

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

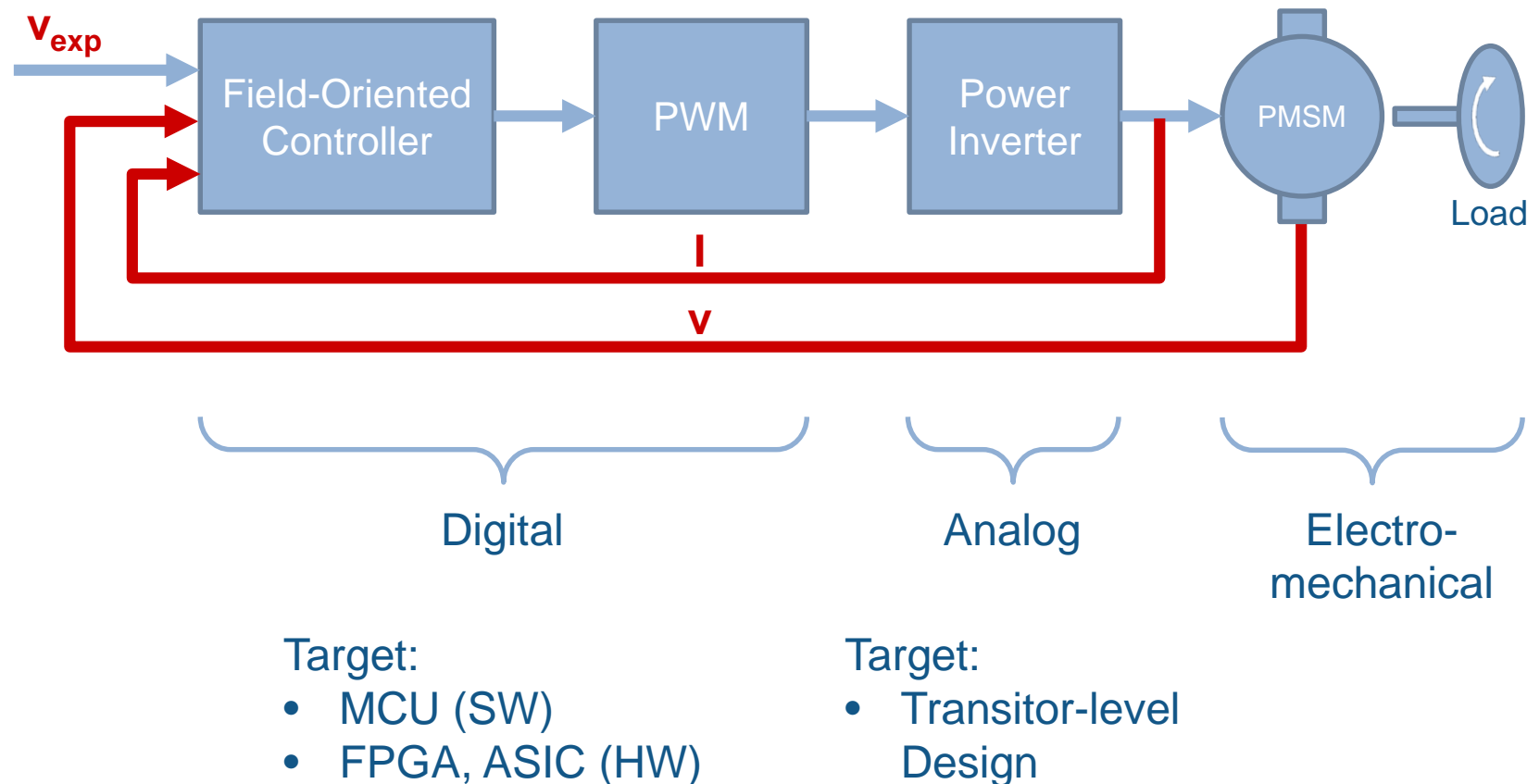
AGENDA

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

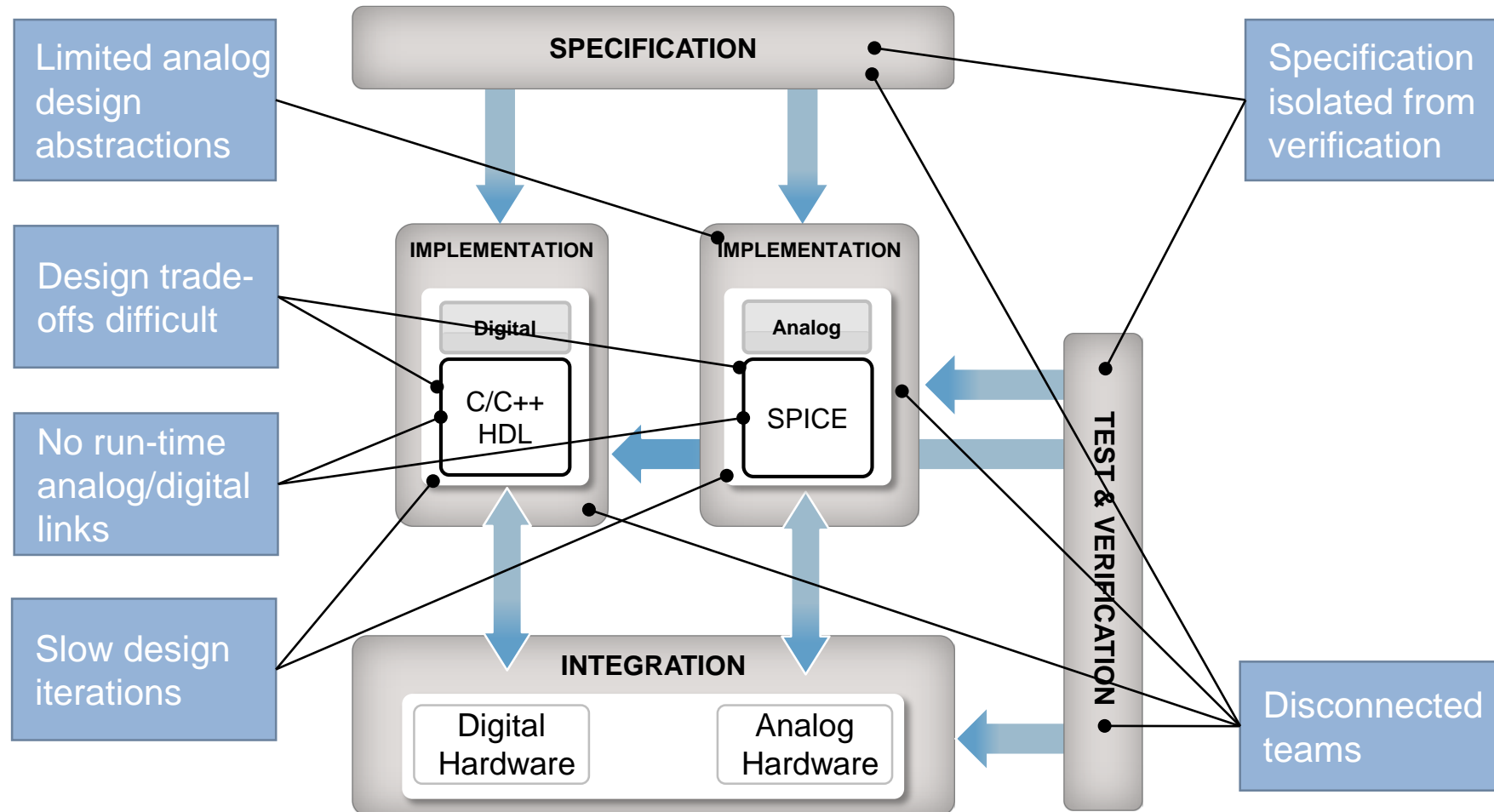
Analog/Mixed-Signal Design

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

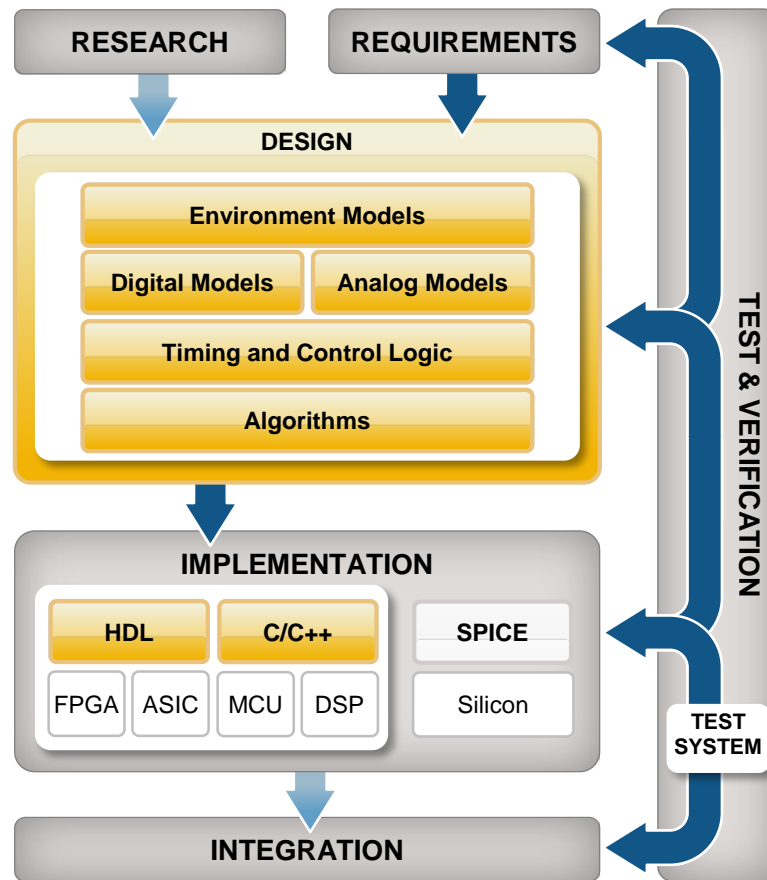
This control technique is common in motor drive systems for hybrid electric vehicles, manufacturing machinery, and industrial automation



