Digital Hardware Emulation of Universal Machine and Universal Line Models for Real-Time Electromagnetic Transient Simulation

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

Abstract—**Real-time electromagnetic transient simulation plays an important role in the planning, design, and operation of power systems. Inclusion of accurate and complicated models, such as the universal machine (UM) model and the universal line model (ULM), requires significant computational resources. This paper proposes a digital hardware emulation of the UM and the ULM for real-time electromagnetic transient simulation. It features accurate floating-point data representation, paralleled implementation, and fully pipelined arithmetic processing. The hardware is based on advanced field-programmable gate array (FPGA) using VHDL. A power system transient case study is simulated in real time to validate the design. On a 130-MHz input clock frequency to the FPGA, the achieved execution times for UM and ULM models are 2.5 μs and 1.42 μs, respectively. The captured real-time oscilloscope results demonstrate high accuracy of the emulator in comparison to the offline simulation of the original system in the EMTP-RV software.**

Index Terms—**Electrical machines, electromagnetic transient analysis, field-programmable gate arrays (FPGAs), hardware-in-the-loop simulation, modeling, parallel algorithms, power system simulation, real-time systems, transmission lines.**

I. Introduction

ROTATING electric machineries are widely used in power systems either as generators or industrial loads, and their accurate modeling is paramount for electromagnetic transient simulation. The machine itself can be of many different types such as induction machine, synchronous machine, and dc machine. Each type of machine may have different electrical representation and custom interfaces between the machine and the network and between the machine and associated mechanical system. It would be extremely cumbersome and time consuming to code each of them individually into the Electromagnetic Transient Program (EMTP) [1]. For this reason, the universal machine (UM) model was proposed [2], [3]. It is a unified generalized model which can be used to represent up to 12 types of rotating machines. The UM model was established on the concept that all types of machines can be represented by some coupled electrical coils and that the associated mechanical system can be replaced by the equivalent electrical analog.

The propagation of electromagnetic transients on a transmission line is greatly influenced by the frequency dependence of its parameters, namely, its series impedance and shunt admittance. Traditionally, the most accurate transmission line model is the frequency-dependent model built in the modal domain [4]. The transformation matrix between the modal domain and the phase domain is real and constant. This model works well for symmetrical and transposed lines; however, for asymmetrical and untransposed line configurations, the use of a constant modal transformation matrix causes error and numerical instability, particularly for cable models. Although this deficiency can be overcome by taking the frequency dependence of the transformation matrix into account [5], the practical implementation of the model is complicated. Phase-domain modeling avoids the transformation matrix by formulating the transmission line equations directly in the phase domain [6]–[8]. Among these phase-domain line models, the universal line model (ULM) [8] is considered numerically efficient and robust for both overhead lines and underground cables.

Offline electromagnetic transient simulation using software such as ATP, PSCAD/EMTDC, and EMTP-RV has been very successful for many years. Although execution time is still a matter of concern in such offline tools, it is not as critical as in a real-time digital simulator which has to interact with external devices in a hardware-in-the-loop scenario. The main applications of real-time transient simulators include testing of protective relays and controllers in power systems as well as power electronic systems, and utility-specific studies in planning and operation of power systems. Accurate modeling of rotating machines and transmission lines is one of the most computationally demanding tasks for a real-time simulator. Moreover, a realistic reproduction of high-frequency transients often requires a very small simulation time step. Existing real-time simulators largely employ sequential processors such as general-purpose CPUs or DSPs as their core computational processors. To meet the stringent real-time step-size constraints, a compromise is usually made between the size of the system simulated and the complexity of the component models. For example, a rotating machine is modeled by its Thevenin equivalent or a low-order lumped machine model, and a transmission line is modeled by a relatively simpler distributed parameter traveling wave model instead of a frequency-dependent model,

Manuscript received December 23, 2010; revised March 27, 2011; accepted May 6, 2011. Date of publication May 23, 2011; date of current version October 18, 2011. This work was supported by the Natural Science and Engineering Research Council of Canada.

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/TIE.2011.2157296

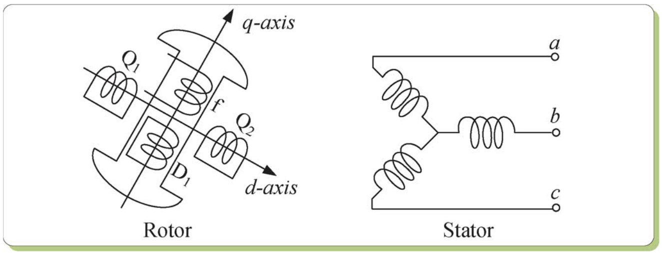


Fig. 1. Winding representations in the UM model.

when large systems need to be simulated in real time. Of course, sacrificing the model complexity degrades the accuracy of the transients the real-time simulator is able to reproduce.

Owing to the fast developments in VLSI technology and concurrent advances in digital hardware design tools, field-programmable gate arrays (FPGAs) are becoming the processors of choice for many applications [9]–[31]. This is because the FPGA has the intrinsic advantages of full hardware parallelism and user programmability. The digital hardware implementation of EMTP in real time is becoming highly attractive. In [23], a parallel EMTP algorithm was proposed to simulate basic linear network elements, including the modal-domain frequency-dependent transmission line model. Further development in [24] included an iterative Newton-based solver for simulating nonlinear element in real time on the FPGA.

This paper proposes a digital hardware emulation of UM and ULMs for real-time electromagnetic transient simulation. The machine equations are solved iteratively using the compensation method. The proposed FPGA-based hardware also includes other power system elements such as supply source, linear lumped RLC, and switch, which makes it a complete real-time electromagnetic transient emulator. The whole design is fully paralleled and deeply pipelined using floating-point number representation to achieve high speed and accuracy requirement. Section II describes the UM model, its interfacing to the network, and detailed hardware implementation on FPGA. Section III explains the ULM formulation and hardware implementation in detail. The overall hardware framework for the network emulator and the simulation procedure are given in Section IV. A real-time transient simulation case study is presented in Section V. The real-time simulation results have been validated using offline simulation results from the EMTP-RV software. Conclusions are given in Section VI.

II. UM Model Hardware Emulation

A. UM Model Formulation

TABLE I
Equivalence Between Mechanical and Electrical Quantities for the UM Model

Mechanical	Electrical
T_m: load torque [Nm]	I_m: current source [A]
T_e: air gap torque [Nm]	I_e: current [A]
J: inertia [kg-m^2]	C: capacitance [F]
D: damping [Nm-s/rad]	$1/R$: conductance [S]
ω: rotor speed [rad/s]	V: voltage [V]

The UM model is established on the synchronously rotating $dq0$ reference frame. It can have, at most, three stator windings, any number of windings $D_1, D_2, \ldots, D_m$ on the rotor direct axis (d-axis), and any number of windings $Q_1, Q_2, \ldots, Q_n$ on the rotor quadrature axis (q-axis). Without loss of generality, as shown in Fig. 1, three stator windings $\{a, b, c\}$, one field winding f, one damper winding D_1 on the d-axis, and two damper windings $\{Q_1, Q_2\}$ on the q-axis are considered in this paper. The voltage–current–flux relationship in these coupled windings can be described by

$$\boldsymbol{v}_{dq0} = -\boldsymbol{R}\boldsymbol{i}_{dq0} - \frac{d\boldsymbol{\lambda}_{dq0}}{dt} + \boldsymbol{u} \tag{1}$$

$$\boldsymbol{\lambda}_{dq0} = \boldsymbol{L}\boldsymbol{i}_{dq0} \tag{2}$$

where

$$\boldsymbol{v}_{dq0} = [v_d \quad v_q \quad v_0 \quad v_f \quad 0 \quad 0 \quad 0]^{\mathrm{T}}$$
$$\boldsymbol{i}_{dq0} = [i_d \quad i_q \quad i_0 \quad i_f \quad i_{D1} \quad i_{Q1} \quad i_{Q2}]^{\mathrm{T}}$$
$$\boldsymbol{\lambda}_{dq0} = [\lambda_d \quad \lambda_q \quad \lambda_0 \quad \lambda_f \quad \lambda_{D1} \quad \lambda_{Q1} \quad \lambda_{Q2}]^{\mathrm{T}}$$
$$\boldsymbol{u} = [-\omega\lambda_q \quad \omega\lambda_d \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^{\mathrm{T}}$$

are vectors of voltages, currents, flux linkages, and speed voltages of the windings

$$\boldsymbol{R} = \mathrm{diag}[R_d \quad R_q \quad R_0 \quad R_f \quad R_{D1} \quad R_{Q1} \quad R_{Q2}]$$

is a diagonal matrix of winding resistances, and

$$\boldsymbol{L} = \begin{bmatrix} L_d & 0 & 0 & M_{df} & M_{dD1} & 0 & 0 \\ 0 & L_q & 0 & 0 & 0 & M_{qQ1} & M_{qQ2} \\ 0 & 0 & L_0 & 0 & 0 & 0 & 0 \\ M_{df} & 0 & 0 & L_f & M_{fD1} & 0 & 0 \\ M_{dD1} & 0 & 0 & M_{fD1} & L_{D1} & 0 & 0 \\ 0 & M_{qQ1} & 0 & 0 & 0 & L_{Q1} & M_{Q1Q2} \\ 0 & M_{qQ2} & 0 & 0 & 0 & M_{Q1Q2} & L_{Q2} \end{bmatrix}$$

is a symmetrical matrix of the winding leakage inductances with L and M denoting the self- and mutual inductances, respectively.

