Electric Power Systems Research 190 (2021) 106716

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

High performance computing engines for the FPGA-based simulation of the ULM

Tarek Ould-Bachir[a,*,1], Hossein Chalangar[b,1], Keyhan Sheshyekani[b], Jean Mahseredjian[b]

[a] Department of Computer Engineering and Software Engineering, Polytechnique Montréal, Montreal, Canada
[b] Department of Electrical Engineering, Polytechnique Montréal, Montreal, Canada

ARTICLE INFO

Keywords:
FPGA-Based simulation
Universal line model
Real-time simulation
Travelling-wave fault location

ABSTRACT

This paper presents a design methodology for the FPGA-based simulation of the Universal Line Model (ULM). The proposed approach yields a higher computational performance compared to alternative implementations reported in the literature. Such performance allows the use of the FPGA model in real-time simulation applications or for the acceleration of offline EMT programs. A state-space approach is used to perform the time-domain simulation of the pole-residue form of the rational fitting of the characteristic admittance and propagation functions. The paper also discusses the appropriate scheduling of the ULM computations and proper management of the history terms that lead to an optimized hardware utilization, low latency response times, and higher computational performances using floating-point arithmetic.

1. Introduction

The Universal Line Model (ULM) [1] is a wide-band frequency-dependent model for lines and cables. The model includes full frequency dependency of the Transmission Line (TL) and cable parameters. The ULM with its various improvements [2–4] is currently recognized as being the most accurate wide-band model. The ability to run the ULM on FPGA is justified by various application requirements such as Hardware-in-the-Loop (HIL)-based testing of Travelling Wave Fault Locators (TWFLs). Modern TWFLs can locate a fault within a tower span ($\pm$ 300 m), but require 1 MHZ sampling rate to achieve such performance. Real-time simulators can indeed be used for providing a comprehensive test-bench for protection relays. Such a setup can be beneficial in the sense that it provides flexibility for selecting power system parameters as well as fault type and location while allowing the regeneration of real-time signals. Moreover, real-time simulators provide the possibility of in-situ testing of protection relays. However, for a HIL setup to function properly as a TWFL testbed, the simulated network necessitates a sub-microsecond time-step as well as accurate frequency-dependent line models [5].

CPU-based real-time simulation of the ULM have been shown to be rather computationally expensive for real-time simulation purposes because of the high fitting order of the lines [6], with simulation time-steps in the $>$ 10 μs range. On the other hand, FPGA-based real-time implementations of the ULM have been proposed in the literature, but fail to reduce the simulation time-step below 1 μs. Recently, an FPGA-based implementation of the ULM for HIL-based testing of TWFLs has been proposed in [7]. The line model is based on the second-order realization of TL's state-space equations[8] and achieves a time-step of 1.42 μs, while sustaining a clock frequency of 175 MHz.

To achieve a better performance, this work proceeds by increasing the clock frequency and deepening the pipeline to improving the computational performance. This however comes typically with a latency penalty. Hence, a thought-out rescheduling of the ULM equations is proposed to achieve more FLOPS at a lower latency. The real-time simulation of the ULM reported in this work can sustain a 250 MHz clock frequency and is shown to achieve a time-step as low as 200 ns, while offering proper accuracy for HIL-based TWFL testing.

The remainder of this paper is organized as follows. The theoretical background of the ULM is presented in Section 2. The proposed FPGA-based design methodology for an ULM simulator is discussed in Section 3. Section 4 elaborates on the test cases used to assess the performance of the FPGA model against EMTP [9] and discusses the footprint of the FPGA implementation as well as the computational accuracy resulting from the use of a non-standard floating-point (FP) format.

* Corresponding author.
E-mail addresses: tarek.ould-bachir@polymtl.ca (T. Ould-Bachir), hossein.chalangar@polymtl.ca (H. Chalangar), keyhan.sheshyekani@polymtl.ca (K. Sheshyekani), jean.mahseredjian@polymtl.ca (J. Mahseredjian).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.epsr.2020.106716
Received 4 October 2019; Received in revised form 19 April 2020; Accepted 2 August 2020
Available online 08 August 2020

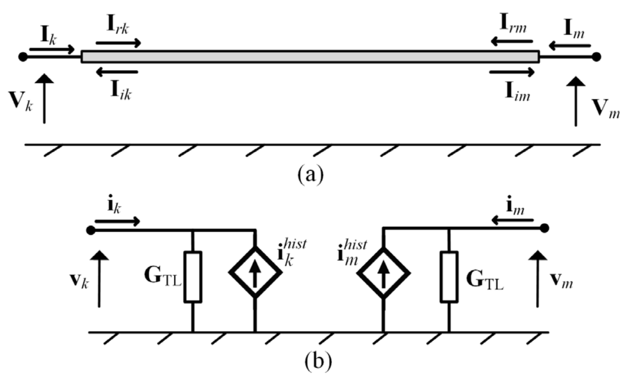

Fig. 1. (a) General view of a TL showing the reflected and incident currents in its terminals k and m in frequency domain; (b) The Norton equivalents of the TL in the time domain.

