FPGA-Based Real-Time EMTP

Yuan Chen, *Student Member, IEEE*, and Venkata Dinavahi, *Member, IEEE*

***Abstract*—Real-time transient simulation of large transmission networks requires significant computational power. This paper proposes a field-programmable gate array (FPGA)-based real-time electromagnetic transient simulator. Taking advantage of the inherent parallel architecture, high density, and high clock speed, the real-time Electromagnetic Transients Program (EMTP) is implemented on a single FPGA chip. It is based on a paralleled algorithm that is deeply pipelined, and uses a floating-point arithmetic calculation to achieve high accuracy for simulating fast electromagnetic transients. To validate the new simulator, a sample system with 15 transmission lines using full frequency-dependent modeling is simulated in real time. A timestep of 12 μs has been achieved based on a 12.5-ns clock period with high data throughput. The captured real-time oscilloscope results demonstrate excellent accuracy of the simulator in comparison to the offline simulation of the original system in the Alternative Transients Program version of EMTP.**

***Index Terms*—Electromagnetic transient analysis, field-programmable gate arrays (FPGAs), parallel algorithms, real-time systems.**

I. Introduction

Electromagnetic transient simulation plays an important role in the planning, design, and operation of modern power transmission systems. Simulation can be accomplished in either offline or real-time mode. For offline transient simulation, the electromagnetic transients program (EMTP) [1], [2] is widely accepted and well used. There are several EMTP-type software packages available commercially that offer a wide variety of modeling capability. Real-time simulation is desired for the testing of manufactured protection and control equipment in a hardware-in-the-loop configuration. Real-time simulation can be based on an analog simulator such as a transient network analyzer (TNA) or on a real-time digital simulator. The analog TNA uses scaled down models of power system components. For example, the frequency response of a long transmission line is approximated by cascading several analog π sections representing shorter line segments. This can be quite cumbersome due to the high cost, large space, and long setup and changeover times, thereby limiting the TNA's application to small and medium sized networks. Over the last two decades, real-time digital simulators have achieved a high level of sophistication to be able to supplant analog TNAs. Nevertheless, there is an ever increasing pressure to accommodate large network sizes, and to reduce the simulation timestep to accurately capture high frequency transients, which essentially translates to the requirement of higher computational power for the simulator hardware.

Historically, real-time digital simulator hardware was built around the general purpose processor (GPP) based on the RISC or x86 architectures or the digital signal processor (DSP) [3]–[10]. The computational engine that performed the EMTP calculations was basically a sequential software program. Since the computational capability required by real-time EMTP is beyond most single general purpose computers due to the increasing complexity of power systems being modeled, parallel computer architectures became necessary. This was accomplished by using multi-DSP [4], [6], [7], multi-RISC [8], or PC-cluster architectures [9], [10]. For instance, [7] proposed a method using two DSPs for each transmission line. Each DSP processes the quantities for one end of the line (sending end and receiving end) simultaneously. For larger network sizes, the system was partitioned into smaller subsystems. Each processor was made responsible for calculation of quantities within a single subsystem. Timestep synchronization, precise communication and data exchange are fundamental in this multiprocessor architecture. Although, theoretically clusters with large number of processors can be constructed, the main computation bottleneck in such an architecture is the inter-processor communication latency. Another approach to accommodate large transmission networks while maintaining sufficient accuracy for real-time simulation [11] is to use an efficient wide-band frequency-dependent network equivalent (FDNE). However, deriving an accurate FDNE model is a finely-tuned art requiring significant setup time due to the various optimizations and fittings required to obtain a low-order equivalent. In contrast to the full-scale representation, there is also a degree of loss of one-to-one mapping between the original system physical layout and the simulator architecture, since the signals inside the equivalenced system are no longer accessible.

The ultimate performance of any real-time simulator depends to a large extent on the capabilities of the underlying hardware. Field-programmable gate arrays (FPGAs) offer a viable alternative for speeding up the real-time simulator without sacrificing accuracy or incurring excessive communication latency. The FPGA is comprised of a variety of logic building blocks. Each block consists of programmable look-up tables and registers, and the interconnections among these blocks can be programmed through a hardware description language such as VHDL. Admittedly, programming in VHDL is more complex than programming in a high-level structured language such as C or C++; however, unlike the GPP or DSP, which is a sequential device, the FPGA allows for true parallel processing, supporting multiple simultaneous threads of execution. Moreover, the FPGA is a fully user configurable device, i.e., it can

Manuscript received November 26, 2007; revised January 07, 2008. First published May 12, 2008; current version published March 25, 2009. This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC). Paper no. TPWRD-00739-2007.

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: yuanchen@ece.ualberta.ca; dinavahi@ece.ualberta.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TPWRD.2008.923392

be configured according to any specific application. It is this inherent hardwired parallelism and configurability that makes the FPGA most suitable for the EMTP hardware implementation. Although [12] presents a lossless line model with shaping to account for distortion of traveling waves for FPGA-based simulation, it does not show the actual implementation details nor the real-time results.

In this paper, we propose a FPGA-based real-time EMTP simulator. The central component of proposed simulator is the full frequency-dependent line model. The auxiliary models include the linear lumped elements, switches, and sources. The simulator is based on a parallelled EMTP algorithm with deeply pipelined computations using floating point number representation. The simulator accepts the standard ATP data input. The size of the transmission network that can be simulated in real time depends the available FPGA resources. The preliminaries on FPGA architecture and operation are described in Section II. Section III explains the skeletal framework of the FPGA-based real-time EMTP, and Section IV presents the implementation and operation details of the simulator. A real-time transient simulation case study is presented in Section V, where the real-time simulation results captured on an oscilloscope are compared with the ATP offline simulation results. Conclusions are given in Section VI.

II. FPGA Primer

The FPGA is generally composed of three types of configurable logic components [13]: configurable logic block (CLB), configurable input/output block (IOB), and programmable interconnect as shown in Fig. 1(a). The CLB provides the functional elements for constructing the user defined logic. Each CLB is composed of several basic logic building blocks whose number depends on the FPGA fabrication process. The basic logic building block in the FPGA, called by different names depending on the FPGA vendor, is a logic element (LE) (Altera®) or a logic cell (LC) (Xilinx®). It usually consists of a 4-input look-up table (LUT), a D-type flip-flop, and a multiplexer as shown in Fig. 1(b). The LUT can quickly generate any combinatorial function of four input variables. The flip-flop is used to register a sequential function. The user configurable IOB provides the interface between external package pins of the FPGA and its internal logic. Each IOB controls one package pin and can be configured for input, output, or bidirectional signals. The interconnection sets up the routing paths to connect the CLB and IOB.

The FPGA design process begins by defining the user application using a hardware description language such as VHDL or Verilog. After the code is developed, automatic synthesis, place, and route tools such as Quartus (Altera) or ISE (Xilinx) are used to generate a binary configuration file called the bitstream. The process of downloading the bitstream into the FPGA from a host computer is called FPGA programming. The user defined FPGA is now ready for use.

