IET Generation, Transmission & Distribution

Research Article

IET Journals
The Institution of Engineering and Technology

Real-time digital multi-function protection system on reconfigurable hardware

ISSN 1751-8687
Received on 14th June 2015
Revised on 17th October 2015
Accepted on 5th April 2016
doi: 10.1049/iet-gtd.2015.0718
www.ietdl.org

Yifan Wang, Venkata Dinavahi ✉

Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada, T6G 2V4
✉ *E-mail: dinavahi@ece.ualberta.ca*

Abstract: This study proposes a multi-function power system protective relay hardware design built with various functional hardware processing cores on the field programmable gate array (FPGA). This practical and systematic method lends itself to a paralleled and pipelined hardware emulation of individual signal processing and protection components. Detailed emulation designs are presented for the following protective relays: distance, directional overcurrent, voltage, and frequency protection. The necessary signal processing functions required to operate these relays are also emulated, allowing the protection system to be stand-alone and fed with instantaneous fault data. Real-time experimental results are presented to verify the functions of the target hardware relay on the Xilinx® Virtex-7 FPGA. Case studies provide the efficacy of the multi-function relay design in terms of accuracy, latency, and resource consumption.

1 Introduction

Protective relays safeguard the normal operation of power systems by tripping circuit breakers to isolate faults and other abnormal conditions [1–3]. They are employed to protect every major power system equipment such as rotating machines, transmission lines, transformers, shunt reactor banks, distribution networks etc. Relay technologies have been through mainly three generations of evolution [4, 5]: electromechanical relays, solid-state relays, and numeric relays. Electromechanical relays relied on the principle of mechanical force to operate relay contacts in response to a stimulus which was mainly provided by electromagnetic torque produced by currents through windings on a magnetic core. While such relays were reliable, their main drawbacks were power consumption, operating speed, size, and the need for maintenance. Solid-state or static relays appearing in the 1980s used analogue electronics such as transistors, diodes, capacitors and inductors, to generate the relay function. Although static relays were smaller in size, low cost, and required less maintenance, user programming was limited to basic relay characteristic, individualised functions, and they lacked communication and event recording functions.

Starting in early 1990s, digital numerical relays were introduced, the first generation of which relied on microprocessors and microcontrollers [5–12]. Earlier versions used 8-bit or 16-bit microprocessors, the limited power and memory of which restricted the number of waveform samples measured per cycle, and the relay operating speed. The second generation numerical relays, employed one or more digital signals processors (DSPs) or CPUs for running the protection algorithms. Taking advantage of software programmability, the most widely used numeric relays currently tend to have more protection functions integrated in one device so that they can execute more tasks under different fault conditions. The adoption of IEC 61850 [13, 14] led to a standardised data and communication protocol between all intelligent electronic devices within a substation resulting in a comprehensive fully-digital monitoring, protection, and control application for modern substation automation. Using dedicated software multiple functions could now be realised in real-time to make the relay more user friendly and flexible; however, some manufacturers provided so much programming capability that extensive customer training was required to set and operate these relays [15]. The relay operating system software mainly consisted of four components: system services software, human machine interface (HMI) software, protection application software, and auxiliary functions software [5]. All of this software overhead contributes to the computational burden. Furthermore, the sequentiality of software code inherently limits the operating speed of the designs, and precludes the exploitation of task and algorithmic parallelism in the implementation. The growing demand for computational speed to accommodate more functions in real-time, or flexible reconfigurable features cannot easily be met by such platforms.

Power systems are undergoing dramatic changes. The smart grid concept integrates newer and disparate energy sources (distributed generators), power electronic interfaces, smart loads, circuit breakers, switches, and information and communication technologies to create a complex and multi-domain system that seeks to better meet the needs of customers and producers alike [16–20]. Operating such systems reliably will require the design of novel protection and control algorithms. In response protective relay technologies must also evolve to meet the computational challenges. The field programmable gate array (FPGA) technology is mature and well positioned to answer the need for high computational demands at low latencies.

Reconfigurable hardware such as FPGAs have many desirable features such as a high-capacity, high-bandwidth, inherent parallel hardwired architecture, high-speed transceivers, and a host of development tools, that a modern multi-function relay needs in demanding applications such as in smart grid application. FPGAs have proven to be efficacious for not only real-time power system transient modelling and simulation [21–23]. In the literature, few works [24–27] have been reported with protective relay design on FPGAs. In [24, 25], single function relays are implemented for transformer and high voltage direct current line protection respectively, however, the designs are not revealed in complete detail. In [26] a multi-function relay was implemented on the FPGA with two functions: a voltage and a frequency relay; however, the implementation required pre-processed signals as input, since the used device was not large enough to implement higher accuracy signal processing algorithms.

In this work we propose a multi-function protection system built with various functional hardware processing cores each of which is a fully self-contained stand-alone digital hardware component with input/output interfaces which can be replicated rapidly. In the

multi-function protection system all the required signal processing and protection elements are implemented entirely in hardware on the 28 nm Xilinx® XC7VX485T Virtex-7 device. The entire hardware design is implemented in VHDL in 32 (64)-bit floating point format for high accuracy. In the hardware emulation, parallelism is fully exploited to achieve a low-latency realisation of complex algorithms. Pipelining design is also used to increase the overall design throughput. The organisation of this paper is as follows. The detailed hardware emulation of multi-function protection system is presented in Section 2. Section 3 presents test cases and real-time experimental results. Finally, Section 4 gives the conclusion of the paper.

2 Hardware emulation of multi-function protection system

The overall architecture of the proposed hardware multi-function protection system on FPGA is shown in Fig. 1. The design is constructed in the form of hardware processing cores. It consists of two major parts: the signal processing core (SPC) and the protective relay core (PRC). Within SPC, the main components include:

- COordinate rotation digital computer (CORDIC).
- Discrete Fourier transform (DFT) with dc offset removal.
- Symmetrical components.
- Zero-crossing detection.

The PRC consists of:

- Fault detection.
- Distance protection.
- Directional overcurrent protection.
- Over/under voltage protection.
- Over/under frequency protection.

After the voltage and current data enter the protection system, they are processed by the SPC. For the sake of simplicity, potential and current transformer nonlinear effects on fault data are not considered in the design. The CORDIC processing core provides trigonometric and nonlinear function values that are necessary for arithmetic computations, while the DFT processing core extracts fundamental amplitude and phase of the required signals. The symmetrical components processing core computes the sequence components of the three-phase system while the zero-crossing processing core is for power frequency estimation. Depending on the protection function chosen, the required signal processing is carried out, and the corresponding trip signal is issued. In the following subsections, the hardware design details of each of the above processing cores are given.

2.1 Signal processing core

The SPC computes the power system quantities needed by the various protection functions. The philosophy and hardware implementations are given below. All the basic floating point computation operators in this design such as adder, subtractor, multiplier etc., are achieved by using the Xilinx® IP Core Generator due to its flexible configuration.

(i) *CORDIC processing core:* The iterative CORDIC method [28] has the advantage of higher precision over the conventional look-up-table (LUT) based method. The CORDIC algorithm was initially designed to compute trigonometric functions. It was generalised and unified to deal with linear and hyperbolic functions as well [29]. Under different coordinate systems, the CORDIC can generate different elementary functions. In order to achieve 10^{-5} degree accuracy in this work for nonlinear value computations, the LUT method required BRAM capacity far in excess of the available amount of BRAM (4.5 MB) on the FPGA; therefore the generalised CORDIC was preferred in this work. In the CORDIC, the basic trigonometric and hyperbolic functions are computed by rotation mode and vectoring mode iteratively. The general form of the CORDIC iteration is as follows with FPGA implementation given in Fig. 2*a*

$$\begin{aligned} x_{i+1} &= x_i - m \cdot \sigma_i \cdot y_i \cdot 2^{-i} \\ y_{i+1} &= y_i + \sigma_i \cdot x_i \cdot 2^{-i} \\ z_{i+1} &= z_i - \sigma_i \cdot \alpha_{m,i} \\ i &= i + 1, \end{aligned} \tag{1}$$