The electromagnetic torque is calculated using

$$T_e = \lambda_d i_q - \lambda_q i_d. \tag{3}$$

Instead of a mechanical model of the mass-shaft system in most machine models, the UM uses an equivalent electric network with lumped R, L, C elements to represent the mechanical part of machine [2]: This gives the greatest flexibility to model different mechanical systems using the existing electrical models in the EMTP framework. By replacing the mechanical parameters with their analog electrical quantities as shown in Table I, the differential equation (4) describing the dynamics of the rotor can be modeled by an electrical circuit. Fig. 2 shows an example of a single-mass mechanical system and its electrical analog

$$T_m = J\frac{d\omega}{dt} + D\omega + T_e. \tag{4}$$

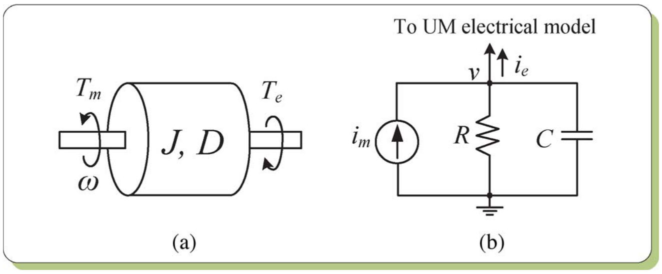


Fig. 2. (a) Mechanical system of rotor and (b) its electrical analog.

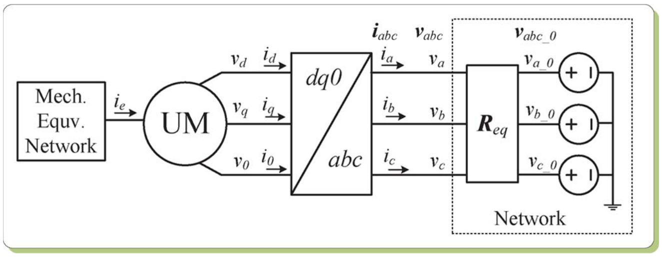


Fig. 3. Interfacing of the UM model to the network using the compensation method.

B. Interfacing UM Model With EMTP

To interface the UM model to the existing EMTP framework, the differential equation (1) is discretized using the trapezoidal rule of integration to give

$$\boldsymbol{v}_{dq0}(t) = -\boldsymbol{R}\boldsymbol{i}_{dq0}(t) - \frac{2}{\Delta t}\boldsymbol{\lambda}_{dq0}(t) + \boldsymbol{u}(t) + \boldsymbol{v}_{\text{hist}} \tag{5}$$

where Δt is the time step and $\boldsymbol{v}_{\text{hist}}$ is the history term expressed as

$$\boldsymbol{v}_{\text{hist}} = -\boldsymbol{v}_{dq0}(t-\Delta t) - \boldsymbol{R}\boldsymbol{i}_{dq0}(t-\Delta t) + \frac{2}{\Delta t}\boldsymbol{\lambda}_{dq0}(t-\Delta t) + \boldsymbol{u}(t-\Delta t). \tag{6}$$

The next step is to link the $dq0$ quantities of the machine with the abc phase quantities of the rest of the network. The Park's transformation matrix $\boldsymbol{P}$ is used for this purpose. For example

$$[i_d \quad i_q \quad i_0]^{\mathrm{T}} = \boldsymbol{P}[i_a \quad i_b \quad i_c]^{\mathrm{T}} \tag{7}$$

where $\boldsymbol{P}$ is an orthogonal matrix defined as

$$\boldsymbol{P} = \sqrt{\frac{2}{3}}\begin{bmatrix} \cos(\beta) & \cos(\beta - 120^0) & \cos(\beta + 120^0) \\ \sin(\beta) & \sin(\beta - 120^0) & \sin(\beta + 120^0) \\ 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \tag{8}$$

where β is the rotor angle.

To incorporate the UM model into the EMTP solution, the compensation method is employed. The network is first solved without the machine and represented by a Thevenin impedance $\boldsymbol{R}_{\text{eq}}$ and a voltage source $\boldsymbol{v}_{abc\_0}$ expressed as

$$\boldsymbol{v}_{abc} = \boldsymbol{R}_{\text{eq}}\boldsymbol{i}_{abc} + \boldsymbol{v}_{abc\_0}. \tag{9}$$

After transforming (9) into the $dq0$ frame, the machine equations are solved iteratively. Finally, the solved machine terminal currents i_a, i_b, and i_c are transformed back to the abc frame and superimposed into the original network to calculate the entire network voltages. Fig. 3 shows this compensation method.

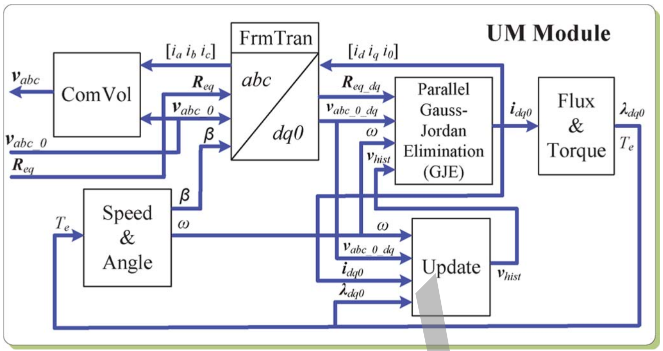


Fig. 4. `UM` module: FPGA implementation of the UM model.

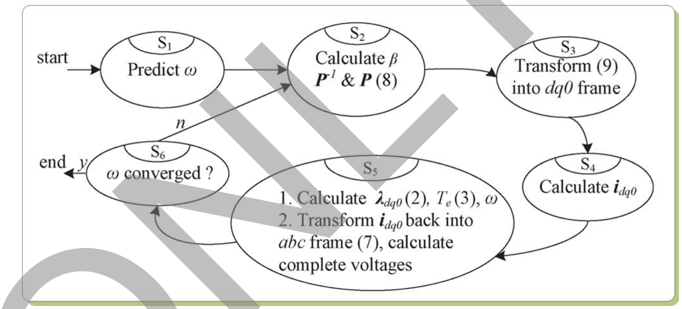

start
S_1
Predict ω
S_2
Calculate β
$\boldsymbol{P}^{1}$ & $\boldsymbol{P}$ (8)
S_3
Transform (9) into $dq0$ frame
S_4
Calculate $\boldsymbol{i}_{dq0}$
S_5
1. Calculate $\boldsymbol{\lambda}_{dq0}$ (2), T_e (3), ω
2. Transform $\boldsymbol{i}_{dq0}$ back into abc frame (7), calculate complete voltages
S_6
ω converged ?
n
y
end

Fig. 5. FSM of the iteration process of the UM model.

C. Real-Time FPGA Implementation of the UM Model

As shown in Fig. 4, a hardware module `UM` is designed to implement the UM model in the FPGA. It mainly consists of six functional units. The `Speed` and `Angle` unit is responsible for predicting and calculating rotor speed (ω) and rotor angle (β). The `FrmTran` unit transforms the UM quantities between the abc and $dq0$ frames. The `Parallel GJE` (Gauss–Jordan elimination) unit [24] is used to calculate the machine current $\boldsymbol{i}_{dq0}$. This procedure involves solving a set of seven algebraic equations for the seven winding representations of the machine. The flux linkages ($\boldsymbol{\lambda}_{dq0}$) and torque ($T_e, T_m$) are solved in the `Flux` and `Torque` unit. The `Update` unit updates machine history terms ($\boldsymbol{v}_{\text{hist}}$) and the history terms of the equivalent mechanical network. Finally, the `ComVol` unit calculates the complete voltages of the network.