The most attractive feature of the FPGA is its inherent parallel architecture. That is, the FPGA can be partitioned and configured into a large number of parallel-processing units which process their data simultaneously. A GPP or DSP, on the other hand, is a sequential device in which one or several instructions are executed sequentially in one or more clock cycles. Modern

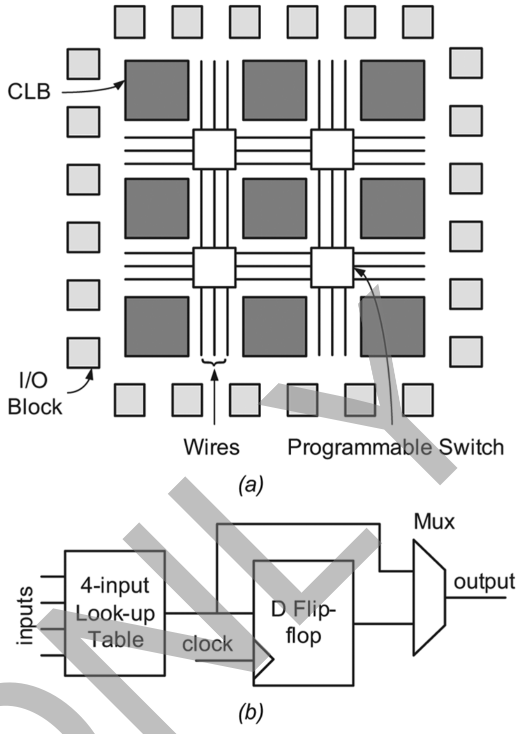


Fig. 1. (a) Generic architecture of FPGA and (b) a logic element.

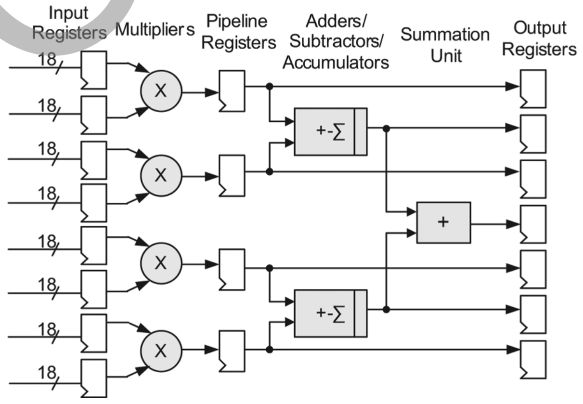


Fig. 2. Details of a *DSP Block* inside the FPGA.

FPGAs contain several blocks of memory which can be configured to be of many different types such as ROM, single-port RAM, and dual-port RAM with any data and address bus width. Thus, many independent memory banks can be accessed in parallel, whereas in a GPP or DSP, all memory and buses are shared. Currently available FPGAs also contain several *DSP Blocks* or *DSP Slices*. A *DSP Block* is a dedicated configurable math circuitry which provides the multiply, accumulate, and add/subtract functions for added flexibility and fast computation capability. Fig. 2 shows the *DSP Block* architecture.

In the paralleled architecture, the *pipelining* technique is another important strategy for improving hardware performance. In a pipeline scheme, a function is divided into n stages depending on the number of operations. Registers are then inserted between these stages to separate the operations. Thus, data can march through the operations at every clock. Although a pipeline has the cost of *latency* which is the input

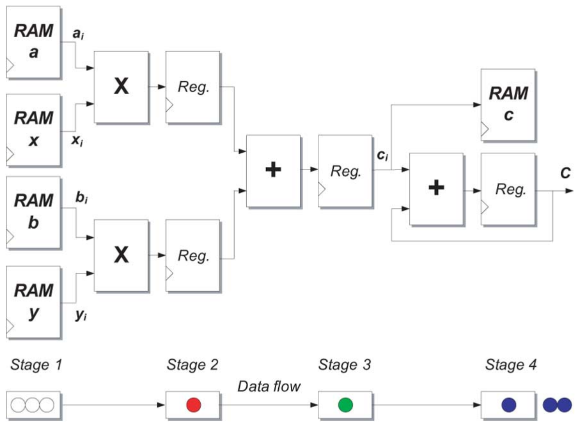


Fig. 3. Example of the convolution operation in the FPGA.

to output delay, the key merit of a pipeline is its computational *throughput*, which is defined as the number of operations that can be performed per unit of time. A good example to illustrate these features is the convolution function which is widely used in digital signal processing. A convolution defined by (1) can be hardware implemented in FPGA as shown in Fig. 3. In *Stage 1*, all input data are accessed from the RAMs simultaneously. The multiplications are carried out at *Stage 2* and the addition is done at *Stage 3*. The final summation is done at *Stage 4*. As shown in the figure, the data flows into the pipeline at every clock and is processed independently at each individual stage. In this case, the latency is 4 clock cycles while the throughput is 1 result per clock cycle. If the convolution operation is performed on a sequential GPP or a DSP, the latency and throughput would be 4 clock cycles and 1 result per 4 clock cycles, respectively. Thus the FPGA performs 4 times faster than a GPP or DSP for this convolution operation

$$C = \sum_{i=1}^{N} c_i = \sum_{i=1}^{N} a_i x_i + b_i y_i. \tag{1}$$

All of the above features make the FPGA outperform the GPP or DSP for most digital signal processing applications. Therefore, the FPGA is chosen in this paper to implement the computationally intensive EMTP calculation.

III. FPGA-Based Real-Time EMTP Skeleton

A. Data Representation

The first question that needs to be addressed for any FPGA implementation is what data format to use, since it will affect the accuracy of computation. Basically there are two types of number systems: fixed point number and floating point number. Compared with the fixed point number, the floating point number has the advantages of a dynamic range, and higher accuracy. It is widely used in real number arithmetic. Considering the FPGA resource utilization and required precision, a 46-bit floating point format is used in the FPGA-based real-time EMTP. A 46-bit floating point number consists of 1 sign bit, 11 exponent bits, and 34 fraction (mantissa) bits as shown in the Fig. 4. A floating-point number v can be described as follows:

$$v = (-1)^{\text{sign}} \times 2^{(\text{exponent}-\text{exponent bias})} \times (1.\text{mantissa}) \tag{2}$$

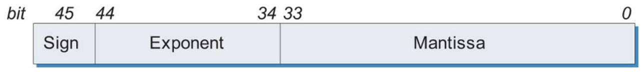


Fig. 4. The 46-bit floating-pointer number format used for real-time EMTP.

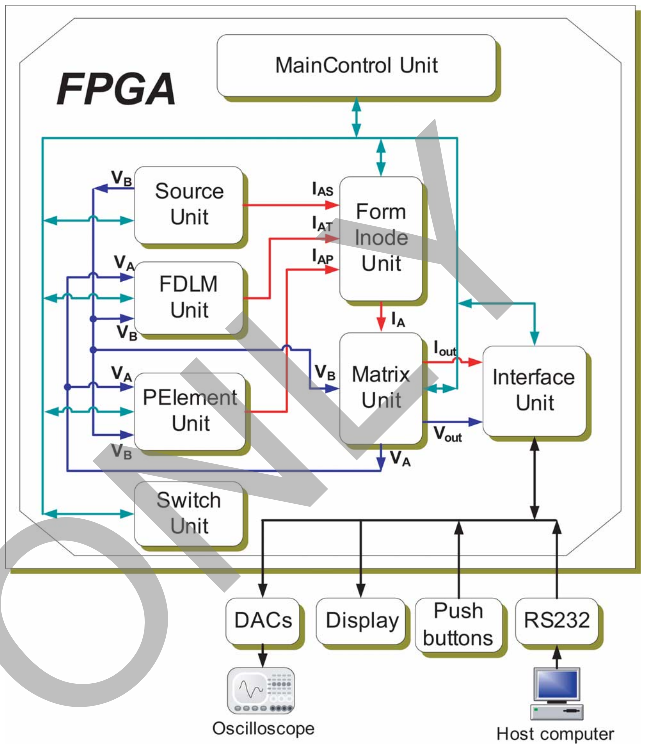


Fig. 5. FPGA-based real-time EMTP simulator architecture.

where the exponent bias is 1023. The range of v is approximately 10^{-308} to 10^{308}. In the EMTP, the convolution integration and matrix multiplication operations require accumulation. However, a floating point accumulation typically consumes more FPGA resources and requires a higher clock speed [14]. For this reason, we used a fixed point accumulator with the number format 40.100 which has 40 integer bits and 100 fraction bits to guarantee both the range and precision. Thus, all data stored in the system are floating point numbers. The addition/ subtraction and multiplication are also in floating point. Only when accumulation is required, a floating point number is converted to fixed point, and the accumulation result is then converted back to floating point.