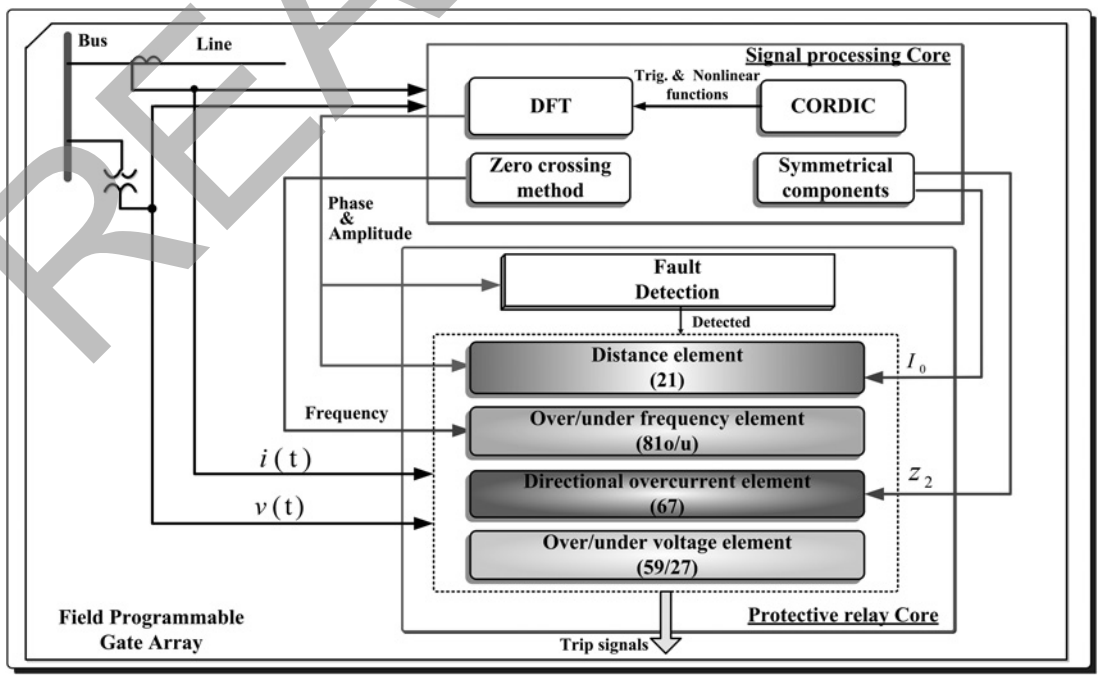

Fig. 1 *Overall architecture of the hardware multi-function protection system*

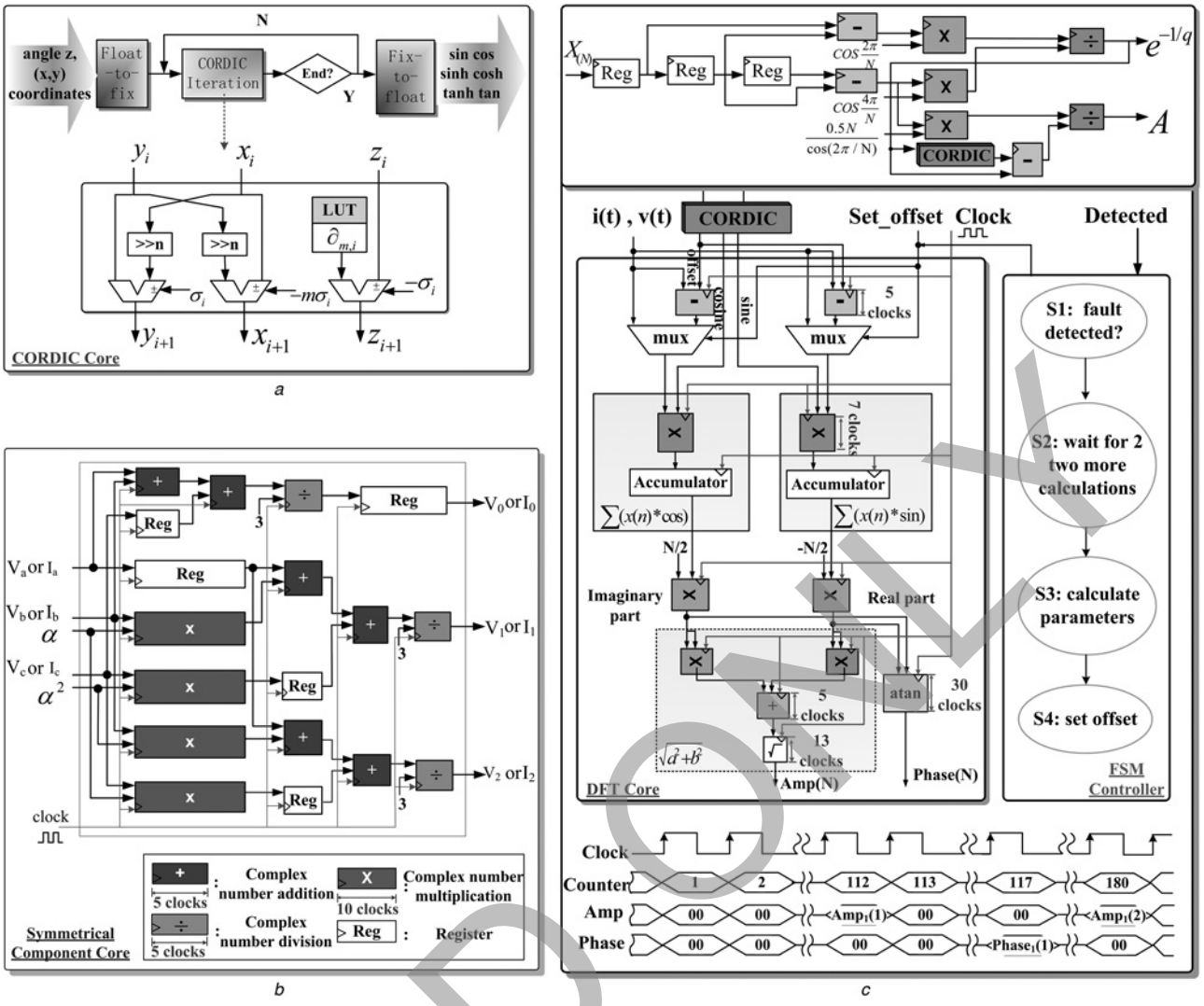

Fig. 2 *Signal processing core*

a Iterative CORDIC processing core
b DFT processing core with timing diagram
c Symmetrical components calculation processing core

where x_i, y_i, and z_i are the 2D rotating vector coordinates and angle of rotation, respectively, in ith iteration. The parameter m stands for the coordinate system; $m = 1$ results in circular coordinates, while $m = -1$ results in hyperbolic coordinates. $\alpha_{m,i}$ is the value of $\tan^{-1}(2^{-i})$ when $m = 1$ or $\tanh^{-1}(2^{-i})$ when $m = -1$. σ_i indicates the rotation direction given as sign (z_i) or $-\text{sign}(y_i)$ for rotation mode and vectoring mode, respectively.

In each iteration, the plane vector $v_i = (x_i, y_i)^T$ rotates to $v_{i+1} = (x_{i+1}, y_{i+1})^T$. z_i tracks the angle at each rotation. For instance, in the rotation mode, when estimating the trigonometric values of a given angle, z_i rotates toward 0 with a micro rotation angle of θ_i which has the tangent value of 2^{-i} (or hyperbolic tangent value of 2^{-i} in hyperbolic coordinates) in each iteration. These values are pre-calculated and stored in the FPGA BRAM. At the last iteration, when the sum of rotated angles reaches the input angle, the x and y coordinates indicate the sin and cos values of the input angle. On the other hand, in the vectoring mode the coordinates of a vector are given a priori while the magnitude and angular argument of the original vector are computed.

FPGA realisation for CORDIC iteration requires only addition/subtraction and shifting operations (multiplication by 2^{-i} can be done by bit shifting to the right), which can be easily achieved in hardware. The iterative stage n in this design is set to 20 to obtain an accuracy of 10^{-5} degrees or more. While the iterative CORDIC algorithm is based on fixed-point data, all the incoming data which are normally in 32-bit floating-point format are converted to fixed-point number using a float-to-fix module. The computation results are converted back to floating-point numbers by a fix-to-float module at the end of iteration.

With the computation of the elementary functions through CORDIC, the non-linear functions required in DFT such as e^x and natural logarithm function $\ln(x)$ could be obtained from

$$e^x = \sinh(x) + \cosh(x),$$
$$\ln(x) = 2\tanh^{-1}\left|\frac{x-1}{x+1}\right|. \quad (2)$$

(ii) *DFT processing core:* This hardware building block computes the fundamental frequency components of input voltage and current signals. Dc offset removal is also available for current signal since it will mostly be distorted by the dc offset during a fault. The full-cycle DFT (FCDFT) calculation with dc offset removal [30] that requires one cycle plus two samples is chosen.