2. Universal line model

2.1. ULM modelling approach

The aim of this section is to briefly recall the theoretical basis of the ULM. Fig. 1 shows a typical multi-conductor line (or cable) of length l with sending end k, receiving end m, characteristic admittance $\mathbf{Y}_c$, and propagation function $\mathbf{H}$. The frequency domain equations of the line are solved at each end. The current $\mathbf{I}_k$ at end k is given by:

$$\mathbf{I}_k = \mathbf{I}_{shk} - 2\mathbf{I}_{ik} \tag{1}$$

$$\mathbf{I}_{shk} = \mathbf{Y}_c\mathbf{V}_k \tag{2}$$

$$\mathbf{I}_{ik} = \mathbf{H}\mathbf{I}_{rm} \tag{3}$$

where $\mathbf{I}_k$ and $\mathbf{V}_k$ are the terminal current and voltage at end k, whereas $\mathbf{I}_{ik}$ is the incident current at node k, and $\mathbf{I}_{rm}$ is the reflected current from node m, as shown in Fig. 1a. Similar equations are obtained for the receiving end m by interchanging subscripts k and m.

The time domain simulation of the TL is carried out using the approach described in [6]. From the equations, the line can be modeled by making use of two Norton equivalents at both terminations, as shown in Fig. 1b. The Norton equivalent for terminal k yields:

$$\mathbf{i}_k(t) = \mathbf{G}_{\mathrm{TL}}\mathbf{v}_k(t) - \mathbf{i}_k^{hist}(t) \tag{4}$$

where $\mathbf{G}_{\mathrm{TL}}$ is the admittance matrix associated with the transmission line (see Appendix A.1). We also have:

$$\mathbf{i}_{rk}(t) = \mathbf{i}_k(t) - \mathbf{i}_{ik}(t) \tag{5}$$

The admittance $\mathbf{Y}_c$ and propagation function $\mathbf{H}$ are rationally fitted in the frequency domain, which results in the following pole-residue identification forms:

$$\mathbf{Y}_c \simeq \mathbf{G}_0 + \sum_{j=1}^{N_{\mathrm{Y}_c}} \frac{\mathbf{R}_j}{s - p_j} \tag{6}$$

$$\mathbf{H} \simeq \sum_{i=1}^{N_g}\left(\sum_{j=1}^{N_{\mathrm{H}}^i} \frac{\mathbf{R}_{i,j}}{s - p_{i,j}}\right)e^{-s\tau_i} \tag{7}$$

where N_{Y_c}, $\mathbf{R}_j$ and p_j are respectively the fitting order, residue matrices and the poles resulting from the fitting of $\mathbf{Y}_c$. Likewise, N_g, N_H^i, $\mathbf{R}_{i,j}$, $p_{i,j}$ denote respectively the number of fitting groups, the fitting order of each modal group i ($i \in \{1, 2, \ldots, N_g\}$), the residue matrices and the poles resulting from the fitting of the propagation function $\mathbf{H}$. τ_i is the minimum phase delay associated with each modal group.

These approximations are evaluated in the time-domain using the state-space approach discussed in [6]. State vectors $\mathbf{x}^{Y_c}$ and $\mathbf{x}^H$ are associated with $\mathbf{Y}_c$ and $\mathbf{H}$ respectively and updated at each simulation time-point using:

$$\mathbf{x}_j^{\mathrm{Y}_c}(t+\Delta t) = \alpha_j^{\mathrm{Y}_c}\mathbf{x}_j^{\mathrm{Y}_c}(t) + \boldsymbol{\beta}_j^{\mathrm{Y}_c}\mathbf{v}_k(t) \tag{8}$$

$$\begin{aligned}\mathbf{x}_{i,j}^{\mathbf{H}}(t+\Delta t) = {} & \alpha_{i,j}^{\mathbf{H}}\mathbf{x}_{i,j}^{\mathbf{H}}(t) + \boldsymbol{\beta}_{i,j}^{\mathbf{H}}\{\mathbf{i}_{rm}(t-\tau_i) + \\ & \mathbf{i}_{rm}(t-\tau_i+\Delta t)\}\end{aligned} \tag{9}$$

The time domain shunt and incident currents at the sending end k are then obtained for the next time-point as follows:

$$\mathbf{i}_{shk}^{hist}(t+\Delta t) = \sum_{j=1}^{N_{\mathbf{Y}}} \mathbf{x}_j^{\mathrm{Y}_c}(t+\Delta t) \tag{10}$$

$$\mathbf{i}_{ik}(t+\Delta t) = \sum_{i=1}^{N_g}\sum_{j=1}^{N_{\mathbf{H}}^i} \mathbf{x}_{i,j}^{\mathbf{H}}(t+\Delta t) \tag{11}$$

and contribute to the total history term as given below:

$$\mathbf{i}_k^{hist}(t+\Delta t) = 2\mathbf{i}_{ik}(t+\Delta t) - \mathbf{i}_{shk}^{hist}(t+\Delta t) \tag{12}$$

The closed form expressions for the line parameters $\mathbf{G}_{\mathrm{TL}}$, α_j, $\boldsymbol{\alpha}_{i,j}$, $\boldsymbol{\beta}_{i,j}$, and $\mathbf{C}_j$ are given in Appendix A.1.

Propagation time delays τ_i are not integer multiples of the simulation time-step. Hence, a circular buffer is employed to store past values of the reflected current $\mathbf{i}_{rm}$ and a linear interpolation is used to obtain $\mathbf{i}_{rm}(t-\tau_j)$ and $\mathbf{i}_{rm}(t-\tau_j+\Delta t)$.