To solve the nonlinear equations of the machine, iterations are required. The number of iterations is generally small (1–3) due to the relatively large inertia of machine. Fig. 5 shows the finite-state machine (FSM) diagram of the iteration process. When the network is solved without machines for applying the compensation method [32], the UM procedure starts with the prediction of rotor speed ω using linear extrapolation in state S_1. Then, in state S_2, the rotor angle β is calculated, followed by Park's transformation matrices P and P^{-1} (8). The P and P^{-1} matrices are calculated using a sinusoidal function lookup table. Very high accuracy has been achieved in this design by means of a 4096-point-long table for half cycle of the cos function and the linear interpolation technique [24]. The Thevenin equivalent for the rest of the network is then transformed into the abc frame in state S_3. Then, in state S_4, the current $\boldsymbol{i}_{dq0}$ is calculated using a parallel Gauss–Jordan elimination method. Once the $\boldsymbol{i}_{dq0}$ is available, the flux linkages

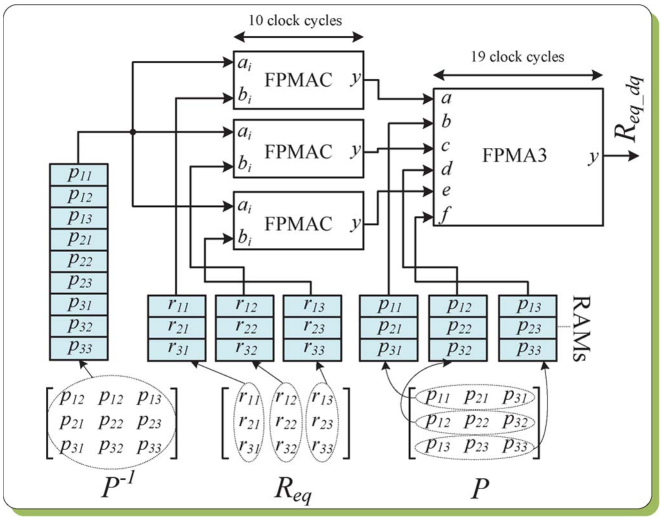


Fig. 6. Parallel computation of $\boldsymbol{R}_{\text{eq}\_dq} = \boldsymbol{P}^{-1}\boldsymbol{R}_{\text{eq}}\boldsymbol{P}$ in the frame transformation (FrmTran) unit of the UM module; FPMAC: $y = \sum a_i b_i$; and FPMA3: $y = ab + cd + ef$.

$\boldsymbol{\lambda}_{dq0}$, electromagnetic torque T_e, and rotor speed ω are computed sequentially in the $dq0$ frame in state S_5. Meanwhile, the $\boldsymbol{i}_{dq0}$ is transformed back to the abc frame to superimpose into the rest of the network to calculate the complete nodal voltages of the network in state S_5. The calculated ω is compared with the predicted ω in state S_6. If the difference is within the given tolerance, the iteration process is terminated; otherwise, the next iteration is started.

Due to its iterative nature, solving the UM model is very time consuming. Moreover, the transformation between the $dq0$ frame and the abc frame also introduces extra latency. To improve the computational efficiency of hardware emulation of the UM model, parallelism has to be considered. Although the overall procedure is sequential, there exist possible parallel processing paths. For instance, in state S_5 of Fig. 5, after $\boldsymbol{i}_{dq0}$ is calculated, the machine variables $\boldsymbol{\lambda}_{dq0}$, T_e, and ω are calculated in the $dq0$ frame, while the complete voltages are calculated in the abc frame simultaneously. Furthermore, parallel processing can also be applied into each individual calculation. For example, in the FrmTran unit, when transforming (9) into the $dq0$ frame, the Thevenin impedance $\boldsymbol{R}_{\text{eq}}$ becomes $\boldsymbol{R}_{eq\_dq} = \boldsymbol{P}^{-1}\boldsymbol{R}_{\text{eq}}\boldsymbol{P}$ in the $dq0$ frame. Fig. 6 shows its parallel computation scheme. First, various RAMs are defined in the FPGA to save the respective matrices. This is an important feature of FPGA that many different memory types (e.g., ROM, single-port RAM, and dual-port RAM) with any width and length can be defined customarily, which makes parallel processing possible because multiple parameters are available to be accessed simultaneously. Here, the matrix $\boldsymbol{P}^{-1}$ is stored in a single RAM; the matrix $\boldsymbol{R}_{\text{eq}}$ is stored in three individual RAMs in a column-wise format; and $\boldsymbol{P}$ is also stored in three individual RAMs but in a row-wise format. The matrix multiplication of $\boldsymbol{P}^{-1}\boldsymbol{R}_{\text{eq}}$ is performed by three floating-point multiply-accumulate (FPMAC) units in parallel. Thus, one row of $\boldsymbol{P}^{-1}$ can multiply three columns of $\boldsymbol{R}_{\text{eq}}$, resulting in three elements in one row of the product matrix simultaneously. Then, these intermediate results, along with three elements in one column of matrix $\boldsymbol{P}$, are multiplied using a floating-point multiply-add (FPMA3) unit, resulting in one element of the final product matrix.

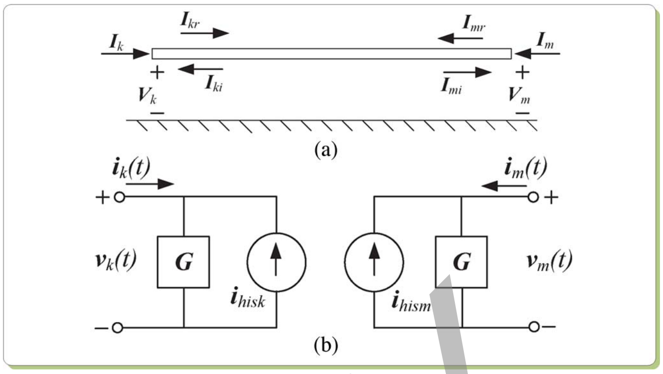


Fig. 7. (a) n-phase transmission line and (b) its time-domain representation.

III. ULM Hardware Emulation

A. ULM Model Formulation in Frequency Domain

For an n-phase transmission line as shown in Fig. 7(a), the solution of the traveling wave equations can be expressed in the frequency domain at the sending end "k" and the receiving end "m" as follows:

$$\boldsymbol{I}_k = \boldsymbol{Y}_c\boldsymbol{V}_k - 2\boldsymbol{I}_{ki} = \boldsymbol{Y}_c\boldsymbol{V}_k - 2\boldsymbol{H}\boldsymbol{I}_{mr} \tag{10a}$$

$$\boldsymbol{I}_m = \boldsymbol{Y}_c\boldsymbol{V}_m - 2\boldsymbol{I}_{mi} = \boldsymbol{Y}_c\boldsymbol{V}_m - 2\boldsymbol{H}\boldsymbol{I}_{kr}. \tag{10b}$$

In the aforementioned equations, $\boldsymbol{I}_k$, $\boldsymbol{V}_k$ and $\boldsymbol{I}_m$, $\boldsymbol{V}_m$ are the n-dimensional current and voltage vectors at both line ends, respectively. $\boldsymbol{I}_{ki}$ and $\boldsymbol{I}_{mi}$ are the incident currents, whereas $\boldsymbol{I}_{kr}$ and $\boldsymbol{I}_{mr}$ are the reflected currents. The two $n \times n$ matrix transfer functions, which characterize a transmission line, are the characteristic admittance matrix $\boldsymbol{Y}_c$ and the propagation matrix $\boldsymbol{H}$ expressed as

$$\boldsymbol{Y}_c = \sqrt{\boldsymbol{Y}/\boldsymbol{Z}} \tag{11}$$

$$\boldsymbol{H} = e^{-\sqrt{(\boldsymbol{YZ})}l} \tag{12}$$

where $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are the $n \times n$ shunt admittance and series impedance matrices per unit length, respectively, and l is the line length.

