following equations,

$$V_r(k) = \frac{2}{m}\sum_{n=0}^{m-1} V_{k+n-m+1} \sin\left(\frac{2\pi n}{m}\right) \tag{4.1}$$

$$V_i(k) = \frac{2}{m}\sum V_{k+n-m+1} \cos\left(\frac{2\pi n}{m}\right) \tag{4.2}$$

where,

k is the reference sample and

m is the total number of samples in one cycle of the fundamental frequency

Equations 4.1 and 4.2 calculate the phasors of power system voltages and currents in the form of their real and imaginary components. The implementation of the DFT algorithm was described in Chapter 3. The DFT algorithm requires m-multiplications, (m-1) additions/subtractions and one division for calculating each component of a phasor.

The input power system data was sampled at different sampling frequencies. The sampling frequency (F_s) specifies how many equally spaced samples per second are taken for computations and this determines how many samples there will be in one cycle of the fundamental frequency (F_0) of an input waveform. The number of samples per cycle of the fundamental frequency is computed as follows:

$$Samples\_per\_Cycle = \frac{Sampling\_Frequency}{Fundamental\_Frequency} \tag{4.3}$$

Consider that the sampling frequency is 720 Hz and the fundamental frequency is 60 Hz. The total number of computations for calculating a phasor is given in Table 4.1.

Table 4.1 DFT calculations for Real and Imaginary parts of the phasor

	Additions/Subtractions	Multiplications	Division	Computations
Real Part, V_r	11	12	1	24
Imaginary Part, V_i	11	12	1	24
Total Computations				48

The peak value and the phase angle of the phasors can be determined using Equations 4.4 and 4.5,

$$V_p = \sqrt{V_r^2 + V_i^2} \tag{4.4}$$

$$\theta_v = \arctan\left(\frac{V_i}{V_r}\right) \tag{4.5}$$

Four computations are needed for calculating the peak value and two computations are required for calculating the phase angle. Thus, the total number of computations required for a voltage or a current signal from one phase of the power system is 54 (48 + 4 + 2).

Numerical relays, when used for protecting a line or a transformer, can require simultaneous data from several inputs. These may include three voltages and three or four current from the three-phases of the power system. Considering seven input signals from the power system, the total number of computations required for one phase

is 378. Table 4.2 shows the computational burden when the DFT algorithm is applied to data sampled at different frequencies.

Table 4.2 Computational burden at different sampling frequencies

Sampling Frequency (Hz)	Number of Samples	Time Between Samples (ms)	Total No. of Computations for One Phase	Total No. of Computations for Seven Input Signals
720	12	1.388	54	378
1080	18	0.926	78	546
1440	24	0.694	102	714
1800	30	0.555	126	882
2160	36	0.463	150	1050
2520	42	0.397	174	1218
2880	48	0.347	198	1386
3240	54	0.308	222	1554
3600	60	0.277	246	1722

In addition to the time taken by the DFT for calculating the components of the phasor, time taken by the other components of the analog input subsystem should also be considered. Main analog input subsystem components are the anti-aliasing low-pass filters and the analog-to-digital converters. In Section 4.3.1, a detailed analysis of low-pass filters and analog-to-digital converters is presented and the analysis on the input power system signal through the analog input subsystem, to the computations done on the digital data by the selected processors at different sampling frequencies is described in Section 4.3.3.

4.3 Hardware Considerations

The computational burden provides the basis for the selection of hardware components. The two important hardware components of numerical relays are the analog input subsystem and the digital processor. These components should be selected considering the dynamic range of the voltages and currents, especially during power

system faults. Anti-aliasing low-pass filters and the ADC are two major components of the analog input subsystem; their performance has an impact on the design of numerical relays. The digital processors are selected keeping in mind the highest burden associated with the selected DFT algorithm. The selection of the hardware components is considered in this section. The analog input subsystem samples and quantizes the instantaneous values of power system voltages and currents, as previously described in Chapter 2. The numerical values are processed by the digital processors; as required by the selected relay algorithm.

4.3.1 Analog Input Subsystem

A numerical relay takes samples of voltage and/or current at regular intervals. There are two important concerns in the selection of the hardware components for the analog input subsystem. The first concern is the speed with which the ADC quantizes a sampled value. The second concern is the accuracy of the conversion process. Before a sampled value is quantized, the analog signal is filtered so that aliasing would not cause errors in the phasor estimates.

4.3.1.1 Need for Filtering Input Signals

Most of the relaying algorithms consider that the input voltages and currents are sinusoids of the fundamental frequency. If this were true, every algorithm suggested for relaying would work perfectly. But the voltages and currents presented to relay algorithms during a fault consist of decaying DC, fundamental frequency and high frequency components. Numerical relays, therefore, must filter the inputs to remove the unwanted components of the inputs but retain the components of interest that are present in the signals. This filtering requirement depends on the protection principle used in a particular numerical relay. For example, in most relays the power system frequency components are considered as information and everything else is considered as noise.

Depending on the rate at which the voltages and currents are sampled, some of the high frequency components can appear to be components of power system frequency, as described below.

Consider a component of a sinusoidal signal, of frequency f Hz, which is defined as

$$x_{hf}(t) = \sin(2\pi ft) \tag{4.6}$$

Now consider that this signal, $x_{hf}(t)$, is sampled at a rate of F_S samples per seconds. The inter-sampling time would be ΔT seconds, where $\Delta T = \frac{1}{F_s}$. The value of the n^{th} sample is given as follows:

$$x_{hf}(n) = \sin(2\pi fn\Delta T) \tag{4.7}$$

Sinusoids are periodic functions that repeat at intervals of 2π radians, i.e. $\sin(\theta) = \sin(\theta + 2\pi m)$, where m is any integer. Equation 4.7, therefore, can be written as follows:

$$\sin(2\pi fn\Delta T) = \sin(2\pi fn\Delta T + 2\pi m)$$

$$\sin(2\pi fn\Delta T) = \sin\left[2\pi(f + \frac{m}{n\Delta T})n\Delta T\right] \tag{4.8}$$

$$\sin(2\pi fn\Delta T) = \sin\left[2\pi(f + kF_s)n\Delta T\right] \tag{4.9}$$

Here k can be any positive or negative integer and $F_s = \frac{1}{\Delta T}$.

Equation 4.9 implies that when a signal is sampled at a rate of F_S samples per second, the sampled values of a sine wave of f Hz cannot be distinguished from sampled values of a sine wave of $(f + kF_s)$ Hz. For example, if voltage signal at $f = 600$ Hz is sampled at $F_s = 540$ Hz and k is -1, then

$$(f + kF_s) = (600 - 540) = 60 \text{ Hz}$$

Thus, in the absence of additional information, a 600 Hz signal appears to be a signal of 60 Hz frequency. This misrepresentation of the input signals increases the chances of the relays making incorrect decisions. This ambiguity in frequency is called aliasing.

Low pass filters are, therefore, used to remove components of frequencies of equal to or more than one half the sampling rates $\left(\frac{F_s}{2}\right)$. The frequency $\frac{F_s}{2}$ is referred to as Nyquist frequency [18].

4.3.1.2 Low-Pass Filter Requirements

A low-pass filter can be built using analog components and must possess certain characteristics. Some of the characteristics are as follows:

1) The bandwidth should be reasonable so that the delay in the filter is not excessive.
2) The transient response should be fast.
3) The delay in the Bandpass should be minimal.
4) The frequencies in the band-reject should be attenuated substantially.
5) The group delay should be minimal.
6) The design should be simple and easy to implement and manufacture.

The cutoff frequency of a filter is defined as the frequency at which the magnitude of the output is 0.707 times the output in the pass-band. Figure 4.2 shows the amplitude response of an ideal low-pass filter.

The group delay is an important characteristic for filters used in conjunction with numerical relays. The group delay, is defined as the negative derivative of a filters phase with respect to frequency, and is expressed as follows:

$$\tau_d = -\frac{d\phi(\omega)}{d\omega} \tag{4.10}$$

The group delay is different at different frequencies. As most of the relaying algorithms use only the fundamental frequency components of voltages and currents, group delay for the fundamental frequency should be considered while designing filters for numerical relays.

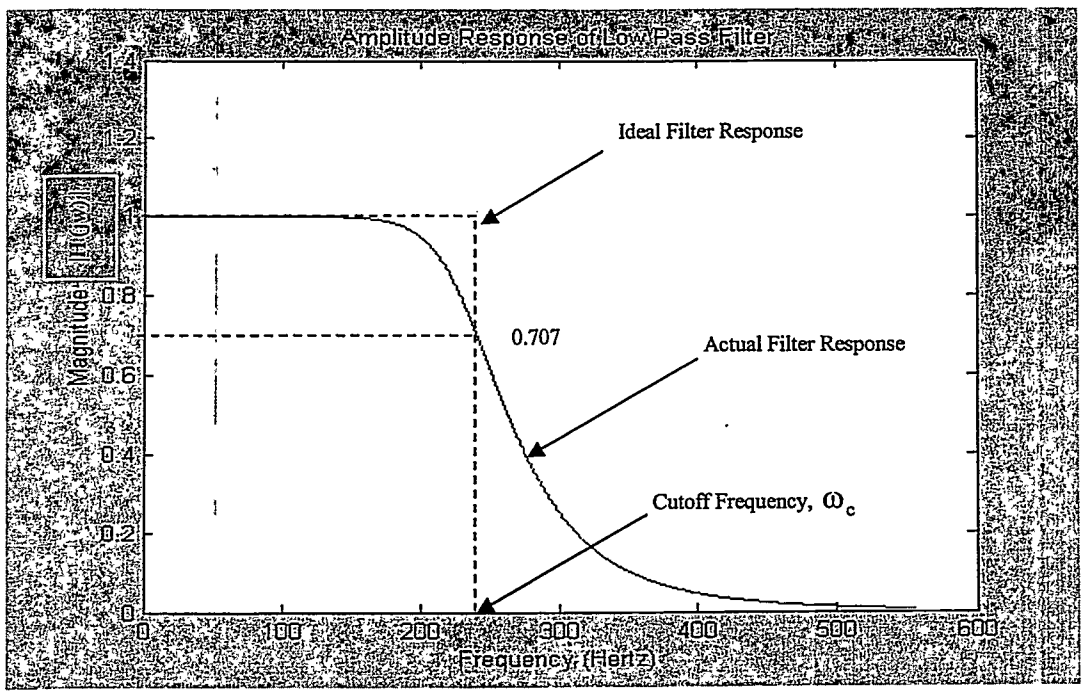


Figure 4.2 Response of low-pass filter

The filtered input power system signals are in analog form. Numerical relays require the input data in a digital form.

4.3.1.3 Analog-to-Digital Converters

The ADCs convert instantaneous values of analog inputs (power system voltages and currents) to equivalent numerical values. When selecting ADCs for numerical relays, speed of conversion is an important criterion. Accuracy of converted values, which depend on the number of bits of the ADC, is another important factor.

Many techniques are used for converting analog signals to corresponding numerical representations. Some of ADC conversion techniques are counter-ramp, integration, successive-approximation, and parallel and its variations.

The counter-ramp and tracking type converters are not suitable for numerical relays. An N-bit Counter-ramp ADC using a clock frequency of Fc takes $T_{\max} = \frac{2^N}{F_c}$ s for one conversion. For example, a 14-bit counter-ramp ADC using 1 MHz clock frequency takes 16.4 ms for converting a full-scale input to an equivalent numerical value [16]. Many numerical relays sample the voltages and currents at 1000 Hz or more and, therefore, the Counter Ramp converters are not suitable for use in numerical relays.

4.3.1.3.1 Successive-Approximation Converters

Successive-approximation ADCs are often used in numerical relays. These converters use a "binary search" algorithm in a feedback loop [16]. Figure 4.3 illustrates the successive-approximation converter architecture that consists of a front-end sample and hold (S&H), a comparator, a shift register, successive-approximation decision logic (SAL), a decision register and a digital-to-analog converter (DAC). During a binary search, the circuit halves the difference between the analog signal, V_H, and the DAC output, V_D, in each clock cycle. The conversion proceeds as follows. First, both the shift register and the decision register are set to mid-scale so that the

DAC produces the mid-scale analog output. The comparator is then strobed to determine the polarity of V_H-V_D. The shift register and the decision logic direct the logical output of the comparator to the most significant bit (MSB) in the decision register. Thus, if $V_H > V_D$, the MSB of the decision register is set to ONE, and if $V_H < V_D$, MSB is set to ZERO. This sequence is repeated until all the bits are determined. An N-bit successive-approximation ADC takes N-cycles to complete a conversion.

