Journal of Electronics and Electrical Engineering
https://ojs.wiserpub.com/index.php/JEEE

Research Article

Fast and Scalable Simulation Framework for Intensive Power Electronics Simulation through Advanced Computing Techniques

Cayden Wagner[1*], Yi Li[2], Shuangshuang Jin[1], Zheyu Zhang[3], Christopher Edrington[2]

[1] School of Computing, Clemson University, Clemson, SC, USA
[2] Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA
[3] Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA
E-mail: caydenw@g.clemson.edu

Received: 16 October 2024; **Revised:** 3 January 2025; **Accepted:** 6 February 2025

Abstract: Power electronics are widely used in power and energy systems, such as electrified transportation and renewable energy systems. These systems are increasing related simulations' size and complexity, resulting in longer computation times. Advanced computing techniques offer new tools for efficient simulation while designing and simulating complex energy systems. This paper introduces high-performance computing for intensive power electronics simulation and demonstrates resultant simulation speedup in a quantified and scalable manner. First, a quantitative study is performed to compare a slower-than-real-time (STRT) simulation benchmark and the proposed faster-than-real-time (FTRT) simulation through a single power electronics building block (PEBB) case study. The impact of switching frequencies in the range of tens to hundreds of kHz considering wide bandgap (WBG) power semiconductors is also investigated. The simulation speed is observed to be accelerated by a factor of 43.8 when using high-performance computing techniques compared to the sequential-based simulation benchmark. Next, a scalable simulation framework is proposed for expanding a single PEBB to an energy system consisting of multiple PEBBs. The framework leverages the high-performance programming language Julia with multi-threaded parallel computing capabilities to reduce the computational burden of power system simulation. The performance gains from the case study demonstrate an average speedup of 2540 times in a 15.0 s multi-PEBB simulation case study compared to its baseline version, with maintained simulation accuracy and ensured scalability.

***Keywords*:** power electronics, power electronics simulation, high-performance computing, simulation speedup, faster-than-real-time simulation

Abbreviation

Symbol/Term	**Description**
V_{LL}	Line-line root mean square (RMS) voltage
V_{DC}	Direct current (DC) link voltage
i_d,i_q	Direct and quadrature currents in the dq domain
v_d,v_q	Direct and quadrature AC voltages
i_a,i_b	Phase currents in the ABC domain

S_{AH},S_{AL}	Switching functions of the upper and lower switches in phase A
S_{BH},S_{BL}	Switching functions of the upper and lower switches in phase B
S_{CH},S_{CL}	Switching function combinations (e.g., S_{AB}, S_{BC}, S_{CA})
PMSM	Permanent magnet synchronous motor
ω_r	Rotor angular speed
J_m	Mechanical inertia of the rotor
B_m	The friction coefficient of the rotor
ϕ_m	Permanent magnet flux
d_d,d_q	The duty cycle for direct and quadrature components in the control scheme
i_{DC}	DC load current
T_L	Load torque
A,B	Coefficient matrices representing system parameters in state-space equations
$\vec{\mathbf{x}}$	Intrinsic variables (i_d, i_q, v_{DC}) as AC currents in the dq domain and DC voltage for rectifier
$\vec{\mathbf{u}}$	Disturbance variables (v_d, v_q, i_{DC}) as the AC input voltages in the dq domain and DC load current for rectifier
STRT	Slower-than-real-time simulation speed
FTRT	Faster-than-real-time simulation speed
HPC	High-performance computing
PEBB	Power Electronics Building Block
WBG	Wide Bandgap (semiconductors)

1. Introduction

The trend for the penetration of power electronics-enabled power and energy systems to decarbonizing society is irreversible. It is happening in areas such as electrified transportation, power grids with abundant renewable resources, and many others [1, 2, 3]. As a fundamental technique for designing and operating such a system, high-fidelity modeling with high-speed simulation is essential. Today's solutions always compromise between simulation time and accuracy, even with a limited number of power electronics converters in the system, which could be more challenging for next-generation power electronics intensive systems [4, 5].

Power Electronics Building Blocks (PEBBs) are platform-based systems built on reliable, modular components. PEBBs feature defined functionality, standardized hardware, and control interfaces, making them highly versatile and widely adopted across various applications. This standardization enables high-volume production while minimizing engineering effort [6].

The core idea of the PEBB concept is to create an integrated structure for energy conversion devices. Figure 1 illustrates an example of a PEBB circuit, where the chosen topology supports different energy conversion processes (e.g., rectification and inversion) by implementing control algorithms. PEBBs significantly simplify the simulation of power electronics-intensive systems using graphical commercial tools or custom algorithmic models utilizing mathematical equations.

Simulation is a tool that uses circuit theory to show the behaviors of an electric circuit. Researchers have adopted commercial tools with built-in toolboxes and power electronics circuit simulation models from component to system [7]. It includes (1) the PSIM software package at the system level to integrate the entire energy and power system with relatively low fidelity power electronics models; MATLAB/Simulink serves as a platform for system and power electronics simulation, utilizing its integrated graphical block diagram capabilities; (2) the PLECS software tool at the power electronics level, providing dedicated toolboxes for power electronics and component simulation with analog and digital control schemes; (3) Synopsys Saber with many detailed component behavior models for power electronics circuits simulation; (4) Spice, like Saber, with better library availability for power electronics circuits simulation, including detailed component models;

(5) ANSYS and other finite element analysis (FEA) simulation tools with the physics-informed model for multi-domain simulation. Overall, the tradeoff between simulation time and accuracy always exists for these simulation tools.

