5

Real-Time Rapid Embedded Power System Control Prototyping Simulation Test-Bed Using LabVIEW and RTDS

Karen Butler-Purry and Hung-Ming Chou
Texas A&M University
United States of America

1. Introduction

When developing new control, protection, and stability methods for power systems, it is important to study the complex interactions of the system dynamics with the real-time operation of the new methods. Historically hardware prototypes were developed to study these interactions. With the recent introduction of the much cheaper rapid hardware and software prototyping tools, developers are choosing instead to use these tools to study dynamic interactions during real-time operation. This technology provides a cost effective option and flexibility in modeling and coding which allows use for versatile applications. Rapid prototyping technology is being widely used in many application areas for real-time data analysis and control such as avionics, power, acoustics, mechatronics, and automotive applications (Keunsoo et al., 2005; Postolache et al., 2006; Spinozzi, 2006; Toscher et al., 2006). Parallel digital signal processors (DSPs) (French et al., 1998; Lavoie et al., 1995) are also being used as rapid prototyping technology for real-time simulation and control.

To study real-time dynamic interactions of isolated power systems and control methods, a test bed is developed that uses rapid prototyping technology. This chapter discusses the real-time test bed which was developed more generally to study real-time issues, and validate and verify new designs of centralized and de-centralized control, protection, and wide-area monitoring methodologies for stand alone power systems, such as naval shipboard power systems, power transmission and distribution system and microgrids.

The National Instruments Compact RIO (*NI CompactRIO*, 2011)low cost reconfigurable control and acquisition system is used to implement the new control and protection methods in the test bed. The NI CompactRIO embedded system contains the NI reconfigurable technology and a real-time controller with an embedded (Pentium) floating-point processor. The 8-slot reconfigurable chassis contains an embedded user-programmable FPGA (field programmable gate array) chip providing direct access to hot swappable input/output (I/O) modules for precise control and flexibility in timing, triggering, and synchronization. The I/O modules contain built-in signal conditioning and isolation. The control and protection methods are implemented using the LabVIEW RT and FPGA Modules on a Host PC. The code is downloaded to the cRIO controller and FPGA for execution. LabVIEW Virtual Instruments (VIs) are used to remotely monitor and control (if desired) the RT Power System and Controller simulation. National Instruments data acquisition cards and hot swappable input and output analog and digital modules are used to pass signals between the cRIO controller

and the real time power system simulation. The inputs and outputs of the NI I/O modules were mapped in VIs. Further data acquisition settings for the I/O modules such as the number of channels, data type, sampling rate, and data buffer length were programmed in VIs.

The power systems are modeled using RSCAD software for a real time digital simulator (RTDS®) (*Real Time Digital Simlator - RTDS*, 2011) which is a unique integration of hardware and software specifically designed to perform electromagnetic transient simulation (EMTP) of AC and DC power systems with external hardware in the simulation loop. Based on its unique architecture, it is equally suitable for simulating large transmission and distribution systems or tightly coupled power systems such as ship systems, locomotives, airplanes, automobiles, and micro grids. The RTDS simulator architecture exploits the parallel computation nature of the mathematical model of large power systems using proprietary parallel processing hardware. The RSCAD run-time interface of the RTDS is interactive and is used to view the simulation results and provide user inputs while the simulation is in progress. RTDS is a modular system consisting of racks that are paralleled. Each rack consists of several processing cards which can be customized to provide the computational power of a specific power system problem. These racks can be used to run simultaneous simulations of multiple smaller systems or large simulations requiring multiple racks. RTDS provides high precision (16 bit) real-time input/output (I/O) interfaces of both analog and digital signals to interface with the external instrumentation in the simulation loop.

The implementation of an overcurrent relay protection scheme is discussed in this chapter to illustrate the operation of the test bed. A two-bus power system which has a generator at one bus, a load at the other bus, and a transmission line between them, is simulated on the RTDS. The overcurrent relay is implemented on the RT processor and FPGA. Measurements of the current flowing through the cable and the node voltages are measured. Further the results of several case studies are discussed.

This book chapter will focus on the concept of real-time rapid embedded control prototyping simulation and its implementation using LabVIEW. Section two will describe RTDS and cRIO as well as the proposed methodology for implementation of embedded controller-in-the-loop simulation. Section three will present the detailed implementation of the overcurrent relay case study, including the programs in RTDS and in cRIO. Also, a synchronization technique to synchronize the RTDS with the cRIO in real-time operation will be discussed. In section four, simulation results will be described. Lastly, the conclusions and some future work will be presented.

2. Real-Time Rapid Embedded Power System Control Prototyping Simulation Test Bed

There are two main components in the Real-Time Embedded Power System Control Rapid Prototyping Simulation Test Bed. One component uses the National Instruments CompactRIO (cRIO) Embedded Design Platform that allows users to rapidly build embedded control or acquisition systems. The other component uses the RTDS real-time digital simulator which allows real-time simulation of power systems and hardware-in-the-loop (HIL) studies with controllers, protective relays, and power system components. Figure 1 shows a high-level system diagram for the Real-Time Embedded Rapid Prototyping Simulation Test Bed which is used to test new embedded power system control and protection methods. Test power systems are implemented in the RTDS and new controller designs are implemented in the cRIO.

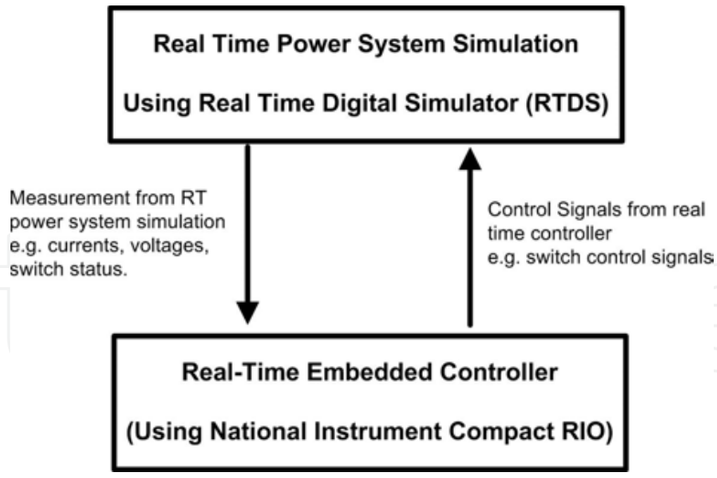


Fig. 1. System Diagram for the Real-Time Rapid Power System Control Prototyping Simulation Test Bed

The test bed is designed based on hardware-in-the-loop (HIL) simulation theory. Hardware-in-the-loop simulation is an approach that is used during the development and testing of complex real-time embedded controller. HIL simulation provides an effective platform by adding the complexity of the plant under control to the test platform. The embedded controller under test interacts with a plant simulation that is a mathematical representation of all related dynamic systems of the plant. HIL simulation includes electrical emulation of sensors and actuators which act as the interface between the plant simulation and the embedded controller under test. The value of each electrically emulated sensor is controlled by the plant simulation and is read by the embedded controller under test. Likewise, this embedded controller outputs actuator control signals based on its control algorithm. Changes in the control signals result in changes to variable values in the plant simulation.
There are several advantages of HIL (Isermann et al., 1999), including

- Designing and testing of the control hardware and software without the need to operate a real process.
- Testing of the effects of faults and failures of system.
- Operating and testing of extreme and dangerous operating conditions.
- Reproducible experiments, frequently repeatable.
- Saving of cost and development time.

For example, HIL simulation can be utilized during the validation of flight controllers (Karpenko & Sepehri, 2006). During the development of flight controllers, their functionality must be tested. The most realistic way is to put the flight controllers in an actual aircraft. In this setting, due to safety and cost issues, the extremely dangerous scenarios cannot be performed, such as engine failure and stalling. However, in HIL simulation, the actual aircraft can be modeled in a real-time simulator and the flight controllers can directly interact with the simulation in real-time. During HIL simulation, all possible scenarios can be performed to test the functionality of the flight controllers without worrying about safety and cost issues. Therefore, a primary benefit of HIL simulation is that by the time the controllers first operate

in the actual system, they will already have undergone a great deal of thorough testing in a realistic simulation environment (Ledin, 2001).
In the Real-Time Embedded Power System Control Rapid Prototyping Simulation Test Bed, a power system is simulated in real-time in RTDS while a controller under test is implemented in NI cRIO. Figure 2 shows the schematic of the major components in the test bed. The host machine has two functions. One is to allow users to write programs for RTDS with RSCAD and programs for cRIO with LabVIEW. The other function is to provide user interface and control of the program execution for RTDS and cRIO.

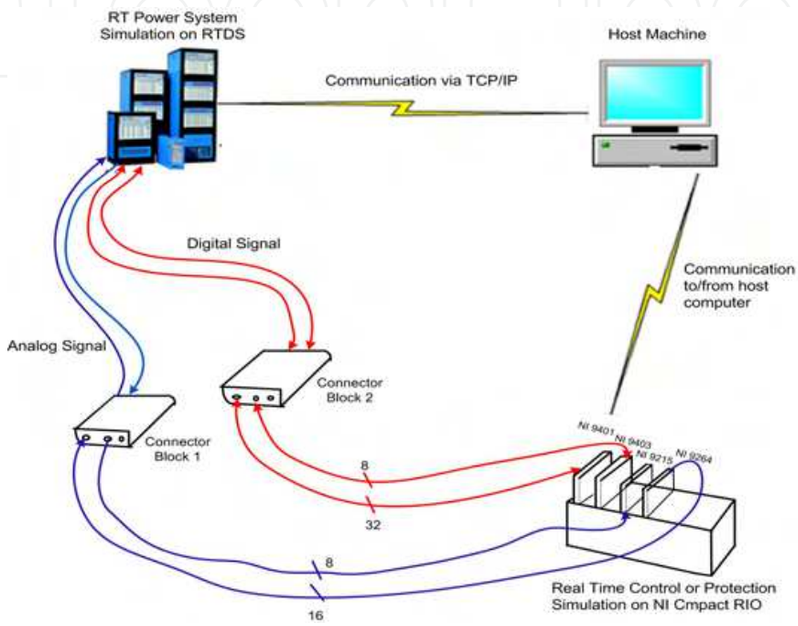


Fig. 2. Schematic of real-time rapid embedded control prototyping simulation test bed

2.1 Real time digital simulator for power system real-time simulation

Real time digital simulator (RTDS), shown in Figure 2, is used to model and simulate power systems in real time. RTDS (*Real Time Digital Simlator - RTDS*, 2011) is a unique integration of hardware and software specifically designed to perform electromagnetic transient simulation of AC and DC power systems. RTDS is a modular system consisting of racks that are parallel. Also it provides high precision real-time input/output interfaces of both analog and digital signals. Detailed information of the input/output module and the processor of RTDS are summarized in Table 1.
The RSCAD software package provides a graphical user interface to write and execute programs in RTDS. RSCAD provides numerous libraries of power system component models as well as control and protection models. RSCAD also provides an interactive run-time interface and allows users to view simulation results and enter or adjust user inputs while the simulation is in progress.

