Co-Simulation of interconnected power electronics using Simulink-PSpice interface and components defined in C/C++ and SystemC

Bao Nguyen, Senior Pilot Engineer, MathWorks
Kishore Karnane, Product Management Director, Cadence

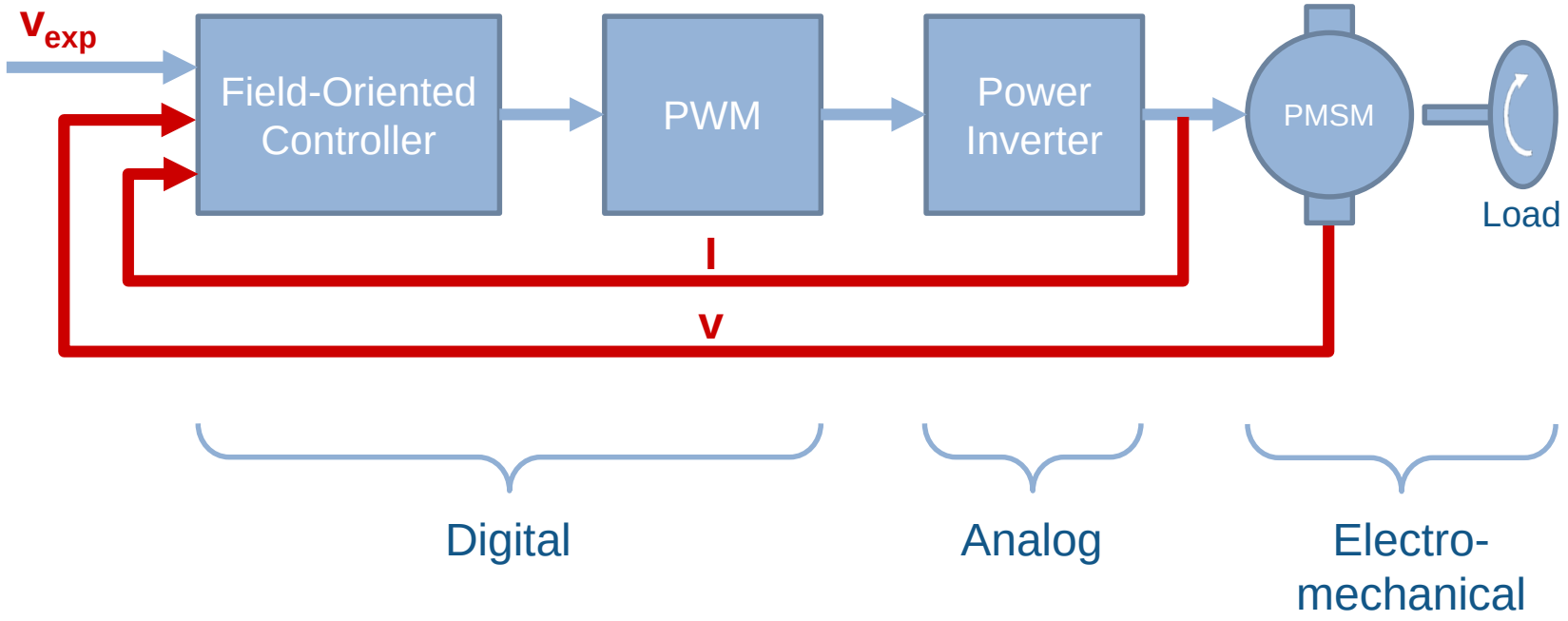
AGENDA

- Challenges in Analog/Mixed-Signal Design
- The SLPS Co-Simulation Interface
- The Device Modeling Interface (DMI) and Exporting Models from Simulink
  - Demo
- Conclusion

Analog/Mixed-Signal Design

Example: Field-oriented Control of a Permanent-Magnet-Synchrone-Machine

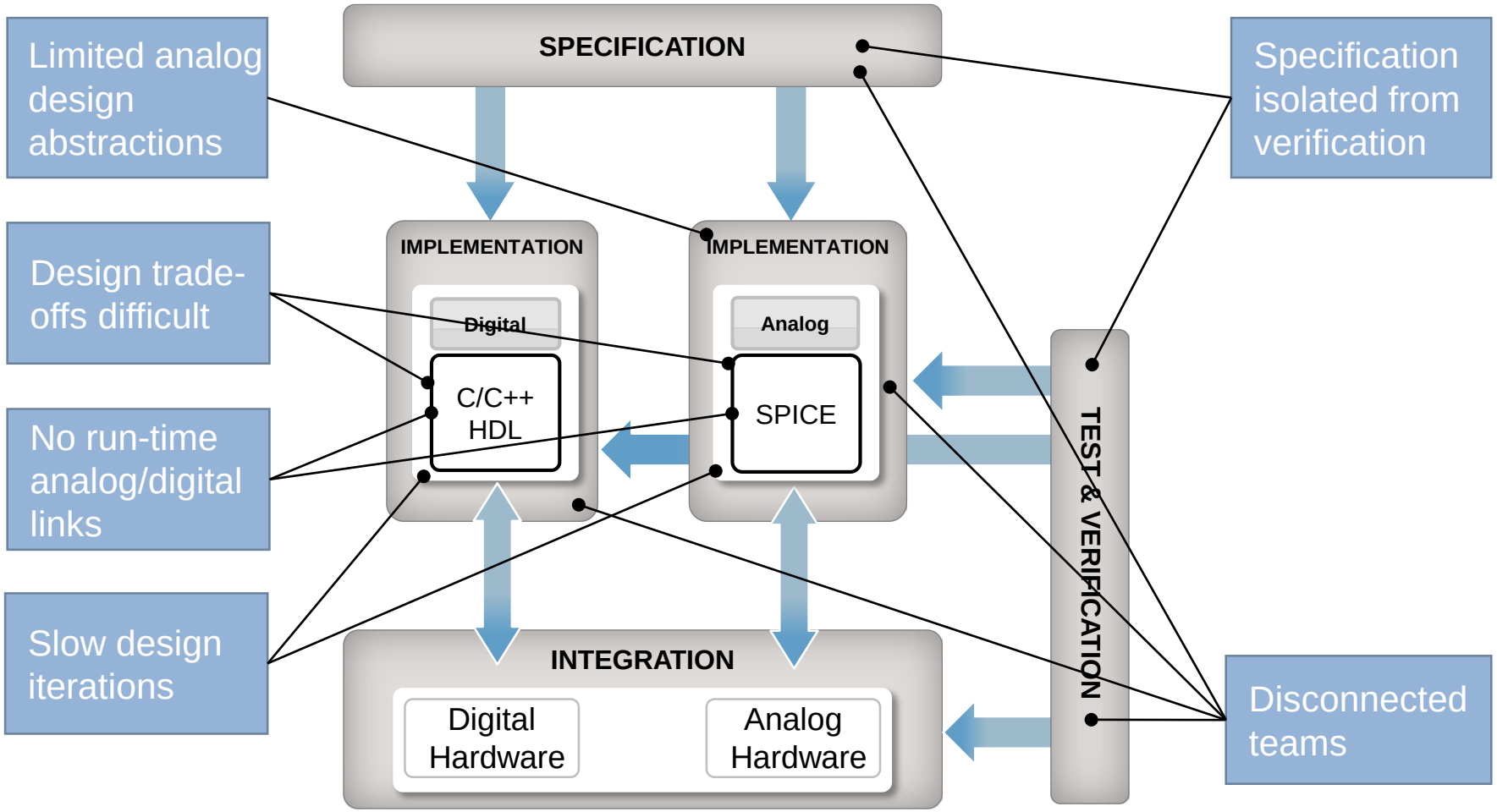
This control technique is common in motor drive systems for hybrid electric vehicles, manufacturing machinery, and industrial automation



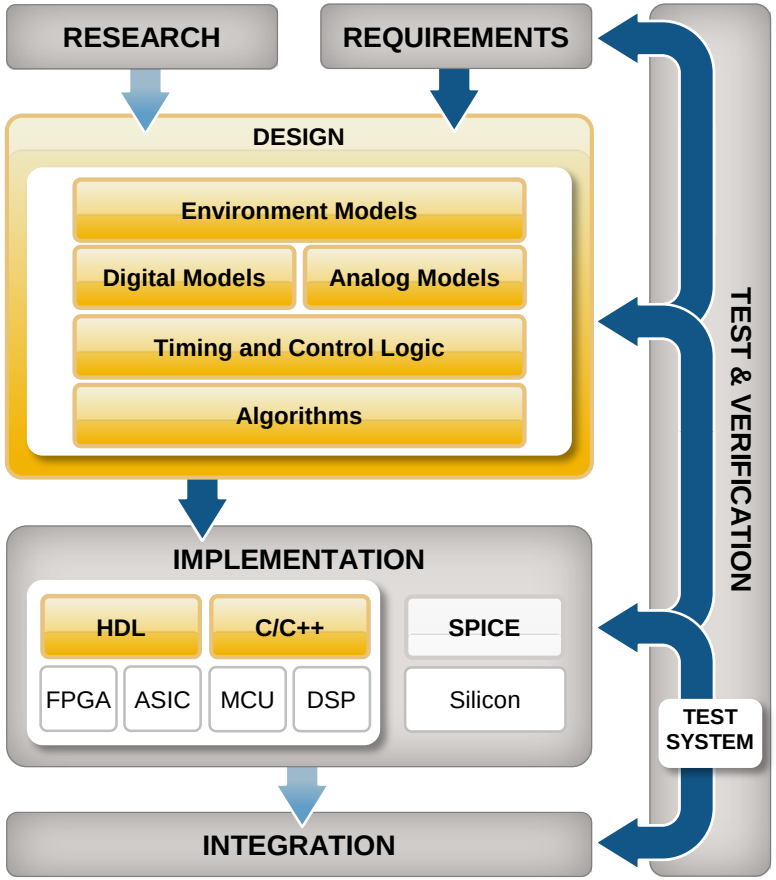
Target:
- MCU (SW)
- FPGA, ASIC (HW)