Challenges in Classical Mixed-Signal Design

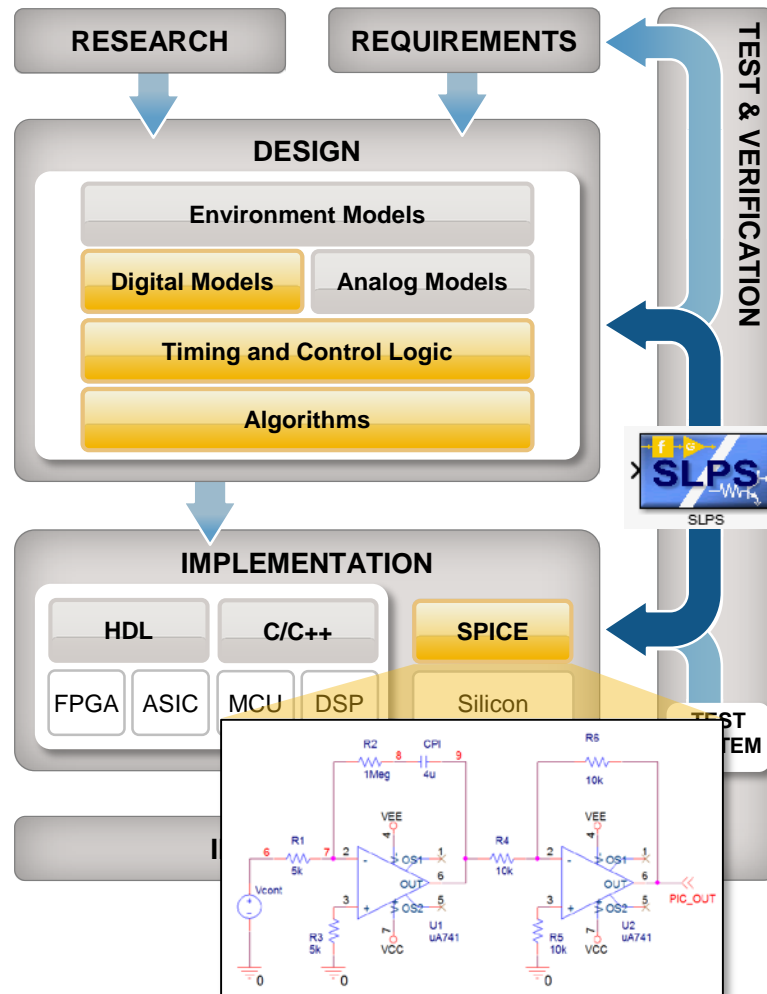


Model-Based Design



- „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

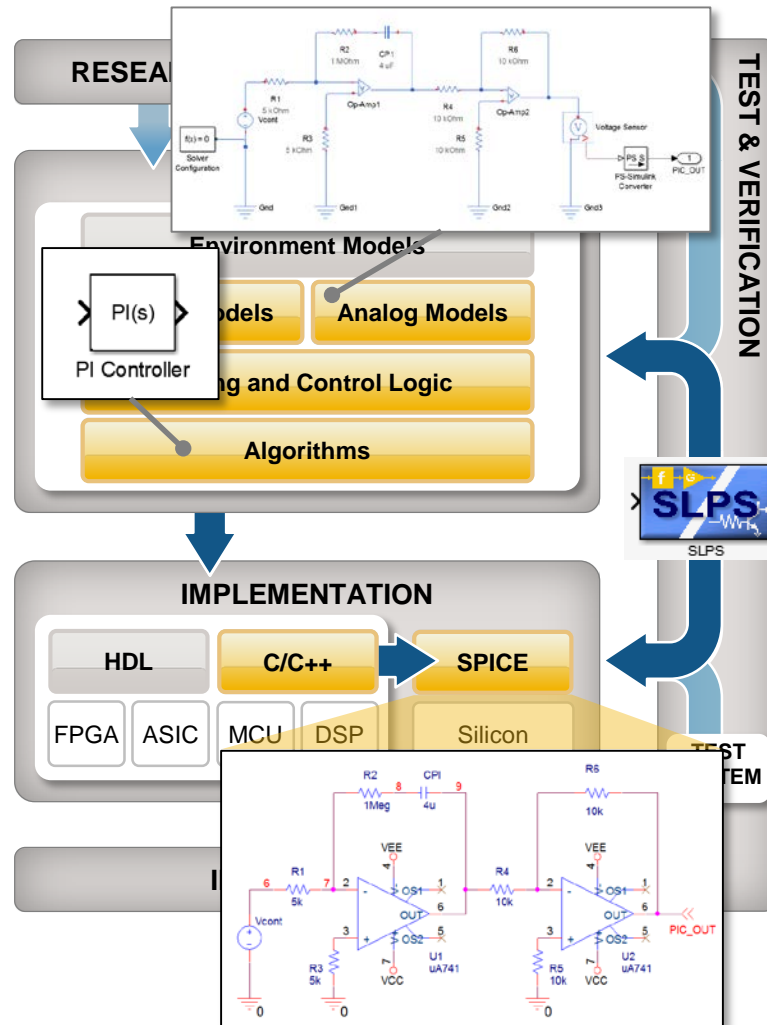
Model-Based Design for Analog/Mixed-Signal



- **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

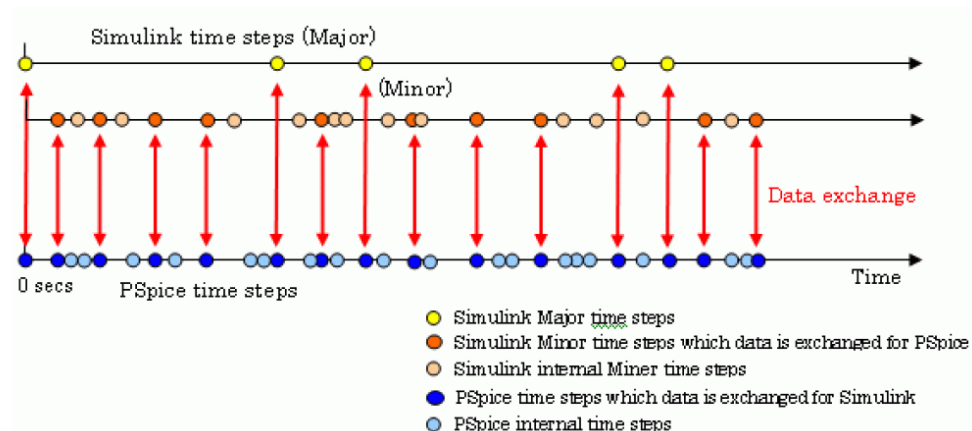
— Solution

- Co-simulation with Simulink and PSpice using SLPS
- Model integration through automatic C code generation and PSpice DMI

What is SLPS?

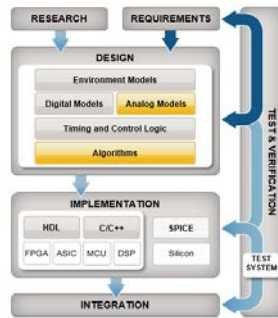
- SLPS = Simulink + PSpice Co-Simulation
 - Simulink
 - Multi-domain simulation environment for dynamic systems
 - Algorithm development and verification platform
 - PSpice:
 - SPICE-based simulator
 - Simulation of electrical and electronic circuits
 - Circuit design platform → Hardware

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

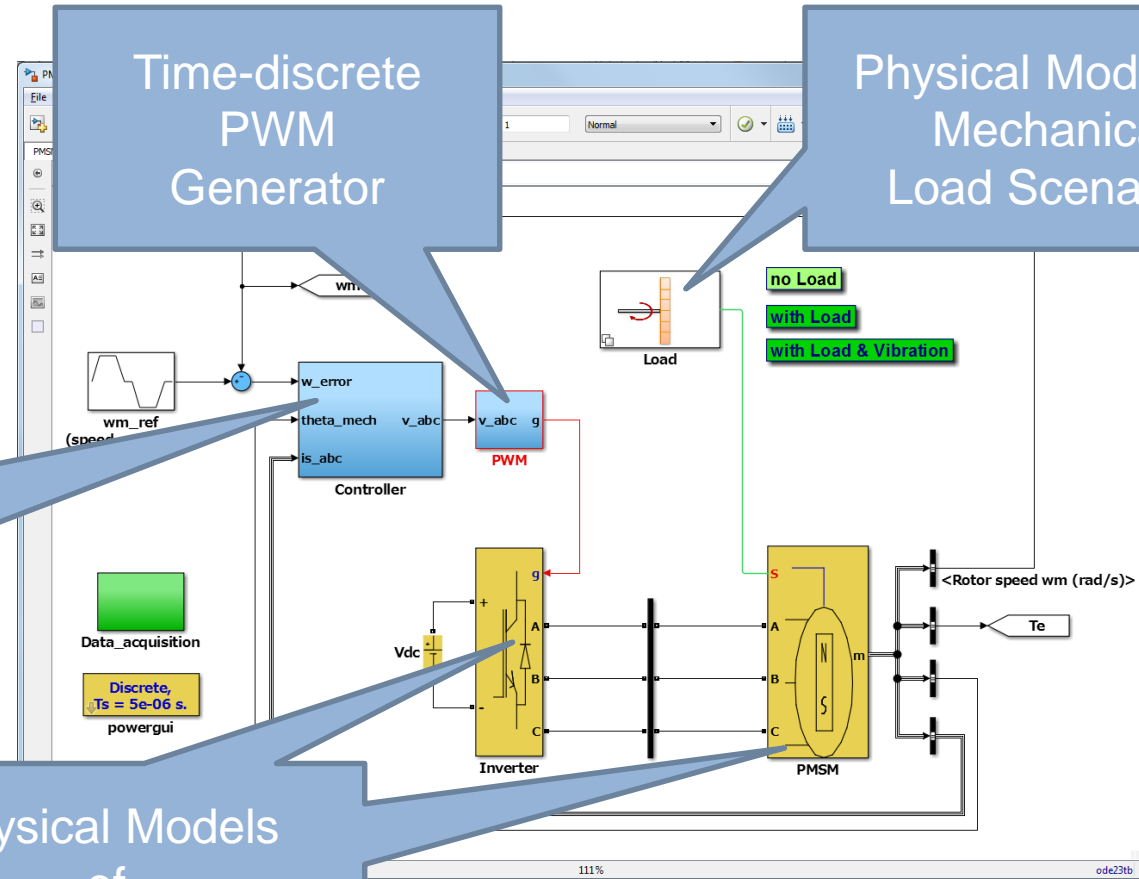


Time-discrete
PWM
Generator

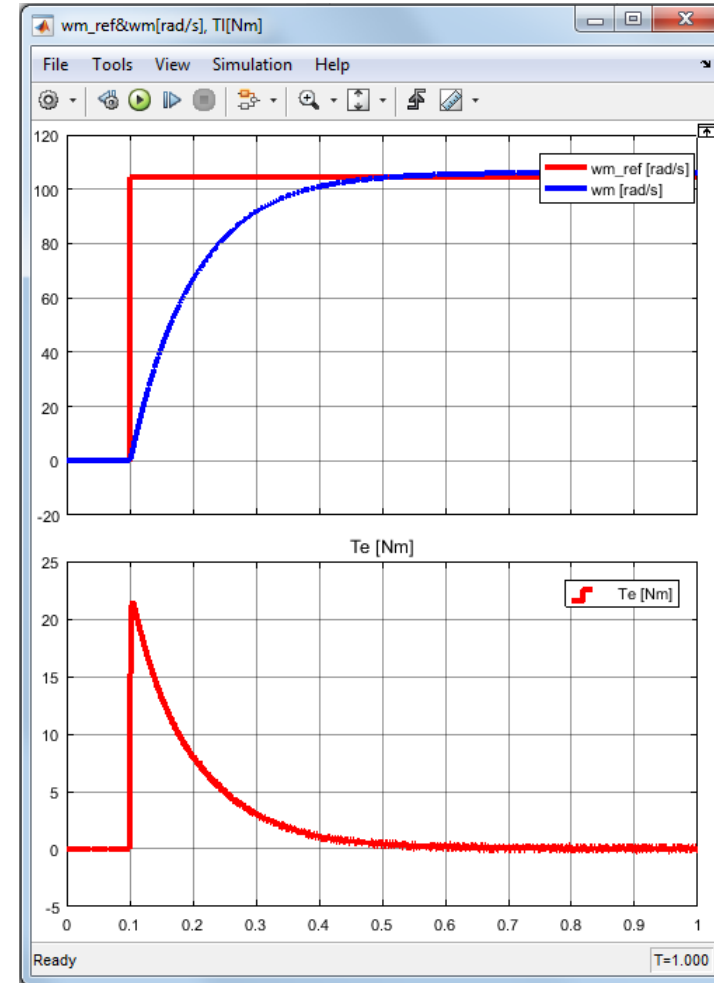
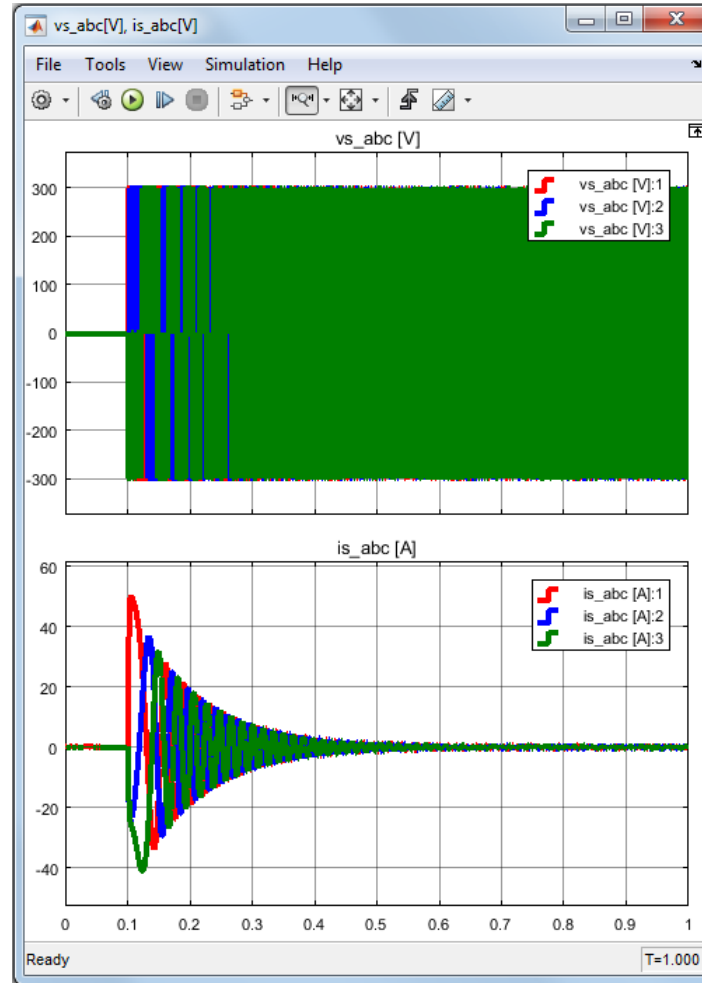
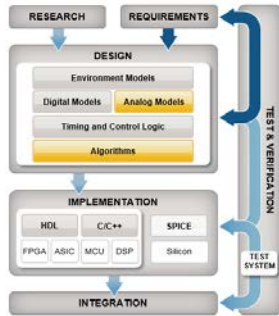
Physical Models of
Mechanical
Load Scenarios

