A State-Space Modeling Approach for the FPGA-based Real-Time Simulation of High Switching Frequency Power Converters

Handy Fortin Blanchette, Member, IEEE, Tarek Ould-Bachir, Member, IEEE, and Jean-Pierre David, Member, IEEE

Abstract—A comprehensive approach to the real-time simulation of power converters using a state-space representation is covered in this work. Systematic formulations of state-space equations as well as a new switch model are presented. The proposed switch model exhibits a natural switching behaviour, which is a valuable characteristic for the real-time simulation of power converters, thereby allowing an individual treatment of the switching devices irrespective of the converter topology. Successful implementations of the proposed switch model on a Field Programmable Gate Array (FPGA) device are reported, with two alternative approaches: 1) pre-computing network equations for all switch states combinations; 2) solving network equations on-chip using the Gauss-Seidel iterative method. A two-level three-phase voltage source converter is implemented on FPGA using the second approach, with a time-step of 75 ns and a switching frequency of 20 kHz. Comparison with SPICE models show that the proposed switch model offers very satisfactory accuracy and precision.

Index Terms—switch model, power converter, state-space, real-time simulation, iterative methods, Gauss-Seidel, FPGA

I. INTRODUCTION

RECENT advances in semiconductor technology tend to raise the operational switching frequency of modern power converters, thereby reducing their size and weight. High-power density converters are crucial in such application fields as the automotive and aerospace industries where typical switching frequencies are in the 10–200 kHz range [1]–[4]. Another motivation for high-frequency switching converters is the reduction of the resulting total harmonic distortion (THD) in order to alleviate the usage of filtering components in the converter [5]–[7]. These breakthroughs in the realm of power electronics are however very challenging for the conduction of real-time simulations. The real-time simulation of power converters is well recognized by industrial practitioners as a valuable tool for the development of controllers, allowing time saving and comprehensive test coverage for faults scenarios and near limits operational behaviour. Real-time simulators are made of very capable computers running real-time operating systems and having recourse to dedicated hardware for I/O purposes [8]. Typical time-steps in these simulators are in the 5–10 µs range. By taking into account the guidelines for accurate modeling of power electronics [9], simulation time-steps should by at least 20 times less than the switching frequency. That limits the switching frequency to the 5–10 kHz range at best.

The high parallelism offered by FPGAs and their potential to conduct real-time simulation in the nanosecond range, as demonstrated by the promising results reported in the literature [10]–[16], make these devices an emerging alternative platform for the real-time simulation of power converters. However, many challenges remain for the broad adoption of FPGAs in the real-time simulation of power converters in the industrial context. One of the main challenges resides in finding an appropriate model for the switches. The simulation of switching networks traditionally falls into two modes [17]: the detailed-mode and the behavioural-mode. Detailed-mode simulation can be conducted using commercial software applications such as SABER and SPICE, where the switch model is formulated in terms of non-linear functions that are hardly usable in real-time simulation. Behaviour-mode simulation is hence consubstantial to real-time simulation. Three types of switch models are acknowledged as belonging to the category of behavioural-mode simulation [17]: 1) the ideal model; 2) the switching function model; and 3) the average model. The ideal switch model and the switching function model are mostly used in FPGA-based simulation.

Switching function models are very powerful and can be used when conducting simulation where faults are inserted [12]. Moreover, it has been shown that precise raise-time and fall-time of switches can be emulated on FPGA using definite \( v-i \) switching characteristics, either with piecewise linear slopes [13] or real device switching \( v-i \) characteristics [14]. These implementations report time-steps of 12.5 ns. Although this approach provides precise switching information, the state machine used to control the simulation flow is heavily dependent on the converter topology; a drawback inherent to the switching function model. This necessitates elaborate identification of all possible modes of the converter, a difficult task when complex converter topologies are considered.

In the case of ideal models, the switch is either an ideal
switch or a binary resistance ($R_{on}/R_{off}$). It has been shown that this model can be used when the switching waveforms do not have to be exact, and only the transition between two circuit states matters. Instantaneous transitions are thus possible at the expense of complex mathematical manipulations [18], [19]. These are not appropriate for an FPGA implementation when seeking reasonably low time-steps and/or moderate on-chip memory requirements. In the early 1990s, an effort was made to develop a switch model that handled each switch individually while keeping the network equations fixed, regardless of the switch status [20]–[22]. This method is a convenient way to simulate power electronic converters on FPGA, as demonstrated in the recently reported FPGA implementation of a three-level converter simulator [15], [23], with a switching frequency of 2 kHz and a time-step of 500 ns. However, this method suffers from its tendency to introduce artificial oscillations [15], [24], which may impose some restrictions on the achievable maximal switching frequency.

The main objective of this work is to push the switching frequency into the 10–200 kHz range, using a state-space modeling approach. The state-space representation offers many advantages, as allowing the study of the system dynamics by computing the eigenvalues of the state matrix [25]. The eigenvalues of a system provide useful information for selecting an appropriate numerical method for solving network equations in the time-domain, and for choosing a computation time-step. However, the use of the state-space approach for modeling switching networks is known to be associated with the following drawbacks [22]: i) possible state-variable discontinuities; ii) potential variability of the state number according to switch combinations; iii) a priori knowledge of the circuit operation is required. This paper proposes a switch model that can be used with a state-space modeling approach, thereby overcoming these drawbacks. Hence: i) it does not imply any state discontinuity during switching transients; ii) the number of states is constant with switch states combinations; iii) it handles each switch individually, and it provides the switch with a natural behaviour that does not require a priori knowledge of the circuit topology nor operation.