In order to implement the model in time domain, the elements of $\boldsymbol{Y}_c$ and $\boldsymbol{H}$ are approximated with rational functions. The fitting method used here is *vector fitting* [33]. The elements of $\boldsymbol{Y}_c$ are, in general, very smooth and can be easily fitted in the phase domain. The (i, j) element of $\boldsymbol{Y}_c$ is expressed as

$$\boldsymbol{Y}_{c,(i,j)}(s) = \sum_{m=1}^{N_p} \frac{\boldsymbol{r}_{Y_c,(i,j)}(m)}{s - \boldsymbol{p}_{Y_c}(m)} + \boldsymbol{d}_{(i,j)} \tag{13}$$

where N_p is the number of poles and $\boldsymbol{r}_{Y_c}$, $\boldsymbol{p}_{Y_c}$, and $\boldsymbol{d}$ are the residues, poles, and proportional terms, respectively. Note that all elements of $\boldsymbol{Y}_c$ have identical poles $\boldsymbol{p}_{Y_c}$.

The fitting of the propagation matrix $\boldsymbol{H}$ is somewhat different. It is first fitted in the modal domain with poles and time delays in each mode, followed by final fitting in the phase domain. The (i,j) element of $\boldsymbol{H}$ is expressed as

$$\boldsymbol{H}_{(i,j)}(s) = \sum_{k=1}^{N_g} \left(\sum_{n=1}^{N_{p,k}} \frac{\boldsymbol{r}_{H,(i,j),k}(n)}{s - \boldsymbol{p}_{H,k}(n)} \right) e^{-s\tau_k} \tag{14}$$

where N_g denotes the number of modes and $N_{p,k}$ and τ_k are the number of poles and time delay used for fitting the kth mode. $\boldsymbol{r}_{H,k}$ and $\boldsymbol{p}_{H,k}$ are the residues and poles for the kth mode. Again, the poles are identical for all elements in each mode.

B. Time-Domain Representation

By transforming (10a) and (10b) into the time domain using inverse Fourier transform, the Norton equivalent circuit for the ULM model is obtained, as shown in Fig. 7(b). The history currents $\boldsymbol{i}_{his_k}$ and $\boldsymbol{i}_{his_m}$ and equivalent impedance matrix $\boldsymbol{G}$ are expressed as

$$\boldsymbol{i}_{his_k} = \boldsymbol{Y}_c * \boldsymbol{v}_k(t) - 2\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau) \tag{15a}$$

$$\boldsymbol{i}_{his_m} = \boldsymbol{Y}_c * \boldsymbol{v}_m(t) - 2\boldsymbol{H} * \boldsymbol{i}_{kr}(t-\tau) \tag{15b}$$

$$\boldsymbol{G} = \boldsymbol{d} + \boldsymbol{r}_{Y_c}\boldsymbol{\lambda}_{Y_c} \tag{16}$$

where the symbol "$*$" denotes the matrix–vector convolution. The coefficient $\boldsymbol{\lambda}_{Y_c}$ is defined as

$$\boldsymbol{\lambda}_{Y_c} = \left(\frac{\Delta t}{2}\right) \Big/ \left(1 - \boldsymbol{p}_{Y_c}\frac{\Delta t}{2}\right). \tag{17}$$

To perform the convolution of $\boldsymbol{Y}_c * \boldsymbol{v}_k(t)$, a state variable $\boldsymbol{x}_{Y_c}$ is defined as in (18), and the convolution is then computed using (19)

$$\boldsymbol{x}_{Y_c}(t) = \boldsymbol{\alpha}_{Y_c}\boldsymbol{x}_{Y_c}(t-\Delta t) + \boldsymbol{v}_k(t-\Delta t) \tag{18}$$

$$\boldsymbol{Y}_c * \boldsymbol{v}_k(t) = \boldsymbol{c}_{Y_c}\boldsymbol{x}_{Y_c}(t) \tag{19}$$

where the coefficients $\boldsymbol{\alpha}_{Y_c}$ and $\boldsymbol{c}_{Y_c}$ are expressed as

$$\boldsymbol{\alpha}_{Y_c} = \left(1 + \boldsymbol{p}_{Y_c}\frac{\Delta t}{2}\right) \Big/ \left(1 - \boldsymbol{p}_{Y_c}\frac{\Delta t}{2}\right) \tag{20}$$

$$\boldsymbol{c}_{Y_c} = \boldsymbol{r}_{Y_c}(\boldsymbol{\alpha}_{Y_c} + 1)\boldsymbol{\lambda}_{Y_c}. \tag{21}$$

The convolution of $\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau)$ is similar except that linear interpolation is used to accurately calculate $\boldsymbol{i}_{mr}(t-\tau)$ since the travel time τ is normally not an integer multiple of the time step Δt.

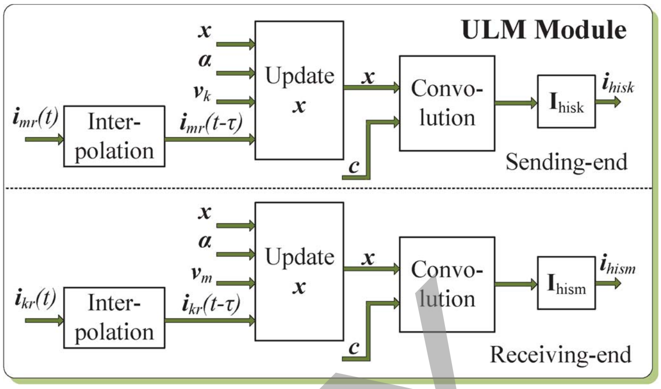


Fig. 8. ULM module: FPGA implementation of the ULM model showing parallel computation of sending- and receiving-end quantities.

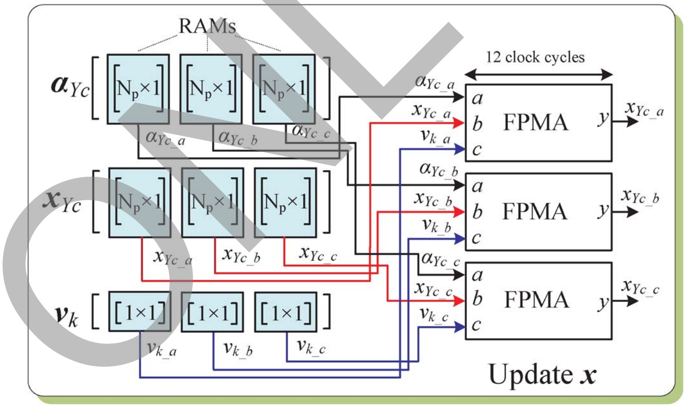


Fig. 9. Parallel computation scheme in Update $\boldsymbol{x}$ unit of the ULM module (Fig. 8) and FPMA: $y = ab + c$.

C. Real-Time FPGA Implementation of the ULM Model

As shown in (15a) and (15b), the main computations involved in the ULM are two convolutions at each end of the line. Depending on the number of phases of the transmission line and the number of poles of fitted rational functions, the convolution is, in general, quite expensive computationally. Parallel computation is essential to improve the computational efficiency. A hardware module ULM is designed in the FPGA, as shown in Fig. 8. It can first be seen that two identical parts are implemented, for sending end and receiving end of transmission line, respectively. Thus, the calculations at both ends can be conducted fully in parallel. The Interpolation unit is responsible for calculating $\boldsymbol{i}_{mr}(t-\tau)$. The Update $\boldsymbol{x}$ unit is used to update the state variable (18), and the Convolution unit is responsible for convolution (19). These two units are used for two convolutions: This is because the convolution of $\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau)$ is independent of the node voltages, so that it can be carried out before the network is solved, whereas the convolution of $\boldsymbol{Y}_c * \boldsymbol{v}_k(t)$ must be processed after the network is solved. This can also be seen clearly in Fig. 12 where the two convolutions are done separately in two different periods (update history term, part (a) and part (b) in *Period 3* and *Period 4*, respectively) of a simulation time step. In this way, the computation time can be reduced significantly. Finally, the two convolutions are added together to obtain the line history term $\boldsymbol{i}_{his_k}$ in the Ihisk unit.