2.2. ULM time domain simulation

The following 6-step algorithm is used to simulate the ULM in time-domain with a fixed time-step Δt.

1. Solve network equations to obtain the terminal voltages and the reflected currents at each terminal of the line for the present simulation time-point, i.e. $\mathbf{v}_k(t)$, $\mathbf{v}_m(t)$, $\mathbf{i}_{rk}(t)$, and $\mathbf{i}_{rm}(t)$;
2. Update the buffers storing reflected currents using the results from Step 1);
3. For each terminal of the line, update the state vectors $\mathbf{x}_j^{Y_c}(t+\Delta t)$ and $\mathbf{x}_{i,j}^{H}(t+\Delta t)$ using Eqs. (8) and (9) respectively;
4. Compute the shunt history vector $\mathbf{i}_{sh}^{hist}(t+\Delta t)$ and the incident current vector $\mathbf{i}_i(t+\Delta t)$ at each terminal using Eqs. (10) and (11) respectively;
5. Compute the total history vectors $\mathbf{i}_k^{hist}(t+\Delta t)$ and $\mathbf{i}_m^{hist}(t+\Delta t)$ for the next time-point using Eq. (12);
6. Set $t = t + \Delta t$ and go to Step 1).

This algorithm is used as a foundation for the development of the proposed FPGA-based ULM simulator shown in Fig. 2.

3. ULM hardware architecture

3.1. FPGA-based ULM solver

The FPGA-based simulation of the ULM is presented in Fig. 2a. It is comprised of two main blocks, a Nodal Solver and a ULM Solver made of multiple instances of the ULM Computing Engine. The nodal solver composes network equations using the modified-augmented nodal analysis (MANA) [9]. To achieve real-time performance, the inverse of the MANA matrix is precomputed for all possible switch combinations [10]. The two solvers proceed serially, which yields the following formula for the minimum simulation time-step Δt_{min}:

$$\Delta t_{\mathrm{min}} = \frac{1}{f_{\mathrm{max}}}\{\ell_{\mathrm{NS}} + \ell_{\mathrm{ULM}}\} \tag{13}$$

where f_{max} is the maximum frequency that the FPGA model can handle, whereas ℓ_{NS}, and ℓ_{ULM} are the latencies of respectively the nodal and the ULM solvers.

A closer view of the ULM Computing Engine is given in Fig. 2b. As one can see, it is comprised of two convolutional kernels, one for

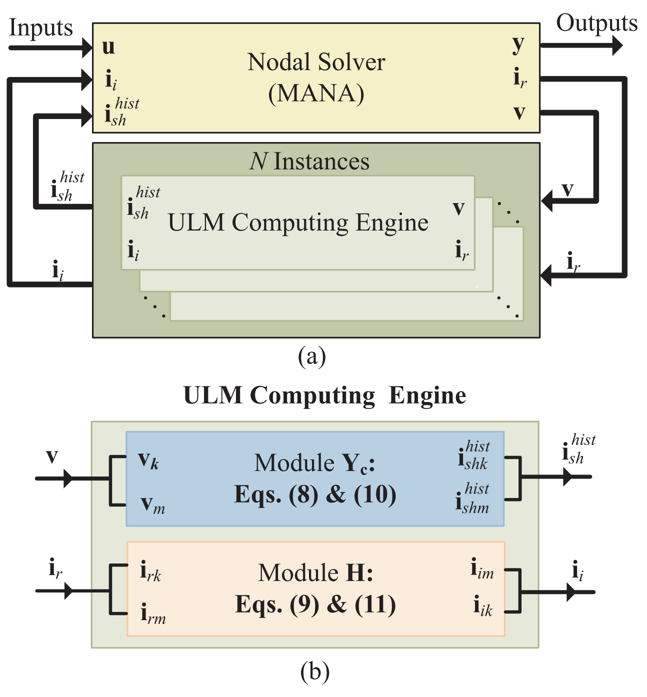

Fig. 2. (a) Nodal Solver connected to N ULM Computing Engines; (b) High-level inside view of an individual ULM Computing Engine.

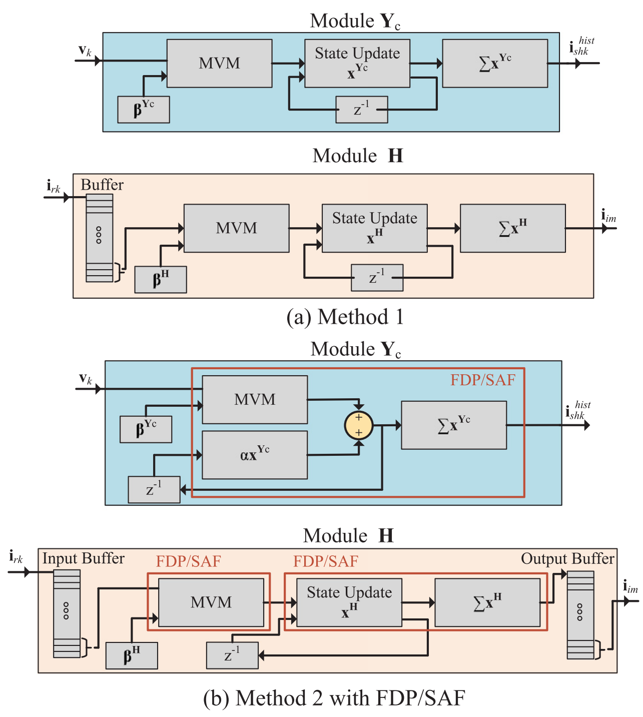

Fig. 3. Detailed view of the convolutional kernels for terminal k: (a) Method 1: Direct hardware implementation of Eqs. (8)–(11); (b) Method 2: Rescheduling of Eqs. (8)–(11) to reduce latency.