2.2 CompactRIO for rapid control prototyping

The National Instruments CompactRIO (cRIO) low cost reconfigurable control and acquisition system, shown in Figure 2, is an embedded hardware controller, which can be programmed to implement various control and protection methodologies for different applications (*NI*

Component	Detail
GPC $\times$ 3	GIGA Processor Card (Solve up to 54 nodes and 56 breakers)
GTDI	Digital Input Module (64 Channels, Sampling rate: 300 ns)
GTDO	Digital Output Module (64 Channels, Updated every time step)
GTAI $\times$2	Analog Input Module (12 Channels, Sampling rate: 6 μs)
GTAO $\times$2	Analog Output Module (12 Channels, Sampling rate: 1 μs)
WIF	Workstation Network Card

Table 1. Detailed RTDS hardware information

CompactRIO, 2011). The cRIO System contains four components: Reconfigurable chassis housing the user programmable FPGA; Hot swappable I/O modules; Real-time controller for communication and processing; and Graphical LabVIEW Software for rapid Real Time and FPGA programming.
The cRIO contains reconfigurable technologies (FPGA) and a real-time controller with an embedded floating-point processor (RT processor). The 8-slot reconfigurable chassis contains an embedded user-programmable FPGA (Field Programmable Gate Array) chip providing direct access to hot swappable I/O modules for precise control and flexibility in timing, triggering, and synchronization. In addition to I/O functionalities, FPGA has the computing power which performs integer math operation. Table 2 summarizes the detailed hardware information for the cRIO used in the test bed.
By using LabVIEW RT module and FPGA module, the user can write programs in the LabVIEW programming language on a host PC and then download these programs to the RT processor and the FPGA for execution. In this way, the cRIO can implement various types of new control methodologies with great flexibility.

Component	Detail
NI cRIO 9004	Real-Time Controller (195 MHz Pentium processor, 512 MB CompactFlash, 64 MB DRAM)
NI cRIO-9104	8-Slot, 3M Gate (FPGA) CompactRIO Reconfigurable Embedded Chassis
NI 9215	8 Channel Analog Input Module
NI 9264	16 Channel Analog Output Module
NI 9401	8 Channel Digital Input/ Output Module
NI 9403	32 Channel Digital Input/ Output Module

Table 2. Detailed cRIO hardware information

2.3 Methodology for implementation of embedded controller-in-the-loop simulation

A methodology for implementing controller-in-the-loop (CIL) simulation in the test bed was developed with the major aspects adapted from chapters 3 and 4 in Ledin's book (Ledin, 2001). Chapter 3 focuses on non-real-time simulation of dynamic systems, while Chapter 4 focuses on issues related to real-time hardware-in-the-loop simulation. The CIL simulation methodology has four phases: off-line simulation of full system, real-time implementation of

full system, ride-alone (open-loop) simulation, and controller-in-the-loop simulation. Figure 3 shows a flowchart that conveys the steps for the four phases.

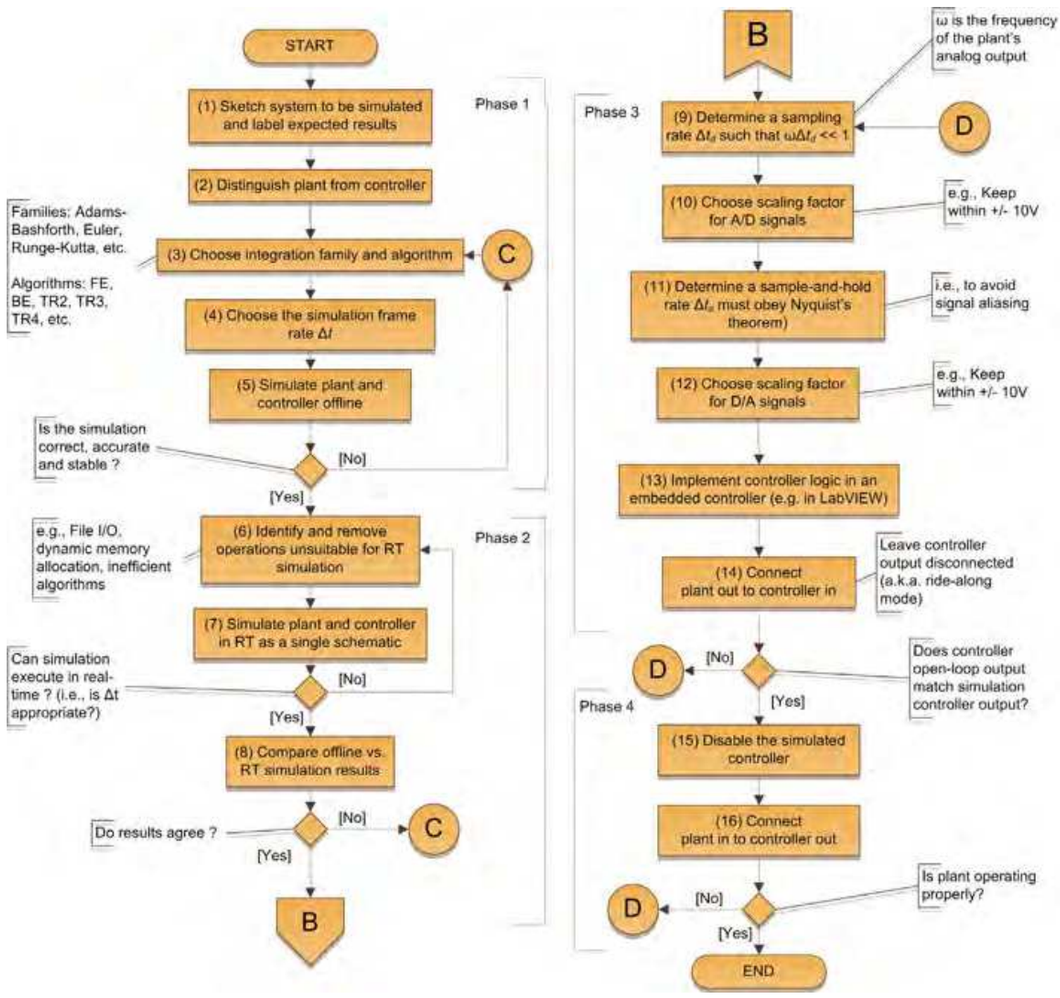


Fig. 3. Procedure of implementing embedded controller-in-the-loop simulation

2.3.1 Phase 1: Offline simulation of full system

In this phase, both the plant and the controller are simulated in a single schematic offline in non-real time by using commercially available software, such as Matlab or PSCAD. An appropriate integration algorithm and simulation step time should be chosen appropriately based on system dynamics. If the simulation results of Phase 1 are correct, the user proceeds to Phase 2.

2.3.2 Phase 2: Real-time implementation of full system

In this phase, both the plant and the controller are simulated in a single schematic in real time using the real-time digital simulation(RTDS) systems .

To convert a simulation from non-real time into real time, operations that are not appropriate for real-time simulation are identified. Some examples are reading or writing disk files, dynamic memory allocation, and inappropriate algorithms with indeterminate execution

time. In addition, the appropriate integration step time based on the system dynamics must be selected. If the time step is too short, the simulation will require more computing power to run in real-time. If the time step is too long, based on Nyquist's theorem, the simulation results will be inaccurate.
The results of Phase 2 and Phase 1 are compared. If the difference is acceptable, the user continues to Phase 3. If the difference is unacceptable, go back to Phase 1 to identify the source of the errors and revise accordingly.

2.3.3 Phase 3: Ride-along mode simulation

In this phase, the actual embedded controller, which was simulated in RTDS in phase 2, is connected to the real-time simulator that models the power system (plant). Input and output interface between the real-time simulator and the embedded controller is set up, including scaling factor, sampling rate, sample-and-hold rate, etc. The control signals from this embedded controller are inputs to the real-time simulator.
These control signals from the embedded controller are not input into the system modelled in the real-time simulator. Instead, the control signals from the simulated controller inside the real-time simulator are input to the modelled system. The purpose of this stage is to compare the control signals from the simulated controller in real-time simulator and that from the embedded controller.
Since the simulated controller will not perfectly represent the actual embedded controller; therefore, in the ride-along mode, we may observe that the response of the actual embedded controller is not actually close to the response of the simulated controller (Ledin, 2001). However, ride-along mode simulation should still give a very good idea of whether or not the controller is operating correctly in this environment. If the difference between these two control signals is acceptable for the application, go to the next and final phase, CIL mode simulation.

2.3.4 Phase 4: CIL mode simulation

With the successful completion of ride-along mode simulation, CIL mode simulation can be performed by using the control signal from the actual embedded controller as input to the real-time simulation. Often the closed loop simulation will work correctly on the first attempt.

3. Case study implementation

A digital protective relay is a protective relay that uses a microprocessor to analyze power system voltages and currents for the purpose of detection of faults in an electric power system. An overcurrent relay discriminates between normal operation and fault operation in a power system. When the current exceeds a specific pickup value, the relay will open the breaker to clear the fault (J. Duncan Glover, 2007). Some of the functions of an overcurrent digital protective relay that are implemented in this case study are listed below.

- The relay applies A/D (analog-to-digital) conversion processes to the incoming currents.
- The relay analyzes the A/D converter output to extract, at a minimum, magnitude of the incoming quantity, commonly using Fourier transform concepts (RMS and some form of averaging are used in basic overcurrent relays).
- The relay applies advanced logic in determining whether the relay should trip or restrain from tripping based on current magnitude, parameters set by the user, relay contact inputs, and the timing and order of event sequences.

A simple two-bus, three-phase power system, whose one-line diagram and detailed information are shown in Figure 4 and Table 3, is implemented in the RTDS. A microprocessor-based overcurrent relay controls the breakers: BRK_A, BRK_B and BRK_C. This overcurrent relay has three current transformers (CT) that measure line current in each of the three phases: $i_a(t)$, $i_b(t)$ and $i_c(t)$. Based on these measurements, the relay logic in the over-current relay will determine the correct value for the control signals that control the breakers. If the RMS value of the line current is higher than the pickup value, the relay logic will open the breaker in the corresponding phases to protect the power system. In this case study, the breaker and CTs are implemented in the RTDS and the relay logic is implemented in the cRIO.

