MODELING AND TESTING OF AN INSTANTANEOUS OVERCURRENT RELAY

USING VTB AND VTB-RT

By

Daxa Patel

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2006
Copyright by

Daxa Patel

2006
MODELING AND TESTING OF AN INSTANTANEOUS OVERCURRENT RELAY
USING VTB AND VTB-RT

By
Daxa Patel

Approved:

Noel N. Schulz
Associate Professor of
Electrical and Computer Engineering
(Major Advisor and Director of Thesis)

Wenzhong Gao
Assistant Research Professor of
Electrical and Computer Engineering
(Committee Member)

Herbert L. Ginn
Assistant Professor of
Electrical and Computer Engineering
(Committee Member)

Nicholas H. Younan
Professor of
Electrical and Computer Engineering
(Graduate Coordinator)

Roger L. King
Associate Dean for Research and
Graduate Studies of James Worth Bagley
College of Engineering
This thesis explores the application of the Virtual Test Bed (VTB) and its real-time extension, VTB-RT, for protective relay modeling, simulation, and testing. An instantaneous overcurrent relay model was developed in VTB for a transmission line protection. The same relay model was built in Matlab/Simulink for validation purposes. Both models were tested for various fault conditions on a radial power system and results were compared. Moreover, a low cost real-time Hardware-In-the-Loop (HIL) simulation platform was implemented for relay model testing using VTB-RT and public domain software packages such as Real-Time Application Interface (RTAI), Comedi, and Comedilib, and notebook computer hardware. The applicability of VTB-RT was verified through an open-loop simulation and a HIL simulation of a simple dynamic system using dSPACE as the control hardware and NI DAQCard-6062E as the input/output interface. Simulation results are presented showing the effectiveness of the VTB-RT platform for model testing.
DEDICATION

This thesis is dedicated to my parents, Bhagvanjibhai Patel and Kantaben Patel, and my husband, Paresh Patel, who have been supporting and encouraging me during my graduate study.
ACKNOWLEDGMENTS

I am deeply indebted to my advisor, Dr. Noel Schulz, for her throughout help and support during my graduate study. Her guidance and invaluable suggestions have helped me to complete this work successfully. I would also like to thank my thesis committee members, Dr. Wenzhong Gao and Dr. Herbert L. Ginn, for their support.

I gratefully acknowledge the Office of Naval Research (ONR) for providing financial support during my graduate program.

I would like to extend my special thanks to Dr. Hernan Figueroa from University of South Carolina and Dr. Wenzhong Gao for their help and guidance in successfully completing this research.

I also like to acknowledge the E-ship research team members of Department of Electrical and Computer Engineering: Dr. Yanfeng Gong and Hymiar Hamilton for their support and cooperation as colleagues. Finally, I would like to thank my parents-in-law for their blessing and encouragement.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEDICATION</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE</td>
<td>xi</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td><strong>I. INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Power System Protection</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Real-Time Simulation</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Model Development for Protection Devices</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Thesis Organization</td>
<td>5</td>
</tr>
<tr>
<td><strong>II. BACKGROUND AND RELATED WORK</strong></td>
<td>6</td>
</tr>
<tr>
<td>2.1 Protective Relays</td>
<td>6</td>
</tr>
<tr>
<td>2.1.1 Functional Characteristics of Protective Relaying</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2 Types of Protective Relaying</td>
<td>9</td>
</tr>
<tr>
<td>2.2 The Virtual Test Bed (VTB)</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1 Important features of VTB</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Real-Time HIL Simulation</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1 Real-Time VTB (VTB-RT)</td>
<td>13</td>
</tr>
<tr>
<td>2.4 Survey of Prior Work</td>
<td>14</td>
</tr>
<tr>
<td>2.4.1 Protective Relay modeling</td>
<td>14</td>
</tr>
<tr>
<td>2.4.2 Real-Time HIL Simulation</td>
<td>16</td>
</tr>
<tr>
<td>2.5 Summary</td>
<td>18</td>
</tr>
<tr>
<td><strong>III. PROTECTIVE RELAY MODELING</strong></td>
<td>19</td>
</tr>
</tbody>
</table>

iv
CHAPTER

5.2.1 Testing Procedures ........................................... 73
  5.2.1.1 Software Part Testing ................................. 73
  5.2.1.2 Hardware Part Testing ............................... 81
  5.2.1.3 VTB-RT Closed-loop (HIL) Simulation ............... 83
5.3 Summary ......................................................... 92

VI. CONCLUSIONS AND FUTURE WORK ............................ 93
  6.1 Conclusions .................................................... 93
  6.2 Future Work ................................................... 95

REFERENCES .......................................................... 96

APPENDIX

A. INSTALLATION OF FEDORA CORE 3 ............................ 99

B. RELAY REPORT ................................................... 102
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Hardware Requirement for VTB-RT</td>
</tr>
<tr>
<td>4.2</td>
<td>VTB-RT Platform at MSU</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Protective Relays Functional Elements [32]</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Major Components of a Digital Relay [32]</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Website of VTB</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Radial Protection System</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Logic diagram of instantaneous overcurrent scheme</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>SEL-351S Relay at Mississippi State University</td>
<td>23</td>
</tr>
<tr>
<td>3.4</td>
<td>Digital Relay Elements</td>
<td>24</td>
</tr>
<tr>
<td>3.5</td>
<td>An Instantaneous Overcurrent Relay Model in VTB</td>
<td>28</td>
</tr>
<tr>
<td>3.6</td>
<td>An Instantaneous Overcurrent Relay Model in Matlab/Simulink</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>VTB-RT structure [20]</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>Website of Linux Distributions</td>
<td>41</td>
</tr>
<tr>
<td>4.3</td>
<td>Website of Generic Linux Kernels</td>
<td>43</td>
</tr>
<tr>
<td>4.4</td>
<td>Latency Test Outputs</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>Comedi Test</td>
<td>52</td>
</tr>
<tr>
<td>5.1</td>
<td>VTB Relay Model Testing for a Single Phase to Ground Fault</td>
<td>58</td>
</tr>
<tr>
<td>5.2</td>
<td>VTB Simulation Results of Single Phase to Ground Fault</td>
<td>59</td>
</tr>
<tr>
<td>5.3</td>
<td>VTB Relay Model Testing for a Line to Line Fault</td>
<td>60</td>
</tr>
</tbody>
</table>
5.25 Simulation Result in VTB-RT ................................................................. 82
5.26 Structure of the VTB-RT Open-loop System [27] ............................. 83
5.27 Open-loop Simulation Circuit in Windows Version VTB ...................... 84
5.28 Open-loop Simulation Result in VTB-RT .............................................. 84
5.29 Structure of the Real-time HIL Simulation System [27] ...................... 85
5.30 Schematic Diagram of the Closed-loop Simulation in Windows Version VTB 85
5.31 Closed-loop Simulation Result in Windows Version VTB ...................... 86
5.32 Schematic Diagram of the Real-time HIL Simulation System ............... 87
5.33 The HIL Simulation Setup on VTB-RT Platform at MSU .................... 88
5.34 Closer Look of the dSPACE Setup for the Real-time HIL Simulation ...... 89
5.35 Plant Model for the Real-time HIL Simulation in Windows Version VTB ... 89
5.36 Controller for the Real-time HIL Simulation in dSPACE ...................... 90
5.37 The Real-time HIL simulation Result in VTB-RT ................................. 91
LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

$V TB$ Virtual Test Bed
$V TB – RT$ Real-Time VTB
$H IL$ Hardware-In-the-Loop
$RT AI$ Real-Time Application Interface
$DAQ$ Data Acquisition
$NI$ National Instruments
$SEL$ Schweitzer Engineering Laboratories
$CAPE$ Computer-Aided Protection Engineering
$USC$ University of South Carolina
$ZOH$ Zero-Order Hold
CHAPTER I

INTRODUCTION

1.1 Power System Protection

A power system needs protection to minimize the effects of faults. A fault is nothing but an abnormal system condition. It is a random event. If the faulted parts of the power system are not quickly removed from the rest of the system, the fault may cause instability in the power system and may damage equipment. Therefore, the protection system must isolate the faulted parts of the system from the rest of the power system as quickly as possible to maintain continuity of the power flow to the rest of the system.

A protection system consists of protection devices such as relays, fuses, breakers, and sectionalizers that detect and quickly remove faults from the power system. A relay is a protection device which responds to the condition of its inputs in such a way that it provides appropriate output signals to trip circuit breakers and isolate the fault. It is the most important component of the protection system. Currently, three main technologies are available in protective relays:

1. Electromechanical Relays:

In the early 1900s, relays were electromechanical types that consisted of plungers, balanced-beams, and induction discs or cups. They were based on mechanical principles.
In these relays, line voltages and currents were used to actuate the moving parts. When these values are too large or out of limits, the mechanical parts of the relay would cause the opening or closing of a set of contacts, which in turn causes an isolation of the faulted parts or elements from the rest of the system. They were robust and reliable for terrestrial power system, but they were slow and required a fairly large amount of energy to operate, which demanded relatively high volt-ampere capacity voltage and current transformers. Also, they have limited flexibility to do multiple relaying functions and the ability to communicate to remote devices [13].

2. Solid-State Relays:

Solid-State relays are also called semiconductor relays, which work like normal relays. They are usually called by the acronym SSR. An SSR is a semiconductor device that can be used in place of a mechanical relay to switch electricity to a load in many applications. Solid-state relays are purely electronic, normally composed of a low current control side and a high-current load side. Typically, SSRs provide electrical isolation to several thousand volts between the control and load sides [26].

3. Digital (Microprocessor) Relays:

Recently, digital relays have become very common in protection applications in industry, terrestrial, and shipboard power systems. They are based on microprocessors. The benefits of these relays include accuracy, improved sensitivity to faults, better selectivity, flexibility, user-friendliness, easy testing, and relay event monitoring/recording capabilities [26]. These relays use power electronics to convert line voltage and current signals
into signals that a microprocessor can sense; the power electronics implement a control algorithm in the digital relays. Digital relays are very flexible in terms of multiple protection functions. They provide high reliability, as well as self checking and adaptive relaying capabilities. Also, the costs are low compared to conventional relays. Moreover, the size of digital relays are very small compared to electromechanical relays because they have no moving parts like in electromechanical relays.

This thesis models the SEL-351S digital overcurrent relay and simulates it for different fault scenarios.

1.2 Real-Time Simulation

Traditional simulation software does not have the ability to replicate exact real operational conditions. Many issues arise that traditional simulation software cannot handle. In this case, the only simulation software that can fill the gap between simulation and real conditions is the real time Hardware-In-the-Loop (HIL) simulation. In the real-time HIL simulation, a simulation model directly interacts with the real hardware. In this simulation, a model in the computer gets signals or data from the real hardware under test and sends back signals or data to the hardware, which provides a more realistic solution than software-only simulation. Today, HIL is a highly recognized technique for many engineering applications such as power electronics, controls, and automotive examples. Many companies and research institutes are working on the development of this type of simulation software hardware. Some are also available in the market, such as RT-LAB,
HyperSim, RTDS (Real-time Digital Simulator), dSPACE, and AutoSim [20]. While RTDS is a very useful simulation tool, the preparation for testing the model in RTDS is time consuming. Also, it is a very expensive simulation tool, and it requires a large space.

The Virtual Test Bed (VTB) is a free simulation software, developed by University of South Carolina, which supports the real-time HIL simulation. The real-time extension of the VTB is called VTB-RT. It is based on the public domain software and standard computer hardware. The VTB-RT was first implemented by University of South Carolina. Therefore, the idea of implementing a VTB-RT is not quite new, but this thesis illustrates the development of VTB-RT platform, using latest versions of open source software and notebook computer hardware.

1.3 Model Development for Protection Devices

This thesis outlines the process of developing a model of a SEL-351S relay for a protection system. The SEL-351S is a Schweitzer Engineering Laboratories overcurrent relay. This thesis develops and tests an instantaneous overcurrent relay model in VTB for different fault conditions. VTB was chosen as the simulation platform for model development. It has many features that make VTB as a unique software tool [28], such as a capability of integrating models that have been created in a variety of languages and advanced displays of simulation results. It also supports real-time simulation. Moreover, VTB is free. While VTB supports many electrical models, such as a source model, load model, and breaker model, there are no protection device models. Doing a complete simulation
of a power system in VTB requires protection device models. Therefore, this thesis supports the development of an instantaneous overcurrent relay model for transmission line protection in VTB. Moreover, author developed a model of same relay in Matlab/Simulink to verify simulation results. Further, the VTB-RT simulation platform was developed to perform Hardware-In-the-Loop simulation of an actual SEL-351S with the same power system model in VTB to verify results.