conducting the convolutions involving $\mathbf{Y}_c$ (Eqs. (8) and (10)), the other for conducting the convolutions involving $\mathbf{H}$ (Eqs. (9) and (11)). The two modules being executed in parallel, the latency of the ULM Solver ℓ_{ULM} is given by:

$$\ell_{\mathrm{ULM}} = \max_{i=1}^{N}(\ell^{i}_{\mathbf{Y}_c}, \ell^{i}_{\mathbf{H}}) \tag{14}$$

where N is the number of ULM instances, $\ell^{i}_{\mathbf{Y}_c}$, and $\ell^{i}_{\mathbf{H}}$ are the latencies of respectively Module $\mathbf{Y}_c$ and Module $\mathbf{H}$ of the ULM Computing Engine instance i. These latencies are implementation-specific.

3.2. Low latency ULM computing engine

Fig. 3 presents two alternative methods for implementing the convolutional kernels. Method 1 is shown in Fig. 3a. It is a direct hardware implementation of Eqs. (8)–(11). The Matrix-Vector Multiplication (MVM) and State Update blocks are used to implement Eqs. (8) and (9) in respectively the $\mathbf{Y}_c$ and $\mathbf{H}$ modules of Fig. 3a. When fully pipelined, Method 1 yields:

$$\ell_{\mathbf{Y}_c} = \ell^{dp}_{\mathbf{Y}_c} + N_{\mathbf{Y}} \tag{15}$$

$$\ell_{\mathbf{H}} = \ell^{dp}_{\mathbf{H}} + N_g \max_{i=1}^{N_g} N^{i}_{\mathbf{H}} \tag{16}$$

where $\ell^{dp}_{\mathbf{Y}_c}$, and $\ell^{dp}_{\mathbf{H}}$ are the datapath latencies of respectively the $\mathbf{Y}_c$ and $\mathbf{H}$ modules.

Our aim is to reduce these latencies in order to minimize the simulation time-step. This is achieved in Method 2 by means of two complementary optimizations:

- First, it is noted that the MVM precedes the State Update block in Fig. 3a. However, the MVM can be executed in parallel to an alternative State Update block in Module $\mathbf{Y}_c$ if both part are properly synchronized, which is done in Fig. 3b, resulting in a lower latency $\ell_{\mathbf{Y}_c}$op. Such an optimization is harder to obtain for Module $\mathbf{H}$ because an interpolation of past terms of i_{rk} needs to be performed, making the rescheduling difficult. Hence, the MVM and the State Update blocks are kept separate in Module $\mathbf{H}$ of Method 2.
- The second optimization results from the observation that Eqs. (9) and (11) involve past terms only. Hence, the datapath latency of Module $\mathbf{H}$ $\ell_{\mathbf{H}}$op could be ignored if the history buffer was moved from input to output. An input buffer would still be required nevertheless because Eq. (9) combines terms involving different modal delays τ_i. However, the input buffer would be less deep than the one needed in Method 1. This optimization is shown in Module $\mathbf{H}$ of Fig. 3b.

Hence, when fully pipelined, Method 2 yields:

$$\ell_{\mathbf{Y}_c} = \ell^{dp}_{\mathbf{Y}_c} + N_{\mathbf{Y}} \tag{17}$$

$$\ell_{\mathbf{H}} = \ell^{ob}_{\mathbf{H}} \tag{18}$$

where $\ell^{ob}_{\mathbf{H}}$ is the latency of the output buffer, which is typically a single clock cycle.

3.3. Low-latency custom-made floating-point operators

All the reported implementations of the FPGA-based ULM Solver of this paper use Method 2. To further decrease the simulation time-step, the ULM Solver is built using low-latency custom-made FP operators: An FPGA can handle real arithmetic using either fixed-point or FP format. Fixed-point format uses less hardware and yields a datapath with lower latency, but the number format suffers from a restricted dynamic range. On the other hand, the FP number format allows for a larger dynamic range, but its hardware arithmetic operators are costly in terms of FPGA resource consumption, and require deeper pipelines than their fixed-point counterparts. This is even more true when double precision is considered. For this reason, most FPGA-based EMT simulations reported in the literature make use of single precision FP[5,11] to save hardware and reduce latency. The approach adopted in this paper is different [12,13].