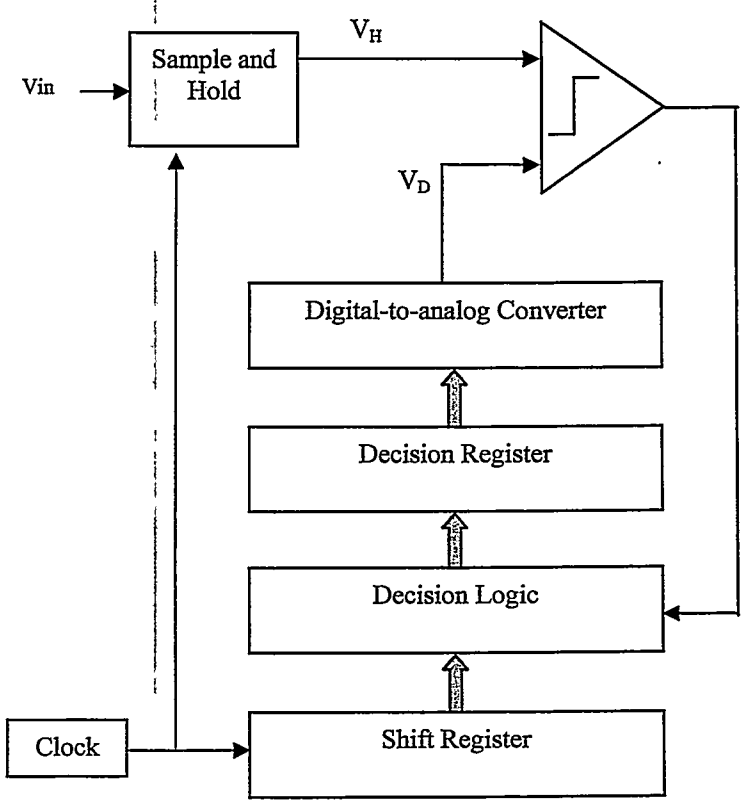


Figure 4.3 Successive approximation converter architecture

In a successive approximation ADC, the bits are determined bit by bit, from the MSB down to the LSB. This serial nature of successive-approximation converter limits the operating speed of commercially available ADCs to no more than a few MHz.

Consider a 14-bit successive-approximation converter with conversion clock of 1 MHz. This converter will take one clock cycle to determine if the input is positive or negative and 14 clock cycles are used to generate the 14-bits. The converter, therefore, will take 15 cycles of the clock that correspond to a time of 15 μs.

Since N-cycles are used to determine the result, a S&H circuit must be used to keep the input signal stable to the desired accuracy while each bit is evaluated. For the above case, the ADC Sample and Hold Period must be equal to the successive-approximation conversion time plus 20% overhead.

Total Conversion Time = (Sample and Hold Period) + (ADC Conversion Period)

$= 3\ \mu s + 15\ \mu s$

$= 18\ \mu s$

A new conversion can begin every 18 μs. Some numerical relays can require up to seven simultaneous input signals to be computed for performing the necessary protection scheme.

Conversion time for a 7 input system = 7 × (Conversion time for one channel)

$= 7 \times 18\ \mu s$

$= 126\ \mu s.$

Successive approximation converters need signals to start A/D conversion. On completion of the conversion, the A/D provide a signal indicating the completion of the conversion process. After the input analog signal is converted to a numerical value by the successive approximation converter, it is fed to a register from where it is given to the processors for computations. These factors also influence the total conversion time.

Thus, the total conversion time for seven input signals will be more than 126 μs. From Table 4.2; for an input signal sampled at 3,600 Hz, the time between consecutive samples is 277 μs. The successive-approximation converters for a seven channel numerical relay application, therefore, can be appropriate for a sampling frequency of 3,600 Hz. Flash ADCs can provide improved conversion speed over the successive-approximation converters and are described in the next section.

4.3.1.3.2 Flash Converters

Flash ADCs provide the fastest approach for converting analog signals to numerical values. An N-bit Flash converter employs 2^N-1 latched analog comparators in parallel. A resistive voltage divider provides reference voltages for the comparators, which are spaced one least significant bit, $V_{ref}/2^N$ volts apart. Analog input signal is applied to the input of the comparator bank. A simplified diagram of a 3-bit flash converter is shown in Figure 4.4. Bipolar reference voltages V_{ref+} and V_{ref-} are placed across a resistor string with evenly spaced taps creating 8 (2^3 bits) references. The input signal is compared with the reference voltages applied to the bank of comparators. The input signal in this manner is simultaneously compared with 2^N-1 reference voltages derived from the voltage divider string.

All comparators, to which the applied reference voltages are less than the voltage of the analog input signal, produce a logical 1 output. The remaining comparators, to which the applied reference voltage is equal to or more than the voltage of the input signal, produce a logical 0 output. The outputs of the comparators are then combined in a decoder that generates the binary output. In this manner, the conversion takes place in only two steps (voltage comparison and decoding).

Although Flash ADC allows very high speed processing (only 1 clock cycle per conversion) and low latency, the problems of large number of components of the resistance divider and stray capacitances make it almost impossible to design and build converters with large number of bits [19]. For example, an 8-bit Flash ADC requires 255 (2^8-1) comparators, while a 14-bit Flash ADC requires 16,383 (2^{14}-1) comparators.

With this exponential increase in comparators, the decoding logic also increases in complexity.

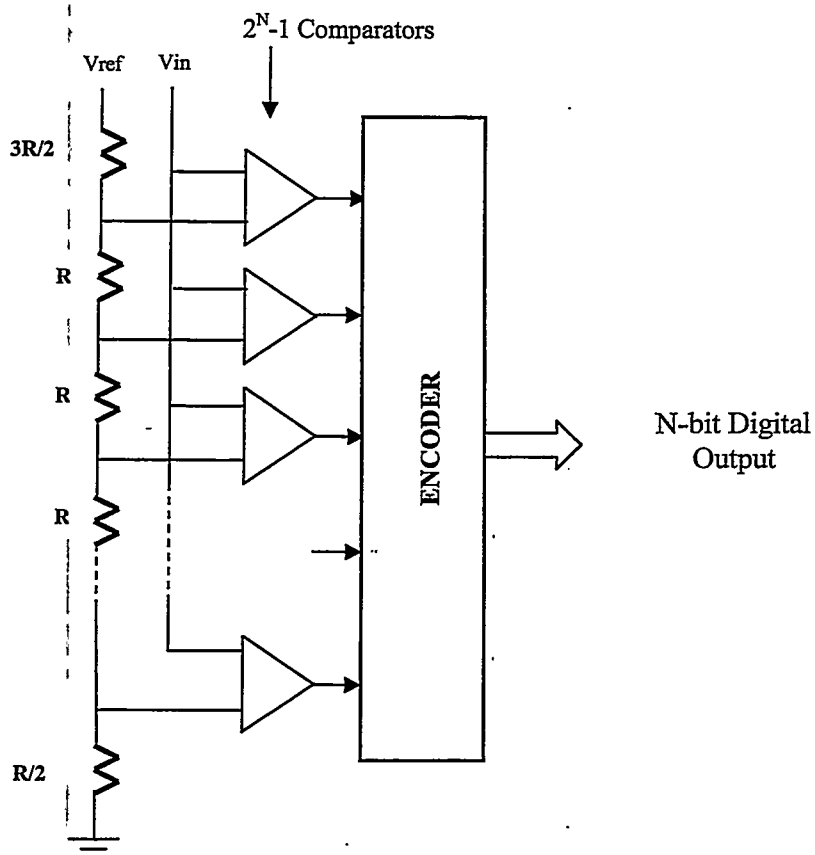


Figure 4.4 Flash Converter Architecture

Flash converters of upto 8-bit resolution are commercially available at this time. The number of bits in the ADC influences the dynamic range of the signal which the ADC converts. For example, an 8-bit ADC can quantize an analog signal into 1 of 256 (2^8) different digital levels.

To analyze the effectiveness of available Flash ADCs for numerical relays, consider a numerical distance relay. For a distance relay, the current rather than the voltage inputs will have a greater dynamic range. Consider that the RMS value of the

maximum fault current on a 230 kV line can be as high as 15 kA. The peak value corresponding to this fault current is 21.3 kA. The current needs to be reduced to a lower level by using a primary CT. A primary CT of ratio 500/5 A is suitable for reducing the level of this current. The fault current may contain a decaying dc offset. Taking the offset factor of 1.8 in to consideration, the maximum current is

$$I_{max} = (1.8) \times (21.3 \text{ kA}) = 38.34 \text{ kA}$$

Power system signals are bipolar; thus the dynamic range of the current is ± 38.34 kA.

Range of the current out of the secondary winding of the CT = (5/500) × (± 38.34 kA)
= ± 400 A

This current level is too high for numerical relays. Thus, the current is further reduced; by using auxiliary CTs of ratio, say 400/1 A. Consider an ADC with an input voltage range of ± 1 V (peak-to-peak voltage of 2 V).

Current on secondary of the auxiliary CT = (1/400) × (± 400 kA)
= ± 1 A

The ADCs work with voltage signals mainly; thus the current level of ± 1 A can be converted to equivalent voltage level by passing through a resistor of 1 ohm to obtain an input range of ± 1 V. The steps in the above mentioned conversion processes are shown in Figure 4.5.

The ± 1 V analog signal is provided to an ADC for converting the sampled signals to equivalent numerical values. The accuracy of the analog to digital conversion is dependent on the resolution of the ADC, which is the smallest input analog level for which a numerical output code is produced. The more steps the input signal of ± 1 V can be divided into, the better it can be represented in the numerical form.

Table 4.3 compares the quantizing differences for a 2 V 8-bit ADC and a 14-bit ADC. An ADC with 8-bits can provide outputs that are in 7.843 mV steps (7.843 mV, 15.386 mV, 23.529 mV and so on up till 2 V). Through its entire 2 V range, the measurements can never be more accurate than ± ½ LSB = 3.9215 mV. At this resolution, the power system signals cannot be represented in the numerical form with desired accuracy. More bits give better resolution and smaller steps. A 14-bit ADC provides a better numerical representation of the input signal. This ADC can digitize in 122.07 μV steps and is 0.96 times better in representing the digital signal compared with an 8-bit ADC. Thus, for representing power system signals effectively and accurately, an ADC with more bits should be selected. Commercially available Flash ADCs can fall short in providing the desired resolution.

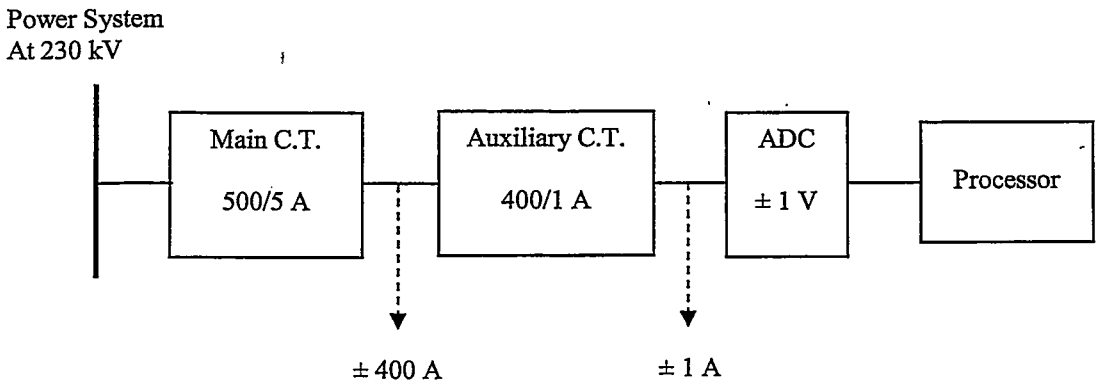


Figure 4.5 Steps in current reduction from power system to ADC

Table 4.3 Comparison of 8-bit and 14-bit ADC

Resolution	1 LSB	½ LSB	FSR
8 bits	7.843 mV	3.9215 mV	2 V
14 bits	122.07 μV	61.038 μV	2 V

To overcome the resolution drawbacks of pure Flash ADCs, pipeline ADC architectures can be used [20]. Pipeline ADCs provide an optimum balance of size, speed, resolution, power dissipation, and analog design effort. They have become increasingly attractive to major data-converter manufacturers and designers. A pipeline ADC tends to have much lower power consumption than a flash converter of comparable performance. Moreover, a pipeline ADC tends to be less susceptible to comparator metastability that can lead to sparkle-code errors; which is a condition in which the ADC provides unpredictable, erratic conversion results. Mismatched comparator delays can turn a logical 1 into 0 (or vice versa), causing the appearance of "bubbles" in an otherwise normal thermometer code. Because the ADC's decoder unit cannot detect these errors, it generates out-of-sequence codes that appear as output "sparks."

4.3.1.3.3 Pipeline Converters

A pipelined ADC employs a parallel structure in which each stage works on a few bits of successive samples concurrently. Pipeline ADCs consist of numerous consecutive stages, each containing S&H, a low-resolution ADC, DAC and a summing circuit that includes an interstage amplifier to boost the residue.

Pipeline ADCs break the conversion process from analog to digital into many stages, each stage consisting of a small number of bits per stage. A block diagram of the pipeline converter with p-stages is shown in Figure 4.6. If a stage provides m_p bits, the resolution of the converter is $N = m_1 + m_2 + \ldots + m_p$ bits.