Target:
- Transitor-level Design

Challenges in Classical Mixed-Signal Design

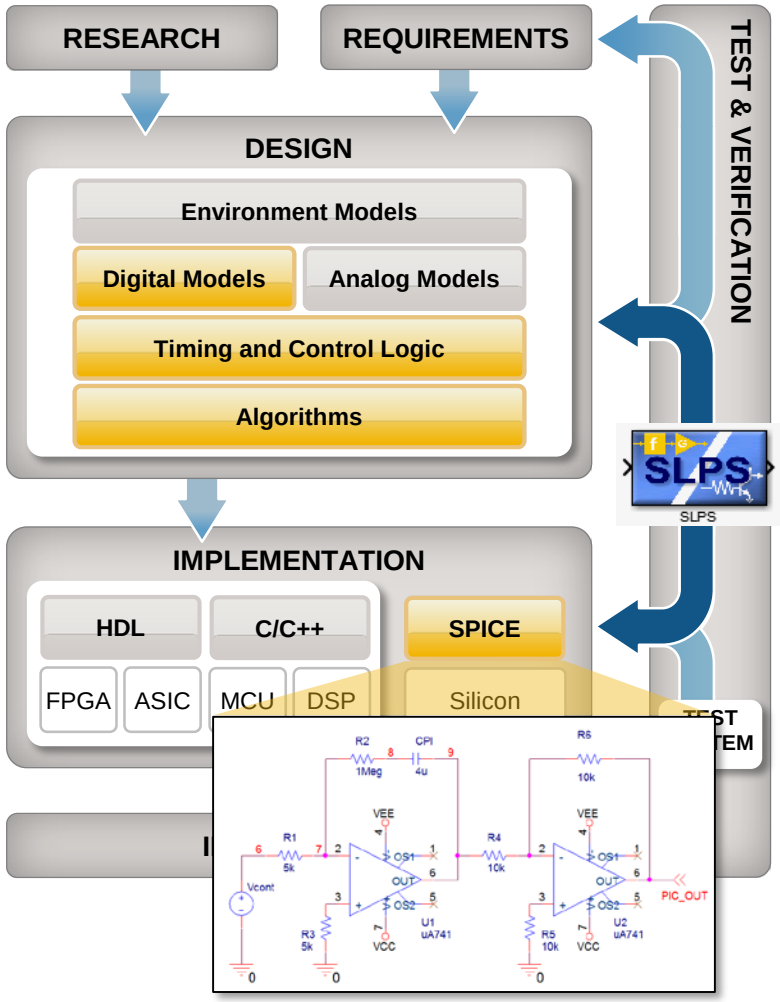


Model-Based Design



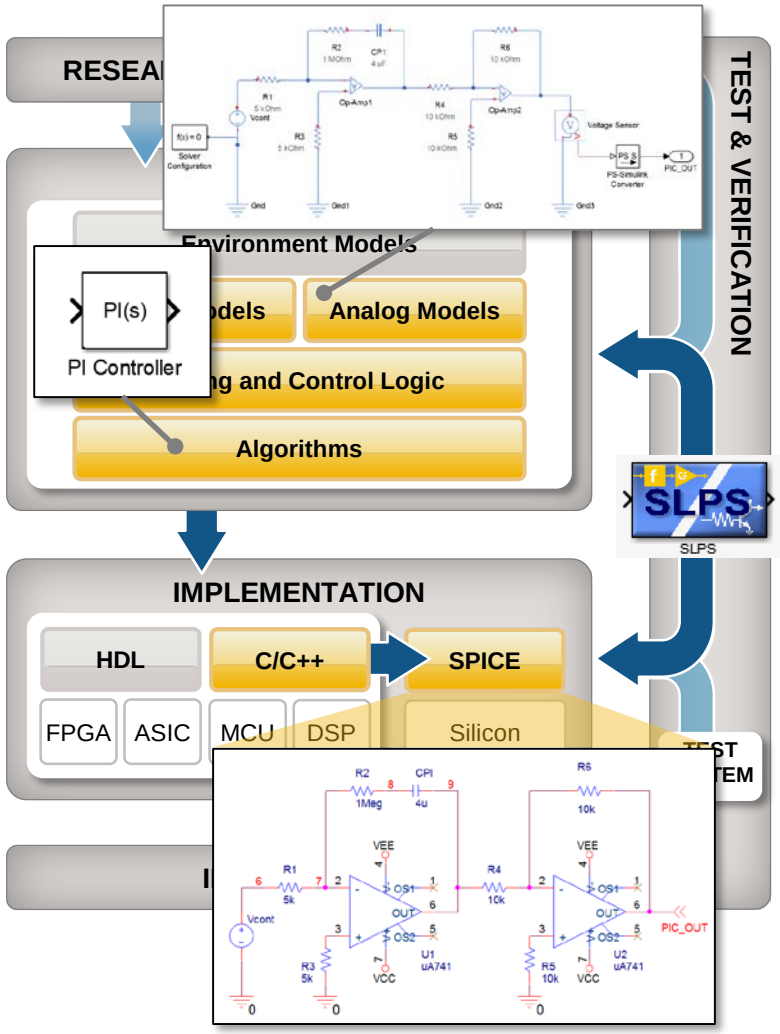
- „Executable Specification“
- Simulink as multi-domain simulation environment
  - Time-continuous and time-discrete (sampled)
  - Event-triggered
  - Mathematical and physical algorithm modeling
  - Robustness through environment modeling
- Automatic code generation (C/C++, HDL)
- Continuous Verification

Model-Based Design for Analog/Mixed-Signal



- Bottom-Up Workflow
  - Starting point: Transistor-level schematic
  - Needs
    - Input stimuli generation
    - Integration in surrounding multi-domain system
    - Analysis in time/frequency domain
  - Solution
    - Co-simulation with OrCAD PSpice using SLPS

Model-Based Design for Analog/Mixed-Signal

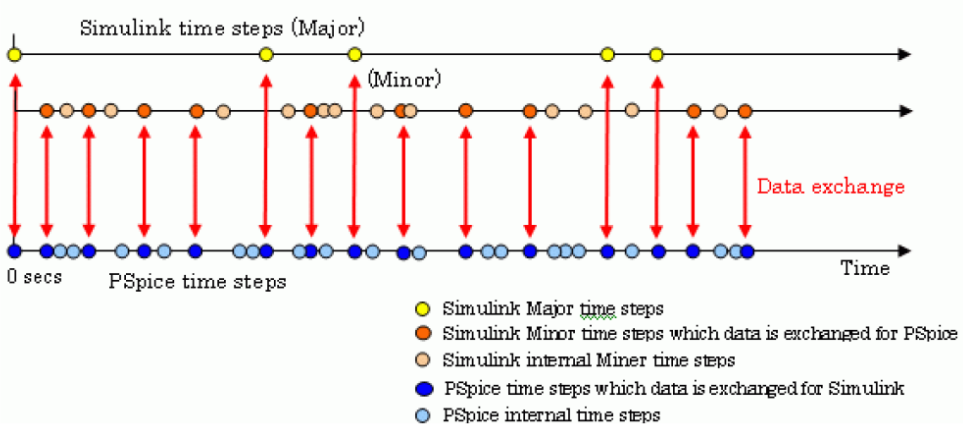


- Top-Down Workflow
  - Starting point:
    - Mathematical Model
    - Physical Model
  - Needs
    - Simulation speed (proof of concept)
    - Reuse of existing testbench
    - Sign-off Transistor-level simulation
  - Solution
    - Co-simulation with Simulink and PSpice using SLPS
    - Model integration through automatic C code generation and PSpice DMI

What is SLPS?

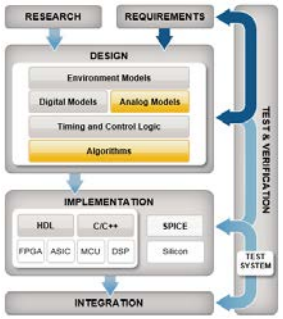
- SLPS = Simulink + PSpice Co-Simulation
  - Simulink
    - Multi-domain simulation environment for dynamic systems
    - Algorithm development and verification platform
  - PSpice:
    - SPICE-based simulator
    - Simulation of electrical and electronic circuits
    - Circuit design platform → Hardware

How does SLPS work?



- Simulink plays the master role
- The SLPS-block in Simulink builds the interface between both simulators
- Both simulators work with their own time-step-control algorithm
  - guarantees the optimal compromise out of simulation accuracy and performance.

Step 1: Algorithm Design and Verification



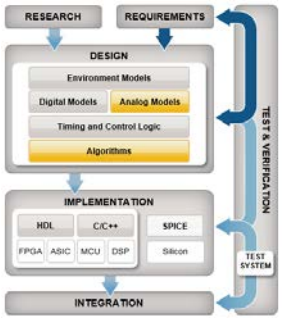
Time-discrete PWM Generator

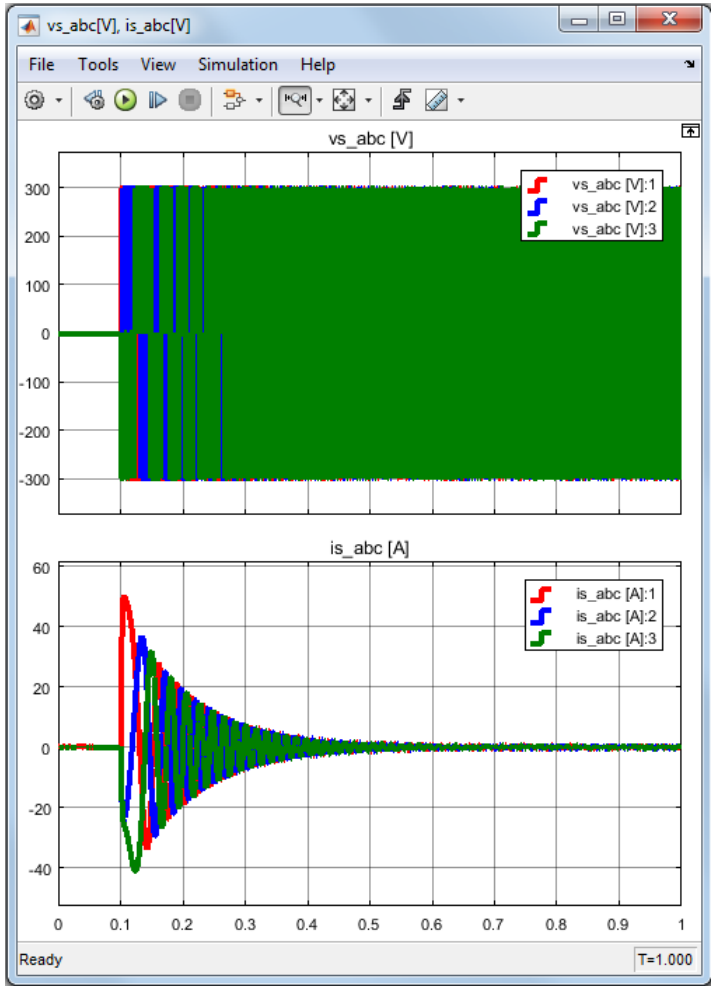
Physical Models of Mechanical Load Scenarios

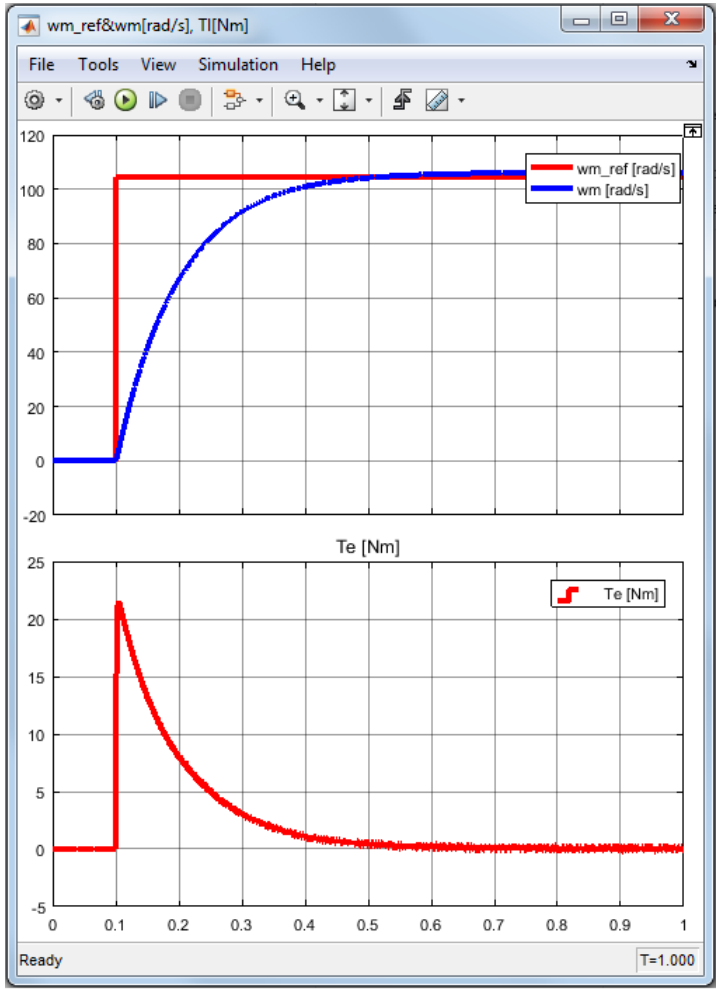
Time-continuous PI Controllers (speed, current)