B. System Architecture

The FPGA-based real-time EMTP consists of the following eight hardware subsystems as shown in Fig. 5: 1) `FDLM` (Frequency Dependent Line Model) unit; 2) `PElement` (Passive Element) unit; 3) `Source` unit; 4) `Switch` unit; 5) `FormInode` unit; 6) `Matrix` unit; 7) `MainControl` unit; and 8) `Interface` unit.

This is the first level of parallelism which means each subsystem has its own independent computational hardware to perform the required calculation. The second level of parallelism is within each unit as will be explained later. Details of the function and configuration of each subsystem are described in Sections III-C–I.

C. *Frequency-Dependent Line Model*

The frequency-dependent line model [15] is the most accurate line model available for calculating transients. It is based on the well-known line model equations in frequency-domain

$$V_s(\omega) = \cosh[\gamma(\omega)\ell]V_r(\omega) - Z_c(\omega)\sinh[\gamma(\omega)\ell]I_r(\omega) \tag{3a}$$

$$I_s(\omega) = \frac{1}{Z_c(\omega)}\sinh[\gamma(\omega)\ell]V_r(\omega) - \cosh[\gamma(\omega)\ell]I_r(\omega) \tag{3b}$$

where $V_s(\omega)$, $V_r(\omega)$, $I_s(\omega)$, and $I_r(\omega)$ are the voltages and currents corresponding to the sending-end (s) and receiving-end (r), respectively; ℓ is the line length; $Z_c(\omega)$ and $\gamma(\omega)$ are the frequency-dependent characteristic impedance and propagation function respectively, defined as

$$Z_c(\omega) = \sqrt{\frac{R(\omega)+j\omega L(\omega)}{G(\omega)+j\omega C(\omega)}} \tag{4a}$$

$$\gamma(\omega) = \sqrt{(R(\omega)+j\omega L(\omega))(G(\omega)+j\omega C(\omega))} \tag{4b}$$

where $R(\omega)$, $L(\omega)$, $G(\omega)$, and $C(\omega)$ are series resistance, series inductance, shunt conductance, and shunt capacitance, respectively.

To relate the line currents and voltages in frequency domain, the following new functions are defined.

Forward traveling wave functions

$$F_s(\omega) = V_s(\omega) + Z_c(\omega)I_s(\omega) \tag{5a}$$

$$F_r(\omega) = V_r(\omega) + Z_c(\omega)I_r(\omega) \tag{5b}$$

and backward traveling wave functions

$$B_s(\omega) = V_s(\omega) - Z_c(\omega)I_s(\omega) \tag{6a}$$

$$B_r(\omega) = V_r(\omega) - Z_c(\omega)I_r(\omega). \tag{6b}$$

By eliminating $V_s(\omega)$, $V_r(\omega)$, $I_s(\omega)$, and $I_r(\omega)$ from (3a) and (3b), (6a) and (6b), we obtain

$$B_s(\omega) = A_1(\omega)F_r(\omega) \tag{7a}$$

$$B_r(\omega) = A_1(\omega)F_s(\omega) \tag{7b}$$

where

$$A_1(\omega) = e^{-\gamma(\omega)\ell} = \frac{1}{\cosh[\gamma(\omega)\ell] + \sinh[\gamma(\omega)\ell]} \tag{8}$$

the time-domain form of which is defined as the *weighting function* $a_1(t)$, obtained by the inverse Fourier transform of $A_1(\omega)$. Equation (6a) and (6b) gives the Thévénin equivalent network shown in Fig. 6. The voltage sources $b_s(t)$ and $b_r(t)$ are the time domain forms of (7a) and (7b), which are the convolution integrals given as

$$b_s(t) = \int_\tau^\infty f_r(t-u)a_1(u)du \tag{9a}$$

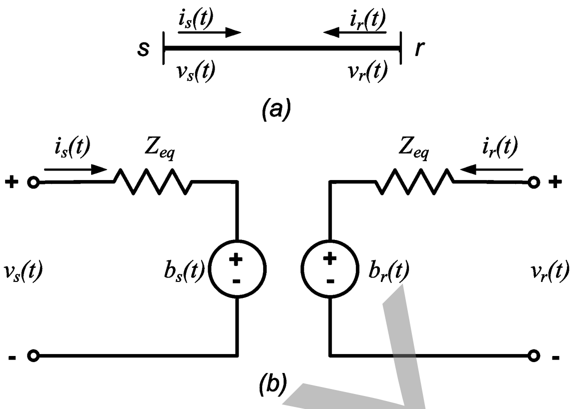


Fig. 6. (a) Transmission line and (b) its frequency-dependent model.

$$b_r(t) = \int_\tau^\infty f_s(t-u)a_1(u)du \tag{9b}$$

where

$$f_s(t) = 2v_s(t) - b_s(t) \tag{10a}$$

$$f_r(t) = 2v_r(t) - b_r(t). \tag{10b}$$

The characteristic impedance $Z_c(\omega)$ can be fitted by a rational function Z_{eq} and realized by a series of RC parallel branches.

Computational efficiency of the convolutions of (9a) and (9b) may be greatly increased if the weighting function $a_1(t)$ has the form of sum of exponential terms. To do so, the same fitting techniques for $Z_c(\omega)$ are applied for approximating the weighting function $A_1(\omega)$ to produce a N_p-order rational function. Finally, the (9a) and (9b) are evaluated by using recursive convolution integrals (11) and (12)

$$\begin{aligned} b_s(t) &= \sum_{i=1}^{N_p} b_{si}(t) \\ &= \sum_{i=1}^{N_p} [M_i b_{si}(t-\Delta t) + P_i f_r(t-\tau) \\ &\quad + Q_i f_r(t-\tau-\Delta t)] \end{aligned} \tag{11}$$

$$\begin{aligned} b_r(t) &= \sum_{i=1}^{N_p} b_{ri}(t) \\ &= \sum_{i=1}^{N_p} [M_i b_{ri}(t-\Delta t) + P_i f_s(t-\tau) \\ &\quad + Q_i f_s(t-\tau-\Delta t)] \end{aligned} \tag{12}$$

where Δt and τ are timestep and propagation time delay, respectively. $b_{si}(t)$ and $b_{ri}(t)$ are the convolution results corresponding to each exponential terms of $a_1(t)$. M_i, P_i, and Q_i are constants depending on the poles and zeros of fitted rational function and Δt.

For balanced three-phase operation, a fully transposed transmission line model is commonly decoupled into three separate systems. In the EMTP, the Clarke's transformation is used to transform the three phases into two aerial modes (α, β) and one ground mode (0) [2]. The transformation matrix $\mathbf{T}$ is given as follows:

$$\mathbf{T} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & \sqrt{2} & 0 \\ 1 & \frac{-1}{\sqrt{2}} & \frac{\sqrt{3}}{\sqrt{2}} \\ 1 & \frac{-1}{\sqrt{2}} & -\frac{\sqrt{3}}{\sqrt{2}} \end{bmatrix}. \qquad (13)$$

All of the above line calculations are performed in the FDLM unit shown in Fig. 5. The main functions of this unit include evaluating recursive convolution integral (11) and (12) to get the equivalent current sources, sending the equivalent current sources $\mathbf{I_{AT}}$ to FormInode unit, and updating the transmission line history values (10a) and (10b). Due to the complexity of the FDLM, this unit uses more FPGA resources (LEs, DSP Blocks, and memory) than other units and is the most time consuming. As such, the operations within this unit are fully parallelled which forms the second level of parallelism. For example, the calculation of transmission line sending and receiving ends are done in parallel; the modal calculations are simultaneous as well. Meanwhile, all the calculations are deeply pipelined to achieve the highest throughput. Fig. 7 shows the pipelined calculation scheme for the recursive convolution integral (11) in one mode which is the most time consuming computation in this EMTP implementation. To employ the pipeline, the memory for $b_{si}(t)$ has to be a dual-port RAM to support simultaneous write and read access.