Consider a 14-bit pipeline ADC. The analog input, V_{in}, is first sampled and held steady by a S&H. A flash ADC, provided in stage-1 quantizes the signal to provide five most significant bits. The four most significant bits of the output are placed in the most significant bit positions of a 5-bit DAC, which converts the quantized value to the analog form. The analog output of the DAC is subtracted from the input analog signal, V_{in}. The "residue" is then amplified by a factor 16 (by a power of $2^4 = 16$) and fed to the next stage (stage-2). The amplified residue is applied to the S&H amplifier at the

input of the second stage which uses a 4-bit flash converter to quantize the signal. The three most significant bits of the output are placed in the most significant bit positions of a 4-bit DAC which converts the quantized value to the analog form. The analog output of the DAC is subtracted from the signal held by the S&H amplifier. The residue is amplified by a factor of 8 and is applied to the S&H amplifier at the input of the third stage. The third stage is exactly the same as the second stage. Finally, the fourth stage converts the input to the last 4 least significant bits. Because the bits from each stage are determined at different points in time, all the bits corresponding to the same sample are time-aligned with shift registers and are then subjected digital-error-correction logic to obtain the final 14-bit digital output [20, 21].

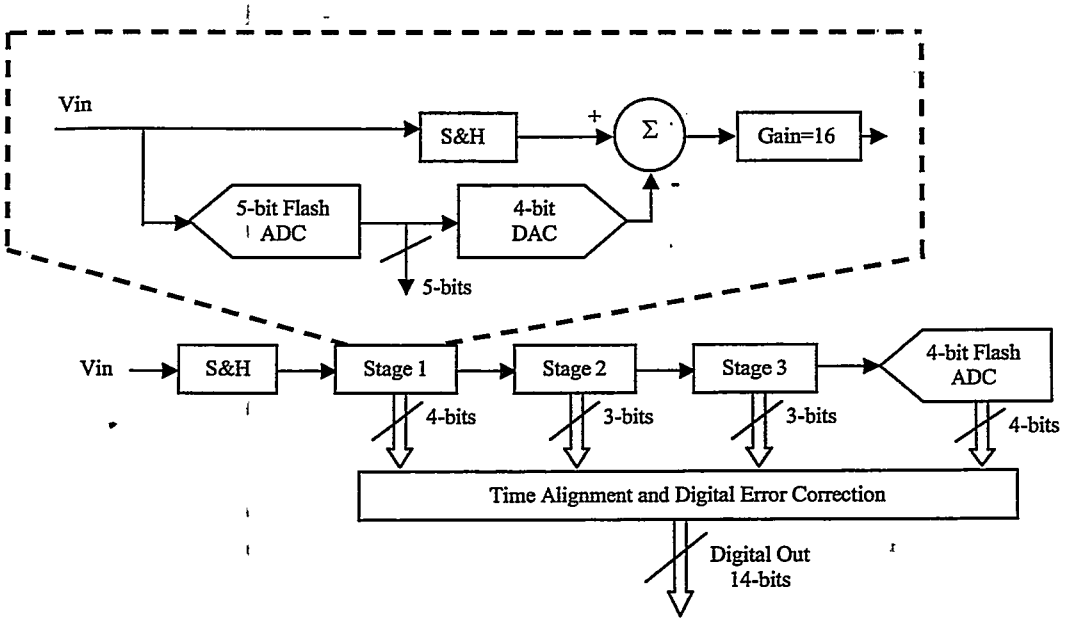


Figure 4.6 Architecture of Pipeline ADC

Important consideration for this project was to investigate the working of numerical relay at higher sampling rates; therefore, the pipeline ADC was selected. The expected digital output response of the selected pipeline ADC is described in Appendix C. Table 4.4 tabulates the various ADC techniques discussed in this section.

The quantized data from ADCs is processed by digital signal processors. Processors are selected keeping in mind the computational burden estimated for the highest sampling frequency of the input signal.

Table 4.4 Comparisons of ADC Techniques

Architecture	Resolution	Speed	Advantages	Drawbacks
Flash	8 bits	250MHz-1GHz	Extremely fast High input bandwidth	Large die size. Highest power consumption. High input capacitance. Expensive.
Successive-Approximation	10-16 bits	50kHz-500kHz	High resolution High accuracy Low power consumption	Low input bandwidth. Limited sampling rate. Vin must remain constant during conversion.
Integrating	>18 bits	<50 kHz	High resolution Good noise rejection	Low speed.
Pipeline	12-16 bits	1MHz-100MHz	High throughput rate Low power consumption	More sensitive to board layout then other architectures. Critical timing needed for synchronization of all outputs.

4.3.2 Digital Processors

Relaying algorithms perform computations that include additions, subtractions, multiplications, divisions and logic functions. The computations are repeated every time a new set of samples is received and these computations must be completed before the next set of samples is received. This has become possible with advancements in technology and advent of fast digital signal processors (DSPs). These processors are similar to the general purpose microprocessors except that they are better optimized to perform operations that contain multiplications and additions.

Consider a relaying application that samples voltages and currents at 3,600 Hz. This sampling rate requires 246 instructions to be performed by the processor as outlined in Table 4.2. If a processor has a cycle time of 60 ns, then it would require

14.88 μs (60ns×246) to complete one set of calculations. As this time is less than the inter-sampling time of 277 microseconds (1/3,600Hz), the application is in a real-time. This process is illustrated in Figure 4.7.

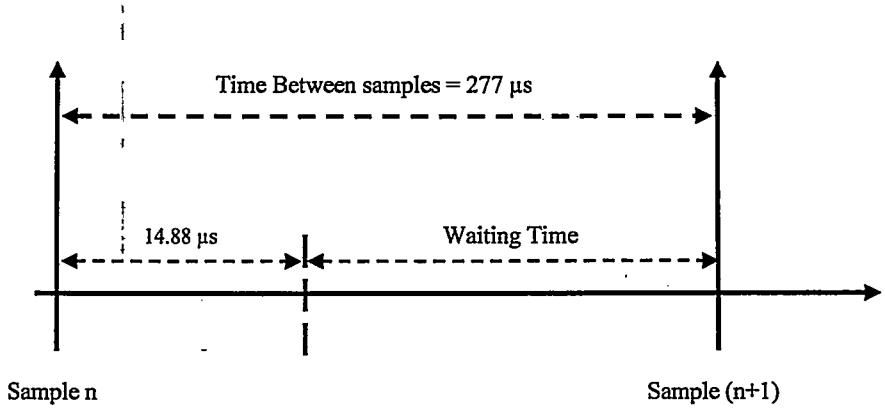


Figure 4.7 Definition of real-time

There are various parameters to be considered when selecting a suitable DSP for implementing numerical relays; some of these are as follows.

1) Speed of the processor
2) Accuracy of the computed results
3) Size of on-board program memory
4) Size of data memory
5) Efficient use of cache memory
6) Adequate instruction set
7) Fixed point versus floating point capabilities
8) Software development tools
9) Capabilities of interfacing with other processors and systems

While the speed of the processor determines the type and number of computations it can perform per second, the accuracy of the results computed by the processor depends on the word-size used in numerical computations.

For fast execution of the algorithms, the program memory should be sufficient for storing the program code in it. In addition to the program memory, sufficient data memory should be available. Use of the internal program and data memory reduces the overheads required to access/load instructions and data from the external memory.

Cache memory is a small amount of very fast memory used as a buffer between the processor and the main memory. Effective use of cache memory enables processors to achieve high performance. When running the algorithms; elementary considerations in the effective use of cache can reduce the execution time by a factor of two. A schematic illustration of the place of cache memory is shown in Figure 4.8.

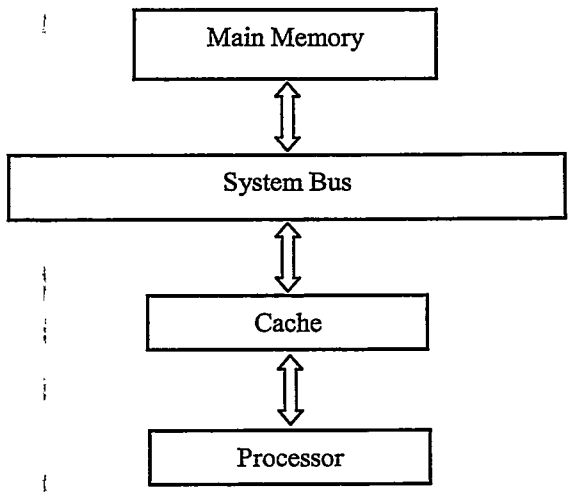


Figure 4.8 Illustration of logical position of cache memory

Most processors provide the ability to use at least some of the internal memory as a program cache; this accelerates the access to the program code that is stored off-chip, at the expense of reducing the amount of program code that can be stored on-chip.

This facility provides a benefit for many applications that are too large to fit entirely into the on-chip program memory.

Digital signal processors provide adequate instruction set to implement the algorithms. If some instructions are not available, it should be possible to implement them using the instructions from the available instruction set.

Digital processors can be divided into two categories, fixed point and floating-point processors. These refer to the format used to store and manipulate numbers in the processors. Fixed-point processors usually represent each number within the device with a minimum of 16-bits, although a different length can be used. Floating-point processors typically use a minimum of 32-bits to store each number. All floating-point processors can also handle fixed-point numbers and arithmetic; this is necessary for implementing counters and loops. Floating point representation of numbers provides better precision and a higher dynamic range than fixed-point processors. In addition, floating point programs often have a shorter development cycle, since there is no need to worry about issues such as overflow, underflow and round-off errors.

An increasingly large part of a numeric relay development is spent on writing the software. Digital signal processors are programmed in usually assembly or high-level language (such as C, Fortran or Basic). Programs written in assembly execute faster, while programs written in a high-level languages are easier to develop and maintain. For real-time applications, it may be necessary to write the programs in the assembly language. For this purpose, software development tool aids are considered. The tools provided with most processors are shown in Figure 4.9 and are as follows [21]:

- C compiler,
- Assembly optimizer,
- Simulator,
- Code converter,

- Assembler, and
- Linker.

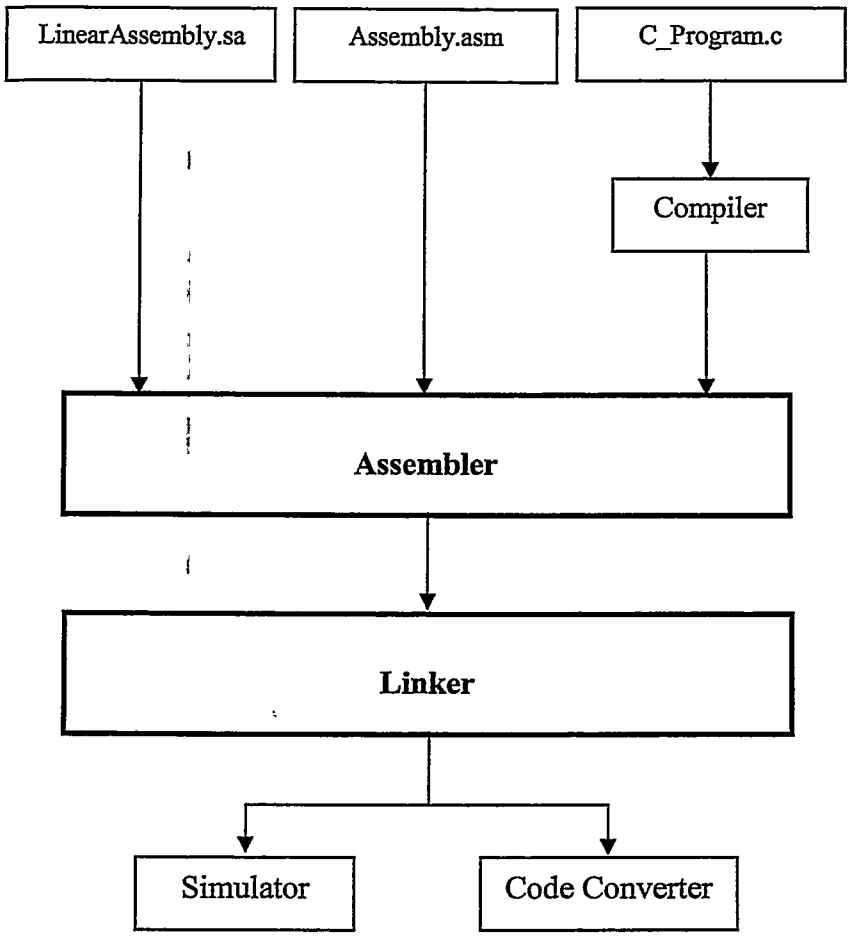


Figure 4.9 Development Tools

If the source code is written in C language, the code should be compiled using the Optimizing C Compiler provided by T.I.. This compiler will translate the C source code into an assembly code. The assembly code generated by either the programmer, the compiler or the linear assembly code is then passed through the assembler that

translates the code into object code. The resultant object files and any library code are combined by the linker, which produces a single executable file.

Various parameters have to be considered when selecting a suitable processor. Many commercially available digital processors from Texas Instruments (T.I.) were considered and compared for this project. Table 4.5 summarizes some of the important parameters of the processors.