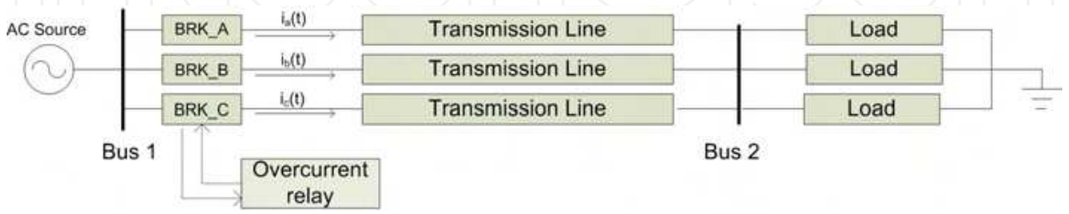


Fig. 4. One line diagram of two-bus, three-phase with overcurrent relay protection

Component	Detail
AC Source	3 Phase, 60 Hz, 2.45 kV (Hz: Hertz, V: Volt)
Transmission Line	66.6 mΩ + 5.3mH (Ω: Ohm, H: Henry)
Load Impedance	21.26mΩ + 10.1mH (Ω: Ohm, H: Henry)
Closed Resistance of Breaker	0.001 Ω
Pickup Value of Overucurrent Relay	0.1 kA (RMS) (A: Amp, RMS: Root Mean Square)

Table 3. Parameters of the simple power system

Figure 5 shows the functional blocks of RTDS and cRIO in CIL simulation. In the RTDS, the power system is modeled and simulated in real time. The RTDS sends to the cRIO three phase current measurements, $i_a(t)$, $i_b(t)$ and $i_c(t)$ via the RTDS analog output (AO) module and sends the triggering signal via the RTDS digital output (DO) module. Based on the current measurements, the RMS calculation block in the FPGA of the cRIO calculates the RMS values of these three current measurements. Based on these RMS values, the relay logic implemented in the RT processor in the cRIO determines the value of the breaker signals. Then the cRIO sends these three breaker signals (BRK_cRIO_A,BRK_cRIO_B,BRK_cRIO_C) back to RTDS via the cRIO digital output (DO) module to RTDS digital input (DI) module to control these three breakers of the simple power system simulation in RTDS.
The program in the FPGA runs at 50 us loop rate while that in the RT processor runs at 5 ms loop rate. The procedure for selecting the loop rates for the FPGA and RT processor will be discussed later in this chapter. Since the loop rates for the FPGA and the RT processor are different, a data buffer: DMA-FIFO is used to transfer data from the FPGA to the RT processor without losing data.

To have a higher chance of having successful CIL simulation, Ledin (Ledin, 2001) recommends that before implementing HIL simulation, it is better to implement ride-along mode simulation. To perform ride-along mode simulation, it requires implementation of identical RMS calculation blocks and over-current relay logic blocks in RTDS and cRIO. In this way, we can compare the breaker signals from RTDS (BRK_Rtds_A, BRK_Rtds_B, BRK_Rtds_C) and that from cRIO (BRK_cRIO_A,BRK_cRIO_B,BRK_cRIO_C). In the ride-along mode, the goal is to determine if the embedded controller simulation is accurate. Figure 6 shows the functional block of RTDS and cRIO in the ride-along mode simulation.

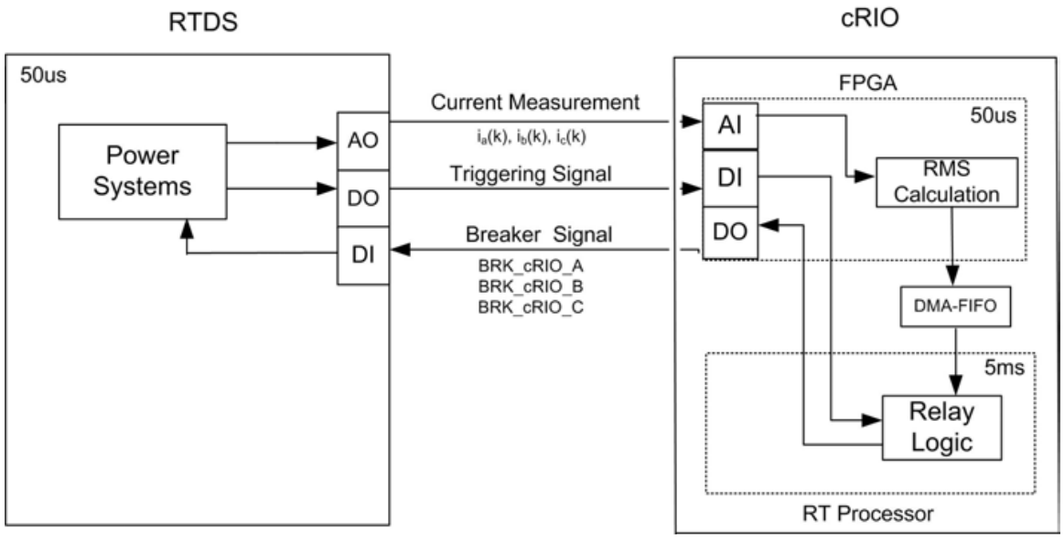


Fig. 5. Functional blocks of RTDS and cRIO in CIL simulation

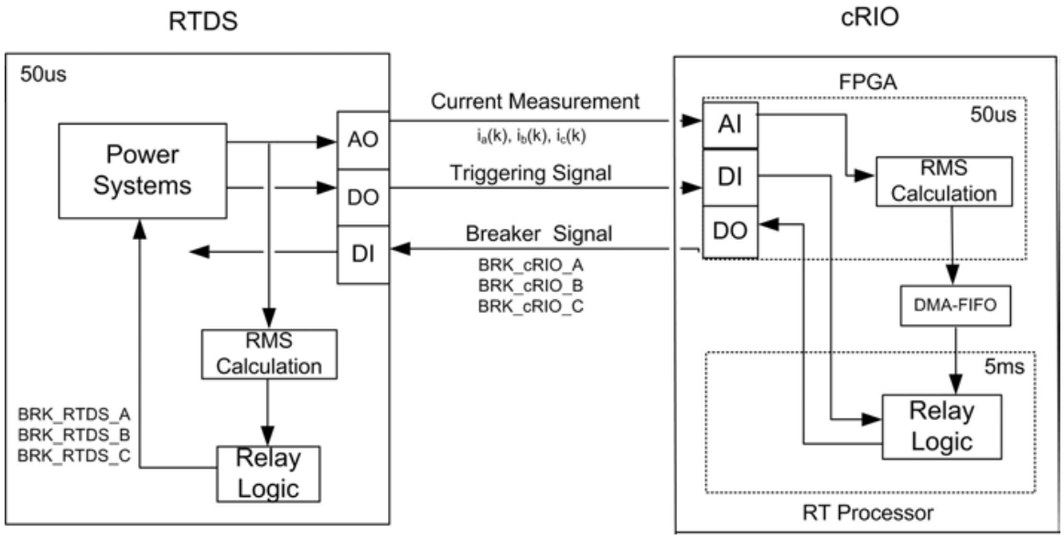


Fig. 6. Functional blocks of RTDS and cRIO in ride-along simulation

In the remainders of this section, a brief description of the implementation of the power system in RTDS will be given. After that, the implementation of overcurrent relay in cRIO, including the program in FPGA and RT processor will be described in detail.

3.1 Brief description of implementation in RTDS

In RTDS, the simulation time step is fixed at 50 us. Figure 7 shows the simple two-bus, three-phase power system implemented in RTDS, while Figure 8(a) shows the analog output module that sends the three current measurements to cRIO, (b) shows the digital input module that receives the breaker signals and one debugging signal from cRIO, and (c) shows the digital output module that sends the triggering signal and one cRIO program control signal to cRIO.

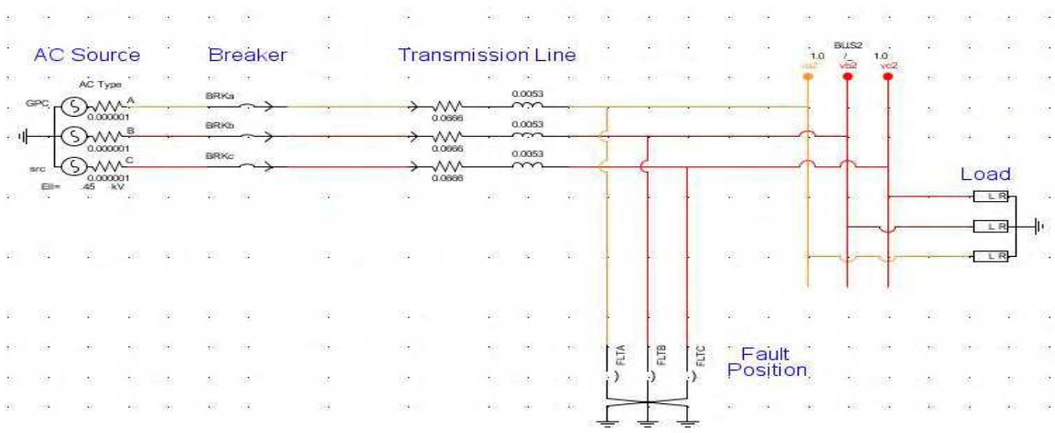
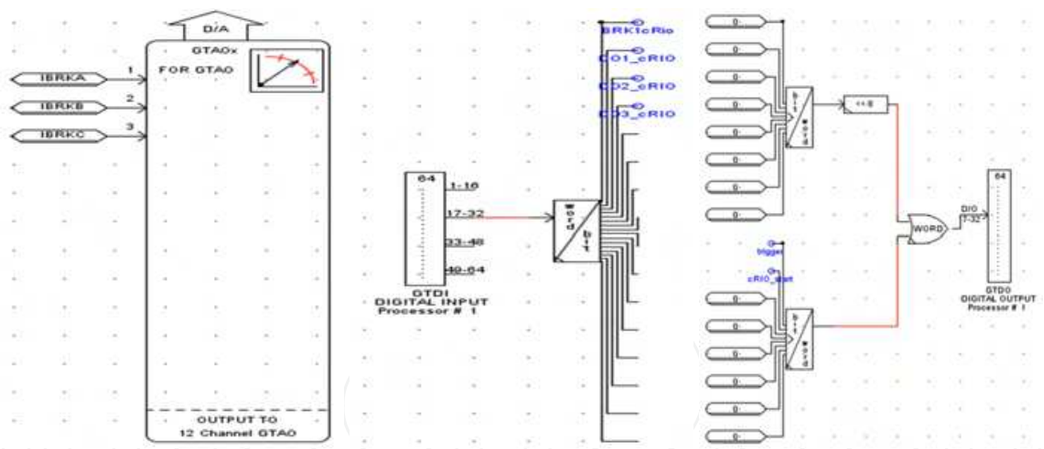


Fig. 7. Circuit diagram in RTDS



(a) Analog output (AO) (b) Digital input (DI) (c) Digital output (DO)

Fig. 8. Input and output modules in RTDS

Table 4 summarizes the detailed description and settings of the I/O modules of RTDS. In the setting of the analog output module, the value of scaling factor should be set according to the range of the current measurements. In this power system, the maximum fault current is found to be 2000A. To make best use of the whole range of the analog output module of RTDS, whose maximum voltage is 10V, the scaling factor should be "5 Volt for 1 Unit". This is because when the current is 2000A, the amp meter will read 2 units. Based on the specified scaling factor, the analog output module of RTDS will output its maximum voltage: 10 V. This scaling factor

should be kept in mind when we write VI program for cRIO so that cRIO can interpret the voltage correctly.