1.4 Thesis Organization

The thesis is organized through different chapters that describe the background, development of a relay model and real time simulation platform, the testing of the relay model, and, finally, the discussion of results and future work. Chapter II presents background information and a literature survey on digital relays and real-time VTB HIL simulation. Chapter III introduces the types and operating principles of an overcurrent relay, issues related to digital relay modeling and, finally the development of a relay model in different softwares such as VTB and Matlab/Simulink. The open source softwares, Fedora Core 3, RTAI-3.1, Comedi-0.7.69, Comedilib-0.7.22, and the installation of the real time VTB platform are described in Chapter IV. Chapter V covers the test cases and simulation results of an overcurrent relay model in VTB and Simulink for different fault scenarios. Additionally, this chapter also includes the VTB-RT open loop and closed-loop (HIL) simulation test cases and results. Finally, a summary of the presented work and future directions are presented in Chapter VI.
CHAPTER II

BACKGROUND AND RELATED WORK

This chapter describes background material such as protective relays, the Virtual Test Bed (VTB), and real-time VTB (VTB-RT). It also provides a literature survey on prior work on protective relays modeling and simulation on various softwares and the real-time Hardware-In-the-Loop (HIL) simulation.

2.1 Protective Relays

The aim of the protection devices is to limit detrimental power system conditions, such as high currents, over/under voltages, and over/under frequency. The relay is one of the most important protection devices. Basically, it is a controller that measures input quantities and compares them with relay settings, pick-up settings, to generate appropriate output signals for circuit breakers to disconnect the faulty element [29]. The relay input quantities can be voltages, currents, or contact status, depending on the operating characteristics of protective relays.

Relays are essential for the safe and reliable operation of a power system. The primary function of protective relaying is to cause the quick removal of an element of a power system when it suffers a short circuit, or when it starts to operate in an abnormal manner that might cause damage or interfere with the effective operation of the rest of the system. Its
secondary function is to provide an indication of the location and type of failure. Such data helps in repairing and in analyzing the effectiveness of the fault-prevention and removal features.

Usually, the protective relay consists of several elements that help detect the system condition, make a decision if the observed variables are over or under the predefined limit, and take proper action if limits are crossed [32]. Figure 2.1 shows the arrangement of relay elements.

![Protective Relays Functional Elements](image)

Figure 2.1

Protective Relays Functional Elements [32]

The protection system measures system quantities such as voltages and currents and compares these quantities with a threshold setting. If measured quantities are outside the desired range than the thresholds, then a decision element is triggered. Sometimes a timing element is involved in a decision element to determine the timing status of the system. After satisfying all the conditions, the relay (action element) operates.
Today, a majority of protective relays are digital that use signal processing techniques and numerical algorithms to determine an abnormal condition. In each calculation step, samples of system quantities, voltages and currents, are processed. Then the measurement element calculates the fundamental frequency component or desired harmonic component based on the samples values. A decision-making element makes a decision whether the relay will trigger a trip signal. A digital relay can be represented with three major blocks as shown in Figure 2.2.

![Figure 2.2](image)

Major Components of a Digital Relay [32]

The data acquisition block is the front end of the relay that links the digital processing segment of the relay with its analog inputs. The measuring block estimates the input signal parameters such as magnitude, phase angle, resistance, reactance, active power, and reactive power [32]. The third block, a decision making block, applies basic relaying principles to compare estimated signal parameters with predefined thresholds. It also applies certain delays and logic functions in order to execute the tripping and alarm signals.
2.1.1 Functional Characteristics of Protective Relaying

Any protective relaying equipment’s functional characteristics can be described by three terms, “Sensitivity”, “Selectivity”, and “Speed” [18]. These functional characteristics are very important to measure relaying operation. *Sensitivity* is a term which describes reliable relay operation when it is required to operate. Any relaying equipment must be sufficiently sensitive so that it will operate when needed. Also, it must be able to select between normal operating conditions and fault conditions. This means a relay must be able to distinguish those conditions for which quick operation is required and those for which no operation, or a time-delay operation is required. *Speed* is also an important characteristic in relaying operations. Every relay must operate at the required speed to isolate the faulted elements from the system as quickly as possible.

Moreover, protective-relaying equipment must be *reliable* is a basic requirement. Any relay is said to be reliable if it is dependable, and it is secure. Dependability implies that the relay will always operate for conditions for which it is designed to operate. The relay securing implies that it will not operate for any other power system disturbances except those for which it is designed.

2.1.2 Types of Protective Relaying

Many different types of protective relays exist in power systems, such as overcurrent relays, differential relays, and distance relays. All types of relays can be classified according to their input characteristics as summarized below [13]:

1. Magnitude Relays: These types of relays respond to the magnitude of the relay input quantity such as voltages and currents. An overcurrent relay is a magnitude relay responding to the changes in the magnitude of either the peak value or the rms value of the input current. They are suitable for radial power systems.

2. Ratio Relays: The relays which respond to the ratio of the two input signals expressed as phasors are known as ratio relays. The most common type of ratio relays are the various types of impedance or distance relays. They are appropriate with long transmission lines.

3. Directional Relays: These relays respond to the phase angle between two ac inputs. Generally, in this type of relays, the phase angle of a current is compared with a phase angle of a voltage, or the phase angle of one current is compared with that of another current. They are ideal for all power system topologies like radial, ring, and zone.

4. Differential Relays: These types of relays respond to the magnitude of the algebraic sum of the two or more inputs. In current differential relays, the relays respond to the algebraic sum of currents entering a zone of protection. This types of relays are best suitable for a power system which consists of multiple sources.

5. Pilot Relays: In this types of relays, communication information from remote locations is used as an input signal, and based on this information, the relay makes a decision. The pilot relays protect transmission lines when providing a high speed protection to the entire line is needed.

2.2 The Virtual Test Bed (VTB)

The Virtual Test Bed (VTB) is simulation software for design, analysis, and virtual prototyping of large-scale multi-technical dynamic systems. It was developed by the University of South Carolina and is a free software. The main goal behind the development of this software is to provide a “common language” to integrate all interdisciplinary work in one simulation environment [17]. Also, it allows a proof of concept design before hardware construction, which saves time and money. Figure 2.3 shows the homepage of VTB, where users can download the latest version of VTB.
2.2.1 Important features of VTB

VTB has the capability of integrating models that have been created in a variety of languages into one simulation environment. This capability of VTB allows users to simulate any complex system that might consist of some electrical, mechanical, and control components. Generally, simulations of this type of system is difficult using one particular simulation software, since the existing simulation softwares are mostly designed for a specific application. For example, Matlab is very suitable software for control systems applications, whereas SPICE is an excellent tool for power electronics applications. However, VTB allows designers to model each individual component of a complex system in
a language appropriate to that particular model and then bring those individual models to a single environment.

VTB also supports the VXE (Visualization Extension Engine) tool that allows the user to display simulation results, including full-motion animation of mechanical components. Through VXE, the simulation results can be displayed in 3D, which provides a better vision of simulation outputs. VTB provides a distributed environment. That means, network simulations can execute on a single processor, while visualizations can run on a separate machine at the same time [17].

Moreover, VTB supports many different solvers and allows users to create their own custom models. It also allows the user to make run time changes in the simulation model. It will automatically reflect the appropriate changes in the simulation output. One of the most attractive features of VTB is its capability to perform real-time simulation. VTB was developed on two parallel platforms to optimize portability. One is the Windows NT/2000 platform and the other is the Linux/Unix platform [28]. The Linux version of VTB makes it a powerful tool for real-time simulation. All of these important features make VTB an unique simulation tool.

2.3 Real-Time HIL Simulation

Real-time HIL simulation is becoming a very popular and essential simulation tool for engineering design, especially in automotive, controls, and power electronics design [27]. It has the capability to replicate the exact real operational conditions. In real-time HIL
simulation, the actual hardware directly interacts with the simulation model in software, which increases the realism of the solution. Many companies and research institutes are working on the development of such types of simulation software hardware.

2.3.1 Real-Time VTB (VTB-RT)

The VTB is a simulation software; it has the capability to perform real-time HIL simulation. The real-time extension of the VTB is called VTB-RT. Real-time is the most attractive feature of VTB-RT, but the implementation of such a system is very challenging. It is based on open source software and standard PC hardware. Open source software are free so that helps reduce cost, but has the disadvantage of poor supportability. They do not have a good support system as available with commercial software. Therefore, users must deal with some source code problems.

In VTB-RT HIL simulation, the deadline for each simulation step is well-defined, and the HIL system must generate the response from the hardware within a fixed time interval, which lead a system that can support hard real-time. Due to the limitations of a Windows operating system, hard real-time support is not available in the Windows version of VTB [27]. Therefore, the Linux/Unix based VTB, VTB-RT is ideal for real-time applications. Moreover, Linux has many advantages over Windows, particularly in real-time applications. One advantage of Linux is its low cost. It is a free operating system. Any one can easily download it from the web without any cost. In VTB-RT, the system framework is based on a specific Linux distribution like Fedora, Mandrake, or Redhat. Other advan-
tages of Linux over Windows are its good performance and high reliability. Experience shows that Linux supports real-time simulation better than Windows NT/2000. It is a more flexible system for designers. In the Windows system the user has no or very limited permission capability. Also, Linux gives a better performance in a real-time system even with older and slower hardware. Since Linux is based on a stable Unix operating system, it has all the stability and reliability as Unix.

2.4 Survey of Prior Work

A relay is an important component of a protection system. To do a complete simulation of a power system in a software requires various equipment models, like load model, source model, breaker model, relay model and so on, in that software. After modeling the power system using those models in a software, the best way to test that system is through real-time Hardware-In-the-Loop simulation, because it contains real signals and real hardware. Normally, in conventional simulation, the time frame is different than the one in real world. But using the real-time HIL simulation, the user can replicate the real operating conditions, which leads to more realistic results. This section presents and discusses a literature survey on the related work.

2.4.1 Protective Relay modeling

Protective relaying is a complex engineering field that requires knowledge of many different engineering disciplines such as power systems, controls, signal processing, computers, and electromagnetics. M. Kezunovic [22] describes a modeling and simulation
software, specially developed for protective relaying applications and design. In this article, the new libraries of signal sources and relay elements are developed in Simulink environment of Matlab. Using these libraries with Power Block Set (PBS), one can model and test any protective relay in Matlab very easily. Also, an example of an overcurrent relay modeling using these libraries is given.

M. Kezunovic et al. [24] indicates the issues related to modeling a protective relay in software. The contemporary protective relays are becoming more and more sophisticated due to digital technology, which requires elaborate simulation tools. This paper introduces the new Matlab-based software, which has the capability to model digital relays. Additionally, it describes the application of programmable relays for developing real-time hardware models of digital relays and the use of digital simulators to test physical relays.

Matlab/Simulink is a good simulation tool for modeling digital relays. Using the capability of this software, L. Wu et al. [35] model and test a digital distance relay for transmission line protection. They used Simulink’s Power System Blockset (PSB) for modeling a power system network and fault simulation, and MATLAB for implementing programs of digital distance relaying algorithms. Also, they describe some filtering techniques which are very important in relay modeling. Moreover, a MHO type distance relay is selected as an example for modeling and testing.

The digital relays are one type of numerical relays that use signal processing techniques and numerical algorithms to calculate the fault. Each digital relay contains several elements that help detect the system condition, make a decision, and take a proper ac-
tion. A digital filter is a very important element of protective relays. Protective relays must filter inputs to reject unwanted signal quantities and keep the signal quantities of interest. Schweitzer et al. [33] talk about filtering requirements for protective relaying and design characteristics of a filter. This article reviews and compares many different types of filters for protective relaying, such as cosine, Fourier, correlator, least-squares and Kalman. The results of their research indicate that the one-cycle cosine filter is the best filter among all the filters that are evaluated in this article, because it rejects all harmonics and exponentially-decaying dc offsets. Also, it has a good transient response. S. E. Zocholl et al. [38] present the 16 sample/cycle full cycle cosine filter in equation form. Moreover, this article also presents the implementation of this filter in Mathcad 6.0.

Almost all the previous protective relaying articles show that Matlab/Simulink, a commercial software, is a useful tool to model and test protective relays. This thesis uses an open source software, VTB, as a simulation tool to model and test an instantaneous over-current relay for a transmission line protection. The same relay is modeled in Simulink to compare and validate the results.