The remainder of this paper is structured as follows. The proposed switch model and operation principles are introduced in Section II. Power converter modeling using the state-space formulation and the proposed switch model is presented in Section III. The FPGA implementation of a two-level three-phase bridge converter using a pre-computed matrix approach, and the FPGA implementation of ideal and non-ideal boost converters using the Gauss-Seidel method are given in Section IV. The FPGA implementation details are presented in Section V. Implementation results are discussed in Section VI. Section VII concludes this work.

II. SWITCH MODEL

The switch model proposed in this paper is shown in Fig. 1.a. The model consists of a binary resistance ($R_{on}/R_{off}$) connected with a parallel parasitic capacitance $C_{sw}$. This capacitance is comparable to some extent to the drain to source capacitance $C_{ds}$ found in a typical SPICE MOS model, such as the one depicted in Fig. 1.b. The choice for appropriate values for $C_{sw}$ is discussed in Section IV-B.

The proposed model can represent any type of switch. Fig. 2 outlines the status updating rule for a selection of switches, namely: a generic diode, a transistor/diode pair and a transistor bidirectional switch. The natural switching behaviour of this model lends itself well to power converter simulation, as demonstrates the following.

A. Boost case study

The following aims to explain how turn-on and turn-off processes are covered by the natural switching characteristics of the proposed model. The circuit used to conduct this analysis is shown in Fig. 3. This is a typical boost converter with a voltage source ($V_{cc}$), a boost inductor, a diode-transistor pair, a load capacitor ($C_{load}$), and a resistive load ($R_{load}$).
Fig. 4. (a) Current distribution before transistor turn-off. (b) The current $I_s$ is diverted by parasitic capacitors. (c) Current distribution after the diode turn-on during the turn-off process. (d) Time-domain waveforms showing turn-off switching intervals. (e) Current distribution at the end of the transistor turn-off process and before the transistor turn-on. (f) Current distribution when both the transistor and the diode are on simultaneously. (g) Current distribution when both the diode and the transistor are in high impedance. The current is diverted in the diode parasitic capacitance. (h) Time-domain waveforms showing turn-on switching intervals.
The proposed study covers the hard switching case, i.e. when the circuit inductance connected to the diode-transistor pair is considered as a perfect current source during switching events.

Switching transients being very fast, the current in the boost inductor is considered to be constant. Therefore, the inductance is replaced by a constant current source \((I_s)\) as shown in Fig. 3. The same reasoning applies to the load capacitor voltage. During switching transients, the voltage across \(C_{\text{load}}\) is considered constant, thus \(C_{\text{load}}\) and \(R_{\text{load}}\) are replaced by a constant voltage source \((V_{\text{load}})\) as shown in Fig. 3. The transistor and the diode are substituted by the proposed switch model. Finally, a leakage inductance \((L_{\text{leak}})\) is connected in series with \(V_{\text{load}}\) as in a real physical setup [26].

1) **Transistor turn-off process**: At the beginning of the turn-off process, the transistor is in conduction mode and the diode is blocked. The current \(I_s\) circulates in the resistive part of the transistor model \((r_T)\) as shown in Fig. 4(a).

The turn-off process starts when the transistor gate signal is set to 0. The resistive part of the transistor model jumps instantaneously to high impedance. The current \(I_s\), which cannot be stopped, is diverted through \(C_D\) and \(C_T\) as shown in Fig. 4(b). Since the transistor was in conduction mode, the voltage across the diode parasitic capacitance is initially \(V_{\text{load}}\) with the polarities indicated on Fig. 4(b). The current source discharges \(C_D\) and charges \(C_T\). The voltage slope at \(V_A\) is given by:

\[
\frac{\partial V_A}{\partial t} = \frac{I_s}{C_D + C_T} \text{ (V/s)} \quad (1)
\]

When the voltage across the diode is greater or equal to zero, the diode resistance settles to low impedance. The current \(I_s\) is gradually transferred from \(C_T\) toward \(V_{\text{load}}\) through the resistive part of the diode model as shown in Fig. 4(c). This time-varying current in the leakage inductance creates a voltage spike across the transistor. When the voltage spike reaches its maximum, a resonance takes place in the circuit formed by \(C_T\), \(r_D\) and \(L_{\text{leak}}\), with a ringing frequency \((\omega_r)\) is given by:

\[
\omega_r = \frac{1}{\sqrt{L_{\text{leak}}C_{\text{diode}}}} \quad (4)
\]

The current circulating in the diode decreases at the same rate, transferring gradually the current source to the transistor. When \(I_{L_{\text{leak}}}\) reaches \(I_s\), the current in the diode passes by zero and progressively becomes negative. This negative current creates a negative voltage across the parasitic capacitor associated with the diode, and settles the diode in high impedance. At that time, the voltage across the diode is still low and the current in the leakage inductance continues to increase negatively to charge \(C_{\text{diode}}\) as shown in Fig. 4(g). Once the voltage across the diode reaches \(V_{\text{load}}\), a resonance with a frequency \((\omega_r)\) occurs between the diode parasitic capacitor and the leakage inductance with a frequency given by:

\[
\omega_r = \frac{1}{\sqrt{L_{\text{leak}}C_{\text{diode}}}} \quad (4)
\]

Similar to the turn-off process, this oscillation is damped by the circuit parasitic resistances. At the end of the damped oscillation, the turn-on process is completed, and the circuit returns to the configuration shown in Fig. 4(a). Figures 4(d) and 4(f) summarize the turn-off and the turn on process respectively. For both figures, circuit configurations are identified for the turn-off (a, b and c) and for the turn-on (e, f and g) on the time-domain waveforms. Diode and transistor logics are also included to clearly show the switching process.