Physical Models of Electrical Components

Step 1: Algorithm Design and Verification

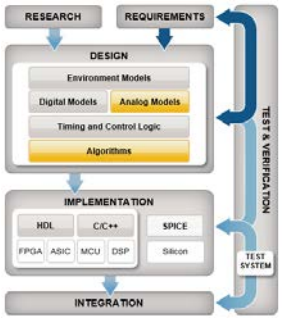


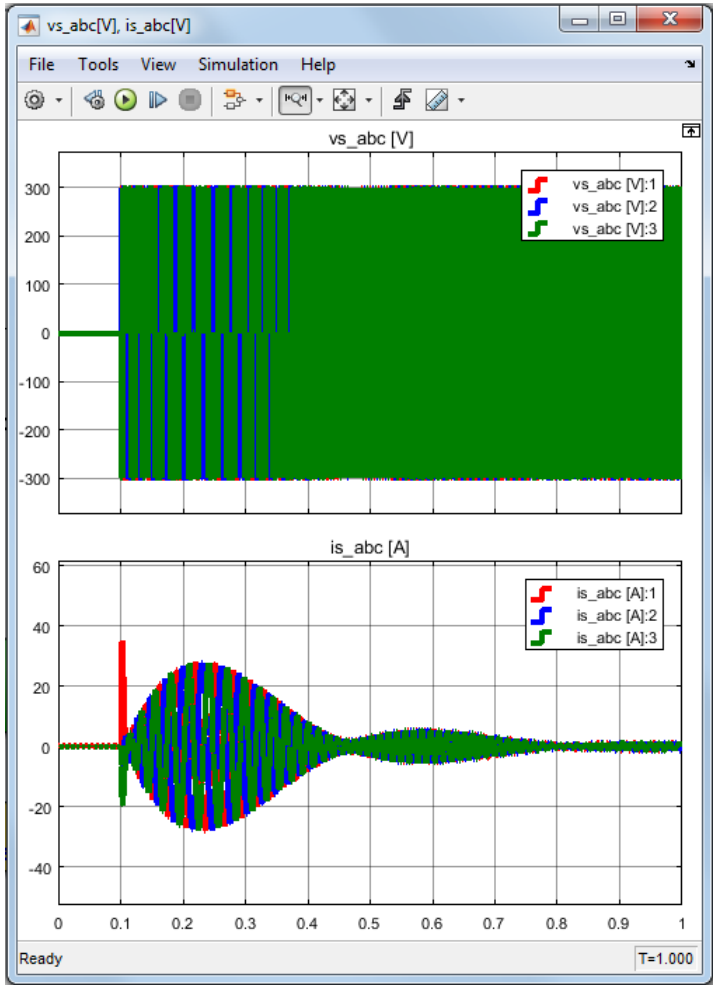


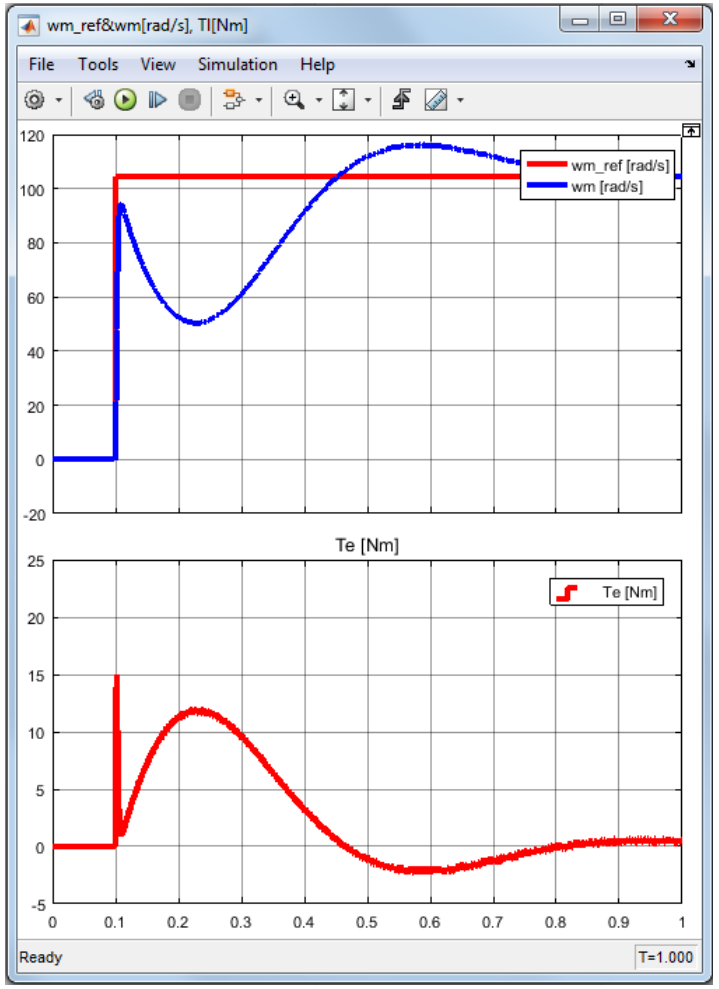


Load Scenario: with Load

Step 1: Algorithm Design and Verification

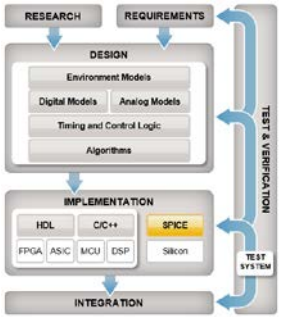


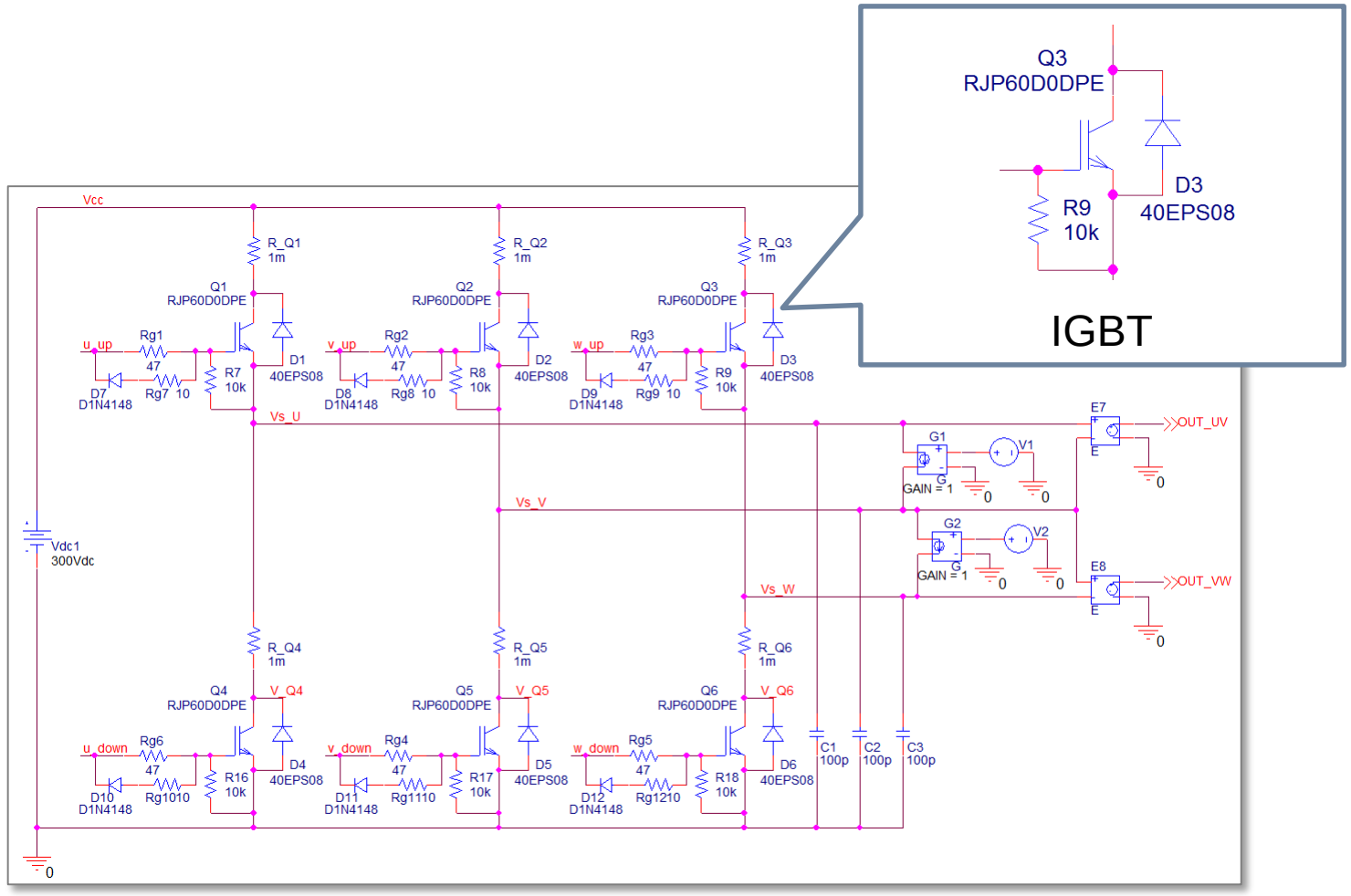




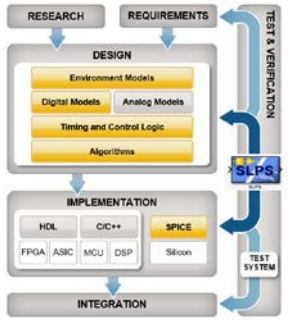
Load Scenario: with Load and Vibration

Step 2: Schematic Entry (PSpice)

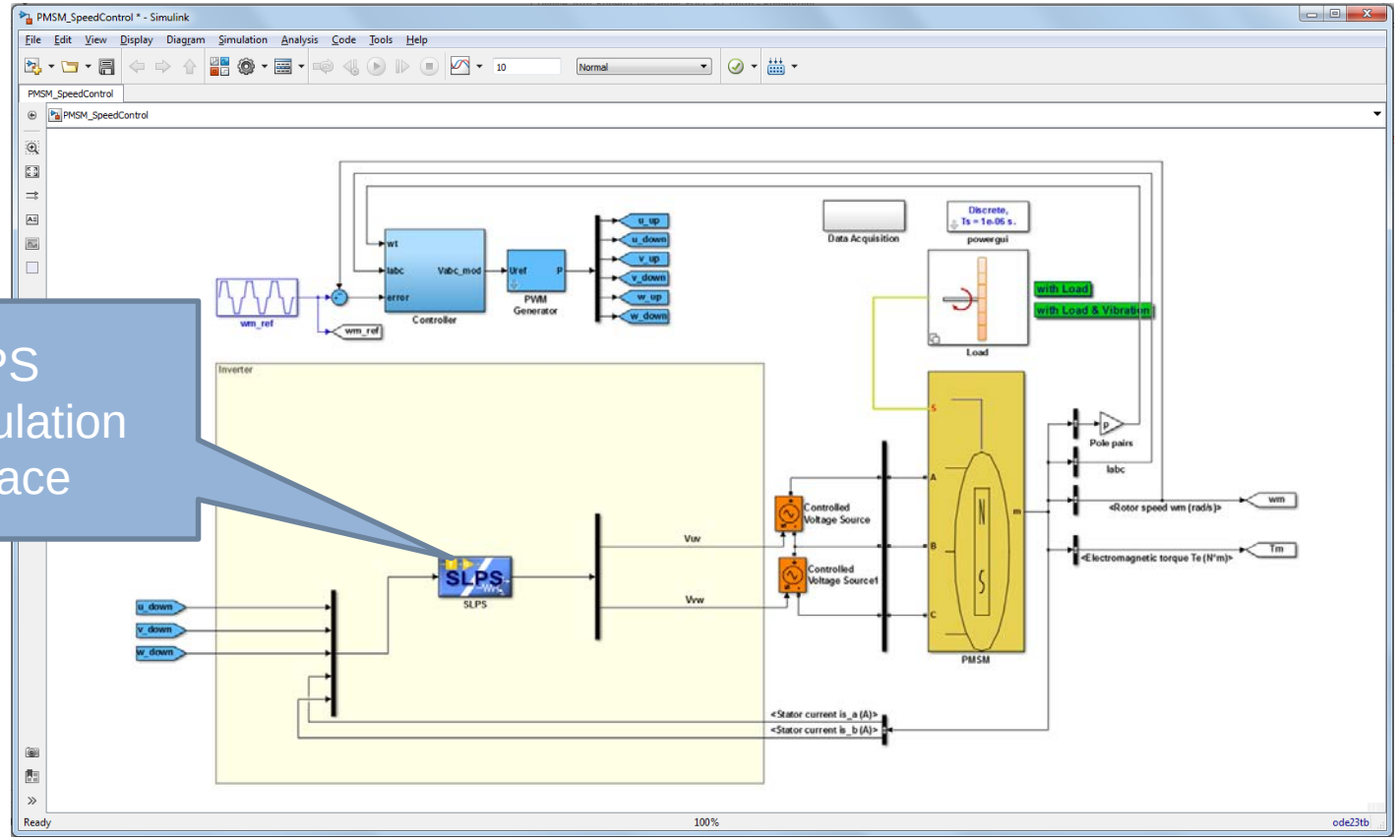




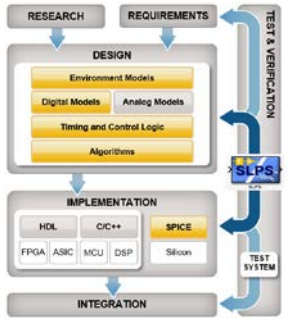
Step 3: Simulink/PSpice Co-Simulation (SLPS)