Time-continuous
PI Controllers
(speed, current)

Physical Models
of
Electrical Components

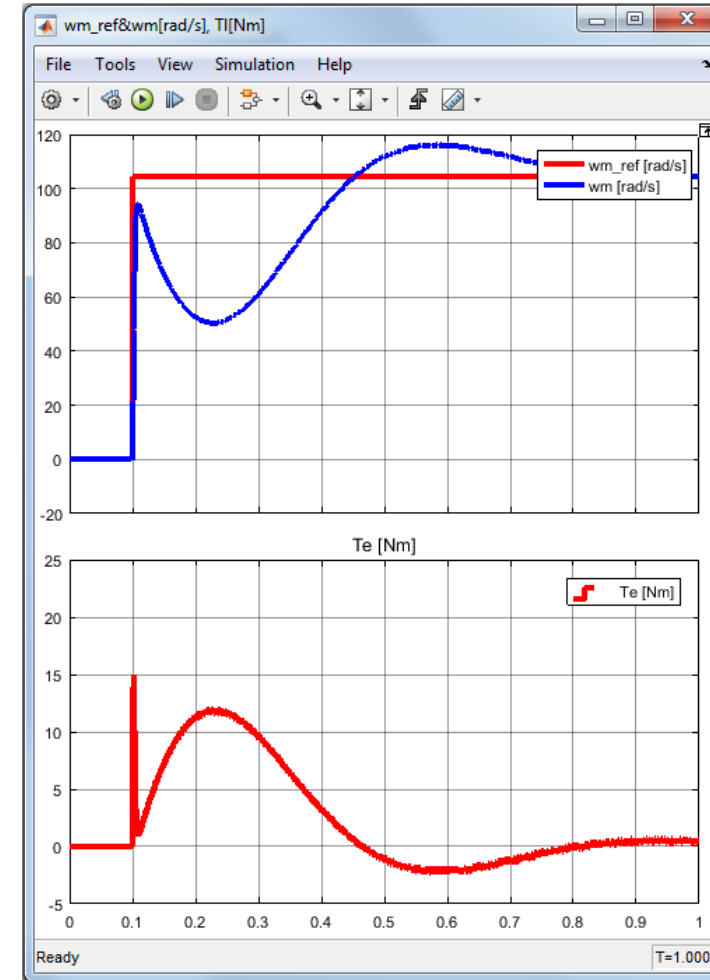
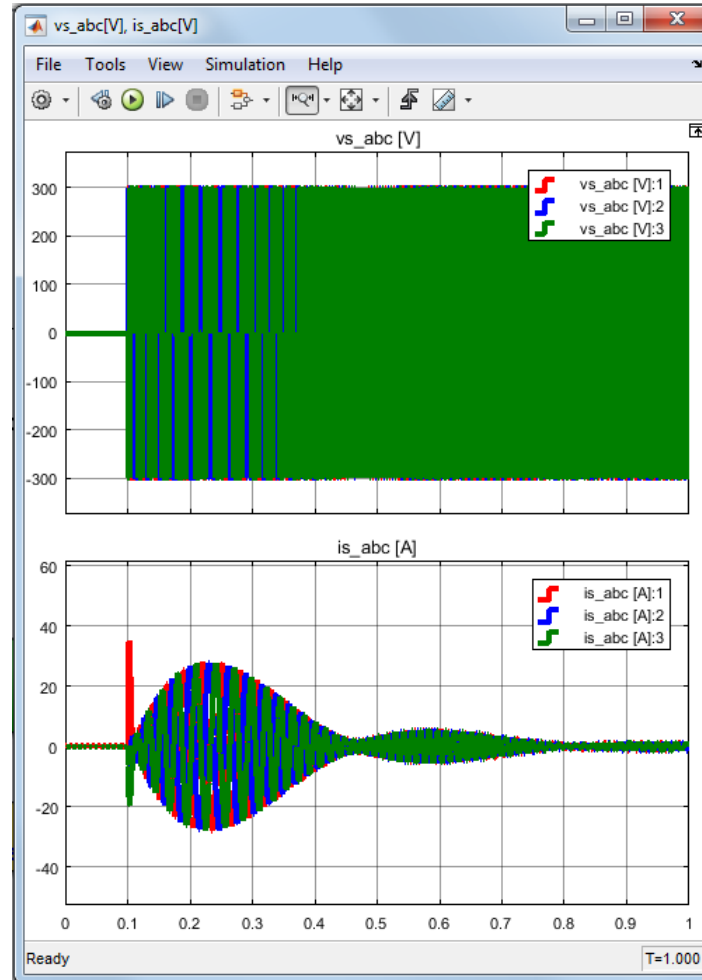
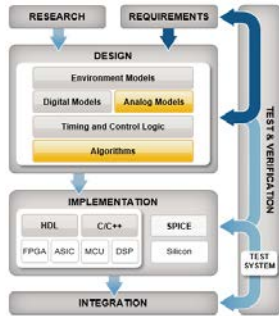


Step 1: Algorithm Design and Verification



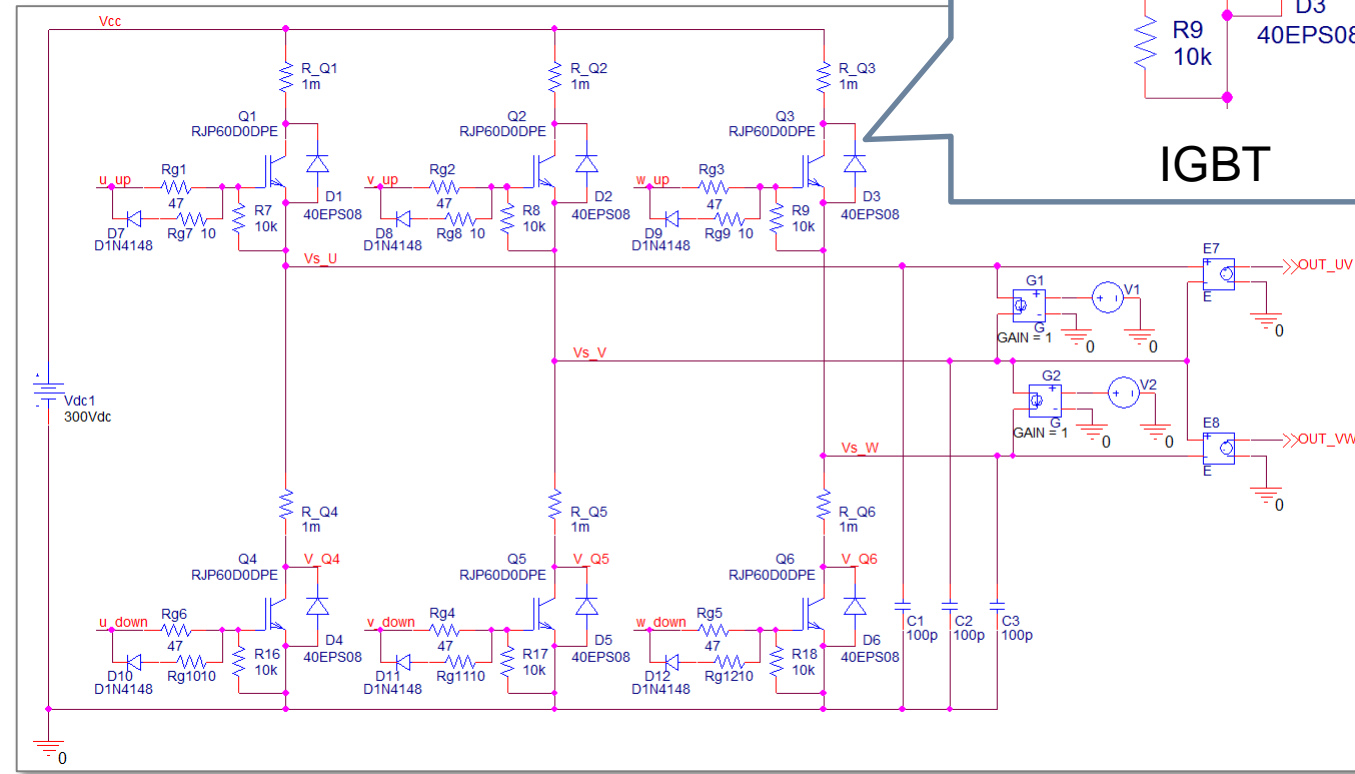
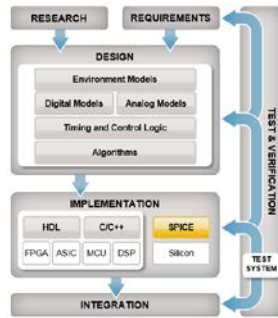
Load Scenario: with Load