### III. State-space circuit modeling

Literature provides various methods to write the state-space equations for general cases [27]. The method proposed in this section is a simplified approach especially suited for modeling power converters.

#### A. Formulation of the state equations

The state-space model of a linear system is given by:

\[
\dot{x} = Ax + Bu \quad (5)
\]

where \(A\) is the state matrix, \(x\) is the state vector, \(B\) is the input matrix and \(u\) the input vector. In a power converter, the states are given by current flowing in the inductances and the voltage across the capacitors. The state-space equations are written in two steps. In the first step, voltage summation around all the loops in the converter is considered to get
the equations for the derivative of the inductor currents. In the second step, the currents flowing in the capacitors of the converter are considered to obtain the derivatives of the voltage at capacitors terminals. The boost converter of Fig. 3 is used as an illustrative example. For that purpose, Fig. 5 presents an equivalent circuit, where the diode and the transistor were replaced by the proposed switch model.

1) Voltage loops: Voltage summation is performed by considering that each surface in the circuit stores the magnetic flux resulting in a leakage inductance. The leakage inductance is neglected in loops containing physical inductances because considering that each surface in the circuit stores the magnetic flux is managed by them. The voltage summation around a loop is given by:

$$\frac{\partial \varphi}{\partial t} = \sum V_{td} + \sum V_C + \sum R \cdot i_l$$

(6)

where $V_{td}$ are the independent voltage sources around the loop, $V_C$ are the voltage drops on the capacitors around the loop and $R \cdot i_l$ are the voltage drops due to the resistive elements in the loop. The derivative of the flux is then converted into current states using the chain-rule:

$$\frac{\partial \varphi}{\partial t} = \frac{\partial \varphi}{\partial i_l} \frac{\partial i_l}{\partial t} = L \frac{\partial i_l}{\partial t}$$

(7)

In the example of Fig. 5, the flux stored in the surface $A$, namely $\varphi_1$, is controlled by the physical boost inductance $L_1$. In the surface $B$, there is no physical inductance and the flux $\varphi_2$ is controlled by a leakage inductance $L_2$ (not explicitly shown in Fig. 5).

It is worth noting that in a SPICE simulation, the switching dynamics are influenced by the mutual inductions. These terms can be neglected in a system-level simulation, since the exact switching transients are not relevant.

2) Currents in capacitors: The second step deals with the currents flowing in the capacitors of the converter. The current for each capacitor is given by:

$$i_c = C \frac{\partial V_C}{\partial t} = \sum i_{cc} + \sum i_l + \sum G V_C$$

(8)

where $i_c$ is the current flowing in the capacitor, $i_{cc}$ are the independent current sources connected in parallel to the capacitors, $i_l$ are the loop currents flowing through the capacitors, $V_C$ is the voltage across the capacitors and $G$ relates to the conductances connected on capacitor terminals.

3) State-space final form: The matrices $A$ and $B$ are expressed by the mean of a reactive matrix $M_r$ as follows: $A = M_r A_r$ and $B = M_r B_r$. The matrix $M_r$ is given by the inverse of the matrix containing reactive components in the converter (i.e. the inductances and the capacitances). The state-space equations are then obtained:

$$\begin{bmatrix} \dot{v}_c \\ \dot{i}_c \\ \dot{v}_e \\ \dot{i}_e \end{bmatrix} = \begin{bmatrix} L & 0 & 0 & 0 \\ 0 & C & 0 & 0 \\ M_r & 0 & G & 0 \\ 0 & M_r & 0 & 0 \end{bmatrix} \begin{bmatrix} i_c \\ v_c \\ i_e \\ v_e \end{bmatrix} + \begin{bmatrix} M_r B_r u \end{bmatrix}$$

(9)

In equation (9), $r$ is a constant sub-matrix that contains all the resistors in the converter and $G$ is a diagonal sub-matrix that contains the conductances connected on capacitor terminals (as are the conductances associated with the switches, which are connected to parasitic capacitances). The sub-matrix $A_r(1, 2)$ in equation (9) represents the voltage drops on the capacitors. The sign convention used for these terms is shown in Fig. 6(a). The sub-matrix $A_r(2, 1)$ represents currents flowing in the capacitors. The sign convention for these terms is given in Fig. 6(b). When considering the boost converter of Fig. 5, the following terms are obtained:

$$i_l = i_1 i_2^T, \quad i_1 = [i_1 i_2]^T$$

$$v_e = [v_{c1} v_{c1} v_{c3}]^T, \quad v_e = [v_{c1} v_{c1} v_{c3}]^T$$

$$M_r = \begin{bmatrix} L_1^{-1} & 0 & 0 & 0 \\ 0 & L_2^{-1} & 0 & 0 \\ 0 & 0 & C_1^{-1} & 0 \\ 0 & 0 & 0 & C_3^{-1} \end{bmatrix}$$

$$A_r = \begin{bmatrix} i_1 & i_2 & v_{c1} & v_{c2} & v_{c3} \\ -r_1 & 0 & -1 & 0 & 0 \\ 0 & -r_2 & 1 & -1 & -1 \\ 1 & -1 & -G_1 & 0 & 0 \\ 0 & 1 & 0 & -G_2 & 0 \\ 0 & 1 & 0 & 0 & -1/R_L \end{bmatrix}$$

(10)

The boxed terms in equation (10) are switch conductances and are located on the state matrix diagonal. This is always the case with the proposed approach because the switches are represented by conductances and are always related to a parasitic capacitor voltage.