SLPS Co-Simulation Interface

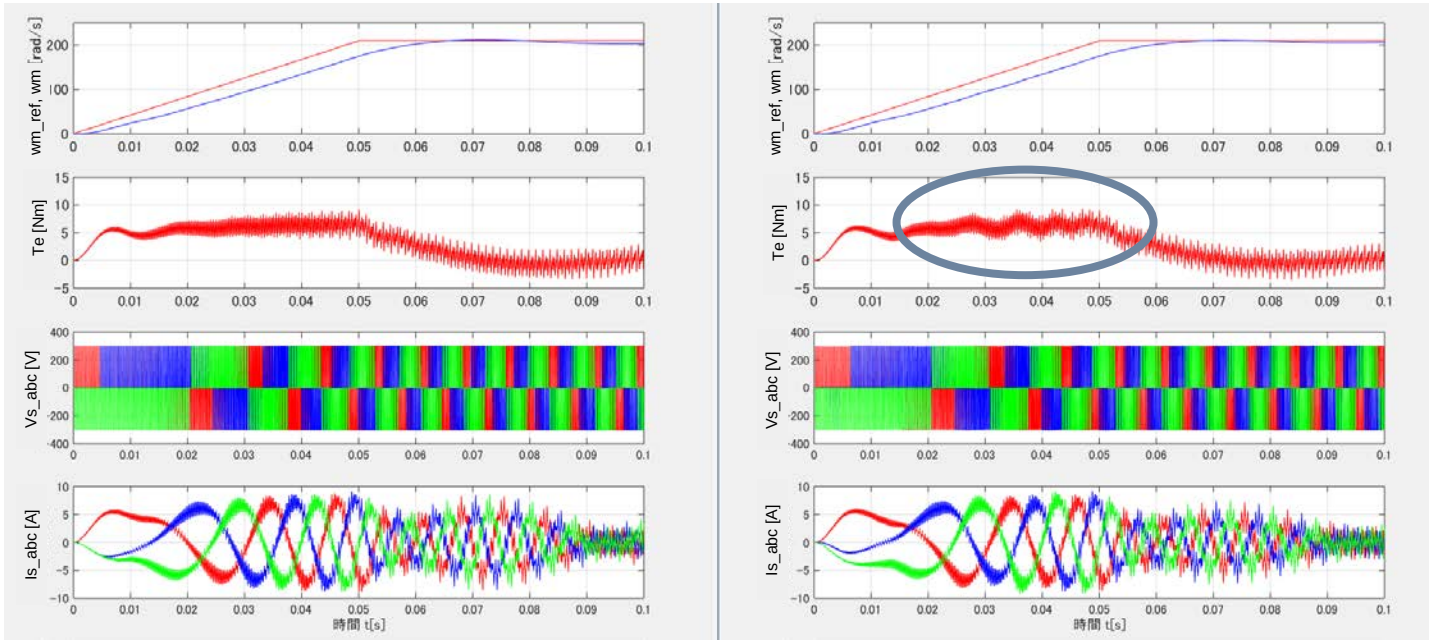


Step 3: Simulink/PSpice Co-Simulation (SLPS)

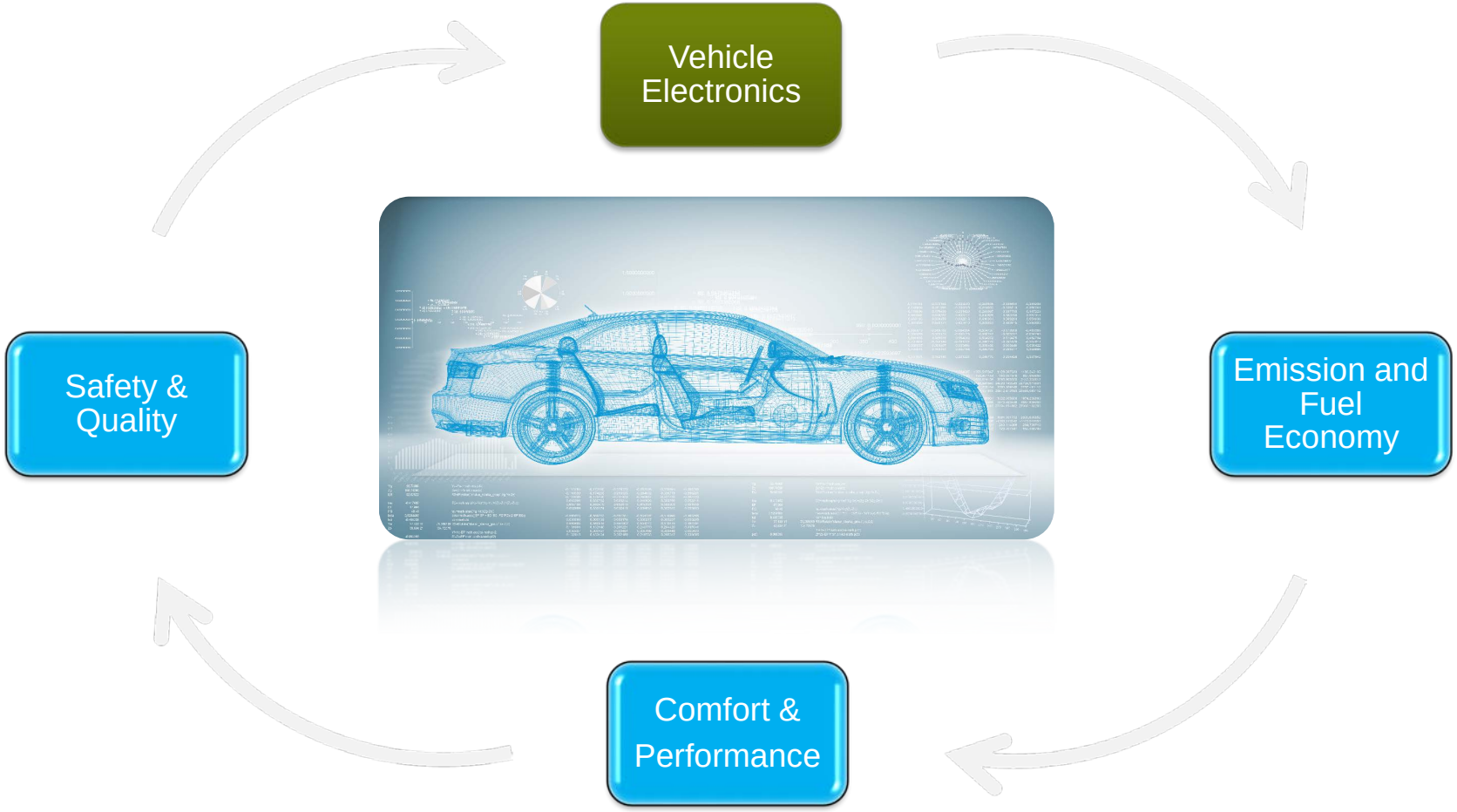


Simulink Simulation

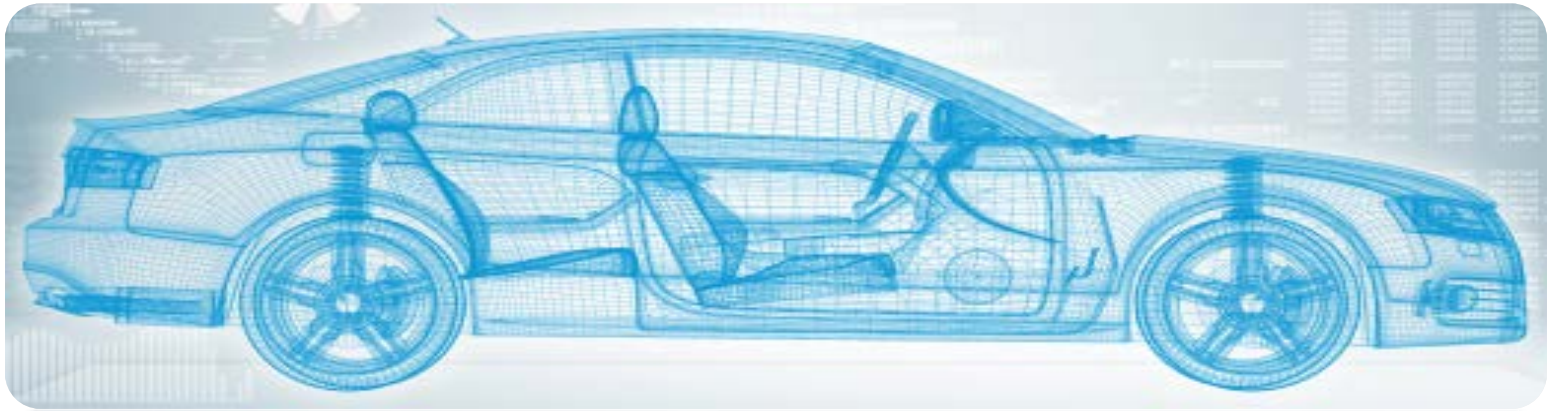
Simulink / PSpice Co-Simulation (SLPS)



Automotive Engineering Interdisciplinary Design Challenge

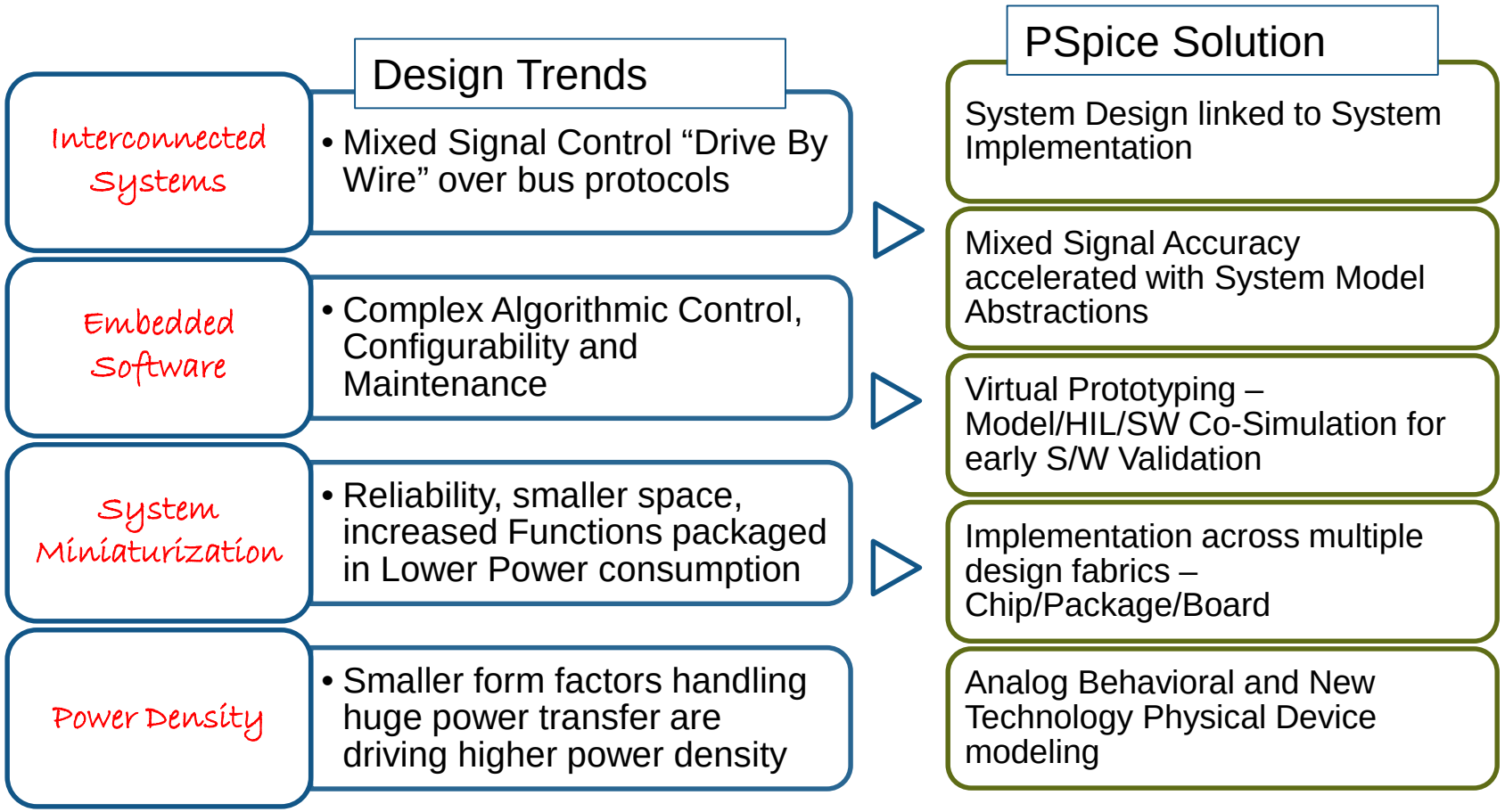


Automotive Engineering
Interdisciplinary Design Challenge

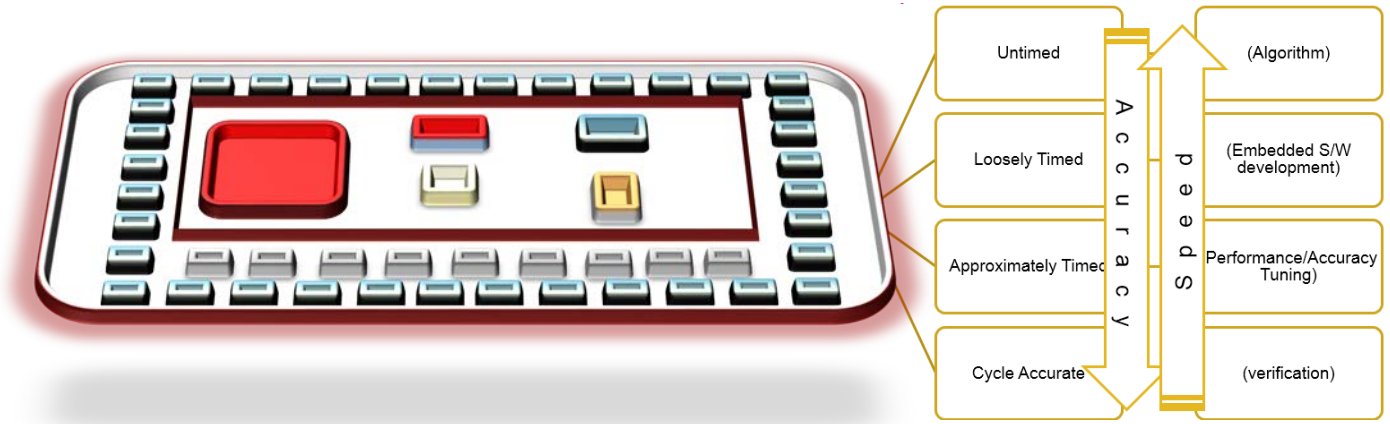


Explosion of Interconnected Electronic Systems with Embedded Software having some very challenging Power Density issues created by System miniatuarization for reliability, form & functions.