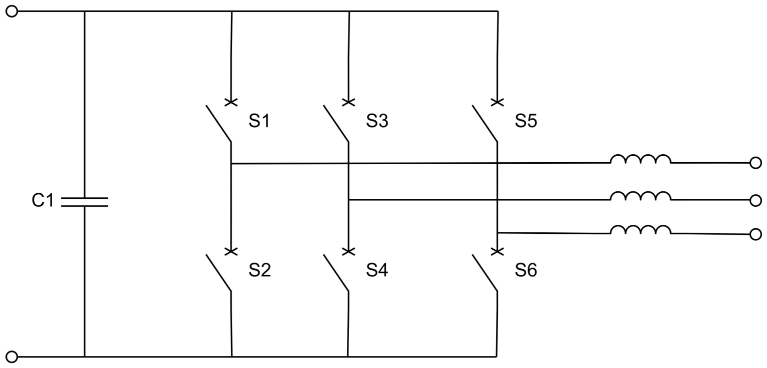


Figure 1. Example PEBB topology

To address the increasing downtime while waiting for simulations to finish, computer science/engineering design concepts can be leveraged to increase simulation speed while maintaining accuracy. High-performance computing (HPC) is one such advanced computing technique employed by the computer science community to improve the "speed of a simulation." When referring to the "speed of a simulation," three main terms describe speed. The first and more common type, seen in yet-to-be-optimized simulation setups, is slower-than-real-time (STRT) simulation, where each simulation step requires more time to execute than the actual time step being simulated. These setups are not acceptable if the execution time of the simulation is an essential factor to the researcher, such as multi-objective converter optimization with extensive design variables, reliability assessment considering lifetime operation, or electric power and energy systems consisting of many interface converters. Next, and quite common, are real-time simulations, which, as the name suggests, run in parallel to the elapsed real time. These types of simulation are common in hardware-in-the-loop (HIL) simulation setups [8]. As these setups are usually commercial products, relying on specialized hardware and licensed software, there is generally no room for further speedup by operators. Lastly, this paper focuses on faster-than-real-time (FTRT) simulation [8]. These simulations can run multiple simulated time steps in an equivalent span of real-time. The main advantage of FTRT simulation is that it takes significantly less time to produce the same results as the STRT simulation. For example, the researcher can run longer stability and reliability simulations or decrease the development time of a power electronics system by reducing wait times for a simulation to complete.

This paper is a revision of [9] with significant additions. This paper uses a direct current (DC) based electrical system consisting of multiple power electronics converters, as shown in Figure 2. It starts with a single power electronics device using the PEBB concept. A quantitative simulation comparison between the sequential-based STRT benchmark and the proposed FTRT simulation is performed, including a brief introduction of the methodology and the results. In addition, to engage more power electronics converters, as practical implementations shown in Figure 2, more than one PEBB simulation is essential. This paper then extends the results of a single PEBB to conduct a PEBB-based energy system. A generic equation is derived based on the standardized PEBB as a foundation to build the STRT simulation model. Finally, a parallel computing technique is adopted to form a scalable FTRT simulation framework. According to this methodology, a case study is presented, demonstrating not only the simulation speedup of a single converter but also the scalability of the simulation framework for a converter-intensive energy system.

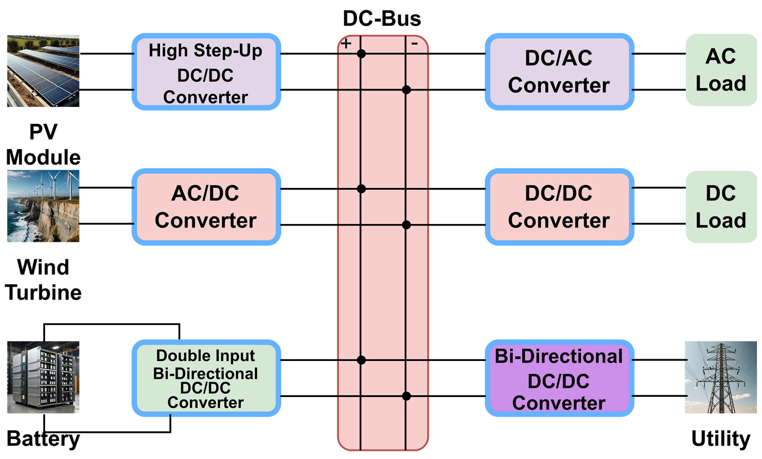


Figure 2. Intensive power electronics in a DC-based energy system

This paper is organized as follows: Section 2 introduces a single PEBB simulation, Section 3 presents a multiple-PEBB-based simulation for a DC-based energy system, and finally, a conclusion is given in Section 4.

2. Single PEBB simulation

2.1 *Methodology*

Taking a three-phase, two-level voltage source converter as an example illustrated in Figure 3a, a single PEBB modeling and simulation can be implemented in the following means: (1) graphical user interface STRT simulation tools, which are widely used for simulating power electronics system (e.g., MATLAB/Simulink), (2) numerical or analytical models are derived from circuit analysis equations and solved using either an ordinary differential equation (ODE) solver or algebraic calculations, and (3) code based model using HPC techniques for FTRT simulation leveraged by the numerical and analytical equations without structural change. Accordingly, a quantitative comparison study is performed based on this same PEBB under a given operating condition.