B. Selection of an appropriate solver

The state-space approach provides network equations in a continuous form that need to be discretized in order to conduct a time-domain simulation. The inclusion of the small parasitic capacitances and inductances (as discussed in Section III-A), tends to increase the stiffness of the state matrix $A$. Consequently, the fixed-step solver used for the discretization of the state-space equations must be able to handle stiffness, be precise, numerically stable, and capable of handling the frequent switching events of the power converter. The second order implicit backward Euler was found to fulfill the aforementioned requirements. Given an ordinary differential equation $\xi = f(\xi(t), \nu(t))$, the scheme of this solver is given by [28]:

$$\xi_{k+1} - \frac{4}{3} \xi_k + \frac{1}{3} \xi_{k-1} = \frac{2}{3} h f(\xi_{k+1}, \nu_{k+1})$$

(11)
where $h = \Delta t$ is the calculation time-step. Applying this scheme to equation (5) yields:

$$(I - \frac{2}{3}hA)x_{k+1} = \begin{pmatrix} \frac{4}{3} x_k - \frac{1}{3} x_{k-1} + \frac{2}{3} h b u_{k+1} \end{pmatrix}$$

This leads to the main challenge in implementing the proposed switch model on FPGA, i.e. solving the system of linear equations given by $M\mathbf{w} = \mathbf{b}$.

### IV. Solving the System of Equations

When an offline simulation is conducted on a CPU-based platform, direct methods like Gauss-Jordan can be used for solving $M\mathbf{w} = \mathbf{b}$. In an FPGA-based computing context, if $M$ is time-invariant, it is more likely to pre-compute and store the inverse of $M$ [29]. However, the state equations used in this paper are time-varying because of the binary conductances associated with the switches. The first approach considered in this paper is common for real-time simulation and consists of pre-computing and storing all the inverse matrices $W_{\sigma} = M_{\sigma}^{-1}$, for every switch combination $\sigma$. This strategy can be used only if the number of switches in the power converter is not too large, because the memory requirement to handle such an implementation grows exponentially with the number of switches [30]. The second approach considered in this paper consists of using an iterative method such as the Jacobi or the Gauss-Siedel methods. To the best knowledge of the authors, solving for the linear system of equations of a state-space model is an original approach, more so when it is performed in real-time and on FPGA. The iterative approach permits the inclusion of non-linear components in the simulated power converter, outlined later in this section.

The FPGA simulation results presented in this section are validated against a SPICE-type simulation, using the test bench illustrated in Fig. 7. The gating signals (that would be normally generated by an external controller) are generated using C code embedded in a Matlab-Simulink S-Function. These gate signals, sampled at the FPGA discretization period (denoted $T_5^*$ in Fig. 7), are then saved in a file which is read by both converter models (SPICE and FPGA - Xilinx System Generator). This approach ensures that each model receives the same signals at the same time, avoiding time-shifting of the two responses. The FPGA calculation engines, which are fully described in Section V, are implemented and simulated using Xilinx System Generator® in the Simulink® environment. The test bench that is used in this work offers two main advantages: 1) A SPICE simulation is a precise and realistic reference; 2) The test bench is very convenient and offers an easy way to change the control strategy depending on the topology of the power converter that is under test.

#### A. Pre-computed inverse matrices approach

Computing all the inverse matrices is a well known approach to implementing real-time simulators for circuits containing switches [30]. This approach was used for the implementation of a three-phase two-level voltage source converter with a passive RL load connected to it, as illustrated in Fig. 8. This power converter contains 6 switches and can be expressed using the proposed model with 11 states, as shown in Appendix VIII-A. High-end FPGAs, such as the Xilinx Virtex 5, have sufficient memory capacity to store all $2^6$ possible combinations of switch states, as confirmed by the area occupation results presented in Section V (see table II).

![Fig. 7. Test setup used to validate FPGA simulations.](image)

![Fig. 8. Two-level three phase inverter with inductive load.](image)

**Table I**

<table>
<thead>
<tr>
<th>Component</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$ to $C_6$</td>
<td>20 nF</td>
</tr>
<tr>
<td>$g_1$ to $g_6$</td>
<td>10 m\Omega when ON, 10 k\Omega when OFF</td>
</tr>
<tr>
<td>$r_6$ to $r_9$</td>
<td>5 m\Omega</td>
</tr>
<tr>
<td>$G_R$</td>
<td>0.03 S</td>
</tr>
<tr>
<td>$L$</td>
<td>1 mH</td>
</tr>
<tr>
<td>$L_1$ to $L_3$</td>
<td>100 mH</td>
</tr>
<tr>
<td>$E_1$</td>
<td>100 V</td>
</tr>
</tbody>
</table>
of 200 kHz, and a modulated frequency of 500 Hz. The whole set of parameters used for the FPGA simulation is summarized in Table I. The calculation time-step is 80 ns (see Section V). Load currents from the SPICE and the FPGA-based implementation are shown in Fig. 9(a). These are not steady-state signals but transients, as can be observed by carefully inspecting the shape of the sine waves for the first 6 ms of the simulation time. The absolute error resulting from the FPGA simulation, where the SPICE simulation is used as a reference, is presented in Fig. 9(b). Throughout the entire simulation time, the absolute error is kept under 30 mA, even with the 200 kHz switching frequency. This is about 1.32 % of the observed peak current amplitude (2.25 A), which is a very acceptable result for a system-level simulation, considering that a behavioural-mode simulation is compared with a high-precision detailed-mode SPICE simulation.

1) Parasitic elements selection: For the precomputed approach, parasitic elements are selected directly from SPICE-level simulation. The leakage inductances are in the range of tens to hundreds nanoHenrys. Switch parasitic capacitors are in the range of picoFarads to nanoFarads.