I/O module	Variable	Setting
Analog Output	$i_a(t), i_b(t), i_c(t)$	GTAO Card Number:2 GPC GTIO Fiber Port Number:1 D/A Output Scaling: 5Volt for 1 Unit Oversampling Factor:1 Advance Factor:1
Digital Input	BRK_cRIO_A BRK_cRIO_B BRK_cRIO_C	GTDI Card Number:1 GPC GTIO Fiber Port Number:2
Digital Output	Trig	GTDO Card Number:1 GPC GTIO Fiber Port Number:2

Table 4. Settings of I/O module of RTDS

3.2 Description of implementation of over-current relay in cRIO

In the cRIO, there are two parts that have computing power: FPGA and RT processor. The VI programs in these two parts run simultaneously. The VI in the FPGA is responsible for input/output functions and the RMS calculation of the current measurement signals from the RTDS. The VI in the RT processor is responsible for the relay-logic implementation and user interface. The relay logic makes the decision based on the RMS calculation from the FPGA and updates the breaker signals that are sent to FPGA to be output at the digital output module. In the remainder of this section, detailed description of VIs in the FPGA and in the RT processor will be described.

3.2.1 FPGA

There are two tasks performed in the FPGA. One is the input/output functions, while the other is calculation of the RMS values of current measurements and writing of RMS values into FIFO. Figure 9 shows the flowchart of the program in FPGA. Some initialization (Step A1, A2 and B1) is performed before the execution of the main program (Step A3, A4 and B2). Figure 10 shows the VI implemented in the FPGA.

3.2.1.1 Calibrate analog input module (Step A1 in Figure 9)

Since the analog input module returns a binary value, which is the uncalibrated representation of the voltage as a signed 32-bit integer (I32), we need to convert this binary value into a calibrated nominal value. Therefore, before using the analog input module, we need to get the calibration data that is read from the input analog module. The calibration data includes LSB weight and Offset. These values are used to convert the binary value into the nominal value by Equation 1 (NI, 2004).

$$\text{Nominal value} = [(\text{Binary Value} \times \text{LSB Weight}) - \text{Offset}] \times 10^{-9} \quad (1)$$

3.2.1.2 Allocate FIFO_RMS (Step A2 in Figure 9)

An iterative algorithm of RMS calculation is used, so it is necessary to have one period of the three-phase current measurements. These measurements are stored in three FIFOs, which

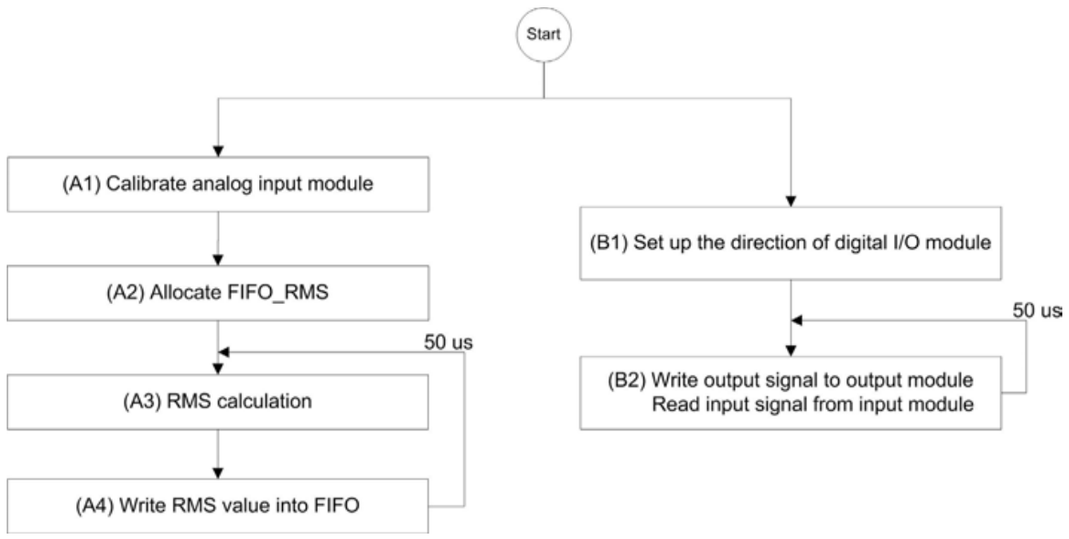


Fig. 9. Flowchart for FPGA VI, shown in Figure 10

are called FIFO_RMS_A, FIFO_RMS_B and FIFO_RMS_C. The following settings are used, including (1) memory type, (2) data representation, and (3) number of elements in FIFOs.
The type of FIFO is a look-up table since it takes less space in FPGA. The representation of the FIFO is set to be I32 since the output of the analog input module is I32. Since the frequency of the current measurements is 60 Hz and the sampling rate is 50us, which is the same as the loop rate at which FPGA is running, the number of elements N in FIFO_RMS is determined by Equation 2.

$$N = \frac{\text{Period}}{\text{Sampling Rate}} = \frac{1/60}{50 \times 10^{-6}} = 333.333 \approx 334 \tag{2}$$

3.2.1.3 RMS calculation (Step A3 in Figure 9)

There are two steps in RMS calculation. The first step is to convert the voltage value read from the analog input module of the cRIO into the correct current value. The second step is to compute the RMS value of the current value using an iterative RMS calculation algorithm.
Step 1: Convert the voltage into current value
An uncalibrated binary value is read from the analog input module of cRIO. To convert this binary value into the nominal value, Equation 1 is used. To get the corresponding current value, the nominal value from Equation 1 is multiplied by a scaling factor, shown in Equation 3. The scaling factor in RTDS makes it output 10 V signal on its analog output module when the current measurement is 2 kA. Therefore 1 V sensed by the cRIO's analog input module means that the corresponding current is 200 A. In other words,the scaling factor of cRIO is 200.

$$\text{Current Value} = \text{Nominal Value} \times \text{Scaling Factor} \tag{3}$$

Since the maximum number for signed 32 bits representation is $2,147,483,647 \cong 2.147 \times 10^9$ (NI, 2004), it is important to make sure the mathematical operation does not overflow. If the operation overflows, the result of the mathematical operation will be incorrect. To make sure the operation does not overflow, LabVIEW provides some operations with an overflow flag. The flag, which can be connected to indicators, will be one if the operation overflows. It is a

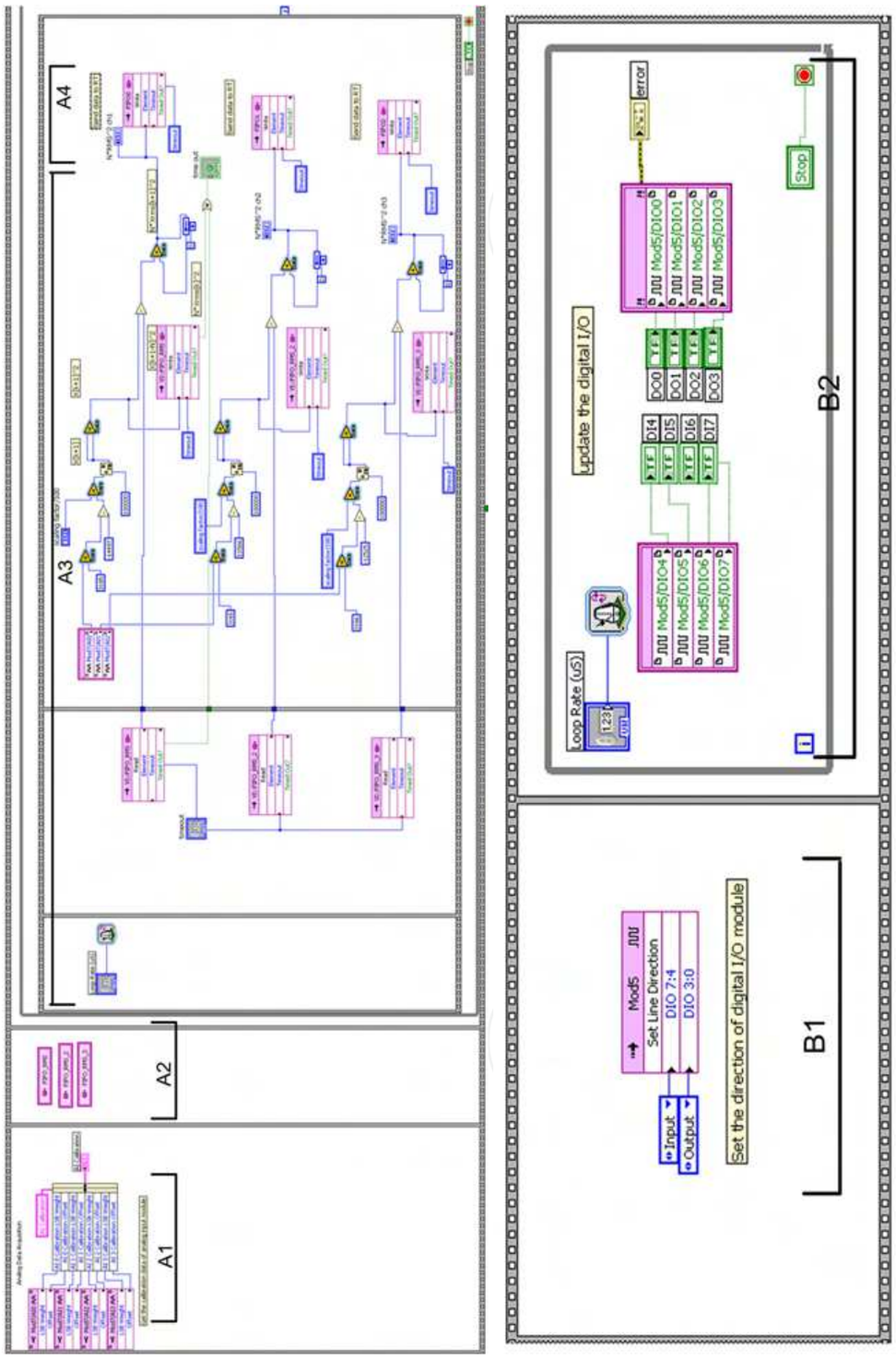


Fig. 10. VI in the FPGA

good practice to check this flag when implementing a program. When you are sure that there is no overflow from the mathematical operations, these overflow indicators can be removed to reduce the FPGA space.