The traditional FCDFT algorithm calculates the fundamental components as

$$X_{(1)} = \frac{2}{N}\sum_{n=0}^{N-1} x(n) \times (\cos \omega_1 n\Delta T - j \sin \omega_1 n\Delta T) = \frac{2}{N}\sum_{n=0}^{N-1} x(n) \times \left(\cos\frac{2\pi n}{N} - j \sin\frac{2\pi n}{N}\right), \quad (3)$$

$$A_1 = \sqrt{X_{1\text{real}}^2 + X_{1\text{imag}}^2}, \quad (4)$$

$$\theta_1 = \arctan(X_{1\text{imag}}/X_{1\text{real}}), \quad (5)$$

where $x(n)$ is the discrete-time sinusoidal input signal, N is the number of samples in a fundamental period, with ω_1 being the fundamental angular frequency, and ΔT the sampling interval. The amplitude A_1 and phase θ_1 of X_1 can be calculated as in (4) and (5). For the dc offset removal strategy, the input signal contains a decaying dc offset $A\,\mathrm{e}^{-t/\tau}$ (time constant $\tau = q\Delta T$). After computing $X_{\text{real}(N)}$, $X_{\text{real}(N+1)}$, $X_{\text{real}(N+2)}$ which are real parts of three FCDFT fundamental phasor calculation results, the parameters of the dc offset are obtained as

$$\mathrm{e}^{-1/q} = \frac{(X_{\text{real}(N+2)} - X_{\text{real}(N+1)})\cos(2\pi/N)}{(X_{\text{real}(N+1)} - X_{\text{real}(N)})\cos(4\pi/N)}, \quad (6)$$

$$A = \frac{0.5N(X_{\text{real}(N+1)} - X_{\text{real}(N)})}{\cos(2\pi/N)\,\mathrm{e}^{-1/q}(\mathrm{e}^{-N/q} - 1)}. \quad (7)$$

The dc offset can be then subtracted from the sampled signal $x(k)$ as

$$z(k) = x(k) - A\,\mathrm{e}^{-k/q}. \quad (8)$$

The DFT calculation is based on a moving window strategy. Once every full-cycle window (68 samples) of data calculation is completed, the window moves forward by one sampling point to the next computation cycle. The DFT processing core gets the *set_offset* control signal from the finite state machine (FSM) controller (Fig. 2*b*) which has four transition states going from *S*1 to *S*4. Under the control strategy, the controller first senses the fault detected signal, and waits for two more sample calculation cycles to finish dc offset parameters calculation. Then the offset could be subtracted if fault is detected and the input signal is current. Within the DFT core shown in Fig. 2*b*, the real and imaginary parts are calculated in parallel. The accumulation process is achieved by using the floating-point adder that has only one clock cycle latency. Fig. 2*b* also shows the logic timing diagram for the module. After the very first round of calculation, which takes 112 clock cycles for amplitude and 117 for phase output, with every fundamental window of interval the module generates a new calculation result.

(iii) *Symmetrical components processing core:* This hardware module calculates the sequence components of a three-phase system. The positive-, negative-, and zero-sequence components are obtained by multiplying a constant sequence transform matrix with the original three-phase signals [31]. For example, the phase current $I_{abc} = [I_a\ I_b\ I_c]^{\mathrm{T}}$ can be transformed into to the sequence domain and the resulting symmetrical components can be obtained as follows

$$I_{012} = A^{-1}I_{abc} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 \\ 1 & \alpha^2 & \alpha \end{bmatrix}\begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} \quad (9)$$

where $A = [1\quad 1\quad 1;\ 1\quad \alpha^2\quad \alpha;\ 1\quad \alpha\quad \alpha^2]$, the subscripts 0, 1 and 2 denote zero-, positive- and negative-sequence components, respectively, and $\alpha = \mathrm{e}^{\mathrm{j}2\pi/3}$.

The FPGA implementation of sequence components calculation is shown in Fig. 2*c*. Inside each elementary complex number computation, the floating-point calculation is processed in the module with latencies shown in the figure. The registers inject synchronising latencies.

(iv) *Zero-crossing detection processing core:* This hardware module provides the algorithm necessary to estimate the fundamental frequency that is crucial for the frequency protection function. Among the various frequency estimation algorithms [32, 33], the zero-crossing method is used here. The theoretical illustration and hardware implementation is given in Fig. 3. Based on the original fault data precision of 10^{-5} and the satisfying experiment result comparison with PSCAD/EMTDC® off-line simulation, a 64-bit precision data format is used for calculation and implementation. The major step in this method is to measure the time interval between the two successive zero crossings of the voltage signal to find the frequency using

$$f = \frac{1}{T} = \frac{1}{2(t_2 - t_1)}. \quad (10)$$

The zero-crossing points are first approximated by comparing the signs of two neighbouring points. Then five samples around the zero-crossing point are used to fit a cubic interpolant. For instance, in Fig. 3*a*, the five sample indices $[n-2,\ n-1,\ n,\ n+1,\ n+2]$, with n and $n+1$ being the indices which have opposite signs, are used to determine the cubic polynomial

$$f(x) = ax^3 + bx^2 + cx + d. \quad (11)$$

The coefficients of (11) can be determined by least squares method as

$$A = [a\quad b\quad c\quad d]^{\mathrm{T}} = (K^{\mathrm{T}}\boldsymbol{K})^{-1}K^{\mathrm{T}}\boldsymbol{V}, \quad (12)$$

where $\boldsymbol{K}$ is the constant Vandermonde matrix, and $\boldsymbol{V}$ is the voltage sample vector, given as

$$\boldsymbol{K} = \begin{bmatrix} (-2)^3 & (-2)^2 & -2 & 1 \\ (-1)^3 & (-1)^2 & -1 & 1 \\ 0 & 0 & 0 & 1 \\ 1^3 & 1^2 & 1 & 1 \\ 2^3 & 2^2 & 2 & 1 \end{bmatrix}, \quad \boldsymbol{V} = \begin{bmatrix} v_{n-2} \\ v_{n-1} \\ v_n \\ v_{n+1} \\ v_{n+2} \end{bmatrix}. \quad (13)$$

Hardware realisation of (12) (see Fig. 3*c*) is the process of matrix multiplication. The matrices are stored in the FPGA distributed RAMs which are basically flip-flops distributed through the FPGA chip. The four inner product computations are done in parallel. After the coefficients are determined, the three roots of the cubic equation can be found. One of the roots represents the zero-crossing point. The algorithm for calculating the roots of the cubic equation in hardware involves computing the eigenvalues of companion matrix of the coefficient polynomial.

For the polynomial in (11), the companion matrix is calculated as

$$\boldsymbol{B} = \begin{bmatrix} -\frac{b}{a} & -\frac{c}{a} & -\frac{d}{a} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \quad (14)$$

Fig. 3*c* gives the hardware design of companion matrix calculation. The eigenvalue calculation relies on an iterative QR decomposition. The iteration is initiated by decomposing the previously calculated $\boldsymbol{B}$ matrix into an orthogonal $\boldsymbol{Q}$ matrix and upper triangular $\boldsymbol{R}$ matrix,