The computational efficiency can be improved even further for multiphase transmission lines where the calculation in each phase can be executed simultaneously. Fig. 9 shows the parallel computation scheme of (18) in the Update $\boldsymbol{x}$ unit for a three-phase transmission line. Here, the state variable $\boldsymbol{x}_{Y_c}$ and coefficient $\boldsymbol{\alpha}_{Y_c}$ are $N_p \times 3$ matrices, whereas the line voltage $\boldsymbol{v}_k$ is a 1×3 vector. Each column corresponds to one phase denoted by subscripts $\{a, b, c\}$. All matrices are stored in three individual RAMs in a column-wise format. The $\boldsymbol{x}_{Y_c}$ is calculated using three FPMA units for three phases in parallel. The updated $\boldsymbol{x}_{Y_c}$

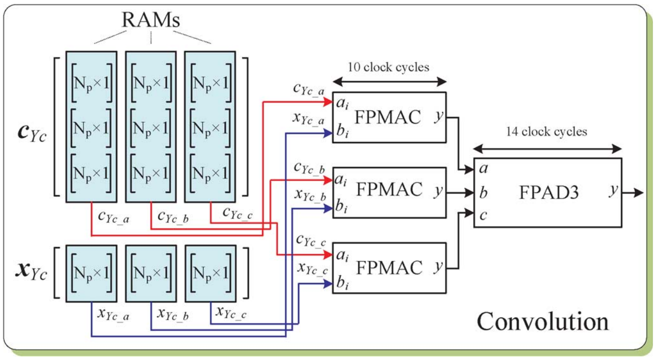


Fig. 10. Parallel computation scheme in Convolution unit of the ULM module (Fig. 8); FPMAC: $y = \sum a_i b_i$; and FPAD3: $y = a + b + c$.

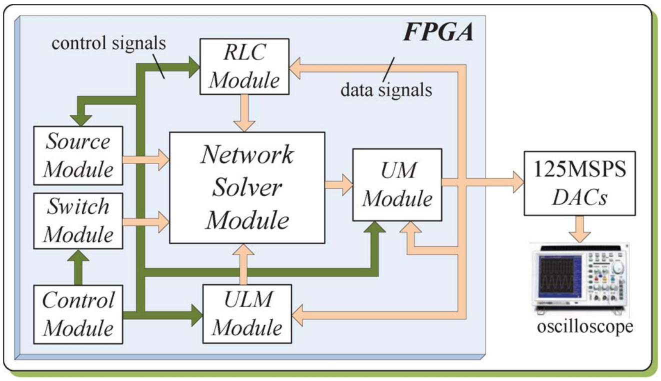


Fig. 11. Overall hardware architecture of the real-time network emulator.

is sent back to the RAMs which are dual port that support the “read” and “write” functions simultaneously.

Once the $\boldsymbol{x}_{Y_c}$ is updated, the convolution of $\boldsymbol{Y}_c * \boldsymbol{v}_k(t)$ can be carried out. Fig. 10 shows the parallel computation scheme of (19) in the Convolution unit for a three-phase transmission line. Here, the coefficient $\boldsymbol{c}_{Y_c}$ is a $3N_p \times 3$ matrix and stored in three RAMs in a column-wise format. Three FPMAC units are used for calculations in the three phases in parallel, when a connected floating-point add (FPAD3) combines them to get the final result of convolution. Note that this final result is not available for three phases at the same time. It comes sequentially as phase a, phase b, and phase c. Full parallelism is possible if more FPMAC and FPAD3 units are emulated subject to FPGA resource utilization.

IV. Network Hardware Emulation

A. Hardware Architecture and Parallelism

The hardware emulation of the ULM and UM models, along with the network solution, is realized on a single FPGA platform. Since the electromagnetic transient simulation requires very high accuracy to capture the transients in detail, the IEEE 32-b floating-point data representation is utilized in this design. The UM and ULM models were implemented in the UM and ULM modules, respectively, as shown in Figs. 4 and 8. The remaining modules that assist in the network solution include the following (see Fig. 11): 1) Source module; 2) Switch module; 3) RLC module; 4) Network Solver module; and 5) Control module.

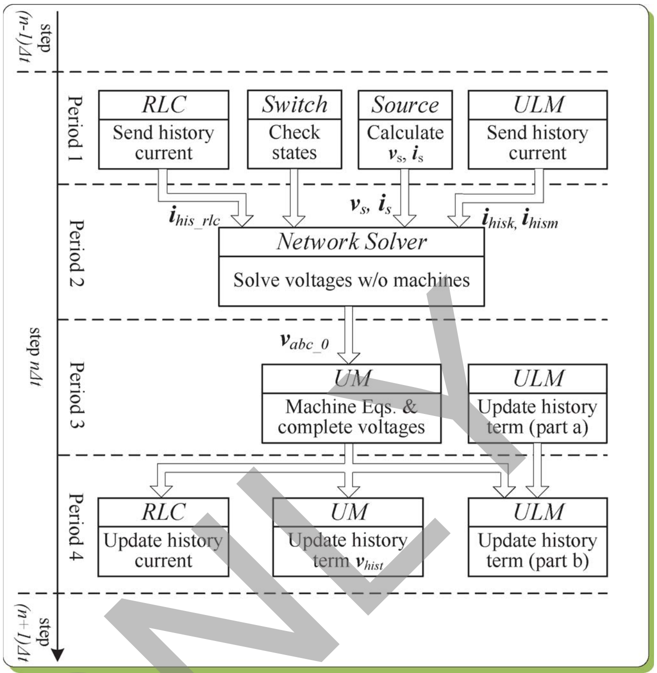


Fig. 12. Operations within one time step of the real-time network emulator.

The Source module is responsible for calculating known voltage and current sources. This is basically a sinusoidal function evaluation which uses the same method as for calculating Park’s transformation matrices. In the RLC module, all linear lumped R, L, C elements are represented as their equivalent conductances in parallel with history current sources. The history currents are sent to the Network Solver module to assemble the current injection vector and updated after the node voltages are calculated in each time step. The Network Solver module is responsible for solving the network node voltages as $\boldsymbol{YV} = \boldsymbol{I}$, where $\boldsymbol{Y}$, $\boldsymbol{V}$, and $\boldsymbol{I}$ are the network admittance matrix, node voltage vector, and current injection vector, respectively. Due to the difficulty of fast solution of a large set of linear algebraic equations in real time, the voltages are calculated by multiplying precalculated inverse system nodal admittance matrix by the nodal current injection vector. When there are switches in the network, the inverse system nodal admittance matrix corresponding to all possible switch combinations is precalculated. The Switch module checks the switch states in every time step, and then, Network Solver module chooses the corresponding matrix according to the current switch state. The Control module oversees the operations of all other modules by sending control signals to carry out their respective computations.

Fig. 12 shows the overall procedure in a simulation time step in the proposed hardware. There are four periods in a simulation time step. In *Period 1*, the Source module calculates the known voltage and current sources for the Network Solver module; the RLC and ULM modules send out their corresponding history current terms to the Network Solver module; and the Switch module checks the switches state and sends it to the Network Solver module. During *Period 2*, the node voltages are solved without taking into account the electric machine equations. In *Period 3*, the UM module starts to solve

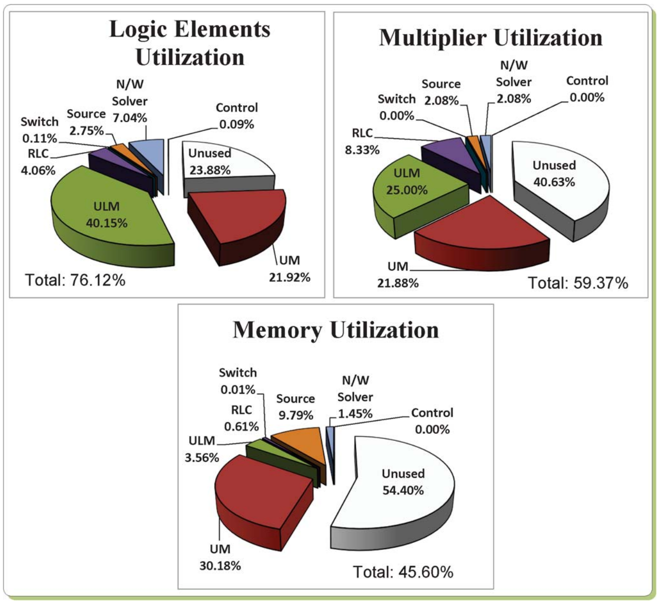


Fig. 13. FPGA resources utilized by modules of the real-time network emulator.

its equations, and the complete node voltages are calculated. Meanwhile, the ULM module computes the convolution of $\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau)$ (updating history term, part a). Finally, the upgrading of history terms in the RLC and UM modules and the calculation of $\boldsymbol{Y}_c * \boldsymbol{v}_k(t)$ and $\boldsymbol{i}_{his_k}$, $\boldsymbol{i}_{his_m}$ (updating history term, part b) in the ULM module are carried out in *Period 4*. From the aforementioned procedure, it is obvious that parallel processing exists in each period while preserving the necessary sequential periods in the overall transient simulation algorithm.