Automotive Engineering Interdisciplinary Design Challenge



PSpice complex device macro-model



Physical device compact model

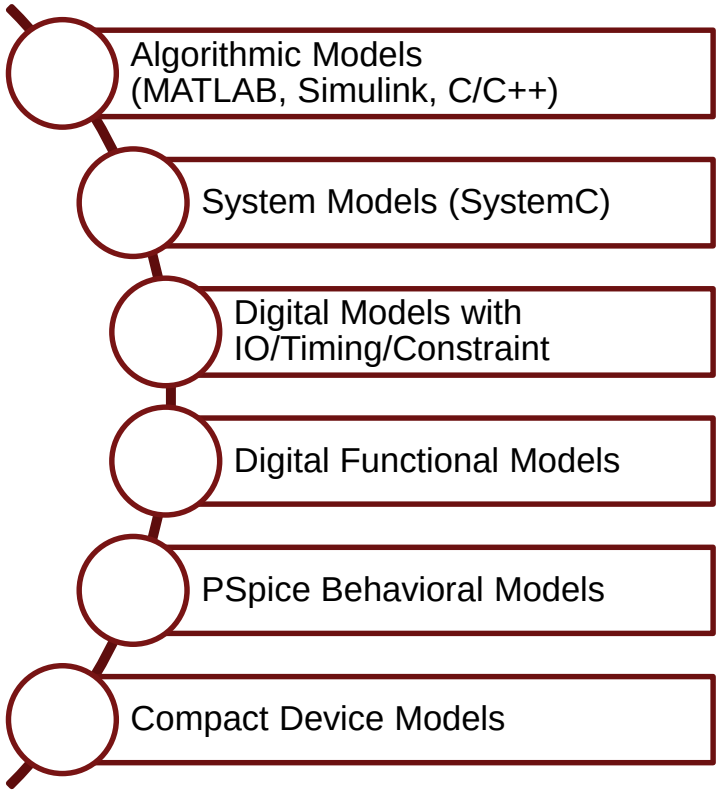
SystemC model supporting embedded S/W and different abstraction levels

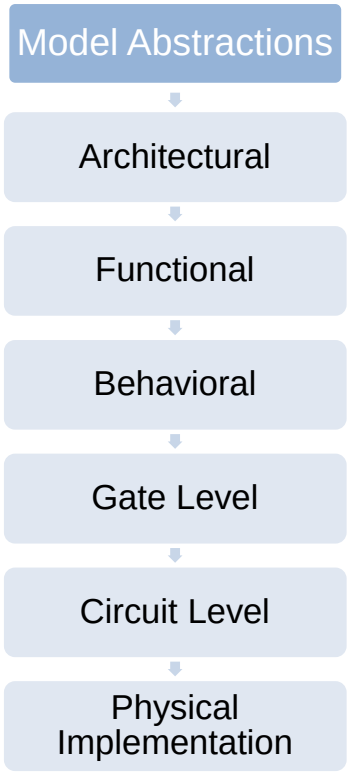
Analog behavioral

Digital C/C++ with embedded SW block

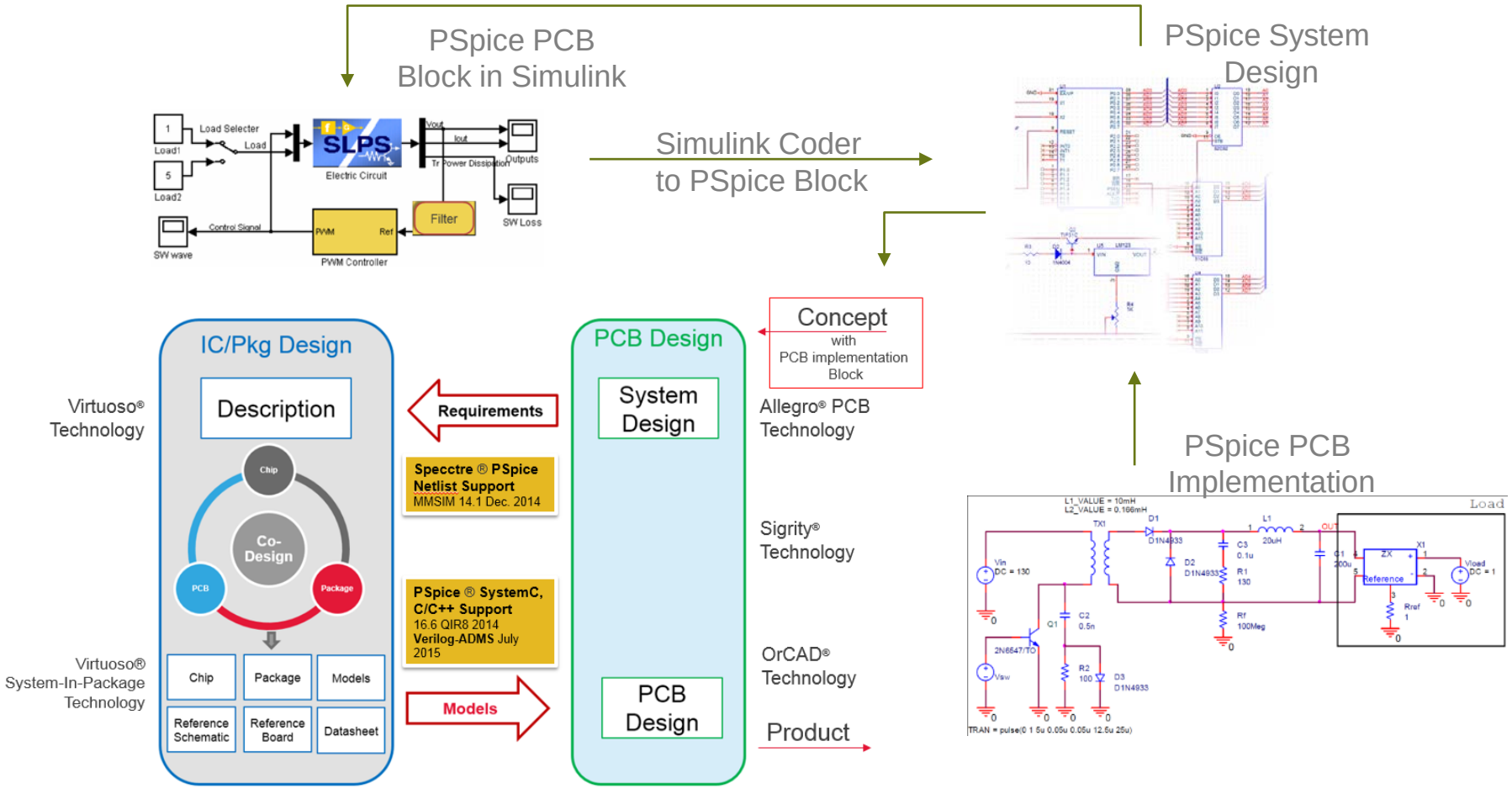
Every complex device on PCB - a system model embedded in mixed-signal device model

PSpice Models

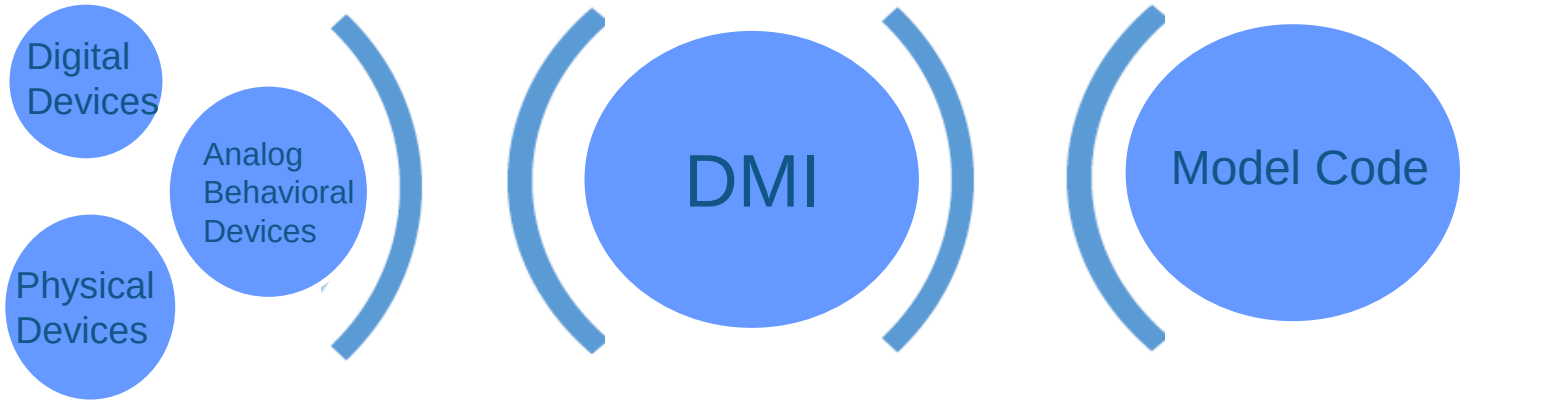




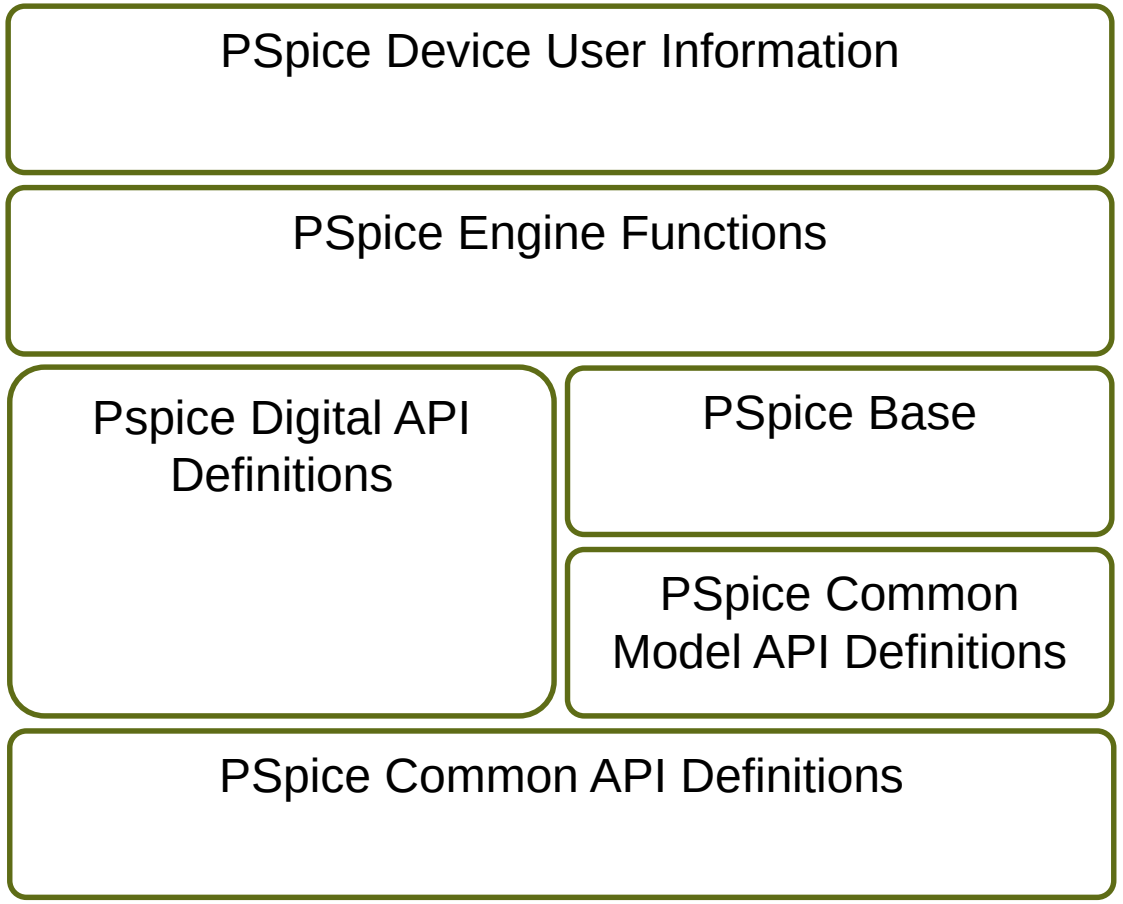
System Design Exploration to Implementation