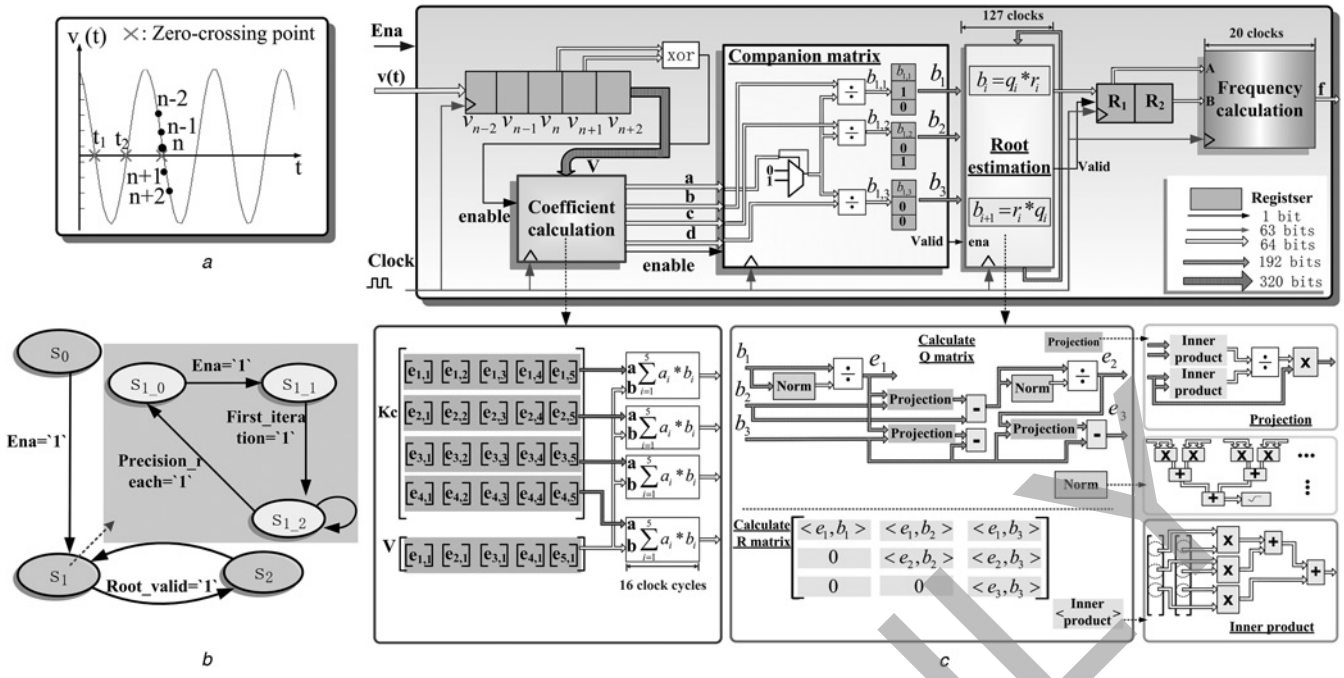

Fig. 3 *Frequency estimation core using the zero-crossing method*

a Principle of operation
b State machine and
c Hardware implementation

$B_0 = Q_0 R_0$. The Gram–Schmidt process [34] is used here. Then, multiplying the two matrices in reverse order leads to a new matrix $\boldsymbol{B}_1 = R_0 Q_0$. Repeating the process we can obtain the $(n+1)$ th matrix $\boldsymbol{B}_{n+1} = R_n Q_n$. After a certain number of iterations (12 in this case), the matrix converges to an upper triangular matrix whose diagonal entries are the eigenvalues of the matrix B. The whole iteration process is controlled by a FSM shown in Fig. 3*b* from S_{1-0} to S_{1-2}. When $ena = '1'$, the input of first iteration in $\boldsymbol{S}_{1-1}$ is the companion matrix. Then in S_{1-2} the output of the present iteration serves as the next iteration's input until required precision is reached. The outer FSM from S_0 to S_2 serves to record the time interval between successive zero-crossing points by using counters.

The Gram–Schmidt process for QR decomposition is described as follows with hardware design also given in Fig. 3*c*. Consider matrix $\boldsymbol{B}$ in (14) with three columns, $\boldsymbol{B} = [b_1, b_2, b_3]$. The $\boldsymbol{Q}$ matrix can be obtained by the following

$$\begin{aligned} e_1 &= \frac{u_1}{\| \boldsymbol{u}_1 \|}, \quad u_1 = b_1, \\ e_2 &= \frac{u_2}{\| u_2 \|}, \quad u_2 = b_2 - \text{proj}_{e_1} b_2, \\ e_3 &= \frac{u_3}{\| u_3 \|}, \quad u_3 = b_3 - \text{proj}_{e_1} b_3 - \text{proj}_{e_2} b_3, \end{aligned} \tag{15}$$

where $\|\boldsymbol{u}_1\|$ stands for the norm operation of a vector and proj stands for the projection operation

$$\text{proj}_e b = \frac{\text{inner product}\langle e, b\rangle}{\text{inner product}\langle e, e\rangle} e. \tag{16}$$

Q_1 equals to $[e_1, e_2, e_3]$. With the results from (15), $\boldsymbol{R}$ can be calculated using

$$\boldsymbol{R} = \begin{bmatrix} \langle e_1, b_1\rangle & \langle e_1, b_2\rangle & \langle e_1, b_3\rangle \\ 0 & \langle e_2, b_2\rangle & \langle e_2, b_3\rangle \\ 0 & 0 & \langle e_3, b_3\rangle \end{bmatrix}. \tag{17}$$

After the eigenvalues are found, the roots of the cubic equation are determined. Two successive roots are stored in the register R_1 and R_2 for the frequency calculation by (10).

2.2 Multi-function protection relay processing core

This section gives the hardware emulation details of the multi-function relay processing core.

(i) *Fault detection processing core:* The fault detection processing core detects the initiation of the fault and triggers the subsequent

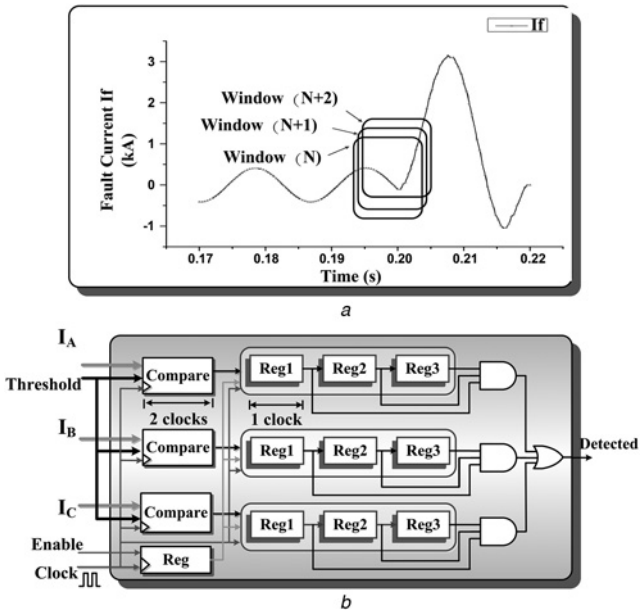

Fig. 4 *Fault detection processing core*

a Principle of operation
b Hardware implementation

protection processes. This detection algorithm uses the DFT current amplitude results to decide if a fault occurred. As the DFT window moves in real-time, the calculated amplitude is sensed by this module. The strategy is shown in Fig. 4*a*. If the amplitude of any phase current exceeds a certain threshold value three times consecutively, it indicates that the protected element is exposed to an abnormal situation, and a fault detection signal is sent out.

The hardware emulation is shown in Fig. 4*b*. The DFT amplitude results for each phase enter the processing core and are compared with the corresponding threshold values. The enable signal indicates validation of each DFT computation result which permits the comparison and registration process. The register after the enable signal is need for synchronisation, while the three single-bit registers record the comparison results. A fault detection signal is generated when all three registers record a '1'.

(ii) *Directional overcurrent protection processing core:* Overcurrent protection is a simple and economical way to protect power system equipment. It is generally used for low cost sub-transmission lines, distribution circuits, and industrial systems [35, 36]. The overcurrent protection operates when the current exceeds the pickup value. In order to provide better selectivity, it is also necessary to determine the fault current direction to avoid unintended outages. A directional element is therefore added. It uses the negative-sequence impedance (Z_2) seen by the relay to get the direction information. Under faulted conditions, an approximate 180° of impedance angle change can be observed [37, 38] for two directional faults as shown in Fig. 5(a {1}). For a reverse fault, $Z_2 = Z_{2sB} + Z_{2line}$, where Z_{2sB} is system-B's negative sequence impedance and Z_{2line} is the apparent negative sequence line impedance. The process to determine the direction is to compare the angle of the measured sequence impedance with the maximum torque angle. The forward and reverse action areas of two direction faults are clearly seen in Fig. 5*a* {1}. Although this method requires the source impedance, other methods for directional overcurrent protection that use voltage as a polarised reference may also be emulated.

From the overall FPGA design (Fig. 5*a*{2}) for the directional overcurrent protection we can see, only when the demands for both the forward direction element and the overcurrent element are met, the relay will initiate final tripping. The three-step zone overcurrent protection design is shown in Fig. 5*a*{3}. Each protection zone has its own setting and delay timer. This module can also be configured for the voltage protection which will be discussed later. After a fault is detected, any timer's time-up signal can activate the trip signal.