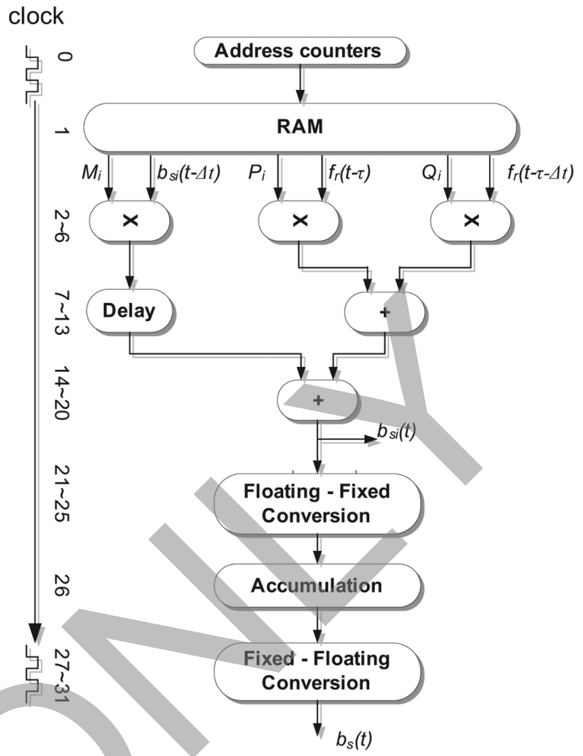


Fig. 7. Pipelined recursive convolution computation in the FPGA.

D. Linear Lumped Elements

All linear lumped elements are represented by their discrete-time models according to the trapezoidal rule of integration [2]. Besides the basic lumped elements R, L, C, their combinations such as RL, RC, LC, and RLC are implemented as individual elements to reduce the number of nodes and nodal equations in the FPGA-based EMTP. Fig. 8(a) and (b) show an example of L element and RLC element. The Norton equivalent circuit of the discretized branch of all lumped elements is shown in Fig. 8(c). The R_{eq} is the equivalent resistance. $I_h(t-\Delta t)$ is the history term known from the solution at the preceding timestep and can be updated recursively using generic (14) for R, L, C, RL, and RC elements, and (15) for LC and RLC elements. See (14)–(15) at the bottom of the page where P_1, P_2, P_3, P_4 are constant coefficients dependent on the element values and Δt.

The PElement unit in Fig. 5 implements all of these passive elements. The functions of this unit include sending equivalent current sources $\mathbf{I_{AP}}$ to FormInode unit, and updating the history values in (14) or (15). The branch current for each lumped element branch is also calculated for later use as measured currents.

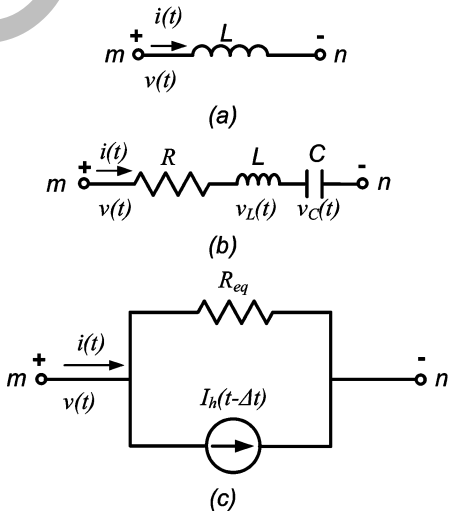


Fig. 8. (a) L element, (b) RLC element, and (c) generic equivalent network for linear lumped elements.

$$I_h(t-\Delta t) = P_1 I_h(t-2\Delta t) + P_2 v(t-\Delta t) \qquad (14)$$

$$\begin{cases} I_h(t-\Delta t) = \frac{[V_{h,L}(t-\Delta t)+V_{h,C}(t-\Delta t)]}{R_{eq}} \\ V_{h,L}(t-\Delta t) = P_1 V_{h,L}(t-2\Delta t) + P_2 i(t-\Delta t) \\ V_{h,C}(t-\Delta t) = P_3 V_{h,C}(t-2\Delta t) + P_4 i(t-\Delta t) \end{cases} \qquad (15)$$

E. Sources

The ideal voltage and current sources at power-frequency are modeled using sinusoidal functions. In the FPGA-based real-time EMTP, the sinusoidal function is precalculated for one cycle (16.66 ms) and stored in a lookup table. The time interval for the sinusoidal LUT is 1 μs which is much less than the simulation timestep Δt. The initial phase angle is converted to the address of the LUT.

The `Source` unit implements the system input voltage and current sources. In this unit, all sources are calculated in real time by multiplying the magnitude by the *cosine* value. The current sources $\mathbf{I_{AS}}$ are sent to `FormInode` unit. The voltage sources $\mathbf{V_B}$ are saved and sent to the `FDLM` unit and the `PElement` unit as known nodal voltages.

F. Switches

Circuit breakers are modeled as time-controlled switches with infinite resistance ($R = \infty$) in the open position and zero resistance ($R = 0$) in the closed position. The operation of a switch usually causes the change of the system topology leading to the change of system admittance matrix. In the FPGA-based real-time EMTP, the system admittance matrices corresponding to all possible switch combinations are pre-calculated and stored in the memory.

The `Switch` unit checks all the switches for changes in real time. A real-time clock generation component is used to provide the present time and the timestep Δt signal. At each timestep, all switches compare the operation time which is saved in a RAM with the present time to update the switch states. The timestep Δt signal is also sent to `MainControl` unit to check if the simulation can be done in the assigned Δt.

G. Network Solution

For a network with N nodes, the nodal equation can be formed as follows:

$$\mathbf{YV} = \mathbf{I} \tag{16}$$

where $\mathbf{Y}$ is the system nodal admittance matrix;$\mathbf{V}$ is the vector of N node voltages, and $\mathbf{I}$ is the vector of N current sources. This equation can be partitioned into a set A of nodes with unknown voltages, and a set B of nodes with known voltages. Thus, (16) becomes

$$\begin{bmatrix} \mathbf{Y_{AA}} & \mathbf{Y_{AB}} \\ \mathbf{Y_{BA}} & \mathbf{Y_{BB}} \end{bmatrix} \begin{bmatrix} \mathbf{V_A} \\ \mathbf{V_B} \end{bmatrix} = \begin{bmatrix} \mathbf{I_A} \\ \mathbf{I_B} \end{bmatrix}. \tag{17}$$

The known current sources $\mathbf{I}_A$ is contributed by transmission lines $\mathbf{I}_{AT}$, lumped passive elements $\mathbf{I}_{AP}$, and known supply current sources $\mathbf{I}_{AS}$

$$\mathbf{I_A} = \mathbf{I_{AT}} + \mathbf{I_{AP}} + \mathbf{I_{AS}}. \tag{18}$$

The `FormInode` unit receives the nodal current injections $\mathbf{I_{AT}}$, $\mathbf{I_{AP}}$, and $\mathbf{I_{AS}}$ from `FDLM` unit, `PElement` unit, and `Source` unit, respectively. Then it adds them up to form the nodal current vector $\mathbf{I_A}$. The unknown voltages are then found by solving the system of linear, algebraic equation

$$\mathbf{V_A} = \mathbf{Y_{AA}^{-1}}(\mathbf{I_A} - \mathbf{I_{A1}}) \tag{19}$$

where

$$\mathbf{I_{A1}} = \mathbf{Y_{AB}V_B}. \tag{20}$$

In the traditional EMTP [2], the above equation is solved first by triangularizing the $\mathbf{Y}_{AA}$ matrix through a Gauss reduction procedure and then back-substituting for the updated values. For the real-time implementation the $\mathbf{Y}_{AA}^{-1}$ matrix is precalculated and stored in memory for all possible switch state combinations.