B. Iterative methods: A Gauss-Seidel approach

FPGA-based iterative processors have been proposed for non-linear problem of load flow [32]–[34]. However, these implementations dealt with the problem of FPGA acceleration; increasing the speed (or performance) of a piece of software by mean of FPGA parallelism. This supposes that very large amounts of data must be processed and extremely long computational times must be expected, which does not fall within the scope of this paper. An FPGA-based real-time simulator using the Newton-Raphson algorithm was recently reported in [35], but that work was mainly concerned with non-linear circuits, and the considered time-steps for the paper’s two proposed case studies were 3 µs and 5 µs.

The Jacobi and Gauss-Seidel methods are well established and efficient iterative techniques for solving systems of linear equations with sparse matrices [36], but in general are only considered as an alternative for direct methods in the case of large systems. These methods solve \( Mw = b \) by initially setting \( w \) to a guess value \( w^{(0)} \), then updating this value from \( w^{(n)} \) to \( w^{(n+1)} \) through a sequence of iterations, until convergence is obtained. To ensure convergence, the two methods require the matrix to be diagonally dominant. A
matrix $M$ is diagonally dominant if $|m_{jj}| \geq \sum_{j=1, j \neq i}^{N} |m_{ij}|$ for every $i = 1, \ldots, N$, where $N$ is the size of $M$ and $m_{ij}$ is an entry from $M$ located at row $i$ and column $j$. The Gauss-Seidel method converges faster than the Jacobi method; however, it’s harder to parallelize because of its sequential formulation [36].

In a state-space solver, a solution to $x_{k+1}$ is sought by solving equation (12), which is equivalent to solving $Mw = b$. The approach proposed here consists of using an iterative solver, and setting the guess value $w^{(0)}$ to $x_k$. Numerical experiments conducted for this work have shown that four iterations are adequate for the Jacobi method to converge with sufficient precision; only two iterations are needed for the Gauss-Seidel method. By inspecting the state-space equations (9) and (12), it can be observed that the iterative methods can be used with a guaranteed converge, as long as appropriate values for the parasitic components are found. This can eventually be difficult and may necessitate some trade-offs, since larger time-steps require larger parasitic capacitances to ensure the diagonal dominance of $M$, while larger parasitic capacitances diminish the switching frequency. This justifies the use of the Gauss-Seidel method over the Jacobi method, since using two iterations rather than four helps lower the time-step, and therefore increase the switching frequency.

1) Boost converter case study: In order to illustrate the achievable performance of the Gauss-Seidel iterative method for solving the state-space equations, the boost converter example of Fig. 5 is considered. The time-step for this case study is set to 75 ns, which is the achievable time-step of the FPGA implementation presented in Section V. The $M$ matrix associated with this circuit is given by:

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

where the matrices $M_{11}$ to $M_{22}$ are given in Appendix VIII-B. Using equation (13), one finds the following conditions to keep $M$ diagonally dominant:

$$\begin{align*}
L_1 &\geq 2h (1 - r_1)/3 \\
L_2 &\geq 2h (3 - r_2)/3 \\
C_1 &\geq 2h (2 - G_1)/3 \\
C_2 &\geq 2h (1 - G_2)/3 \\
C_3 &\geq 2h (1 - 1/R_L)/3
\end{align*}$$

Fig. 11. Non-linear boost inductance versus current.

Fig. 12. Simulation of the non-ideal boost converter: (a) Boost inductor current resulting from a SPICE simulation and from the proposed model simulated on FPGA using the Gauss-Siedel method; (b) Relative error – comparison between the results from an SPS simulation and the FPGA-based simulation; SPICE simulation is used as a reference.

Fig. 13. Comparison between the current computed from the FPGA-based simulation and SPICE of a two-level converter using the Gauss-Siedel iterative method.
The terms \( r_1 \) and \( r_2 \) are the copper trace resistors. A value of 5 mΩ is realistic for them. Hence, \( L_1 \) and \( L_2 \) must be greater or equal to 40 nH and 140 nH respectively. The constraint on \( L_1 \) is readily achieved because the boost inductance is generally in the order of \( \mu \)H. The constraints on \( C_1 \) and \( C_2 \) are solely determined by the off-state values associated with the switch conductivity, \( G_{1\text{off}} \) and \( G_{2\text{off}} \), which are close to zero (0.001 is used). This leads to \( C_1 \geq 4h/3 \) and \( C_2 \geq 2h/3 \). \( C_1 = 100 \) nF and \( C_2 = 50 \) nF are convenient. These parasitic capacitors impose a restriction on the voltage rise time. Hence, model error increases with the switching frequency. One finally finds \( C_3 \geq 32 \) nF, which is a constraint easily met since \( C_3 \) is the boost converter load capacitor, and its value is clearly beyond 32 nF.

The case study considers multiple transients. The switching frequency used for this application is set to 20 kHz. From \( 0 \leq t \leq 4 \) ms, the duty cycle of the PWM is set to 40%. At \( t = 4 \) ms, the duty cycle is decreased to 30%, then increased to 50% at \( t = 8 \) ms. The DC voltage of the boost converter is set to 75 V. At \( t = 12 \) ms, the voltage drops instantaneously to 45 V. Two types of boost inductance are considered here. In the first case, the boost inductance is constant and set to 600 \( \mu \)H. In the second case, a non-linearity is introduced into the simulation by considering that the boost inductance is saturable. In both cases, capacitor \( C_3 \) is set to 100 \( \mu F \). The saturation behaviour of the boost inductance is shown in Fig. 11. In both cases, the 75 ns time-step holds, as explained in Section V.