(iii) *Over/under voltage protection processing core:* The maintenance of normal and stable voltage magnitude is important for the health of any power system. Voltage deviations deteriorate the power quality and life-expectancy of electrical infrastructure. Over/under voltage elements in a multi-function relay can provide the phase-to-ground and phase-to-phase protection by sensing voltage deviations. The phase-to-ground undervoltage elements operate when the phase voltage is smaller than the low-setting threshold while the overvoltage elements trips when voltage is larger than the

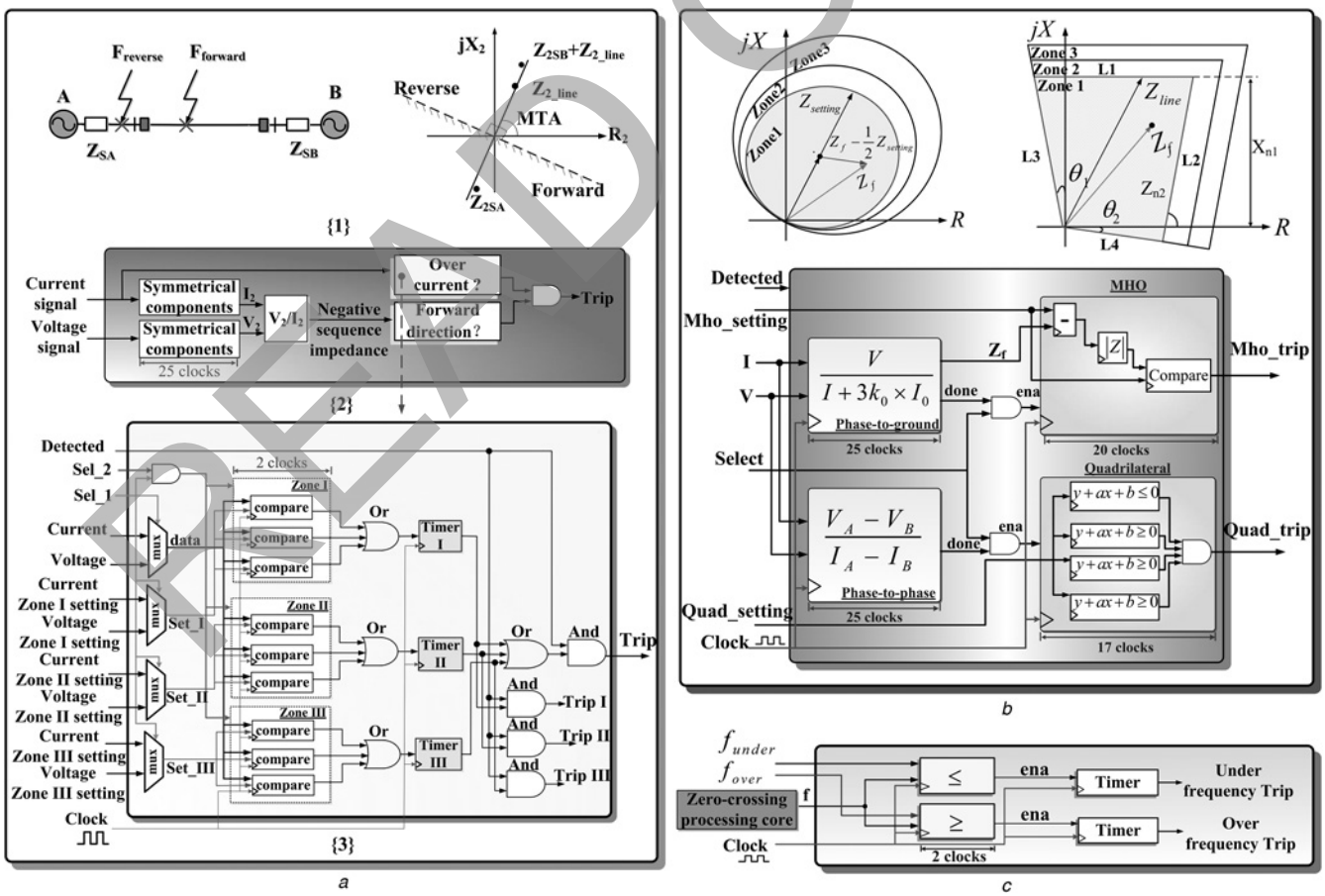

Fig. 5 *Protective relay processing core*

a Directional overcurrent protection
b Distance protection with mho and quadrilateral characteristics
c Frequency protection

IET Gener. Transm. Distrib., 2016, Vol. 10, Iss. 10, pp. 2295–2305

high-setting threshold. The phase-to-phase element works under the same mechanism. If any one of the three voltages rise above or drop below a certain set value, a trip signal is generated. This element also comes with three-step zone protection scheme similar to the overcurrent protection. In Fig. 5*a*{3}, after configuring the input signal to voltage signal and the choosing of under or over options, the hardware module can be used for voltage protection.

(iv) *Distance protection processing core:* The operation of distance protection is based on the measurement and evaluation of the short-circuit impedance which is proportional to the distance between the relay and the fault location. The two types of impedance calculation in the distance protection processing core are phase-to-ground (Z_g) and phase-to-phase (Z_p) impedance given as follows

$$Z_g = V_{p-g}/(I_{p-g} + 3k_0 * I_0),$$
$$Z_p = (V_{phase1} - V_{phase2})/(I_{phase1} - I_{phase2}), \quad (18)$$

where I_0 is zero sequence current calculated from $(I_A + I_B + I_C)/3$; $k_0 = (Z_0 - Z_1)/3Z_1$. Z_0 and Z_1 are the zero- and positive-sequence line impedances from the relay location to protection zone, respectively. Hardware realisation of impedance calculation involves straightforward complex number computations. The calculated impedance is then processed by a mho or quadrilateral characteristic [5, 39, 40] relay element to decide in-zone tripping. Fig. 5*b* gives the two distance characteristics with three protection zones. If the calculated impedance locates within the tripping area which is inside the characteristic shape, a trip signal is generated. The application of zone protection can be achieved by setting different zone areas of the characteristic shapes.

Fig. 5*b* also gives the hardware implementation for the distance protection processing core. The voltage and current DFT calculation results enter the module at first. After the apparent impedance calculation, mho or quadrilateral characteristic can be selected for tripping decision. For the mho characteristic, if the fault impedance is inside the circle, the vector $\boldsymbol{Z}_f - 1/2Z_{setting}$ has a magnitude less than $1/2Z_{setting}$. For the quadrilateral characteristic, the trip area is shaped by four straight lines. L1 is the reactance relay setting (X_{n1}) and L2 is the angle-impedance relay setting (Z_{n2}). L3 and L4 are the directional relay with angle settings (θ_1, θ_2). The apparent impedance coordinates are then substituted into the four line equations to determine whether they are in the tripping region. In quadrilateral zone, the calculated impedance is compared with the four sides of the shape by checking the relative position of impedance point and the line. It should meet the conditions of locating below L1, above L4, on the left of L2 and right of L3 to get an in-the-zone decision.

(v) *Under/over frequency protection processing core:* Frequency is another important quantity to monitor the well-being of a power system. It reflects the balance between the generation and system load and losses. During sudden generation-load imbalances, frequency can deviate from its nominal value. The frequency protection processing core senses the abnormal frequency deviation to make the trip decision. It provides under/over-frequency protection with or without a time delay. The functional architecture of frequency protection processing core is shown in Fig. 5*c*. Using the output of zero-crossing frequency calculation (see Fig. 3) results, the under/over-frequency elements compare the signal with the pre-specified low-setting or high-setting to determine the trip logic.

3 Test hardware setup and real-time results

The complete hardware multi-function system design was targeted to the Xilinx® Virtex-7 XC7VX485T FPGA. The design was implemented wholly in VHDL with 32 (64)-bit floating-point precision. The Xilinx ISE® environment was used to synthesize and map the design to the physical device. The highest frequency

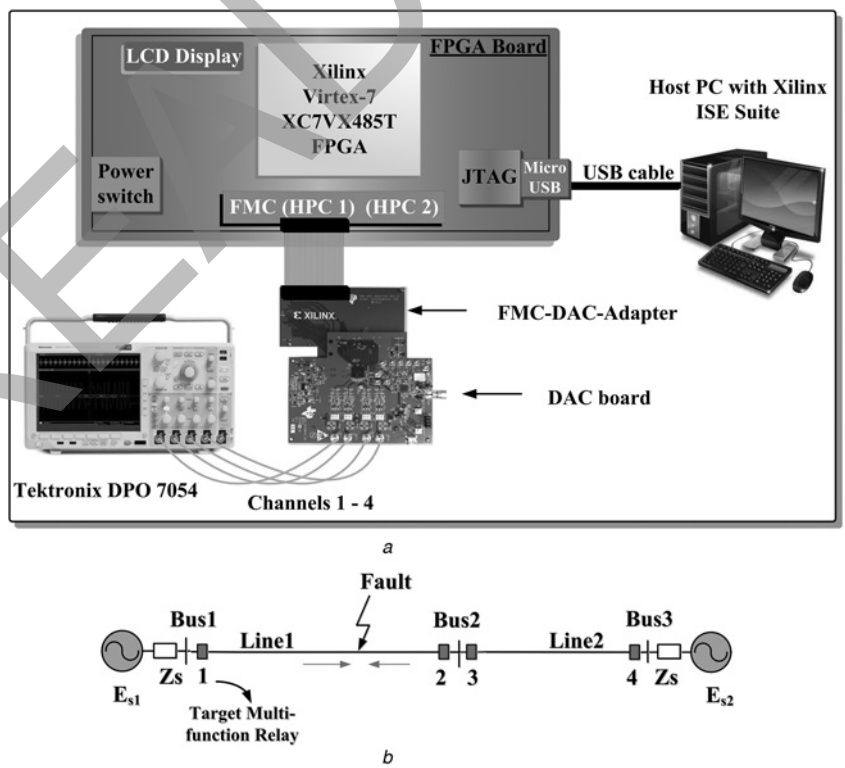

Fig. 6 *Test set-up for target multi-function protection system*

a Experimental set-up
b Test Case I power system

Table 1 FPGA hardware resource consumption and latencies of the multi-function protection system