The `Matrix` unit is responsible for the network solution (19). The result $\mathbf{V}_A$ is sent to the `FDLM` unit and `PElement` unit as calculated nodal voltages. Moreover, the `Matrix` unit calculates the measured current outputs and transforms output voltages and currents from modal-domain to phase-domain using Clarke's transformation (13).

H. Paralleled EMTP Algorithm

To fully exploit the parallel architecture of the FPGA, the inherent parallelism in the EMTP algorithm needs to be analyzed. The EMTP algorithm consists of three stages: *Stage 1*—forming the nodal current injection vector $\mathbf{I}_A$(18), *Stage 2*—solving for the unknown node voltages (19), and *Stage 3*—updating the history terms for network components. These three stages form sequential bottlenecks in the EMTP algorithm. For example, to solve the nodal equation, the nodal current source injections have to be known from *Stage 1*. Similarly, to update the history terms in *Stage 3*, we need the node voltages from *Stage 2*. Nevertheless, we can parallelize the operations within each of these stages. Fig. 9 shows the parallelled real-time EMTP algorithm flowchart. In *Stage 1* four operations are performed simultaneously: the calculation of the supply sources, calculation of transmission lines equivalent current sources at both sending and receiving ends, retrieving the lumped passive elements' history current sources from memory, and checking switch states. Similarly, in *Stage 3* the history terms for the transmission lines and passive elements are updated simultaneously.

The `MainControl` unit coordinates the operation of whole system to carry out the parallelled EMTP algorithm. It sends out control signals to each unit and receives acknowledge signals. As shown in Fig. 9, the three main tasks to be performed are: initialization, resetting, and simulation. During the initialization task, all the parameters of system to be simulated are downloaded into FPGA. The resetting task resets all the initial variables to zero. The real-time simulation task is performed in an infinite loop.

I. User Interface

The `Interface` unit implements the user interface with the external I/O devices. Three push-button switches and a dual 7-segment display are used to reset the system, switch functions, and set values. The functions include changing the output voltage node number, output current number, setting the timestep, etc. A RS232 serial interface is implemented to download system input data from the host PC. The output three-phase voltages and currents to be displayed on the oscilloscope are sent to a 16-channel digital to analog converter (DAC) board. Each DAC chip AD5545 is a dual serial input, 16-bit DAC from Analog Devices®. Before sending the simulation results to the DACs, they are converted to the fixed point (40.100)

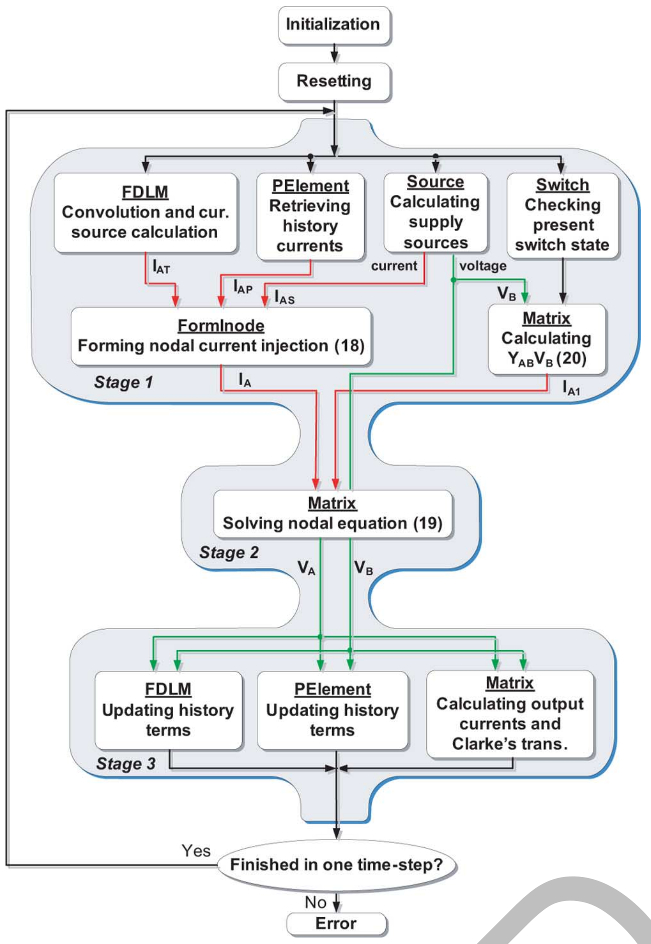

Initialization
Resetting
FDLM
Convolution and cur. source calculation
PElement
Retrieving history currents
Source
Calculating supply sources
Switch
Checking present switch state
I_{AT}
I_{AP}
I_{AS}
current
voltage
V_B
FormInode
Forming nodal current injection (18)
Matrix
Calculating $Y_{AB}V_B$ (20)
Stage 1
I_A
I_{A1}
Matrix
Solving nodal equation (19)
Stage 2
V_A
V_B
FDLM
Updating history terms
PElement
Updating history terms
Matrix
Calculating output currents and Clarke's trans.
Stage 3
Yes
Finished in one time-step?
No
Error

Fig. 9. Parallelled real-time EMTP algorithm for FPGA implementation.

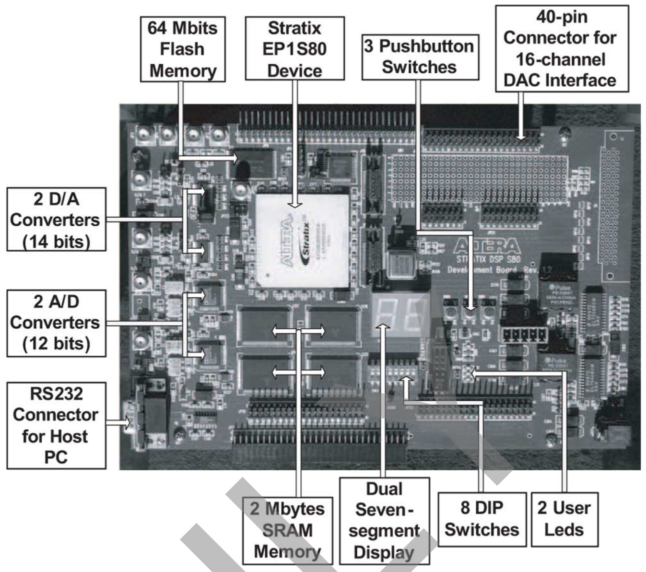

64 Mbits Flash Memory
Stratix EP1S80 Device
3 Pushbutton Switches
40-pin Connector for 16-channel DAC Interface
2 D/A Converters (14 bits)
2 A/D Converters (12 bits)
RS232 Connector for Host PC
2 Mbytes SRAM Memory
Dual Seven-segment Display
8 DIP Switches
2 User Leds

Fig. 10. Altera Stratix S80 FPGA development board.

TABLE I
FPGA RESOURCES UTILIZED BY EMTP UNITS

Unit	Logic Elements	DSP Blocks	Memory Bits
`FDLM`	37245 (47.1%)	96 (54.5%)	733632 (9.8%)
`PElement`	10909 (13.8%)	40 (22.7%)	64896 (0.87%)
`Source`	2231 (2.8%)	16 (9.0%)	162278 (2.18%)
`Switch`	366 (0.4%)	0	768 (0.01%)
`FormInode`	6725 (8.5%)	0	8767 (0.12%)
`Matrix`	11863 (15.1%)	16 (9.0%)	40256 (0.54%)
`MainControl`	163 (0.2%)	0	0
`Interface`	2100 (2.6%)	0	1130496 (15.2%)
Total	71602 (90.5%)	168 (95.2%)	2141093 (28.8%)

format. An online 16-bit moving window was implemented to select any consecutive 16 bits from these 140 bits by using the push-button. For example, if the signal is too small to be seen on the oscilloscope, the push-button can be used to select more bits from the fractional part. Alternately, the data from real-time simulation can also be saved in the RAM and retrieved later for plotting with MATLAB. The saved results are 4096 points floating-point three-phase node voltages and measured currents after the switch operation time when transients are introduced.