The range of the current measurements is between 20A and 2000A in this simple power system; therefore, the output voltage of RTDS is from 0.1V to 10 V. The corresponding range of the binary value from the analog input module of cRIO is between 327.67 to 32767. To prevent the overflow when Equation 3 is implemented, some arrangements of this equation were made:

$$\begin{aligned}\text{Current Value} &= [\text{Binary Value} \times \text{LSB Weight} - \text{Offset}] \times 10^{-9} \times \text{Scaling Factor} \\ &= [\text{Binary Value} \times \text{LSB Weight} - \text{Offset}] \times 10^{-2} \times 10^{-7} \times \text{Scaling Factor} \\ &= [\text{Binary Value} \times \text{LSB Weight} - \text{Offset}] \times 10^{-2} \times 10^{-5} \times \frac{\text{Scaling Factor}}{100} \\ &= [\text{Binary Value} \times (\text{LSB Weight} \times 10^{-2}) - \text{Offset} \times 10^{-2}] \times 10^{-5} \times \frac{\text{Scaling Factor}}{100}\end{aligned} \tag{4}$$

Figure 11 shows the program that implements the last equation in Equation 4, where the value 318506 is the LSB weight while -1449794 is the offset, both of which are calibration data.

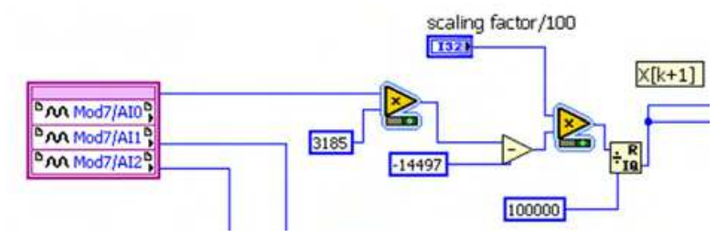


Fig. 11. Program to convert binary value into current value

Step 2: Iterative RMS Calculation

Equation 5 shows the iterative RMS calculation algorithm used in this program. N is the number of elements in the calculation window, x_{rms}^{k+1} is the new RMS value of next time step, x_{rms}^{k} is the present RMS value, x^{k+1} is the new current value, x^{k+1-N} is the current value that is N time steps ago. Since the division operation takes lots of space (slice) of FPGA and time to execute, the formula is converted from Equation 5 to Equation 6 to reduce the number of operations.

$$(x_{rms}^{k+1})^2 = \frac{(x^{k+1})^2 - (x^{k+1-N})^2}{N} + (x_{rms}^{k})^2 \tag{5}$$

$$N(x_{rms}^{k+1})^2 = (x^{k+1})^2 - (x^{k+1-N})^2 + N(x_{rms}^{k})^2 \tag{6}$$

From Equation 6, $N(x_{rms}^{k+1})^2$ is calculated and written into DMA-FIFO, which will be discussed next. So instead of using pickup value of x_{rms}^{k+1}, the pickup value of $N(x_{rms}^{k+1})^2$ is used in over-current relay logic, which is implemented in RT processor.

One point worth discussion is that the mathematical operation in FPGA is integer operation, so the fraction part of a number will be truncated, resulting in some errors. However, as seen from the formula implemented, the integer operation is operated on a large number (the smallest input voltage is 327.68), therefore, the error percentage is very low, less than 1%.

3.2.1.4 Write RMS value into FIFO (Step A4 in Figure 9)

As mentioned before, the speed of FPGA is much faster than that of RT processor. In this case, the loop rate of FPGA is 50 us while the loop rate of RT processor is 5 ms. In order to transfer data (RMS of current value) from FPGA to RT processor without losing data, another FIFO is used. FPGA will put one current RMS value into the FIFO every 50 us, while the RT processor will retrieve data from the FIFO every 5 ms. The number of data that the RT processor takes each time is 100 $[= (5ms)/(50us)]$.
The function of this FIFO is different from that of FIFO_RMS. FIFO_RMS is used to store one period of current values to calculate the RMS value of current measurements, while this FIFO is used as a buffer between the FPGA and the RT processor due to their different execution speeds. Since there are three RMS values of three-phase current value, three FIFOs are used.
To allocate these FIFOs, go to the "Project Explorer", right click on "FPGA Target", select "New"-> "FIFO". Then the properties of FIFO are set: "Target-to-Host DMA" is used and the depth is 255, which is the number of elements of FIFO in FPGA part. (The number of elements of FIFO in RT processor is set in RT program, which will be discussed later in this book chapter) DMA is the abbreviation of Direct Memory Access, which transfers data from the FPGA directly to the memory on the RT processor. A DMA-FIFO consists of two parts, an FPGA part and RT processor part. LabVIEW uses a DMA engine to connect these two parts. When the DMA engine runs, it transfers data between the two parts of the FIFO automatically so they act as one FIFO array. For more information on DMA-FIFO, please refer to Lesson 8, Section F in (NI, 2004). After the DMA-FIFO is set, the value to be passed into RT processor can be written into DMA-FIFO by using FIFO Write function.

3.2.1.5 Set up the direction of digital I/O module (Step B1 in Figure 9)

Because the channels in the digital I/O module (NI 9401 or NI 9403) can be set as either input or output, the input and output channels must be specified.

3.2.1.6 Update the value of digital I/O module (Step B2 in Figure 9)

A while-loop is used to update the value of the digital I/O module. Controllers (DO0, DO1, DO2, DO3) are connected to the output ports of the digital I/O module while indicators (DI4, DI5, DI6, DI7) are connected to the input ports of the digital I/O module.
To be able to run this FPGA VI, this VI needs to be compiled into FPGA code. For detailed information of FPGA compilation, please refer to Lesson 4, Section I in (NI, 2004). To start the execution of this VI, you can hit the RUN button in the LabVIEW window; however in this project, the VI in the RT processor controls the execution of this FPGA VI, including start, stop as well as parameter settings, such as loop rate and scaling factor.

3.2.2 RT processor

There are two functions implemented in RT processor. One function is overcurrent relay logic, shown on the left side of Figure 12. The other function is user interface and parameter settings via front panel communication, shown on the right side of Figure 12. The first function is in the time-critical loop while the second function is in a non-time-critical loop. The reason we separate the VI into two timed-loops is to prioritize these two different functions.

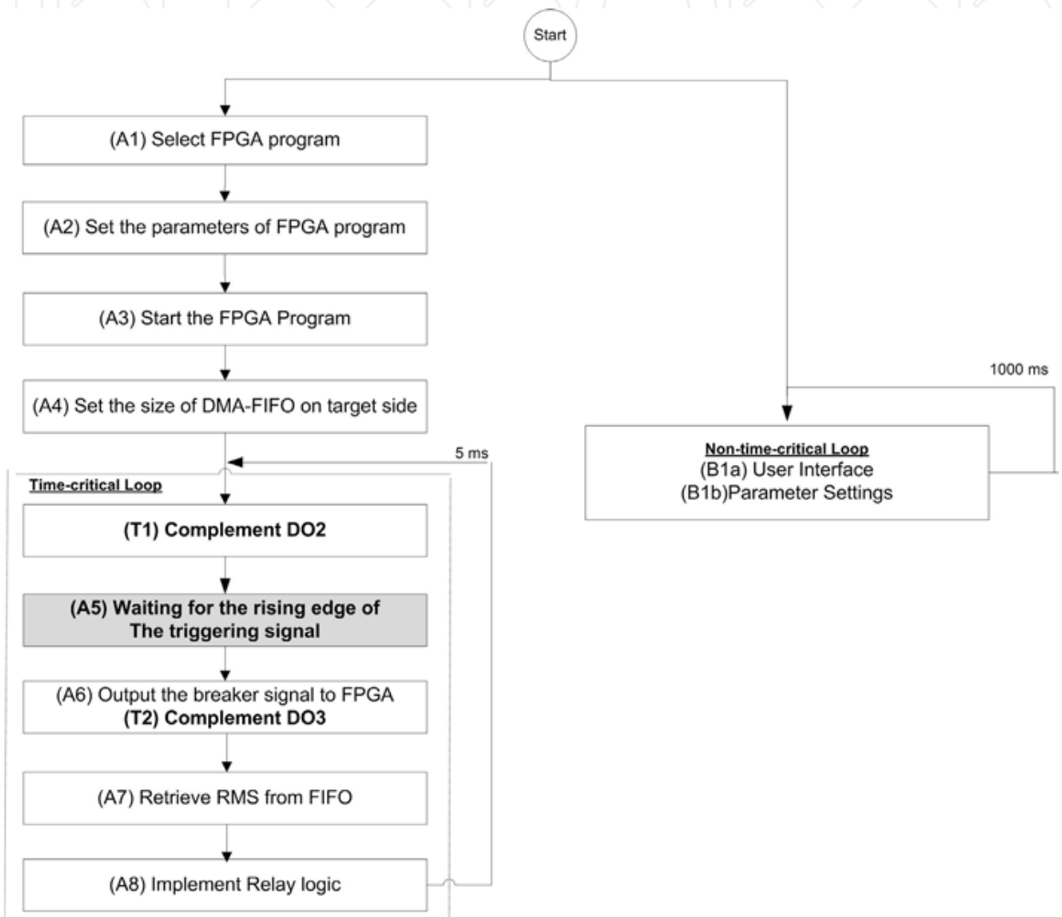
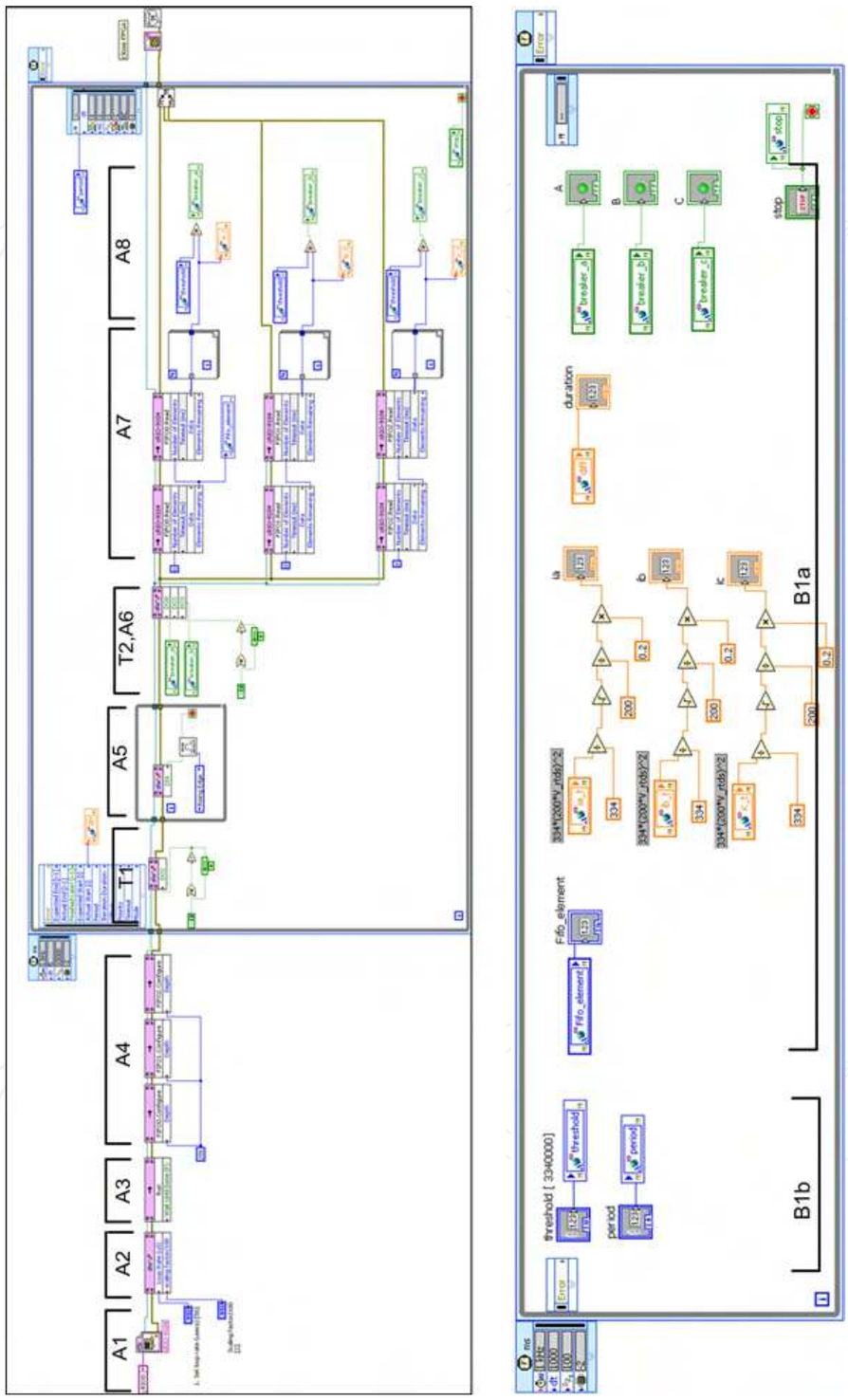