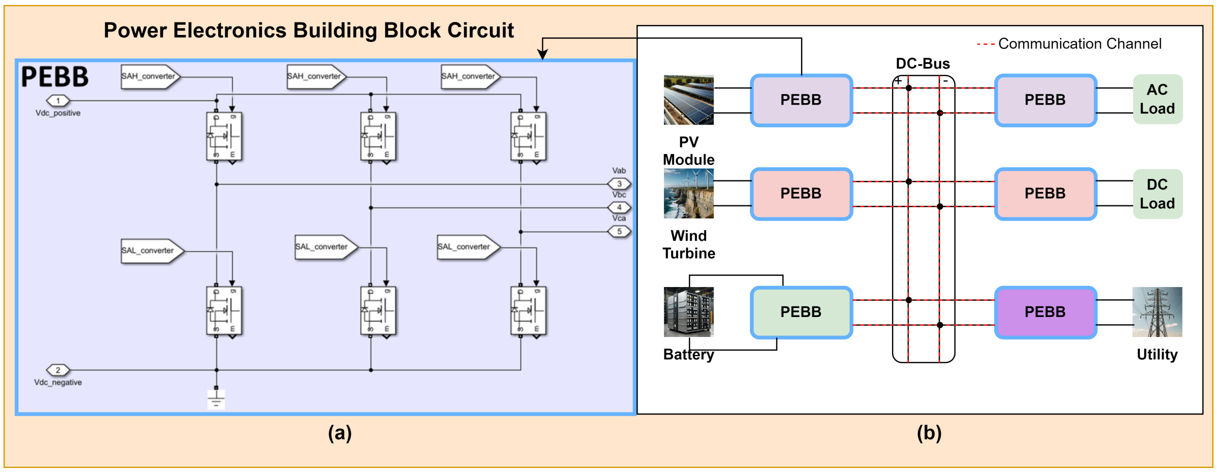


Figure 3. PEBB-based energy system: (**a**) single PEBB configuration and (**b**) high-level illustration of the simulation framework using parallel computing, considering communication

The methodology for conducting a quantitative comparison study is illustrated in Figure 4. It begins with a benchmark STRT model implemented in MATLAB/Simulink, followed by the development of numerical and analytical models. Notably, the numerical and analytical models were initially implemented using MATLAB .m code and subsequently in Julia. Julia is a scientific computing language that happens to be a compiled language, contributing to its competitive efficiency [10].

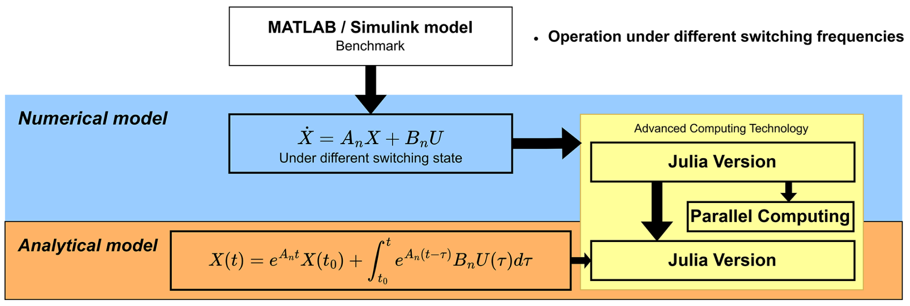

MATLAB / Simulink model
Benchmark
• Operation under different switching frequencies
Numerical model
$\dot{X} = A_n X + B_n U$
Under different switching state
Advanced Computing Technology
Julia Version
Parallel Computing
Analytical model
$X(t) = e^{A_n t} X(t_0) + \int_{t_0}^{t} e^{A_n (t-\tau)} B_n U(\tau) d\tau$
Julia Version

Figure 4. Single PEBB quantitative comparison study methodology

2.1.1 *Benchmark model using MATLAB/Simulink*

Following the example PEBB configuration in Figure 3a, the sequential-based model using MATLAB/Simulink is designated as the benchmark with the circuit-based functional block provided in the MATLAB/Simulink toolbox.

2.1.2 *Numerical & analytical model using MATLAB .m code*

In the numerical model, following a generic equation in Figure 4 and a specific expression of this case study in Equation (1), at the given simulation step, the combination of switching functions associated with individual power devices in the PEBB configuration determines a specific linear circuit consisting of source, load, and passive components. Then, based on the circuit analysis, the coefficient matrices ***A*** and ***B*** in the space state equations can be fixed, allowing the equations to be solved numerically using an ODE solver. Going one step further, the analytical model can be formulated explicitly based on the derived equations, as shown in Figure 4, which provide the same level of accuracy and a more computationally efficient mean than the numerical model.

2.1.3 *Measurement of execution times*

An explanation of how execution times were measured for both single and multi-PEBB systems can be found in Section 3.1 under *Setup and Measurement of Execution Times*.

$$\begin{bmatrix} \mathbf{i_a} \\ \mathbf{i_b} \end{bmatrix} = \begin{bmatrix} -\frac{\mathbf{R_a}}{\mathbf{L_a}} & \mathbf{0} \\ \mathbf{0} & -\frac{\mathbf{R_b}}{\mathbf{L_b}} \end{bmatrix} \begin{bmatrix} \mathbf{i_a} \\ \mathbf{i_b} \end{bmatrix} + \begin{bmatrix} \frac{\mathbf{S_{LL}}}{\mathbf{L_a}} \\ \frac{\mathbf{S_{LL}}}{\mathbf{L_b}} \end{bmatrix} \mathbf{v_{DC}} \tag{1}$$

2.2 *Case study*

Based on the outlined methodology, a case study is conducted using the parameters listed in Table 1, evaluating three tiers of switching frequencies: (30, 100, 300) kHz. The simulation runs for a duration of ten milliseconds (ms).

Mathematical modeling employs numerical and analytical approaches, which rely on the switching functions of individual power devices and circuit analysis. The switching functions of upper and lower switches in the three phases determine the $S_{LL}(S_{ab}, S_{bc}, S_{ca})$ in Equation (1), resulting in eight possible combinations (i.e., eight cases in total). After formulating the equation for each case, the ODE solver is used to solve the corresponding state-space equations.

Table 1. Parameters for the PEBB simulation

Line-line rms voltage	Power rating	Dc voltage	Fundamental frequency	Load
230 V	10 kW	750 V	60 Hz	LR load with 0.9 power factor

Table 2 lists all eight cases and the corresponding switching functions S_{AH}, S_{AL}, S_{BH}, S_{BL}, S_{CH}, S_{CL}, as well as the derived S_{AB}, S_{BC}, S_{CA} (i.e., S_{LL} in Equation (1)) where S_{AH} represents the switching function of the upper switch in A phase, S_{AL} represents that of the lower switch in A phase, and so on.