IV. Implementation and Operation of FPGA-Based Real-Time EMTP

A. Implementation

The FPGA-based real-time EMTP simulator was implemented on a Altera® Stratix™ S80 Development Board, shown in Fig. 10, supplied by the Canadian Microelectronics Corporation (CMC). The FPGA EP1S80 used on this board has the following features: 79 040 logic elements (LEs); 7 427 520 total RAM bits; 176 DSP blocks based on 9×9 multiplier; and 679 maximum user I/O pins. The external clock is from the 80-MHz oscillator on the development board.

Table I lists resources utilization by each unit of the FPGA-based real-time EMTP simulator. As can be seen from this table, the logic elements and DSP blocks are almost fully utilized. Also, as mentioned earlier, the `FDLM` unit utilizes the most logic resources of the FPGA. The second most resource intensive unit is the `Matrix` unit.

B. Input System Data Preparation

The standard ATP data files [16] are used to input data to the FPGA-based real-time EMTP. To simulate different electrical networks, we can simply draw the network using the `ATPDraw` software which generates the ATP data automatically. Meanwhile, the offline simulation is also done, which can be used to validate the real-time simulation. The size of a network that can be simulated in real time with an acceptable timestep depends on the available FPGA resources. The developed real-time EMTP code is quite general; the same VHDL program can be executed on a larger FPGA if necessary for simulating large power systems.

A MATLAB script program `iniFPGA` is developed to generate all initial data required by FPGA-based real-time EMTP including transmission lines parameters, lumped elements parameters, source parameters, and switches parameters. Inside this program, a function `readATP` is called to read the ATP data file. The function allows four main sorting cards to be read: the `branch` card, the `source` card, the `switch` card, and the `output` card. During the data processing, network reduction methods are employed to reduce the number of nodes. For example, the elements in series are combined into one component if there is no voltage output required for the intermediate nodes.

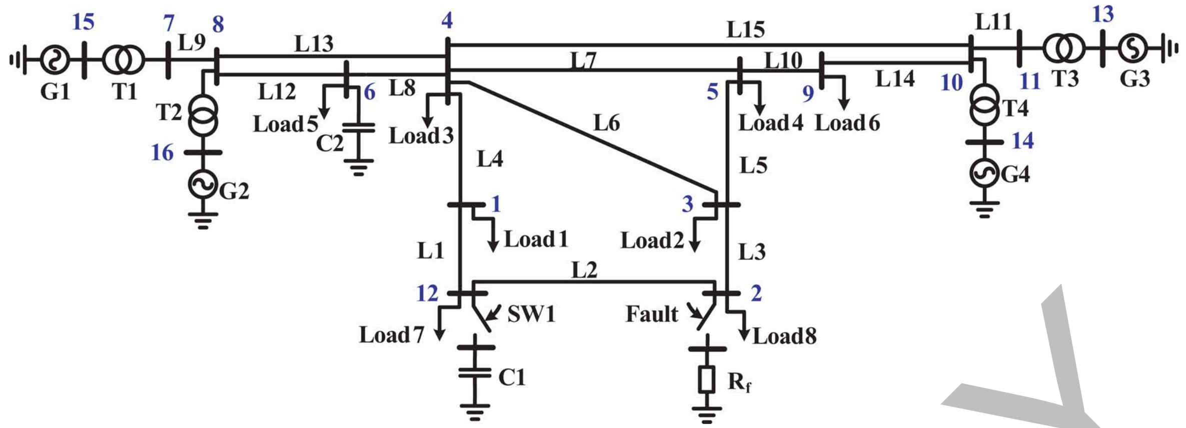


Fig. 11. Single-line diagram of the power system used in the case study.

TABLE II
EXECUTION TIME OF FPGA-BASED REAL-TIME EMTP FOR THE CASE STUDY

Stage	Time
Stage1	5.500μs
Stage2	3.375μs
Stage3	2.338μs
Total	11.213μs

C. Operation

Once the VHDL code for the simulator is compiled, the bitstream is downloaded into the SRAM on the FPGA development board through the JTAG interface. The simulator is now ready to receive the system input data. The input data in ASCII format is then downloaded into the RAM inside the Stratix FPGA device using the Microsoft Windows *HyperTerminal* program. The simulation starts immediately once the download is finished. An initial timestep is sent to the FPGA during the input data download. If the first simulation loop cannot be finished in this timestep, an error message is displayed. In such a case, the timestep can be incremented by 1 μs by using the push-buttons and the simulation can be rerun until an adequate timestep is found. Then system input data is regenerated and the simulation can be run based on that timestep.

V. Real-Time Simulation Case Study

An example power system is simulated to show the effectiveness of the FPGA-based real-time EMTP. The system consists of 15 transmission lines, 4 generators, and 8 loads as shown in Fig. 11. The complete system data is listed in the Appendix. The lines are modeled using frequency-dependent line model. The generators and transformers are modeled as Thévénin equivalent networks with voltage sources behind constant impedances. Each of the system component is allocated to the appropriate FPGA unit (Fig. 5) to process its calculation. For example, the 15 transmission line models are pipelined through the FDLM unit. Table II shows the execution time for each stage of the paralleled EMTP algorithm (Fig. 9) for implementing this system. The total execution time is 11.213 μs, while the actual timestep is 12 μs. Based on the 12.5 ns clock period used for the FPGA, this implies that it took 960 clock cycles to complete one loop of simulation for this case study. The achieved timestep $\Delta t = 12\ \mu$s is at least 4 times smaller than the acceptable timestep of 50 μs for transient simulation. As such it is possible to simulate a system that is at least 4 times as large as the present system, i.e., a system with 60 lines, 16 generators, and 32 loads can be simulated in real time with a timestep of $\Delta t = 50\ \mu$s on this FPGA. We can also see from this table, that the *Stage1* utilizes the most execution time. This is due to the convolution operation of FDLM as mentioned earlier. The timestep could be decreased significantly if more parallel `FDLM` units were implemented on a larger FPGA. Two transients events are simulated. The first transient event is the capacitor $C1$ switched at Bus 12 at time $t = 0.05$ s. Figs. 12 and 14 show the three-phase voltages and currents waveforms at Bus 12 obtained from the offline ATP simulation with a timestep of 12 μs. As can be seen in these figures, during the capacitor transient, voltage v_a drops from 110 kV to almost 0 kV immediately. A negative peak current of 1.5 kA happens on i_b at $t = 0.05$ s and a positive peak current of 1.3 kA appears on i_a at $t = 0.053$ s. The transient lasts for about one cycle. Identical behavior can be observed from Figs. 13 and 15 which were captured by a real-time oscilloscope connected to the DACs. The second transient event is the Bus 2 three-phase to ground fault with 2 Ω resistance to ground which occurs at $t = 0.05$ s. Figs. 16 and 18 show the three-phase voltages and fault currents waveforms at Bus 12 obtained from the offline simulation. As can be seen, the voltages drop from 180 kV peak to 60 kV peak, and high frequency transient currents occur during the fault. The transient lasts for about 2 cycles. Figs. 17 and 19 show the transient waveforms captured by real-time oscilloscope. Again, detailed agreement between ATP offline simulation and the FPGA-based real-time EMTP results can be observed.