Processing cores	Slice registers [607,200 total]	Slice LUTs [303,600 total]	DSP blocks [2800 total]	BRAM 36E1 [1030 total]	Latency in clock cycles & μs	Operating trip time
directional overcurrent	4614 (1%)	1669 (0.6%)	32 (1%)	0	47 (1.175 μs)	15.10 ms
distance relay	64,118 (10%)	28,576 (9%)	53 (2%)	13	209 (5.225 μs)	24.75 ms
over/under voltage	2520 (0.4%)	1417 (0.5%)	0 (0%)	0	7 (0.175 μs)	13.33 ms
over/under frequency	34,983 (6%)	88,042 (29%)	510 (18%)	0	1566 (39.15 μs)	2 ms
total usage	106,235 (17%)	119,704 (39%)	595 (21%)	13 (1%)	n/a	n/a

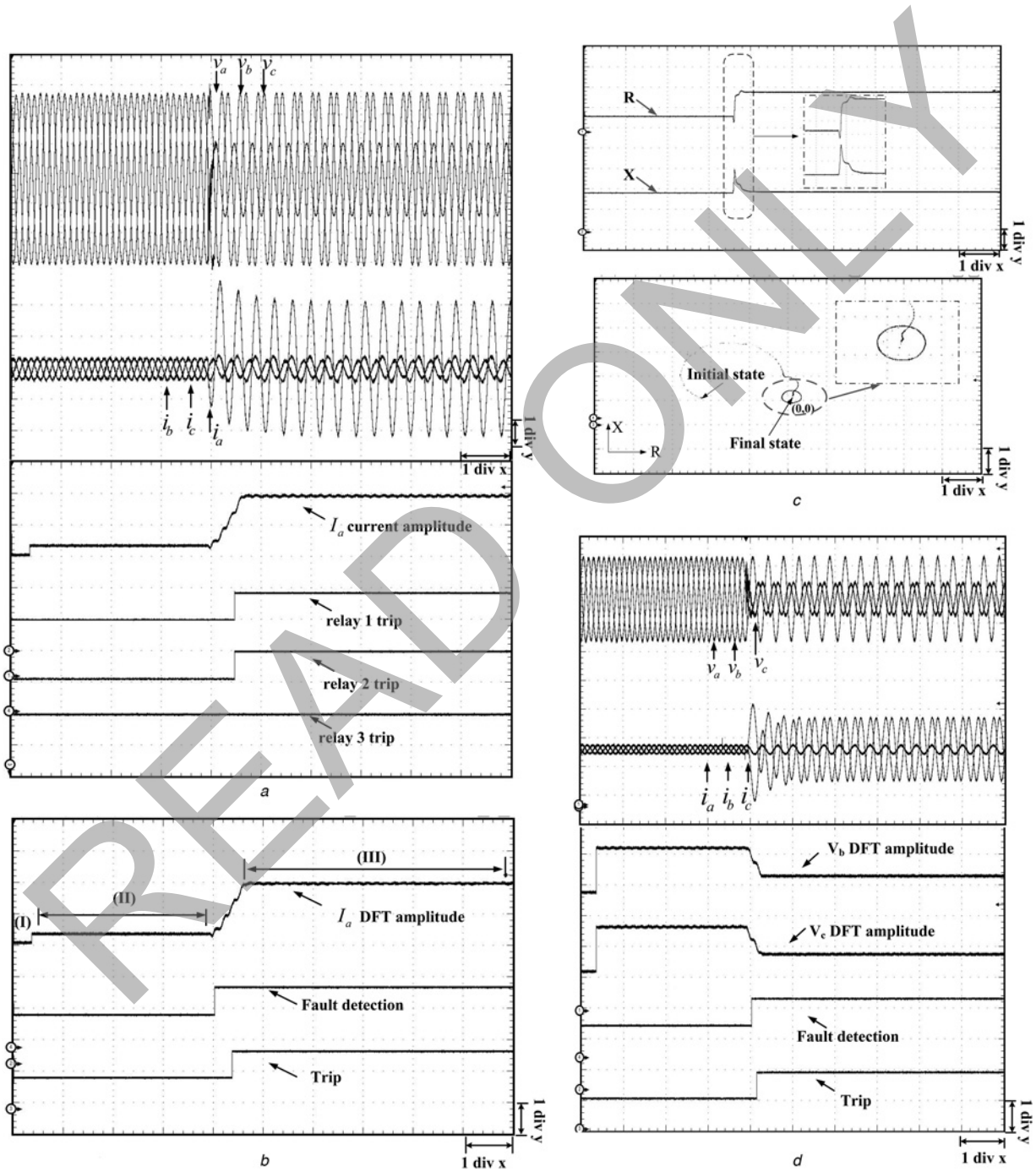

Fig. 7 *Real-time test results for Case I*

a Voltage and current waveforms of a phase-a-to-ground fault, and directional overcurrent protection signals
b DFT, detection and trip signals for distance protection
c Impedance captures of distance protection
d Voltage and current waveforms of a phase-*b*-to-*c* fault, and voltage protection signals. Scale [time: 1 div $x = 46.5$ ms; voltage: 1 div $v = 70$ kV; current: 1 div $i = 0.97$ kA; 1 div r or $x = 500$ Ω]

the whole design can run at is 40 MHz due to the long critical path of the frequency processing core. The Coregen provided by ISE was used to generate the basic 32-bit and 64-bit floating point operations such as adder and subtracter which are further employed to implement the other complex functional units. While the maximum frequency of the generated operations can reach up to 500 MHz, for the multi-function relay design the maximum operating clock frequency is 40 MHz. This is mainly due to the massive operations involved in each clock cycle which results in high complexity of the hardware implementation which increases the latency between flip-flops. To observe output waveforms, a 16-bit four-channel DAC board was connected to the FPGA board. The Tektronix® 500 MHz DPO7054 4-channel oscilloscope is used to capture the output of the DACs as shown in Fig. 6. First, off-line simulation of the test power systems was conducted using the PSCAD/EMTDC® software. Then the generated floating point fault data was transmitted through the PCIe bus, and stored in the FPGA block RAM. After the fault data buffering, the various protection functions are realised in real time. This type of testing is quite common in the industry and is termed as real-time playback which is used when a newly designed relay is tested under open-loop conditions i.e. the relays trip outputs are not sent back to the modelled power system to open/close a circuit breaker. The hardware resource usage and overall latency of each