Consider the TMS320C30 processor with instruction time of 60 ns. At a sampling frequency of 3,600 Hz, the time between consecutive samples is 277 μs. From Table 4.2, the total number of computations required by the DFT at 3,600 Hz is 248.

Table 4.5 Digital Signal Processor Specifications

Part Number	TMS320C30	TMS320C40	TMS320C50	TMS320C60
Instruction Cycle Time	60 ns	33 ns	20ns	5 ns
Word Length	32-bit	32-bit	16-bit	32-bit
Data/Program Memory	16 Mbytes	4 Gbytes		64K/64Kbytes
Fixed/Floating	Floating	Floating	Fixed	Both
Operating Frequency	33 MHz	50 MHz	144 MHz	200 MHz

The time needed to compute one phasor magnitude value = 60 ns × 248

= 14.88 μs

Overhead associated with software = 2 μs

Total time needed for one sample = 14.88 μs + 2 μs = 16.88 μs

Some numerical protection applications can require seven input signals to be computed for performing the necessary protection scheme. Thus, the computational time required for seven input signals is as follows.

Time needed for seven samples = 7 × (Total time needed for one sample)

= 119 μs

Total time = (Settling time of the board) + (Time needed to compute seven samples) + (Overhead of 150 %)

= 10μs + 119 μs + 178.5 μs

= 307.5 μs

Thus;

(Time between samples, 277 μs) < (Time needed to compute 7-samples, 307.5 μs)

TMS320C30 processor is, therefore, not suitable for numerical relays that would sample data at 3,600 Hz for seven simultaneous channels.

TMS320C40 processor with an instruction time 33 ns can be suitable for sampling seven channels simultaneously. With this processor,

(Time between samples, 277 μs) > (Time needed to compute 7-samples, 190 μs)

TMS320C50 processor being a fixed-point processor with word length of 16-bits is not suitable for relaying applications that calculates phasor magnitudes and phase angles. TMS320C6x series would, however, be suitable for numerical relaying applications.

Communicating with other devices is a prime requirement for protection systems. Diamond RTOS [23] provides easy programming of the multi-processor systems and is a multi-DSP extension to the Texas Instruments complier, assembler and linker.

4.4 Selected Hardware

The provision of sampling more than one channel simultaneously is an important factor central in power system protection. Keeping this factor in mind, data acquisition boards with more than one-channel and based on Flash ADC or its variants were considered. List of different boards considered for this application is given in Appendix B. The complete hardware and software package, which meets the requirements, is provided by devices manufactured by Sundance.

The Sundance SMT356 analog-to-digital board was selected. This consists of eight Analog Devices AD9240 converters. All ADCs simultaneously sample using the same clock. Considering the computational burden associated with the DFT algorithm, Texas Instruments (T.I.) TMS320C6000 series of processors were considered for this project. While the TMS320C40 series of processors would be suitable for seven input channels, they would not provide ample margin for future needs of research in power system protection.

The following processors were selected for this project.

1) TMS320C6201, fixed-point processor, and
2) TMS320C6701, floating-point processor.

The architecture of the above processors and other important parameters discussed in the previous section are described in Appendix D. The ADC board interfaces with the digital processor board SMT375 with TMSC320C6701 (floating-point processor) and also with the digital processor board SMT 335 with TMSC320C6201 (fixed-point processor). Different modules; like the ADC and, the processors C6201 and C6701 (SMT375 and SMT335) are placed on a carrier board (SMT310Q). The carrier board has the capability of holding four modules. Thus, there is a provision for expanding the system in the future. The carrier board consists of one eight-channel ADC module and two digital processor boards.

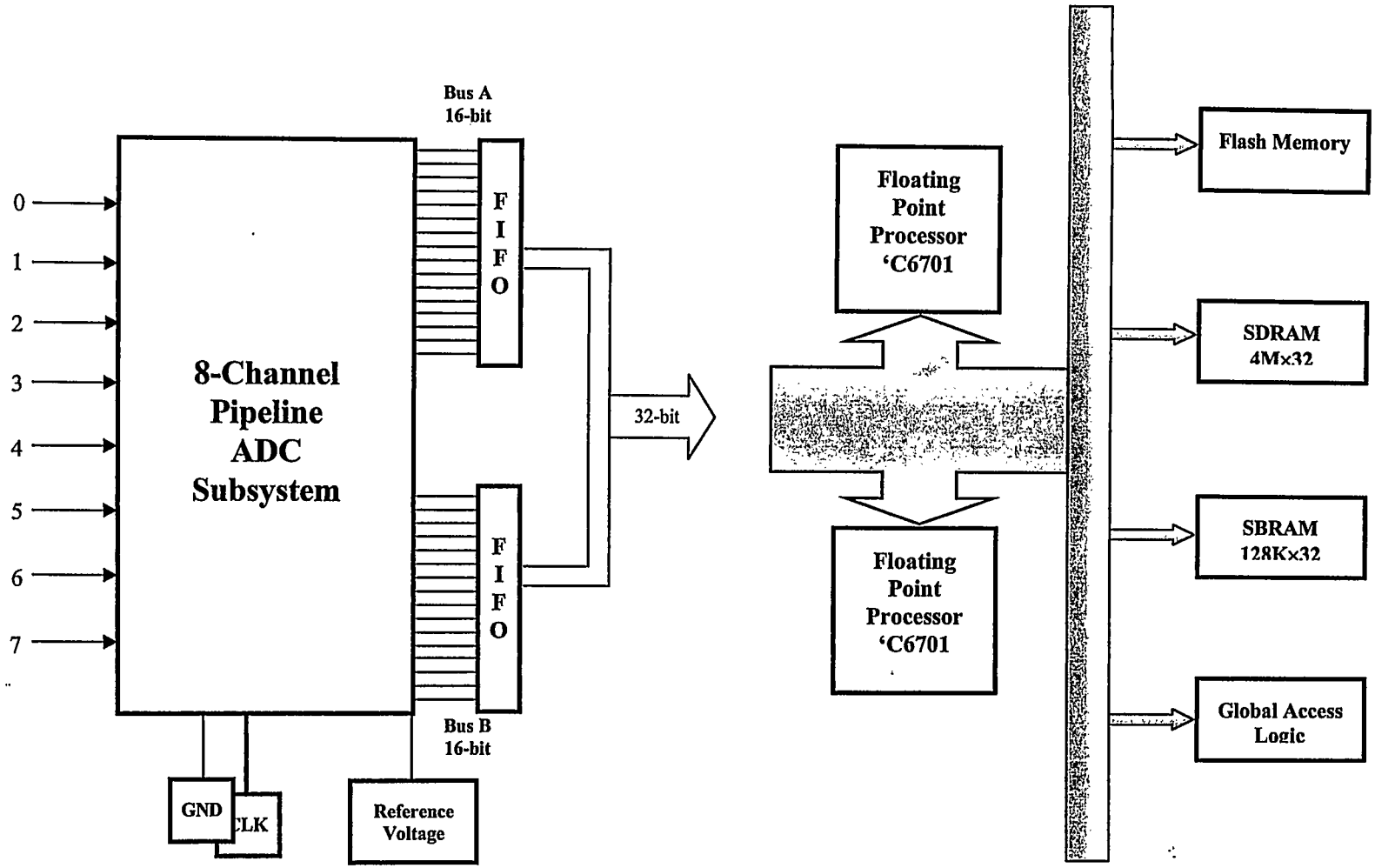


Figure 4.10 Detailed description of hardware

Various modules on the carrier board can exchange data using communication ports and internal buses, Sundance Digital Buses, SDBs which are 16-bits wide. All channels of the ADC module sampled at the same time provide outputs that are placed in the higher 14-bits on the SDB bus; the lower two bits are clamped to ground. The sampled data is buffered within a 256-word First Input First Output (FIFO) table. This data is presented to digital signal processors for processing. Figure 4.10 shows the basic description and interconnection of the various modules.

4.5 Summary

The procedures followed in the selection of the relay algorithm and in the selection of the components of the analog input subystem are explained in this Chapter. The use and importace of anti-aliasing low-pass filters in numerical relays was described. Different analog-to-digital converter techniques used in numerical relays are discussed and compared. It is concluded that the pipelined ADC is the architecture of choice for sampling rates from a few MHz up to 10 MHz. Complexity of these converters increases linearly with the number of bits, providing converters with high speed, high resolution, and low power consumption.

The important parameters to be considered in the selection of digital processors for numerical relays were outlined and described. Finally, a brief description of the selected hardware for this project was presented.

CHAPTER 5

NUMERICAL RELAY DEVELOPMENT

5.1 Introduction

Numerical relay development includes the selection of suitable components for the analog input subsystem and the digital signal processors. The major functional blocks in the analog input subsystem of a numerical relay and the digital signal processors considered for this project are presented in Chapter 4. The design of a numerical relay with the selected hardware components and the Diamond Real Time Operating Software is described in this Chapter.

5.2 Diamond Real Time Operating Software

The Diamond development platform is shown in Figure 5.1 [21]. The Diamond software is hosted by a PC, called the HOST. The HOST also controls the C6000 processor network. Diamond Real Time Operating Software (RTOS) provides easy programming of multiple connected processors.

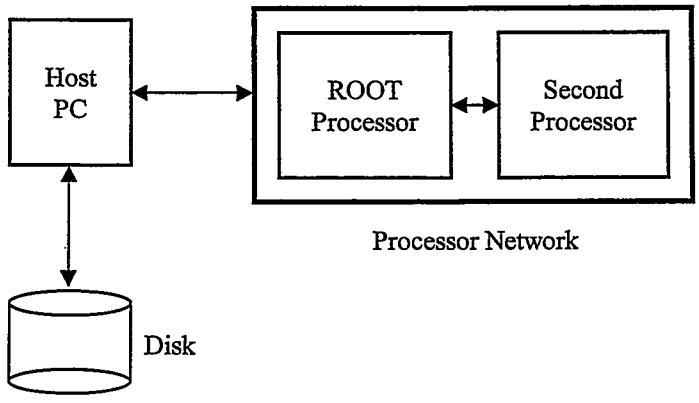


Figure 5.1 Diamond development platform

The source program files are compiled on the HOST computer using a standard C compiler or an assembler to produce the object files. The object files are linked with the run-time libraries using the linker to produce individual task image files. The task image files are then combined together to generate a single C6000 executable application file. The HOST computer runs a Diamond file server program, which loads the C6000 executable application file from the HOST onto the target processor board and executes it. The HOST communicates directly with only one processor on the processor network, called the ROOT processor.

5.2.1 Network Loading and Configuration

A block diagram of the complete application is described first. For each block, complete C and/or assembly programs are written separately. These programs are referred to as TASKS. A complete application is a collection of one or more tasks that are concurrently executed. Each task has its own main function and a region of memory for code and data. The TASKS receive input data, process the data and provide the required output. Each task has a vector of input ports to receive the input data and a vector of output ports to provide the outputs. Using input and output ports, different tasks can be connected together. Figure 5.2 shows a task and its related input and output ports.

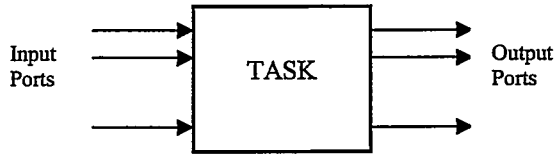


Figure 5.2 A TASK with input and output ports

All the tasks are independent and separate from the other tasks in an application and are compiled and/or assembled individually producing object files (.OBJ). The individual object files are then linked with the run-time library (.LIB) producing task

image files (.TSK) for each task. Some of the features offered by the Diamond's run-time library are as follows.

- Provides the ANSI standard functions
- Facilitates communication between tasks by sending and receiving messages
- Helps in the memory allocation for different tasks
- Provides access to the host operating system

Figure 5.3 shows the necessary steps for creating a task image file.

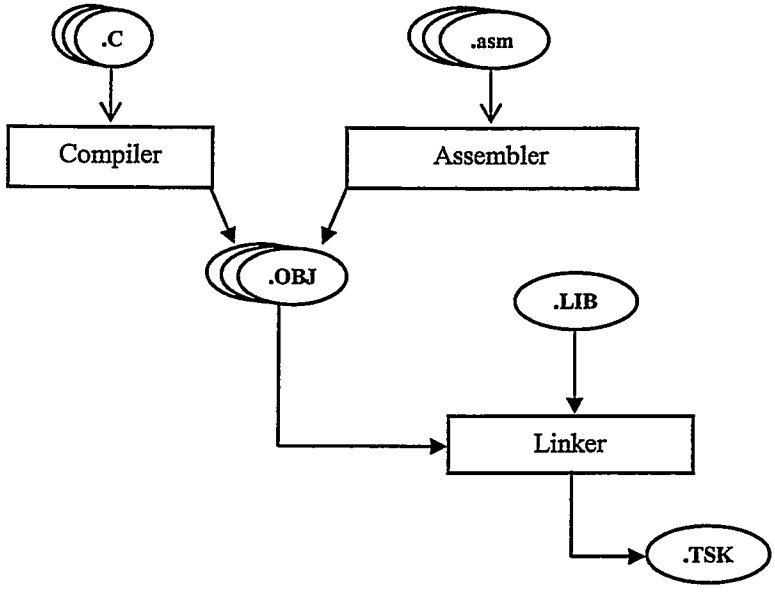


Figure 5.3 Steps in creation of a TASK image file

The complete application is built by joining the individual tasks together. This is accomplished by connecting the input and output ports of tasks with the CHANNELS. A channel sends data from one task to another task, thus allowing different tasks to communicate with each other. The vector of input and output ports are passed to a task as arguments of its main function.