a) **Ideal boost converter simulation:** The results obtained for the FPGA-based simulation of the ideal boost converter are shown in Fig. 10(a). As can be observed, the accuracy of the simulation is good considering that the case study includes very strong current transients and the comparison is made against a SPICE reference simulation. Except for the initial transient where the reference current is close to zero, the relative error is kept below 2% for the duration of the simulation time. Fig. 10(b) compares this relative error to that obtained by a SimPowerSystems (SPS) model, which indicates that the result is acceptable.

b) **Non-ideal boost converter simulation:** The Gauss-Seidel algorithm is useful for handling non-linear components [37]. The results obtained for the FPGA-based simulation of the non-ideal boost converter are shown in Fig. 12(a). Once again, as can be observed, the accuracy of the simulation is good considering that the case study includes stronger current transients than the linear case due to lower boost inductance and that the comparison is being made with a SPICE reference simulation. Except for the initial transient where the reference current is close to zero, the relative error is kept below 3% for the duration of the simulation time. Fig. 12(b) compares this relative error to that obtained by a SimPowerSystems (SPS) model.

2) **Three-phase two-level converter with inductive load:** A third example showing the utilization of the Gauss-Seidel algorithm with the proposed approach is the simulation of the three-phase two-level converter of Fig. 8. The switching frequency is set to 50 kHz and the results are compared with a SPICE simulation. The simulation results and the error are shown in Fig. 13 and Fig. 14 respectively. Fig. 14 shows the absolute error for four switching frequencies (100 kHz, 50 kHz, 25 kHz and 10 kHz), where it clearly appears that the error decreases as the switching frequency decreases. As mentioned previously (Section IV-B), the effect of the parasitic capacitance is prone to errors at higher switching frequencies. However, even for a switching frequency of 100 kHz, the absolute error is well below 0.1 A, which about 4.4% of the 2.25 A current peak.

V. FPGA IMPLEMENTATION

The implementation of a hardware calculation engine for solving the state-space equations in real-time benefits from reformulating the problem in a time-constrained matrix-vector multiplication form [29]. This approach permits the use of an efficient and very regular solver based on parallel multiply-accumulators (MACs) or dot-product (DP) operators [10], [11], [15], [16], [23]. Each operator is then responsible for computing the dot-product of one line of the matrix and the state/input vector (see Fig. 15). Fig. 17 depicts a DP operator with four multipliers at its input. A higher level of parallelism is achieved by a solver based on parallel DP operators rather than MACs, since the operator is capable of processing more columns from the matrix at the same time. For instance, it takes 20 clock cycles for a MAC-based approach (that is, 20 parallel MACs) to read all the operands when multiplying a...
A. **A state-space solver based on the dot-product operator**

As previously stated, it is considered a good practice to rewrite the network equations to ease their on-chip computation, to reduce the area utilization on the FPGA, and to minimize the calculation time-step [16], [29]. Consider equation (12), which presents the general form of the discretized state-space equations used in this paper: \( \mathbf{M} \mathbf{w} = \mathbf{b} \), where \( \mathbf{w} = [x_{k+1}, \mathbf{b}] = \frac{1}{3}x_k - \frac{1}{3}x_{k-1} + \frac{2}{3}h\mathbf{B}u_{k+1} \), and \( \mathbf{M} = \mathbf{I} - \frac{2}{3}h\mathbf{A} \). Let us suppose that the solver stores all the pre-computed matrices \( \mathbf{W}_\sigma = \mathbf{M}_\sigma^{-1} \) for every switch combination \( \sigma \). Then, for a given switch combination \( \sigma \), the new state variables can be computed using:

\[
x_{k+1} = \begin{bmatrix} \frac{4}{3} \mathbf{W}_\sigma, \mathbf{a}, \frac{2}{3}h\mathbf{W}_\sigma \mathbf{B} \end{bmatrix} [x_k, x_{k-1}, u_{k+1}]^T \quad (15)
\]

Hence, the whole computation consists of a very regular time-constrained matrix-vector product. This is efficiently solved using the DP-based solver topology of Fig. 15, which contains six main units: \( i \) the first unit stores the state variables \( (x) \) and the system inputs \( (u) \) processed by the solver; \( ii \) a second unit updates and stores the switch states; \( iii \) these units are followed by memory elements that hold the operands treated by the arithmetic operators, i.e., the matrix variables (stored in ROMs), and sub-vectors from \( x(u) \); \( iv \) a collection of parallel DP operators (the number of DP operators is that of rows in \( \mathbf{A} \)) processes the data; \( v \) the resulting values are registered and provided to the physical world as output values; \( vi \) finally, a control unit is responsible for scheduling all the operations involved in processing the solver. The time constraint in this topology resides in the feedback path that carries the state variables from the outputs of the operators to the inputs of the multipliers. This feedback path coerces the calculation time-step, since the solver cannot step from one calculation time-point to the next until the new state variables are computed. It is worth noting that the implementation of the unit that updates the switch state is straightforward thanks to the natural switching property of the proposed model, and consists of OR-ing the gating signal with the sign of the switch voltage (as suggested in Fig. 2).