Table 2. Combination of switching functions in PEBB case study

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
S_{AH}	On	On	Off	Off	On	On	Off	Off
S_{AL}	Off	Off	On	On	Off	Off	On	On
S_{BH}	On	Off	Off	Off	On	Off	On	On
S_{BL}	Off	On	On	On	Off	On	Off	Off
S_{CH}	On	On	On	Off	Off	Off	On	Off
S_{CL}	Off	Off	Off	On	On	On	Off	On
S_{AB}	0	1	0	0	0	1	-1	-1
S_{BC}	0	-1	-1	0	1	0	0	1
S_{CA}	0	0	1	0	-1	-1	1	0

A circuit analysis process is then conducted. Once the switching functions (i.e., on/off) at the given simulation step are determined, the PEBB forms a specific linear circuit consisting of source, load, and passive components, which could be derived as differential equations and state space representations. In detail, the equations for each case are summarized in Table 3, following the generic format in Equation (1) with the specific S_{LL} values derived in Table 2.

Table 3. Equations for eight cases of the example PEBB configuration

Case	Equation
Case 1/Case 4	$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & 0 \\ 0 & -\frac{R_b}{L_b} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} v_{DC}$
Case 2	$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & 0 \\ 0 & -\frac{R_b}{L_b} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix} + \begin{bmatrix} \frac{1}{3L_a} \\ -\frac{2}{3L_b} \end{bmatrix} v_{DC}$
Case 3	$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & 0 \\ 0 & -\frac{R_b}{L_b} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix} + \begin{bmatrix} -\frac{1}{3L_a} \\ -\frac{1}{3L_b} \end{bmatrix} v_{DC}$
Case 5	$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & 0 \\ 0 & -\frac{R_b}{L_b} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix} + \begin{bmatrix} \frac{1}{3L_a} \\ \frac{1}{3L_b} \end{bmatrix} v_{DC}$
Case 6	$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & 0 \\ 0 & -\frac{R_b}{L_b} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix} + \begin{bmatrix} \frac{2}{3L_a} \\ -\frac{1}{3L_b} \end{bmatrix} v_{DC}$
Case 7	$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & 0 \\ 0 & -\frac{R_b}{L_b} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix} + \begin{bmatrix} -\frac{2}{3L_a} \\ \frac{1}{3L_b} \end{bmatrix} v_{DC}$
Case 8	$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & 0 \\ 0 & -\frac{R_b}{L_b} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix} + \begin{bmatrix} -\frac{1}{3L_a} \\ \frac{2}{3L_b} \end{bmatrix} v_{DC}$

2.3 *Results*

The first comparison is conducted among three models: MATLAB/Simulink (used as the benchmark), MATLAB .m code with a numerical model and DOE solver, and MATLAB .m code with an analytical model, all evaluated at a switching frequency of 30 kHz.

As summarized in Table 4, the numerical solution executes the simulation in half the time the benchmark model requires. Furthermore, the analytical solution achieves an impressive execution speed, running 50 times faster than the benchmark.

Table 4. MATLAB/Simulink benchmark vs. MATLAB .m code

Version	Execution time
MATLAB/Simulink (Benchmark)	3624 ms (1.00x)
MATLAB .m code Numerical solution	1303 ms (2.78x)
MATLAB .m code Analytical solution	80 ms (45.30x)

The benefits of creating a power electronics simulator using advanced computing techniques implemented using the Julia language are further explored. Additionally, simulations at two higher switching frequencies (100 and 300 kHz) are conducted to assess the impact of high-frequency power conversion systems enabled by emerging WBG power semiconductors.

Table 5 summarizes the results, which are also illustrated in Figure 5. The findings show that Julia's parallel solution significantly reduces the execution time for numerical simulations. At a switching frequency of 100 kHz, Julia achieves an execution time that is 18.43 times faster than the MATLAB-based solution using .m code. At 300 kHz, this improvement increases to 43.84 times faster than MATLAB.

Even when analyzing the analytical solution, Julia demonstrates a substantial performance advantage over MATLAB, simulating at least 18 times faster for both 100 kHz and 300 kHz switching frequencies.

Table 5. MATLAB .m code vs. HPC Julia language

Version			Execution time	
		MATLAB .m code	Julia	Julia parallel
Numerical solution	30 kHz	1303 ms (1.0x)	147 ms (8.8x)	68 ms (19.2x)
	100 kHz	7691 ms (1.0x)	503 ms (15.3x)	417 ms (18.4x)
	300 kHz	63,922 ms (1.0x)	1511 ms (42.3x)	1458 ms (43.8x)
Analytical solution	30 kHz	80 ms (1.0x)	3 ms (30.7x)	N/A
	100 kHz	150 ms (1.0x)	8 ms (18.3x)	N/A
	300 kHz	584 ms (1.0x)	25 ms (23.7x)	N/A

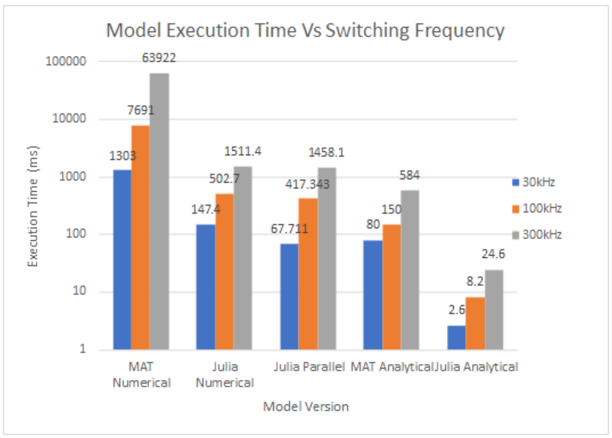


Figure 5. Comparison between MATLAB and Julia

Some key findings that emerge from the results require more illumination: (1) Execution time decreases significantly when transitioning from MATLAB to Julia without significant modification of the existing code architecture. This improvement stems from Julia being a compiled language, inherently faster than MATLAB, an interpreted language. Additionally, Julia offers superior memory management compared to MATLAB. When a program uses more memory, the limited capacity of fast memory (e.g., cache) is exhausted, forcing the program to rely on slower (main) memory, which reduces performance. This difference in memory management is another reason for Julia's faster execution. (2) The computational advantages of Julia's parallel implementation decrease as switching frequency increases, primarily due to slow memory bottlenecks. At higher switching frequencies, the simulation generates more data than can fit into fast memory, resulting in a reliance on slower "main" memory, which negates the parallel speedup observed at lower frequencies, such as 30 kHz. This behavior also explains the observed performance improvement exceeding the theoretical 2× speedup between sequential and parallel Julia implementations. Within a computer's fast memory, there are three cache levels (L1–L3). Splitting a system of differential equations across multiple CPU cores effectively doubles the program's access to the L1 (primary) cache, the fastest memory level, leading to more significant speedups. One possible solution to this issue would be to concurrently extract old data to the main memory or filesystem, freeing up cache space.

3. Multiple-PEBB-Based system simulation

3.1 *Methodology*

This section focuses on the performance and scalability of the multiple-PEBBs-based DC energy system, as illustrated in Figure 3b, with the following methodology.

3.1.1 *Model generalization*

The PEBB concept, with its fixed topology, simplifies modeling for various energy conversion scenarios by standardizing intrinsic, control, and disturbance variables. This allows for the creation of a universal model using state-space equations. PEBBs enable standardized power electronics hardware and their simulation through generalized models. As a result, a foundational PEBB can be designed and implemented for universal parallel computational simulation. This approach benefits designers by reducing design complexity, facilitating time-efficient modeling, and enabling scalable power electronics-intensive energy systems simulation.

Power electronics circuits can be defined using differential equations derived from average modeling techniques and *abc/dq* coordinate transformation. The equations can be rearranged into a standardized format represented by Equation (2) for specific energy conversion scenarios, such as rectification and inversion. In this format, the parameters $\vec{\mathbf{X}}$ represent intrinsic variables (e.g., (i_d, i_q, v_{DC} as AC currents in the *dq* domain and DC voltage for a rectifier), $\vec{\mathbf{u}}$ denotes disturbance variables (e.g., (v_d, v_q, i_{DC} as AC input voltages in the *dq* domain and DC load current for the rectifier). The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ contain coefficients related to power stage parameters (e.g., inductance L and capacitance C) and control variables (e.g., duty cycles d_d and d_q).

$$\dot{\vec{\mathbf{X}}} = \mathbf{A} \bullet \vec{\mathbf{X}} + \mathbf{B} \bullet \vec{\mathbf{u}} \tag{2}$$

3.1.2 *Model implementation*

PEBBs, by nature, can be easily separated computationally with a partial coupling of converters through an information-sharing channel, as seen in Figure 3b. This approach is highly scalable in both design complexity and parallel computational viability. In this case, design complexity and computational viability refer to how easily a new PEBB adds to the simulation and how much slowdown will occur because of this latest addition.

As mentioned, the PEBB concept is promising for creating a scalable power electronics-based architecture that lends itself to parallel code execution. The approach for designing such a framework and scalable system for simulation is analogous to the design of a power electronics system. The PEBBs are considered CPU threads/cores, and the wiring or connections between PEBBs are represented as communication channels. These virtual communication channels allow the PEBBs to affect each other much in the same way as a physical connection. This analogy can be visualized by observing Figure 6a as the physical model and Figure 6b as the abstracted simulation model of Figure 3b. Virtual communication channels also allow for parallel execution, as the PEBBs no longer need to be on the same CPU thread to share information. Consequently, as long as more CPU threads are available to add more PEBBs, the simulation will not slow down appreciably compared to a sequential single-threaded simulation.

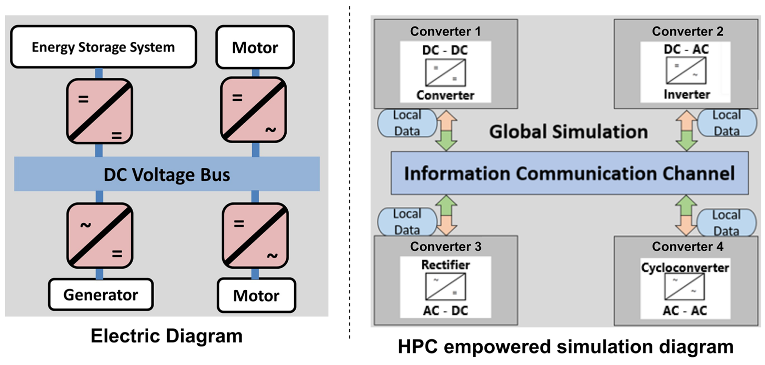


Figure 6. Power electronics building block configuration: (Left: **a**) electrical diagram (Right: **b**) modeling diagram with parallel computing

3.1.3 *Scalability design*

An essential aspect of this PEBB simulation study is testing a simulation platform's scalability. Scalability in this context is the ability of a simulation to add more PEBBs without a noticeable increase in execution time. If a two-PEBB simulation is conducted, this setup can give a sense of how scalable a simulation will be in real-world applications.

3.1.4 *Setup and measurement of execution times*

To test the performance improvement when using Julia and the parallel PEBB simulation, two simulation periods, a half-second (0.5 s) and 15-second (15.0 s) simulation span, are evaluated on all relevant models. Each model has been run 15 times to create a representative mean execution time. Each PEBB is tested individually (labeled "Isolated" below), tested while communicating data and running sequentially (labeled "Combine Sequential" below), and tested while communicating data and running in parallel (labeled "Combine Parallel" below). The data are gathered for MATLAB and Julia on every test configuration mentioned above. All other simulation and program parameters are kept constant for the version of the simulation tested. The hardware and operating system are also kept constant. The operating system is Windows 10 Pro, and the CPU is a 12th Gen Intel i7-12700KF. For the software, Julia version 1.6.5 and MATLAB R2021b.

One crucial constraint is imposed due to time and difficulty: no parallel implementations are created in MATLAB due to the difficulty of creating parallel code in MATLAB compared with Julia.

Lastly, the software used to measure the execution times of all simulation models differs between MATLAB and Julia but remains equivalent in function. For MATLAB, the built-in tic and toc functions were used within the code to measure and record the execution time. In Julia, a macro called @btime from the software package called BenchmarkTools was used.

3.2 *Case study*

This section starts with an architecture with two PEBB devices, as shown in Figure 7. One PEBB is for AC/DC rectification conversion to form a DC link as a rectifier. The other PEBB is a conversion unit for DC/AC inversion, linking a permanent magnet synchronous motor (PMSM) as the AC load. Both PEBBs are designed based on the topology in Figure 3a. In this case study, an average PEBB model is adopted.

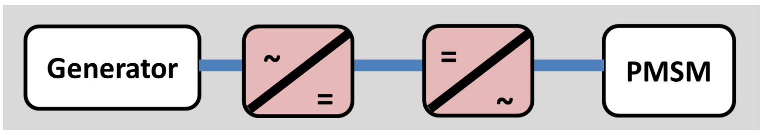


Figure 7. Architecture of the simulation setup used for testing

Considering the PEBB topology in this case study, Equations (3) and (4) show the state space equations of the rectifier and inverter plus PMSM, respectively. These equations are derived based on the average model of PEBB, considering the abc/dq coordination transformation following the format expressed in Equation (2). The intrinsic variables $\vec{\mathbf{X}}$ include AC currents in the dq domain, i_d and i_q. In addition, for rectification-based PEBB, DC link voltage v_{DC} is another intrinsic variable, while for inversion-based PEBBs with a PMSM, one more intrinsic variable is included: the angular speed of the rotor ω_r. Regarding the disturbance variables $\vec{\mathbf{u}}$, the rectifier in Equation (3) includes AC input voltages v_d, v_q and DC load (i.e., the inverter in this case study) current i_{DC}. Also, the inverter and PMSM in Equation (4), DC link voltage v_{DC} regulated by the rectifier and torque T_L are considered.

$$\begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \\ \frac{dv_{DC}}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 2\pi f & -\frac{d_d}{3L} \\ -2\pi f & 0 & -\frac{d_q}{3L} \\ \frac{d_d}{C} & \frac{d_q}{C} & 0 \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ v_{DC} \end{bmatrix} + \begin{bmatrix} \frac{1}{3L} & 0 & 0 \\ 0 & \frac{1}{3L} & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v_d \\ v_q \\ i_{DC} \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \\ \frac{d\omega_r}{dt} \end{bmatrix} = \begin{bmatrix} \frac{-R_s}{L_s} & 2\pi f & 0 \\ -2\pi f & \frac{-R_s}{L_s} & \frac{-\varphi_m}{L_s} \\ 0 & \frac{3p^2\varphi_m}{2J_m} & \frac{-B_m}{J_m} \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ \omega_r \end{bmatrix} + \begin{bmatrix} \frac{d_d}{L_s} & 0 \\ \frac{d_q}{L_s} & 0 \\ 0 & \frac{-p}{J_m} \end{bmatrix} \begin{bmatrix} v_{DC} \\ T_L \end{bmatrix} \tag{4}$$

Terms associated with matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in Equation (2) in this case study include DC link capacitance C and AC inductor L for rectifier; stator resistance R_s, stator inductance L_s, number of pole p, mechanical inertia J_m, friction coefficient B_m, and permanent magnet flux ϕ_m for the inverter with PMSM; and control parameters d_d and d_q. One important note is that the information channel includes DC link voltage data v_{DC} being passed to the inverter from the rectifier and DC load current data i_{DC} passed to the rectifier from the inverter. i_{DC} is determined by Equation (5) as a function of inverter variables in the model Equation (4).

$$\mathbf{i_{DC} = d_d i_d + d_q i_q} \tag{5}$$

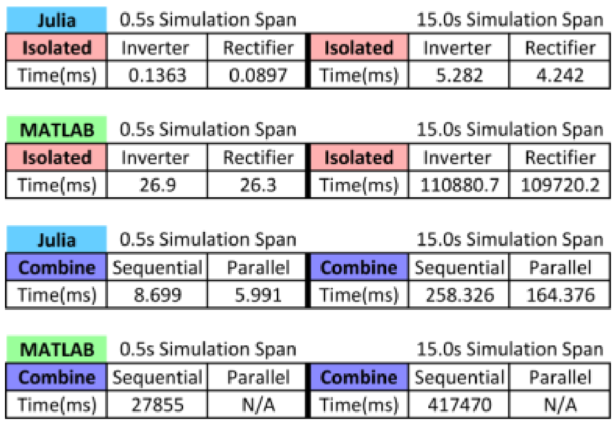
The advanced computing platform and the programming language on which these simulations are set up also play an essential role in performance. The role of these languages is as important as the implementation of the actual simulation. There are many languages used to simulate power electronics-based systems. This paper selects MATLAB, a widely adopted tool used in power electronics, as the benchmark. Julia, a lesser-known but arguably more optimized high-performance programming language for simulation, is adopted for high-speed simulation implementation [11]. Julia has many merits over MATLAB, but the most relevant is that Julia is a compiled language. Unlike interpreted languages like MATLAB, compiled languages enjoy significant performance improvements "out of the box" because of underlying functional differences. Compiled languages have a larger picture of the program, whereas interpreted languages can only make optimization decisions line by line [12]. Furthermore, Julia is designed from the ground up for easy implementation and optimization of HPC schemes [13]. Creating fast and optimized simulations with comparatively minimal effort is objectively more straightforward. Thus, this work chooses Julia as the prime candidate for designing highly optimized FTRT simulations.

Comparing Julia to MATLAB, Julia does have some inconveniences surrounding initial development difficulty. MATLAB has a built-in graphical user interface for designing circuit components like Simulink. Julia is lacking in this area, as all circuits must be initially modeled as mathematical equations before being added to a simulation. However, once these modeled circuits have been added, nothing stops a researcher from creating scalable and copied versions of said circuit. This added inconvenience is more than made up in time saved when running the simulation. Advanced computing techniques of this nature have been used in other areas, such as power systems, to increase simulation speed but not to this degree of performance and not within the domain of power electronics.