The individual task image files are combined to form a single executable APPLICATION file (.APP) by using a tool known as the CONFIGURER. This tool is driven by a file called the CONFIGURATION file, which is a user written text file, with an extension of CFG. This file describes the system to be built by giving to the CONFIGURER details of the hardware and software on which the application has to be run. The configuration file specifies the following information.

1) Hardware structure
   - Available processors
   - Links connecting them

2) Software structure
   - Tasks to be included
   - Channel connections between tasks.

3) The location of the tasks and when they are to be executed

The CONFIGURER takes the description of an application from the configuration file, builds a single application file that contains everything needed to load on the processor network. The Diamond server program on the HOST runs the executable application file. The loading process proceeds as follows.

- The target processor system is first reset
- The code contained in the application file is sent to the processors
- Once the application has been loaded, the server starts all the tasks on the assigned processors

5.2.2 Analysis Procedure

The complete application process is described as a block diagram shown in Figure 5.4. Each block represents a task that can have many input and output channels.

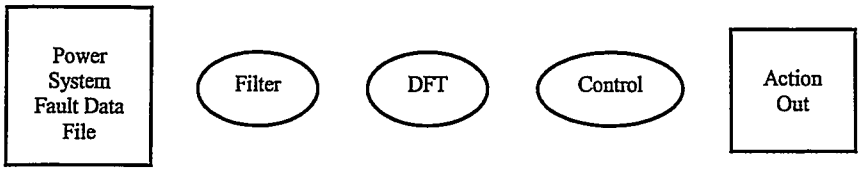


Figure 5.4 Single Phase Application

Tasks shown in Figure 5.4 can be loaded onto a processor named ROOT and run. The configuration file should contain the name of the ROOT processor, such as

```
! Hardware Description

Processor ROOT Type = SMT 375
```

The first line is a command statement that starts with an apostrophe and the next line is a statement that declares a physical processor. The processor statement includes the processor name, ROOT, and the Type, SMT 375. The CONFIGURER regards every processor as having a TYPE. For example, if a processor is a C6000 processor, the statement should identify the TYPE of C6000 as well. In this project, the processor is identified by its name that is SMT 375, the floating type processor. The various TASKS are described as follows.

```
! TASK Declaration

TASK Data File INS = 1 OUTS = 1 DATA = 2K
TASK Filter    INS = 1 OUTS = 1 DATA = 1K
TASK DFT       INS = 1 OUTS = 1 STACK= 2K HEAP=2K
TASK Control   INS = 1 OUTS = 1 DATA = 5K  URGENT
TASK Action    INS = 1 OUTS = 1 DATA = 3K
```

A TASK statement declares a task and may contain a number of attribute clauses, each of which describes some aspect of the task. The INS attribute is used to specify the number of elements in the task's vector of inputs. The OUTS attribute is used to specify the number of elements in the task's vector of outputs. The CONFIGURER must be told the dynamic data storage requirements for each task in an application. This is accomplished using the STACK, HEAP and DATA attributes of the TASK statement. The DATA clause tells the CONFIGURER to allocate a single contiguous area of memory for the applications stack and heap areas. This area is the maximum memory area left when the memory required for the code and static data has been allocated. The code and static areas are not specified, as the CONFIGURER finds the size of these areas by inspecting the task image file.

With the statement DATA = 3K; the stack and heap logical areas are allocated a single contiguous area of 3K of memory. The statement STACK = 2K HEAP = 2K; allocates the logical areas separately and are given the sizes indicated.

The URGENT attribute of the TASK statement specifies that the task is to be started at the urgent priority level.

The individual TASKS are connected by the following statements; the connected tasks are shown in Figure 5.5.

```
! Set up connection between TASKS

CONNECT ?   Data File [0]   Filter [0]
CONNECT ?   Filter [0]      DFT [0]
CONNECT ?   DFT [0]         Control [0]
CONNECT ?   Control [0]     Action [0]
```

The CONNECT statement connects an output port of one TASK with an input port of another TASK. The CONNECT statement describes a logical connection

between two tasks, running in one direction from the output port (left-hand) to the input port (right-hand). Communication in both the directions between a pair of TASKS, therefore, requires two CONNECT statements.

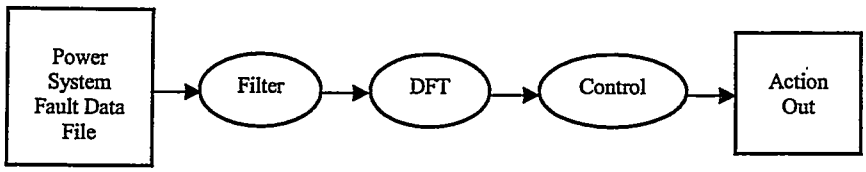


Figure 5.5 TASKS connected together

Every TASK must be placed on a processor. The assignment of TASKS to a particular processor is done by the following commands.

```
! Assign TASKS to the available processors

PLACE   Data File   ROOT
PLACE   Filter      ROOT
PLACE   DFT         ROOT
PLACE   Control     ROOT
PLACE   Action      ROOT
```

The PLACE statement determines on which processor a particular TASK is to be executed. In the above case, all the tasks are placed on the ROOT processor. The fully connected application, which is placed on the ROOT processor, is shown in Figure 5.6.

5.2.3. Generation of Executable Application File

After writing the configuration file, the application is built by compiling the programs, linking them and running the CONFIGURER to generate a single executable application file. The following steps are performed for this purpose.

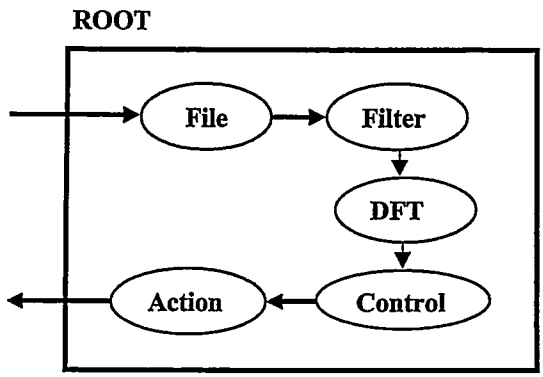


Figure 5.6 Application placed on ROOT processor

- To compile each TASK separately written in C, the following command is used:

  C6xc *FileName.c*

- To compile each TASK separately written in assembly language, the following command is used:

  C6xc *FileName.asm*

- To link all the separate TASKS together into one file, the command used is as follows:

  C6xlink *FileName*

- To CONFIGUER the above file to generate an executable application file (.APP), the following command is used:

  Config *FileName.cfg FileName.app*

5.2.4. Running the Application

To select and run the executable application file the following steps are performed.

1) The executable application can be opened in one of the following two ways.
   a) Use the command WS3L *FileName.app* from the DOS prompt. This opens the Diamond Server Window with the required application specified by *FileName* selected.
   b) From the START pop up menu, open Diamond Server Window. In this window, from the "File" drop down menu select the Application that is to be run.

   After an application has been selected, its name is displayed in the Diamond Server Window.

2) To run the selected application
   a) Select drop down menu "GO"
   b) Select "RUN" to run the application
   c) Results are displayed in the Diamond Server Window

5.3 Design of a Numerical Relay

A numerical relay was designed for processing the samples of voltage and currents normally acquired from a power system. In this project, the voltages or currents were taken from a power system simulation on the Electro Magnetic Transient Program (EMTDC) software. The calculated values were processed by low-pass anti-aliasing filters. The data was then down-sampled. The data was then processed by a 14-bit converter model. The converted data was provided to the DFT algorithm that calculates phasors. The calculated phasors were then used to determine the impedances seen by the relay.

The down-sampled data was also used without having been processed by the anti-aliasing filters. The voltage and current phasors were calculated and then used for calculating the

impedances seen by the relay. The results obtained for both approaches were then compared. Figure 5.7 shows the tasks that are performed by the relay for this purpose.

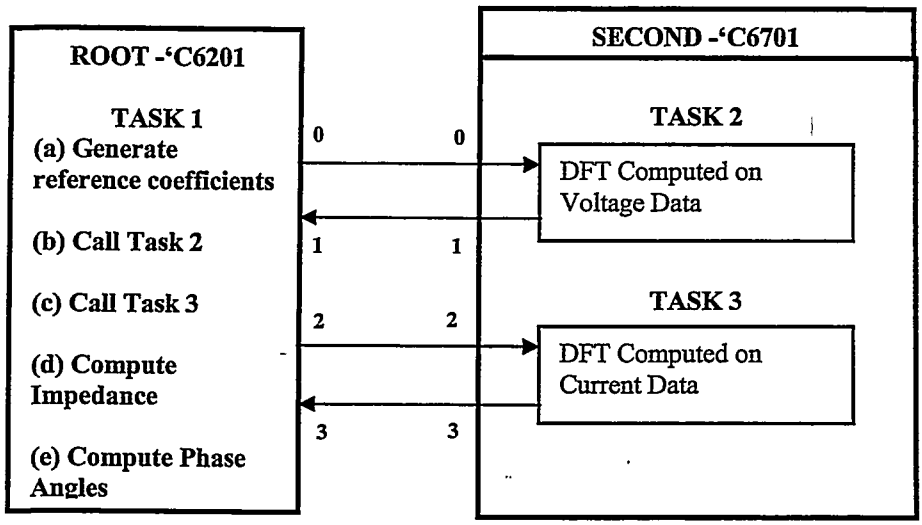


Figure 5.7 Various tasks for calculation of Impedance

5.3.1 Impedance Computation

This figure shows the organization of the problem into a set of parallel tasks that are performed by the two processors, ROOT and SECOND. There are three software tasks. The user interface TASK 1; runs on the ROOT processor and communicates with the user via the host computer keyboard and the screen using standard C printf and scanf library calls. The desired sampling rate is entered and corresponding reference sine and cosine coefficients are generated. These are sent over physical channels to the SECOND processor for use by TASK 2 and TASK 3, which compute the phasors and return them to TASK 1. The TASK 1 computes the impedances and phase angles and displays them in the Diamond server window. The configuration file, Application.cfg, for this setup is as follows.

```
! Application.cfg

! Hardware Configuration
Processor ROOT
Processor Second
Wire ? ROOT[0] Second[1]

! Software Configuration
Task input INS=2 OUTS=2 DATA=?
Task voltage INS=1 OUTS =1 DATA =20k urgent
Task current INS=1 OUTS =1 DATA =20k urgent

Place input ROOT
Place voltage Second
Place current Second

Connect ? input[0] voltage[0]
Connect ? voltage[1] input[1]
Connect ? input[2] current[2]
Connect ? current[3] input[3]
```

This file, Application.cfg, describes the target system that consists of two C6000 processors called the ROOT and the SECOND, connected by a single hardware link. Tasks that are not important for the relaying purposes, such as, the user interface are placed on the ROOT processor, whereas the important tasks that require immediate attention are placed on the SECOND processor. These tasks are executed with high priority. The configurer arranges for loading the code of the voltage and current tasks onto the Second processor when the application is started by the server. Communications between the processors is carried by a LINK, which is a physical connection between the two processors.

The complete list of commands, to build and run the application described above is as follows.

- C6xc *input.c*-Compiles User Interface Task 1
- C6xc *voltage.c Assembly_voltage.asm*-Compiles Task 2

- C6xc *current.c Assembly_current.asm*-Compiles Task 3
- C6xlink *input*-Links Task 1 with Library files
- C6xlink *voltage Assembly_voltage*-Links Task 2 with Library files
- C6xlink *current Assembly_current*-Links Task 1 with Library files
- Config *Application.cfg Application.app*-Generates Application file
- WS3L *Application.app*- Runs the Application file

5.3.2 Design of a Three-Phase Numerical Relay

Figure 5.8 shows the block diagram of a numerical relay that analyzes voltages or currents from a three-phase power system. The relay system is essentially three single phase relays in one software package. The voltage and current from each is processed in an identical manner that is described in the previous section. The processing times needed at different sampling frequencies are obtained and compared. There are eight software tasks in this package.

The user interface task (UI) runs on the root processor and communicates with the user via the host computer's keyboard and screen using standard C printf and scanf library calls. UI provides a command interpreter which allows the user to specify sampling frequency and the fundamental frequency.

Based on the specifications provided through the UI, the reference sine and cosine coefficients are computed for use by the tasks for computing the DFTs.