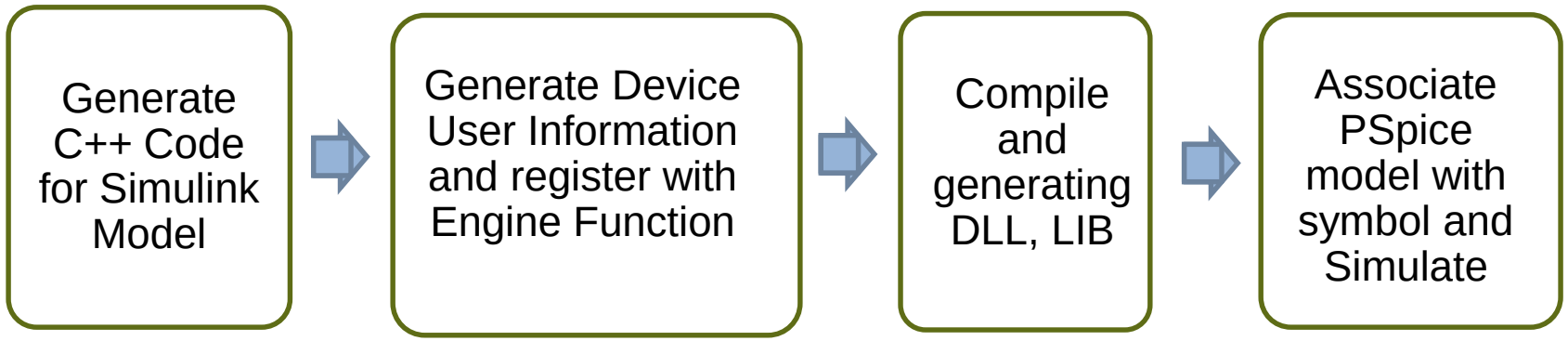
Device Modeling Interface



Device Modeling Interface Libraries

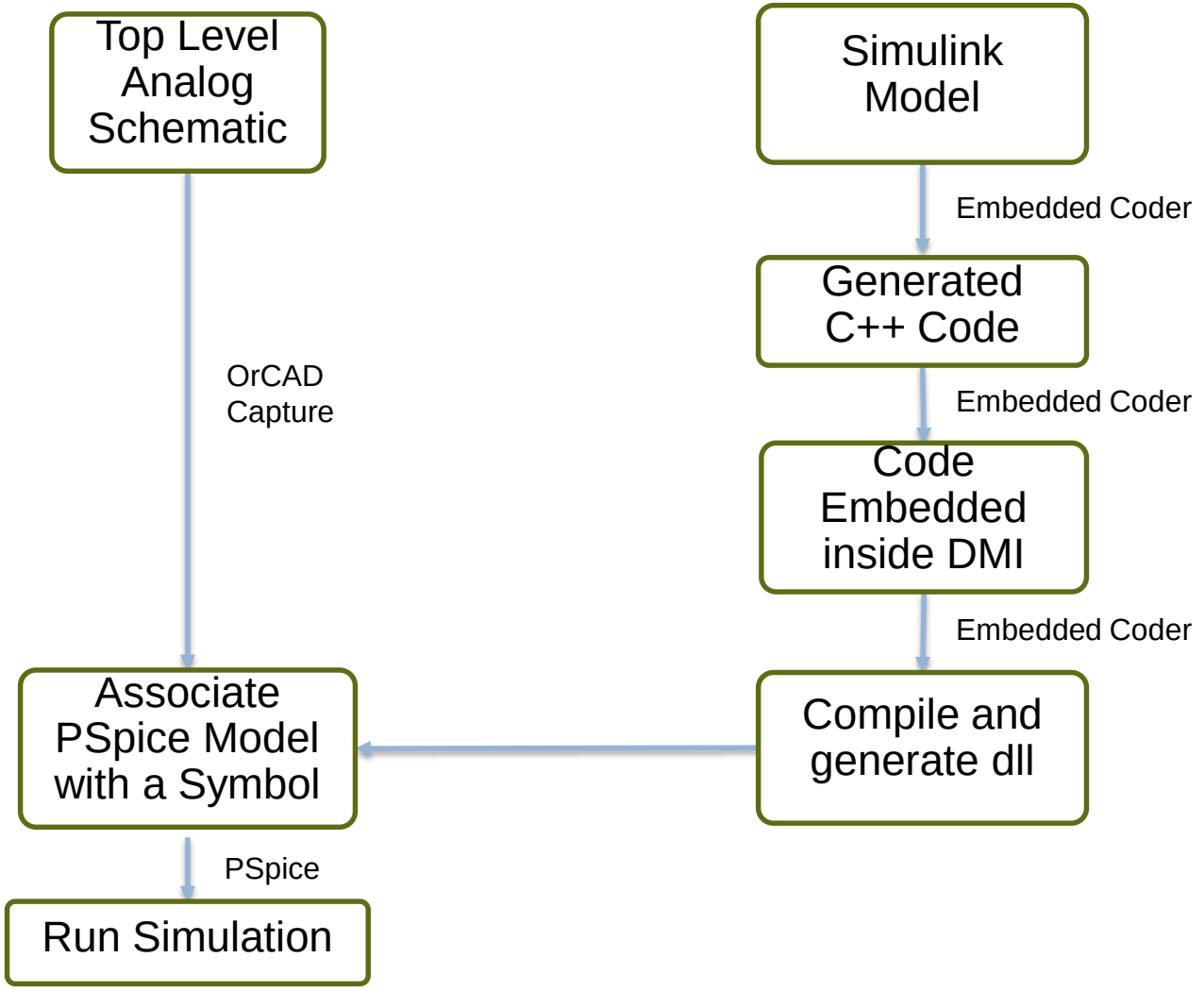


Device Modeling Interface – Embedded Coder Steps

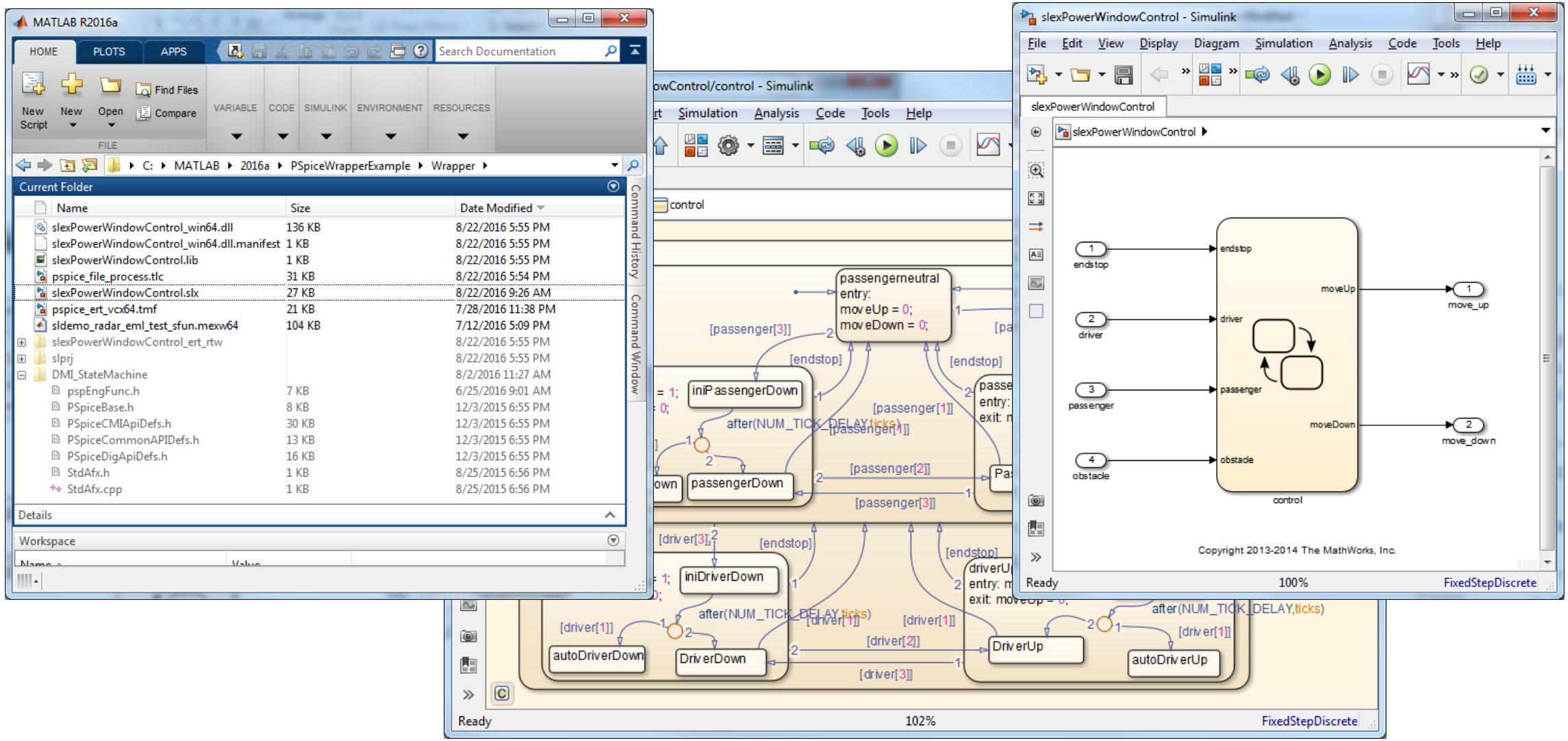


- Requires Embedded Coder license

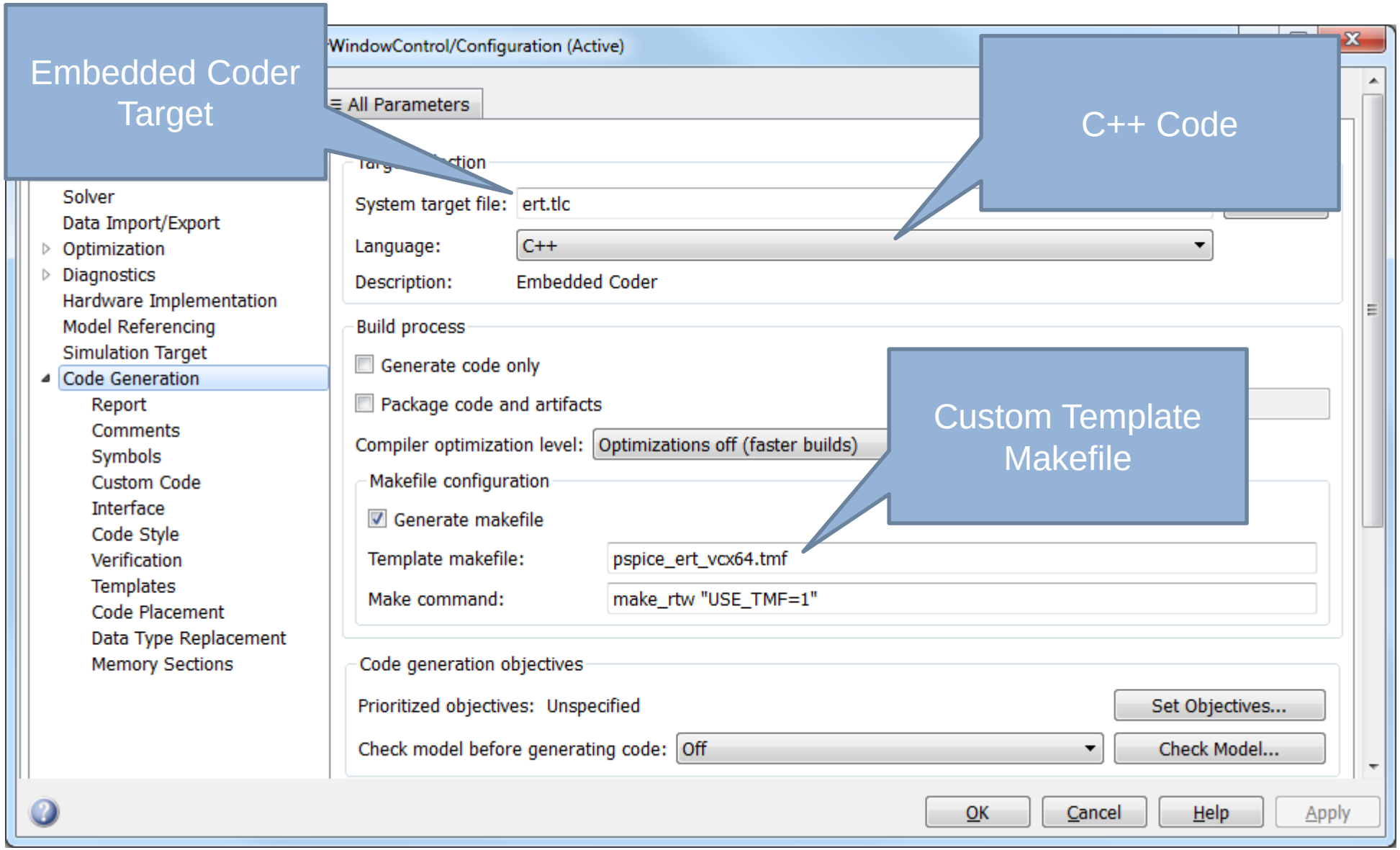
Device Modeling Interface – Steps for integrating Simulink models