2.4.2 Real-Time HIL Simulation

As mentioned earlier, the real-time HIL simulation has gained recognition as an essential tool for various engineering applications such as embedded control, power electronics, aerospace, and automotive applications. The literature is full of examples of applications of this technology. The purpose of the development of real-time HIL simulation technologies
is to test various hardware design prior to actual hardware construction and on-the-field testing. J. L. Page and K. E. Wills [30] address the Hardware-in-the-Loop techniques for developing modern missiles. While P. Baracos et al. [16] describe new technologies that use affordable PC hardware as a computing platform for high-performance automotive applications. In [14], E. Acha et al. present the use of real-time simulator for power quality disturbance applications. All these articles represent the use of real-time HIL technologies to solve various engineering problems.

X. Wu et al. [36] proposed a rapid, low-cost prototyping and testing procedure for digital controller design using the real-time HIL simulation. This paper designs a digital controller for an H-bridge inverter, implements it in VTB and tested on VTB-RT HIL simulation platform. The results of this article suggest that RT HIL makes the testing of new control equipment fast, safe and reliable.

VTB is a new environment for design, analysis, and virtual prototyping of multidisciplinary systems [36], and it has the capability to perform the real-time simulation [17, 36, 28, 37]. B. Lu et al. [28] demonstrate the possibility of developing the real-time HIL simulation platform using VTB. This article shows the use of public domain software, including Linux, RTAI, Comedi and standard PC hardware to develop real-time HIL simulation platform using VTB.

The above articles show that real-time HIL simulation is an ideal simulation tool to test and verify hardware design. Most of the papers and references on real-time HIL simulations are on controls, automotive, and power electronics applications. Few publications
are available on power systems applications. The lack of papers and references concerning power Hardware-in-the-Loop simulation suggests the need for more research in this area. This thesis demonstrates the implementation of VTB-RT, the hard real-time HIL simulation environment, using the Linux/Unix version of VTB, the latest release versions of public domain software, and a laptop as hardware. The purpose of the implementation of VTB-RT HIL simulation platform is to use it for verifying various power systems equipment models. This thesis uses it to validate the model of an instantaneous overcurrent relay that author built in VTB and Simulink.

2.5 Summary

This chapter provides necessary background information related to this research on protective relays types, relays functional elements and characteristics, VTB, VTB-RT, and real-time HIL simulation. Additionally, this chapter reviewed the previous research works on digital relay modeling and real-time HIL simulation. Lack of publications in the area of using a VTB, a free simulation software, for modeling of protective equipments and testing of developed models using the real-time extension of VTB (VTB-RT) suggests that this area needs further study. Chapter III will detail the modeling steps of an instantaneous overcurrent relay in VTB and Simulink.
CHAPTER III

PROTECTIVE RELAY MODELING

The previous chapter discusses protective relaying, VTB, and real-time HIL simulation, along with a literature review. This chapter describes the methodology of developing a model of the SEL351S relay - an instantaneous overcurrent relay in two different software packages: VTB and Matlab/Simulink. VTB was chosen as a simulation platform because of its useful features, as mentioned in the previous chapter. Also, using the real-time capability of VTB, a real-time HIL simulation platform was developed to test and validate the relay model, so that later on, this relay model can be used in place of a real relay to perform power system simulation with protection devices in VTB. It also provides the flexibility to use multiple relay models as needed at the same time. Furthermore, the same relay model was developed in Matlab/Simulink to allow a comparison of the simulation results.

The first part of this chapter indicates the types and protection principles of an overcurrent relay. It details the logic of an instantaneous overcurrent relay and setting range of pickup taps. Next, it discusses the modeling criteria and elements of a digital relay, along with simulation tools. Finally, the developed relay model in different software is presented for various fault scenarios.
3.1 An Overcurrent Relay Overview

The overcurrent relay is one type of magnitude relay in which current magnitude is used as an indication of a fault. This relay responds to the changes in the magnitude (either the peak value or the rms value) of the input current in such a manner that it provides appropriate output signals to trip the circuit breakers. It is the simplest relay among all protective relays used in power system protection. It is widely used at all voltage levels to protect transmission lines, transformer, generators, and motors. Generally, overcurrent relays can be classified in two types [8]:

1. Instantaneous overcurrent relay:
   It operates without any intentional time delay. They are normally used for faults close to the source when the fault current is very high so that the relay can detect and respond to a fault in a few cycles.

2. Time-delay overcurrent relay:
   This relay combines time and magnitude for determining when the breaker should be opened. If a large fault current is sensed, it will open quickly while for a smaller fault current, the relay will delay its signal to open a breaker

3.1.1 Principle of Protection

The instantaneous overcurrent protection scheme is the simplest protection scheme. It is widely used because of its quick reaction time. Figure 3.1 shows the radial overcurrent protection system along with the placement of protection components such as current transformer (CT), circuit breaker, and relay for transmission lines protection. As shown in the figure, the overcurrent relay is connected between a circuit breaker and the current transformer. The relay receives input signals from the secondary winding of the CT and
sends control commands to the circuit breaker to open or close its contact. The CT is necessary in the power system, because it reduces the high magnitude currents of power system to more manageable levels, so that low energy devices such as relays work in the system. Its function is to reproduce a current in its secondary winding that is proportional to the primary current. It converts primary currents in the kiloamperes range to secondary currents in the 0-5 amperes range for convenience of measurement. The secondary rating of CT has been standardized at 5 amperes or 1 ampere, which implies that the maximum load current in the primary winding of the CT would produce 5 amperes or 1 ampere or less in its secondary winding. This leads to a desired CT winding ratio.

![Figure 3.1 Radial Protection System](image)

The circuit breaker is another important component of a protection system. Its function is to isolate the faulted circuit by interrupting the current at or near zero crossing. In normal operating conditions, the breaker contacts are closed. But when a fault occurs on the power
system, these closed contacts of circuit breaker are moved to the open position, through relay control commands, to isolate the fault.

A simple logic diagram of an instantaneous overcurrent protection scheme is shown in Figure 3.2. Basically, an instantaneous overcurrent relay is a comparator as shown in Figure 3.2 in which the magnitude of the relay input current is compared with threshold (pickup setting) value to check the fault status in the power system.

In an instantaneous overcurrent relay, the proper setting of threshold is very important. Normally, a threshold value is set well above the nominal load current, so that the relay can release a trip signal to the breaker as soon as the current exceeds this threshold. The

Figure 3.2
Logic diagram of instantaneous overcurrent scheme

\[
|I| \geq I_p \quad \text{Fault, trip} \quad (3.1)
\]

\[
|I| < I_p \quad \text{No fault, do not trip} \quad (3.2)
\]

where, \( I \) is the current in the relay and \( I_p \) is the pickup setting of the relay.
above equation 3.1 describes the fault condition while equation 3.2 describes the normal operating condition.

![Figure 3.3](image)

**Figure 3.3**

SEL-351S Relay at Mississippi State University

In this research, the author developed a model of a SEL-351S overcurrent relay that is available from the Mississippi State University Power Systems Laboratory. Figure 3.3 shows the front faceplate of the SEL-351S overcurrent relay. Generally, the setting range for pickup settings (threshold) of SEL-351S relay is [10]:

- 0.25 - 100.00 A secondary (5 A nominal phase current inputs, $I_A, I_B, I_C$)
- 0.05 - 20.00 A secondary (1 A nominal phase current inputs, $I_A, I_B, I_C$)

### 3.2 Digital Relay Elements

An overcurrent relay is a magnitude relay, which responds to the magnitude of the input quantity. Developing a model of an instantaneous overcurrent relay in software need values of 3-phase currents. Current signals from the power system through secondary
windings of CT are in analog form, but to build a model of the relay, the signals must be converted to digital form. Figure 3.4 shows the relay elements, which are used in this thesis to develop a model of an instantaneous overcurrent relay:

![Figure 3.4](image)

Digital Relay Elements

### 3.2.1 Analog to Digital Converter

In an instantaneous overcurrent relay, the relay inputs are current signals in analog form. These signals must be converted to voltage signals suitable for conversion to digital form. This conversion is done by the Analog to Digital Converter (ADC). This research implements an analog to digital conversion block using a Zero-Order Hold (ZOH) device. Normally, an ideal sampler and ZOH are equivalent to ADC. Since an ideal sampler is not a physical sampler, a ZOH is used in modeling, which works similarly to the sampler and the ZOH device, in a physical sense, to convert an analog signal to a digital signal.
3.2.2 Filtering

The digital filter is a very important element of a microprocessor (digital) relay. The behavior of digital relays is directly dependent on the output of the digital filter. Different relays have different filtering requirements depending on the protection algorithm principle. Therefore, choosing the right filter for a particular relaying application is very important. For an overcurrent relay, the filter should be able to preserve the fundamental frequency component and reject the other components.

After conversion of analog signals to digital signals using ZOH, a digital filter block fulfills the filtering requirements of the relay. Digital filtering is nothing but the process of multiplying the successive samples by predefined coefficients and then combining them to obtain digital quantities representing the phasor components of the input. Normally, a 16-sample/cycle cosine filter is suitable for protective relaying [38].

3.2.3 Phasor Calculation

After filtering the relay input signals, converting them to phasors is very important. The phasor can be represented by two forms: rectangular form and polar form. In this thesis, the author used rectangular form, where the real and imaginary components define the phasor. The Discrete (Digital) Fourier Transform technique is the most popular technique for estimating the phasors for the digital relaying.

The Discrete Fourier Transform calculation determines the real and imaginary parts of each relay input currents. In order to get real and imaginary components, two signals, 90
electrical degrees apart, are required. The sine and cosine coefficient meet this requirement, so the real and imaginary components of phasor can be obtained by just multiplying each filtered current sample by a sine factor and a cosine factor. Another approach to obtain real and imaginary components of phasors is to use only the cosine coefficients, with the second signal being the cosine value that was calculated 90 electrical degrees ahead of time [21]. In this research, the second approach is used to calculate the real and imaginary components of the current phasor.

### 3.2.4 Logic Gates

Once the phasor of the relay input signal is calculated, finding the absolute value of the signal is easy. Then, a comparator or relational element compares the absolute value of the relay input signal with the pickup setting (threshold value) to check the status of the power system. If a fault exists in the protection zone of the system, then the absolute value of the relay input signal is greater than the pickup setting; otherwise it is smaller than the pickup value. The output of the comparator is the “Trip” signal for circuit breaker, if fault exists within the zone of protection.

### 3.3 Tools

The main tools used in this research work to develop a model of an instantaneous overcurrent relay are VTB from the University of South Carolina [11] and Matlab/Simulink from Mathworks [6].
3.3.1 Relay Modeling in VTB

VTB is a good software tool, designed by the University of South Carolina, for power engineering applications. Many libraries are available in it that help the user to model and simulate any complex system, including protective relays in it. This work develops the model of an instantaneous overcurrent relay for transmission line protection using the electrical and control libraries. Developing a model of an instantaneous overcurrent relay in software requires three phase relay input signals. Basically, the relay input signals are the signals from the secondary side of the CT. Once the current signals in the primary side of the CT are available, getting relay input signals is easy by using appropriate CT ratio.

In this research work, the standard CT ratio of 600:5 is used to develop the relay model. After having a relay input signals, the complete model of microprocessor based instantaneous overcurrent relay is developed using the above mentioned relay elements such ZOH, Comparator, and OR gate models available in the control library. As discussed above, the digital filter is one of the most important elements in digital relays, but no digital filter model exists in VTB. Therefore, using wrapper capability of VTB, a complete relay model was developed. The VTB wrapper allows user to use Simulink model within VTB simulation. Figure 3.5 shows the schematic layout of the developed instantaneous overcurrent relay model in VTB. The schematic diagram of the VTB relay model is divided into three parts according to elements. Part A represents the conversion of input current signals in analog form to relay input signals in digital form, while part B represents the
filtering and phasor calculation of the current signals. Part C of the relay model represents
the comparator, determining whether there is fault occurred in the system.

Figure 3.5
An Instantaneous Overcurrent Relay Model in VTB

3.3.2 Relay Modeling in Matlab/Simulink

This research additionally uses Matlab/Simulink as a simulation tool to model an
instantaneous overcurrent relay. The relay model is developed in Simulink to compare
and validate the simulation results. Particular factors that support the selection of Mat-
lab/Simulink for model validation are [22, 23, 24, 25]:

1. Matlab/Simulink already dominates in the university environment and is also a more
   and more recognized software in industries including the power industry.
2. Its flexible structure, including libraries, models, and programs enables users to integrate different models in one package easily.

3. It provides a friendly and open system, which helps users to add new models and libraries without having a detailed knowledge of the existing parts.

4. A wide selection of Toolboxes is already available in Matlab/Simulink, which enables the user to model and simulate any complex system, for example, protective relays.