To compute a DFT on the input data, six identical tasks, placed on the SECOND processor in the network, are performed. They include the reading of input power system voltages and currents and performing the DFT computations and estimating the phasor magnitudes using programs written in assembly language. These phasor magnitudes are sent back to "Switch" task , where they are displayed. From these values impedances are calculated.

The switch task is interposed between the UI and the six DFT tasks. It merges the computed result packets from the six tasks into a single stream of output messages which are displayed on Diamond server window.

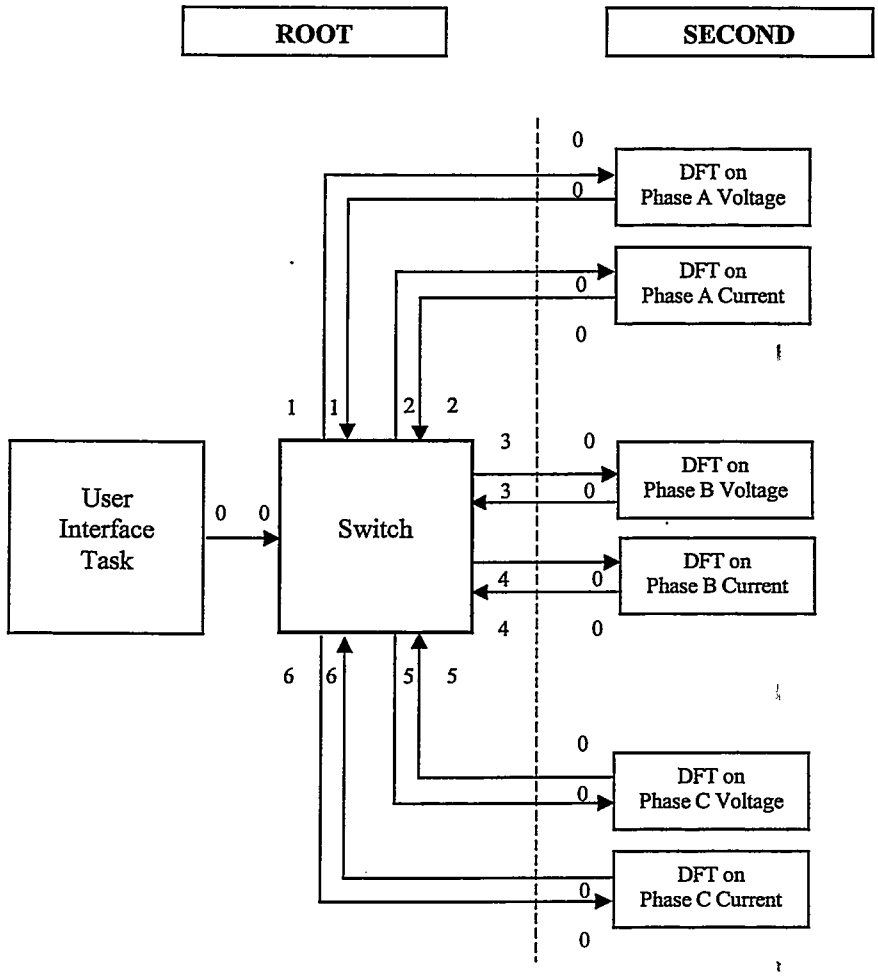


Figure 5.8 Application for three phases of a power system

Each processor in the network can support a number of tasks subject to the availability of memory. In Figure 5.8, there are eight tasks; two of them, User Interface task and Switch are placed on one processor named ROOT, while the remaining six tasks are placed on a second processor, named SECOND. The six tasks used to find the DFT on the input data execute in sequence; first the voltage data of phase A is passed on to the DFT algorithm of task one,

phasors are computed and results sent back to the calling task for displaying on the server and for saving in a file. Then the data of phase A current is passed on to the DFT algorithm of task two, phasors are computed and results are sent back to the calling program for displaying on the server and saving in a file.

```
! App_ThreePhase.cfg

Processor ROOT
Processor Second
Wire ? ROOT[0] Second[1]

Task UI INS = 1 OUTS = 1 DATA = ?
Task switch INS =7 OUTS = 7 DATA = 20k urgent
Task W1 INS =1 OUTS = 1 DATA = 20k
Task W2 INS =1 OUTS = 1 DATA = 20k
Task W3 INS =1 OUTS = 1 DATA = 20k
Task W4 INS =1 OUTS = 1 DATA = 20k
Task W5 INS =1 OUTS = 1 DATA = 20k
Task W6 INS =1 OUTS = 1 DATA = 20k

Place UI ROOT
Place switch ROOT
Place W1 Second
Place W2 Second
Place W3 Second
Place W4 Second
Place W5 Second
Place W6 Second

Connect ? UI[0] switch[0]
Connect ? switch[1] W1[0]
Connect ? W1[0] switch[1]
Connect ? switch[2] W2[0]
Connect ? W2[0] switch[2]
Connect ? switch[3] W3[0]
Connect ? W3[0] switch[3]
Connect ? switch[4] W4[0]
Connect ? W4[0] switch[4]
Connect ? switch[5] W5[0]
Connect ? W5[0] switch[5]
Connect ? switch[6] W6[0]
Connect ? W6[0] switch[6]
```

It is important to identify which channel has an incoming message waiting to be read. When such a channel is identified, a command is issued to read from that channel alone. After a task connected with that channel completes the computations, the next channel is selected and the procedure is repeated. The channel that becomes available first is identified to the calling

program, which can then goes onto read its message using one of the same channel I/O functions used to send messages.

The <alt.h> header function provided in Diamond run-time libraries allows a program to identify the input channel that is ready. The functions contained in this header file allow a program to wait until any one of a selected group of channels becomes ready to communicate.

Figure 5.9 shows the different tasks, related to finding DFT of various phases, on a time axis. UI task and switch task are not shown in this figure to simplify it. First control is passed to task 1, which computes the DFT on Phase A voltage. At this time other tasks are waiting for task 1 to finish. Task 1 computes the voltage phasors and sends the result back and finishes. Time taken by this task will depend on the sampling frequency at which the input signal is sampled. Sampling frequency determines the number of data points for which DFT is computed. For higher sampling rates, tasks will take longer time to execute. After completion of one task, some overhead (W1) is provided. This overhead is to make sure that the system stablizes before the control is passed on to task 2 in the sequence. Then the sequence is repeated for task 2 and so on, till all the tasks are finished.

5.4 Summary

The technique for implementing the DFT algorithm using the selected digital processors and Diamond Real Time operating software has been described in this chapter. The implementations of related tasks have also been described. The Diaomond software makes it easy to change the manner in which tasks communicate with each other and the manner in which they are placed on the processors. These aspects are controlled by the program placed in the Configuration file that is written for the CONFIGRER.

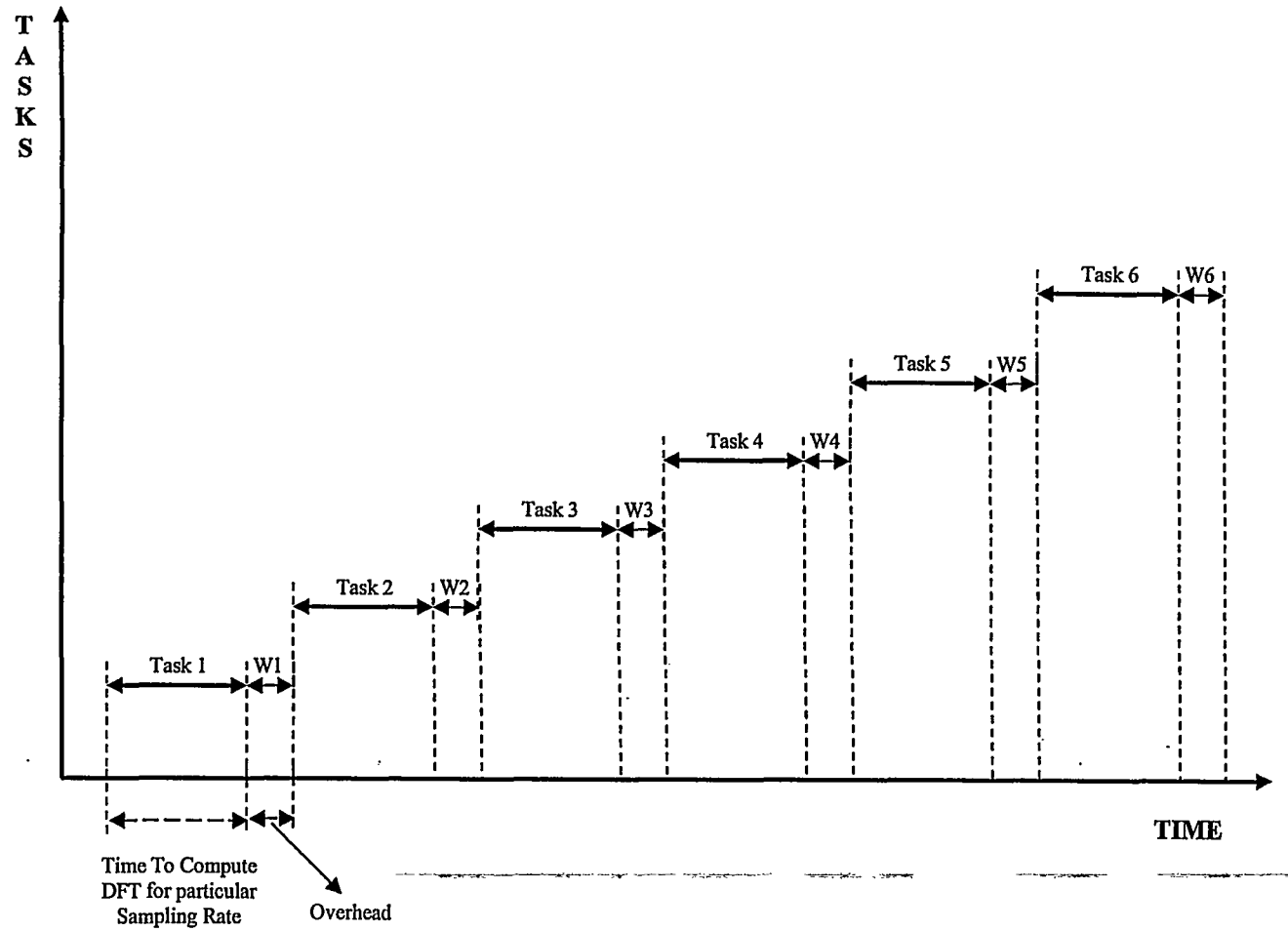


Figure 5.9 Assignment of Different Tasks

CHAPTER 6

SIMULATION AND SYSTEM STUDIES

6.1 Introduction

In Chapter 4, the important components of the analog input subsystem, namely the low-pass filters and the analog-to digital converters are described. The need for filtering the input signals, basic requirements and the important features of the low-pass filters are discussed. The analysis and the comparison of different ADC architectures used in numerical relays was presented and the selected pipeline ADC are described. The selection criterion for the processors for numerical relays is formed and the selected processors used in this project are described. Finally, the hardware selected for the project is presented. In Chapter 5, the Diamond RTOS is explained. The implementation of the DFT algorithm on the available hardware and the development of a numerical relay using the selected software are presented.

In most numerical relaying applications, the peak values of voltages and currents are calculated. The calculations of the peak values using the DFT algorithm were conducted on the selected T.I. 'C6000 series of processors. For real- time relay applications, the usual practice is to leave the phasor magnitudes as squares. In this project, the phasor magnitudes were also calculated when the DFT algorithm was implemented on the processors. The DFT code was written in T.I. assembly language and C programming language. The speed of execution of all the instructions of the DFT algorithm at different sampling frequencies was computed and compared for both the assembly and C written programs. The power system voltage and current were taken from a power system simulation on the EMTDC software done at a sampling frequency of 21,600 Hz. With this frequency, the data can be down sampled to frequencies of 720 Hz, 1,440 Hz, 1,800 Hz and 2,160 Hz. For example, to perform the DFT computations at a sampling frequency of 1,440 Hz, every fifteenth sample is taken from the power system

sampled at a sampling frequency of 21,600 Hz. The down-sampled data is provided to the DFT algorithm that calculated phasors.

The low-pass anti-aliasing filters are an important part of the numerical relay. The performance of a sixth-order Butterworth low-pass filter was evaluated and the group delay associated with this filter was calculated at different sampling frequencies. To test the validity of the functioning of the low-pass filters, they were simulated in EMTDC and Matlab. The numerical relay response was also investigated without having the power system data processed by the filters. The simulations were conducted for the sampling frequencies of 720 Hz, 1,440 Hz, 1,800 Hz and 2,160 Hz. The voltage and current phasors were calculated and the impedances seen by the relay are calculated. The cutoff frequency for the low-pass filters was selected to be at one-third of the sampling frequency.

6.2 Implementation of the DFT Algorithm

The basic steps followed for the DFT algorithm to run optimally were as follows:

1) The complete application was designed carefully as a block diagram, so that the overall structure is appropriate for the application
2) Each module was carefully coded in assembly and C programming languages
3) Areas in the code were located where more time is consumed
4) The codes were optimized using T.I. software development tools
5) Memory optimization was done

The reference sine and cosine coefficients for different sampling rates were computed. The input power system data was scaled down to suitable levels of ± 1V to be compatible with the ADC input level. The output digital data from the ADC was used to calculate the phasor estimates using the DFT algorithm.