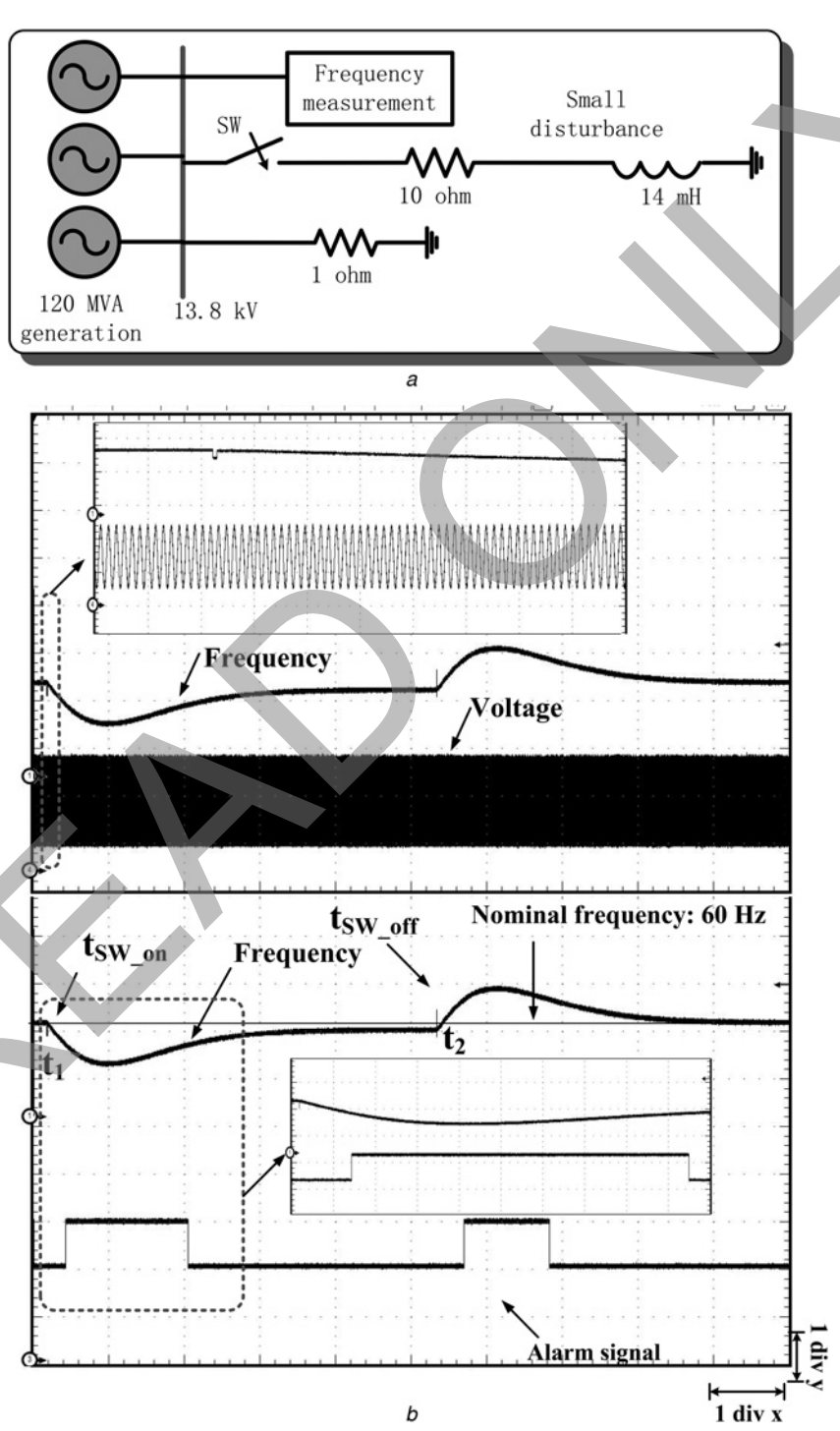

Fig. 8 *Test power system and waveform results*

a Test power system
b Results for Case II (frequency protection). Scale: [time: 1 div $x = 6.5$ s; voltage: 1 div $v = 7.3$ kV; frequency: 1 div $f = 1.0$ Hz]

protection function (the signal processing and detection functions are included in the respective protective function designs) is summarised in Table 1. The frequency protection consumes the most hardware resources due to the large floating point and iteration computational burden, while the voltage protection consumes the least resource. Overall, the hardware consumption remains at a reasonable level. The hardware resource count for the frequency protection function includes the 'zero-crossing processing core' which comprises of the iterative root calculation process. Due to this the overall Slice LUT consumption and latency reported in Table 1 for the frequency protection are the highest at 29% and 1566 clocks, respectively. Table 1 also gives the trip operating time for each relay function.

3.1 Test case I

This case study shows the test results of directional overcurrent protection, voltage protection and distance protection. The test power system can be seen in Fig. 6*b*, and consists of two synchronous generators and two transmission lines (1 and 2) which are 110 and 100 km long, respectively. The parameters of this system are given below.

Source parameters: $Z_s = 9.2 + j52\ \Omega$, $E_{s1} = 230\angle 0^\circ$ kV, $E_{s2} = 230\angle 20^\circ$ kV, 60 Hz.

Transmission line impedance (Ω/km): $Z_0 = 0.363 + j1.326$, $Z_1 = 0.0357 + j0.5078$, $Z_2 = 0.0357 + j0.5078$.

The studied fault for overcurrent and distance protection is a phase-a-to-ground fault. The oscilloscope captures the current waveform (in Fig. 7*a*) at location 1 from $t = 0.1$ s to $t = 0.565$ s. The fault happens on Line 1 at $t = 0.28$ s, 50 km away from the relay location 1. Fig. 7*a* gives the DFT and no-time-delay trip signals for the directional overcurrent protection case study. The relays 1 and 2 see the overcurrent fault in the forward direction and send the trip signals, while relay 3 sees the reverse direction and remains inactive. Due to the coordination between relay 4 and 2, the trip signal of relay 4 is blocked since relay at 1 and 2 on Line 1 successfully trip.

For distance protection, the real-time results can be observed in Figs. 7*b* and *c*. Fig. 7*c* gives the real-time trajectory of the apparent impedance seen by the distance protection relay. The mho zone 1 reach is 80% of the line length, which is $(3.14 + j44.69)\ \Omega$. By coupling the time-based resistance R and reactance X coordinates, the real-time impedance locus is obtained. Before the fault at the initial state $(-628.3 + j1.7)\ \Omega$, the impedance is outside the mho circle. After the fault occurs, the impedance enters into the circle and converges to its final state of $(5.85 + j24.33)\ \Omega$, which is 56% of the zone reach and 44% of the line length. In the DFT processing results of the faulted phase shown in Fig. 7*b*, the amplitude of Segment I stays zero until one full cycle of fault data collection is finished. During Segment II, the I_a amplitude value stays constant under normal operating condition until at $t = 0.28$ s when the ground fault occurs. Accordingly from the DFT calculation, the amplitude of I_a starts to increase after a small transient oscillation. The fault detection module senses the abnormal current increase and operates to send the fault detected signal. The smooth Segment III indicates the effectiveness of the dc offset removal of fault current in the faulted phase. The final trip signal gives at $t = 0.303$ s.

A *b*-*c* double-phase fault applied to the same power system is studied to verify the voltage protection function. Fig. 7*d* shows the three-phase voltage, current waveforms and the DFT amplitude of faulted phase voltage. The fault detection and tripping signals are also depicted in this figure. After the fault happens, the faulted phases *b* and *c* voltage amplitudes drop dramatically. The undervoltage element senses this decrease in voltage and issues a trip signal.

3.2 Test case II

The test power system shown in Fig. 8*a* is used to test the performance of frequency protection. In this case study, three generators are connected to a common bus that supplies the loads. The synchronous generators are each rated at 120 MVA, 13.8 kV, 60 Hz whose main parameters are the following:

$X_d = 1.014$ p.u., $X'_d = 0.314$ p.u., $X''_d = 0.280$ p.u. $X_q = 0.770$ p.u., $X'_q = 0.228$ p.u., $X''_q = 0.375$ p.u.

$T'_{do} = 6.55$ s, $T''_{do} = 0.039$ s, $T_a = 0.278$ s, $R_a = 0.0025$ p.u., $R_f = 0.00043$ p.u., Inertia constant $H = 3.117$ s.

IEEE Type SCRX Exciter with time constant $T_E = 0.02$ s, and IEEE Type 2 hydro governor with time constant $T_G = 0.05$ s.

This test system was simulated in PSCAD/EMTDC® to generate the voltage data which was later fed to the hardware multi-function relay. The waveforms in Fig. 8*b* trace the time period of $t = 0$ s to $t = 60.5$ s. The real-time tracking of the system frequency has a precision of ±0.01 Hz. One branch that has a R-L load can emulate the frequency disturbance source. When the load is suddenly connected at $t_1 = 1$ s, there is a consequent sudden drawing of active power from the generators, which causes the frequency to drop as shown in first switching event in Fig. 8*b*. The system frequency goes down gradually from 60 to 59.5 Hz over a time period of 5.2 s which is 313 cycles. As suggested in [41], abnormal frequency can be operated continuously between 59.5 to 60.5 Hz. In this case, when frequency reaches to 59.5 Hz, a no-time-delay alarm signal is given. When the frequency gets back to above 59.5 Hz after about 10 s, the alarm signal is cleared. At t_2 when the disturbance branch is switched off, this introduces an over-frequency condition. The alarm signal is given when the frequency goes over 60.5 Hz.