B. FPGA Resource Utilization

The real-time electromagnetic transient network emulator was implemented on an Altera Stratix III EP3SL150. This FPGA has the following main features: 142 500 logic elements (LEs); 6390 total RAM kbits; 384 18 × 18-bit multipliers; 8 phase-locked loops, and 744 maximum user I/O pins.

Fig. 13 shows the FPGA hardware resource utilization generated by the Altera Quartus II software in percentage for each module of the real-time network emulator. As can be seen from this figure, the ULM module utilizes the most logic resource of the FPGA, and the UM module has the second highest resource consumption in the FPGA.

V. Real-Time Simulation Case Study

To show the effectiveness of the proposed hardware design, an example power system is simulated. The system consists of three UM synchronous generators, two ULM lines, three transformers, and three loads as shown in Fig. 14. The complete system data are listed in Appendix. The transformers are modeled as equivalent leakage reactances only. Each ULM utilizes ninth-order rational functions to fit its characteristic admittance matrix and propagation matrix. Each UM is a full-order model as described in Section II-A.

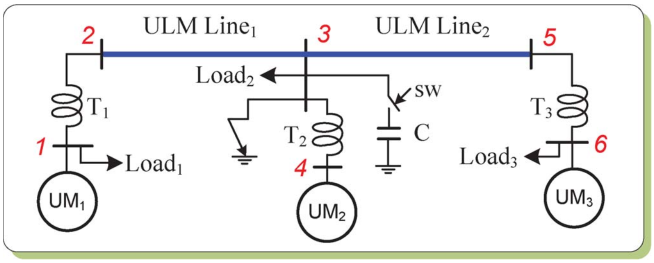


Fig. 14. Single-line diagram of power system for the case study.

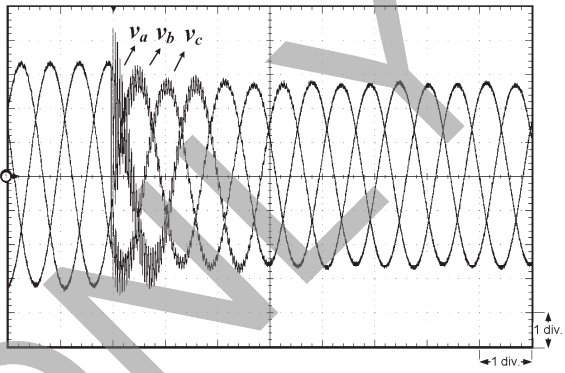


Fig. 15. Real-time oscilloscope trace of the three-phase voltages at Bus 2 during a three-phase fault at Bus 3 (x-axis: 1 div. = 10 ms, y-axis: 1 div. = 5.2 kV).

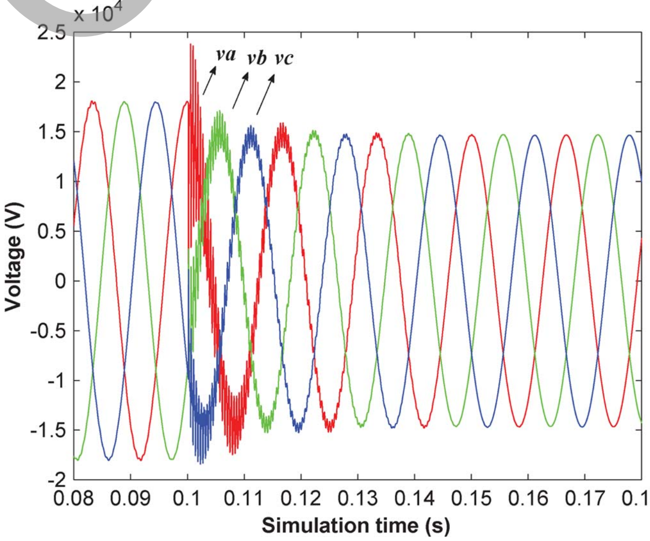


Fig. 16. Offline EMTP-RV simulation of three-phase voltages at Bus 2 during a three-phase fault at Bus 3 ($t = 0.1$ s).

Two transient events are simulated. The first transient event is a three-phase-to-ground fault at Bus 3 which occurs at $t = 0.1$ s. Fig. 15 shows the three-phase voltage waveforms at Bus 2 captured from a real-time oscilloscope connected to the 125MSPS DACs. Identical behavior can be observed from Fig. 16 which shows the offline EMTP-RV simulation with a time step of 8 μs. The second transient event is the capacitor C switched at Bus 3 at time $t = 0.1$ s. Figs. 17 and 18 show the three-phase voltage waveforms at Bus 3 and electromagnetic torque of UM_2 from the real-time oscilloscope. Figs. 19 and 20 show the corresponding transient waveforms obtained from the offline simulation. Again, a detailed agreement between the

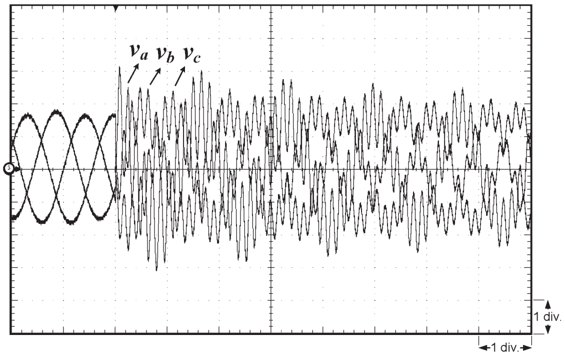


Fig. 17. Real-time oscilloscope trace of the three-phase voltages at Bus 3 during a capacitor switching at Bus 3 (x-axis: 1 div. $=$ 10 ms, y-axis: 1 div. $=$ 10.4 kV).

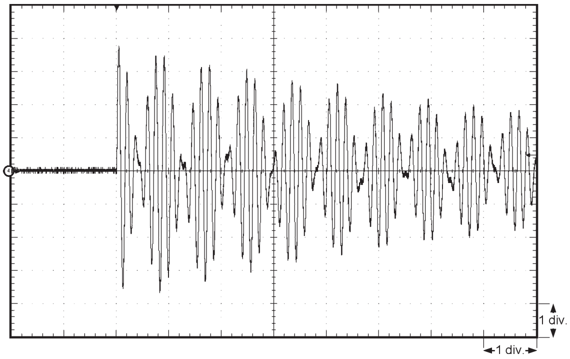


Fig. 18. Real-time oscilloscope trace of the electromagnetic torque of UM_2 during a capacitor switching at Bus 3 (x-axis: 1 div. $=$ 10 ms, y-axis: 1 div. $=$ 4.7 kN · m).

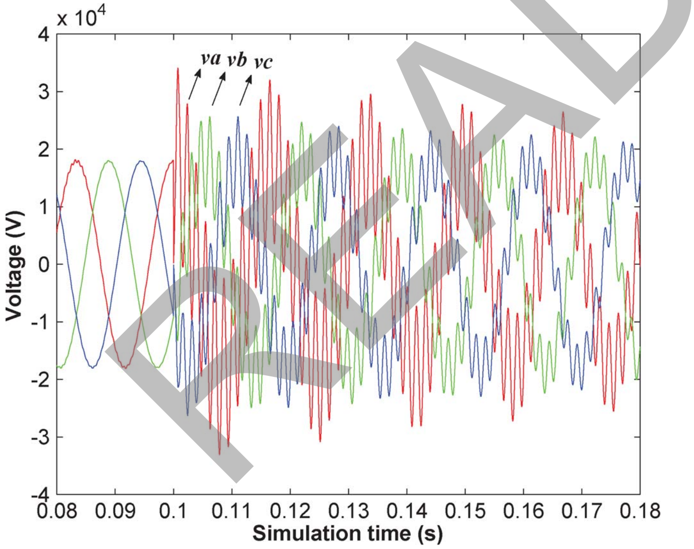


Fig. 19. Offline EMTP-RV simulation of three-phase voltages at Bus 3 during a capacitor switching at Bus 3 ($t = 0.1$ s).