- A non-standard FP format with intermediate precision (between single and double) is used. More specifically, we used a 34-bit mantissa, with 8-bit exponent. This choice is a compromise between

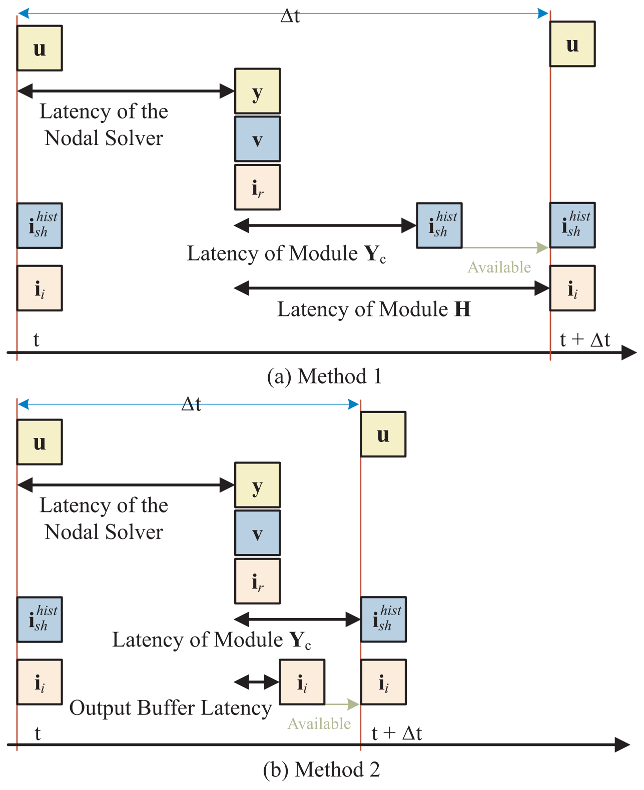

Fig. 4. Timing diagrams for: (a) Method 1; (b) Method 2.

hardware consumption (only 3 DSP blocks per multiplication are be used vs. 2 for single precision and 12 for double precision) and computational accuracy.

- The arithmetic operators were built using a so-called fused-datapath (FDP) approach [14]. FDP is an approach to the implementation of complex floating-point datapaths which consists in removing all intermediate packing and unpacking stages and performing all the computations jointly within a fused-datapath. The approach does not adhere to IEEE-754 Standard but provides better resource consumption.
- All internal additions (including accumulation) are carried out using an internal FP format called the Self-Alignment Format (SAF) [15]. SAF provides higher numerical accuracy while allowing the implementation of complex fused-path operators with very low latencies. SAF has also the ability to provide single-cycle FP accumulation, which is needed for the summation unit of our convolutional kernels. FDP/SAF operators are clearly identified in Fig. 3b.

The impact of the proposed improvements on the simulation time-step are illustrated in Fig. 4. Fig. 4a shows the timing diagram for Method 1. For each time-point, the Nodal Solver starts by reading **u**, $\mathbf{i}_{sh}^{hist}$, and $\mathbf{i}_i$, then produces **y**, **v**, and $\mathbf{i}_r$ after a latency of ℓ_{NS}. At that moment, the Y_c Module and H Module start their processing and produce $\mathbf{i}_{sh}^{hist}$, and $\mathbf{i}_i$ for the next time-point of the simulation after a latency of ℓ_{Y_c} and ℓ_H respectively. The computations for the following time-point can only start when the results of the convolutional kernels are both available, hence Eq. (14). Fig. 4b. illustrates how Method 2 manages to reduce the simulation time-step Δ*t*. On the first hand, the FDP/SAF approach reduces the latency ℓ_{Y_c} needed to produce $\mathbf{i}_{sh}^{hist}$. On the second hand, the required latency to obtain $\mathbf{i}_i$ (i.e. ℓ_H) is a single clock cycle. The total latency of the ULM Solver becomes $\ell_{ULM} = \ell_{Y_c}$.

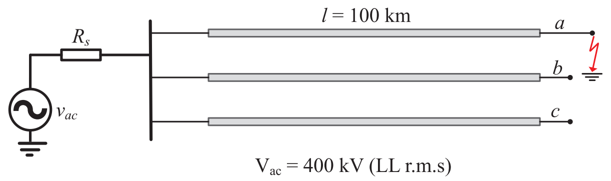

Fig. 5. 100 km aerial line test case.

4. Results

4.1. Test case 1: aerial transmission line

The first test case considered here consists of an aerial transmission line of length 100 km, as shown in Fig. 5. The geometry and data of the line are given in Appendix A.2, see Fig. 10 and Table 2. The fitting of the line parameters was performed using EMTP for 8 decades starting at $f_{\min} = 0.1$ Hz, and resulted in $N_{Y_c} = 7$, $N_g = 2$, and $\max N_{\mathbf{H}}^{i} = 4$. The line is energized by a three-phase sinusoidal source while the receiving end is left open. After reaching steady-state, a single-phase to earth fault is applied on phase-a (AG fault) at the receiving end of the line at $t = 0.1$ s of the simulation time. The fault is cleared after 0.1 s. For this test, the simulation time-step was set to 1 μs. However, the FPGA design can achieve a time-step as small as 200 ns for this case, as discussed in Section 4.3.

Fig. 6 a superimposes the voltages at the receiving end from the FPGA and EMTP for the first 0.25s of the simulation time. Close-up views of the fault initiation and clearing instants are given in Figs. 6b and c respectively. As one can see, the hardware implementation matches perfectly the EMTP reference during steady state as well as during transients.

The computational accuracy of the ULM Solver is assessed in Fig. 7 by drawing the incident current computed by Module **H** for the receiving end of the line and comparing to EMTP. Fig. 7a shows the incident current at the receiving end while Fig. 7b gives the computed relative errors of the FPGA-based ULM using standard single and double precision arithmetics as well as the proposed FDP/SAF. It is obvious from Fig. 7b that FDP/SAF brings better accuracy than strict IEEE-754 compliant single precision, while offering a lower datapath latency, a higher clock frequency at reasonable hardware cost, as further demonstrated in Section 4.3.