5. Simpower System Blockset in Simulink enables the modeling of the basic components of the power systems.

Due to the above mentioned features, Simulink is selected as a simulation tool to verify the VTB relay model. After having a relay input signals, the relay model was developed using the control and DSP Blocksets in Simulink, exactly the same way as in VTB. Figure
3.6 shows the schematic layout of the Simulink relay model. Just like the VTB relay model, the Simulink relay model can be divided into main three parts: part A conversion of analog input signals to digital signals, part B filtering and phasor calculation, and part C comparator to check the status of the power system. To compare the results of VTB simulation, the relay model in Simulink experiences the same types of faults simulations. Chapter V, Results and Discussion, discusses the relay model test cases and simulation results.

3.3.2.1 Relay Model Testing in CAPE

CAPE is a very powerful software package for protection engineering, designed and partially implemented by the Georgia Power Company of Atlanta, Georgia, and Electrocon International Inc., of Ann Arbor, Michigan [19, 31]. In CAPE, the power system circuit and associated protection equipment can be modeled and simulated easily. The latest information about CAPE can be found on its website http://www.electrocon.com/cape/cape.htm [1]. It also has a large database of relay models for almost all commercially available mechanical and digital relays. In CAPE, specific relay settings are the same settings as in the actual relay. Also, the user can have flexibility to download settings from an actual digital relay in the field to CAPE, or CAPE relay settings can be uploaded to a digital relay [15].

In this research, CAPE tests the developed instantaneous overcurrent relay model and gets the settings for real-time hardware-in-the-loop (HIL) simulation of SEL351s relay in VTB-RT. First, using one-line diagram module of CAPE, the radial power system is
modeled. Then, with the help of the relay setting module, the SEL351s relay model is applied as a protective device. After modeling the complete system, it is simulated for a particular fault using short circuit and system simulator modules. Moreover, in CAPE, an option exists to get element reports in which the user can get the information about all the settings that are used in the simulation of the power system. Once all the settings are available, putting these settings into actual SEL351s relay to perform the real-time HIL simulation using developed VTB-RT platform is easy. The CAPE screen shots during fault analysis and isolating a fault on the radial power system are shown in Chapter V, Results and Discussion.

3.4 Summary

In this chapter, the model development of an instantaneous overcurrent relay in different software is presented along with the background information on overcurrent relay. Also, this chapter indicates the types and protection scheme of an overcurrent relay. The digital relay requires relay elements such as an analog to digital converter, digital filters, and logic gates, which are explained in details in this chapter. The next chapter describes the implementation of the real-time HIL simulation platform using Linux/Unix version of VTB, public domain software, and standard notebook computer hardware for testing of the developed relay model.
CHAPTER IV

VTB-RT INSTALLATION

This chapter details the implementation of hard real-time hardware-in-the-loop simulation environment using the Linux/Unix version of VTB called VTB-RT. The real-time HIL simulation tool is a very popular tool for verifying engineering designs before hardware construction. It offers many advantages over traditional simulation tools, such as allowing the user to test dangerous situations, to repeat the same test again and again, and to ensure a correct implementation by reducing the gap between design and implementation. Moreover, the user can save time and money by testing design model using real-time HIL simulation software without going through the field testing. Many companies and research institutes are working on the development and application of real-time HIL simulation software, therefore, the idea of implementing HIL platform is not new. The literature is full of examples of real-time HIL applications. However, in most cases, the system is hardware dependent or based on proprietary solutions [27]. This thesis demonstrates the implementation of VTB-RT, a real-time HIL environment that used public domain software and standard notebook hardware.

This chapter is in two parts. The first part describes the structure of VTB-RT. Mainly, VTB-RT requires three public domain software called VTB-RT components, Linux, RTAI,
and Comedi. The first part details the functionality of these components in the implementation of VTB-RT HIL simulation environment. The second part of this chapter focuses on system requirements and package installation requirements for VTB-RT.

4.1 VTB-RT Structure

The structure of VTB-RT consists of three main software components, called VTB-RT components as shown in the Figure 4.1. Before going through the functionality of these components in the VTB-RT implementation, the user needs to have some ideas about the real-time system.

4.1.1 Definitions of Real-Time

As mentioned in the above, the real-time aspect is one of the most attractive features of VTB-RT. To understand the system structure of VTB-RT, some definitions of real-time are necessary. Here are some necessary definitions that may help to understand VTB-RT structure [27]:

1. Real-Time: According to the IEEE definition, a real-time system is a system whose correctness includes its response time as well as its functional correctness. In a real-time system, the management of time constraints are very important. The answers and the time when the answers are produced are important in real-time.

2. Hard Real-Time: Hard Real-Time means that the entire system must be designed which can give the GUARANTEE that the response requirements are met. In a hard real-time system, the deadlines are fixed and the system must respond within a fixed and predefined time. The hard real-time is not based on average response times, because it is used for applications where the time accuracy requirement is close to the microsecond [27].

3. Soft Real-Time: Soft Real-Time is exactly the same as hard real-time in its infrastructure requirements, but it is not necessary for the system success that EVERY
time constraint be met. The soft real-time response is not necessarily exact or precise, but it is based on average time. In a soft real-time system, the system takes an average response time to reach a particular goal.

VTB-RT HIL simulation system requires a hard real-time simulation environment, which must produce the system response within a fixed time interval. This special requirement leads to a Linux/Unix based VTB for real-time simulations.

### 4.1.2 VTB-RT Components

From the software point of view, VTB-RT requires the following three free software packages, Figure 4.1 shows: Linux, RTAI, and Comedi. The following section gives detailed information about these software packages.

![Figure 4.1](image_url)

**Linux OS**

VTB-RT structure [20]
4.1.2.1 Linux

Linux is a stable operating system, developed by Linus Torvalds. It is based on Unix; hence, it is both multi-users and multi-tasking. It has all the features of Unix including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and TCP/IP networking.

Linux is nothing but a kernel of a computer operating system, which provides communications between computer applications and hardware, and system services such as file management, virtual memory, device I/O, and much more. Linux is the framework of VTB-RT. The performance of the VTB-RT directly depends on the performance of Linux. As discussed above, Linux is more suitable for real-time applications over Windows because of its advantages like free, reliable, and stable software. It also provides the full access to the computer resources. Thousands of companies and individuals all around the world are constantly developing Linux. Its functionality, adaptability, and robustness make it the most appealing operating system over Unix and Windows. However, it suffers from a lack of real-time support. There are many distributions of Linux available online, like Mandrake, Fedora, Redhat, Dabian, Suse, Caldera and so on. All Linux distributions are not similar. They are different in their installation, the included software, the kernel version, and their compilation.

VTB-RT requires a clean generic Linux kernel to make it as a real time kernel. The kernel, which comes with the installation of Linux from distribution, is just used as a framework so it does not matter which distribution of Linux is used in VTB-RT. Also, all
Linux distributions come with a specific release that contains the necessary development tool which is required in VTB-RT, like gcc compiler. Free Linux distributions are available on many websites, such as http://www.linuxiso.org.

4.1.2.2 Real-Time Application Interface (RTAI)

RTAI is a kernel modification and enhancement package of Linux that creates a hard real-time system clock. It comes with a set of functions that support real-time simulation. The RTAI is initially developed by The Dipartimento di Ingegneria Aerospaziale - Politecnico di Milano (DIAPM) in 1998. Like Linux, RTAI is open software and can be obtained freely from the web.

RTAI enables VTB to handle real-time applications. It adds the features of an industrial real-time operating system to the standard Linux distribution, which provides the capability of obtaining the correct timing behavior without losing the useful non real-time characteristics of Linux. Moreover, it gives high priority to the real-time tasks and low priority to Linux operations, which are performed during non real-time activity.

The source code of the latest version of RTAI is freely available on the DIAPM website http://www.aero.polimi.it/RTAI/ [9]. Also, other useful information is available on the website about installation, the latest release, and so on. In the mailing archive of RTAI, the user can see the messages of others and ask questions.
4.1.2.3 Comedi

The Comedi project develops free device drivers, tools, and libraries for many data acquisition cards (DAQ cards) from various manufacturers. It is freely available on the web like Linux and RTAI. The user can download the source code from Comedi website: http://www.comedi.org.

The comedi source code is distributed in two packages: “comedi” and “comedilib”. Comedi is a collection of device drivers for a variety of common data acquisition plug-in boards, while comedilib is a user-space library that provides a developer-friendly interface to Comedi devices. Comedi works not only with standard Linux kernels, but also with its real-time extensions such as RTAI.

In short, Linux is a framework for VTB-RT. It provides hard real-time support that VTB-RT needs to become a hard real-time simulation environment. The function of RTAI is to create the real-time tasks that schedule and manage the time constraints for real-time simulations, while the function of Comedi is to provide the communications between the simulation and real-world.

4.2 Software Installation

As mentioned above, VTB-RT basically requires the following software components: Linux, RTAI, and Comedi (including Comedi and Comedilib). Setting up a real-time simulation platform using the Linux version of VTB requires basic knowledge of operating systems, computer architecture, and hardware drivers [27]. Besides the above software
components, VTB-RT requires a solver. The solver is an important part of the VTB-RT. VTB supports many different solvers, but currently only two solvers are available in VTB-RT, transplanted from the Windows version VTB to Linux version VTB. One is Simulated Analog Computer Solver and the other is Signal Extension Resistive Companion Solver. This thesis uses the Signal Extension Resistive Companion Solver, which supports signal coupling as well as natural coupling.

4.2.1 System Requirements

Before installing the above mentioned software packages for VTB-RT, the host PC or notebook computer must meet some hardware requirements. The following two subsections describe the hardware and software requirements for the real-time VTB.

4.2.1.1 Hardware Requirements

Table 4.1 shows the hardware requirements that the host PC or notebook computer should meet for VTB-RT installation. Before selecting an I/O DAQ Card for VTB-RT, the user should check the card in the Comedi supported hardware list. It should be listed in the comedi supported hardware list; otherwise comedi does not support the device. Table 4.2 shows the hardware and software configuration of the VTB-RT platform that the author implemented at MSU.
### Table 4.1

**Hardware Requirement for VTB-RT**

<table>
<thead>
<tr>
<th></th>
<th>Minimum Configuration</th>
<th>Suggested Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>200 MHz</td>
<td>800 MHz</td>
</tr>
<tr>
<td><strong>Hard Drive</strong></td>
<td>2 Gigabytes</td>
<td>10 Gigabytes</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>64 M</td>
<td>128 M</td>
</tr>
<tr>
<td><strong>Video RAM</strong></td>
<td>2 M</td>
<td>8 M</td>
</tr>
<tr>
<td><strong>CD ROM</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Floppy Drive</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Free PCI/PCMCIA Slots</strong></td>
<td>1</td>
<td>At least 1</td>
</tr>
<tr>
<td><strong>I/O DAQ Cards</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Table 4.2

**VTB-RT Platform at MSU**

<table>
<thead>
<tr>
<th></th>
<th>Hardware Configuration</th>
<th>Software Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>Intel Pentium III 1.2 GHz</td>
<td>Linux Distribution</td>
</tr>
<tr>
<td><strong>Hard Drive</strong></td>
<td>20 Gigabytes</td>
<td>Linux kernel</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>256 M</td>
<td>RTAI</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>Integrated 10/100</td>
<td>Comedi</td>
</tr>
<tr>
<td><strong>I/O DAQ</strong></td>
<td>NI DAQCard-6062E</td>
<td>Comedilib</td>
</tr>
<tr>
<td><strong>Free PCMCIA Slots</strong></td>
<td>1</td>
<td>gcc version</td>
</tr>
</tbody>
</table>
4.2.1.2 Software Requirements

The general software requirements for implementing a VTB-RT platform are listed below [12]:

- Clean generic Linux kernel
- RTAI real-time patch for the particular Linux kernel
- Comedi and Comedilib that contain a device driver for DAQ card embedded in the host computer.
- Required versions of some software packages in Linux.

4.2.2 Packages Installation

A clean, generic Linux kernel is highly recommended for successful installation of VTB-RT system [12]. Usually, installing the latest versions of Linux kernel and RTAI is always a good idea because the most recent versions of the software have more advanced functions and also they are easier to debug. Moreover, more people are working on them so a user can get help easily if there is any problem. Mostly, RTAI supports the latest versions Linux kernel, but it is not always true. Before installing the Linux kernel, the user should check the supported kernel versions on the RTAI website. There is a “./rtai/patches/” directory on the RTAI website where the user can find the information about latest RTHAL patch [27].
4.2.2.1 Linux Installation

Linux is a platform for the VTB-RT system. There are many websites where the user can find different versions of Linux distributions freely, like \url{http://www.linuxiso.org}, as shown in Figure 4.2.