Step 1: Algorithm Design and Verification

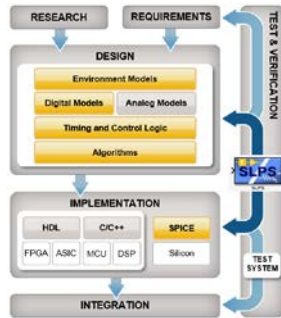


Load Scenario: with Load and Vibration

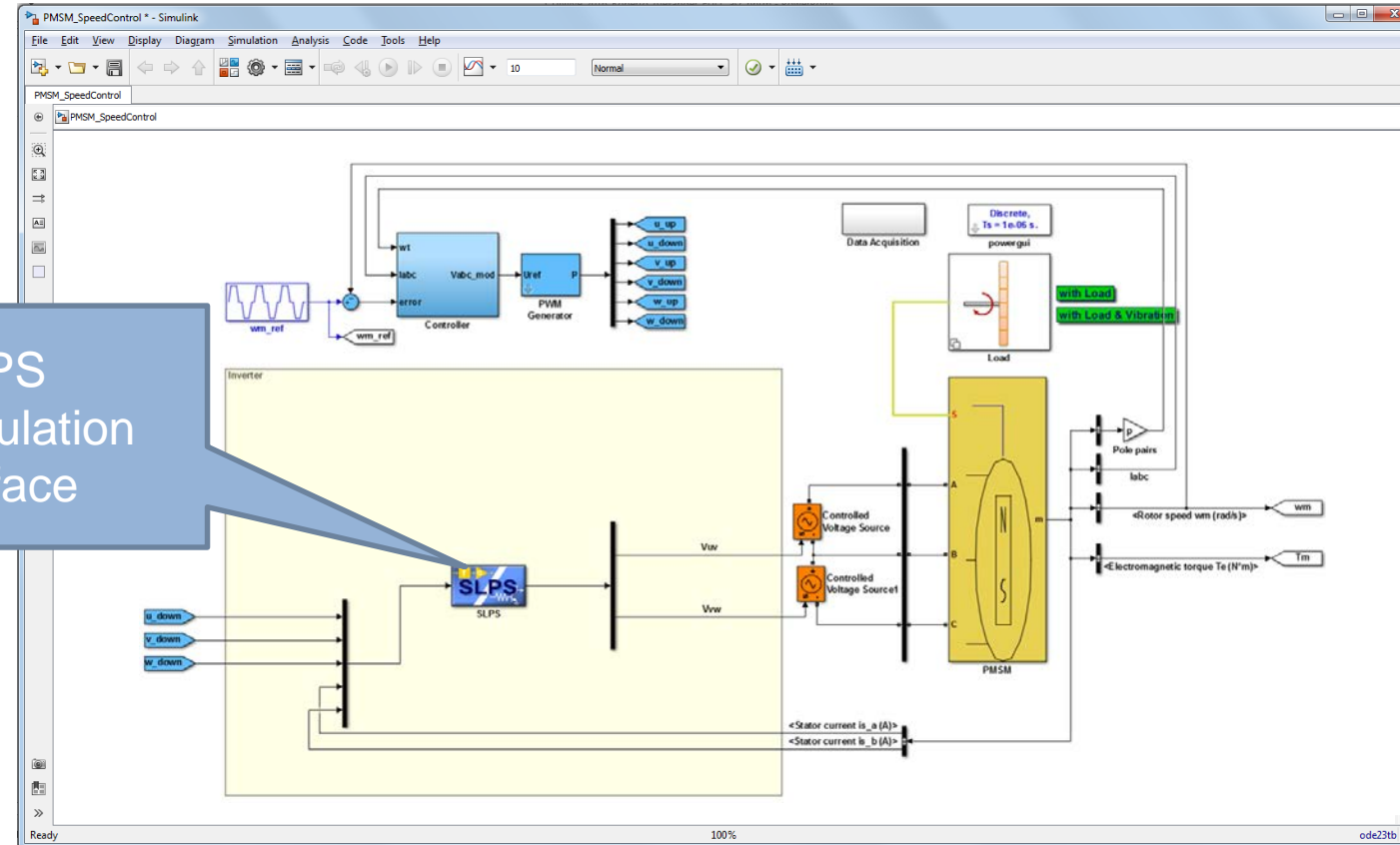
Step 2: Schematic Entry (PSPice)



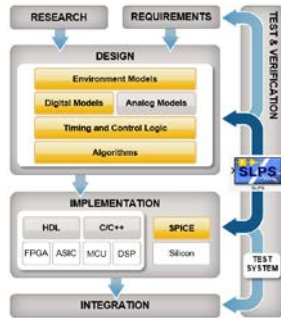
Step 3: Simulink/PSpice Co-Simulation (SLPS)



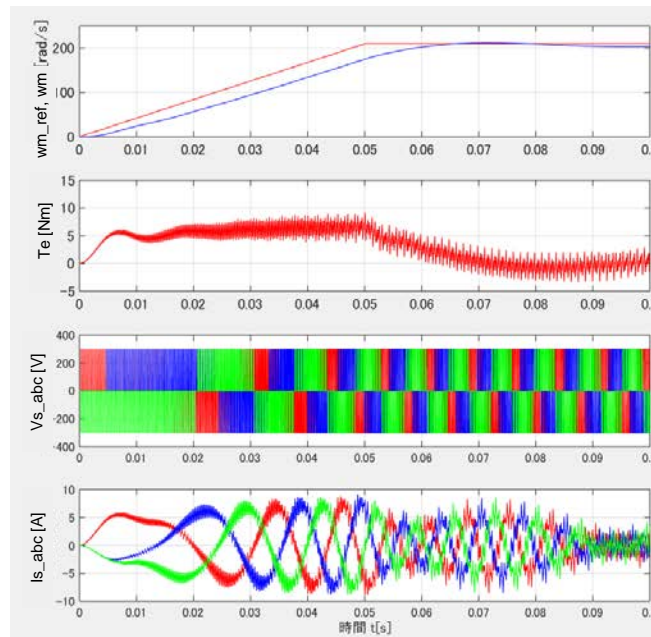
SLPS
Co-Simulation
Interface



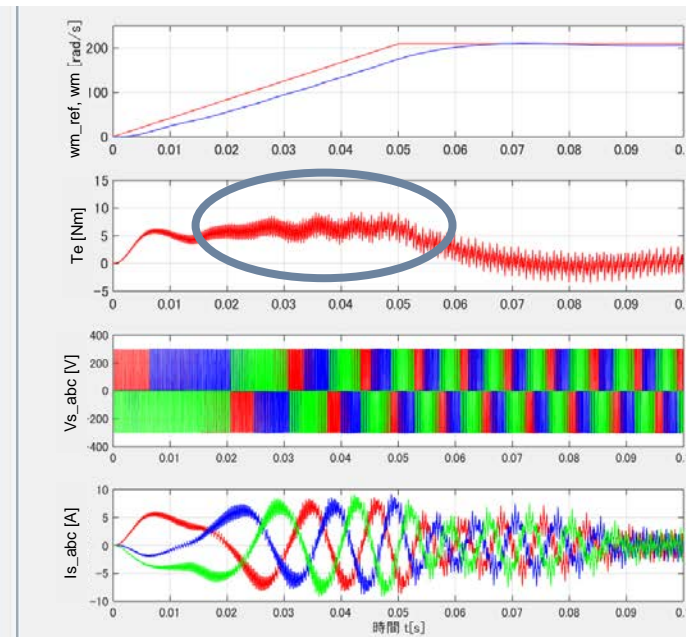
Step 3: Simulink/PSpice Co-Simulation (SLPS)



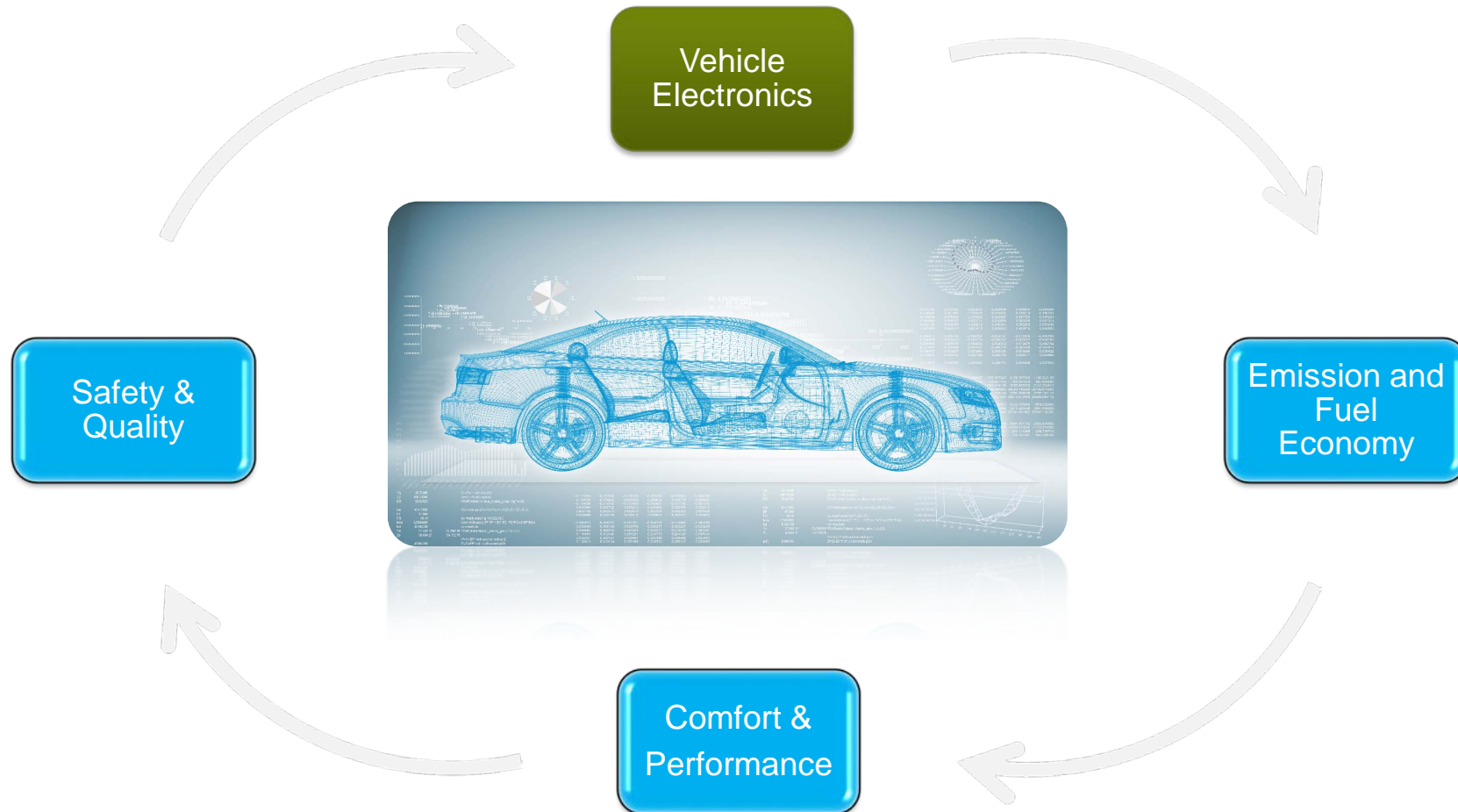
Simulink Simulation



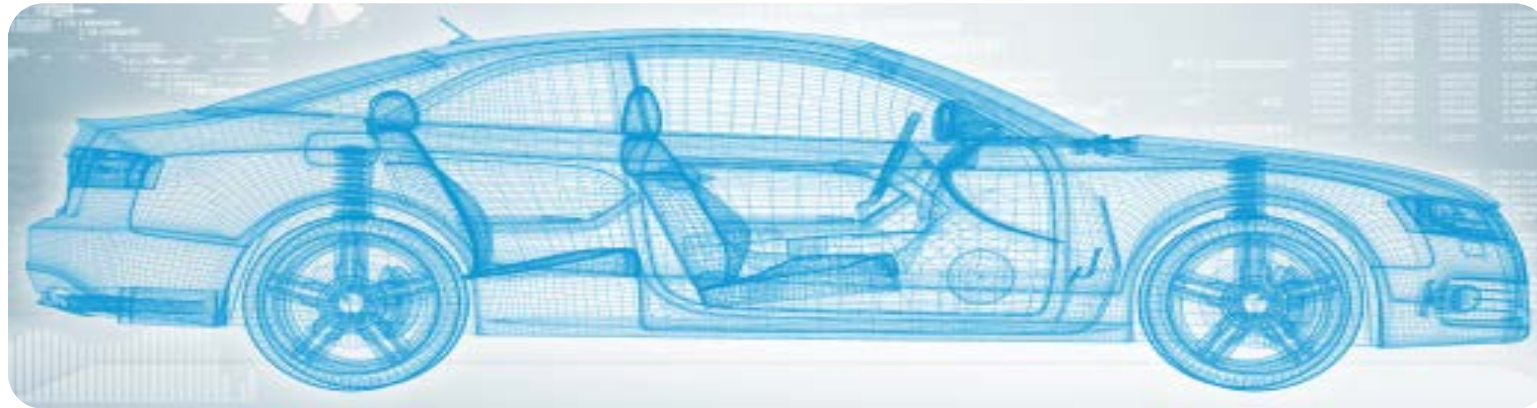
Simulink / PSpice
Co-Simulation
(SLPS)