Fig. 12. Flowchart of RT processor VI, , shown in Figure 13



(a) Time-critical loop (b) Non-time-critical loop

Fig. 13. VI in the RT processor

It is possible to put the above two functions in the same timed-loop. However, since front-panel communication takes care of the user interface, such as displaying value in the host-PC screen and sending control value from host-PC to cRIO, it takes time to execute. More importantly, the time it takes is not deterministic. Therefore, to make the over-current relay logic deterministic, two timed-loops with different priorities are used. The time-critical loop with higher priority includes the over-current relay logic while the non-time-critical loop with lower priority includes user interface and parameter settings. Therefore, the RT processor resource will be given to the time-critical loop whenever this time-critical loop needs the resource. Only when time-critical loop doesn't need the resource, will the non-time-critical loop use the resource. In this way, the program implemented in the time-critical loop can be ensured to run deterministically. For more information about the concept of time-critical loop, please refer to Lesson 3 in (NI, 2009) .
Figure 13 shows the VI implemented in the RT processor. Part(a) is the program that includes the initialization and the time-critical loop while part(b) includes the non-time-critical loop.

3.2.2.1 Select FPGA program (Step A1 in Figure 12)

Figure 14(a) shows the program that selects the desired FPGA VI to be executed. The selected file is a FGPA bitfile, which is generated after the FPGA VI is compiled.

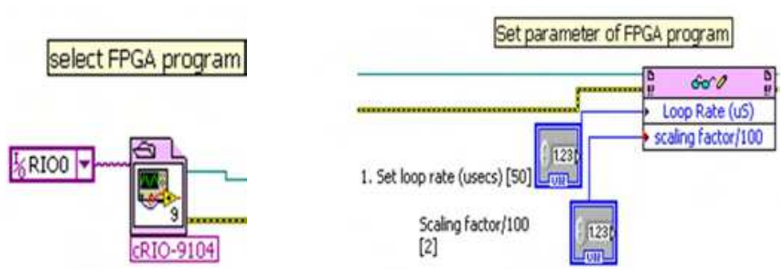


Fig. 14. (a)Select FPGA program, (b)set the parameters of FPGA VI

3.2.2.2 Set the parameters of FPGA program (Step A2 in Figure 12)

Some parameters of the FPGA VI, including the loop rate of FPGA and the scaling factor, are set by the program shown in Figure 14(b). Therefore, users can use the front panel to set these parameters of FPGA VI.

3.2.2.3 Start the FPGA program (Step A3 in Figure 12)

Figure 15(a) shows the program that starts the FPGA VI execution.

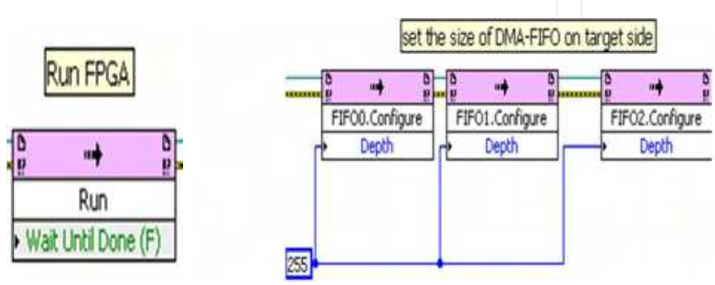


Fig. 15. (a)Run FPGA program, (b)set the FIFO size on target side

3.2.2.4 Set the size of DMA-FIFO on target side (Step A4 in Figure 12)

As mentioned, there are two parts of DMA-FIFO. One is in the FPGA and the other is in the RT processor. The size of DMA-FIFO in the FPGA is set in the FPGA program while the size of DMA-FIFO in the RT processor is set by using the function shown in Figure 15(b). If the size of DMA-FIFO in RT processor side is not specified, then the system will automatically set the size to be twice the size of the DMA-FIFO in the FPGA. Care must be made when determining the size of DMA-FIFO. If the size is too small, the DMA-FIFO tends to be full and data cannot be written into the DMA-FIFO. If the size is too big, it will waste memory space.

3.2.2.5 Waiting for the rising edge of the triggering signal (Step A5 in Figure 12)

Since the clock rates of RTDS and cRIO are different, there should be some mechanism such that the program in RTDS and the program in cRIO do not run out of step. In other words, there needs to be something used to make the two programs proceed together in time.
Using real-time programming concept, the time-critical loop can run in real-time and deterministically. So why do we need to use triggering signal? Because even if we can run the program in the RT processor deterministically, say it runs every 5 ms, it is still possible that in the time frame of RTDS, the program in RT processor runs every 5.001 ms. This is because the accuracy of clock rate of cRIO and RTDS are not exactly the same. Even though the difference is very small, in some applications, this difference may accumulate to cause some problems. Since RTDS is running in real time, the difference will accumulate quickly. Further discussion of triggering signal will be given in the last part of this section.
Figure 16(a) shows the rising edge detection. If the rising edge of the triggering signal is not detected, the program will stay in this while loop. When the rising edge of the triggering signal is detected, the program will leave the while loop and go to the next stage.

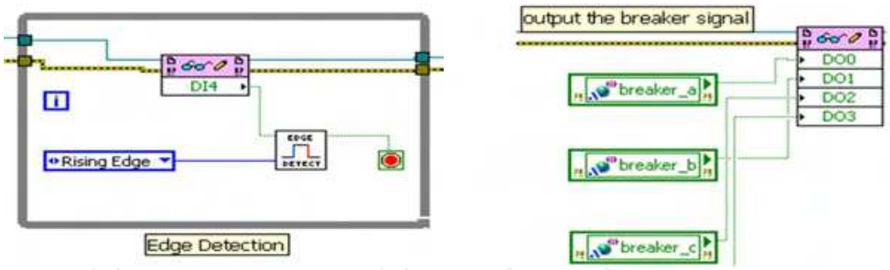


Fig. 16. (a)Rising edge detection of triggering signal, (b)output the breaker control signals

3.2.2.6 Output the breaker signal to FPGA (Step A6 in Figure 12)

Figure 16(b) shows that the breaker control signal is output to the digital output channel DO0, DO1 and DO2. This signal is based on the result of the previous iteration which is 5 ms ago, the loop rate of RT program.

3.2.2.7 Retrieve RMS data from DMA-FIFO(Step A7 in Figure 12)

Figure 17 shows the program of retrieving data (RMS of current value) from the DMA-FIFO. Since the execution speed of RT processor is much slower than that of the FPGA (the loop rate of RT processor is 5 ms while the loop rate of FPGA is 50 us), to prevent this DMA-FIFO from overflowing, the program in RT processor should retrieve all data in the DMA-FIFO each time.

There are two steps to retrieve all data in the DMA-FIFO. First, the left function block in the Figure 17(a) gets the number of elements in the DMA-FIFO and passes this number to the "Number of Elements" of the right function block. The right function block reads "Number of Elements" number of data from DMA-FIFO and transfers it to "Data" as an array. Since the data from DMA-FIFO is an array and only the latest data($N \times RMS^2$)is needed, a for-loop without the N value specified as shown in Figure 17(b) was used.

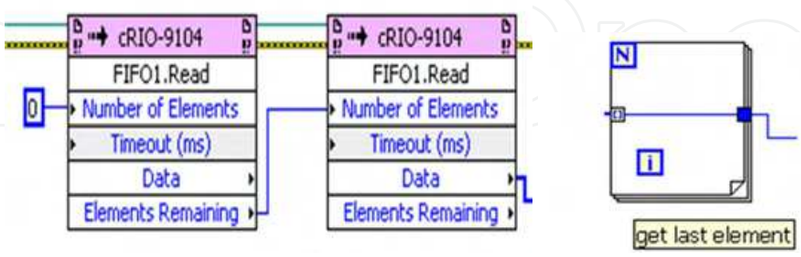


Fig. 17. (a)Retrieve data block in FIFO, (b)get the last element of data block

3.2.2.8 Implement relay logic (Step A8 in Figure 12)

Figure 18 shows the implementation of the overcurrent relay logic, which updates the breaker signal. The retrieved $N \times RMS^2$ value is compared with the threshold (pickup) value. Since the retrieved data is $N \times RMS^2$, the threshold value is also in the same format. In this example, the threshold current is 0.1 kA, so based on the scaling factor of RTDS, the output voltage is 0.5 V. Since the scaling factor of cRIO is 200 and N is 334., the threshold value is 3,340,000, which is equal to $334 \times (200 \times 0.5)^2$. If the retrieved $N \times RMS^2$ value is higher than the threshold value, the breaker signal will be equal to one (open value). Otherwise, the breaker signal will be zero (closed value). Note that this breaker signal is updated, but is not output to the digital output module via FPGA. The new breaker signal is output in the step A6 of next iteration of the time-critical loop.