![Website of Linux Distributions](image)

**Figure 4.2**

Website of Linux Distributions

To install VTB-RT, the first step is to select the versions of Linux distribution, RTAI, and Linux kernel. These versions must be compatible for successful implementation of VTB-RT platform. Each version of RTAI supports only a limited subset of Linux kernel versions, so choose the Linux kernel version, supported by RTAI. After selecting the Linux
kernel, select the Linux distribution that is compatible with the Linux kernel version. It is also possible to first select the Linux distribution and then select Linux kernel and RTAI. Basically, the selection order does not matter much, but all selected versions must be compatible with each other.

This thesis uses the Fedora Core 3 Linux distribution for the VTB-RT, which uses a 2.6 kernel. Fedora Project is a Red Hat sponsored and community supported open source project [4]. The detailed installation process of Fedora Core 3 is in Appendix A.

4.2.2.2 RTAI Installation

RTAI installation occurs after the successful installation of the Linux operating system. RTAI installation can be divided into two parts: update Linux kernel and setup RTAI [27]. This section gives the detailed RTAI installation steps that are described in RTAI installation guide [34].

Step 1. Downloading the Linux Source:

Go to http://www.kernel.org website and download the source code of Linux kernel compatible with the Linux distribution [5]. Here, the Linux kernel 2.6.8.1 is compatible with Fedora Core 3 kernel. To download the source code of Linux kernel 2.6.8.1, click on linux-2.6.8.1.tar.bz2 or linux-2.6.8.1.tar.gz. The download button will start to download. After completing the download, save the downloaded package. Save the source code into “/home/daxa/kernel/RTAI” directory.

Step 2. Downloading the RTAI Source:

The source code of all RTAI versions are available on its website [9]. To download the source code of the selected version, go to http://www.aero.polimi.it/RTAI. For this research, RTAI-3.1 is compatible with the selected Linux kernel version. The source code of RTAI-3.1 is rtai-3.1.tar.bz2. The tar file is put in the same directory, “home/daxa/kernel/RTAI,” where the Linux kernel source tar file was saved. The location of the source file really does not matter, but keeping them together is recommended.
Step 3. Unpacking the Linux kernel and RTAI source code:

In the Linux terminal, use the following commands to unpack the source files of Linux “/home/daxa/kernel/RTAI” directory.

For Linux:

```
bunzip2 linux-2.6.8.1.tar.bz2
```
```
tar -xf linux-2.6.8.1.tar
```

or

```
gzip linux-2.6.8.1.tar.gz
```
```
tar -xvf linux-2.6.8.1.tar
```

For RTAI:

```
bunzip2 rtai-3.1.tar.bz2 tar -xf rtai-3.1.tar
```

This will unpack the source code of Linux kernel and RTAI into a directory named “linux-2.6.8.1” and a directory named “rtai-3.1”, respectively, in the current location “/home/daxa/kernel/RTAI”.

Step 4. Applying the RTAI patch to the Linux source:
RTAI patches are held in the directory called “rtai-core/arch/i386/patches”. When the user goes to this directory and types “ls” in the command line, it will show all the patches contained in that directory. Choose the patch that is suitable to the Linux version selected. For more explanation about patches, the reader is directed to [34] and [27]. Since the Linux kernel version is 2.6.8.1, here hal7-2.6.8.1.patch is applied.

Under location “/home/daxa/kernel/RTAI”, use the following command to apply a selected patch:

```
patch -p1 < /rtai-3.1/rtai-core/arch/i386/patches/hal7-2.6.8.1.patch
```

**Step 5. Configuring the Linux kernel:**

This step is the most important step in the whole installation process. Controlling the Linux kernel configuration requires a .config file, but the problem is, after the download and unpacking of the kernel source, there is not a .config file in it. There are many options to get it. One option is to use the default file provided with the Linux kernel source. Starting at the Linux source top level directory, the file is:

```
arch/i386/defconfig
```

This file provides a minimal set of definitions that can be used to build a minimal system. This minimal .config provided with the Linux source will build a kernel that will run something, but probably will not support everything needed.

The other option is to use the .config file that is used to build the Linux distribution. Most distribution include this file, particularly if the kernel source is included when the distribution was installed.

After having a .config file, some useful commands exist that can be used to modify it for the system. These commands are:

```
make oldconfig
```

or

```
make xconfig
```

or

```
make menuconfig
```

where,

“make oldconfig” : Default all questions based on the contents of existing

“make xconfig” : X windows based configuration tool

“make menuconfig” : Test based color menus, radio lists and dialogs

According to the author’s experience, after copying the selected configuration file over the top level Linux directory as .config, use “make oldconfig” to bring .config
file up to date. Then use “make xconfig” to tune the configuration. Save from the GUI and quit.

In the xconfig GUI, the following options were selected in addition to the defaults provided in the minimal 2.6.8.1 .config file:

**Processor type and features → Processor family.** Make sure a suitable processor is selected.

**Device drivers → Networking support → Ethernet (10 or 100Mbit).** Click this and it will show all the possible device drivers. Pick the one that matches to your ethernet adapter.

**Device drivers → Block devices → Loopback device support.** Select this option, otherwise kernel will not support the ‘mkinitrd’ step in the kernel installation.

**Loadable module support → Enable loadable module support.** This option should already be selected by default. Also select the Module unloading sub-option, and make sure the Automatic kernel module loading option is selected. Module versioning support should be off, because it causes problems with RTAI.

**Adeos support.** This option only appears after applying the RTAI patch. Just make sure it is selected.

**Kernel hacking.** Make sure that the compile the kernel with frame pointers option is not selected.

**Processor type and features → Preemptible kernel.** Make sure it is selected.

**Device drivers → Character devices → Enhanced Real-Time Clock support.** Make sure it is selected as a module.

**Step 6. Building the Linux kernel:**

This step is really very easy, once a .config file is available through the above step. In this step, the “make” command is used to the main kernel file, which is kept in /boot in a compressed form and is loaded as part of the boot process. This file is called bzImage when it is built. We also need to build various kernel modules. All these steps can be done using following commands:

make dep
make clean
make bzImage
make modules

Note that the ‘make dep’ command is not used under 2.6 Linux version. This command should be used under 2.4 Linux version.
Step 7. Installing the Linux kernel:

There are two main parts to this step. The first step is to copy the relevant files - the bzImage file and the various modules - into /boot and the second part is to set up the bootstrap loader to pick them up. For detailed information about this step, the reader should look at [34].

The following steps need to be performed for the installation of Linux kernel:

1. Change to root
   su
   and enter the root password.

2. Install the kernel modules
   make modules_install
   This command installs the modules into /lib/modules/2.6.8.1-adeos. Note that the name used depends on the version string for the new kernel and this is needed for the next step. In our system, the name is ‘2.6.8.1-adeos’.

3. Create the initrd RAM disk file
   mknitrd /boot/initrd-2.6.8.1-adeos.img 2.6.8.1-adeos
   The second argument, the version string, allows mknitrd to locate the files it needs to build the disk file whose name is given in the first argument.

4. Copy the compressed kernel bzImage file into /boot
   cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.8.1-adeos

5. Copy the new system map file into /boot and set up the correct symbolic link for it
   cp System.map /boot/System.map-2.6.8.1-adeos
   ln -s /boot/System.map-2.6.8.1-adeos /boot/System.map
   The next thing is to fix the configuration file for the boot loader. Most of the 2.6 Linux kernel systems use GRUB, which is much more flexible than the traditional LILO. GRUB is controlled by a configuration file called /boot/grub/menu.lst. The syntax for this file is a little bit unclear, but the existing entries should show what works for our system, and then just cut and paste to add a new entry.
   My menu.lst file looks like this:
   default=1
   timeout=5
   splashimage=(hd0,0)/grub/splash.xpm.gz
   hiddenmenu
   title Fedora Core (2.6.9-1.667)
   root (hd0,0)
Step 8. Configuring RTAI:

To configure the RTAI, we do not need to be in a root. This work can be done as a normal user. RTAI is configured in a similar way as Linux kernel, using a “make xconfig”. The RTAI installation guide recommends building RTAI in a separate build directory, not the actual source directory. Therefore, create a directory called “rtai-build”, set the default to that, and run “make xconfig” in that, using the -f option to make to point it at the makefile that is available in the RTAI source directory.

```
cd /home/daxa/kernel/RTAI
mkdir rtai-build
cd rtai-build
make -f ../rtai-3.1/makefile xconfig
```

The “make xconfig” will open a configuration window in which make changes. For more information on RTAI configuration, see [34]. The only option changed in our configuration was:

**General → Linux source tree.** Set this option to home/daxa/kernel/RTAI/ linux-2.6.8.1. To set a new string, double click on entries like this to bring up a dialogue box. This dialogue box allows you to change a string. After setting a string to new string, save it and then quit.

Step 9. Building RTAI:

To build a RTAI, use the ‘make’ command.

```
make
```

Step 10. Installing RTAI:

This step is also pretty easy. For the installation of RTAI, you need to be in root.

```
su
make install
```

Step 11. Rebooting:

After installing RTAI, we need to reboot our computer to use our new system.
shutdown -r now

And the system will shutdown, and reboot itself. This time, if all the installation steps went well, then GRUB will present an option when it comes up. Select the new ‘adeos’ system when the GRUB options come up to load and try to start up.

**Step 12. Testing:**

RTAI comes with a number of test programs. One of these tests is a latency test. It reschedules a 10KHz rate and measuring the difference for each reschedule between the actual and expected rescheduling times. To run this test, we need to go to the root directory and then use ‘run’ command:

```
su
cd /usr/realtime/testsuite/kern/latency
./run
```

Usually the system will freeze or generate some error if there are mistakes during installation. If a series of integers are displayed at certain frequency as shown in the Figure 4.4, the RTAI is installed correctly.

![Latency Test Outputs](image.png)

**Figure 4.4**

Latency Test Outputs
4.2.2.3 Comedi Installation

Comedi installation is detailed in the two parts. One is Comedi and the other is Comedilib. Since Comedi and Comedilib installation are independent of each other, it does not matter which is installed first.

1. Install Comedi

**Step 1.** Downloading the Comedi Source Code:

Go to [http://www.comedi.org/download/](http://www.comedi.org/download/) and download the source code of the Comedi-0.7.69 (Comedi-0.7.69.tar.gz) [2]. Save the downloaded source code to the directory “usr/src/comedi”.

**Step 2.** Unpack the Source Code:

Use the following commands to unpack the source code:

```
tar -xvzf comedi-0.7.69.tar.gz
```

This will create a directory called “comedi-0.7.69”.

**Step 3.** Go to the Comedi directory:

```
cd comedi-0.7.69
```

**Step 4.** Edit Makefile:

In the “/usr/src/linux”, edit “Makefile”. At the top of the Makefile, give a value “2” for VERSION, “6” for PATCHLEVEL, “8” for SUBLEVEL, and “rthal5” for EXTRAVERSION.

**Step 5.** Comedi Configuration:

Configure comedi using the command.

```
./configure
```

Here, note that in this newer version of comedi, the file checks automatically RTAI, and it generates the kcomedilib library. Also, it will install all the drivers. Use the following command in “/usr/src/comedi” to check if the modules are installed:

```
ls /lib/modules/2.6.8.1-adeos/comedi/
```

**Step 6.** Compile Comedi modules by using command:

```
make
```

**Step 7.** Install the compiled modules into library by using command:

```
make install
```
Step 8. If it is the first time to install Comedi on the system, also run:
make dev
This step creates the following 16 device nodes:
/dev/comedi0
/dev/comedi1
/dev/comedi2
.....
/dev/comedi15
These files are very important in VTB-RT real-time implementation, since these files enable the access to the hardware from a User-Space process.

2. Install Comedilib

Step 1. Downloading the Comedilib Source Code:
Go to http://www.comedi.org/download/ and download the source code of the Comedilib-0.7.22 (Comedilib-0.7.22.tar.gz). Save the downloaded source code to the directory “usr/src/comedi”.

Step 2. Unpack the Source Code:
Use the following commands to unpack the source code:
tar -xvzf comedilib-0.7.22.tar.gz
This will create a directory called “comedilib-0.7.22”.