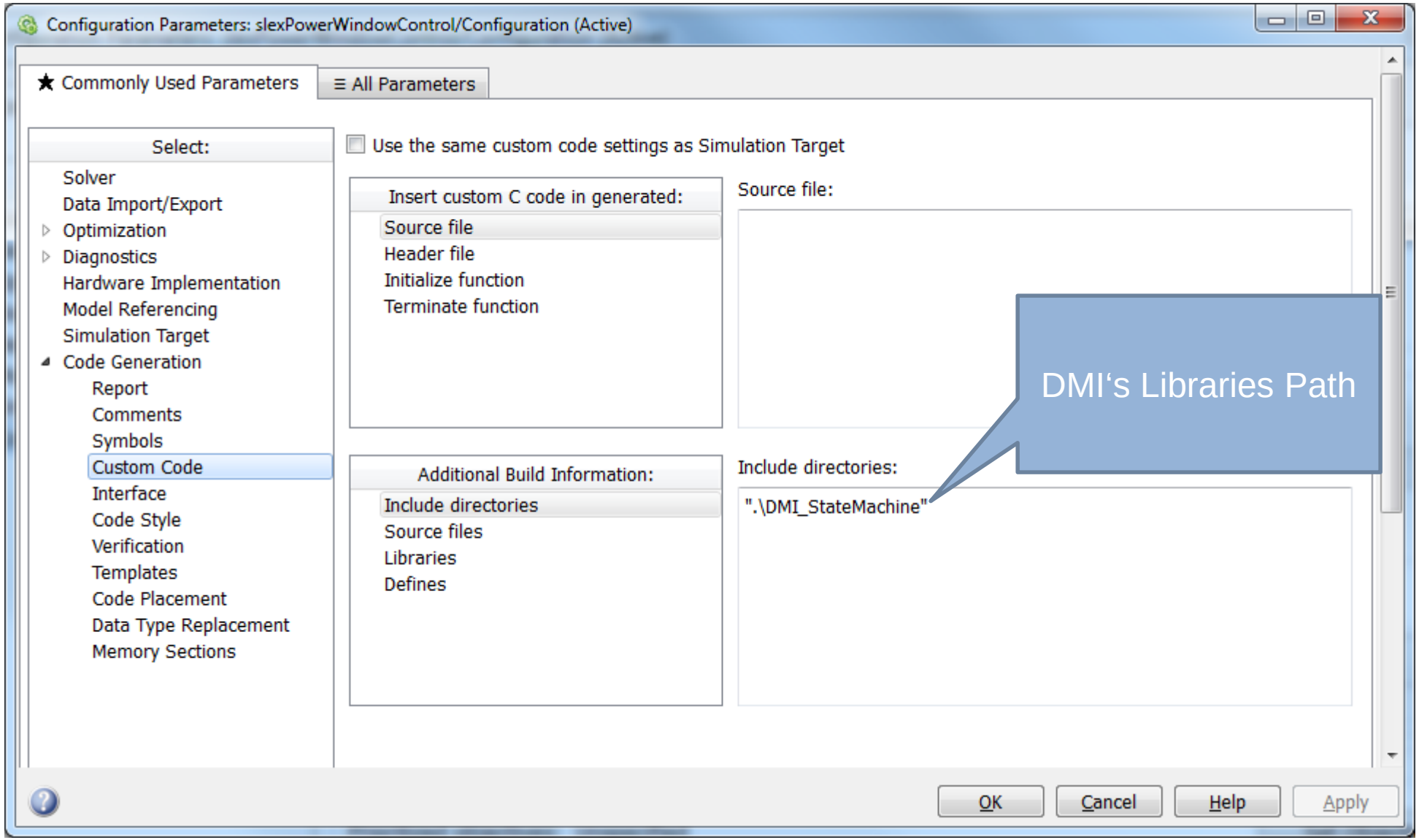
Simulink Model Example



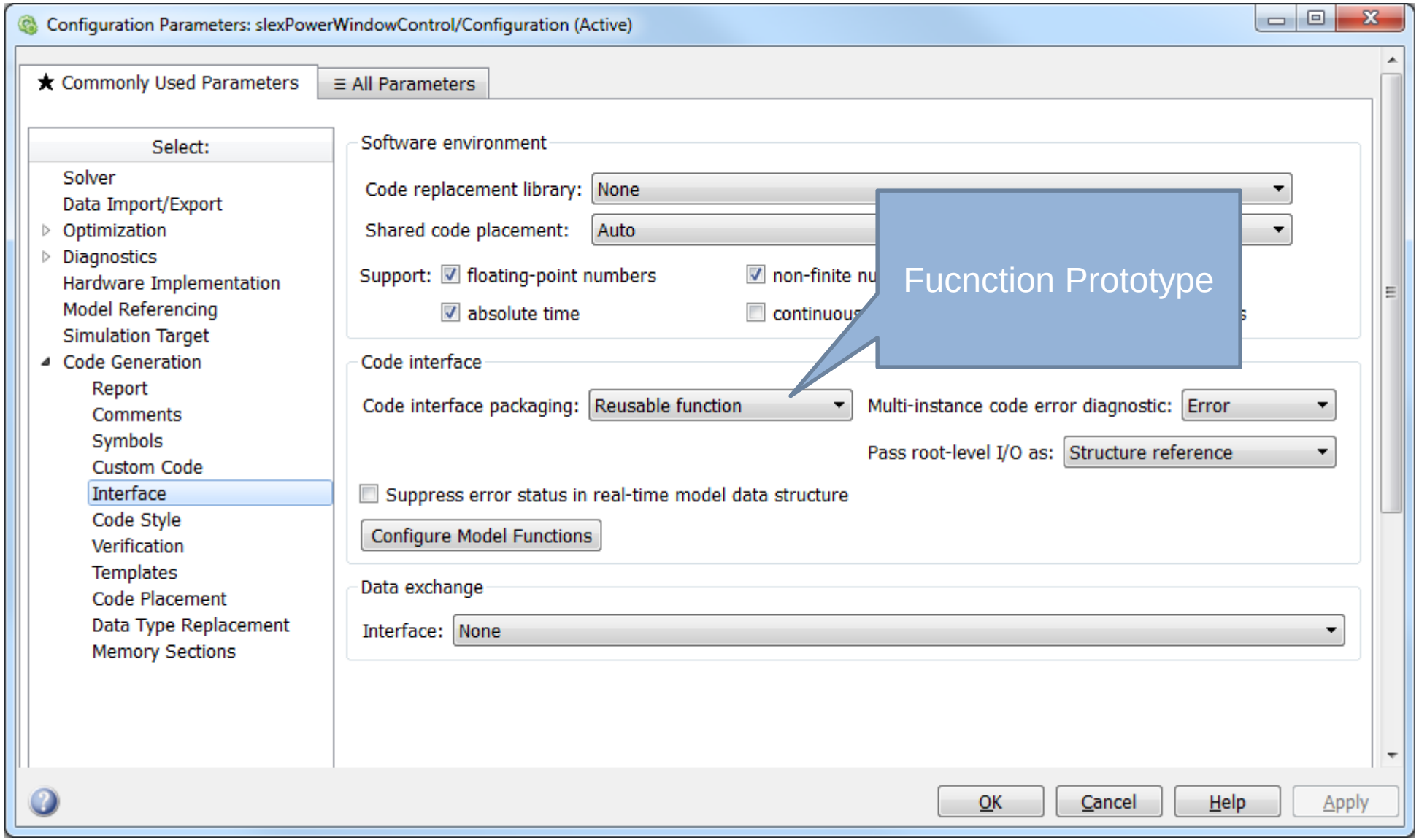
Simulink-PSpice Target Configuration – Code Generation



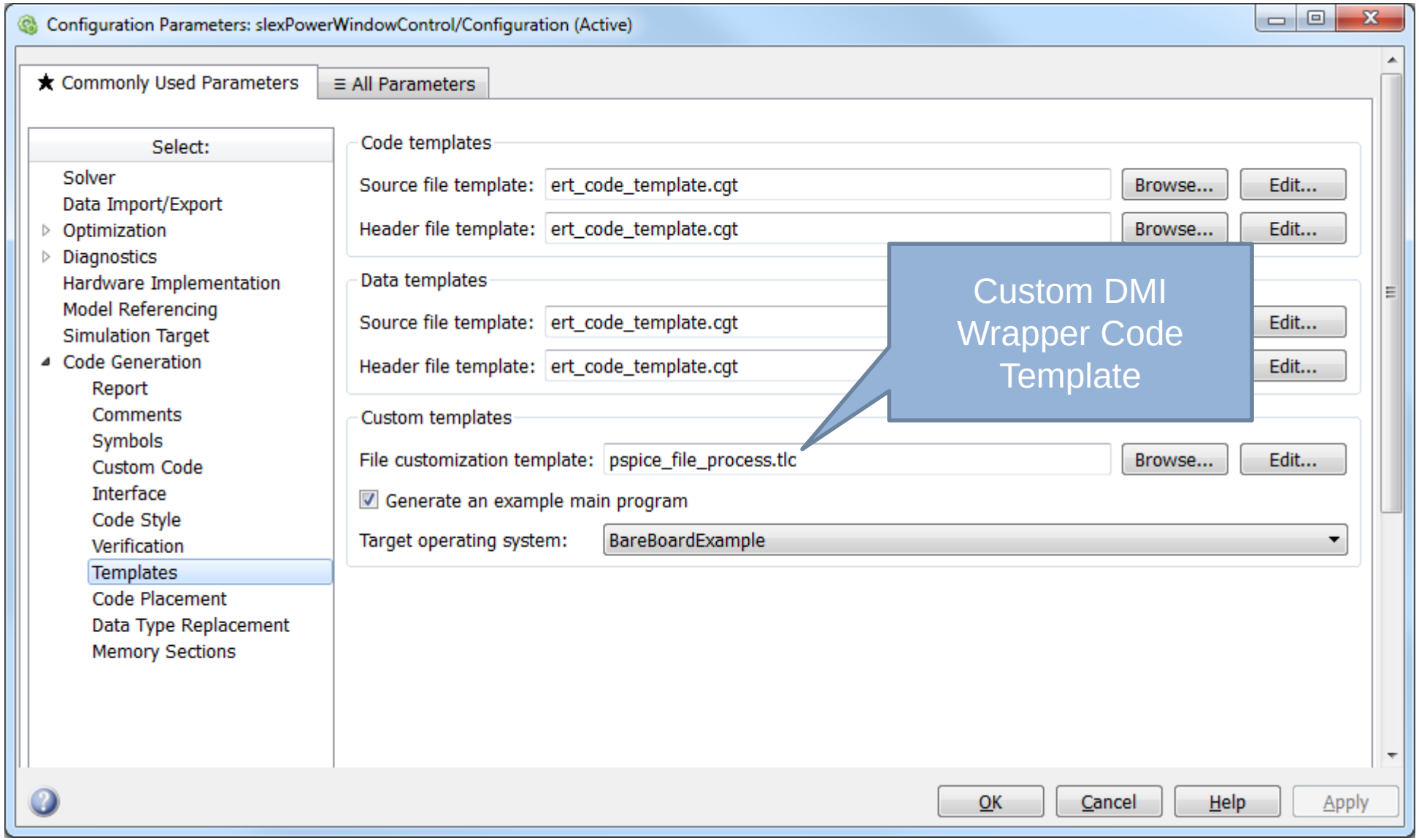
Simulink-PSpice Target Configuration – Custom Code