6.2.1 Implementation Using C Language

The DFT algorithm was coded in C programming language and optimized using T.I. compiler directives. The sine and cosine reference coefficients were generated for different sampling rates. The peak phasor values were estimated and the time taken for the execution was compared for different sampling rates.

The time for the execution of all the instructions of the DFT was compared for different sampling rates. The execution times taken by DFT when coded in C language are listed in Table 6.1. A plot of the execution time in microseconds versus different sampling rates for C coded programs is shown in Figure 6.1.

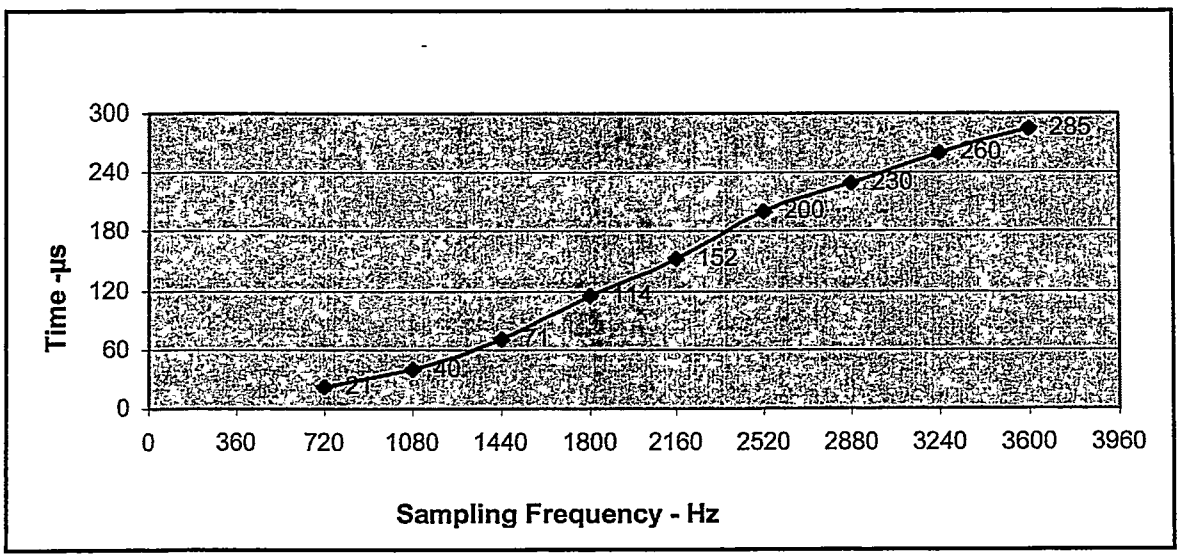


Figure 6.1 Sampling rate versus time for C programs

6.2.2 Implementation Using Assembly Language

The DFT algorithm was coded in assembly programming language. The sine and cosine reference coefficients were stored in a fixed memory location and called when the algorithm is executed. The peak phasor values were also estimated in the assembly language.

The execution times taken by DFT when coded in assembly language are listed in Table 6.1. A plot of the execution time in microseconds versus different sampling rates in Hz is shown in Figure 6.2.

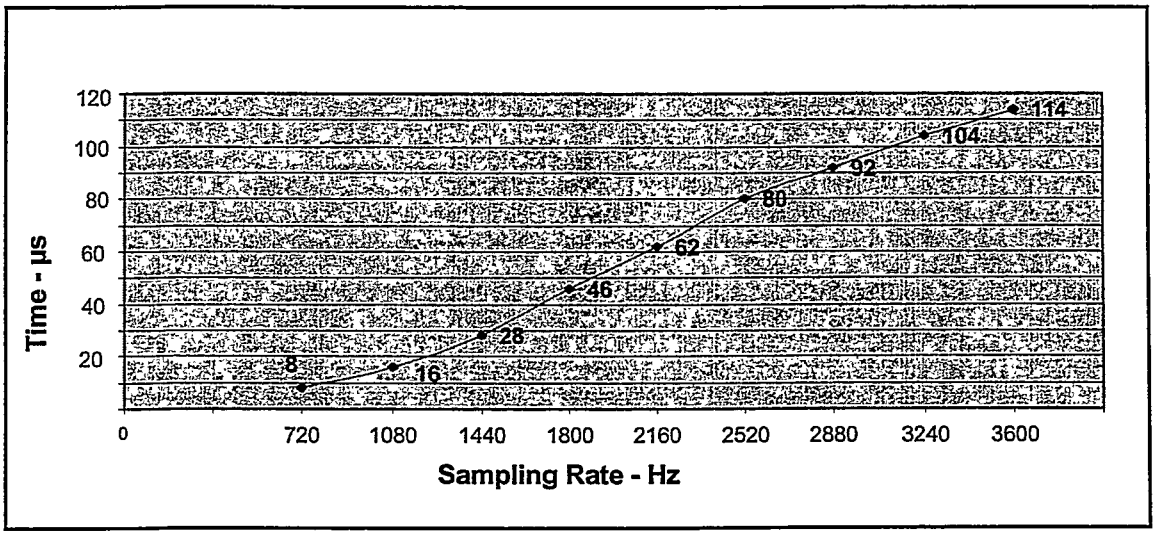


Figure 6.2 Sampling rate versus time for Assembly programs

Table 6.1 Execution times for different sampling frequencies for C and Assembly programs

Sampling Frequency (Hz)	Coded in C (Execution Time in µsec)	Coded in Assembly (Execution Time in µsec)
720	21	8
1080	40	16
1440	71	28
1800	114	46
2160	152	62
2520	200	80
2880	230	92
3240	260	104
3600	285	114

6.2.3 Comparison of DFT Programs Written in Assembly and C Languages

The comparison of the execution times for computing DFT using the assembly and C languages showed that the assembly language programs are much faster than the programs written in a C language. For numerical relays, therefore, every effort should be made to write programs in the assembly language.

Assembly language is close to a one-to-one correspondence between the symbolic instructions and the executable digital signal processor codes. They also include directives to the assembler, directives to the linker and directives for organizing data space, thus making their execution faster. Assembly language programs are more difficult and time consuming to write than the high-level language programs. The architecture of the processor must be studied carefully and proper care and attention must be taken to write the code. If assembly language programs are coded properly, they run much faster than a similar program written in a high level language. High-level language programs are easy to write and implement. They allow faster development times than assembly programs.

6.2.4 Application Downloading to Processor Network

The procedure followed for the creation of an executable application using Diamond RTOS is described in Chapter 5. The executable application files were downloaded to the C6x processor network and executed. The peak phasor values computed using the DFT algorithm and the execution times for all the DFT instructions are displayed in the Diamond server window. The application download process on to the processor network is shown in Figure 6.3.

The downloaded DFT algorithm on the processor network was run, the outputs were recorded, plotted and compared. Figures from 6.4 to 6.9 display the estimates of the phasors for sampling rates of 720 Hz, 1,200 Hz, 1,440 Hz, 2,440 Hz, 2,880 Hz and 3,600 Hz, respectively, with and without the low-pass filters for phase-A of the power system. The peak estimates using DFT for phase-B and phase-C are also be similar to the ones

shown. From these figures, it can be observed that at higher sampling frequencies the DFT algorithm requires an increased number of samples to estimate the peak values of the phasors.

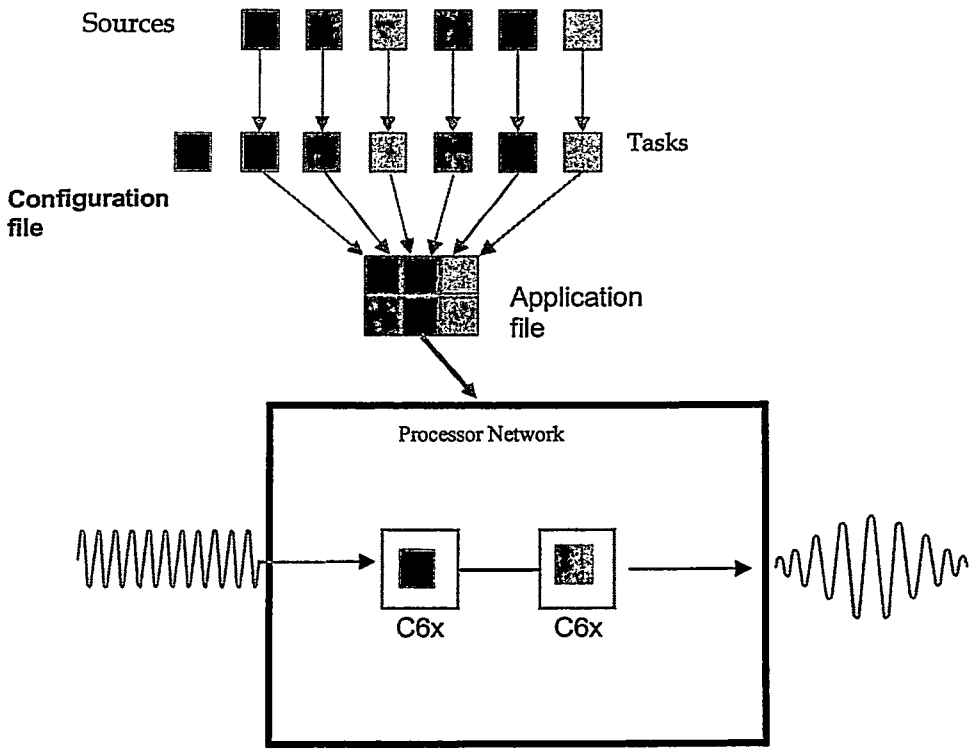
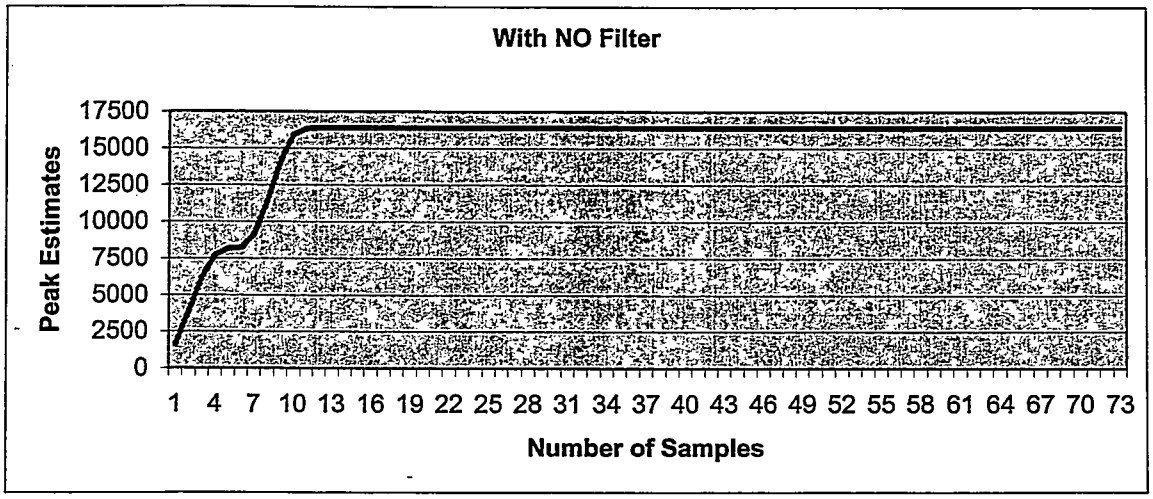