Step 3. Go to Comedilib directory:
cd comedilib-0.7.22

Step 4. Compile Comedilib using command:
make

Step 5. Install Comedilib files by using command:
made install

Step 6. Post-install configuration:
To use Comedi, the driver module and the core Comedi module must be loaded into the kernel. Our system used a National Instrument DAQ Card 6062E. The driver module for this card is “ni_mio_cs”. Use the following commands to load both modules into the kernel:
su
/sbin/insmod /usr/realtime/modules/rtai_hal.ko
/sbin/insmod /usr/realtime/modules/rtai_lxrt.ko
/sbin/insmod /usr/realtime/modules/rtai_fifos.ko
modprobe comed
modprobe ni_mio_cs
If the module dependencies are set up correctly, then this command will load both modules into the kernel. To check it, use command:
/sbin/lsmod
To use Comedi to drive the hardware, the hardware device has to be mounted to one of the device nodes that has been created in step 8 of comedi installation section.
NI DAQ Card 6062E was mounted to “/dev/comedi0” by using command:
/usr/local/sbin/comedi_config /dev/comedi0 ni_mio_cs
To unmount this device, use command:
/usr/local/sbin/comedi_config -r /dev/comedi0 ni_mio_cs

Step 7. Test hardware with attached Comedilib examples:
There are many examples available in the directory “usr/src/comedi/comedilib/demo” that can be used to test Comedi with the installed hardware. For example, under “usr/src/comedi/comedilib/demo” directory, type the command:
./info
As shown in the following Figure 4.5, it will display the main specifications of the hardware. If the displayed specifications are correct, the installation of Comedi and Comedilib should be correct.

4.2.2.4 VTB-RT Installation

The complete VTB-RT installation consists of two parts: vtbrt clock installation and vtb-rt installation. Normally, the order does not really matter, but VTB-RT was installed first so a directory called “VTB-RT” was ready. Following steps describes the VTB-RT installation procedure in details.

Step 1. Create VTB-RT directory:
Create a directory called VTB-RT in user.
cd /usr
mkdir VTB-RT

Step 2. Save the source code file:
Save the source code file VTB1204.tar.bz2 into a “/usr/VTB-RT” directory.
Figure 4.5

Comedi Test
Step 3. Unpack the source code of VTB-RT:

In the Linux terminal, use the following commands to unpack the source code of VTB-RT “/usr/VTB-RT” directory.

```
bunzip2 VTB1204.tar.bz2
tar -xvf VTB1204.tar
```

Step 4. Go to the location of VTB-RT and then go to the Signal_Resistive_Companion directory:

```
cd Solvers_and_Models/Signal_Resistive_Companion
```

Step 5. Compile the Code:

```
make
```

Step 6. Go to the VTB-RT root and make a soft link to the LinuxRelease directory:

```
cd ../..
ln -s ./Solvers_and_Models/Signal_Resistive_Companion/Solver/RCSignalExSolver/LinuxRelease/./LinuxRelease
```

Step 7. Make the models:

Compile the extra real-time models. Go to “usr/VTB-RT/Solvers_and_Models/Signal_Resistive_Companion/Models”. Create a directory called “RTmodels”. Save the source codes from the “RTmodels” to the Linux machine. Then compile the models in the “RT-models” directory:

```
make
```

Step 8. Save vtb_run script file:

Save the “vtb_run” script file to the “/usr/VTB-RT/LinuxRelease” directory.

Step 9. Save the VTB-RT clock source:

Save the source code of VTB-RT clock into VTB-RT directory.

Step 10. Unpack the VTB-RT clock source:

```
tar -xvf vtbrt_clk.tar
```
This will create a directory called “vtbrcr_clk”.

Step 11. Go to the vtbrcr_clk directory:

```
cd vtbrcr_clk
```

Step 12. Compile the source:

```
make
```
**Step 13.** install loadable kernel module:

```bash
insmod ./vtbrt_clk.ko
```

**Step 14.** Test the vtbrt_clk:

Use the following command to test vtbrt clock into “vtbrt_clk” directory:

```bash
./check 1000
```

This command will show the numbers increasing in a second basis. The 1000 refers to the default vtbrt_clk period (1000us).

**Step 15.** Create a vtspool directory:

Having all the schematics files in one directory is convenient. An alternative is to create a directory “usr/VTBRT/vtspool” where all the .vts files are going to be saved.

**Step 16.** Test VTBRT:

Once all the above steps are completed, we can run the .vts file in the VTB-RT system. To run .vts file, go to the “LinuxRelease” directory.

```bash
cd /usr/VTBRT/LinuxRelease
```

Then use the following command to run the VTB-RT simulation:

```bash
./vtbrt_run nnn /usr/VTBRT/vtspool/myfile
```

where nnn is the time step in microseconds.

Now the installation of the VTB-RT platform is complete.

### 4.3 Summary

This chapter gives detailed information on the internal structure of the VTB-RT system, along with VTB-RT components. Also, it describes hardware and software requirements that should be met for VTB-RT platform. Finally, it discusses detailed installation steps of each VTB-RT component for successful implementation of VTB-RT platform. The next chapter discusses the real-time HIL testing procedure using VTB-RT and some test cases and results.
CHAPTER V

RESULTS AND DISCUSSION

The previous chapters describe the instantaneous overcurrent relay modeling methodology and the real-time HIL simulation platform implementation in detail. This chapter is divided into two parts. The first part describes the test cases that were used in this work to test the developed relay model. The instantaneous overcurrent relay model was tested for different fault scenarios such as single phase to ground fault, line to line fault, and three phase to ground fault in VTB and Matlab/Simulink. Moreover, the power system modeling and simulation with a SEL-351S relay model in CAPE for a single phase to ground fault in phase A is described. The second part of this chapter focuses on the testing of real-time HIL simulation platform, implemented using real-time capability of VTB, public domain software, and a standard notebook computer. Making sure that the software, hardware, and their communications are functioning correctly is very important before testing the developed relay model on VTB-RT; otherwise the real-time HIL simulation will not work. To test the software and hardware parts, the real-time simulation and open-loop simulation of simple circuit were performed. Finally, the real-time HIL simulation of the same circuit was performed.
5.1 Relay model testing

A protection system is a very important part of the power system in order to minimize and isolate the effects of faults and to maintain the continuity of the power flow to the rest of the system. A protection system consists of protection devices, such as relays, fuses, breakers, and sectionalizers that help detect and remove the fault from the power system. The relay is one of the important components of the protection system. During the past few years, protective relays have become very sophisticated due to more powerful digital technology. This requires elaborate simulation tools for modeling, simulation, and testing of these relays. This thesis explores the application of the Virtual Test Bed (VTB) and its real-time extension, VTB-RT, for protective relay modeling, simulation, and testing. An instantaneous overcurrent relay model was developed in the VTB for a transmission line protection application. The same relay model was also built in Matlab/Simulink for validation purposes. Both models were tested for various fault conditions on a radial power system and results were compared. Moreover, a low cost real-time Hardware-In-the-Loop (HIL) simulation platform was implemented for relay model testing purposes using the Linux/Unix version of VTB (VTB-RT), the latest public domain software packages including the Real-time Application interface (RTAI), Comedi, and Comedilib, and standard notebook computer hardware. The applicability of VTB-RT was verified through a real-time simulation, an open-loop simulation, and a HIL simulation of a simple dynamic system using dSPACE as the control hardware and NI DAQCard-6062E as the input/output.
interface. Simulation results are presented showing the effectiveness of the VTB-RT platform for model testing.

This section mainly focuses on the testing of a developed instantaneous overcurrent relay model for a transmission line protection using a radial power system in VTB, Simulink, and CAPE (Computer Aided Protection Engineering).

5.1.1 Testing of VTB Relay Model

Once the relay model was built in VTB as described in Chapter III, Protective Relay Modeling, it had to be tested to determine if it worked as expected. A radial power system was used to test the VTB relay model. First, the power system model was built in VTB using models available in the electrical library such as three-phase source model, three-phase breaker model, cable model, and PQ load model. Using the current sensor models, the currents in all three phases were measured. These measured current signals are high ampere signals, and a digital relay does not allow the high ampere signals. Therefore, using a proper CT ratio, they should be converted to the low level signals that are allowed by relays.

After connecting the developed relay model with the power system model in VTB, a single phase to ground fault was applied in phase A to test the relay model for transmission line protection. Figure 5.1 shows the schematic diagram of the developed instantaneous overcurrent relay model in VTB for a transmission line protection for a single phase to
ground fault in Phase A; Figure 5.2 shows the simulation results from the VTB simulation of this model.

![VTB Relay Model Testing for a Single Phase to Ground Fault](image)

**Figure 5.1**

VTB Relay Model Testing for a Single Phase to Ground Fault

The instantaneous overcurrent relay is similar to a comparator, which outputs 0 or 1 depending upon the relay input signal and pickup value. Comparator output 0 means no fault has occurred in the power system, and output 1 means a fault has occurred in the system. The VTB simulation results in Figure 5.2 show that a fault occurred in phase A because the output of comparator A is varying from 0 to 1, while the outputs of comparators B and C remain at 0.
Figure 5.2

VTB Simulation Results of Single Phase to Ground Fault
The same relay model was also tested for a Line to Line fault as shown in Figure 5.3 for a transmission line protection in VTB.

Figure 5.3

VTB Relay Model Testing for a Line to Line Fault

Figure 5.4 shows the simulation results of a VTB simulation for a line to line fault. The simulation results show that a fault occurred in phase A and B because comparator outputs of phase A and B are changing from 0 to 1 at the instant when the fault occurred in the power system. The comparator output does not change for phase C and it remains at 0.

Figure 5.5 shows the VTB schematic diagram of the relay model testing for a three phase to ground fault for transmission protection, while Figure 5.6 shows the VTB simulation results for this test case.
Figure 5.4

VTB Simulation Results of Line to Line Fault
As mentioned above, if the comparator output changes from 0 to 1, then a fault occurred in a power system. In this case, the output of comparators A, B, and C changed from 0 to 1 because of the three phase fault on the power system.

Figure 5.5

VTB Relay Model Testing for a Three Phase to Ground Fault

5.1.2 Testing of the Simulink Relay Model

In this research, the Simulink tool helped build an instantaneous overcurrent relay model so that the author could compare the simulation results and validate the relay model. In Simulink, the same power system was modeled to test the relay model for various faults.
Figure 5.6

VTB Simulation Results of Three Phase to Ground Fault
As mentioned in the Chapter III, Protective Relay Modeling, Simulink has Simpower System Blockset which enables the user to model the basic components of the power system very easily. Using the Simulink Simpower Systems Blockset, a power system model was developed that was used for VTB relay model testing. After modeling the power system, the single phase to ground fault was applied in phase A. Figure 5.7 shows the Simulink diagram of a relay model testing for a single phase to ground fault for a transmission line protection, while the Simulink simulation results can be seen in Figure 5.8 for this test case.

Figure 5.7
Simulink Relay Model Testing for a Single Phase to Ground Fault
The simulation results show that there was a fault on the power system in phase A because the output of comparator A changed from 0 to 1; however, no fault occurred in phase B and C shown by 0 comparator output. Moreover, it validates the VTB single phase to ground fault simulation results. In the same way, the relay model was tested for a line to line fault as shown in Figure 5.9. The fault was applied from line A to line B. The Simulink simulation results for a line to line fault can be seen in Figure 5.10. The results match with the VTB line to line simulation results.

The developed Simulink relay model was tested for a single phase to ground fault and a line to line fault; simulation results show a good match with the VTB simulation results for the same type of faults. Figure 5.11 shows the schematic diagram of the relay model testing for the most common fault in power system, a three phase to ground fault. The simulation results for this case can be seen in Figure 5.12. The results show the fault in all three phases, which validates the VTB relay model for a three phase to ground fault test case.

5.1.3 CAPE Power System Modeling

As mentioned in Chapter III, CAPE is a protection engineering software package, which contains almost all manufactured relay models. In this research work, the author used CAPE to model the power system used in VTB and Simulink for model testing and to simulate the power system model with the available SEL-351S relay model.
Figure 5.8

Simulink Simulation Results of Single Phase to Ground Fault in Phase A
Figure 5.9

Simulink Relay Model Testing for a Line to Line Fault

The CAPE software contains various modules that help users to model any complex power system very easily. The one-line diagram module is very helpful in building the simulation system. There are all the power system components models available such as cable model, source model, load model, breaker models, which are necessary to build a power system. In this work, first, a radial power system was built using a one-line diagram module, and then a SEL-351S relay model was applied as a protection device. CAPE has a relay setting module; this module helps to set the relay according to the application requirements. After modeling the system, a single phase to ground fault was applied in phase A. Figure 5.13 shows the CAPE screen shot during the fault analysis of a power system.
Figure 5.10
Simulink Simulation Results of Line to Line Fault
After applying a single phase to ground fault on phase A, the system was simulated using system simulation module in CAPE. Also, the CAPE coordination graphic (CG) module generated the relay response plot and report. In the relay report, the user can get all the relay settings information, which in this case is very necessary for the real-time HIL simulation of SEL-351S relay, to validate the developed relay model. Moreover, as mentioned earlier, user can download this settings to the SEL-351S relay to perform the HIL simulation of relay with power system model. Figure 5.14 is a CAPE screen shot of the protection system isolating a single phase to ground fault in phase A for a transmission line protection. In Figure 5.14, the circuit breaker is opened, which means the protection system is isolating the fault from the power system.
Figure 5.12

Simulink Simulation Results of Three Phase to Ground Fault
Figure 5.13
CAPE Screen Shot During Fault Analysis on a Radial Power System

Figure 5.14
CAPE Screen Shot of Protection System Isolating a Fault on a Radial Power System
This section of the chapter described the test cases for testing an instantaneous overcurrent relay model in VTB and Simulink. The developed relay model was tested for a single phase to ground fault, line to line fault, and three phase to ground fault in VTB and Simulink. Also, the comparison between the VTB simulation results and Simulink simulation results was done; results matched well, which validates the developed instantaneous overcurrent relay model. Moreover, this section described the CAPE modeling and simulation. Using the same setting and parameters, the CAPE program simulated the power system. The CAPE simulation output shows the isolation of a fault for a given setting, which also validates the developed relay model.

5.2 VTB-RT Platform Testing

This section describes the testing procedure of the VTB-RT platform before using it for testing a developed relay model. Presently available digital relays are very complex, therefore testing the relay directly in the developed VTB-RT platform is not a good idea. The VTB-RT platform should be tested for a simple test case to get the idea of the platform performance.

As mentioned in Chapter IV, the VTB-RT platform was built using the Linux/Unix version of VTB, public domain software packages, and standard notebook hardware at MSU. Performing a real-time HIL simulation on VTB-RT requires both software and hardware working correctly. Also, it is key to make sure that the communication between hardware and software is working well, otherwise the HIL simulation will fail.
To test the VTB-RT platform that was built at MSU, the author performed several test cases. This section describes some test cases and simulation results in detail.

5.2.1 Testing Procedures

This subsection describes the VTB-RT HIL simulation testing procedures. Basically, the testing procedures can be divided into the following four steps:

**Step 1.** Design the dynamic system in the Windows version of VTB.

**Step 2.** In the Linux version of VTB, VTB-RT, perform the real-time simulation with all components in VTB-RT.

**Step 3.** In VTB-RT, perform the open loop simulation with an input signal source from an actual device such as a function generator.

**Step 4.** Perform the closed-loop (HIL) simulation in VTB-RT with actual hardware.

5.2.1.1 Software Part Testing

To test the software part of the VTB-RT platform, first a dynamic system was designed in the Windows version VTB. After testing the system in the Windows version VTB, it was imported to the VTB-RT. Currently, two solvers are available in VTB-RT, being transferred from the Windows version VTB [27]. One is the Simulated Analog Computer Solver, and the other is the Signal Extension Resistive Companion Solver. The Simulated Analog Computer Solver is a block diagram solver like Matlab/Simulink, while the Signal Extension Resistive Companion Solver is based on numerical methods. This research uses the Signal Extension Resistive Companion Solver because it supports signal coupling as well as natural coupling.
Figure 5.15 shows the VTB schematic diagram of the RL system created in the Windows version VTB. After creating this system, it was saved as a “.vts” file and moved into Linux VTB-RT. This file records all the information about the system, including components values, simulation step size, and simulation methods. In VTB-RT, this information is retrieved, and the same simulation is performed as in the Windows version VTB. However, before importing the system file into VTB-RT, making certain changes in the diagram and system properties is very important, otherwise the real-time simulation will not work. First, change the sensor models such as the current sensor and the voltage sensor from Windows VTB models to RT models, because in RT models, there is an option to enter the DAQ card information that is necessary to run a real-time simulation. Second, RTAI model in the Windows VTB circuit before importing it to VTB-RT should be used. RTAI is a real-time application interface that defines the real-time clock controlling the evolution of the time-step. At each time step, VTB-RT solves the system model and performs necessary operations. Finally, the change in the simulation properties is shown in Figure 5.16. In Figure 5.16, the “Finite Simulation” was chosen because the author wanted the simulation to stop after the time defined in “Simulation Stop Time” field, in this case, 60 seconds. If the “Continuous Simulation” field is checked, then the simulation will run continuously. The real-time simulation of this system on VTB-RT was performed using the NI DAQCard - 6062E. This card is a PCMCIA card, which is designed for use in the notebook computer [7]. This card has 2 analog outputs, 16 analog inputs, 8 digital I/O lines, two 24-bit counters, and analog triggering.
Figure 5.15

RL System in Windows Version VTB

Figure 5.16

Simulation Properties in Windows Version VTB
Figure 5.17 shows the snapshot of the NI DAQCard-6062E for PCMCIA. Before selecting a DAQ card from the VTB-RT, it is very important to check it in the Comedi supported list. If the selected card is not in a Comedi supported hardware list, it will not work for VTB-RT simulation. This list is available in the Comedi website [3].

![NI DAQCard-6062E for PCMCIA](image)

**Figure 5.17**

NI DAQCard-6062E for PCMCIA [7]

Figure 5.18 shows the Windows version VTB simulation result of the RL circuit. The output of the current sensor and output of the voltage sensor are plotted together. The “pink” plot shows the current flowing through the system, while the “green” plot shows the voltage across the voltage source in the system. After running the simulation in Windows VTB, it was imported into VTB-RT; it run the simulation using NI DAQCard-6062E. Figure 5.19 shows the VTB-RT simulation results. Here, channel 1 shows the voltage sensor output, while channel 2 shows the current sensor output. The VTB simulation results and VTB-RT simulation results have a very good match; the good match means the software part of the VTB-RT is working properly.
Figure 5.18

Windows Version VTB Simulation Result of RL Circuit

Figure 5.19

VTB-RT Simulation Result of RL Circuit
The other test case that the author performed in VTB-RT was a RLC circuit with a sinusoidal input. Figure 5.20 shows a second order system, constructed using a RLC series circuit in the windows version VTB. The Windows version VTB simulated the system, and then it was imported to the VTB-RT for real-time simulation. Figure 5.21 displays the simulation results obtained through VTB simulation.

![RLC Circuit Diagram](image)

**Figure 5.20**

System Configuration File Created in Windows Version VTB

This test case used the same Signal Extension Resistive Companion Solver. The output of the solver was sent to the analog output port of the NI DAQCard-6062E card via an “EXT Voltage Sensor” model. An oscilloscope connected with the analog output port of the DAQ card to display the waveform. VTB-RT performed the real-time simulation: Figure 5.22 shows the simulation results obtained from the oscilloscope. In this test case, the results obtained from VTB and VTB-RT simulation match fairly well.
Figure 5.21

RLC Circuit Simulation Result in Windows Version VTB

Figure 5.22

RLC Circuit Simulation Result in VTB-RT
The above test cases clearly show that the software part of the developed VTB-RT platform is working very well. Now an experiment was done to control the voltage across the capacitor in the RLC circuit. The output voltage can be controlled through the hardware-in-the-loop (HIL) simulation in VTB-RT. Before performing the HIL simulation, one must test the hardware part of the VTB-RT platform. Performing the open-loop simulation of the above RLC circuit tests the hardware part.

Figure 5.23 shows the modified version of the same RLC circuit in the Windows version of VTB. The RLC circuit was modified using the inverter models to perform open-loop and HIL simulation.

![Modified RLC Circuit in Windows Version VTB](image-url)
Figure 5.24 illustrates the simulation results obtained using the Windows version of VTB; Figure 5.25 shows the VTB-RT simulation results. The simulation results of this test case matched with the results of original RLC circuit simulation results that means this modified RLC circuit can be used to perform the open-loop simulation and HIL simulation on VTB-RT platform.

![Simulation Result in Windows Version VTB](image)

**Figure 5.24**

Simulation Result in Windows Version VTB

### 5.2.1.2 Hardware Part Testing

The hardware part and the communication between software and hardware parts testing of VTB-RT platform is accomplished by performing a open-loop simulation of the RLC circuit with sinusoidal input from the function generator. Figure 5.26 illustrates the basic system structure of the open-loop system.
To perform the open-loop simulation of the RLC circuit, a “source” model was replaced by “External Interface” model and then the system was imported from Windows VTB to VTB-RT. The same Signal Extension Resistive Companion Solver performed the open-loop simulation in VTB-RT. In the open-loop simulation, the input of the system was a sine wave from a function generator through the analog input port of NI DAQCard-6062E, and the output of the solver went to the oscilloscope through analog output port of the same DAQ card. Figure 5.27 shows the schematic diagram of the open-loop simulation circuit in Windows version VTB. Figure 5.28 shows the simulation result for the RLC open-loop system in VTB-RT, where channel 1 is the input sine wave reference from the function generator, and channel 2 is the system output. It also illustrates that the sys-
tem output corresponds very well with the system input, reference signal, which means SW/HW parts of VTB-RT are working well.

![Figure 5.26](image)

Structure of the VTB-RT Open-loop System [27]

5.2.1.3 VTB-RT Closed-loop (HIL) Simulation

The above two sections clearly show that the software part, hardware part, and communication between them work well, so now the VTB-RT platform can be used to perform the closed-loop (HIL) simulation. In the real-time HIL simulation, the hardware under test directly interacts with software model in computer. Figure 5.29 illustrates the standard structure of the VTB-RT and HIL simulation system.

The structure of the real-time HIL simulation system on VTB-RT can be divided into two main parts by function [27]:

1. Linux PC
2. Control Hardware
Figure 5.27
Open-loop Simulation Circuit in Windows Version VTB

Figure 5.28
Open-loop Simulation Result in VTB-RT
Figure 5.29
Structure of the Real-time HIL Simulation System [27]

Figure 5.30
Schematic Diagram of the Closed-loop Simulation in Windows Version VTB
Normally, a commercial DAQ Card is applied for the input/output interface between the real-time solver and control hardware. As shown in the real-time HIL simulation structure, the control hardware receives two analog signals, one from the reference source and the other from the VTB-RT solver. These signals are converted from analog to digital through the ADC (analog to digital converter). After converting these signals to digital, the output of the controller goes to VTB-RT, which simulates a dynamic system using the VTB-RT solver. The output of the solver then goes to DAC (digital to analog converter), where the signals are converted to analog form. In this way, the real-time HIL simulation is a closed loop process between the software model and the hardware under test.

Figure 5.31
Closed-loop Simulation Result in Windows Version VTB
Figure 5.32

Schematic Diagram of the Real-time HIL Simulation System
Figure 5.30 shows the closed-loop simulation diagram in the Windows version VTB. In this diagram, the PI controller controls the voltage across the capacitor. First, the parameters of the PI controller were found using the Matlab program. After procuring the PI controller parameters, the closed-loop simulation of the above RLC circuit was performed in Windows VTB to make sure that the selection of PI controller’s parameters were right.

![Figure 5.33](image)

The HIL Simulation Setup on VTB-RT Platform at MSU

Figure 5.31 displays the obtained VTB closed-loop simulation results, where sinusoidal input signals and the output voltage are shown in the same graph. The VTB closed-loop simulation results clearly show that the output voltage is following the reference signal which verifies the PI controller. After verifying the PI controller, it was implemented in dSPACE to perform the real-time HIL simulation of the RLC circuit in VTB-RT.
Figure 5.34
Closer Look of the dSPACE Setup for the Real-time HIL Simulation

Figure 5.35
Plant Model for the Real-time HIL Simulation in Windows Version VTB
Figure 5.36 shows the schematic diagram of the real-time HIL simulation system in the VTB-RT platform. As mentioned above, the real-time HIL structure consists of two main parts. In this research, the plant was modeled in the Windows version VTB which is shown in Figure 5.35, while the PI controller was implemented in dSPACE. The controller diagram in dSPACE is shown in Figure 5.36. The actual system setup of the real-time HIL simulation of RLC circuit on VTB-RT platform in the Power Systems lab of the Electrical and Computer Engineering Department at Mississippi State University is shown in Figures 5.33 and 5.34. As shown in the figures, only two computers are needed for HIL simulation. One is a VTB-RT host PC or notebook computer, and the other is for control hardware. Figure 5.33 displays the host VTB-RT Notebook computer and Windows VTB desktop computer; Figure 5.34 shows the dSPACE hardware and PC that downloaded the program for dSPACE.
During the real-time HIL simulation, VTB-RT performs simulation at each simulation time step, which in this case is 1 ms. VTB-RT simulates the plant model for one sample interval and outputs the simulation result to the controller in dSPACE through the analog output port of NI DAQCard-6062E. The controller in dSPACE receives two analog signals, one from the reference and the other from the solver. In this case, the reference is also in dSPACE. The PI controller in dSPACE performs the control function and outputs control signals to control the duty cycle of inverters. The control signals go back to the solver through the analog port of the NI DAQCard-6062E. Figure 5.37 shows the simulation results obtained for the real-time HIL simulation on VTB-RT, where channel 1 is a sinusoidal reference signal and channel 2 is the voltage output. The results obtained here match as the one obtained in the closed-loop simulation in Windows version VTB.

![The Real-time HIL simulation Result in VTB-RT](image)
The real-time simulation, open-loop simulation, and the HIL simulation results on VTB-RT show that this hard real-time simulation platform, VTB-RT, can be used to test and validate the developed instantaneous overcurrent relay model.

5.3 Summary

This chapter presented test cases for testing relay model and VTB-RT platform, along with detailed discussion of the simulation results. The developed relay model in VTB and Simulink was tested for single phase to ground, line to line, and three phase to ground faults for protection on a radial power system. The simulation results for various test cases showed very good matches for both software. Also, using CAPE for modeling and simulation for a single phase to ground fault was described, which helped to verify the relay model, and procure the relay setting for the real-time HIL simulation. Moreover, this chapter illustrates the VTB-RT platform testing procedure and test cases to check the performance of the VTB-RT platform. To test the platform, the real-time simulation, open-loop simulation, and the real-time HIL simulation of a simple circuit were performed on VTB-RT. Finally, simulation result comparison were discussed. The next chapter outlines the conclusions and future work of this research.
CHAPTER VI

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

The main goals of this research were to develop a protection device model in VTB and to test the developed device model using the real-time capability of VTB. In this thesis, an instantaneous overcurrent relay was chosen for model development because of two main reasons. The instantaneous overcurrent protection is considered as the simplest protection scheme in protection engineering. Additionally it is a common scheme in relaying applications. The relay modeling in VTB was done because no protection device models are available in VTB, and to perform a complete simulation of any power system model in VTB requires protection device models in the software. Moreover, having a protection device model in VTB gives user flexibility to use as many relay models as required to test the power systems design. Additionally, the user can perform the same simulations again and again using the relay model. Therefore, an instantaneous overcurrent relay model was developed in VTB for a transmission line protection. The same relay model was developed for testing purposes in a very popular simulation software, Simulink, with industry and research institutes. The relay models in both software packages were tested for single phase to ground, line to line, and three phase to ground faults in a radial power system. The
comparison between simulation results for various faults was done in both software. The simulation results showed very good matches which verify the instantaneous overcurrent relay modeling.

As mentioned earlier, the Hardware-In-the-Loop testing is a very popular testing and validating methods for engineering designs, prior to the real hardware construction. It also helps to reduce the risks of discovering an error in the very last stage of the field testing and assembling. Therefore, this work presented the implementation of the VTB-RT, an environment designed for the hard real-time HIL simulation. The beauty of the VTB-RT is the integration of Linux/Unix version of VTB, various public domain software packages, and standard computer hardware. In this research, the implementation of VTB-RT platform was done using the latest public domain software packages such as Fedora Core 3, RTAI-3.1, Comedi-0.7.69, and Comedilib-0.7.22, and Dell Latitude C400 Notebook computer hardware. The implemented VTB-RT platform was tested for various test cases to check their performance prior to its use for relay model testing. A simple dynamic system test case was presented in this thesis, which was simulated using different simulation methods. The Windows version VTB, VTB-RT, open-loop, and VTB-RT HIL simulation of this system were performed to check the software part, hardware part, and communication between SW/HW. The simulation results were compared and analyzed, proving that the VTB-RT is a promising environment for the hard real-time HIL simulation.

In addition to relay modeling and VTB-RT implementation, the CAPE power system modeling and simulation with SEL-351S overcurrent relay model was done to validate
the relay model. CAPE modeling also helps to get the relay settings for the real-time HIL simulation of the radial power system model with actual SEL-351S relay to test and validate the instantaneous overcurrent relay model.

6.2 Future Work

At present, the instantaneous overcurrent relay model was created in VTB and Simulink. The VTB relay model was validated with the Simulink relay model and CAPE modeling. Also, the VTB-RT platform was implemented at MSU and was tested for a very simple, series RLC circuit. Future work includes the testing and validating the relay model using the real-time HIL simulation on the developed VTB-RT platform because only software simulation cannot replicate the actual operational conditions. Therefore, the real-time HIL simulation of SEL-351S overcurrent relay will be performed to test and validate the created relay model. In this work, the VTB relay model was created using “Simulink11” VTB wrapper model to realize the digital filtering and phasor calculation parts. VTB wrapper will allow users to use the Simulink models that is currently not available in VTB, but it effects the simulation speed. Therefore, the future work will include to develop custom dll models for digital filtering and phasor calculation parts of an instantaneous overcurrent relay model. Moreover, in this work, the relay model was tested on a small radial power system for a transmission line protection. In future, the simulation of larger system will be performed with multiple relay models to investigate the fidelity of models and analysis time for realistic systems.
REFERENCES

[8] “Overcurrent Protection Handout,” Protection Laboratory, Mississippi State University, Mississippi State, MS.


APPENDIX A

INSTALLATION OF FEDORA CORE 3
1. Download and burn the five Fedora Core 3 CDs from the iso images from any Linux website.

2. Partition the hard drive using any commercial product such as PartitionMagic. You may leave the partition that you wish to use for Fedora as unformatted space. The installation program will format this partition, create swap space within it, and make a directory structure.

3. Configure your bios settings to boot first from the CD drive.

4. Insert the first Fedora Core 3 CD and reboot your computer.

5. At the boot prompt, hit enter.

6. Hit enter for “ok” to test your CD media or right arrow key and enter to skip this test.

7. Click “next” at the “welcome” page.

8. Click “next” at the “language selection” page for default English.

9. Click “next” at the “keyboard configuration” page for default U.S. English.

10. Select “custom” on the “installation type” page. Click “next”.

11. Select “automatically partition” on the “disk partitioning setup” page. Click “next”.

12. Select “keep all partitions and use existing free space” on the “automatic partitioning” page. Click “next”.

13. Click “next” on the “disk setup” page.

14. Configure the “boot loader configuration” according to your requirements. And then click “ok”.

15. Click the “default” box next to “Fedora Core” to make it your default boot operating system. Click “next”.

16. Leave “eth0” and hostname “automatically via DHCP” on the “network configuration” page. Click “next”.

17. Leave “enable firewall” selected on the “firewall configuration” page and click “next”.

18. Select appropriate options on “additional language support” page and “time zone selection” page and after that click “next”.

19. Set your preferred root password on the “set root password” page. Click “next”. You will see a message “reading package information”.
20. Scroll down the “package group selection” page and click in the “everything” box under “miscellaneous”. Click “next”. You will see a message “checking dependencies...”.

21. Click “next” on “about to install” page.

22. Click “continue” to get to the “installing packages” page. You will eventually be prompted to insert the remainder of the installation CDs.

23. When the installation is complete, remove the last CD and click “reboot” for the first boot screen.
Fault at bus 999001

Fault at temporary bus 999001

Center bus 2 Bus 2 Substation Bus2

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

Fault 1 of 1:

Midline node on “1 Bus 1” to “2 Bus 2” Ckt 1

“999001 Bus 1” (NEWBUS1) distance 0.500 from “1 Bus 1”

SLG_Z at midline node “999001 Bus 1” (NEWBUS1)

Simulation Area Definition

Center(Line “2 Bus 2” to “1 Bus 1” Ckt 1

Simulation_Depth 1 Mutual_Depth 0

Skip_Over_XFMR_Sim OFF Skip_Over_Bus_Tie_Sim ON

CTI Definition TB_LZOP - TP_LZOP Desired CTI 0.3 seconds

Pilots ON (pilot flags ignored) Show_Infinite_Times OFF (omit idle LZOPS and elements)

Element_Code AUX DIR DIST IOC TIMER TOC VOLT

LZOP_Type LINE MISC

AND_Limit_Relays_By X

AND_Limit_Elements_By X
AND_Limit_Fuses_By X

AND_Limit_Reclosers_By X

Select_LZOPs_Where X

Exclude Elements ON: Read only elements with contact logic & their supervisors

Check_By Simulation: Open breakers in successive steps

Simulation statistics:

line lzop 1 relays; 0 fuses; 0 reclosers

1 distinct lzops

1 relay/fuse/recloser elements used; 5000 allowed

Simulation area ready

***Starting event # 1

Fault 1 of 1:

Midline node on “1 Bus 1” to “2 bus 2” Ckt 1

“999001 Bus 1” (NEWBUS1) distant 0.500 from “1 Bus 1”

SLG_Z at midline node “999001 Bus 1” (NEWBUS1)

Simulation to Breaker operation # 1

After event 1 0.2 c 0.003 s

Fastest Primary:

Primary LZOP: 1 Generator,L_1 at Generator; 3-pole
LZOP 0.003; Breaker 0.000; LZOP+Bkr 0.003 sec

Trip path GR IOC New Device 1 IOC 50G1

0.003 sec from start

Backup LZOP: None available

Desired CTI: 0.300 seconds Available CTI: infinite (no backup) CTI Defn:

TB_LZOP - TP_LZOP

LZOP Summary Report

LZOP Operating Times (s) predicted at 0.003 seconds from start:

<table>
<thead>
<tr>
<th>Substation ID</th>
<th>LZOP Name</th>
<th>Type</th>
<th>P/B</th>
<th>Trip Path</th>
<th>LZOP Breaker</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generator 1</td>
<td>Generator, L_1</td>
<td>LINE</td>
<td>Primary</td>
<td>GR IOC</td>
<td>0.003</td>
<td>0.000 0.003 Asserts in event</td>
</tr>
</tbody>
</table>

Logical breakers for all LZOPS asserted in this step:

<table>
<thead>
<tr>
<th>LZOP</th>
<th>Breaker type &amp; location</th>
<th>Bkr op cyc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tripped by

________________________________________
1 LINE PRIMARY LINE Branch “1 Bus 1” to “2 Bus 2” Ckt 1 Zero

Opened 3-pole NEW

Network changes now in effect:

Midline node on “1 Bus 1” to “2 Bus 2” Ckt 1

“999001 Bus 1” (NEWBUS1) distant 0.500 from “1 Bus 1”

Open breaker on “1 Bus 1” to “2 Bus 2” Ckt 1 at “1 Bus 1”; New bus “999002 Bus 1” (NEWBUS2)

SLG.Z at midline node “999001 Bus 1” (NEWBUS1)

Fault is cleared after 0.2 cycles 0.003 seconds

****************************************************
***** Relay Element Report for 1 IOC “50G1” *****
****************************************************

Name New Device SCHWEITZER Type SEL-351 Model SEL-351

Style SEL-351S_5A

Substation Generator Branch Main CT: 1-2 Ckt 1 (500.0 kV) CT quantity

NEUT

Rated current 5.00 A

LZOP 1 LINE Generator,L,1

Contact logic code: GR_IOC

(Effective +seq CTR)/(CT turns ratio):

1.0 for YY; 1/sqrt(3) for YD CT connection
CT Connection Primary Base kV Effective Secondary Amps & polarity branch +seq CTR Phase A Neutral (3IO)
MainBr +YY + 3-phase 1 2 1 500.0@ 0 120.00@ 0 48.78@ -69 48.78@

Data base Present
CTR 120.0 120.0
Pickup tap (Secn. A.) 0.25 0.25
Timer 67G1D (sec) 0.0 0.025 Modified

Substation: Generator

LZOP: “Generator, L_1” (LINE)

New Device Tag:
Model SEL-351
Style SEL-351S_5A

1. Inst. OC 50G1

Branch Main CT: 1-2 Ckt 1(500.0 kV) to 2 Bus 2 (Bus2)

CT Ratio: 120.00
Pickup 0.25 R. Amps
Total Oper. Time 1.74 cycles

Fault: A

SLG_Z at midline node 999001 Bus 1 (NEWBUS1)

Midline node on “1 Bus 1” to “2 Bus 2” Ckt 1
“999001 Bus 1” (NEWBUS1) distant 0.500 from “1 Bus 1”

<table>
<thead>
<tr>
<th>Curve</th>
<th>Current</th>
<th>Operating</th>
<th>Source/Total line (+ seq SIR)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>A/Pickup</td>
<td>Cycles</td>
</tr>
<tr>
<td>Primary</td>
<td>5853.52</td>
<td>195.12</td>
<td>1.70</td>
</tr>
</tbody>
</table>