4 Conclusion

This paper proposed a real-time multi-function protection system design in hardware. FPGA technology provides a flexible and efficient way to apply complex protection functions in power systems. The total hardware usage of the multi-function protection system is 39% which is at a reasonably low level to reduce power consumption and increase life cycle of the device. All the relay functions are emulated in VHDL using 32 (64)-bit floating point precision ensuring high fidelity. The targeted device is the Xilinx® Virtex-7 FPGA. Parallelism and pipelining was fully exploited in both the signal processing hardware emulation blocks as well as the protection function blocks. Essentially, the designed multi-function relay has similar capability in terms of operating time as that of a commercial relay; however, since it is entirely hardware, it has the advantages of lower computational latency, higher bandwidth, and larger capacity. These advantages would enable expanding the functions of this relay to include high frequency transient operation, intelligent functions, adaptation etc., for smart grid systems in the future. Depending on the size of the system and the real-time computation requirement, various algorithms within this approach that are most time- and resource-consuming can be further optimised to lower the operating time and hardware consumption. Functions such as high-speed communications, advanced control, and smart metering could also be furnished to the existing system. Other auxiliary functions, for example, a local HMI for setting, data downloading and printing, could also be added.

5 Acknowledgment

This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC).

6 References

1 Horowitz, S.H., Phadke, A.G.: 'Power system relaying' (Research Studies Press Ltd. and John Wiley & Sons, Ltd., 2008, 3rd edn.)
2 Thorp, J.S., Phadke, A.G.: 'Protecting power systems in the post-restructuring era', *IEEE Comput. Appl. Power*, 1999, **12**, (1), pp. 33–37

3 Russell Mason, C.: 'The art and science of protective relaying' (GE Digital Energy, Wiley, 1956), pp. 1–357
4 Sidhu, T.S., Burnworth, J., Darlington, A., *et al.*: 'Bibliography of relay literature, 2007 IEEE Committee Report', *IEEE Trans. Power Deliv.*, 2010, **25**, (1), pp. 88–101
5 ALSTOM: 'Network Protection & Automation Guide', 2012
6 Phadke, A.G., Hlibka, T., Adamiak, M.G., *et al.*: 'A microcomputer based ultra-high-speed distance relay: field tests', *IEEE Trans. Power Appar. Syst.*, 1981, **100**, (4), pp. 2026–2036
7 Horp, J.S., Phadke, A.G.: 'A microprocessor based three-phase transformer differential relay', *IEEE Trans. Power Appar. Syst.*, 1982, **101**, (2), pp. 426–432
8 Kezunovic, M., Russell, B.D.: 'Microprocessor applications to substation control and protection', *IEEE Comput. Appl. Power*, 1988, **1**, (4), pp. 16–20
9 Manzoul, M.A.: 'Multiple overcurrent relays using a single microprocessor', *IEEE Trans. Ind. Electr.*, 1990, **37**, (4), pp. 307–309
10 Kezunovic, M., Kasztenny, B.: 'Design optimization and performance evaluation of the relaying algorithms, relays and protective systems using advanced testing tools', *IEEE Trans. Power Deliv.*, 2000, **15**, (4), pp. 1129–1135
11 Mozina, C., Young, M.: 'Multifunction digital relay commissioning and maintenance testing', *IEEE Ind. Appl. Mag.*, 2005, **11**, (5), pp. 50–58
12 Working Group of the Relaying Practices Subcommittee: 'Understanding microprocessor-based technology applied to relaying'. Power System Relaying Committee, Report of Working Group I-01, January 2009
13 *IEC TC57: Communication Networks and Systems in Substations – Part 1: Introduction and Overview*, IEC TR 61850-1, April 2003
14 Kanabar, M.G., Sidhu, T.S., Zadeh, M.R.D.: 'Laboratory investigation of IEC 61850-9-2-based busbar and distance relaying with corrective measure for sampled value loss/delay', *IEEE Trans. Power Deliv.*, 2011, **26**, (4), pp. 2587–2595
15 Ransom, D.L.: 'Upgrading relay protection? -- Be prepared'. IEEE/IAS 49th Industrial and Commercial Power Systems Technical Conf (I& CPS), May 2013, vol. 5, no. 3, pp. 1–8
16 Adamiak, M.G., Apostolov, A.P., Begovic, M.M., *et al.*: 'Wide area protection-technology and infrastructures', *IEEE Trans. Power Deliv.*, 2006, **21**, (2), pp. 601–609
17 Li, F., Qiao, W., Sun, H., *et al.*: 'Smart transmission grid: vision and framework', *IEEE Trans. Smart Grid*, 2010, **1**, (2), pp. 168–177
18 Qi, H., Wang, X., Tolbert, L.M., *et al.*: 'A resilient real-time system design for a secure and reconfigurable power grid', *IEEE Trans. Smart Grid*, 2011, **2**, (4), pp. 770–781
19 Davies, S.: 'Grid gets the smarts [Power Smart Grid]', *Eng. Technol.*, 2013, **7**, (12), pp. 42–45
20 Liu, S., Hou, Y., Liu, C.-C., *et al.*: 'The healing touch: tools and challenges for smart grid restoration', *IEEE Power Energy Mag.*, 2014, **12**, (1), pp. 54–63
21 Chen, Y., Dinavahi, V.: 'FPGA-based real-time EMTP', *IEEE Trans. Power Deliv.*, 2009, **24**, (2), pp. 892–902
22 Chen, Y., Dinavahi, V.: 'Multi-FPGA digital hardware design for detailed large-scale real-time electromagnetic transient simulation of power systems', *IET Gener. Transm. Distrib.*, 2013, **7**, (5), pp. 451–463
23 Liu, J., Dinavahi, V.: 'A real-time nonlinear hysteretic transformer transient model on FPGA', *IEEE Trans. Ind. Electron.*, 2014, **61**, (7), pp. 3587–3597
24 Valsan, S.P., Swarup, K.S.: 'Protective relaying for power transformers using field programmable gate array', *IET Electr. Power Appl.*, 2008, **2**, (2), pp. 135–143
25 Liu, X., Osman, A.H., Malik, O.P.: 'Real-time implementation of a hybrid protection scheme for bipolar HVDC line using FPGA', *IEEE Trans. Power Deliv.*, 2011, **26**, (1), pp. 101–108
26 Manzoul, M.A.: 'Multi-function protective relay on FPGA', *Microelectron. Reliab.*, 1998, **38**, (12), pp. 1963–1968
27 Wang, Y., Dinavahi, V.: 'Real-time distance protective relay on FPGA'. IEEE Power and Energy Society General Meeting, July 2013, pp. 1–5
28 Volder, J.E.: 'The CORDIC trigonometric computing technique', *IRE Trans. Electr. Comput*, 1959, **EC-8**, (3), pp. 330–334
29 Walther, J.S.: 'The story of unified CORDIC', *J. VLSI Signal Proc. Syst. Signal Image Video Tech.*, 2000, **25-2**, pp. 107–112
30 Gu, J.C., Yu, S.L.: 'Removal of DC offset in current and voltage signals using a novel Fourier filter algorithm', *IEEE Trans. Power Deliv.*, 2000, **15**, (1), pp. 73–79
31 Glover, J.D., Sarma, M.S., Overbye, T.J.: 'Power system analysis and design' (Cengage Learning, 2012)
32 Begovic, M.M., Djuric, P.M., Dunlap, S., *et al.*: 'Frequency tracking in power networks in the presence of harmonics', *IEEE Trans. Power Deliv.*, 1993, **8**, (2), pp. 480–486
33 Xue, S.Y., Yang, S.X.: 'Accurate and fast frequency tracking for power system signals'. IEEE Int. Conf. on Systems, Man and Cybernetics, October 2007, pp. 2754–2759
34 Golub, G.H., Van Loan, C.F.: 'Matrix computations' (The Johns Hopkins University Press, 1996, 3rd edn.)
35 GE Digital Energy: 'Line Protection with Overcurrent Relays'. Available at http://www.gedigitalenergy.com/multilin/notes/artsci/art13.pdf
36 Blackburn, J.L., Domin, T.J.: 'Protective relaying principles and applications' (Taylor and Francis Group, LLC., 2006, 3rd edn.)
37 Horak, J.: 'Directional overcurrent relaying (67) concepts'. 59th Annual Conf. for Protective Relay Engineers, 2006, p. 13
38 Zimmerman, K., Costello, D.: 'Fundamentals and improvements for directional relays'. 2010 63rd Annual Conf. for Protective Relay Engineers, March–April 2010, pp. 1–12
39 Anderson, P.M.: 'Power system protection' (McGraw-Hill, IEEE Press, New York, 1999)
40 Schweitzer III, E.O., Roberts, J.: 'Distance relay element design' (Schweitzer Engineering Laboratories, Inc, 1993)
41 IEEE Guide for AC Generator Protection, *IEEE Std C37.102-2006*, August 21, 2013