Figure 6.3 Application loading to processor network



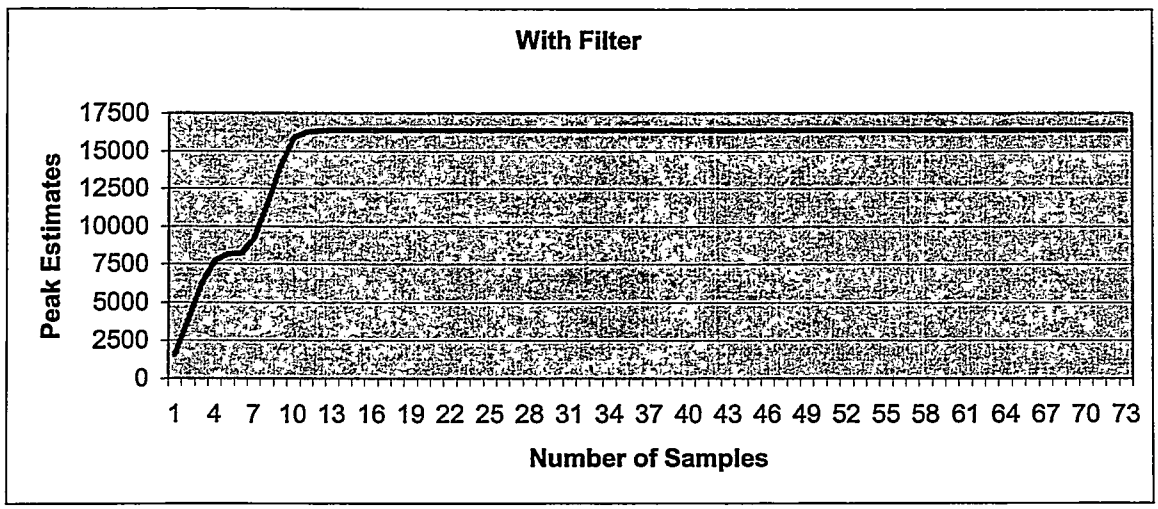
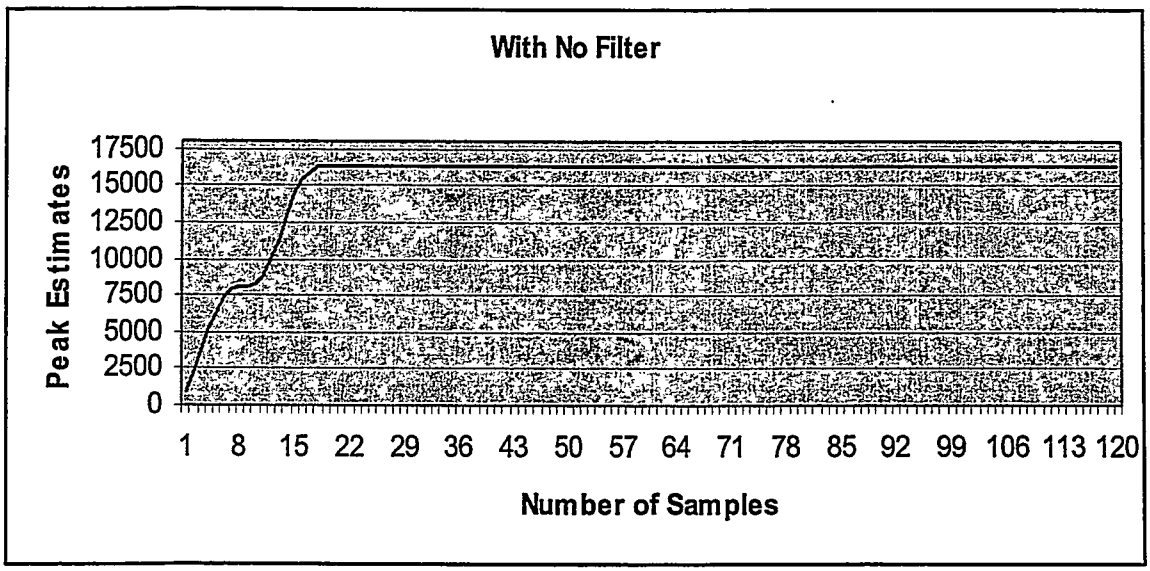


Figure 6.4 DFT peak value estimates for sampling rate of 720 Hz without filter and with filter



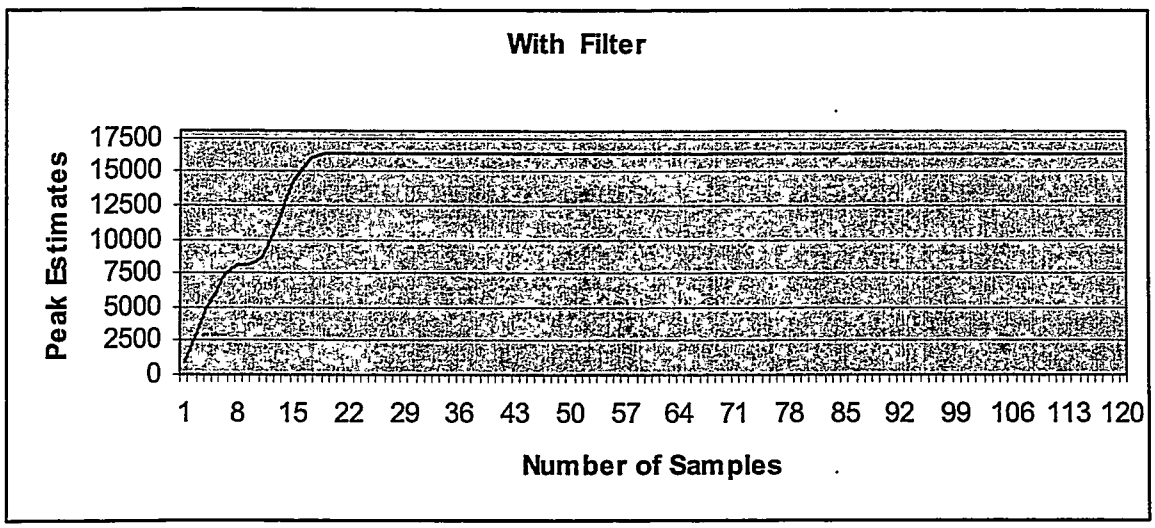
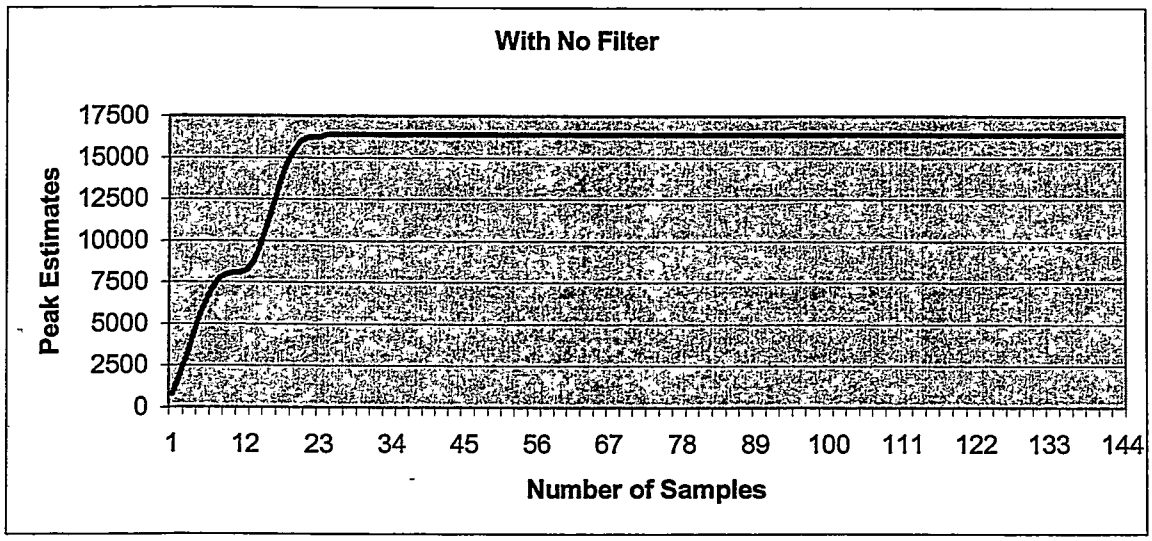


Figure 6.5 DFT peak value estimates for sampling rate of 1,200 Hz without filter and with filter



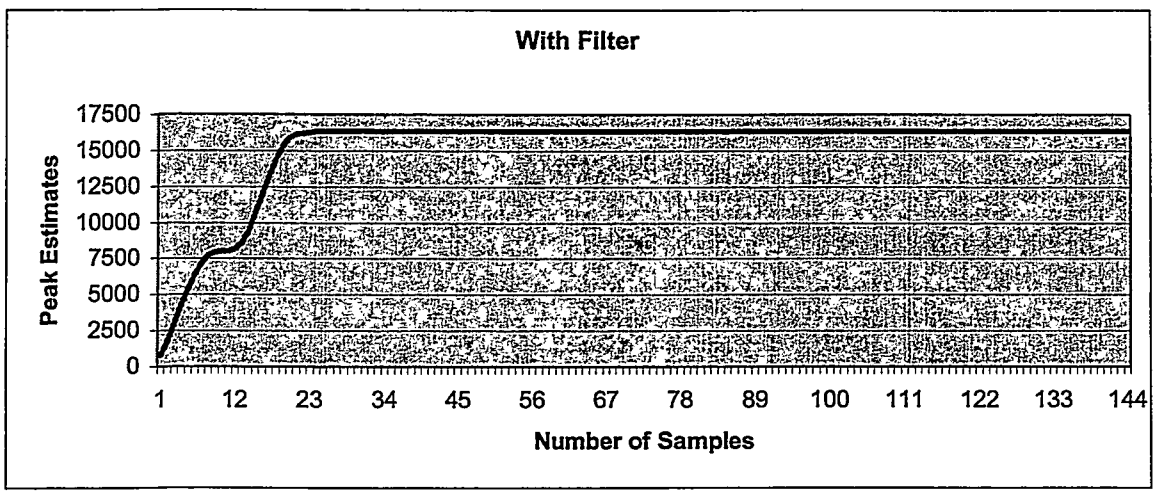
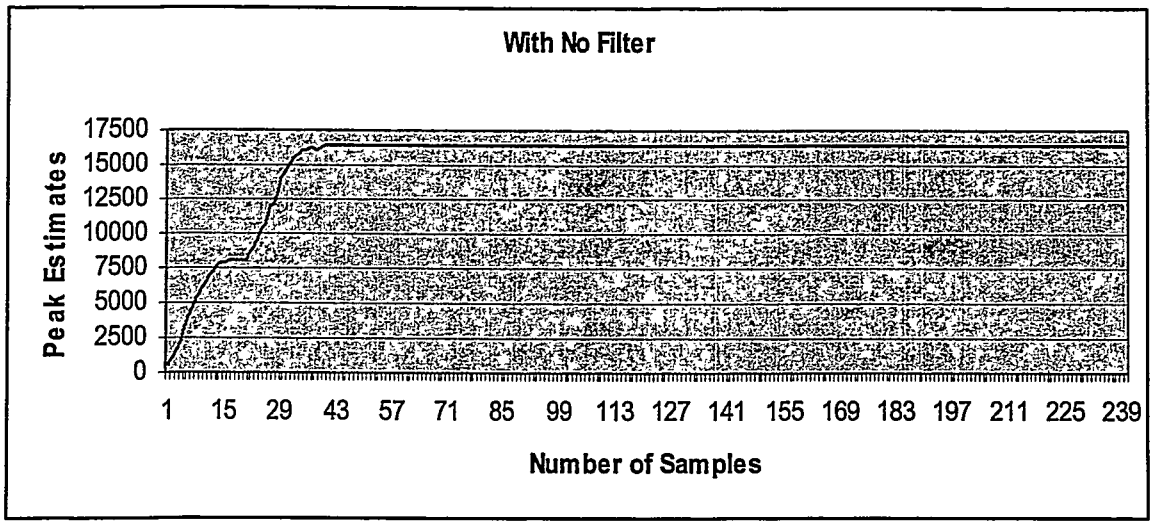


Figure 6.6 DFT peak value estimates for sampling rate of 1,440 Hz without filter and with filter



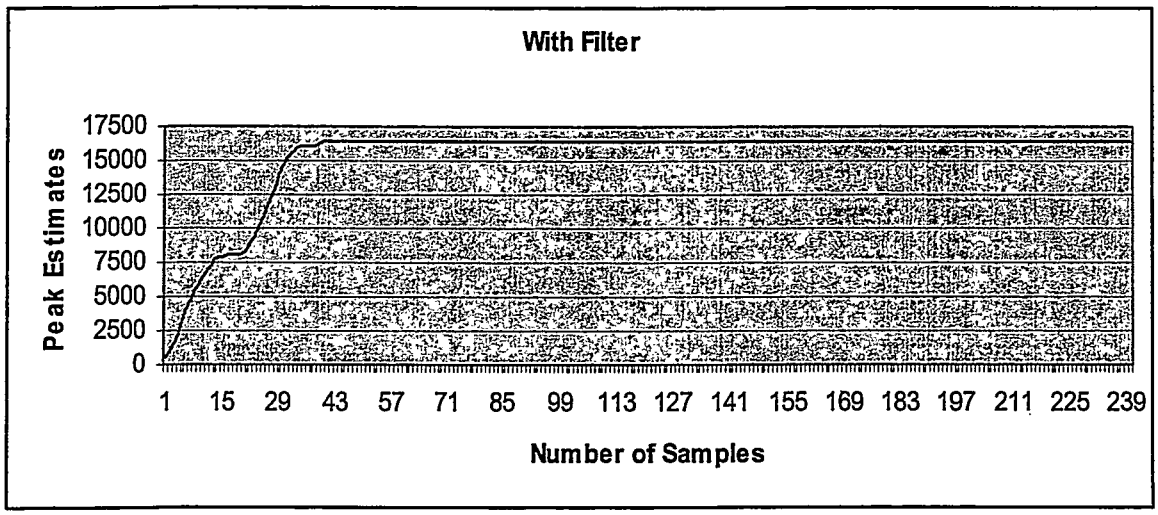


Figure 6.7 DFT peak value estimates for sampling rate of 2,440 Hz without filter and with filter