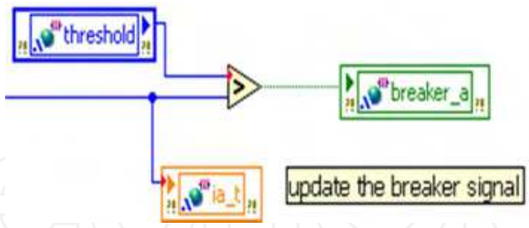


Fig. 18. Comparison with threshold value

3.2.2.9 Implementation in non-time-critical loop (Step B1 in Figure 12)

Below are the tasks for non-time-critical loop:

1. Send the settings of VI from the host PC to the RT processor via front panel communication, including threshold value, period of time-critical loop, and loop rate of time-critical-loop.
2. Send the results of VI to the host PC, including breaker status, time duration of time-critical loop, RMS value of current, number of FIFO elements.
3. Convert $(N \times V_{RMS})^2$ to actual RMS value of current value.

3.2.2.10 Communication between time-critical and non-time critical loops

The signal passing between the time-critical loop and the non-time-critical loop is done via "Shared Variable". The type of this shared variable is set to be "Single-Process", since these two timed-loops are in the same single process.
Since the period of these two timed-loops is different, like the situation where there is FPGA and RT processor, to pass data without loss, FIFO mechanism should be used. Since we just care about the newest data, then FIFO with "single element" is used. To pass multiple data lossless, the FIFO with "multi-element" is used and the number of elements of this FIFO is based on the loop rate of these two timed loops. These settings are done in the "Properties Dialog" of shared variables.

3.2.2.11 How to measure the timing of signal (Step T1, T2 in Figure 12)

cRIO and RTDS execute in real time, so it is impossible to store all the simulation results, making it difficult to debug the program. To check whether the program is executed as expected, especially the timing of the program, we can take advantage of the digital output module of the cRIO as well as the RTDS plotting function.
Figure 19 shows one block of program in the RT processor. This block of program complements the digital output channel DO1 when it is executed. We can place this program around the place of which we want to check the timing. In this VI, we complement DO2 in the beginning of the time-critical loop and complement DO3 when the rising edge of the triggering signal is detected.
To measure these timing signals from the digital output module of the cRIO, we can use oscilloscopes that have limited numbers of channels. Another way is to use RTDS that can plot signals from its digital input module. In this way, we can overlay these numerous timing signals in the same plot and check the timing of the RT program. Moreover, it is possible to store a portion of data points in the plot of RTDS. RTDS greatly facilitates the investigation of these timing signals.

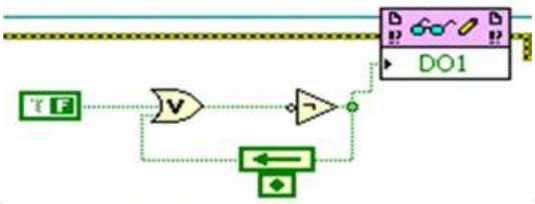


Fig. 19. Complement the digital output when executed

3.3 Discussion of synchronization of RTDS and cRIO

The benefit of outputting the breaker control signals at the rising edge of the triggering signal is that only at the rising edge of the triggering signal can the breaker control signals be output. The breaker control signal cannot be output at other times.
Even though there are some benefits of implementing this synchronization technique, there are some disadvantages. Since each iteration of the time-critical loop in RT processor needs to wait for the rising edge of the triggering signal, it slows down the response of the embedded controller. This amount of the time delay varies, depending on the phase difference between the triggering signal and the time-critical loop. It also depends on the frequency of the triggering signal. If this frequency is low, the time-critical loop will take longer time to wait

for the rising edge of triggering signal. However, if the frequency is too high, then EMI issues will occur. Therefore, the frequency of the triggering signal needs to be selected carefully. In this case study, 1 kHz is selected.
However, if this small time difference is acceptable and doesn't cause any problems for the application, it is not necessary to implement this synchronization technique. This embedded controller will have quicker response.

4. Case studies

In these case studies, a three-phase ground fault was applied at the end of the transmission line as shown in Figure 20. When this fault occurs, the value of the phase currents $i_a(t)$, $i_b(t)$ and $i_c(t)$, and corresponding RMS values increase. As soon as the RMS value was larger than the pickup value of the overcurrent relay, the overcurrent relay would open the breaker to protect the system.

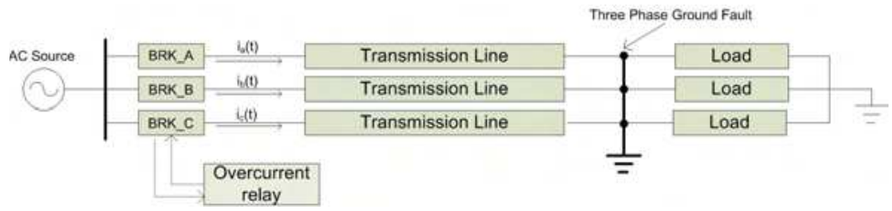


Fig. 20. One-line diagram of system with three-phase ground fault

The loop rate of FPGA was selected to be 50 us, which was the same as the time step in RTDS. The loop rate of the RT processor was selected to be as fast as possible, while at the same time permitting the time-critical loop to run deterministically. Therefore, this loop rate would depend on the computing power of RT processor as well as the complexity of the RT processor VI. In these case studies, three cases where the period of the RMS calculation window was half cycle, one cycle and two cycle, respectively, were compared. For the half cycle, a period is 8.33 ms, for one cycle, a period is 16.66 ms and for two cycle, a period is 33.33 ms. For the half-cycle and one-cycle case, the loop rate of RT processor was 5 ms while for the two-cycle case, the loop rate was 10 ms. This was because in the two-cycle case, the amount of data was larger, requiring more time for RT processor to execute.
The first case study was a modified ride-along mode. The overcurrent relay was implemented in the RTDS and the cRIO. The purpose of ride-along mode is to compare breaker control signals from each overcurrent relays program and see if they are equivalent for determining the accuracy of the implementation.
To ensure that both overcurrent relays observed the same dynamics, the breakers in the RTDS were always closed even when the fault was applied, which was a modified version of the ride-along mode shown in Figure 6. This was done because if one of the overcurrent relays responded first and opened the breaker, the current would decrease, making the other slower overcurrent relay unable to detect the fault. Therefore the breaker signals from this slower overcurrent relay remained closed. The comparison between these two breaker signals would not be right, invalidating the results of ride-along simulation. Therefore, a modified-ride-along mode was used, where the control signals from RTDS and cRIO were not used to control the breaker in RTDS. The breakers were always closed in these studies.
Figure 21 shows the response around the time when the fault was applied. Table 5 summarizes three cases where the window size of RMS calculation were different. The time difference

between the breaker signals from RTDS and cRIO was a little more than the loop rate of the RT processor. The loop rate was 5 ms for half cycle and one cycle windows while the loop rate was 10 ms for two cycle window. This was because when the window size of RMS calculation is two cycles, the number of elements in FIFO_RMS were twice the size of one cycle case, requiring more time to retrieve the data for cRIO computation. Therefore, the loop rate of RT processor had to be increased to ensure deterministic operation.
Another observation was that when the RMS calculation window was smaller, the time difference between fault and breaker signal was smaller since the RMS calculation was more sensitive if its calculation window was small.

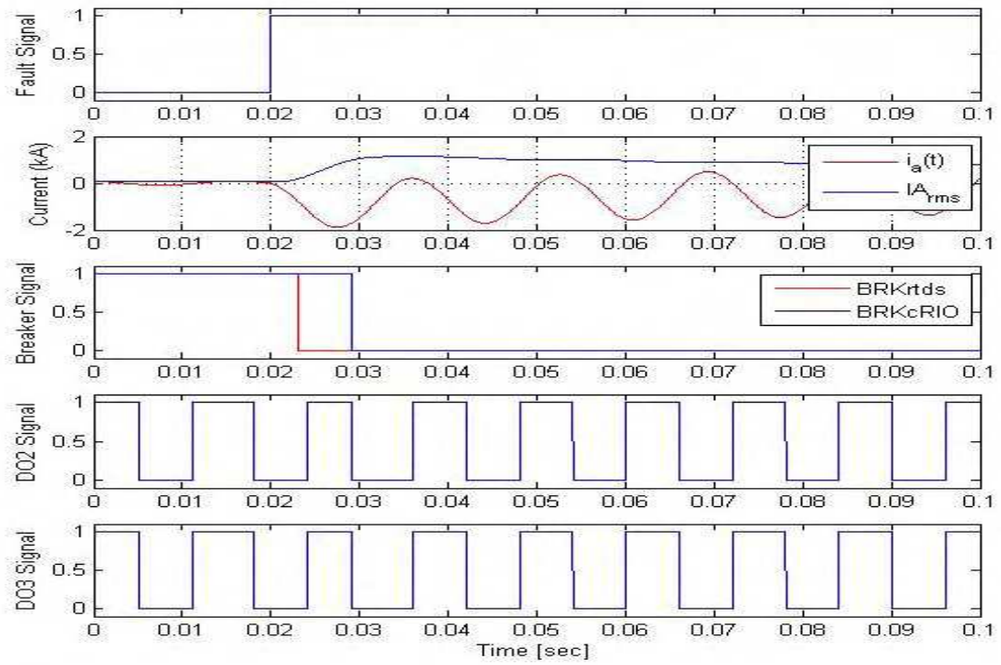


Fig. 21. Response in modified-ride-along mode to a fault scenario

Signals \ RMS window size	0.5 cycle	1 cycle	2 cycle
Fault and BRK_RTDS	0.0043 sec	0.0051 sec	0.0053 sec
Falut and BRK_cRIO	0.0104 sec	0.0102 sec	0.0159 sec
BRK_RTDS and BRK_cRIO	0.0060 sec	0.0051 sec	0.0106 sec

Table 5. Time difference among signals for three RMS calculation window in modified-ride-along mode

The second set of case studies was CIL mode. The overcurrent relay implemented in RTDS was disabled and the breaker signals from cRIO controlled the breakers implemented in RTDS. Figure 22 shows the waveform around the time when a three-phase ground fault was applied. It can be observed that the breaker signals from cRIO successfully opened the breakers after the fault occurred. The time difference between the fault occurrence and the opening of the

breaker was approximately 10 ms in each phase which was two times the loop rate of RT processor. Like the first set of case studies, if the window size of RMS calculation was smaller, it took less time to open the breaker after fault was applied, which is shown in Table 6.