Automotive Engineering Interdisciplinary Design Challenge



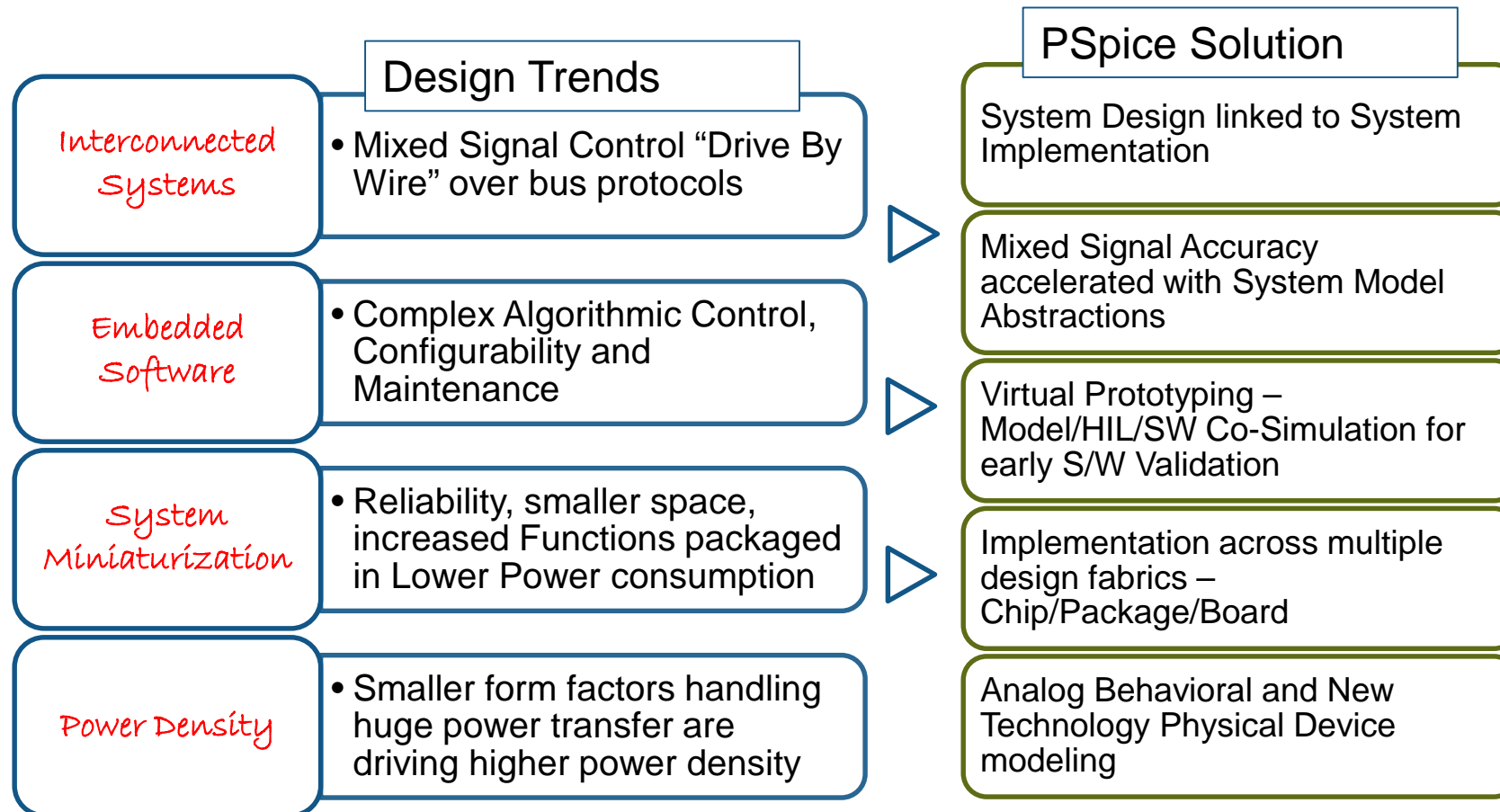
Automotive Engineering Interdisciplinary Design Challenge



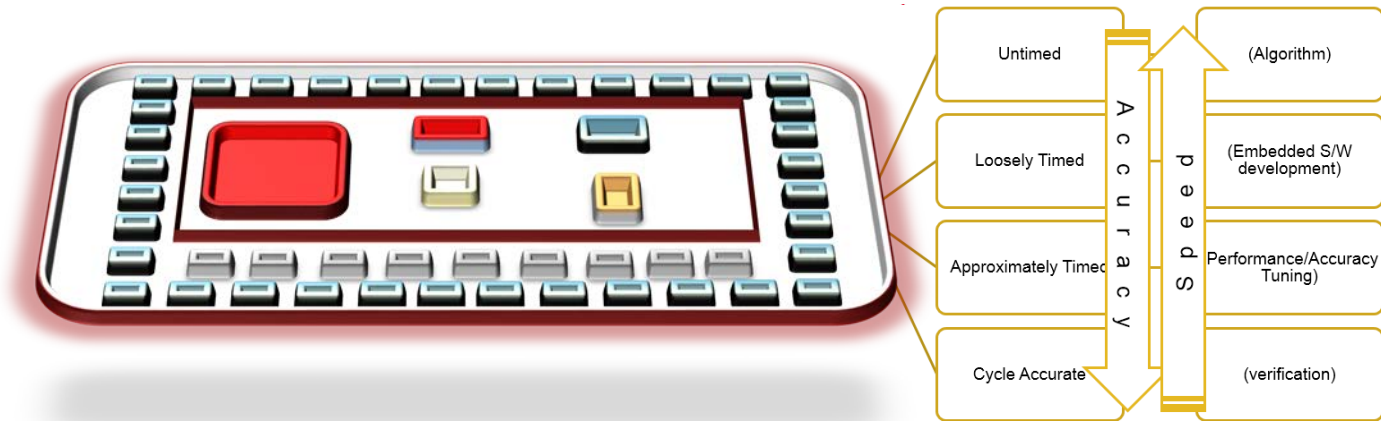
Vehicle Electronics

Explosion of **Interconnected** Electronic Systems with **Embedded Software** having some very challenging **Power Density** issues created by **System miniaturization** 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