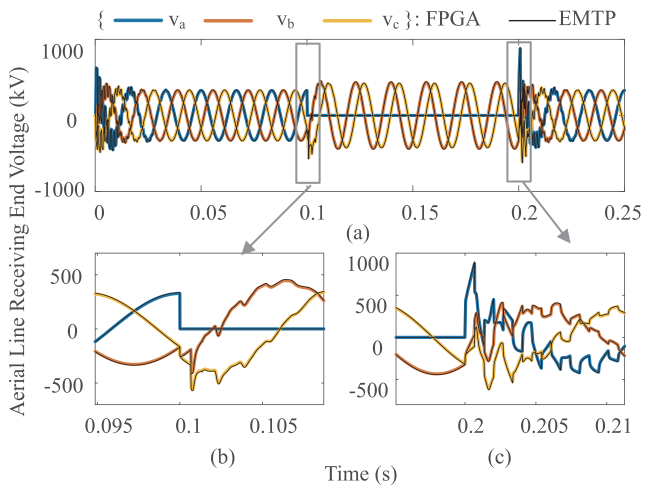

Fig. 6. Test Case 1: an AG fault is applied at receiving end at t = 100 ms and is cleared at t = 200 ms. (a) Receiving end phase voltages; (b) Close-up view of phase-voltages during fault initiation; and (c) Close-up view of phase-voltages during fault clearing.

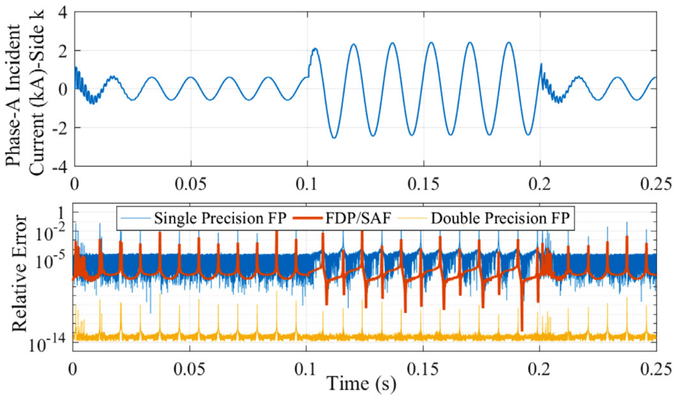

Fig. 7. Assessment of the accuracy of the FPGA-based simulation. (a) Incident current at the receiving end from Test Case 1, phase-a; (b) Relative errors compared to EMTP.

4.2. Test case 2: TWFL test network

Test Case 2 deals with a HIL TWFL testing. The principles of a TWFL read as follows: A fault on a protected line generates Travelling Waves (TWs) that propagate towards both ends of the line where they are captured by TWFLs for processing. By comparing the arrival times of the wave fronts at both ends, it is possible to locate the fault in a so-called double-ended approach [16]. In such a configuration, the TWFLs exchange data through fiber optic links. Single-ended fault location is realized by comparing the arrival time of two consecutive wave fronts seen from a TWFL device [17].

An FPGA-based ULM simulator with sub-microsecond time-step capability has been shown to be adequate for the HIL testing of TWFLs in both single- and double-ended configurations [5]. This test case assesses the performance of the proposed ULM Solver for such an application. The test considers the power system of Fig. 8 which consists of a 100 km protected line surrounded by two TLs of 50 km connected to lumped equivalent networks at each end. Each equivalent network consists of 3-phase resistance and inductance of $R = 0.08929\ \Omega$ and $L = 1.658$ mH, and a 500 kV RMS, 60 Hz 3-phase voltage source. An AG fault is applied at the distance DF= 20 km from the sending end of the protected TL at $t = 0.191$ s of the simulation time.

Fig. 9 a and b show the terminal currents at both ends of the protected line, as executed by the FPGA model. For this test, the simulation time-step was set to 500 ns, yet the FPGA design can achieve a time-step as small as 200 ns, as discussed in Section 4.3. The TWs captured at both ends are shown in Fig. 9c. As one can see, the relative arrival time of the first two peaks at two terminals can be used to accurately locate the fault in a double-ended configuration: 202 µs ≡ DF = 20.003 km. Alternatively, the relative arrival time of the first two peaks of the TW seen at the sending end (k) can be used to locate the fault in a single-ended configuration: 135 µs ≡ DF = 20.047 km. Hence, the FPGA model produced accurate transients that allowed testing single- and double-ended TWFL configurations.

4.3. Area occupation and speed performance

Table 1 gives the FPGA resource consumption for the two test cases

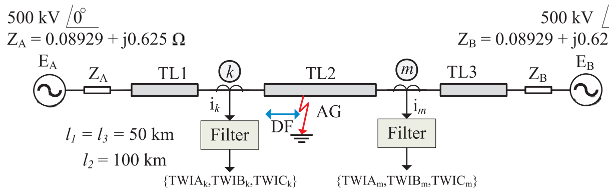

Fig. 8. Network for the TWFL test case.