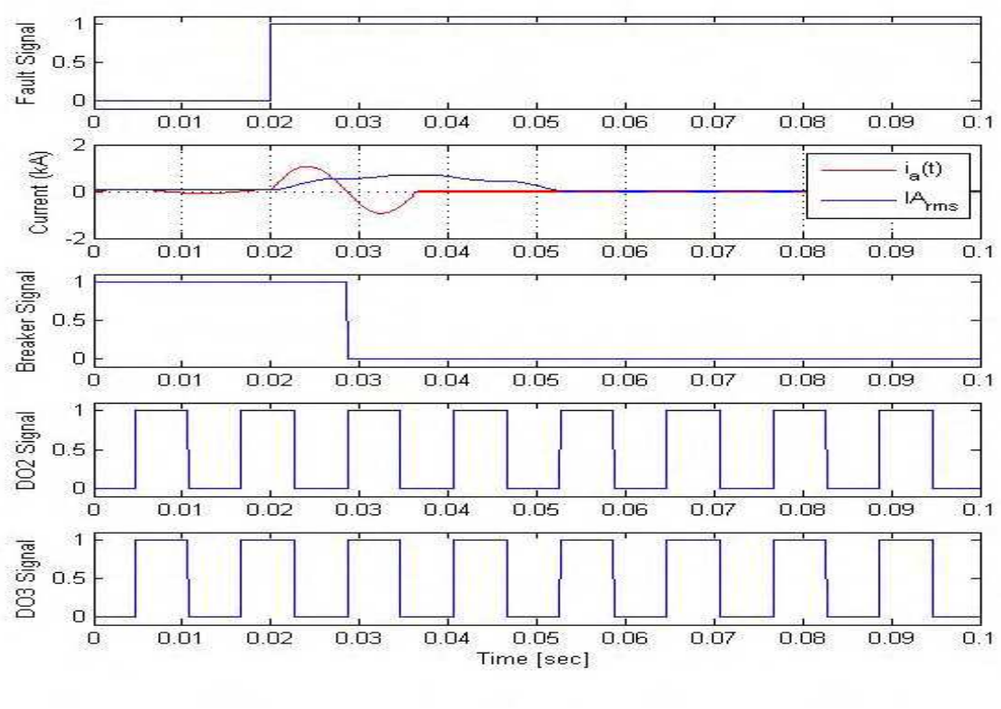


Fig. 22. Response in CIL mode to a fault scenario

RMS window size / Signals	0.5 cycle	1 cycle	2 cycle
Fault and BRK	0.01034 sec	0.0111 sec	0.016 sec

Table 6. Time difference among signals for three RMS calculation window in CIL

The results of a timing analysis of the measured signals from the modified-ride-along mode simulation are shown in Figure 23. Since the breaker signal (BRK_RTDS) from the overcurrent relay in RTDS is updated when the time-critical loop starts (when DO2 changes its polarity), the difference between the fault signal and BRK_RTDS is T_1. The breaker control signal from the overcurrent relay in cRIO (BRK_cRIO) is updated when there is a rising edge during the execution of time-critical loop (when DO3 changes its polarity), therefore the difference between the fault signal and BRK_cRIO is $T_1 + T_2$. Sometimes it may take an extra loop iteration for the RMS value to be greater than the pickup value, so the difference maybe $T_1 + T_2 + T_3$, where T_3 is close to T_{RT}. Therefore, the time difference between BRK_RTDS and BRK_cRIO is $T_2 + T_3$.The range of T_1 is between 0 and T_{RT}, range of T_2 is between 0 and to T_{tri} and the range of T_3 is between 0 and T_{RT}, where T_{RT} is the loop rate of the time-critical loop in RT processor and T_{tri} is the period of the triggering signal.

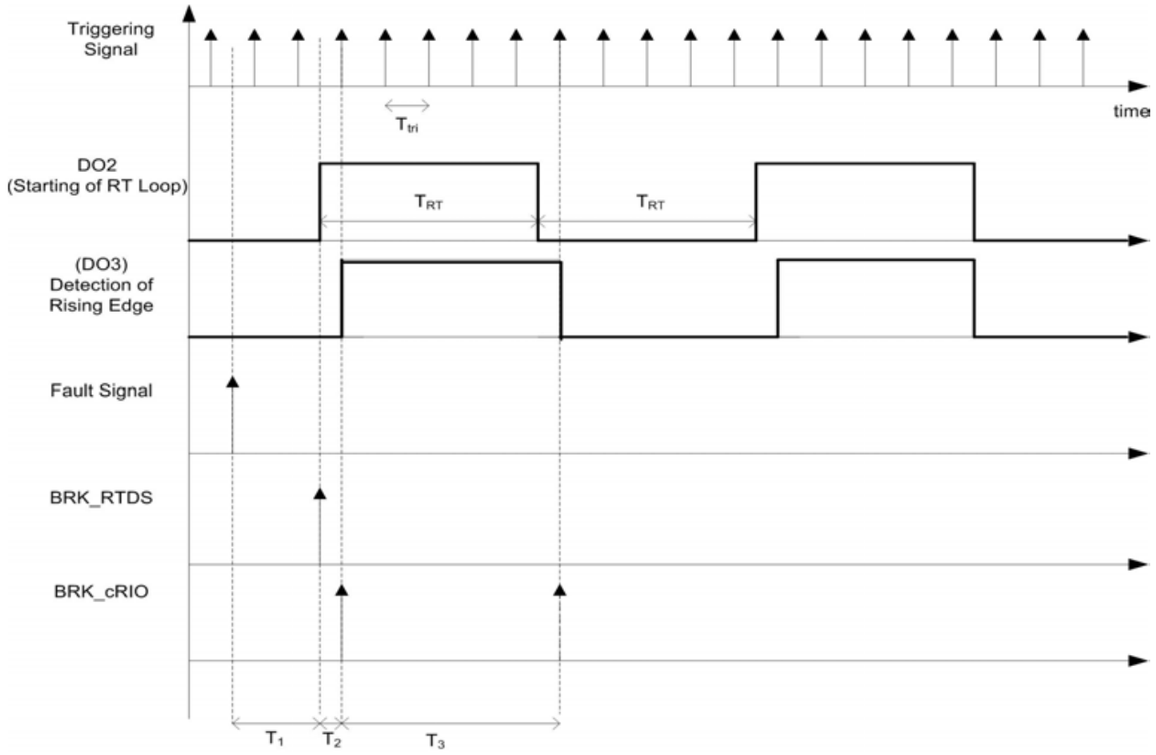


Fig. 23. Time analysis of simulation results in Figure 21

5. Conclusions and future work

In this book chapter, the concepts and the implementation of a real-time rapid embedded power system control prototyping simulation test bed were described. The methodologies for implementing embedded controller-in-the-loop simulation (CIL) was explained. To illustrate the functionality of the test bed, the detailed implementation of an overcurrent relay protection scheme for CIL simulation was described, including the settings and programming of RTDS, the programming in cRIO, which included the FPGA and RT processor, by using LabVIEW. Also, a synchronization approach for the RTDS and cRIO was discussed.

Additional studies are ongoing to refine the test bed design, such as studies to investigate the time delay and data synchronization and its relationship to system performance and stability. For example, in a power system, to be transient stable, a fault has to be cleared within a specific amount of time. If the response of the breaker is too late or incorrect, the system will become unstable.

Further the test bed is being utilized to study new control strategies and their performance. In power system applications, voltage regulation and load management are used to maintain the proper operation of power systems. These control strategies can be verified and validated using the test bed discussed in this chapter. Due to the real-time environment of RTDS and RT target, it is possible to observe the effectiveness of these control strategies in the real-time simulation environment.

6. Acknowledgment

The authors gratefully acknowledge the contributions of Tania Okotie and the funding from Office of Naval Research under grants N00014-09-1-0579, N0014-04-1-0404, and N00014-07-1-0939.

7. References

French, C. D., Finch, J. W. & Acarnley, P. P. (1998). Rapid prototyping of a real time dsp based motor drive controller using simulink, *Simulation, International Conference on*, pp. 284–291.

Isermann, R., Schaffnit, J. & Sinsel, S. (1999). Hardware-in-the-loop simulation for the design and testing of engine-control systems, *Control Engineering Practice* pp. 643–653.

J. Duncan Glover, Mulukutla S. Sarma, T. O. (2007). *Power System Analysis and Design*, 4 edn, CL-Engineering.

Karpenko, M. & Sepehri, N. (2006). Hardware-in-the-loop simulator for research on fault tolerant control of electrohydraulic flight control systems, *American Control Conference*, p. 7.

Keunsoo, H., Seok-Gyu, O., MacCleery, B. & Krishnan, R. (2005). An automated reconfigurable fpga-based magnetic characterization of switched reluctance machines, *Industrial Electronics, Proceedings of the IEEE International Symposium on*, pp. 839–844.

Lavoie, M., QuÃl'-Do, V., Houle, J. L. & Davidson, J. (1995). Real-time simulation of power system stability using parallel digital signal processors, *Mathematics and Computers in Simulation* pp. 283–292.

Ledin, J. (2001). *Simulation Engineering*, CMP Books, Lawrence, USA.

NI (2004). *CompactRIO and LabVIEW Development Fundamentals*.

NI (2009). *LabVIEW Real-Time Application Development Course Manual*.

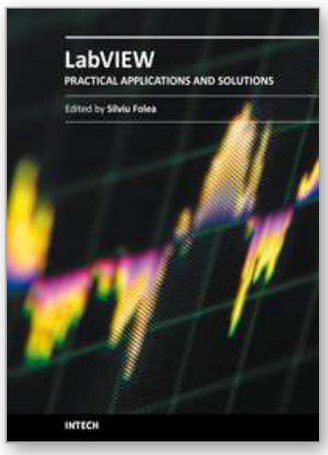
NI CompactRIO (2011). Available at: `http://www.ni.com/compactrio/`.

Postolache, O., Dias Pereira, J. M. & Silva Girao, P. (2006). Real-time sensing channel modelling based on an fpga and real-time controller, *Instrumentation and Measurement Technology Conference, Proceedings of the IEEE on*, pp. 557–562.

Real Time Digital Simlator - RTDS (2011). Available at: `http://www.rtds.com/`.

Spinozzi, J. (2006). A suite of national instruments tools for risk-free control system development of a hybrid-electric vehicle, *American Control Conference, 2006* p. 5.

Toscher, S., Kasper, R. & Reinemann, T. (2006). Implementation of a reconfigurable hard real-time control system for mechatronic and automotive applications, *Parallel and Distributed Processing Symposium, IPDPS 20th International* p. 4.



Practical Applications and Solutions Using LabVIEW™ Software

Edited by Dr. Silviu Folea

ISBN 978-953-307-650-8
Hard cover, 472 pages
Publisher InTech
Published online 01, August, 2011
Published in print edition August, 2011

The book consists of 21 chapters which present interesting applications implemented using the LabVIEW environment, belonging to several distinct fields such as engineering, fault diagnosis, medicine, remote access laboratory, internet communications, chemistry, physics, etc. The virtual instruments designed and implemented in LabVIEW provide the advantages of being more intuitive, of reducing the implementation time and of being portable. The audience for this book includes PhD students, researchers, engineers and professionals who are interested in finding out new tools developed using LabVIEW. Some chapters present interesting ideas and very detailed solutions which offer the immediate possibility of making fast innovations and of generating better products for the market. The effort made by all the scientists who contributed to editing this book was significant and as a result new and viable applications were presented.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen