Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
0(0) 1–18
© 2013 The Society for Modeling and Simulation International
DOI: 10.1177/0037549712472755
sim.sagepub.com
SAGE SCS

SimConnect and SimTalk for distributed cyber-physical system simulation

Dylan Pfeifer, Jonathan Valvano and Andreas Gerstlauer

Abstract

Real-time embedded and cyber-physical systems challenge simulation disciplines due to the heterogeneous tools used to model components in the system exploration and design phases. Termed "heterogeneity," the mixed-model problem challenges multi-simulator coordination, where event causality must be preserved among simulators with different models of computation, signals, criteria for time advancement, and levels of abstraction. SimConnect and SimTalk enable heterogeneous, distributed, hardware/software co-simulation with a simplified backplane approach, emphasizing the simulation of software interacting with simulated world-model electrical, mechanical, and physical effects. The structure of SimConnect and SimTalk is described, adhering to the properties of a Kahn Process Network. Application of the tools to the coordination of three different simulators (TExaS, Ngspice, and Simulink) is presented to simulate closed-loop, hardware/software-based, Proportional-Integral-Derivative/Pulse-Width-Modulated control of a direct current motor. Results demonstrate agreement among simulator coordinations with configurable tradeoffs in speed versus accuracy.

Keywords

Kahn Process Networks, distributed hybrid co-simulation, heterogeneous co-simulation, co-simulation backplanes, cyber-physical system simulation, direct current motor Proportional–Integral–Derivative/Pulse–Width–Modulated simulation

1. Introduction

SimConnect and SimTalk address the challenges of system-level design (SLD), considered the "next frontier" in electronic design automation (EDA),[1] as it applies to the design and simulation of cyber-physical systems (CPSs). CPSs are engineered systems integrating "computation with physical processes".[2] Inheriting the challenges of real-time embedded system design, applications of CPSs "arguably have the potential to dwarf the 20th century IT revolution" by virtue of ubiquity and impact.[2] Further discussion and elaboration of systems evolving in this category are given by the National Science Foundation,[3] Klesh et al.,[4] and Rajkumar et al.[5] SLD of CPSs is challenging in that it requires heterogeneous co-simulation of hardware components (such as digital processors and analog electronics), software components (real-time operating systems, software-based digital filters, software-based control, networking protocols), and physical models (such as transducers, dynamical systems, mechanical devices, and biological systems) at flexible levels of abstraction.[1] These component examples are by no means exhaustive. Applying distributed co-simulation techniques to this challenge to gain the speedup benefits of parallelism[6,7] requires coordinating multiple simulators running in their own process spaces with potentially different notions of time. This parallel and distributed simulation (PADS) challenge has been actively researched over the years, with fundamental contributions by Fujimoto,[7,8] Chandy and Misra,[9] and Jefferson.[10] The use of software "backplane" techniques to provide distributed coordination services enabling PADS is covered by Schmerler et al.,[11] Atef et al.,[12] and Sung and Ha.[13] Co-simulation backplanes are independent software or hardware agents that distribute information among process-separated simulators. They may also control and coordinate simulator time advancement for

Electrical and Computer Engineering, The University of Texas at Austin, USA

Corresponding author:
Dylan Pfeifer, The University of Texas at Austin, Electrical and Computer Engineering, 2501 Speedway, Stop C0803, Austin, TX 78712-1684, USA.
Email: dcpfeifer@ieee.org

synchronization. Simulators independently interact with backplanes through software interfaces in their local processes, allowing simulators to execute without global state shared with other simulators. Non-shared state is a requirement of logical process separation in PADS.[8]

A specific challenge arising in the domain of CPSs is the coordination of discrete event (DE) and continuous time simulators, or ''hybrid simulation,'' formal approaches for which are given by Gheorghe et al.[14] and Bouchhima et al.[15] Another challenge is that software is a design component of a CPS.[2] Embedded software is developed and tested on debuggers that single-step, break, and inspect execution of the software over real, emulated, or simulated processor targets. Since a CPS may involve close coupling between software-implemented algorithms and physical processes, it is desired that simulated processor targets interact with extra-processor models (such as electronics, transducers, and physical models) in a way supporting breakpoints and state inspection across the system.

Traditional coordination protocols for backplane techniques, however, can be complex.[16] SimConnect and SimTalk, rather, offer a lightweight backplane architecture and communication protocol that implements the rules of a Kahn Process Network (KPN),[17] such that the tokens of the network are of a type we define called interpolated events (IEs). This simplifies connector software implementation (less than 500 lines of code per simulator in this study), and abstracts the backplane and discrete/continuous synchronization problem to a dataflow model. ''Full'' or ''predicted event'' discrete/continuous coordination[15] may be achieved by adjusting signal producer rates of IEs in the KPN.

2. Related work

SimConnect and SimTalk as they fit in the PADS literature and backplane techniques are introduced in our previous work.[16] Summarizing, the SimConnect backplane implements a KPN and introduces the concept of an IE data type as the token of the KPN. IEs and KPNs applied to the co-simulation synchronization challenge are revisited in Section 3. This structure conforms to the required *local causality constraint* (LCC) of distributed simulation, a coordination rule that simulators must process external events in timestamp order if global event causality is to be preserved.[8] Pfeifer and Gerstlauer[18] apply the SimConnect and SimTalk method to the parallel-execution speedup of an Ngspice simulated circuit, emphasizing the scalability of SimConnect across homogenous simulators. In this paper, we report on the heterogeneous performance of the approach by coordinating a range of independent, heterogeneous, and hybrid simulators, namely the Texas Execute and Simulate (TExaS) microcontroller simulator,[19] Ngspice,[20] and MATLAB/Simulink[21] for the overall simulation of a hardware/software-based control system.

Causality in distributed simulation is covered in the literature in the field of Parallel Discrete Event Simulation (PDES),[7,8] and DE/continuous time co-simulation.[14] For discrete-event/continuous system co-simulation, solutions are offered in the *-AMS languages (SystemC-AMS, Verilog-AMS, VHDL-AMS), and hybrids such as Xspice.[22] Performance comparisons of *-AMS environments are given by Narayanan et al.[23] Heterogeneous two-simulator connections (called ''ad-hoc'' solutions by Schmerler[11]) nearly span the combination set of any two popular environments, but coordinated synchronizations among three or more different simulators apply techniques used in backplanes.

With co-simulation backplanes, an *interface* must be constructed for each simulator that connects to the backplane software agent and implements the coordination API. Because existing solutions implement variants of the discrete/continuous coordination described by Gheorghe et al.,[14] the interfaces and application programming interfaces (APIs) are complex and may not be available for all simulators required. Although the Common Framework Initiative (CFI)[11] offers a standard for simulator connection, it is a complex protocol that may not be fully implemented in every EDA environment. The US Department of Defense (DoD) High-level Architecture (HLA)[24] for distributed simulation is another interfacing approach, arguably ''the most influential standard in the field of distributed simulation''.[25] HLA offers PDES synchronization through time management services in the Run-time Infrastructure (RTI) layer, controlling when federated simulators may advance time.[26] The HLA RTI offers techniques from the PADS intellectual legacy for conservative and optimistic coordination solutions.[26] However, RTI ambassadors for CPS SLD components presently lack widespread instantiation in software debuggers and logic simulation tools. The early potential for HLA as an embedded system development PADS solution is described by de Mello and Wagner,[27] but the remainder of work required to map very large-scale implementation (VLSI) electronic design automation into HLA federates is also acknowledged there. A shift of support for HLA plugins among the major VLSI design vendors, such as Cadence[28] and Synopsis[29] would indicate industry EDA migration to this area. While MATLAB/Simulink[21] now supports a HLA Toolbox, and MATLAB/Simulink is used as a numerical simulation federate by You and Mao-zhi[30] to simulate servo dynamics, in CPSs we also desire to simulate numerous electronic components (such as ones widely supported with Spice models).[20] You and Mao-zhi[30] state that their technique to transform Simulink models into a HLA federate merits improvement (supporting only fixed-step advancement, for example). Therefore, for complexity challenges of integrating numerical system simulation, software debuggers, and VLSI design automation integration into RTI-enforced coordination, HLA as a stand-out solution to SLD for CPSs remains to be developed. However, it could be valuable to CPSs for the multitude of numerous-agent world effects by HLA-compliant simulators and its Institute of Electrical and Electronics Engineers (IEEE) standardization.[24]

ADEVS ("A Discrete EVent Simulator")[31] is another coordination solution in a set of open source C++ libraries that offers hybrid and distributed co-simulation inheriting DEVS (Discrete Event System Specification) formalisms in Zeigler.[32] There is a domain challenge, however, with HLA and ADEVS-like real-time solutions. In CPS design, there may be a range of different scales in simulated time focus, although the requirements and solutions of PADS and PDES still apply. Present HLA simulations may emphasize real-time simulation of minutes or hours of wall-clock time with humans and hardware-in-the-loop, where a CPS simulation may focus on many device internal micro-events happening on a range from picoseconds to milliseconds of simulated time. The time required to simulate the CPS micro-level events could ineffectively prolong the wait of HLA macro-level world events, and would certainly inhibit real-time interactive simulation with present compute technology. SimConnect and SimTalk are designed explicitly for supporting simulators of the internal micro-causality of CPS devices in engineering system design. However, a SimTalk ambassador to HLA is highly conceivable for world-effects model reuse, if the simulated time-scale of the HLA federation is of the order of the simulation time of the SimConnect federate. A SimTalk simulation class instantiation for co-simulation with an ADEVS hierarchy is also conceivable for world-effect model reuse.

In addition, in HLA and ADEVS, simulator time advancement is managed explicitly by an external controlling software agent (the controller libraries in ADEVS, the RTI in HLA). By design, SimConnect and SimTalk rather achieve coordination and simulator advancement strictly through a dataflow network, freeing the backplane from simulator object management, thereby simplifying implementation. The backplane becomes a token router, subject to a KPN (Section 3.3), and analysis of coordination is localized to capture the IE traces independent of internal simulator and backplane structure. This facilitates signal replay and incorporation of simulators with hidden or proprietary internal structure such that they offer a system-level software interface.

3. SimConnect and Simtalk

SimConnect offers a simplified interface and structure, the detail and design of which are described by Pfeifer and Valvano,[16] reducing to the properties of a KPN, summarized in Section 3.2. SimConnect maps the PDES synchronization requirement to a higher-level KPN dataflow structure with restrictions on data tokens as covered in Section 3.3. This greatly simplifies and focuses the backplane software and simulator interfacing. Simulators connect to the backplane in a client/server relationship. Information is distributed in a publish/subscribe architecture,[33] where each client publishes its output signal activities to the server, while also subscribing to selected input signal events for one-to-one or one-to-many information distribution. The IE token restriction allows analytical integration of heterogeneous computational models, since IEs compose to piece-wise constant functions, enabling exploration of speed-versus-accuracy tradeoffs by static or dynamic variation of IE duration.

3.1 SimTalk

At a high level, clients send messages to the SimConnect server through the SimTalk message protocol,[16] which facilitates signal publish and subscribe requests, and enables read or write operations to connection first in, first outs (FIFOs) per the rules of a KPN for signal updates. The simulator interface therefore only implements token passing in the SimTalk protocol, and the backplane only implements the KPN FIFOs and the node connection graph. Operationally, the backplane forwards tokens from signal producer FIFOs to signal consumer FIFOs in a cyclic service loop. KPN nodes are connected, concurrent, and independently running simulators. Blocking reads on FIFOs combined with signal producer rates realizes the requirements of simulator coordination if the data tokens are of a type we call IEs,[16] expanded in Section 3.3. An IE token, read from an input FIFO, communicates both signal value and value expiration, providing a future event time when the simulator must re-sample the input FIFO. This removes time step and synchronization control from the responsibility of the backplane and interface API. The synchronization and control are implied in the token data and dataflow, configured by token update rates of signal producers. These can be assigned statically or changed dynamically during the simulation. SimConnect and SimTalk therefore map the synchronization challenge to a dataflow network, facilitating debug, mathematical analysis, and replay.

3.2 Kahn Process Networks

KPNs, named for Gilles Kahn,[17] are dataflow networks with the following properties:

- The KPN is a directed graph with arcs, representing simplex FIFOs, and nodes, representing concurrent compute elements without interdependent side-effects.

- Nodes may read from input FIFOs and write to output FIFOs, but reads are blocking (the node stalls) if the FIFO is empty, while writes always succeed (the node does not stall).
- Nodes may not conditionally execute by FIFO sniffing.
- FIFOs are unbounded (infinite depth).
- The KPN is independent of the order of node execution if the KPN dataflow rules are followed. The KPN is completely determined by its node set, arc set, initial conditions, and FIFO producer rates.

KPNs reduce to synchronous dataflow networks if token production rates are static and known a priori.[17] KPNs are deterministic based on initial conditions if the FIFO rules are followed. The KPN rules are sufficient to solve the simulator coordination problem if the KPN tokens are IEs, if FIFOs are simulator signal connections, and if the KPN nodes are concurrent, distributed simulators. This simplifies the co-simulation interface and backplane architecture while providing a strong model structure (KPN).

3.3 Interpolated events

SimConnect KPN data tokens are IE data types, introduced by Pfeifer and Valvano.[16] IEs are 3-tuple set elements (v, t_m, t_n) from the product set $V \times T \times T$, where $\{V\}$ is a set of values, and $\{T\}$ is a set of tags. This nomenclature borrows from the value/tag ''(v, t)'' definition of an event covered by Lee and Sangiovanni-Vincentelli.[34] For a given IE (v, t_m, t_n), the value v is defined to be constant on the interval $[t_m, t_n)$ specified in the IE, such that the tag set $\{T\}$ is ordered. $\{T\}$ is conventionally the real number set $\boldsymbol{R}^1$ in timed, event-driven simulations, representing the simulation time stamp when an event occurs. For an IE (v, t_m, t_n), the range $[t_m, t_n)$ assigns a ''stable'' time to the signal value v for producers and consumers.

If a simulator consumes an IE (v, t_m, t_n), it may assume the value v is constant on the tag range $[t_m, t_n)$, and not need to sample the value again until expiration time t_n. So, an IE encapsulates both communication (the signal value) and synchronization (the start and end time). Mapped to nodes in a KPN, simulators consume IEs, run, and produce IEs until the expiration tag of the last consumed IE, at which point simulators sample their FIFOs again for a new IE. If their input FIFOs are empty, simulators block, enforcing the LCC, because each simulator cannot advance in time beyond the expiration tags of IEs on its input FIFOs.

As a consequence of sampling implied in the duration of an IE, there is a tradeoff in speed versus accuracy with this approach. An IE assigns a stable value for duration to a signal, during which local time a consuming simulator can operate on it without re-querying the value. During that time, however, the signal may change, resulting in sample-and-hold error for continuous values, or change-delay error for digital values, since the state change information of the digital value is delayed until the start time of the next IE. This speed-versus-accuracy tradeoff can be statically tuned or dynamically adjusted, however, as explored by Pfeifer and Gerstlauer[18] and Section 5.

A benefit of the KPN approach is that control of the global simulation is achieved by dataflow dynamics, rather than a central controller (the backplane is not a controller, but a token router). If one signal producer is paused, for example, each consuming simulator of the signal blocks when it reads its input FIFO for that signal. This has a desired effect in source-based debugging of software in a co-simulation. When a software breakpoint is reached in a debugger, the architectural states (registers and memory) freeze for inspection. With SimConnect, any other consuming simulators also freeze, allowing inspection of system components (such as circuit levels) at the time of the breakpoint without probe interference. When the breakpoint is passed, and the dataflow resumes, the consuming simulators continue with their local time preserved, *without direction of a central controller*. Any signal-producing simulator can be paused to pause all of its consuming simulators and resume with time correctness, completely as a result of dataflow and the KPN blocking read property.

Implementing the SimTalk protocol can be done through any blocking, distributed message-passing API. In this study, SimTalk is implemented through Berkeley Software Distribution (BSD) socket calls with blocking reads, non-blocking writes, and ASCII string message content sent over transmission control protocol/internet protocol (TCP/IP) for reliable delivery.

4. Synchronization

As an example of synchronization, IEs can achieve conservative, predicted event[15] synchronization between the TExaS simulator, a time-driven, clocked synchronous model of computation (MoC), and Ngspice, a time-driven, dynamically stepped MoC. In the clocked synchronous MoC, input and output signal exchanges occur at ends of a fixed period, but a one evaluation cycle delay occurs from signal input to output result. For example, the TExaS simulator, which realizes a cycle-estimating simulator for the Freescale 9S12 microcontroller, continually executes a GetInputs(), Evaluate(), PostOutputs() cycle in its time advancement. With a local evaluation cycle of 125 ns, and static IE duration of 125 ns, there is a 125 ns delay from the operational effect of an input appearing on an output if they are related.

In Figure 1, the Xspice socket devices and TExaS post output IEs of $(t_n - t_m)$ = 125 ns duration, the predicted event interval, or the TExaS evaluation cycle. Initial condition IEs of 125 ns duration are posted to FIFOs at startup so each simulator can advance and post after the first Get() SimTalk operation. The IEs posted by TExaS, consumed

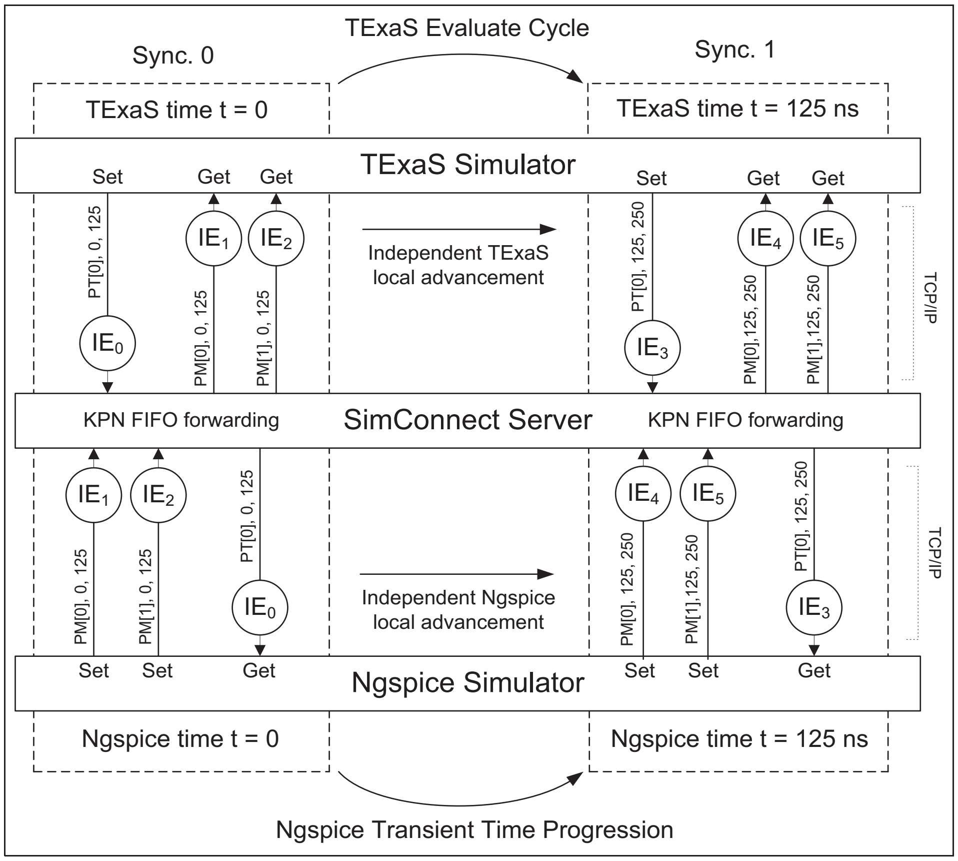

Figure 1. Conservative, predicted event synchronization.

by Ngspice, allow the Ngspice kernel to compute freely for 125 ns, posting IEs before blocking at sync point 1 in the figure, where it re-queries the SimConnect server again. Up to each sync point, established by the expiration time of an input IE, simulators advance independently without local time coordination. For resolution, if the IE durations are less than the TExaS evaluation cycle, that is $(t_n - t_m) <$ 125 ns, the FIFOs oversample. If the IEs are greater in duration, $(t_n - t_m) > 125$ ns, the execution is optimistic, since TExaS advances to the next evaluation cycle on an IE that could change during the evaluation cycle, but which was declared to be constant on an IE range larger than the cycle. If the IE duration increases further, $(t_n - t_m) \gg 125$ ns, the signal resolution decreases, decreasing message count, increasing the time between synchronizations, but decreasing accuracy. However, for low-frequency input signals compared to the TExaS clock, resolution can be decreased to an appropriately large $(t_n - t_m)$ period, say the period of the Nyquist frequency of the input signal. If an appropriate resolution is unknown, it can be observed first at a high resolution rate $(t_n - t_m) \ll 125$ ns, then adjusted to a lower resolution to increase simulation speed. The LCC is observed because simulators do not advance beyond time points on inputs until an input IE is provided for that time point with a future expiration value (t_n).

If the co-simulation is strictly composed of DE simulators, with no continuous time components, the expiration t_n value can be seen as a look ahead value[8] and lower bound on the t_m value on all future IEs in an input FIFO. The LBTS value[26] can also be considered for the signal in the HLA RTI conservative synchronization terminology. The signal value is declared invariant over the IE duration. An IE (v, t_m, t_n) can also be considered as the union of event (v, t_m) and the set of all Chandy/Misra/Bryant NULL messages[8,9] $\{(t_i, \text{NULL})\}$ such that $t_m < t_i < t_n$. If the IEs are configured in duration conforming to the cyclic constraints given by Chandy and Misra[9] (no collective zero timestamp increments around the cycle),[8] then deadlock cannot occur by construction of the dataflow graph and token delivery. Deadlock can occur if a single simulator internally halts, ceasing its signal production (thus blocking consuming simulators), or if finite-memory, finitely sized FIFO buffers in the backplane fill up.

SimConnect presently makes no attempt to break deadlock, since so far in testing a deadlock indicates an IE duration configuration error. In simulations by Pfeifer and Gerstlauer,[18] deadlock would occur when a Spice simulator rejected a time-point solution and re-queried the SimConnect server for an input IE delivered earlier. To prevent this deadlock, SimConnect keeps a configurable deep (presently 1-deep) IE-sent history linked-list for each FIFO. A Spice rollback could generate different output IEs, but in these simulations the rollback time point was less than the next configured Xspice discrete time point, so the Xspice discrete engine did not recompute discrete outputs. It is noted this could happen, however, requiring rollback support in the other connected simulators or a cancellation method (such as lazy cancellation)[8] due to a production of a straggling IE of a different value due to a Spice recalculation from a dynamic time-step reduction. These effects emphasize that choice of IE duration is the determining quality of a SimConnect-based simulation. IE duration affects both synchronization and the dynamically configurable tradeoff in speed versus accuracy[18] offered by a SimConnect solution.

5. Study: control system simulation

Pfeifer and Gerstlauer[18] apply SimConnect and SimTalk to the parallel speedup of an Ngspice simulation over up to 128 independent, coordinated Ngspice simulators. In this study, SimConnect and SimTalk are applied to coordinate three simulators – TExaS, Ngspice, and Simulink – to simulate a closed-loop, software-based, Proportional–Integral–Derivative (PID), Pulse–Width–Modulated (PWM) controller of a direct current (DC) motor. We progressively build up the model in terms of realism and heterogeneity.

5.1 Configuration: one-simulator classical continuous PID controller and second-order DC motor model in Simulink

For a truth condition, we model the DC motor initially in Simulink in continuous time as a second-order system in the Laplace domain, with a transfer function given in Figure 2. The model is taken from Franklin et al.,[35] where it is derived from first principles of Kirchhoff's voltage law (KVL), Kirchhoff's current law (KCL), and Newton's laws applied to rotation. The transfer function in the complex frequency domain is s, where V is the applied terminal voltage in volts and Θ is the rotor output position in radians. The model is parameterized for the simulation as shown in Figure 3.

The torque constant K_t represents the electromechanical multiplier of the armature current to rotor torque. The electrical constant K_e represents the multiplier of back electromotive force (back-EMF) to rotor speed. The electrical equivalent circuit is given in Figure 4. Coefficient b is a drag force, and coefficients L and J are integrating resistances to applied voltage and applied torque, which go to zero in the steady state with a constant terminal voltage. The motor runs up to a steady-state

$$\frac{\Theta(s)}{V(s)} = \frac{K}{s(\tau(s)+1)}$$

$$\text{such that } K = \frac{K_t}{bR+K_tK_e}$$

$$\text{and } \tau = \frac{RJ}{bR+K_tK_e}$$

Figure 2. Second-order direct current motor model transfer function.[35]

R	motor terminal electrical resistance	1.0 Ω
L	motor terminal inductance	0.001 H
K_t	torque constant	0.1 Nm/A
K_e	electrical constant	0.1 Nm/A
b	rotor viscous friction coefficient	0.001 Nm · s
J	rotor moment of inertia	0.01 kg · (m/s)^2

Figure 3. Direct current motor model parameters.

speed as the back-EMF increases per the rotor speed, and current equalizes to meet resistive and friction losses.

The model is converted to a Simulink block-diagram form as summing, integrating, and gain blocks in Figure 5, illustrating the feedback relationship between the motor electrical and mechanical dynamics.

The model is encapsulated as a Simulink subcircuit, and an open-loop 5 V step-function is applied in Figure 6 to achieve the no-load, open-loop speed run up plot in Figure 7.

5.2 Results: one-simulator classical continuous PID controller and second-order DC motor model in Simulink

With the no-load 50 radians/s speed as a ceiling, we add a Simulink continuous time PID block in Figure 8, configured to a set point of half-speed 24 radians/s, 5 V output ceiling, with K_p, K_i, and K_d coefficients of 8, 2, and 1, respectively. The closed-loop transient response is plotted in Figure 9, with controller effort from the PID block in Figure 10.

There is an expected overshoot in the continuous controller model with the given coefficients and lack of any limiter, such as anti-integrator windup.[35] From the controller effort, the PID block outputs full ceiling power (5 V) until the set point is approached, after which it drops to the steady-state output necessary to equalize electrically resistive and mechanically viscous friction losses at the constant speed.

For a first departure away from the idealized continuous model, we quantize the motor speed output to a range of 128 values with a Simulink 8-bit quantizing block with an

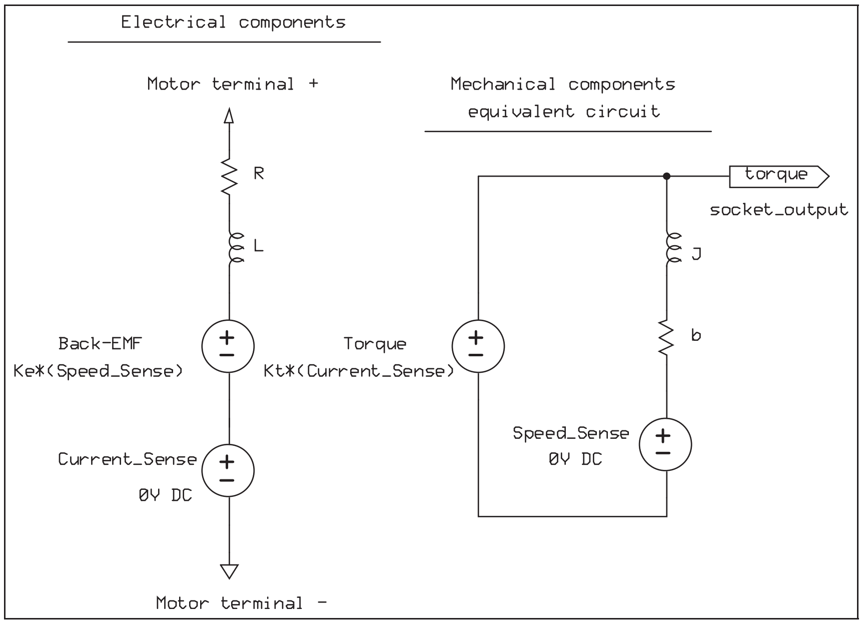

Figure 4. Ngspice models for direct current (DC) motor electrical and mechanical components.

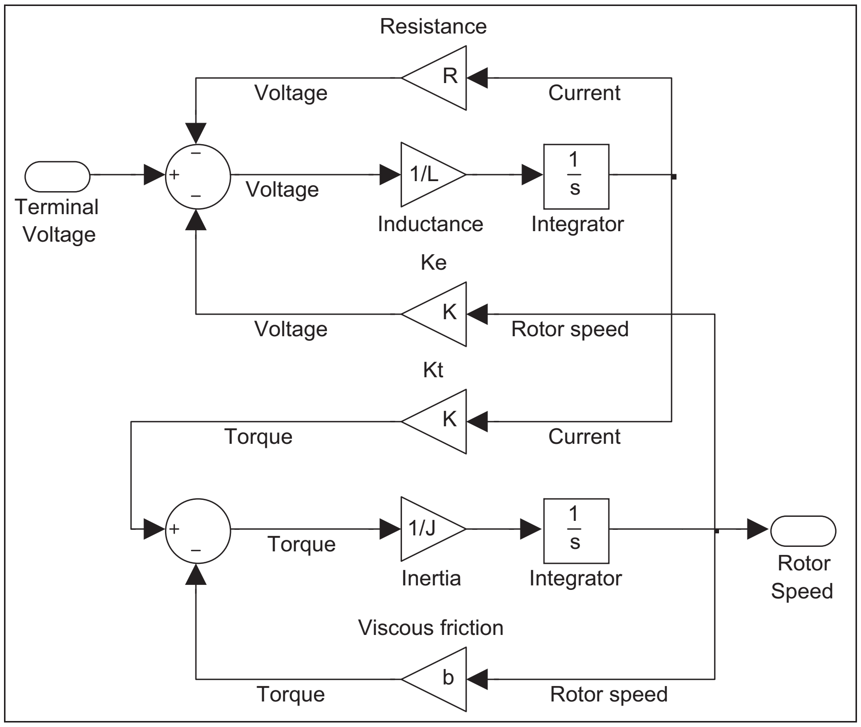

Figure 5. Simulink direct current motor electro-mechanical model.

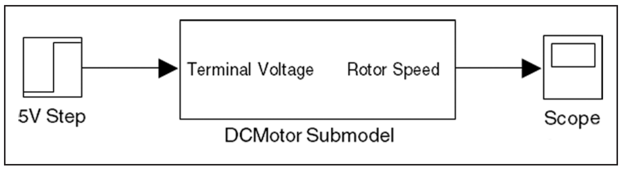

Figure 6. Simulink open-loop 5 V step-function stimulus.

offset given in Figure 11. This allows us to express the set point as one-half (hex 40) of full value (hex 7F) rather than an absolute rotor speed, and serves as an abstracted 7-bit analog-to-digital converter (ADC) that will be used as in input to the software-based controller. The effect of quantizing the measured speed for the PID transient response is given in Figures 12 and 13.

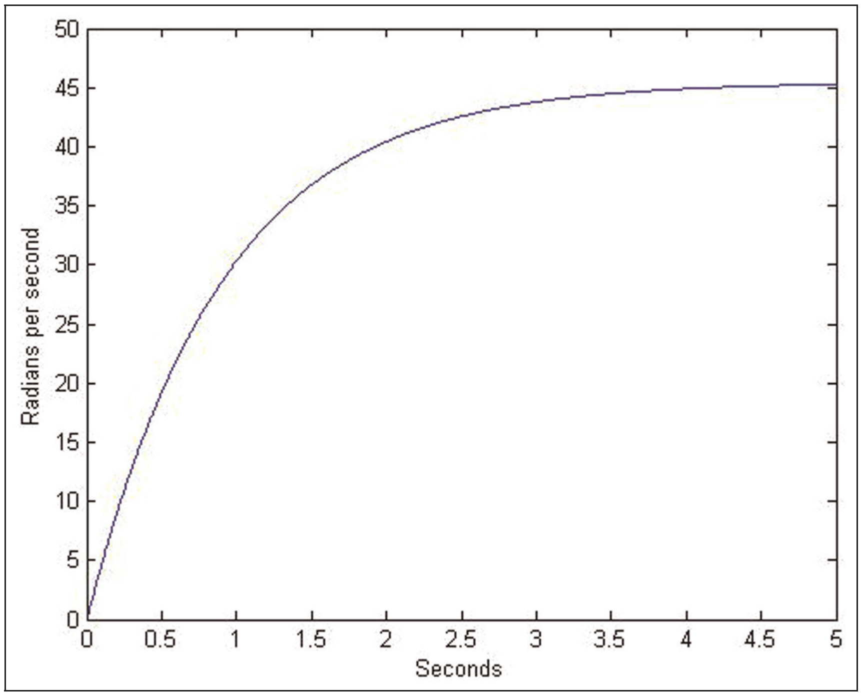

Figure 7. Model rotor speed versus time, open-loop transient response to a 5 V step-function.

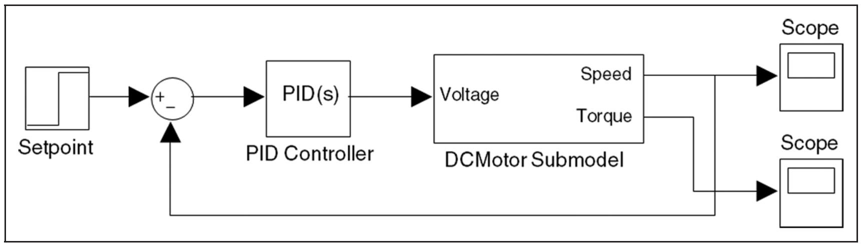

Figure 8. Simulink continuous Proportional–Integral–Derivative (PID) controller.

5.3 Configuration: two-simulator digital software PID with PWM actuator in TExaS and Simulink DC motor model

Next, the controller is refined to a software-based PID difference-equation algorithm with PWM actuators hosted on the 9S12 microcontroller, simulated with TExaS at the cycle-estimating, instruction-set architecture (ISA) level. The co-simulation signal structure is given in Figure 14.

The software-based PID algorithm in the 9S12 assembly is adapted from Valvano.[19] The PID coefficients are changed to $K_p = 24$, $K_i = 2$, and $K_d = 1$ for the difference-equation PID coefficients in fixed-point arithmetic. Conversion of a continuous time frequency domain specified controller to a digital controller is covered in Franklin et al.[35] Setup and allocation code for the 9S12 out of reset, and the PID assembly language code adapted from Valvano.[19] This implementation uses a free-running 1 kHz sampling rate PID main loop of 63 9S12 assembly instructions, and a total code length of 158 instructions. The algorithm also incorporates anti-integrator windup and output limit checking. The refined Simulink model is given in Figure 15. The ‘‘socket_input’’ and ‘‘socket_output’’ S-Functions register SimTalk signals PortT[0] and PortM[7:0] for exchange with the SimConnect server. The PortT[0] digital signal is the PWM wave generated by TExaS. The 0/1 signal is amplified to 5 V for application to the motor terminals. The PWM wave and voltage in this configuration is modeled as ideal (zero rise/fall time).

5.4 Results: two-simulator digital software PID with PWM actuator in TExaS and Simulink DC motor model

The two-simulator model is conducted at 100 µs IE resolution on signals PortT[0] and PortM[7:0]. The transient

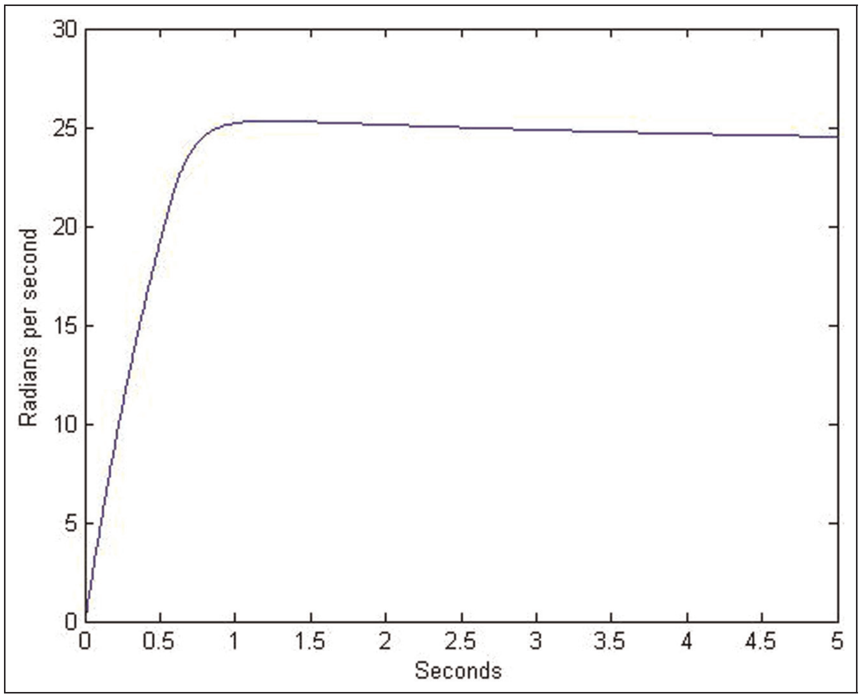

Figure 9. Model rotor speed versus time in Simulink continuous Proportional–Integral–Derivative (PID) controller closed-loop transient response.

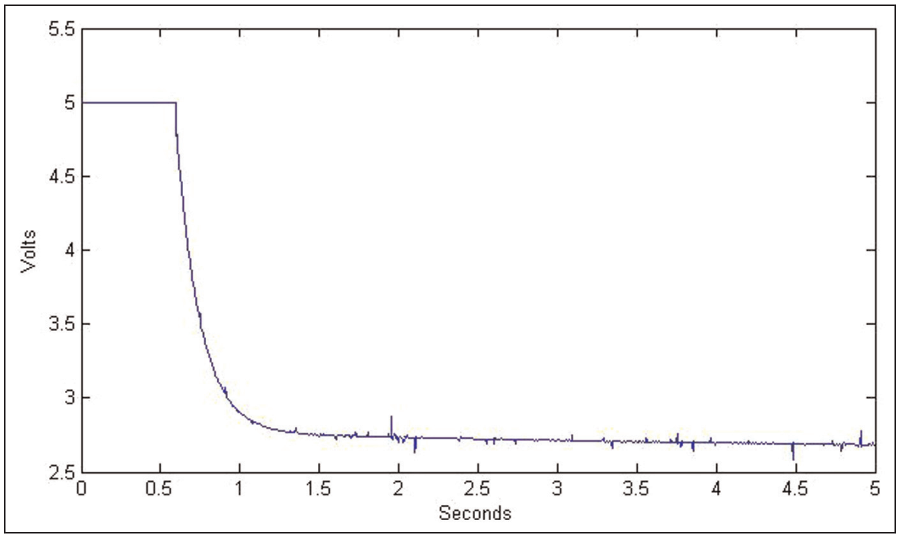

Figure 10. Model controller effort in Simulink continuous Proportional–Integral–Derivative (PID) controller transient applied voltage.

response is plotted against the Simulink-only classical continuous and encoded continuous cases in Figure 13.

The functionality of the software-based PID controller is verified in Figure 13, as the set point is reached in the steady state. The set point approach differs from the classical case due to sampling, digitization, increased proportional gain, and anti-integrator windup. However, there is new departure from the continuous model because the applied terminal signal is a PWM signal from the 9S12 microcontroller, and the PID algorithm is realized in the microcontroller software. The 100 µs two-simulator response is used as a baseline for checking the three-simulator case, where electrical realism in the motor driver is added to the simulation.

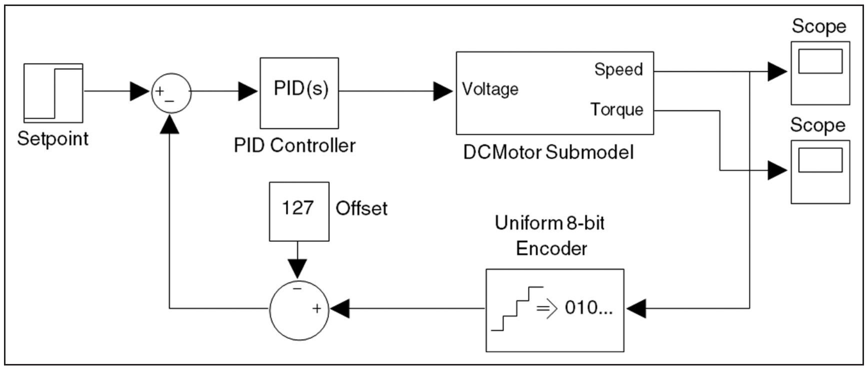

Figure 11. Simulink quantized Proportional–Integral–Derivative (PID) controller.

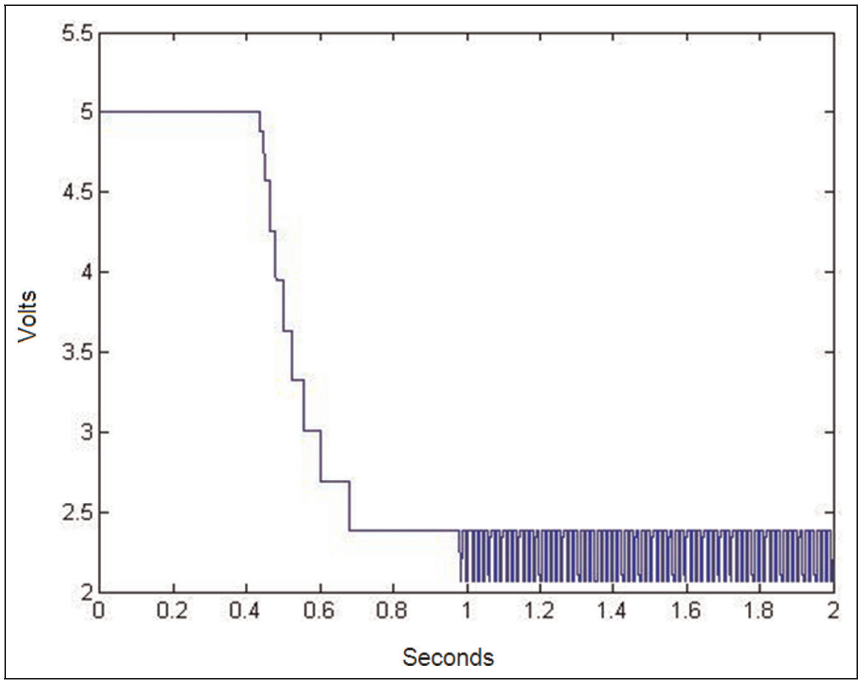

Figure 12. Quantized output controller effort in Simulink quantized Proportional–Integral–Derivative controller applied voltage.

5.5 Configuration: three-simulator digital software PID with PWM actuator in TExaS, electrical driver and DC motor model in Ngspice, and Simulink DC motor mechanical model

In the final configuration, electrical realism is added by modeling the motor driver circuitry and motor electro-mechanical model in Ngspice, duplicating the motor mechanical model in Simulink for output speed. Figure 16 illustrates the co-simulation structure.

The PWM driver circuit is adapted from Valvano[19] and given in Figure 17. Sometimes called a ‘‘chopper’’ circuit,[36] the power MOSFET circuit in Figure 17 takes the low-power PWM PortT[0] PWM signal from the 9S12 and amplifies it across the motor terminals to the power voltage. When the PWM signal is high (5 V), the MOSFET is fully on, so current flows through the motor coil, and when the PWM wave is low (0 V), the MOSFET is off, interrupting the flow of current from the power source. There is still current flow when the MOSFET is off, however, due to the back-EMF and impedance of the DC motor. The high-voltage back-EMF and impedance when the PWM wave changes is collected through the 1N4004 ‘‘flyback’’ diode to protect over-voltage at the MOSFET drain.

The DC motor electro-mechanical equivalent circuit is modeled in Figure 4, where the applied torque is captured with the Xspice SimTalk ‘‘socket_output’’ device for delivery to the Simulink model.

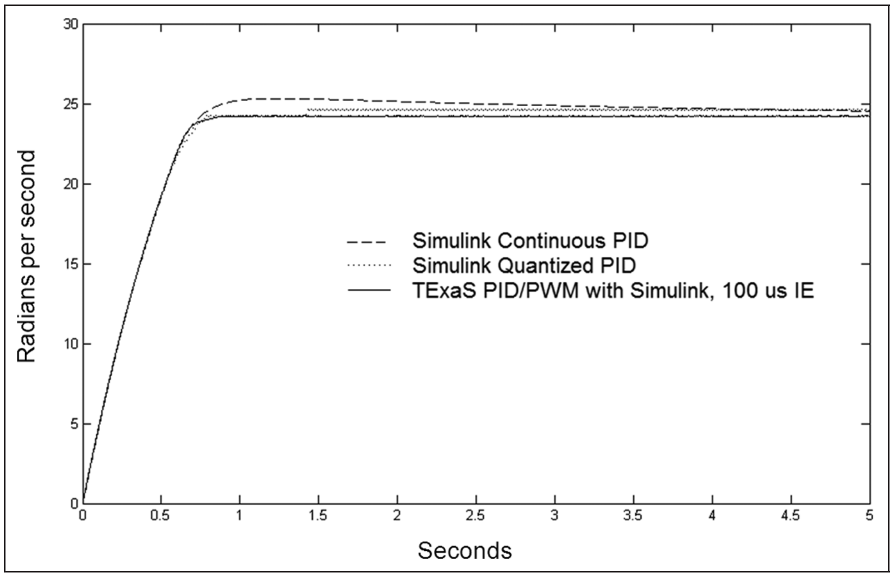

Figure 13. Model output speed versus time with Simulink-only and two-simulator Proportional–Integral–Derivative (PID) control model cases.

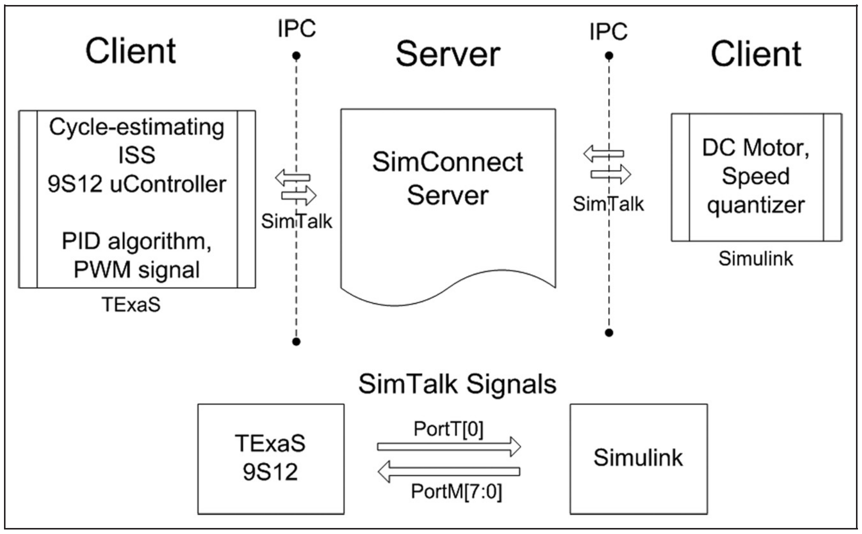

Figure 14. Two-simulator configuration.

The sources ‘‘Back-EMF’’ and ‘‘Torque’’ are Ngspice Current-controlled Voltage Sources (CCVS) driven from the current sensed in 0 V DC sources ‘‘Current_Sense’’ and ‘‘Speed_Sense.’’ The mechanical components are modeled by their model-equivalent electrical components: an inductance for the rotor inertia, resistance for the viscous drag force, and voltage source for the torque. The Ngspice deck for the DC motor subcircuit and driver is given in Figure 18, with parameters from Figure 3 as in the Simulink model. Spice-3 compliant subcircuit models for the IRF540N, 1N4004, and Q2N2222 semiconductors are downloaded from ON Semiconductor Corp[37] and International Rectifier Corp.[38]

Two SimTalk sockets in the Ngspice deck provide the co-simulation input/output (I/O), an input to capture the 9S12 PortT[0] PWM wave output, and one output to sample the circuit motor torque for consumption by Simulink. The Xspice and Simtalk components are specified in Figure 19.

Finally, the Simulink submodel for the DC motor replicates the mechanical-only components in Figure 20 for speed sensing in Figure 21, and reports back to TExaS through SimTalk signal ‘‘PortM[7:0].’’

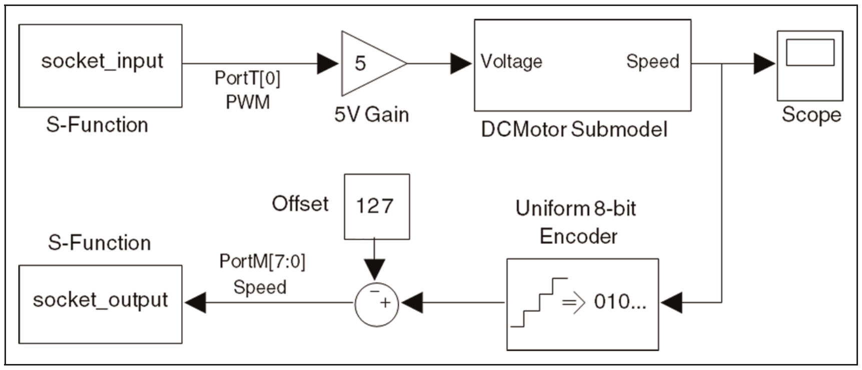

Figure 15. Simulink direct current motor model with SimTalk input/output interface.

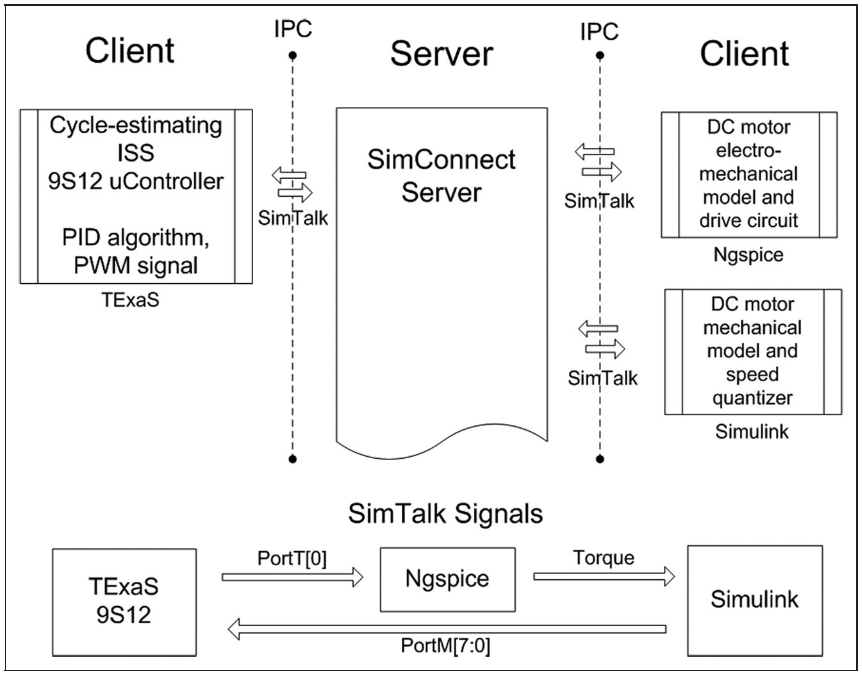
Client
IPC
Server
IPC
Client
Cycle-estimating ISS 9S12 uController
PID algorithm, PWM signal
TExaS
SimTalk
SimConnect Server
SimTalk
DC motor electro-mechanical model and drive circuit
Ngspice
SimTalk
DC motor mechanical model and speed quantizer
Simulink
SimTalk Signals
TExaS 9S12
PortT[0]
Ngspice
Torque
Simulink
PortM[7:0]

Figure 16. Three-simulator configuration.

5.6 Results: three-simulator digital software PID with PWM actuator in TExaS, electrical driver and DC motor model in Ngspice, and Simulink DC motor mechanical model

The three-simulator configuration was executed over IE resolutions 10, 50, and 100 µs to measure speed versus accuracy against the two-simulator case at 100 µs IE resolution in Figure 13. Speed of the three-simulator execution is affected by the IE event rate (SimConnect traffic and time points) and internal simulator rates. In Figure 22, the three-simulator case rise time is plotted against the two-simulator case baseline.

As can be seen in Figure 22, the motor speed output profile as the electrical driver realism is added in Spice in the three-simulator case agrees with the two-simulator case output to within 10% error of measurement. The difference in output speed at a time point in Figure 22 for the three-simulator cases against the two-simulator cases is due to the realism added in the electrical driver, where a voltage divider is created between the motor coil R and the IRF540N MOSFET R_{on} resistance, to a voltage divider ratio of 1/1.077. This results in the motor coil not seeing a full 5 V power when the MOSFET is on, but 1/1.077 less, where in the two-simulator case, the electrical driver is ideal and created through a Simulink 5 V gain block. Significant in Figure 22 is that the motor model profile agrees when modeled in two completely different simulators (Ngspice and Simulink), and the PID controlled speed output agrees in regard to rise time and steady state.

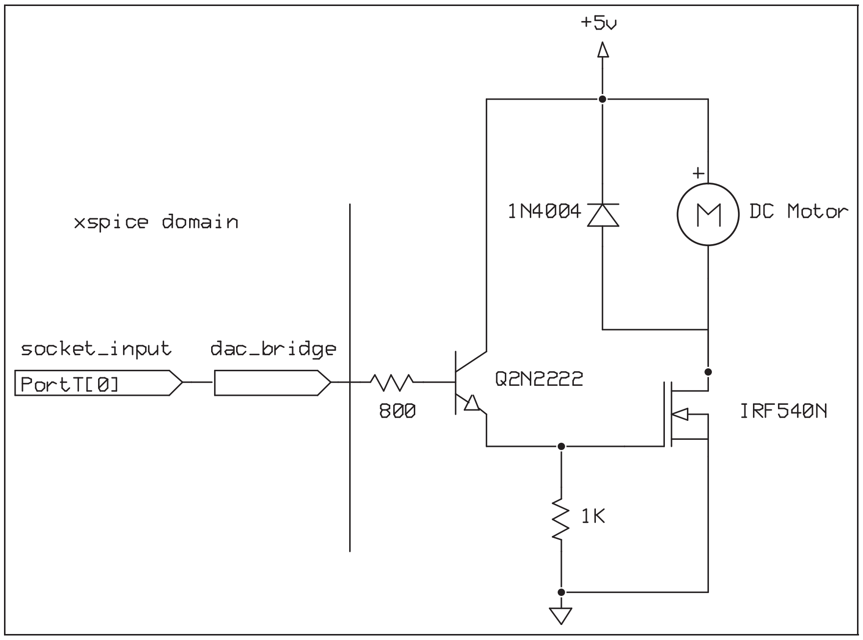

Figure 17. Ngspice model for motor driver circuit.

```
V_Power 1 0 DC +5V

*Electrical motor components
D0 5 1 1n4004rl
R0 1 1 1
L0 2 3 .001
H_Back-EMF 3 4 v_Speed_Sense .1
V_Current_Sense1 4 5 dc 0V

* Mechanical motor components
H_Torque 10 0 V_Current_Sense .1
LJ 10 15 .01
Rb 15 16 .001
V_Speed_Sense 16 0 dc 0

* Power MOSFET
R1_pulldown 20 0 1000
X_IRF 5 20 0 irf540n

* IRF540N gate driver
R2 20 21 800
Q2 1 21 22  Q2N2222
```

Figure 18. Ngspice deck for motor and driver.

```
* SimTalk Inputs *
*
a4 100 socket_input_a
.model socket_input_a d_socket_input
+ (signal_name="portt0"
+               ip_address="SimConnectServer" port=8000)
*
a2 [100] [20] dac_bridge0
.model dac_bridge0 dac_bridge (out_low=0.0 out_high=5
+ out_undef=0
+               input_load=1.0e-12
+               t_rise=1.0e-8 t_fall=1.0e-8)

* SimTalk Outputs *
*
a5 [10] socket_output_a
.model socket_output_a a_socket_output
+ (signal_name="motor_torque"
+               ip_address="SimConnectServer" port=8000)
+               update_period=.00001
+               initial_value=0
+               initial_duration=.00001)
```

Figure 19. Ngspice SimTalk devices 10 μs interpolated event resolution.

Figure 22 also indicates that from 100 to 10 μs IE resolution, there is no significant difference in output profile.

In Figure 23, however, as the IE resolution is decreased for simulation speed, the measured rotor speed output begins to depart from the baseline result.

The departure from the control case in Figure 23 can be attributed to the coarse resolution in the IE period of measuring signals varying at 1 kHz in the PortT[0] PWM wave signal. In the 100 ms IE case, for example, the IE is

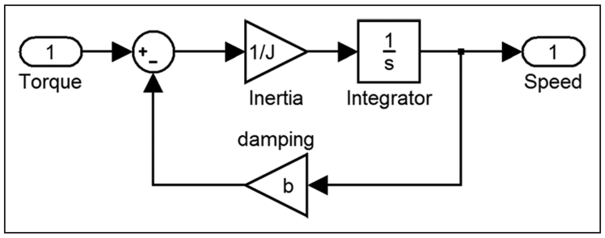

Figure 20. Simulink mechanical-only direct current (DC) motor submodel.

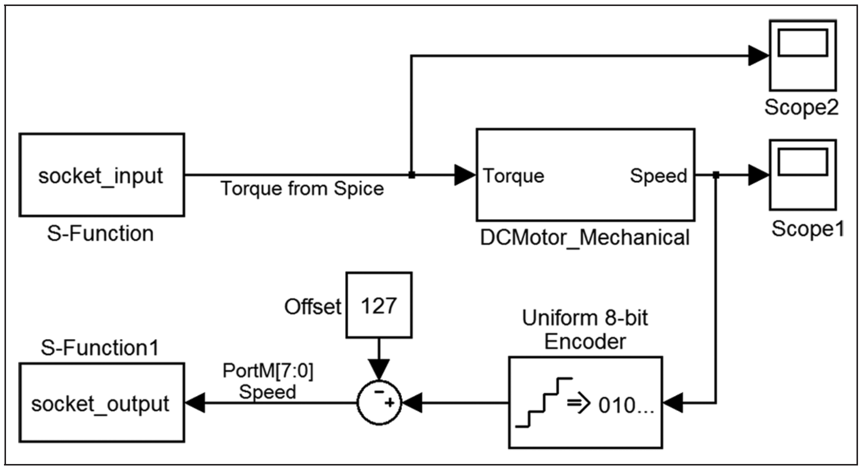

Figure 21. Simulink co-simulation model with mechanical-only direct current (DC) motor model.

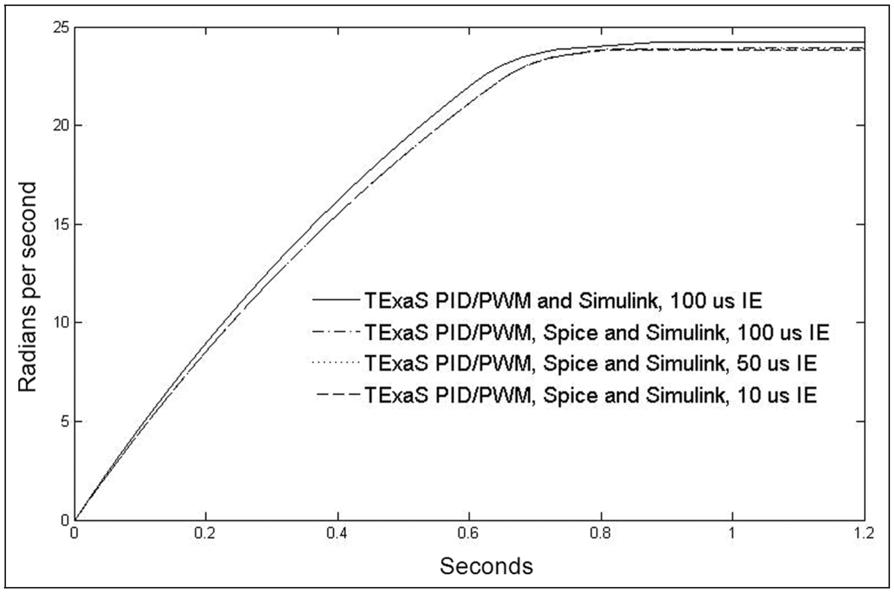

Figure 22. Model output speed versus time in two- and three-simulator configurations.

not able to pick up the transition of the PWM wave to high until 100 ms into the simulation as the initial condition IE is set to zero (and expires at 1 ms). If at 100 ms the signal happens to be sampled at zero, the consuming simulator (Ngspice) will process that value until its next expiration time at 100 ms later. This results in the Ngspice motor model not getting a power value in Figure 23 in the 100 ms case until 250 ms into the simulation. As the PID algorithm samples at 1 kHz, its PortM speed input is only reported every 100 ms, resulting in not recalculating a new speed value until every 100 PID loops, and any PWM updates only being sampled every 100 ms. As a result, although the simulation runs faster, the accuracy of measured output begins to decrease.

5.7 Discussion: speed versus accuracy in the three-simulator case

Figures 23 and 24 indicate that the IE resolution for a co-simulation should be scrutinized against the bandwidth of the signals sampled by the IEs. The 100 ms IE resolution in Figure 23 curve five does not meet the period of the PWM wave in the simulation at 1 ms. However, a ceiling of 100 μs in the simulation meets the baseline of the two-simulator case, and increasing the IE resolution does not appreciably change the accuracy of the observed rotor output speed.

Another point of comparison is to look at the controller effort in the three-simulator case against the truth condition in the one-simulator classical PID case. This is a

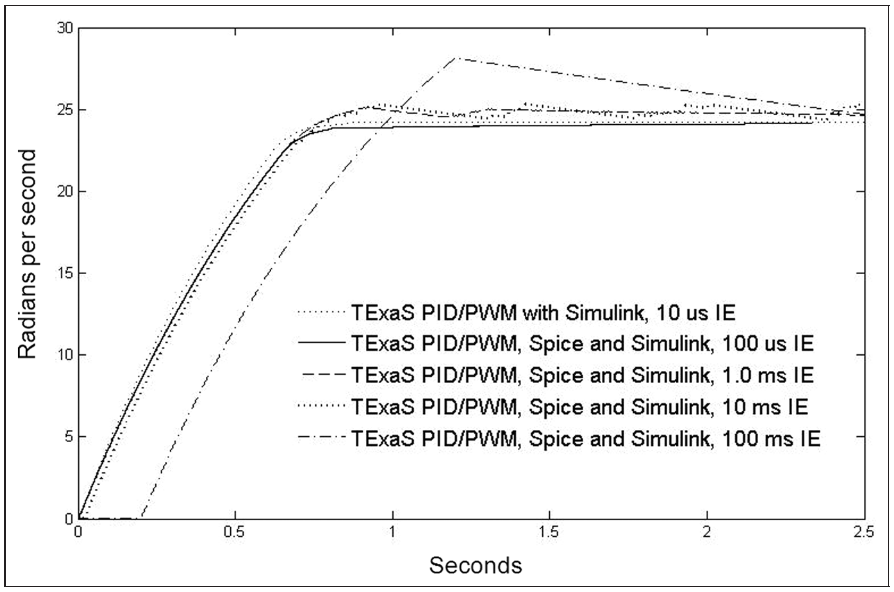

Figure 23. Variation in model rotor output speed versus time as a function of interpolated event (IE) resolution.

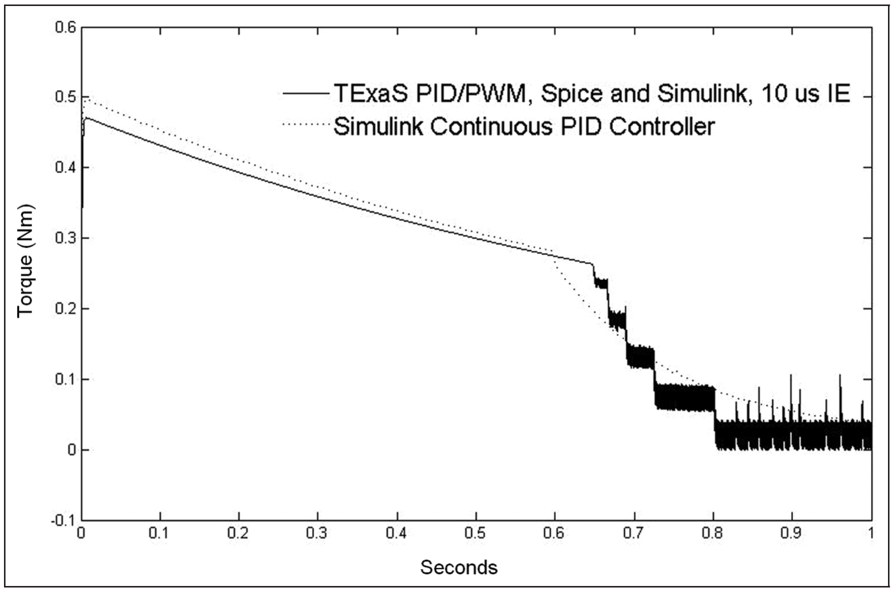

Figure 24. Discrete versus continuous model controller effort: applied motor torque versus time in one-simulator and three-simulator cases.

signal with more variation over time than the controller rotor speed. The plot of the applied motor torque in two cases is given in Figure 24.

The applied motor torque is the motor current through the armature times the motor K_t coefficient (0.1). With a coil resistance of 1 Ohm, this plot also tracks the curve of the applied terminal current. The difference between the applied torque through 0.5 seconds in Figure 24 is due to the electrical realism in the three-simulator case of the motor coil resistance and IRF540N on resistance voltage divider, reducing the applied torque by a factor of 1/1.077. Through 0.5 seconds the PWM wave is 100% in the discrete case, and at maximum value (5 V) in the continuous case. The step nature of the discrete output is due to the quantized PID speed and finite sample rate of the PID algorithm in software.

In Figure 25, the controller effort is plotted against the continuous case for 100 μs and 10 μs IE resolution and zoomed to 40 ms. The offset from the continuous case is again due to the voltage divider electrical realism also seen

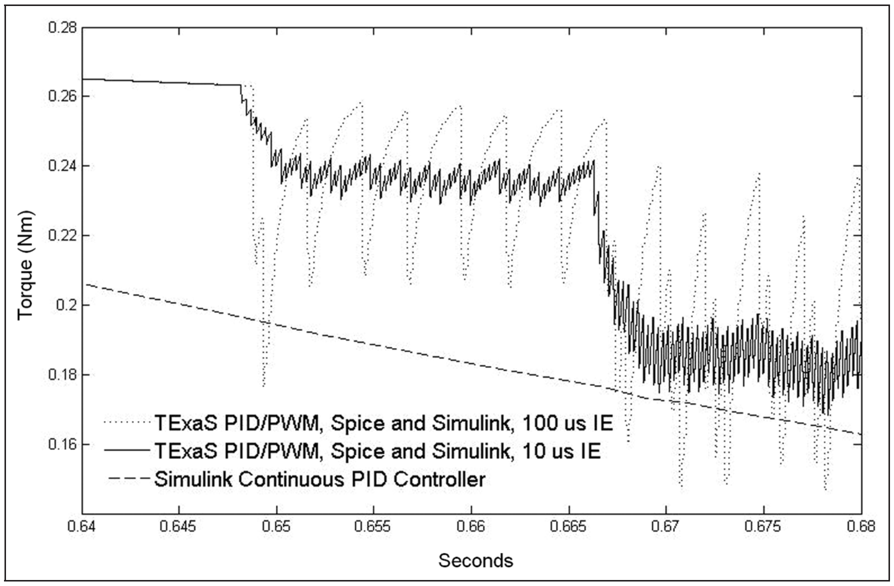

Figure 25. Discrete versus continuous model controller effort: applied motor torque versus time in one-simulator and three-simulator cases.

A	B	C	D	E	F	G	H	I
1. S,T,I	10 μs	4	200	20	11.8	5.0	6.39	1
2. S,T,I	100 μs	4	20	20	11.8	5.0	3:07	1
3. S,T,I	10 μs	4	200	20	11.8	5.0	30:12	3
4. S,T,I	100 μs	4	20	20	11.8	5.0	3:20	3
5. S,N,T,I	10 μs	6	81.6	5.44	3.2	1.36	3:45	2
6. S,N,T,I	100 μs	6	12	8	4.7	2.0	3:36	2
7. S,N,T,I	1 ms	6	1.8	12	7.1	3.0	3:12	2
8. S,N,T,I	10 ms	6	0.3	20	11.8	5.0	4:09	2
9. S,N,T,I	100 ms	6	0.03	20	11.8	5.0	3:07	2

Column Legend

A	Trial No. and Simulators: Simulink(S), Ngspice(N), TExaS(T), SimConnect Server(I)
B	Interpolated event (IE) resolution
C	Number of instantiated SimTalk connectors
D	Number of SimTalk messages (times 10^4)
E	Number of 9S12 cycles (times 10^6)
F	Number of 9S12 instructions (times 10^6, rounded 10^5)
G	Simulated Time (seconds)
H	Simulation Execution Time (minutes: seconds)
I	Number of host machines

Figure 26. Simulation times, configurations, and traffic.

in Figure 22. The plot, however, shows the significance of an IE resolution matching the bandwidth of changing signals shared between simulators. The spread effect in the 100 μs case is due to the time constant of the resistor-inductor (RL) motor circuit as the PWM wave is held constant over a 100 μs IE versus a 10 μs IE. The resolution does not affect the general curve or the PID rotor output, but resistor-inductor-capacitor (RLC) transient effects monitored in the Ngspice circuit will be more extreme. Guidelines for choosing digital controller sample rates are

Software Component	Lines of C	Build	Framework
SimConnect server	975	Linux 2.6.16, gcc 4.2.2	Custom source code
SimTalk TExaS connector	264	Microsoft Visual Studio 2010	TExaS source code
SimTalk Ngspice connector	290	Linux 2.6.16, gcc 4.2.2	Ngspice user-defined device
SimTalk Simulink connector	250	Microsoft Visual Studio 2010	MATLAB .mex level-1 file

Figure 27. Software factors.

given by Franklin et al.,[35] and it is suggested here that the same approach apply to IE resolutions for simulated control systems.

6. Execution times, counters, and software factors

Figure 26 summarizes the different execution times, configurations, and metrics for reported co-simulation configurations. This data emphasizes that seconds of CPS simulated time simulation may take minutes of wall-clock time, differentiating it from real-time HLA or real-time Distributed Interactive Simulation (DIS)–like[26] simulations. Simulations are compute bound in Figure 26 and limited by the execution time of the Simulink variable-step ode45 solver. They become communication bound in Trial 3 as the distribution increases across three machines, including a wireless link. In Trial 4, with the same configuration, a 10 × reduction in IE resolution (10 × less traffic) decreased the network latency closer to the compute bound of the single-machine case (Trial 2). In Trials 5–9, the SimConnect server and Ngpsice execute on one machine, and TExaS and Simulink execute on another machine for a wired local area network (LAN) and local host separation between the SimConnectServer and clients (all clients and the server could not execute on the same machine). By comparison, in Pfeifer and Gerstlauer,[18] configurations become communication bound only as the number of simulators approaches 128. There is no general rule for timing in this method, as latencies are a function of individual simulator speed, network topology, and IE resolution. Communication costs will always increase with IE resolution, however, since there is more IE token communication per time advancement.

Figure 27 describes the software plugins for each simulator created to support the SimTalk protocol, and the factors for the SimConnect server backplane. The SimConnect server is from-scratch code supporting simulator connection requests, signal FIFOs, and publish/subscribe token forwarding, written in the C language on a GNU/Linux 2.6.16 kernel machine compiled with the GCC 4.2.2 compiler. The TExaS SimTalk connector is written in C and Microsoft Visual Studio 2010 to compile with the original TExaS simulator code written in the same environment. The Ngspice SimTalk connector is developed in the Ngspice user-defined device framework[20] on the same machine as the SimConnect server, and the Simulink SimTalk connector is built in Microsoft Visual Studio 2010 as supported by the MATLAB/Simulink .mex level-1 file format. For a rough comparison, most open source HLA-RTI backplanes are more than 10k lines of source code, but complexity is difficult to assign by code size. Significant in Figure 27, however, is that less than 500 lines of code each were required for SimTalk connectors in different simulator software architectures.

7. Summary and conclusions

SimConnect and SimTalk in this paper enabled disrtributed, heterogeneous hardware/software co-simulation for three independent simulators – TExaS, Ngspice, and Simulink – for modeling of a PID/PWM control system, and was proved against a baseline condition of a single Simulink simulation of the controller PID response.

Signficant to the SimConnect/SimTalk architecture is that once a SimTalk plugin is written for one simulator, it can communicate through the SimConnect server to any other simulator supporting a SimTalk plugin, for combinatorial growth in the number of simulator configurations possible. This differs from two-simulator or ‘‘ad-hoc’’ approaches[11] written with specific simulator structures in mind. The scalability of the SimConnect/SimTalk approach was shown for the parallel speedup of an Ngspice circuit over 128 Ngspice simulators by Pfeifer and Gerstlauer.[18] SimConnect/SimTalk meets a design requirement of source-based debugging with the ability to pause the global simulation by breakpoints in the software-simulators by interrupting the KPN dataflow. In these experiments, when a breakpoint was inserted in TExaS, or the PID algorithm was single-stepped in the TExaS debugger, the Simulink and Ngspice interactive plots paused and advanced accordingly, without trace intrusion. This adds circuit-level inspection during the simulation as well as register-level inspection in software source debugging. The authors feel this is a significant benefit of the topology. To the authors' knowledge, this paper represents the first time TExaS, Ngspice, and Simulink were all coordinated together for a simulation. For future development, the range of SimConnect simulator support is being increased by adding new SimTalk connectors, with studies in speed-versus-accuracy tradeoffs and dynamic resolution.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

1. Sangiovanni-Vincentelli A. Quo Vadis, SLD? Reasoning about the trends and challenges of system level design. *Proc IEEE* 2007; 95: 467–506.
2. Lee EA. Cyber-physical systems: design challenges. *The University of California at Berkeley Center for Hybrid and Embedded Software Systems.* Technical Report No.UCB/EECS-2008-8, 2008.
3. National Science Foundation (NSF). Cyber-pysical systems. http://www.nsf.gov/pubs/2013/nsf13502/nsf13502.htm, 2012. Accessed December 27, 2012.
4. Klesh AT, Cutler JW and Atkins EM. Cyber-physical challenges for space systems. In: *the 2012 IEEE/ACM third international conference cyber-physical systems (ICCPS)*, 2012, pp.45–52.
5. Rajkumar R, Lee I, Sha L, et al. Cyber-physical systems: the next computing revolution. In: *the 2010 ACM/IEEE 47th design automation conference (DAC)*, 2010, pp.731–736.
6. Amory A, Moraes F, Oliveira L, et al. A heterogeneous and distributed cosimulation environment. In: *proceedings of the 15th symposium on integrated circuits and systems design*, 2002, pp.115–120.
7. Fujimoto RM. Parallel and distributed simulation. In: *proceedings of the 1995 winter simulation conference*, 1995, pp.118–125.
8. Fujimoto RM. Parallel discrete event simulation. In: *proceedings of the 1989 winter simulation conference*, 1989, pp.19–28.
9. Chandy KM and Misra J. Distributed simulation: a case study in design and verification of distributed programs. *IEEE Trans Software Eng* 1979; 5(5): 440–452.

10. Jefferson DR. Virtual time. *ACM Trans Program Lang Syst* 1985; 7(3): 404–425.
11. Schmerler S, Tanurhan Y and Muller-Glaser KD. A backplane approach for cosimulation in high-level system specification environments. In: *proceedings of the European design automation conference (EURO-DAC '95 with EURO-VHDL)*, 1995, pp.262–267.
12. Atef D, Salem A and Baraka H. An architecture of distributed cosimulation backplane. In: *the 42nd Midwest symposium on circuits and systems*, vol. 2, 1999, pp.855–858.
13. Sung W and Ha S. A hardware software cosimulation backplane with automatic interface generation. In: *proceedings of the Asia and South Pacific design automation conference (ASP-DAC)*, 1998, pp.177–182.
14. Gheorghe L, Bouchhima F, Nicolescu G, et al. Formal definitions of simulation interfaces in a continuous/discrete cosimulation tool. In: *proceedings of the seventeenth IEEE international workshop on rapid system prototyping*, 2006, pp.186–192.
15. Bouchhima F, Briere M, Nicolescu G, et al. A SystemC/Simulink cosimulation framework for continuous/discrete-events simulation. In: *proceedings of the 2006 IEEE international behavioral modeling and simulation workshop*, 2006, pp.1–6.
16. Pfeifer D and Valvano J. Kahn Process Networks applied to distributed heterogeneous HW/SW cosimulation. In: *the 2011 electronic system level synthesis conference (ECSI)*, 2011.
17. Kahn G. The semantics of a simple language for parallel programming. *Inf Process* 1974, pp.471–475.
18. Pfeifer D and Gerstlauer A. Expression-level parallelism for distributed spice circuit simulation. In: *the 15th IEEE/ACM international symposium on distributed simulation and real time applications (DS-RT)*, 2011.
19. Valvano J. *Embedded microcomputer systems: real time interfacing*. 3rd ed. Stamford, CT: Cengage Learning, 2011.
20. Nenzi P and Holger V. Ngspice user's manual (version 22). http://ngspice.sourceforge.net/docs.html, 2010. Accessed December 27, 2012.
21. The MathWorks Corp. www.mathworks.com (2012).
22. Cox FL, Kuhn WB, Li HW, et al. Xspice user's manual, Computer Science and Information Technology Laboratory, Georgia Tech Research Institute, 1992.
23. Narayanan R, Abbasi N, Zaki M, et al. On the simulation performance of contemporary AMS hardware description languages. In: *the 2008 ICM international conference on microelectronics*, 2008, pp.361–364.
24. IEEE Std 1516-2010. IEEE standard for modeling and simulation (m&s) high level architecture (HLA)– framework and rules, 2010, pp.1–38.
25. Dong H, Dong W and Ji Y. A HLA-based hierarchical architecture for the CTCS hardware-in-the-loop simulation system. In: *the 2nd IEEE international conference on computer science and information technology*, 2009, pp.86–91.
26. Fujimoto RM and Weatherly RM. Time management in the DoD high level architecture. In: *the proceedings of the 1996 10th workshop on parallel and distributed simulation*, 1996, pp.60–67.
27. de Mello BA and Wagner FR. A standardized co-simulation backbone. In: *the 11th international conference on very large scale integration of systems-on-chip*, 2001, pp.121–131.
28. Cadence Design Systems, Inc. www.cadence.com (2012).
29. Synopsys, Inc. www.synopsys.com (2012).
30. You Z and Mao-zhi W. Launch vehicle testing simulation system. In: *the 2011 international conference on computational and information sciences*, 2011, pp.921–924.
31. 'Nutaro J. *Building software for simulation: theory and algorithms, with applications in C++*. Hoboken, NJ: Wiley, 2010.
32. Zeigler B. *Theory of modeling and simulation.* 2nd ed. San Diego, CA: Academic Press, 2000.
33. Crowley P. A dynamic publish-subscribe network for distributed simulation. In: *the 22nd workshop on pricinples of advanced and distributed simulation*, 2008, p.150.
34. Lee EA and Sangiovanni-Vincentelli A. Comparing models of computation. In: *the IEEE/ACM international conference on computer-aided design digest of technical papers (ICCAD)*, 1996, pp.234–241.
35. Franklin G, Powell JD and Emami-Naeini A. *Feedback control of dynamic systems.* 4th ed. Saddle River, NJ: Prentice Hall, 2002.
36. Hughes A. *Electric motors and drives: fundamentals, types and application*. 3rd ed. Oxford, UK: Elsevier, 2006.
37. ON Semiconductor Corp. www.onsemi.com (2012).
38. International Rectifier Corp. www.irf.com (2012).

Author biographies

Dylan Pfeifer received his BA (music) degree from Yale University in 1997, and his BS (mathematics), and MS (electrical engineering) degrees from The University of Texas at Austin in 2004 and 2008. He is currently a PhD student in electrical and computer engineering at The University of Texas at Austin under Jonathan Valvano and Andreas Gerstlauer. He has also been employed as a design engineer since 2009 at Intel Corporation, with interests in real-time embedded systems and embedded software.

Jonathan W Valvano (M'83) received his BS degree in computer science and engineering and his MS degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1977. He received his PhD degree in medical engineering from the Harvard University/MIT Division of Health Sciences and Technology in 1981.He is currently a full professor in the Electrical and Computer Engineering Department at The University of Texas at Austin, performing research in the fields of embedded systems, simulation, and medical instrumentation.

Andreas Gerstlauer received MS and PhD degrees in information and computer science from the University of California, Irvine (UCI), in 1998 and 2004, respectively. Since 2008, he has been with The University of Texas at Austin, where he is currently an assistant professor in electrical and computer engineering. Prior to joining The University of Texas, he was an assistant researcher with the Center for Embedded Computer Systems at UCI. Dr. Gerstlauer serves on the program committee of major conferences such as DAC, DATE, and CODES+ISSS. His research interests include system-level design automation, system modeling, design languages and methodologies, and embedded hardware and software synthesis.