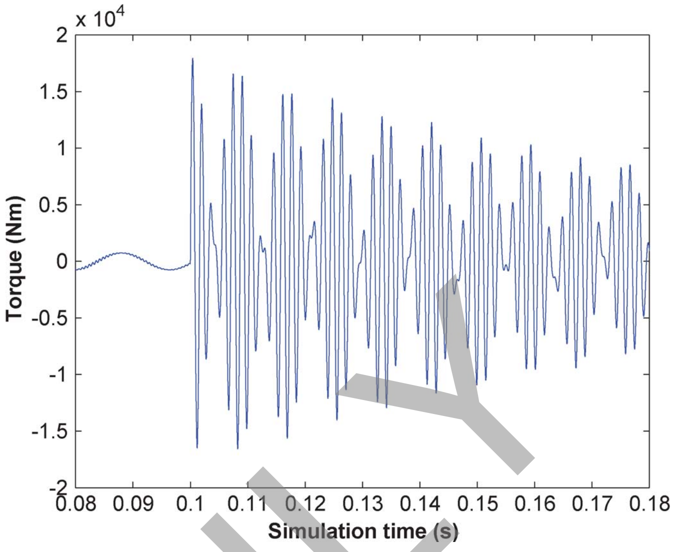


Fig. 20. Offline EMTP-RV simulation of electromagnetic torque of UM_2 during a capacitor switching at Bus 3 ($t = 0.1$ s).

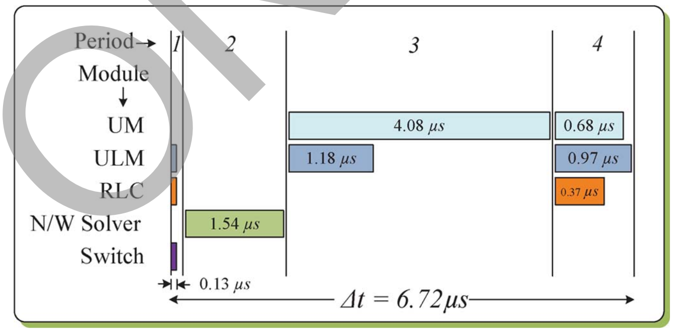
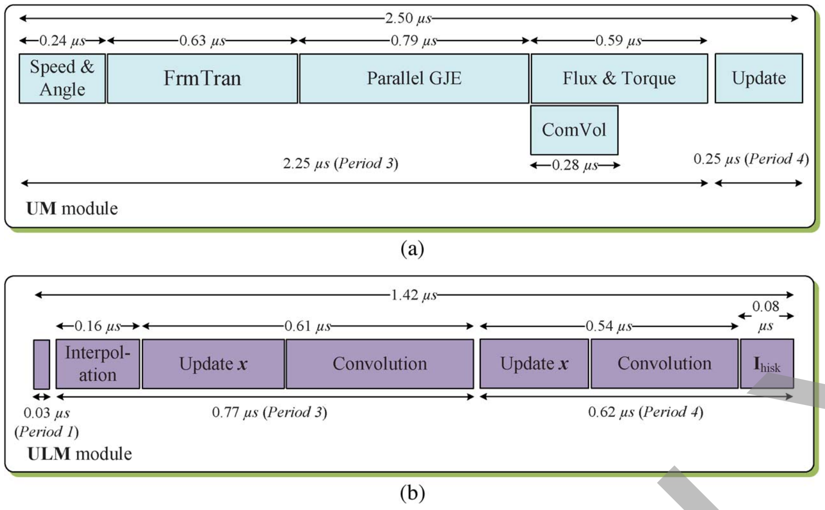


Fig. 21. Breakdown of one time step (Δt) in microseconds for various periods and modules for the case study.

EMTP-RV offline simulation and the real-time electromagnetic transient network emulator results can be observed. Due to the 14-b resolution of the DACs and their consequent limited output range, small discrepancies exist such as the initial low-frequency torque oscillation shown in Fig. 20 (offline simulation), which was truncated from Fig. 17 (real-time simulation). Nevertheless, when the transient data in a 32-b floating-point format were stored in the memory and plotted, the low-frequency torque oscillation was visible.

The achieved time step in this implementation is 6.72 μs based on an FPGA clock frequency of 130 MHz. With this time step, transmission line transients up to 75 kHz can be accurately reproduced. A detailed breakdown of this time step for various periods of computation (Fig. 12) and modules is shown in Fig. 21. The computation for the UM equations in *Period 3* is most time consuming, and the updating of history terms in the `ULM` module has the second highest latency. The detailed execution times of one UM generator and one ULM line are shown in Fig. 22(a) and (b), respectively. The total latency for one UM computation is 2.5 μs, and that for one ULM computation is 1.42 μs. The computation of the three UM generators in the system is pipelined through one `UM` module, while the computation of the two ULM lines is pipelined through one `ULM` module. As shown in Fig. 13, the FPGA is almost 80% full on LEs and 60% full on the multipliers. Depending on the size of the system and the time step required for a certain transient, more UM and ULM modules can be replicated on a larger FPGA to either reduce the time step or accommodate a large system. Lower data precision can also result in a reduction of time step and hardware resource cost, however, at reduced transient accuracy.



(a)

(b)

Fig. 22. Detailed execution time of (a) one UM generator in the UM module (Fig. 4) and (b) one ULM line in the ULM module (Fig. 8).

VI. Conclusion

Accurate electromagnetic transient simulation of power systems requires detailed and generalized models of rotating machine and transmission line. The UM model is a sufficiently generalized machine model which can accurately represent several types of rotating machines for transient studies. The ULM is the most accurate and general model for both overhead line and underground cable in both symmetrical and unsymmetrical scenarios. These two models have been implemented in offline simulation programs successfully; however, due to the complexity of UM and ULM models and limited computational power of traditional sequential hardware, the real-time implementation of UM and ULM models is challenging, often requiring a compromise on the system size in order to meet the real-time step-size constraint to model a specific transient. This paper has proposed a digital hardware emulation of UM and ULM models for real-time electromagnetic transient simulation on the FPGA. Taking advantages of the inherent parallel architecture of FPGA, the proposed hardware is paralleled and fully pipelined to achieve efficient real-time simulation of electromagnetic transients. An example system with three UM and two ULM models is simulated within a 6.72-μs time step on a 130-MHz FPGA clock frequency. The achieved latencies for one UM module and one ULM module computation are 2.5 μs and 1.42 μs, respectively. Floating-point representation with a 32-b data precision is employed in this hardware design to maintain high accuracy of the emulated transients. The captured real-time waveforms are validated by offline EMTP-RV simulation results.

Appendix

I) ULM transmission line ($Line_1$ and $Line_2$) parameters: three conductors, diameter: 3.105 cm, resistance: 0.0583 Ω/km, and 150 km. The tower geometry is shown in Fig. 23. $\boldsymbol{Y}_c(s)$ and $\boldsymbol{H}(s)$ are 3×3 matrices, and their elements are approximated with ninth-order rational functions [(13) and (14)].

II) UM synchronous machine (UM_1, UM_2, and UM_3) parameters: 1000 MVA, 22 kV, Y-connected, field current: 2494 A, 2 poles, 60 Hz, single-mass, $J = 5.5628\text{e}4$ [kg $\cdot$ m^2], and $D = 0.678\text{e}4$ [N $\cdot$ m $\cdot$ s/rad]. The winding resistances (in ohms) and leakage inductances (in henries) are listed in Table II.

III) Loads and transformer parameters: $Load_1$, $Load_2$, and $Load_3$: 5 Ω and 4 H; X_{T1}, X_{T2}, and X_{T3}: 15 mH; and C: 20 μF.

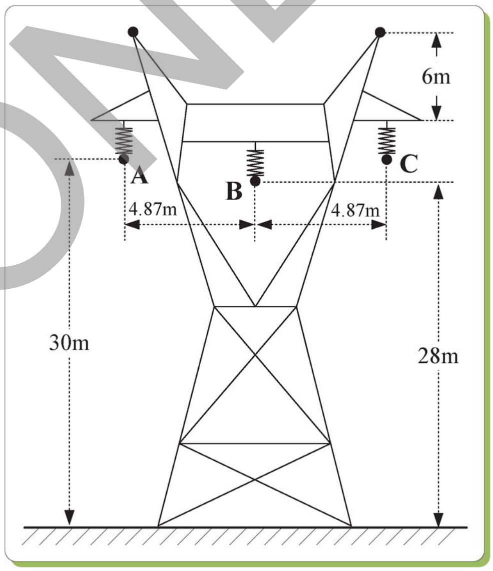


Fig. 23. Tower geometry of transmission lines in the case study.