The simulation of two PEBBs is implemented for a set amount of time during each run. The first PEBB is a rectifier isolated on a single CPU thread. Alongside the rectifier, but tested separately on a single CPU thread, is the inverter with PMSM. During the simulation, the two PEBBs send the necessary information to each other through a communication channel at every timestep. This process is self-clocking, meaning that each PEBB will wait to receive information from the others before continuing to ensure the accuracy of the generated data.

3.3 *Results*

As shown in Figure 8, Julia has a considerable performance increase due to its greater flexibility compared to MATLAB when applying advanced computing concepts. Looking at the specific values of the 15.0 s Isolated Inverter Simulation Span as an example, when no parallel or combined computation occurs, Julia provides up to a 20,992 (110,880.7 ms/5.3 ms in Figure 9) times speed increase. Focusing on the 15.0 s Combined Sequential simulation results, further performance improvements of 1161 times speedup (417,470 ms/258 ms in Figure 10) are seen in the simulation created in Julia, making it FTRT with a significant reduction in total execution time. The speedup in Julia simulation, when transitioning from sequential to parallel, is between 1.45x to 1.57x, which leads to an average of 2540 times speedup ((417,470 ms/164 ms in Figure 10) Combine Parallel) as compared to its baseline MATLAB/Simulink Sequential simulation. This differential in speedup is expected to increase as more PEBBs are added to the simulation, and the sequential simulation begins to slow down as the computational workload increases.



Julia	0.5s Simulation Span			15.0s Simulation Span	
Isolated	Inverter	Rectifier	Isolated	Inverter	Rectifier
Time(ms)	0.1363	0.0897	Time(ms)	5.282	4.242

MATLAB	0.5s Simulation Span			15.0s Simulation Span	
Isolated	Inverter	Rectifier	Isolated	Inverter	Rectifier
Time(ms)	26.9	26.3	Time(ms)	110880.7	109720.2

Julia	0.5s Simulation Span			15.0s Simulation Span	
Combine	Sequential	Parallel	Combine	Sequential	Parallel
Time(ms)	8.699	5.991	Time(ms)	258.326	164.376

MATLAB	0.5s Simulation Span			15.0s Simulation Span	
Combine	Sequential	Parallel	Combine	Sequential	Parallel
Time(ms)	27855	N/A	Time(ms)	417470	N/A

Figure 8. Experimental data produced by MATLAB Simulink and Julia (Isolated: single-PEBB-based converter–Combine: two-PEBB-based energy system)

It is also worth noting that accuracy is tested between all simulation versions, and the mean error did not reach more than 0.068 for any output variable observed. This is expected as the PEBB equations are algebraic and should not vary. There is some variance due to design choices by MATLAB and Julia on how much precision to use when storing a variable [14, 15].

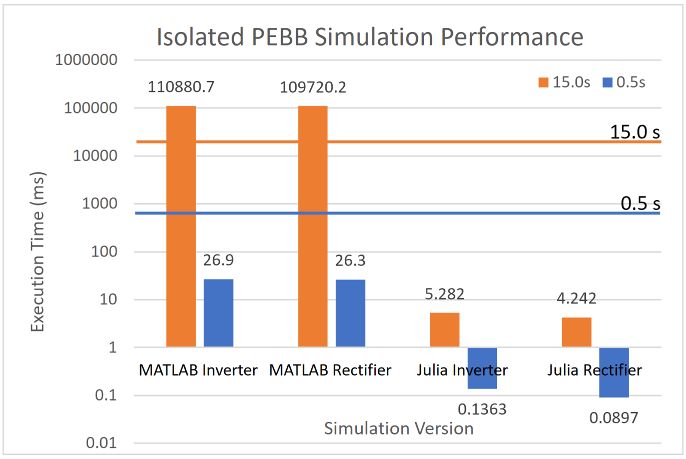


Figure 9. Graphical representation of Figure 7: Single PEBB

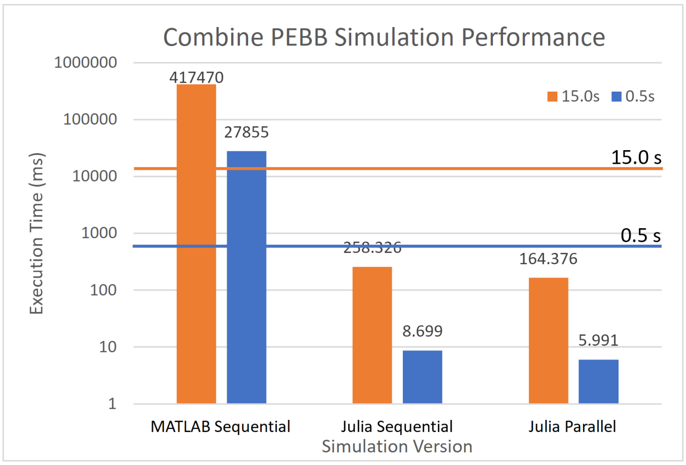


Figure 10. Graphical representation of Figure 7: Combined PEBBs

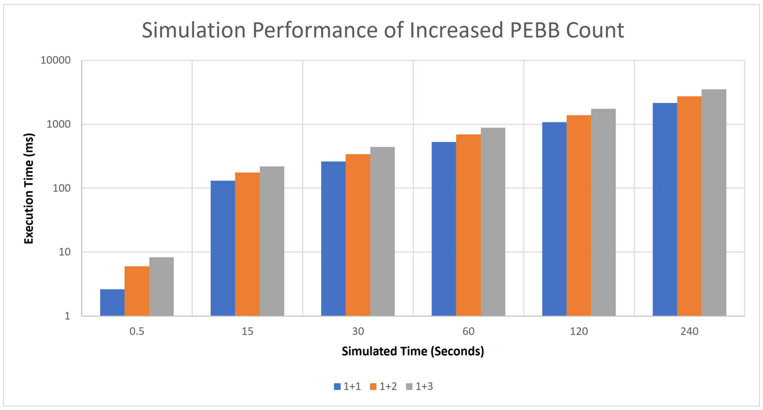


Figure 11. Scalability when increasing PEBB counts

The results in this section show a conclusive narrative that MATLAB has increasingly long wait times for intensive power electronic system simulation. At the same time, our scalable advanced computing framework can maintain an