VI. Conclusions

FPGAs are increasingly being used as core computational hardware in many applications which traditionally used sequential GPPs or DSPs for carrying out intensive calculations. This is due to their parallel architecture, high clock speed, and high logic resource capacity. All of these features make FPGAs most suitable for electromagnetic transient simulation. This paper proposed a FPGA-based real-time EMTP simulator. The central feature of this simulator is the frequency-dependent line model which allows for accurate line transient calculations. To fully exploit the parallel processing capacity of the FPGA, a

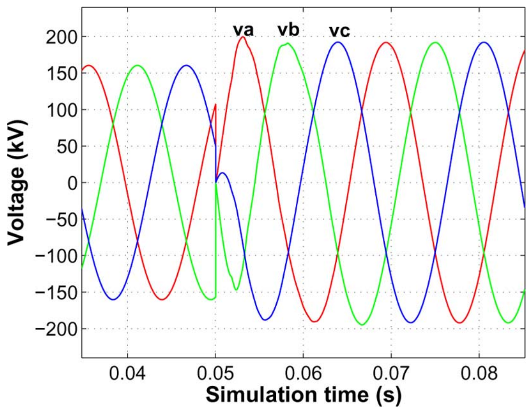


Fig. 12. Three-phase voltages at Bus 12 during a capacitor $C1$ switching transient at $t = 0.05$ s (ATP simulation).

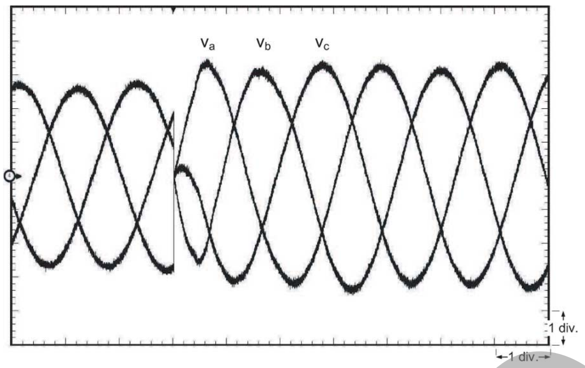


Fig. 13. Real-time oscilloscope trace showing three-phase voltages during a capacitor $C1$ switching transient at Bus 12 ($X-\text{axis}: 1div. = 10$ ms, $Y-\text{axis}: 1div. = 58$ kV).

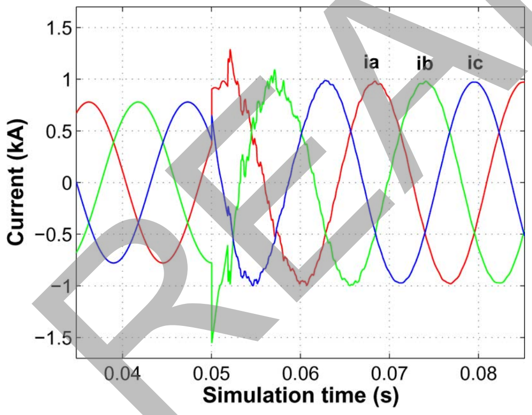


Fig. 14. Three-phase currents at Bus 12 during a capacitor $C1$ switching transient at $t = 0.05$ s (ATP simulation).

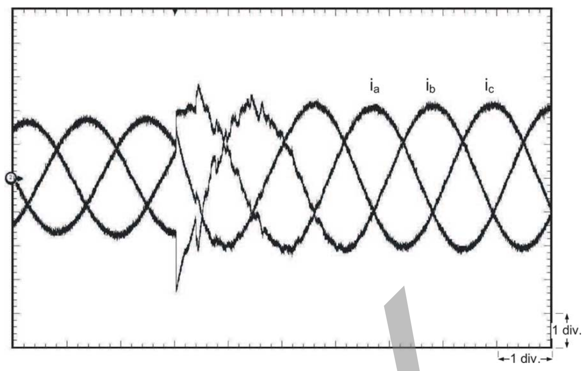


Fig. 15. Real-time oscilloscope trace showing three-phase currents during a capacitor $C1$ switching transient at Bus 12 ($X-\text{axis}: 1div. = 10$ ms, $Y-\text{axis}: 1div. = 0.44$ kA).

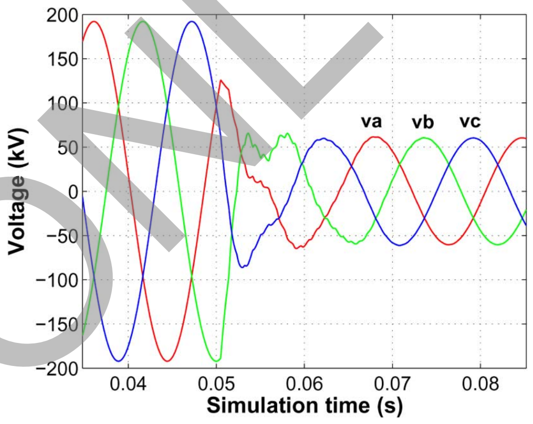


Fig. 16. Three-phase voltages at Bus 12 during a three-phase to ground fault at Bus 2 at $t = 0.05$ s (ATP simulation).

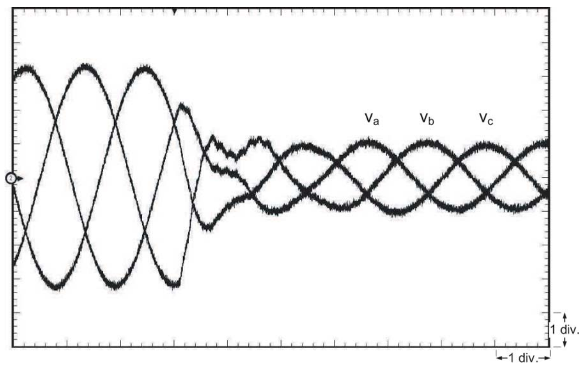


Fig. 17. Real-time oscilloscope trace of the three-phase Bus 12 voltages during a three-phase to ground fault transient at Bus 2 ($X-\text{axis}: 1div. = 10$ ms, $Y-\text{axis}: 1div. = 58$ kV).

paralleled EMTP algorithm is proposed which utilizes a deeply pipelined computation and a high precision floating point number representation. The developed real-time simulator accepts the network data in standard ATP input format, making the simulator quite general. An example of a power system with full frequency-dependent line modeling is used to show the accuracy of the real-time simulator. The transient results from the real-time simulator show excellent agreement with an offline ATP simulation of the original system.

One of the promising applications of the developed FPGA-based real-time EMTP simulator is that it can be interfaced with a PC-cluster based simulator for accommodating large network sizes. In such a scenario, FPGAs can be used as accelerators for modeling a part of the system in great detail with the remainder modeled in the PC-cluster. Building real-time digital simulators using multiple-FPGA architectures is also a feasible option based on current market pricing of these devices. Due to the high clock speed and large density of next generation FPGAs,

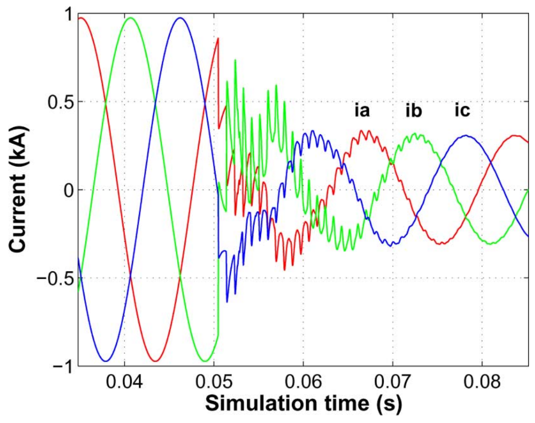


Fig. 18. Three-phase fault currents at Bus 12 during a three-phase to ground fault at Bus 2 at $t = 0.05$ s (ATP simulation).