B. **Implementing the Gauss-Seidel method**

Only slight modifications are needed to implement the Gauss-Seidel method using the solver topology presented in Fig. 15. The Gauss-Seidel method is usually considered difficult to parallelize as pointed before. In this paper, the parallelization is achieved by exploiting the fact that the problem is known beforehand, and that certain values can be pre-computed to ease the computation. The element-wise formulation of the Gauss-Seidel method is given by [36]:

\[
\mathbf{w}^{(n+1)} = (\mathbf{D} + \mathbf{L})^{-1}(-\mathbf{U})\mathbf{w}^{(n)} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b}
\]

(16)

where \( \mathbf{D} \) is the diagonal matrix of \( \mathbf{M} \), \( \mathbf{U} \) is its upper matrix, \( \mathbf{L} \) is its lower matrix. By initializing \( \mathbf{w}^{(0)} \) to \( \mathbf{x}_k \), by setting \( x_{k+1} \) to \( \mathbf{w}^{(2)} \), and by applying equation (16) to equation (12), the Gauss-Seidel iteration for the state-space solver may be expressed as:

\[
\mathbf{w}^{(n+1)} = \mathbf{M}_w \mathbf{w}^{(n)} + \mathbf{M}_b \mathbf{b}
\]

(17)

It clearly appears from equation (17) that this problem can be solved using the calculation engine illustrated in Fig. 15. In equation (17), the term \( \mathbf{M}_b \mathbf{b} \) can be computed only
once for all the iterations of the Gauss-Seidel method, and added at each iteration to the new $M_w W^{(n)}$ term. However, since this implementation necessitates multiple iterations and pre-computations, two questions come to mind: 1) How to efficiently compute multiple iterations of the Gauss-Seidel technique? 2) Are the memory requirements for the Gauss-Seidel method less than those of the purely pre-computed approach? These questions are answered in Fig. 16, where the boost converter problem is considered. On one hand, it shows that the matrices of the Gauss-Seidel method are sparse. This sparseness is taken into account to speed up the computations. On the other hand, it illustrates the light dependency of each row of the matrices towards the time-varying terms (the switches, the boost inductance). This is always the case when the Gauss-Seidel method is used with the state-space modeling approach proposed in this paper, because the time-varying elements are located on the diagonal of the state matrix. Hence, the memory requirement grows linearly with the number of switches.

C. Number representation

The binary encoding of real numbers is an important issue for FPGA-based solvers. The floating-point format offers certain advantages: it has a large dynamic range coverage and presents a raw compatibility with CPU-based data formatting. However, FPGA-based floating-point operators suffer from an important area occupation and a characteristically long latency, which may limit the complexity and size of the target application. Custom floating-point cores offer an interesting workaround [16], [35], but this may be cumbersome for non-experts. The approach adopted in this paper relies on the fixed-point format; a machine representation for reals that use integers with a fixed number of bits before and after the radix point. The fixed-point DP operators used in this paper are built by assembling multiple DSP blocks in a way that ensures good precision, low latency, and high operation frequency [38].

The DP operator used in this paper is shown in Fig. 17. It has asymmetric $35 \times 25$ input multipliers for the effective usage of the DSP blocks, since these come with $25 \times 18$ signed multipliers in the Virtex 5 and Virtex 6 families. State variables and system inputs are encoded in Fix_25_16 (ie. 25-bit signed fixed-point, with 16 fractional bits), whereas the matrix entries are coded in Fix_35_30. All the intermediate computations within the DP operators are carried out with large bit-widths to ensure high precision arithmetic. As shown in Fig. 17, the adders used in the DP operators are realized either with a single or two DSP blocks. The latter case was used in the design of the three-phase two-level bridge. Its total latency is 8 clock cycles. The first option on the other hand was used in the design of the Gauss-Seidel solver. It has a lower latency (5 clock cycles) but necessitates truncations at the output of the multipliers.

VI. IMPLEMENTATION RESULTS

The time-step of a state-space solver is determined by the number of multipliers in the DP operator, the total latency of the DP operator, and the size of the matrix (number of columns) in equation (15) or equation (17). For instance, the problem formulation for the two-level bridge, in terms of equation (15), uses a $11 \times 23$ matrix. The solver used in this paper for that case study disposes of 11 4-input DP operators, with a total latency of 8 clock cycles each. The time-step can then be evaluated by summing the latency of the DP operator (8), the number of clock cycles needed for processing all the data ($[23/4] = 6$, where $[\cdot]$ is the ceil function), plus one clock cycle for retrieving the data from memory elements and one additional clock cycle for the initialization of the accumulator. This yields $8 + 6 + 1 + 1 = 16$ clock cycles. The solver is operated with a clock period of 5 ns, thus the time-step is $16 \times 5 = 80$ ns.
As previously mentioned, the adders used in the DP operator can be made of a single DSP block rather than two DSP blocks. This yields a lower total latency for the whole operator \((8 - 3 = 5\) clock cycles) at the expense of a slight additional truncation error. This strategy was used in the implementation of the boost converter. The problem formulation of the boost converter in terms of equation (17) uses the sparse matrices presented in Fig. 16. The third row contains the maximal number of elements, which is 10. Hence, the first iteration necessitates \([10/4] = 3\) clock cycles to process the data. The second iteration necessitates only \([3/4] = 1\) clock cycle. The total latency per time-step is given by the sum of the cycles needed to feed the DP with data \((3 + 1)\), the cumulative latencies of the operators during the two iterations \((5 + 5)\), plus one clock cycle for retrieving the data from memory elements: \(3 + 1 + 5 + 5 + 1 = 15\) clock cycles. The solver being operated with a clock period of 5 ns, the time-step is \(15 \times 5 = 75\) ns. The non-linearity is taken into account by means of LUTs for the elements in rows \#1 and \#3. The same solver is used for the linear boost converter case, so the 75 ns time-step holds for both cases.

Table II reports the speed performance and area utilization achieved by the proposed solvers on the VC5VSX50T entry-level high-end FPGA from Xilinx, which is provided with the Xilinx ML506 development board. The utilization of reconfigurable logic is very thrifty, since most of the arithmetic operations are implemented using DSP blocks. The regularity of the proposed topology guarantees that a timing constraint of 5 ns is easily met, as demonstrated with the two case studies considered in the paper.