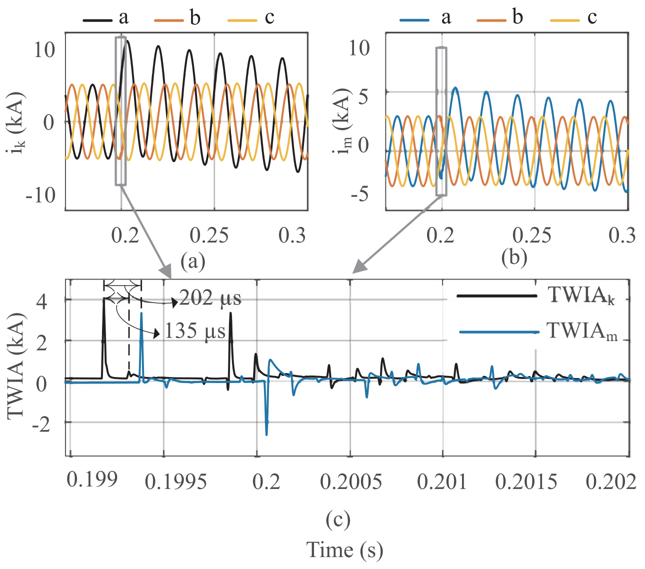

Fig. 9. Test Case 2: an AG fault is applied at DF = 20 km (see Fig. 8). (a) Currents from protected line at sending end; (b) Currents from protected line at receiving end; (c) Phase-a Travelling-Wave currents from sending and receiving ends of the protected line.

Table 1
FPGA Footprint.

Test Case 1				
Item	LUTs	Registers	BRAM	DSP
Y_c	5,962 (2.9%)	6070 (1.49%)	0 (0.0%)	36 (4.3%)
H	9559 (4.7%)	11,004 (2.7%)	9 (2.0%)	63 (7.5%)
ULM	31,042 (15.2%)	34,148 (8.4%)	18 (4.0%)	198 (23.6%)
Test Case 2				
Item	LUTs	Registers	BRAM	DSP
Y_c	23,848 (11.7%)	24,280 (6.0%)	0 (0.0%)	144 (17.1%)
H	38,236 (18.8%)	44,016 (10.8%)	36 (8.1 %)	252 (30.0%)
ULM	124,168 (60.9%)	136,592 (33.51 %)	72 (16.2 %)	792 (94.3%)

considered in this paper and for each Module $\mathbf{Y}_c$ and Module **H** as well as the ULM Solver. The target FPGA is the Kintex 7 325T, a mid-range FPGA from Xilinx introduced in 2010. The designs are fully pipelined and can sustain a clock frequency of up to 250 MHz. The Table 1 also reports the latency associated with the datapath of each module. It is worth mentioning that the ULM Solver consists of two $\mathbf{Y}_c$ and two **H** modules, which explains the results of the table (for instance, the ULM

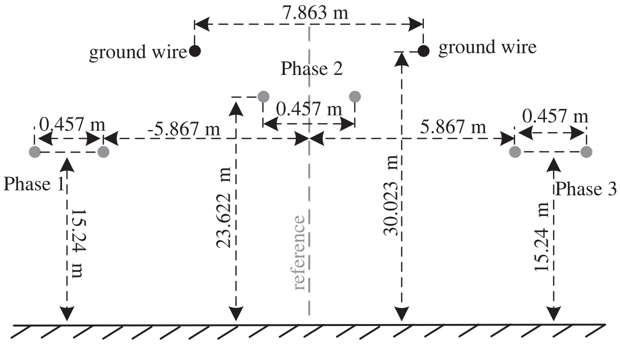

Fig. 10. Geometry of the aerial TL used in this paper.

Table 2
Data of the used aerial TLs.

Parameter	Value	
Outside diameter of phase wire	4.06908	cm
Outside diameter of ground wire	0.98044	cm
DC resistance of phase wire	0.0324	Ω/km
DC resistance of ground wire	1.6216	Ω/km
Ground resistivity	100	$\Omega \cdot$ m

of Test Case 1 requires 31,042 LUTs, which is twice as much as $5,962 + 9,559$). Similarly, Test Case 2 consists of four (4) line segments, hence the numbers of Table 1 ($124,168 = 4 \times 31,042$). It is also worth mentioning that $\ell^{dp}_{\mathbf{Y}_c} = 16$, $\ell^{dp}_{\mathrm{NS}} = 25$ for both test cases. With $N_{\mathbf{Y}} = 7$, we get $\Delta t_{\min} = 4$ ns$\times$ $((16 + 7) + 25) = 192$ ns.

As one can see from Table 1, a ULM Solver made of a single Computing Engines (Test Case 1) occupies a little more than 20%, a number mainly dominated par the consumption of DSP blocks. It is worth mentioning that the FPGA considered in this paper offers 840 DSP blocks, whereas more recent devices can provide thousands DSP blocks.