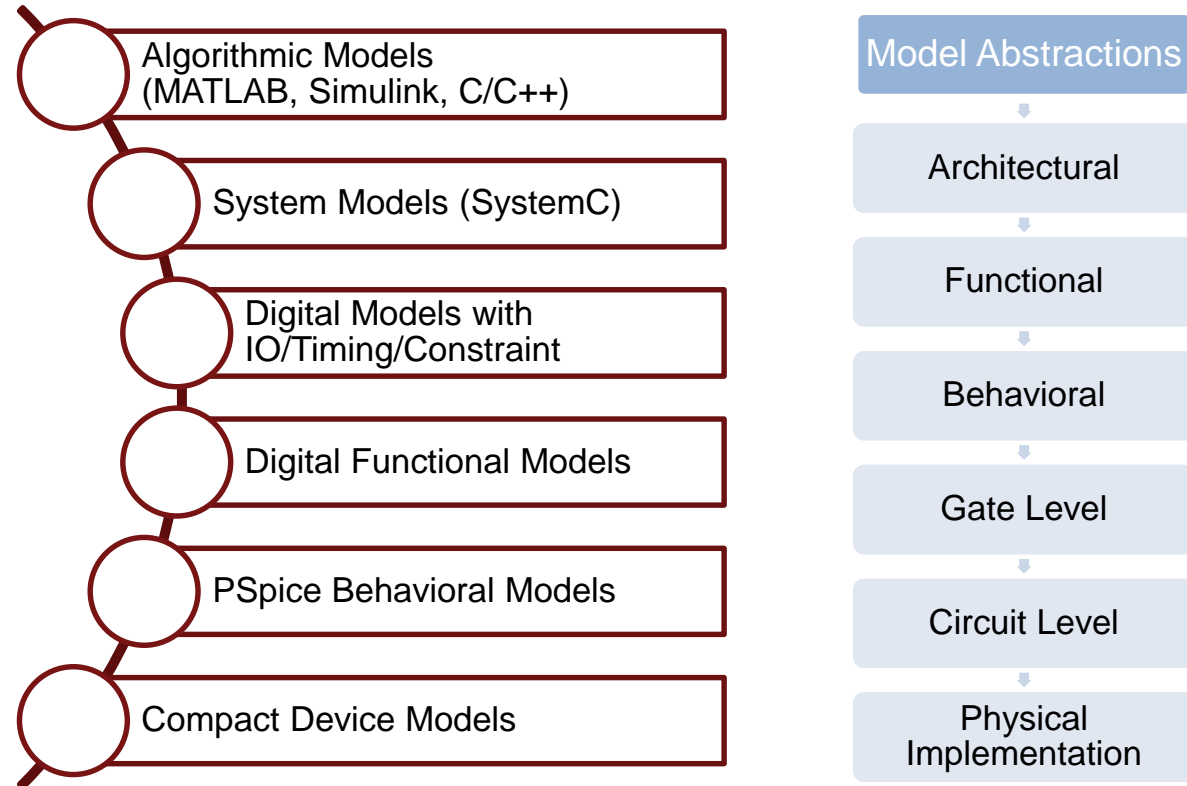
Analog behavioral



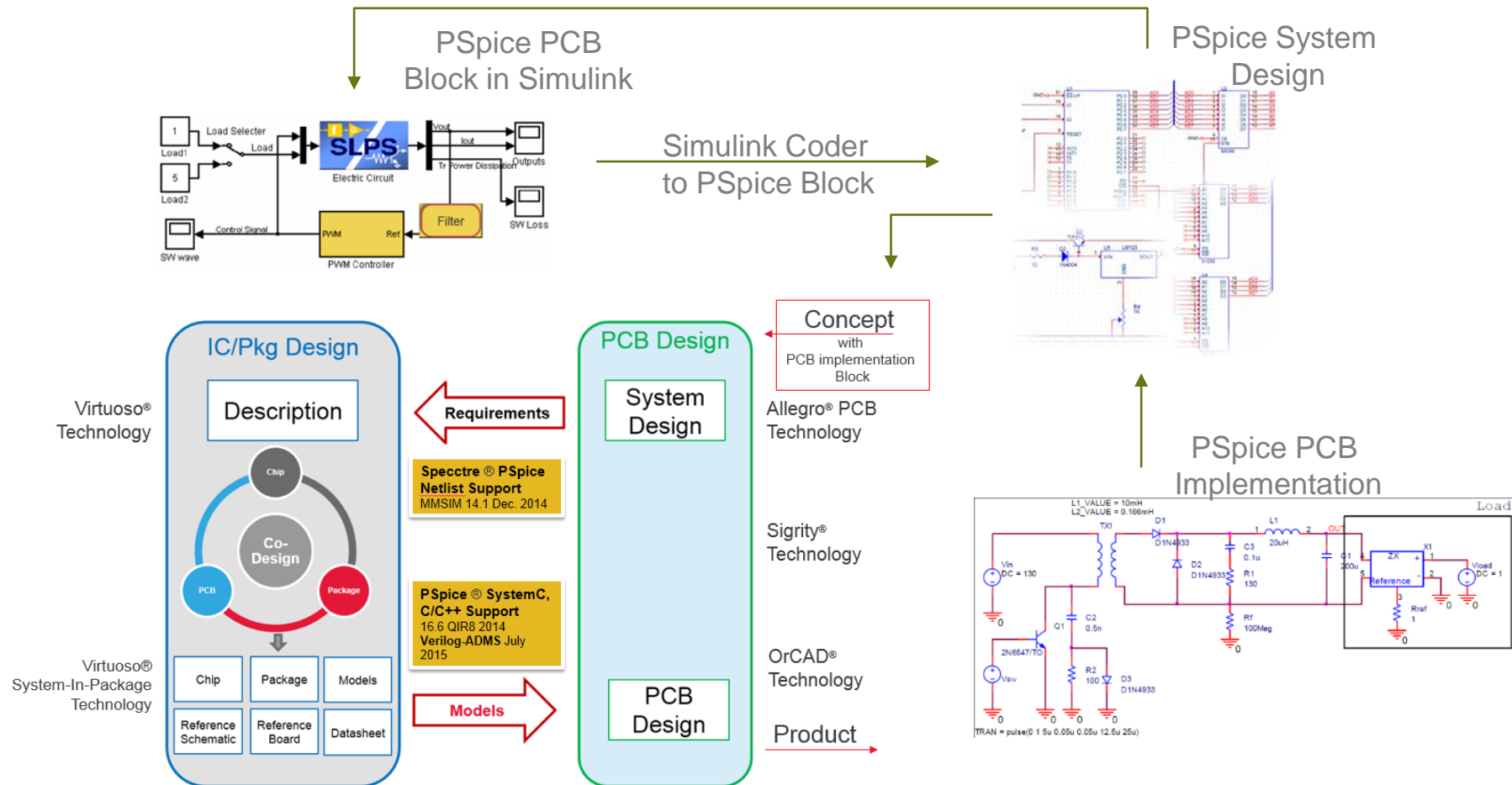
Digital C/C++ with embedded SW block

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

PSpice Models



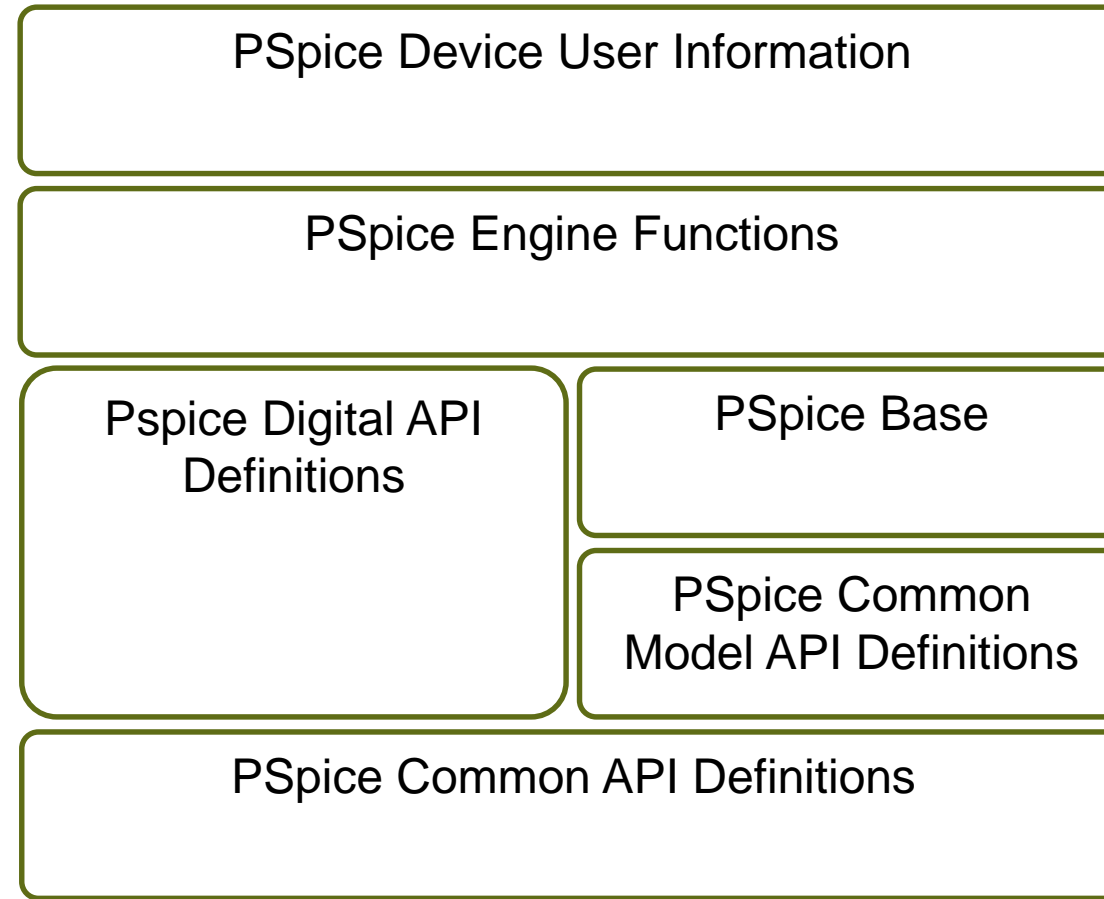
System Design Exploration to Implementation



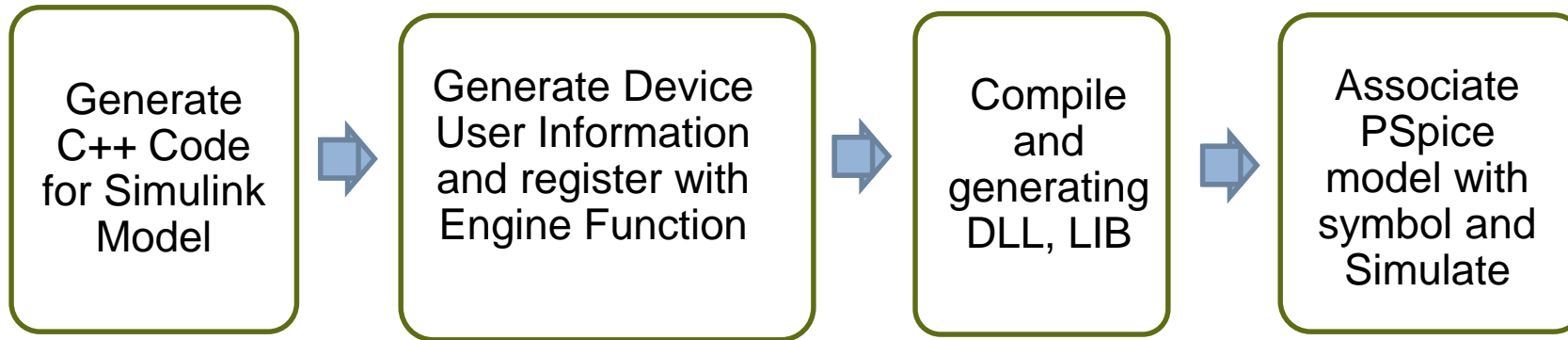
Device Modeling Interface



Device Modeling Interface Libraries

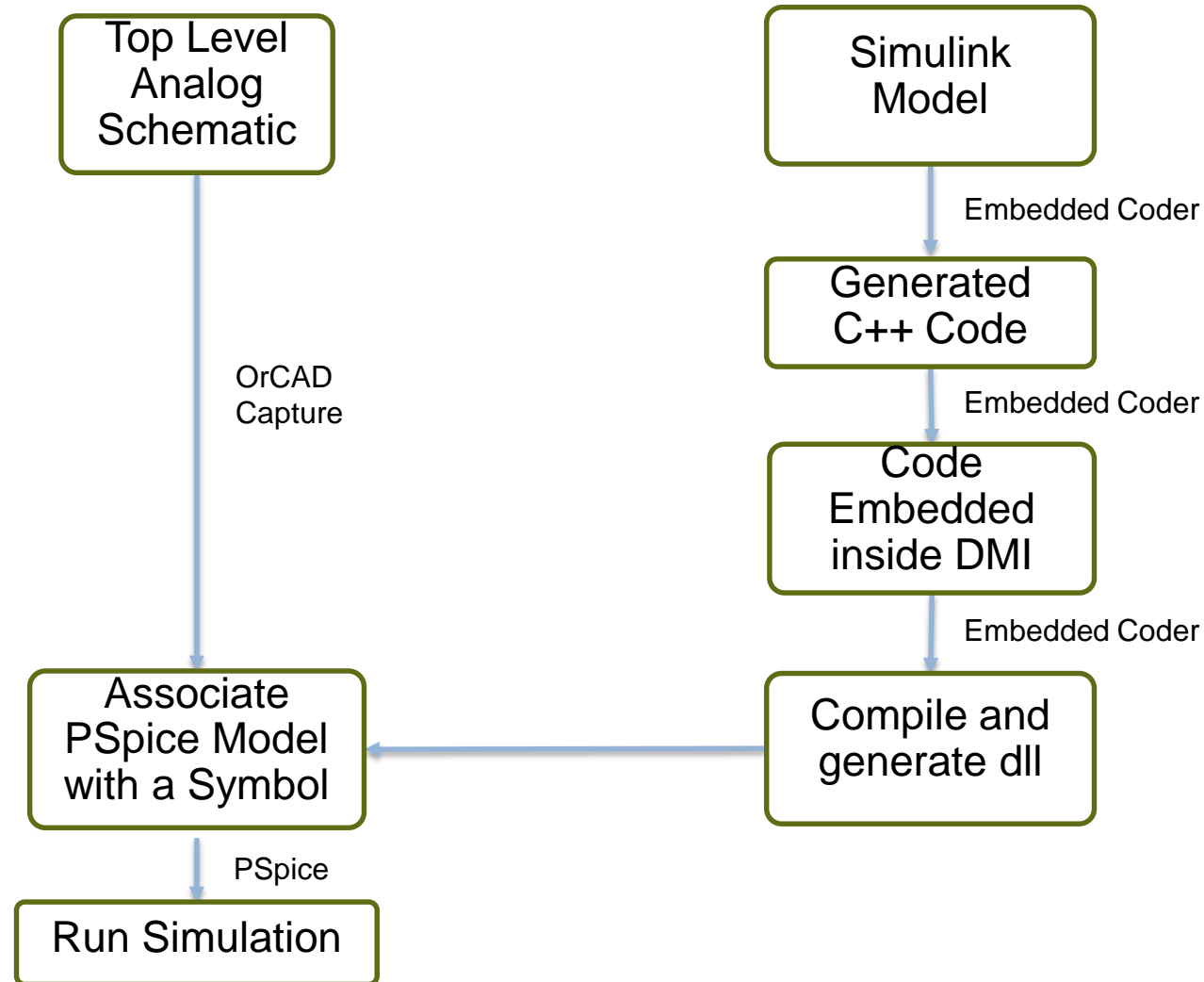


Device Modeling Interface – Embedded Coder Steps



- Requires Embedded Coder license

Device Modeling Interface – Steps for integrating Simulink models



Simulink Model Example

The image displays the MATLAB R2016a environment with a Simulink model. The left pane shows the file explorer for the 'Wrapper' folder, listing files such as 'slexPowerWindowControl_win64.dll', 'slexPowerWindowControl.lib', and 'pspice_file_process.tlc'. The main workspace shows a Simulink model diagram with various blocks and signals. The right pane shows a detailed view of the 'control' block, which is a state machine diagram with inputs for 'endstop', 'driver', 'passenger', and 'obstacle', and outputs for 'moveUp' and 'moveDown'. The diagram includes state transitions and logic for controlling a system.