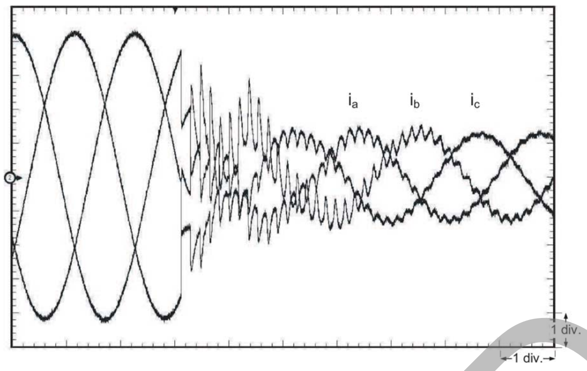


Fig. 19. Real-time oscilloscope trace of the three-phase Bus 12 fault currents during a three-phase to ground fault transient at Bus 2 ($X - \text{axis} : 1div. = 10$ ms, $Y - \text{axis} : 1div. = 0.22$ kA).

TABLE III
LINE PARAMETERS

Line	Len. (km)	Z_c order	A_1 order	Line	Len. (km)	Z_c order	A_1 order
L1	120	12	19	L9	10	12	18
L2	136	11	19	L10	35	12	24
L3	165	12	19	L11	15	12	12
L4	150	11	18	L12	65	12	19
L5	120	11	20	L13	133	12	19
L6	400	12	30	L14	42	12	21
L7	220	12	14	L15	375	12	30
L8	35	12	24				

we foresee their application for the real-time simulation of fast front transients in power systems.

APPENDIX

The system data for the real-time simulation case study is given in Tables III–V. Fig. 20 shows the tower geometry for the lines used in the case study.

TABLE IV
LOAD PARAMETERS

Load	Z_{ph}
Load1	1200Ω, 500mH
Load2	2150Ω, 380mH
Load3	250Ω, 25mH
Load4	350Ω, 60mH
Load5	250Ω, 25mH
Load6	420Ω, 30mH
Load7	200Ω, 130mH
Load8	650Ω, 250mH
C1	5μF
C2	20μF

TABLE V
GENERATOR AND TRANSFORMER PARAMETERS

Generator	Parameters per phase
G1	$1.03\angle 20.2°$, Z_{G1}: 1.2Ω, 38.98mH
G2	$1.01\angle 10.5°$, Z_{G2}: 1.1Ω, 45.52mH
G3	$1.03\angle$-6.8°, Z_{G3}: 0.9Ω, 38.98mH
G4	$1.01\angle$-17.0°, Z_{G4}: 0.8Ω, 35.23mH
Transformer	**Parameters per phase**
T1	Z_{T1}: 1.5Ω, 23.4mH
T2	Z_{T2}: 0.8Ω, 29.5mH
T3	Z_{T3}: 1.6Ω, 23.4mH
T4	Z_{T4}: 0.6Ω, 20.8mH

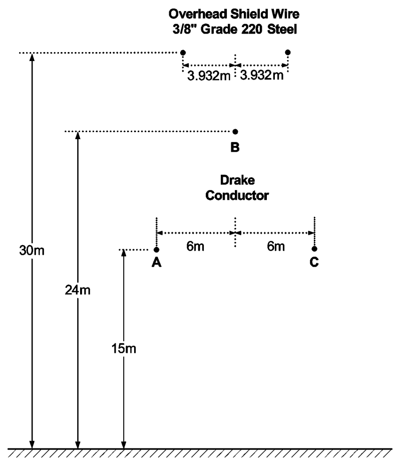


Fig. 20. Tower geometry of transmission lines in the case study.

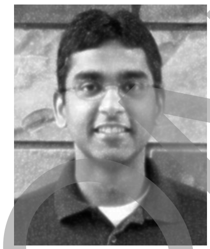
REFERENCES

[1] H. W. Dommel, "Digital computer solution of electromagnetic transients in single and multiphase networks," *IEEE Trans. Power App. Syst.*, vol. PAS-88, no. 4, pp. 388–399, Apr. 1969.
[2] H. W. Dommel, *EMTP Theory Book*. Portland, OR: Bonneville Power Admin., 1985.
[3] X. Wang and R. M. Mathur, "Real-time digital simulator of the electromagnetic transients of transmission lines with frequency dependence," *IEEE Trans. Power Del.*, vol. 4, no. 4, pp. 2249–2255, Oct. 1989.
[4] P. G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A real-time digital simulator for testing relays," *IEEE Trans. Power Del.*, vol. 7, no. 1, pp. 207–213, Jan. 1992.

[5] J. R. Marti and L. R. Linares, “Real-time EMTP-based transients simulation,” *IEEE Trans. Power Syst.*, vol. 9, no. 3, pp. 1309–1317, Aug. 1994.
[6] C. Dufour, H. Le-Huy, J. Soumagne, and A. E. Hakimi, “Real-time simulation of power transmission lines using marti model with optimal fitting on dual-DSP card,” *IEEE Trans. Power Del.*, vol. 11, no. 1, pp. 412–419, Jan. 1996.
[7] X. Wang, D. A. Woodford, R. Kuffel, and R. Wierckx, “A real-time transmission line model for a digital TNA,” *IEEE Trans. Power Del.*, vol. 11, no. 2, pp. 1092–1097, Apr. 1996.
[8] O. Devaux, L. Lavacher, and O. Huet, “An advanced and powerful real-time digital transient network analyser,” *IEEE Trans. Power Del.*, vol. 13, no. 2, pp. 421–426, Apr. 1998.
[9] J. A. Hollman and J. R. Marti, “Real-time network simulation with PC-cluster,” *IEEE Trans. Power Syst.*, vol. 18, no. 2, pp. 563–569, May 2003.
[10] L. Pak, M. O. Faruque, X. Nie, and V. Dinavahi, “A versatile cluster-based real-time digital simulator for power engineering research,” *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 455–465, May 2006.
[11] X. Nie, Y. Chen, and V. Dinavahi, “Real-time transient simulation based on a robust two-layer network equivalent,” *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1771–1781, Nov. 2007.
[12] M. Matar, M. Abdel-Rahman, and A. Soliman, “FPGA-based real-time digital simulation,” in *Proc. Int. Conf. on Power System Transients*, Montreal, QC, Canada, Jun. 2005, pp. 1–6.
[13] J. Rose, “Architecture of field-programmable gate arrays,” *Proc. IEEE*, vol. 81, no. 7, pp. 1013–1029, Jul. 1993.
[14] Z. Luo and M. Martonosi, “Accelerating pipelined integer and floating-point accumulations in configurable hardware with delayed addition techniques,” *IEEE Trans. Comput.*, vol. 49, no. 3, pp. 208–218, Mar. 2000.
[15] J. R. Marti, “Accurate modeling of frequency-dependent transmission lines in electromagnetic transient simulations,” *IEEE Trans. Power App. Syst.*, vol. PAS-101, no. 1, pp. 147–155, Jan. 1982.
[16] *Alternative Transients Program Rule Book*. West Linn, OR: Canadian/American EMTP Users Group, 1999.

Yuan Chen (S’06) received the B.Sc. and M.Sc. degrees in electrical engineering from Hunan University, Hunan, China, in 1992 and 2000, respectively, and is currently pursuing the M.Sc. degree in electrical and computer engineering at the University of Alberta, Edmonton, AB, Canada.

He was an Electrical Engineer in China. His research interests include real-time simulation of power systems and field-programmable gate arrays.



Venkata Dinavahi (M’06) received the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2000.

He is an Associate Professor at the University of Alberta, Edmonton, AB, Canada. His research interests include electromagnetic transient analysis, power electronics, and real-time digital simulation and control.