5. Conclusion

This paper presented a methodology for the design of high-performance, low-latency FPGA-based solvers for the simulation of the ULM. The solver is comprised of two convolutional kernels for which appropriate scheduling guidelines were proposed. Custom-made FP operators were used to reduce the FPGA footprint while allowing the solver to sustain higher clock frequencies. The proposed design was tested against two test cases: an aerial TL as well as for the HIL testing of TWFLs, demonstrating good accuracy and performance.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

A1. ULM Line model coeffcients

The coefficients in Eqs. (4) and (8)–(11) read as follows:

$$\mathbf{G}_{\mathrm{TL}} = \mathbf{G}_0 + \sum_{j=1}^{N_{\mathbf{Y}}} \boldsymbol{\lambda}_j \tag{19}$$

$$\boldsymbol{\lambda}_j = \frac{\Delta t \mathbf{R}_j}{2 - \Delta t p_j} \tag{20}$$

$$\alpha_j^{\mathbf{Y}_c} = \frac{2 + \Delta t p_j}{2 - \Delta t p_j} \tag{21}$$

$$\boldsymbol{\beta}_j^{\mathbf{Y}_c} = (\alpha_j + 1)\boldsymbol{\lambda}_j \tag{22}$$

$$\boldsymbol{\alpha}_{i,j}^{\mathbf{H}} = \frac{2 + \Delta t p_{i,j}}{2 - \Delta t p_{i,j}} \tag{23}$$

$$\boldsymbol{\beta}_{i,j}^{\mathbf{H}} = \frac{\Delta t \mathbf{R}_{i,j}}{2 - \Delta t p_{i,j}} \tag{24}$$

A2. Aerial TL line parameters and geometry

The geometry and parameters of the aerial TL used for Test Case 1 and 2 are given in Fig. 10 and Table 2 and respectively.

References

[1] A. Morched, B. Gustavsen, M. Tartibi, A universal model for accurate calculation of electromagnetic transients on overhead lines and underground cables, IEEE Trans. Power Deliv. 14 (3) (1999) 1032–1038.

[2] I. Kocar, J. Mahseredjian, Accurate frequency dependent cable model for electromagnetic transients, IEEE Trans. Power Deliv. 31 (3) (2016) 1281–1288.

[3] I. Kocar, J. Mahseredjian, G. Olivier, Improvement of numerical stability for the computation of transients in lines and cables, IEEE Trans. Power Deliv. 25 (2) (2010) 1104–1111.

[4] B. Gustavsen, Passivity enforcement for transmission line models based on the method of characteristics, IEEE Trans. Power Deliv. 23 (4) (2008) 2286–2293.

[5] H. Chalangar, T. Ould-Bachir, K. Sheshyekani, S. Li, J. Mahseredjian, Evaluation of a constant parameter line-based TWFL real-time testbed, IEEE Trans. Power Deliv. 35 (2) (2020) 1010–1019.

[6] O. Ramos-Leanos, J.L. Naredo, J. Mahseredjian, C. Dufour, J.A. Gutierrez-Robles, I. Kocar, A wideband line/cable model for real-time simulations of power system transients, IEEE Trans. Power Deliv. 27 (4) (2012) 2211–2218.

[7] R. Mirzahosseini, R. Iravani, Y. Zhang, An FPGA-based digital real-time simulator for hardware-in-the-loop testing of traveling-wave relays, IEEE Trans. Power Deliv. (2020). 1–1.

[8] O. Ramos-Leaos, J. Mahseredjian, J.L. Naredo, I. Kocar, J.A. Gutierrez-Robles, J.A. Martinez, Phase-domain line/cable model through second-order blocks, IEEE Trans. Power Deliv. 30 (6) (2015) 2460–2467.

[9] J. Mahseredjian, S. Dennetire, L. Dub, B. Khodabakhchian, L. Grin-Lajoie, On a new approach for the simulation of transients in power systems, Electric Power Systems Research 77 (11) (2007) 1514–1520. Selected Topics in Power System Transients - Part II.

[10] H.F. Blanchette, T. Ould-Bachir, J.P. David, A state-space modeling approach for the FPGA-based real-time simulation of high switching frequency power converters, IEEE Trans. Ind. Electron. 59 (12) (2012) 4555–4567.

[11] Y. Chen, V. Dinavahi, FPGA-based real-time EMTP, IEEE Trans. Power Deliv. 24 (2) (2009) 892–902.

[12] T. Ould-Bachir, H.F. Blanchette, K. Al-Haddad, A network tearing technique for FPGA-based real-time simulation of power converters, IEEE Trans. Ind. Electron. 62 (6) (2015) 3409–3418.

[13] T. Ould-Bachir, H. Saad, S. Dennetire, J. Mahseredjian, CPU/FPGA-based real-time simulation of a two-terminal MMC-HVDC system, IEEE Trans. Power Deliv. 32 (2) (2017) 647–655, https://doi.org/10.1109/TPWRD.2015.2508381.

[14] M. Langhammer, High performance matrix multiply using fused datapath operators, Asilomar Conference on Signals, Systems and Computers, (2008), pp. 153–159, https://doi.org/10.1109/ACSSC.2008.5074382.

[15] T. Ould-Bachir, J.P. David, Self-alignment schemes for the implementation of addition-related floating-point operators, ACM Trans. Reconfigurable Technol. Syst. 6 (1) (2013) 1:1–1:21.

[16] S. Marx, B. Johnson, A. Guzmàn, V. Skendzic, M. Mynam, Traveling wave fault location in protective relays: design, testing, and results, Annual Georgia Tech Fault and Disturbances Analysis Conference, (2013), pp. 1–14.

[17] E. Schweitzer, A. Guzmàn, M. Mynam, V. Skendzic, B. Kasztenny, C. Gallacher, S. Marx, Accurate single-end fault location and line-length estimation using traveling waves, International Conference on Developments in Power System Protection, (2016), pp. 1–6.