TABLE II
UM Parameters

R_d	9.6800e-4	R_q	9.6800e-4	R_0	9.6800e-4
R_f	1.1109	R_{D1}	3.4985	R_{Q1}	0.7627
R_{Q2}	1.2270				
L_d	0.0018	L_q	0.0017	L_0	2.4136e-4
L_f	0.6345	L_{D1}	0.5483	L_{Q1}	1.1811
L_{Q2}	0.5882	M_{df}	0.0234	M_{dD1}	0.0234
M_{qQ1}	0.0226	M_{qQ2}	0.0226	M_{fD1}	0.5480
M_{Q1Q2}	0.4184				

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. K. Lauw and W. Scott Meyer, “Universal machine modeling for the representation of rotating electric machinery in an electromagnetic transients program,” *IEEE Trans. Power App. Syst.*, vol. PAS-101, no. 6, pp. 1342–1351, Jun. 1982.

[3] H. K. Lauw, “Interfacing for universal multi-machine system modeling in an electromagnetic transients program,” *IEEE Trans. Power App. Syst.*, vol. PAS-104, no. 9, pp. 2367–2373, Sep. 1985.

[4] J. R. Marti, “Accurate modeling of frequency-dependent transmission lines in electromagnetic transients simulations,” *IEEE Trans. Power App. Syst.*, vol. PAS-101, no. 1, pp. 147–157, Jan. 1982.

[5] L. Marti, “Simulation of transients in underground cables with frequency-dependent modal transformation matrices,” *IEEE Trans. Power Del.*, vol. 3, no. 3, pp. 1099–1110, Jul. 1988.

[6] T. Noda, N. Nagaoka, and A. Ametani, “Phase domain modeling of frequency-dependent transmission lines by means of an ARMA model,” *IEEE Trans. Power Del.*, vol. 11, no. 1, pp. 401–411, Jan. 1996.

[7] H. V. Nguyen, H. W. Dommel, and J. R. Marti, “Direct phase-domain modeling of frequency-dependent overhead transmission lines,” *IEEE Trans. Power Del.*, vol. 12, no. 3, pp. 1335–1342, Jul. 1997.

[8] A. Morched, B. Gustavsen, and M. Tartibi, “A universal model for accurate calculation of electromagnetic transients on overhead lines and underground cables,” *IEEE Trans. Power Del.*, vol. 14, no. 3, pp. 1032–1038, Jul. 1999.

[9] D. Puyal, L. A. Barragan, J. Acero, J. M. Burdio, and I. Millan, “An FPGA-based digital modulator for full- or half-bridge inverter control,” *IEEE Trans. Power Electron.*, vol. 21, no. 5, pp. 1479–1483, Sep. 2006.

[10] G. Parma and V. Dinavahi, “Real-time digital hardware simulation of power electronics and drives,” *IEEE Trans. Power Del.*, vol. 22, no. 2, pp. 1235–1246, Apr. 2007.

[11] F.-J. Lin, C.-K. Chang, and P.-K. Huang, “FPGA-based adaptive backstepping sliding-mode control for linear induction motor drive,” *IEEE Trans. Power Electron.*, vol. 22, no. 4, pp. 1222–1231, Jul. 2007.

[12] E. Monmasson and M. N. Cirstea, “FPGA design methodology for industrial control systems—A review,” *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1824–1842, Aug. 2007.

[13] M.-W. Naouar, E. Monmasson, A. A. Naassani, I. Slama-Belkhodja, and N. Patin, “FPGA-based current controllers for AC machine drives—A review,” *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1907–1925, Aug. 2007.

[14] M. N. Cirstea and A. Dinu, “A VHDL holistic modeling approach and FPGA implementation of a digital sensorless induction motor control scheme,” *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1853–1864, Aug. 2007.

[15] X. Lin-Shi, F. Morel, A. M. Llor, B. Allard, and J.-M. Retif, “Implementation of hybrid control for motor drives,” *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1946–1952, Aug. 2007.

[16] H. Hu, W. Yao, and Z. Lu, “Design and implementation of three-level space vector PWM IP core for FPGAs,” *IEEE Trans. Ind. Electron.*, vol. 22, no. 6, pp. 2234–2244, Nov. 2007.

[17] E. Duman, H. Can, and E. Akin, “Real time FPGA implementation of induction machine model—A novel approach,” in *Proc. Int. ACEMP*, 2007, pp. 603–606.

[18] D. Zhang and H. Li, “A stochastic-based FPGA controller for an induction motor drive with integrated neural network algorithms,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 2, pp. 551–561, Feb. 2008.

[19] O. Lopez, J. Alvarez, J. Doval-Gandoy, F. D. Freijedo, A. Nogueiras, A. Lago, and C. M. Penalver, “Comparison of the FPGA implementation of two multilevel space vector PWM algorithms,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 4, pp. 1537–1547, Apr. 2008.

[20] F.-J. Lin, L.-T. Teng, C.-Y. Chen, and C.-K. Chang, “Robust RBFN control for linear induction motor drive using FPGA,” *IEEE Trans. Power Electron.*, vol. 23, no. 4, pp. 2170–2180, Jul. 2008.

[21] M. W. Naouar, A. Ammar Naassani, E. Monmasson, and I. S. Belkhodja, “FPGA-based predictive current controller for synchronous machine speed drive,” *IEEE Trans. Power Electron.*, vol. 23, no. 4, pp. 2115–2126, Jul. 2008.

[22] S. Karimi, A. Gaillard, P. Poure, and S. Saadate, “FPGA-based real-time power converter failure diagnosis for wind energy conversion systems,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 12, pp. 4299–4308, Dec. 2008.

[23] Y. Chen and V. Dinavahi, “FPGA-based real-time EMTP,” *IEEE Trans. Power Del.*, vol. 24, no. 2, pp. 892–902, Apr. 2009.

[24] Y. Chen and V. Dinavahi, “An iterative real-time nonlinear electromagnetic transient solver on FPGA,” *IEEE Trans. Ind. Electron.*, vol. 58, no. 6, pp. 1–8, Jun. 2010.

[25] A. Sathyan, N. Milivojevic, L. Young-Joo, M. Krishnamurthy, and A. Emadi, “An FPGA-based novel digital PWM control scheme for BLDC motor drives,” *IEEE Trans. Ind. Electron.*, vol. 56, no. 8, pp. 3040–3049, Aug. 2009.

[26] L. Idkhajine, E. Monmasson, M. W. Naouar, A. Prata, and K. Bouallaga, “Fully integrated FPGA-based controller for synchronous motor drive,” *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4006–4017, Oct. 2009.

[27] S. Karimi, P. Poure, and S. Saadate, “An HIL-based reconfigurable platform for design, implementation, and verification of electrical system digital controllers,” *IEEE Trans. Ind. Electron.*, vol. 57, no. 4, pp. 1226–1236, Apr. 2010.

[28] Q. N. Le and J. Jae-Wook, “Neural-network-based low-speed-damping controller for stepper motor with an FPGA,” *IEEE Trans. Ind. Electron.*, vol. 57, no. 9, pp. 3167–3180, Sep. 2010.

[29] T. O. Bachir, J.-P. David, C. Dufour, and J. Bélanger, “Effective FPGA-based electric motor modeling with floating-point cores,” in *Proc. 36th Annu. Conf. IEEE IECON*, 2010, pp. 829–834.

[30] A. Myaing and V. Dinavahi, “FPGA-based real-time emulation of power electronic systems with detailed representation of device characteristics,” *IEEE Trans. Ind. Electron.*, vol. 58, no. 1, pp. 358–368, Jan. 2011.

[31] M. Matar and R. Iravani, “Massively parallel implementation of AC machine models for FPGA-based real-time simulation of electromagnetic transients,” *IEEE Trans. Power Del.*, vol. 26, no. 2, pp. 830–840, Apr. 2011.

[32] H. W. Dommel, “Nonlinear and time-varying elements in digital simulation of electromagnetic transients,” *IEEE Trans. Power App. Syst.*, vol. PAS-90, no. 6, pp. 2561–2567, Nov. 1971.

[33] B. Gustavsen and A. Semlyen, “Simulation of transmission line transients using vector fitting and modal decomposition,” *IEEE Trans. Power Del.*, vol. 13, no. 2, pp. 605–614, Apr. 1998.

[34] B. Gustavsen, G. Irwin, R. Mangelrod, D. Brandt, and K. Kent, “Transmission line models for the simulation of interaction phenomena between parallel AC and DC overhead lines,” in *Proc. IPST*, Budapest, Hungary, Jun. 1999, pp. 61–68.

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. He is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, 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 (S’94–M’00–SM’08) received the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2000.

He is currently a Professor with the University of Alberta, Edmonton, AB, Canada. His research interests include real-time simulation of power systems and power electronic systems, large-scale system simulation, and parallel and distributed computing.