File Explorer (Current Folder):

Name	Size	Date Modified
slexPowerWindowControl_win64.dll	136 KB	8/22/2016 5:55 PM
slexPowerWindowControl_win64.dll.manifest	1 KB	8/22/2016 5:55 PM
slexPowerWindowControl.lib	1 KB	8/22/2016 5:55 PM
pspice_file_process.tlc	31 KB	8/22/2016 5:54 PM
slexPowerWindowControl.slx	27 KB	8/22/2016 9:26 AM
pspice_ert_vcx64.tmf	21 KB	7/28/2016 11:38 PM
sldemo_radar_eml_test_sfuns.mexw64	104 KB	7/12/2016 5:09 PM
slexPowerWindowControl_ert_rtw		8/22/2016 5:55 PM
slprj		8/22/2016 11:27 AM
DMI_StateMachine		6/25/2016 9:01 AM
pspEngFunc.h	7 KB	12/3/2015 6:55 PM
PspiceBase.h	8 KB	12/3/2015 6:55 PM
PspiceCMIApiDefs.h	30 KB	12/3/2015 6:55 PM
PspiceCommonAPIDefs.h	13 KB	12/3/2015 6:55 PM
PspiceDigApiDefs.h	16 KB	12/3/2015 6:55 PM
StdAfx.h	1 KB	8/25/2015 6:56 PM
StdAfx.cpp	1 KB	8/25/2015 6:56 PM

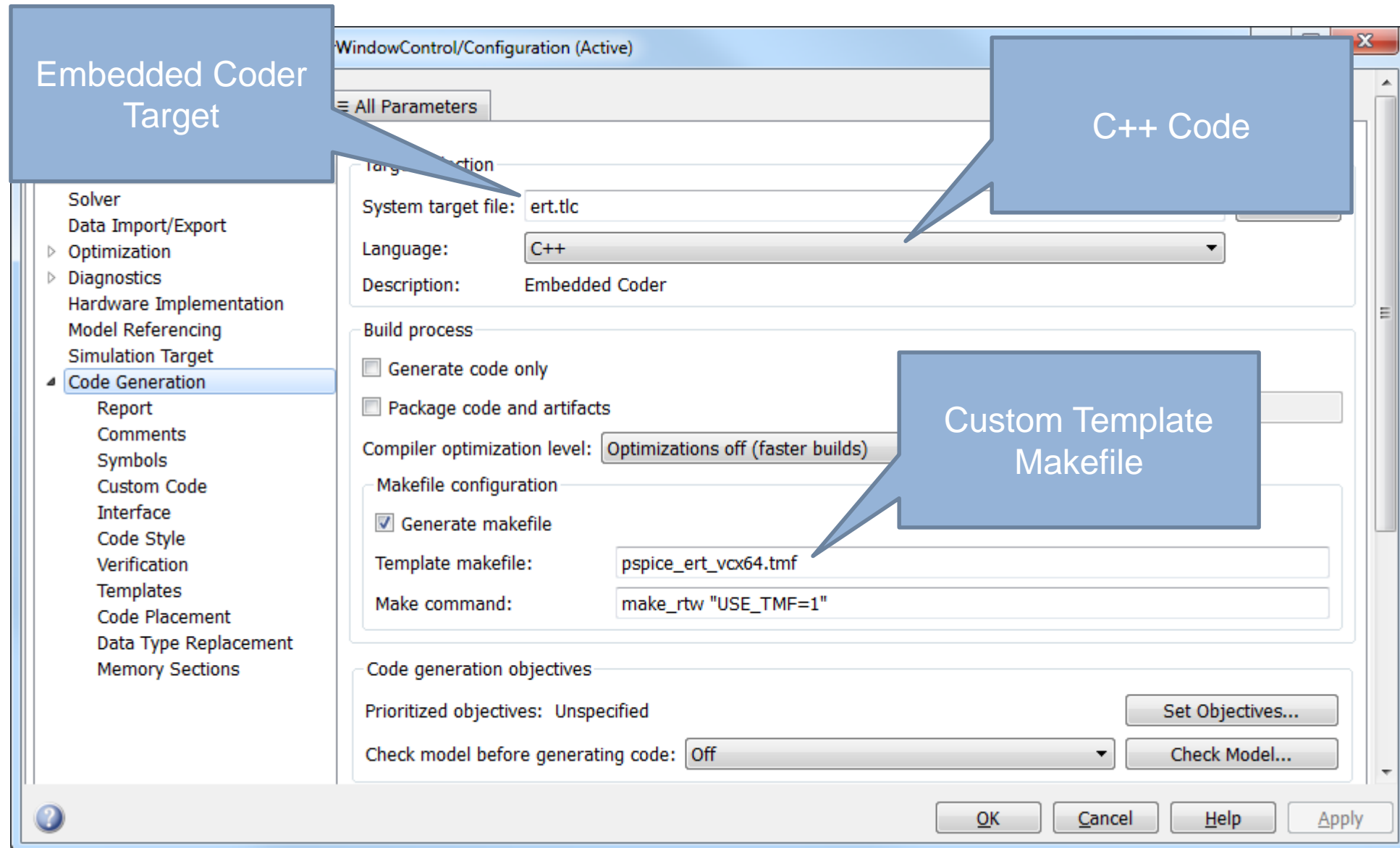
Simulink Model Diagram:

The main diagram shows a control system with inputs for 'endstop', 'driver', 'passenger', and 'obstacle'. The outputs are 'moveUp' and 'moveDown'. The diagram includes state transitions and logic for controlling a system.

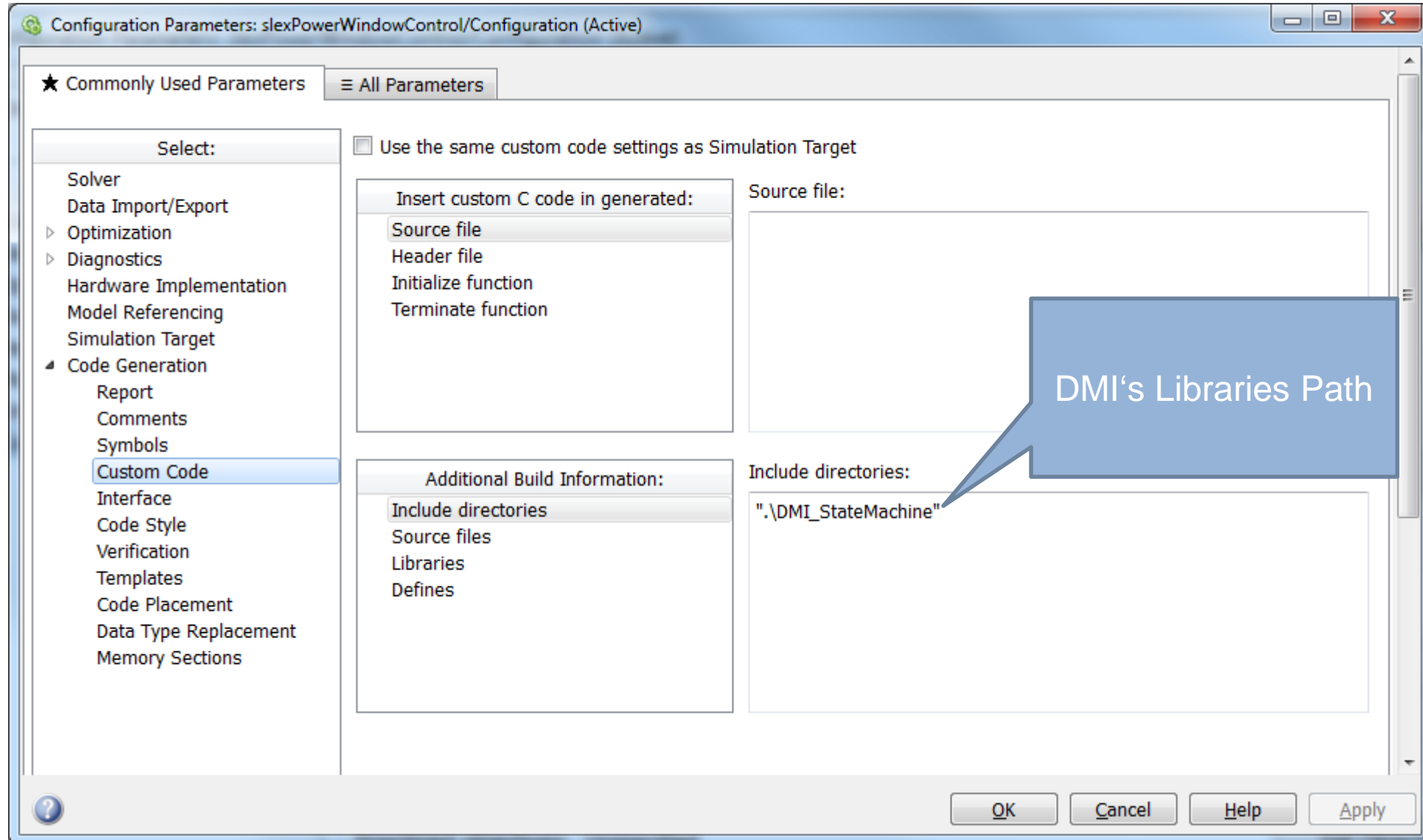
Control Block Diagram:

The 'control' block is a state machine diagram with inputs for 'endstop', 'driver', 'passenger', and 'obstacle'. The outputs are 'moveUp' and 'moveDown'. The diagram includes state transitions and logic for controlling a system.

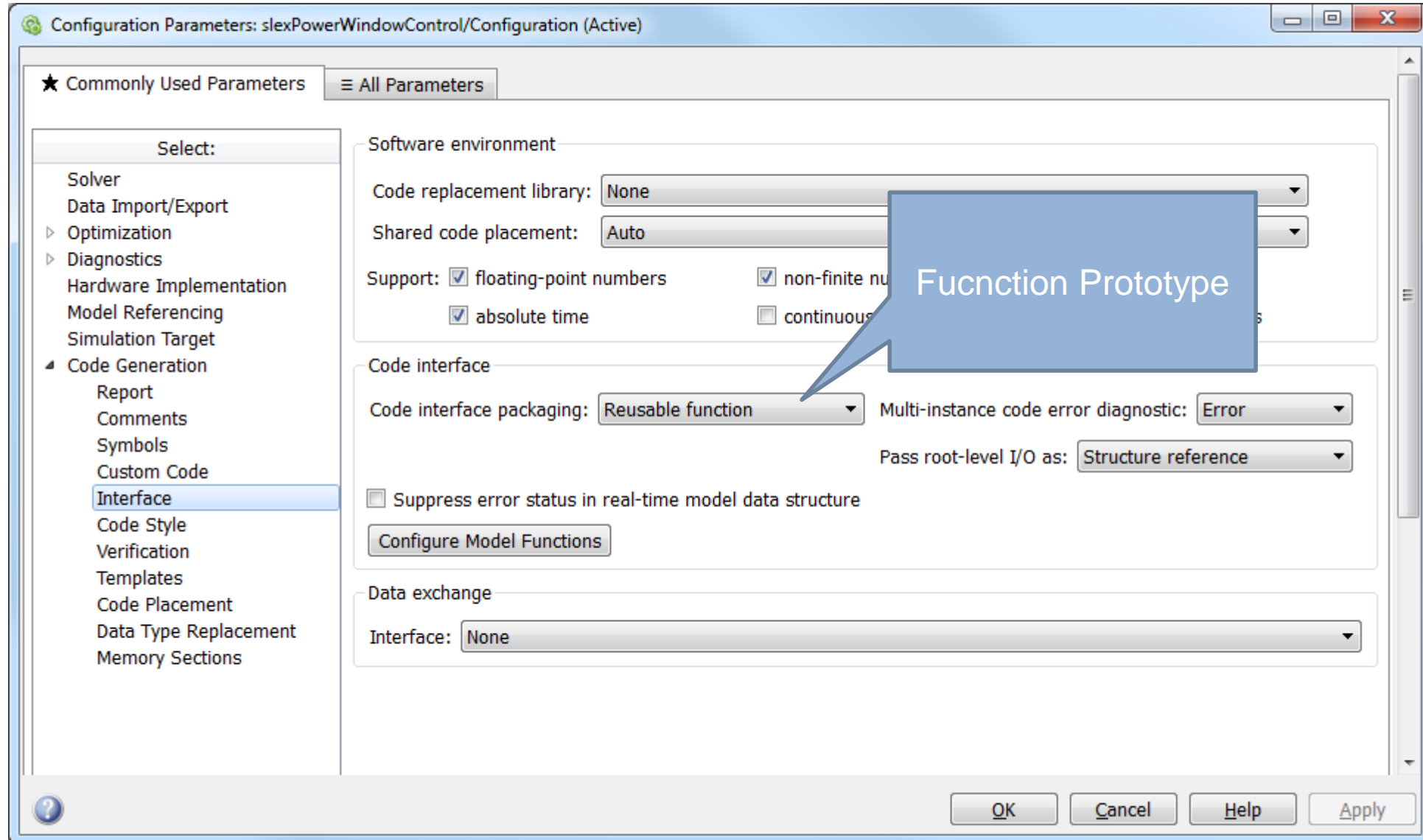
Simulink-PSpice Target Configuration – Code Generation