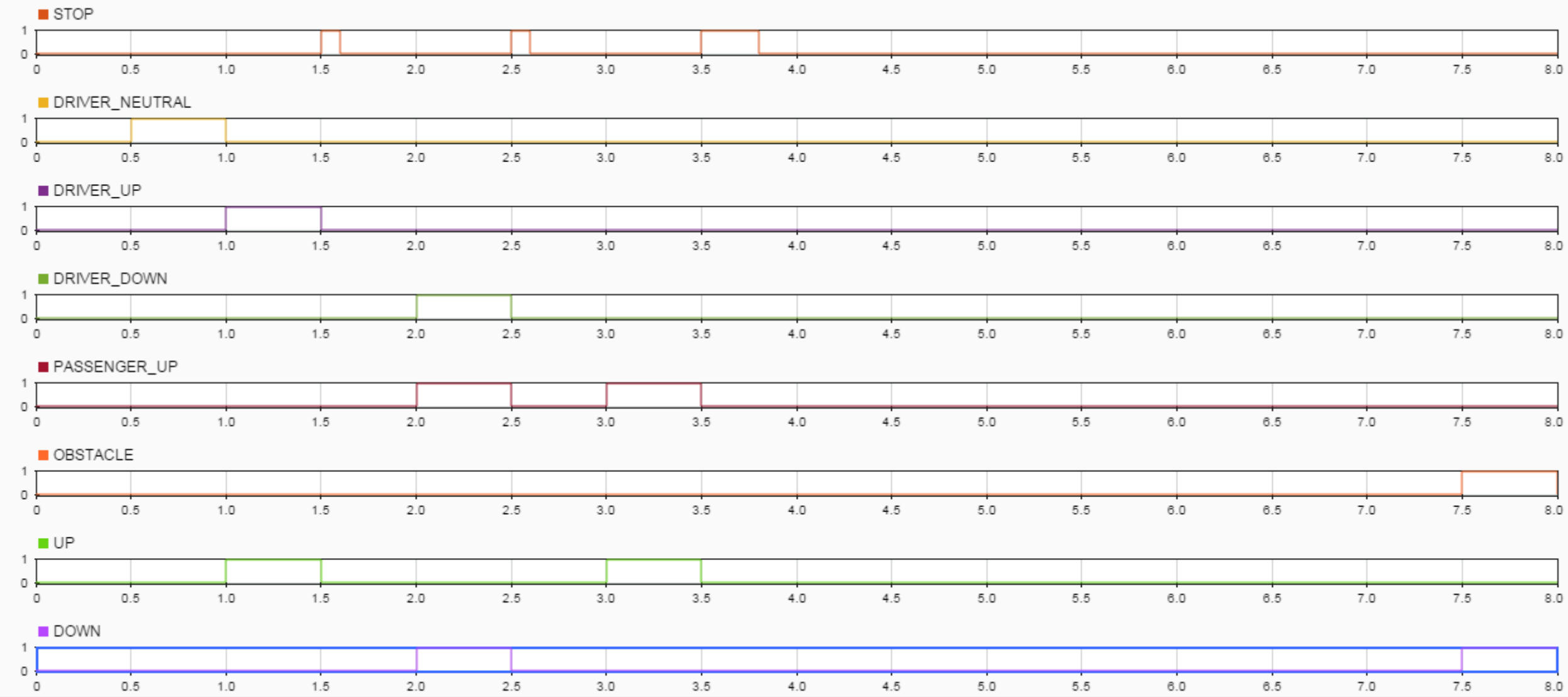
Simulink-PSpice Target Configuration – Interface



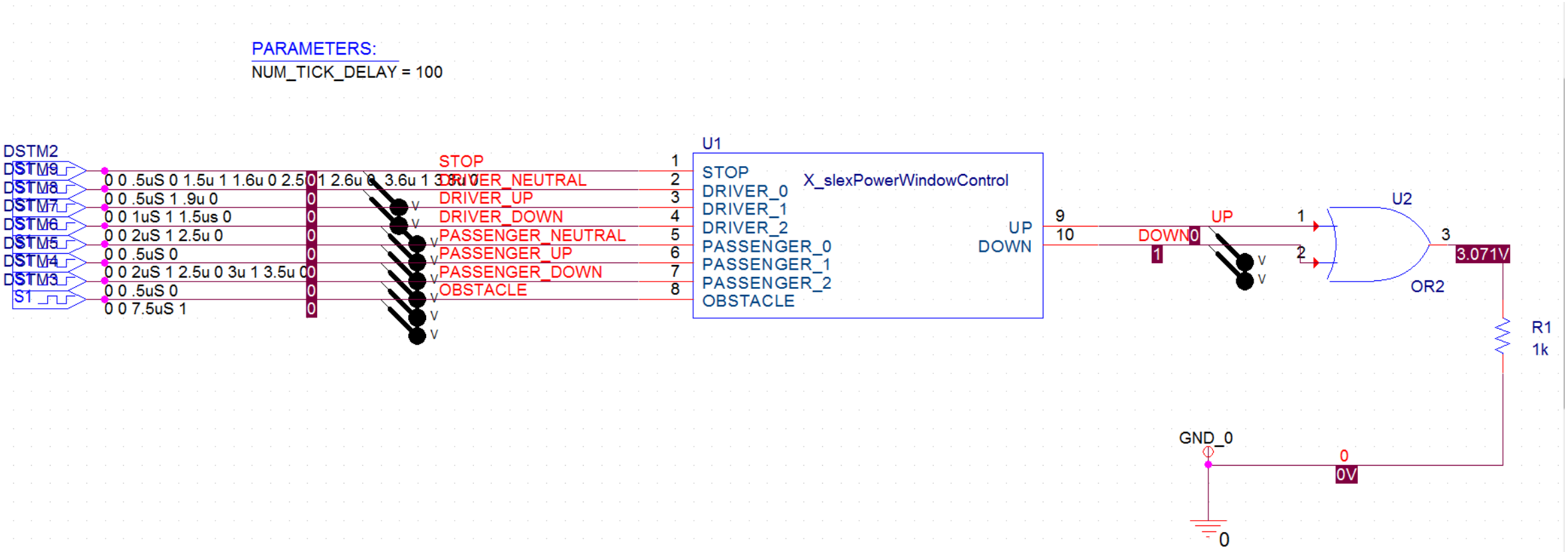
Simulink-PSpice Target Configuration – Templates



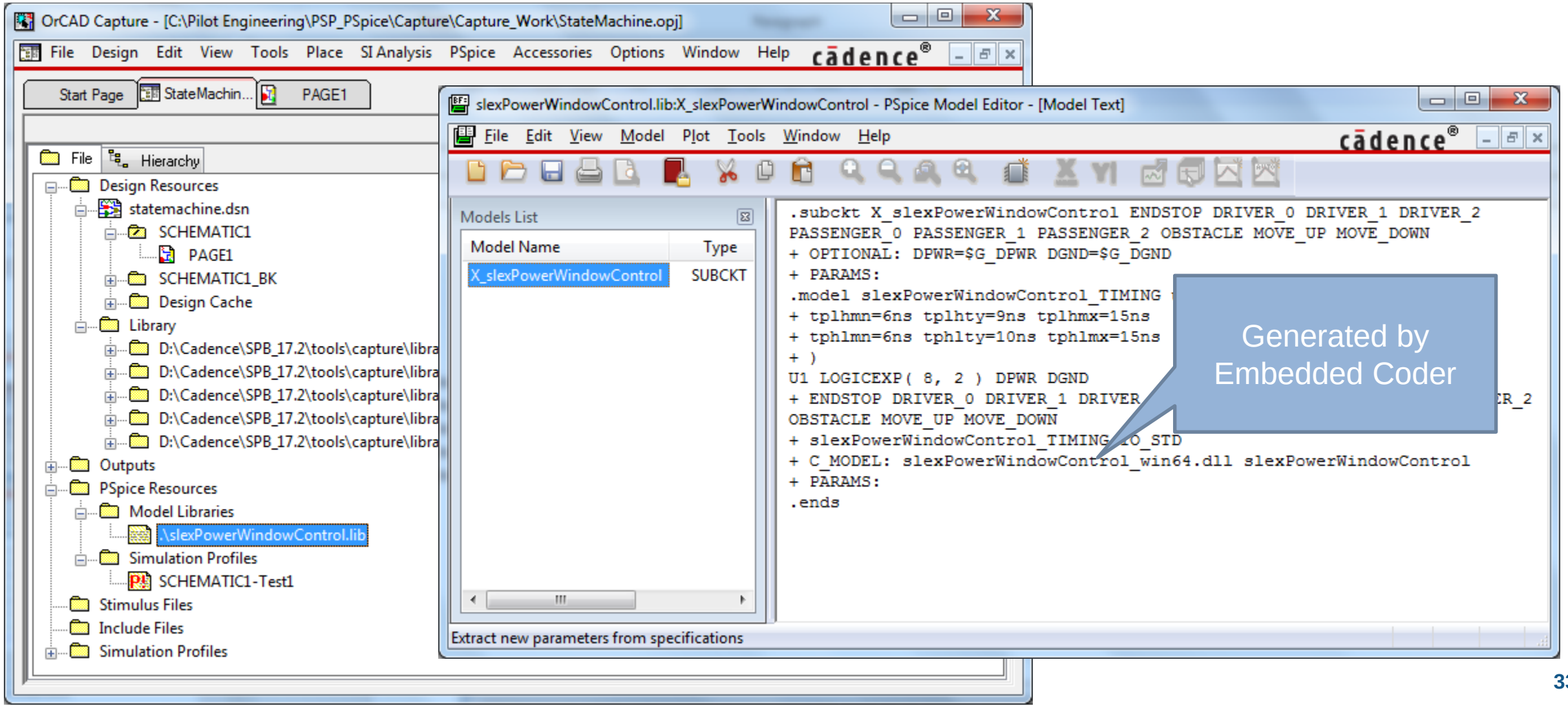
Simulink Simulation Results



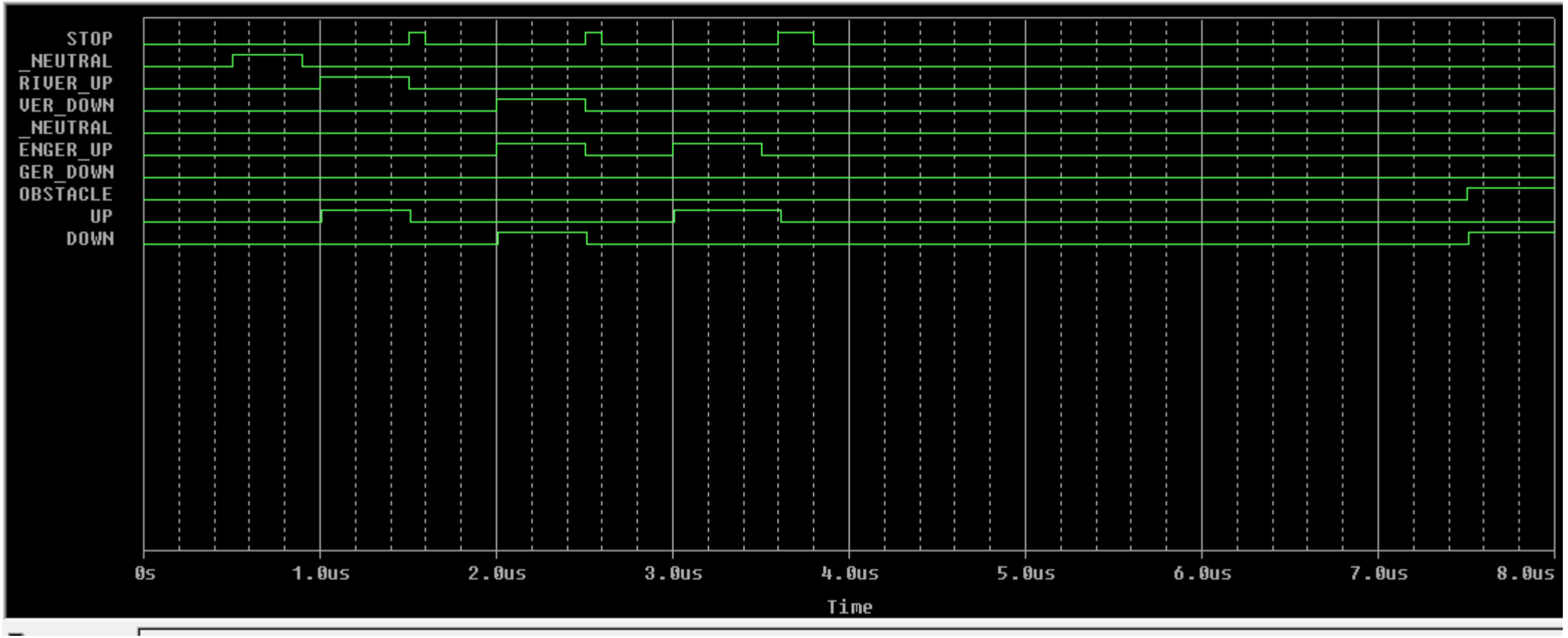
PSpice Model Example



PSpice DMI Library



PSpice Simulation Results



Demo

Q&A

- MathWorks's Point-of-Contact:
  - Bao Nguyen Bao.Nguyen@mathworks.com
  - Corey Mathis Corey.Mathis@mathworks.com

- Cadence's Point-of-Contact :
  - Kishore Karnane karnane@cadence.com

Conclusion

- SLPS is a needed tool because of:
  - Introduction of newest technologies and efficient methods.
  - Possibility to verify and optimize SW-Algorithms with HW-Models.
  - Reconnaissance and compensation of errors during the specification and implementation reducing development time.

- DMI increase the possibilities:
  - System Level Simulation importing C/C++/SystemC and Simulink Blocks into a unique simulator.
  - Hardware in the Loop, getting the results in a completely reliable environment to test the new critical functions.

Conclusion