VII. CONCLUSION

A new state-space approach intended for the real-time simulation of power converters has been presented in this work. The proposed approach is based on a new switch model, which exhibits a characteristically natural switching behaviour. This feature is particularly well-suited for system-level simulations, since the treatments of the semiconductors in the converter are not dependent upon each other, and various converters can be considered without prior knowledge of their topology. The FPGA implementation of the proposed approach for the real-time simulation of power converters has been successfully conducted, and convincing results were presented in terms of achievable time-steps, overall accuracy and computational precision. Two methods have been considered for solving the state-space equations on FPGA: i) pre-calculating network equations for all switch state combinations and ii) the use of the Gauss-Seidel iterative method. The former supports very high switching frequencies but, is limited in terms of the number of switches that can be handled at once. The latter has very low memory requirement regardless of the number of switches in the converter, and thus can be used for the simulation of complex converter topologies. Moreover, it permits the inclusion of non-linear elements in the circuit as demonstrated with the non-linear boost converter case study. Future work will deal with the implementation and real-time simulation of complex converters such as the matrix converter on FPGA.

VIII. APPENDIX

A. State equations for the three phase two-levels bridge

\[
\begin{align*}
\dot{\phi}_1 &= E_1 - i_1 \cdot r_7 - V_{C1} - V_{C2} \\
\dot{\phi}_2 &= -i_2 \cdot r_5 + V_{C1} + V_{C2} - V_{C3} - V_{C4} \\
\dot{\phi}_3 &= -i_3 \cdot r_9 + V_{C3} + V_{C4} - V_{C5} - V_{C6} \\
\dot{\phi}_4 &= -i_4 / G_R - (i_4 + i_5) / G_R + V_{C2} - V_{C4} \\
\dot{\phi}_5 &= -i_5 / G_R - (i_4 + i_5) / G_R + V_{C6} - V_{C4} \\
i_{C1} &= i_1 - i_2 - g_1 \cdot V_{C1} \\
i_{C2} &= i_1 - i_2 - i_4 - g_2 \cdot V_{C2} \\
i_{C3} &= i_2 - i_3 - g_3 \cdot V_{C3} \\
i_{C4} &= i_2 + i_4 + i_5 - g_4 \cdot V_{C4} \\
i_{C5} &= i_3 - g_5 \cdot V_{C5} \\
i_{C6} &= i_3 - i_5 - g_6 \cdot V_{C6}
\end{align*}
\]

B. Terms of the \(M\) matrix for the boost converter case study

\[
\begin{align*}
M_{11} &= 2h \times 2 + \left(\frac{2h}{3}\right) \begin{bmatrix}
\frac{r_1}{L_1} & 0 & 0 \\
0 & \frac{r_2}{L_2} & 0 \\
0 & 0 & \frac{r_2}{L_2}
\end{bmatrix} \\
M_{12} &= 2h \times 2 + \left(\frac{2h}{3}\right) \begin{bmatrix}
\frac{1}{L_1} & 0 & 0 \\
-\frac{1}{L_2} & \frac{1}{L_2} & 0 \\
0 & -\frac{1}{L_2} & \frac{1}{L_2}
\end{bmatrix} \\
M_{21} &= 2h \times 2 + \left(\frac{2h}{3}\right) \begin{bmatrix}
\frac{1}{C_1} & 0 & 0 \\
0 & \frac{1}{C_2} & 0 \\
0 & 0 & \frac{1}{C_5}
\end{bmatrix} \\
M_{22} &= 1 \times 3 + \left(\frac{2h}{3}\right) \begin{bmatrix}
\frac{G_1}{C_1} & 0 & 0 \\
0 & \frac{G_2}{C_2} & 0 \\
0 & 0 & \frac{G_5}{C_5}
\end{bmatrix}
\end{align*}
\]

REFERENCES


Handy Fortin Blanchette (S’07-M’10) received the B.Eng., M.Eng., and Ph.D. degrees in electrical engineering from the École de Technologie Supérieure (ETS), Montreal, P.Q., Canada, in 2001, 2003, and 2010, respectively. From 1994 to 1997, he was engaged in industrial automation. From 1998 to 2000, he was with the Bombardier Transport-ETS Research Laboratory, Montreal, where he worked on a high power traction system. From 2001 to 2003, he was involved in the development of an electrical drive library for the Simulink (MATLAB) environment. From 2007 to 2010, he was with OPAL-RT Technologies, where he led CPU-based and FPGA-based power electronics real-time simulation projects. From 2010 to 2011, he was a Visiting Scholar at Center for Power Electronic System, Virginia Polytechnic Institute and State University, Blacksburg, where he was involved in the packaging of high temperature converters for aircraft applications. He is presently an associate professor of Electrical Engineering at the École de Technologie Supérieure. His current research interests include EMI prediction, circuit modeling, and high density power converters packaging. He is a member of the IEEE.

Tarek Ould-Bachir (S’05-M’08) received the B.Eng. and the M.A.Sc. degrees in electrical engineering from École Polytechnique de Montréal, P.Q., Canada, in 2005 and 2008 respectively. He is a Ph.D. student at the same university, where he also lectures undergraduate courses since 2005. From 2007 to 2009, he worked as an FPGA application specialist at Opal-RT Technologies, a real-time simulation company. His research interests include real-time simulation, digital arithmetic and stochastic computing. He is a member of the IEEE.

Jean Pierre David (M’04) received the degree in electrical engineering from the Université de Liège, Belgium, in 1995 and the Ph.D. degree from the Université Catholique de Louvain, Belgium, in 2002. He has been an assistant professor at the Université de Montréal for three years and is currently an assistant professor at the École Polytechnique de Montréal since 2006. His research interests include reconfigurable computing, high-level synthesis and hardware-software codesign. He is a member of the IEEE.