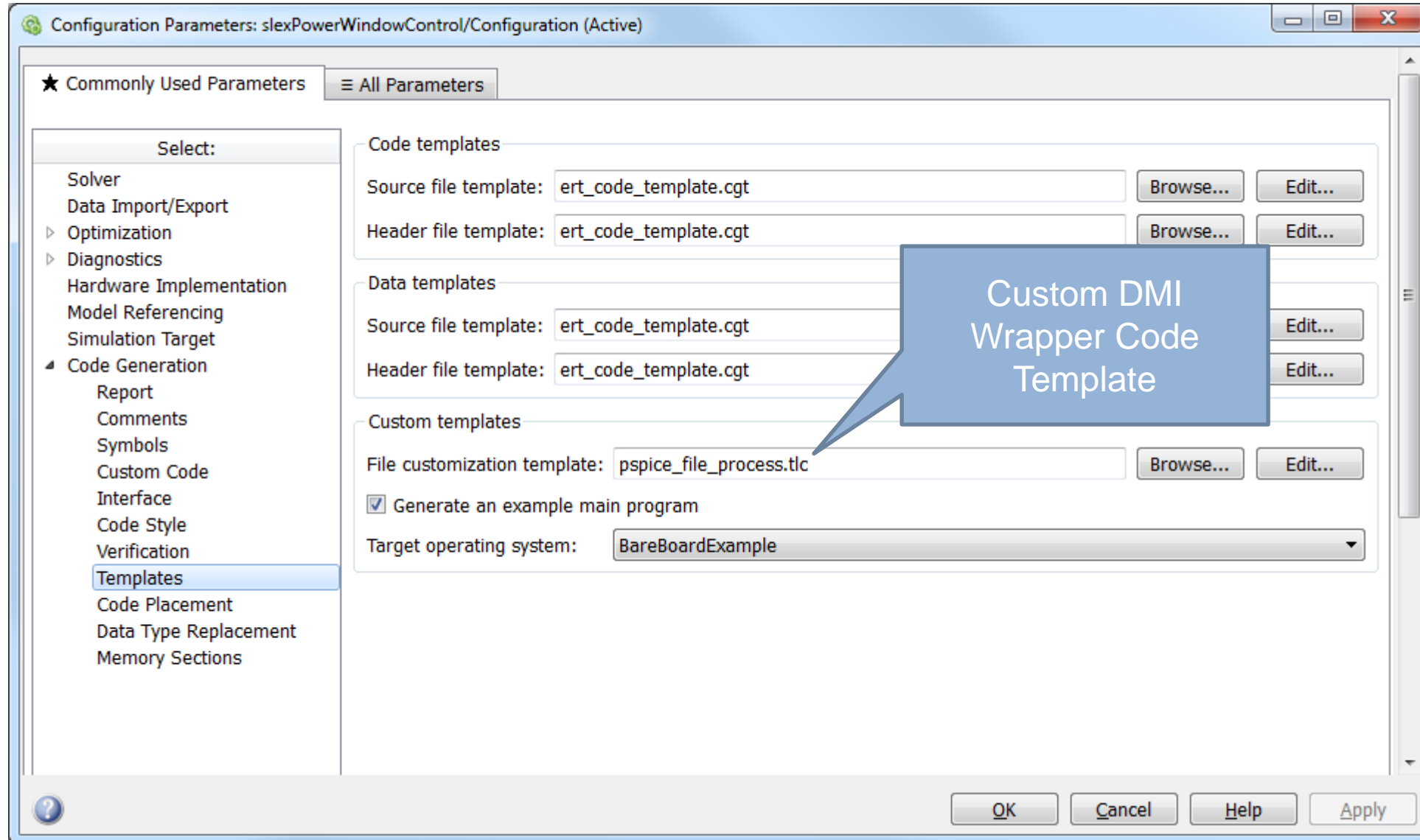
Simulink-PSpice Target Configuration – Custom Code



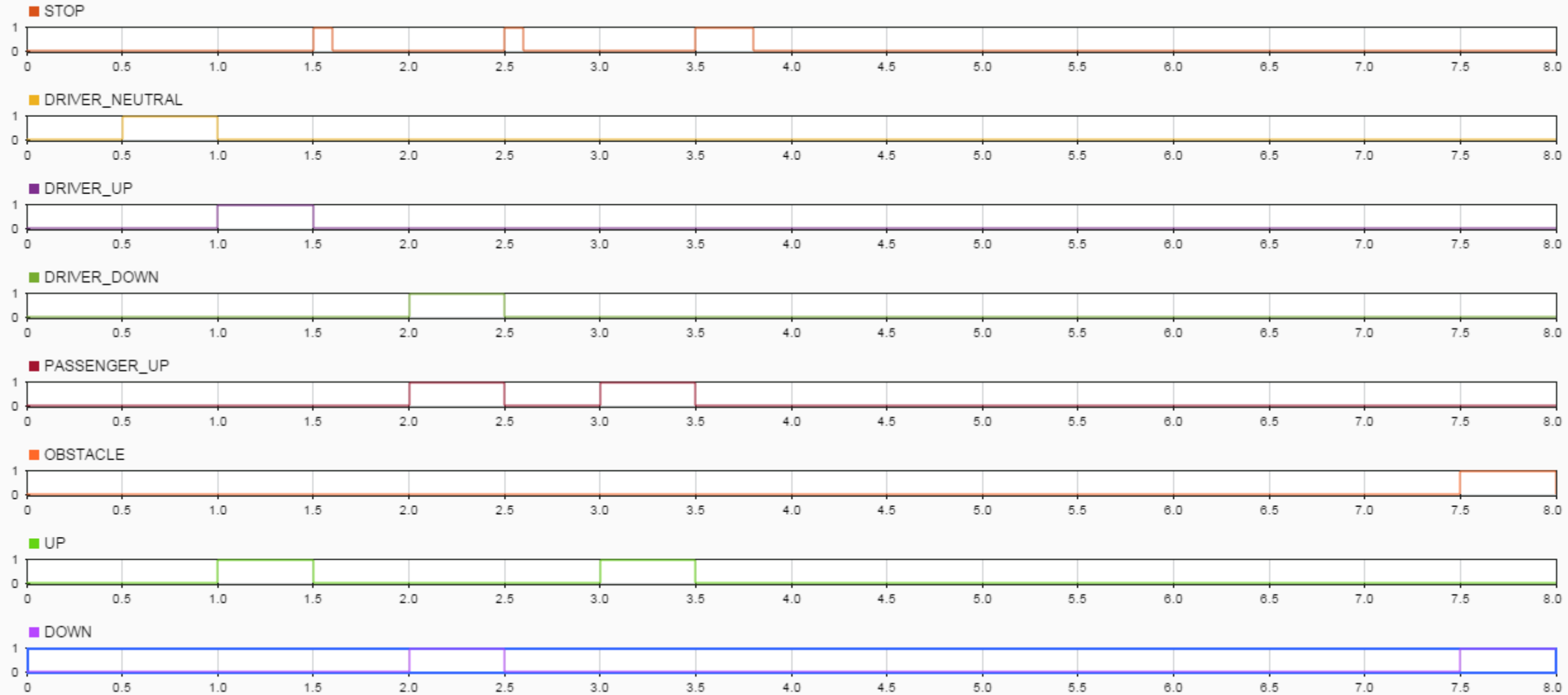
Simulink-PSpice Target Configuration – Interface



Simulink-PSpice Target Configuration – Templates



Simulink Simulation Results





PSpice DMI Library

The image shows two overlapping windows from the Cadence OrCAD suite. The background window is 'OrCAD Capture - [C:\Pilot Engineering\PSP_PSpice\Capture\Capture_Work\StateMachine.opj]', displaying a project hierarchy with folders like 'Design Resources', 'Library', and 'Outputs'. The foreground window is 'slexPowerWindowControl.lib:X_slexPowerWindowControl - PSpice Model Editor - [Model Text]', which contains a 'Models List' table and a text editor with Verilog-AMS code.

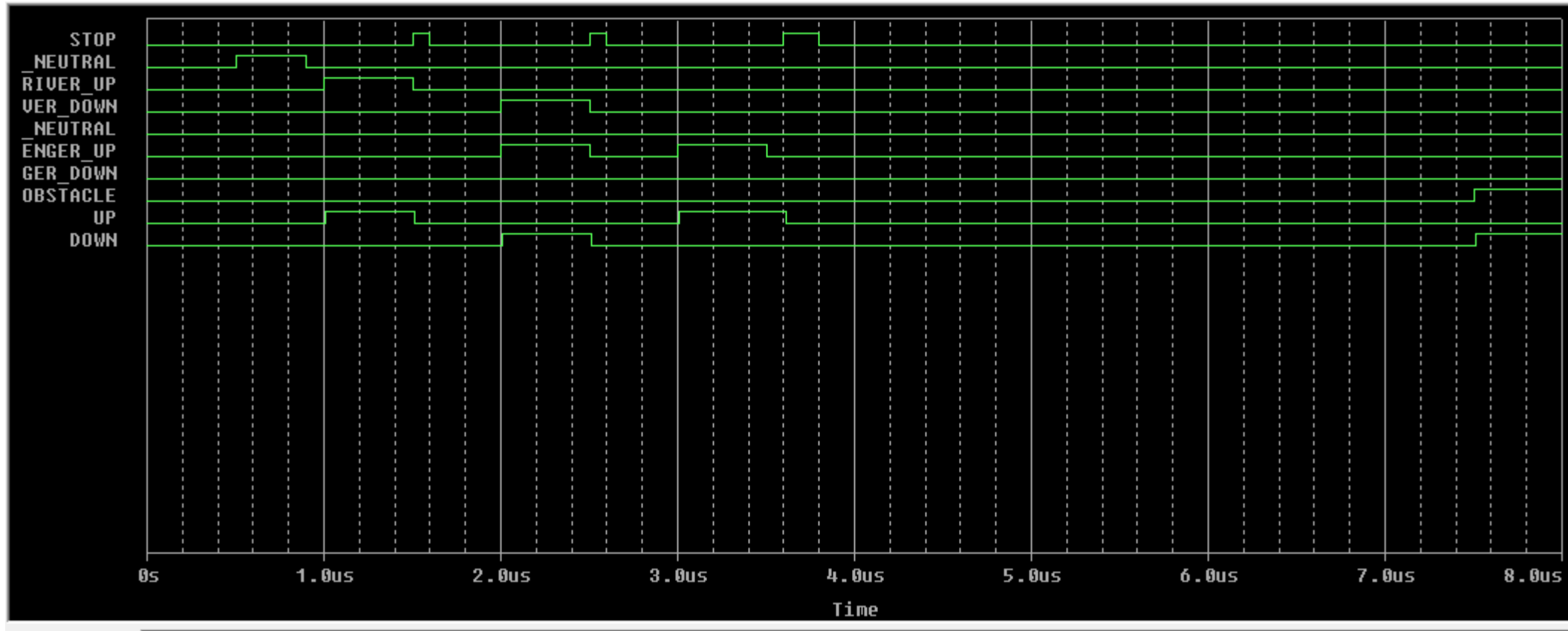
Model Name	Type
X_slexPowerWindowControl	SUBCKT

```
.subckt X_slexPowerWindowControl ENDSTOP DRIVER_0 DRIVER_1 DRIVER_2
PASSENGER_0 PASSENGER_1 PASSENGER_2 OBSTACLE MOVE_UP MOVE_DOWN
+ OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+ PARAMS:
.model slexPowerWindowControl_TIMING
+ tpltmn=6ns tpltty=9ns tpltmx=15ns
+ tpltmn=6ns tpltty=10ns tpltmx=15ns
+ )
U1 LOGICEXP( 8, 2 ) DPWR DGND
+ ENDSTOP DRIVER_0 DRIVER_1 DRIVER_2
OBSTACLE MOVE_UP MOVE_DOWN
+ slexPowerWindowControl_TIMING TO STD
+ C_MODEL: slexPowerWindowControl_win64.dll slexPowerWindowControl
+ PARAMS:
.ends
```

Generated by Embedded Coder

Extract new parameters from specifications

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