FTRT simulation speed, as shown in Figure 11, with up to four PEBBs in an energy system. This fundamental difference in capability comes from the design and setup of a scalable simulation while fully utilizing Julia's advanced computing features. Although the initial time needed to create a scalable simulation in Julia is more than the simple graphical interface of MATLAB, it can be argued that with a large enough simulation, both in scope and duration, a researcher can reclaim that lost time spent in development. It is also shown that the sequential execution of a simulation becomes slower and less viable for rapid prototyping as it does not scale well in large systems. Parallel simulation solves this problem with ensured scalability, in both design and computational complexity, of the simulation.

4. Conclusions

This paper has developed a fast and scalable simulation framework leveraging advanced computing techniques to facilitate single PEBB and multi-PEBB-based energy system simulations. First, the performance of multiple simulation models and frameworks is presented. We observed a significant reduction in execution time when switching from the MATLAB benchmark model to the Julia model using advanced computing techniques. This improvement was evident in scenarios using WBG semiconductors with switching frequencies below 100 kHz. Additionally, the parallel implementation of the Julia model further improves the execution time in high-switching frequency applications. Second, a fast and scalable simulation framework is proposed, and its effectiveness is evaluated through a multi-threaded parallel PEBBs implementation in Julia. The result shows accelerated computational performance in simulation with maintained accuracy and ensured scalability. It can be concluded from the simulation results that HPC can help achieve FTRT simulation (2540x) with a mean error of 0.068 as compared to the MATLAB benchmark. Also, the proposed multi-threaded parallel framework can potentially increase the number of converters in a system with a limited simulation time increase, as evidenced by the case study with an energy system consisting of four PEBBs.

Acknowledgments

This work was supported by the Simulation-Based Reliability and Safety (SimBRS) Program for modeling and simulation of military ground vehicle systems, under technical services contract W56HZV-17-C-0095 with the US Army DEVCOM Ground Vehicle Systems Center (GVSC). Distribution A. Approved for public release; distribution unlimited (OPSEC 7236).

Conflict of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] D. Qin, Q. Sun, R. Wang, D. Ma, and M. Liu, "Adaptive bidirectional droop control for electric vehicles parking with vehicle-to-grid service in microgrid," *CSEE J. Power Energy Syst.*, vol. 6, no. 4, pp. 793–805, 2020.

[2] Z. Zhang, H. Tu, X. She, T. Sadilek, R. Ramabhadran, H. Hu, et al., "High-efficiency silicon carbide-based buck-boost converter in an energy storage system: Minimizing complexity and maximizing efficiency," *IEEE Ind. Appl. Mag.*, vol. 27, no. 3, pp. 51–62, 2021.

[3] C. Zhao, B. Trento, L. Jiang, E. A. Jones, B. Liu, Z. Zhang, et al., "Design and implementation of a GaN-based, 100-kHz, 102-W/in^3 single-phase inverter," *IEEE J. Emerg. Sel. Topics Power Electron.*, vol. 4, no. 3, pp. 824–840, 2016.

[4] B. Liu, Z. Zhang, E. Jones, and F. F. Wang, "Application of GaN in hard-switching converters: Challenges and potential solutions," *Electron. Tech.*, 2017.

[5] F. F. Wang and Z. Zhang, "Overview of silicon carbide technology: Device, converter, system, and application," *CPSS Trans. Power Electron. Appl.*, vol. 1, no. 1, pp. 13–32, 2016.

[6] T. Ericsen, N. Hingorani, and Y. Khersonsky, "PEBB-power electronics building blocks from concept to reality," in *2006 Rec. Conf. Papers-IEEE Ind. Appl. Soc. 53rd Annu. Pet. Chem. Ind. Conf.*, Philadelphia, PA, USA, Sept. 11–15, 2006, pp. 1–7.

[7] O. Apeldoorn, "Simulation in power electronics," in *Proc. IEEE Int. Symp. Ind. Electron.*, Warsaw, Poland, Jun. 17, 1996, vol. 2, pp. 590–595.

[8] X. Liu, J. Ospina, I. Zografopoulos, A. Russel, and C. Konstantinou, "Faster than real-time simulation: Methods, tools, and applications," in *Proc. 9th Works. Model. Simul. Cyber-Phys. Energy Syst.*, Virtual Event, May 19–21, 2021, pp. 1–7.

[9] Y. Li, C. Wagner, C. Edrington, S. Jin, and Z. Zhang, "Quantitative analysis of accelerated power electronics simulation using advanced computing technology," in *Proc. 2022 IEEE Appl. Power Electron. Conf. Expo. (APEC)*, Houston, TX, USA, Mar. 20–24, 2022, pp. 274–278.

[10] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, 2017.

[11] M. Stanitzki and J. Strube, "Performance of Julia for high energy physics analyses," *Comput. Softw. Big Sci.*, vol. 5, pp. 1–11, 2021.

[12] L. Xiao, G. Mei, N. Xi, and F. Piccialli, "Julia language in computational mechanics: A new competitor," *Arch. Comput. Methods Eng.*, vol. 29, no. 3, pp. 1713–1726, 2022.

[13] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *arXiv*, 2012, arXiv:1209.5145.

[14] J. Docs, "Integers and floating-point numbers," 2024. Accessed: Sep. 22, 2024. [Online]. Available: https://docs.julialang.org/en/v1/manual/integers-and-floating-point-numbers/.

[15] MathWorks, "Integers and floating-point numbers," 2024. Accessed: Sep. 22, 2024. [Online]. Available: https://www.mathworks.com/help/symbolic/digits.html.