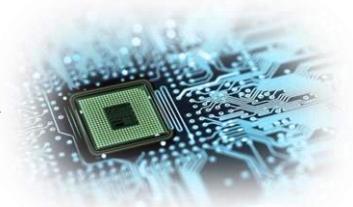
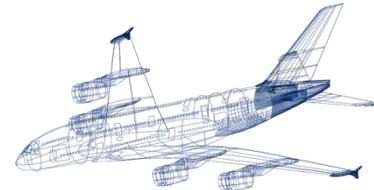
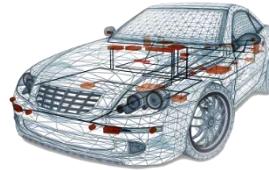


Tutorial: Bridging the Gap to and from Matlab/Simulink

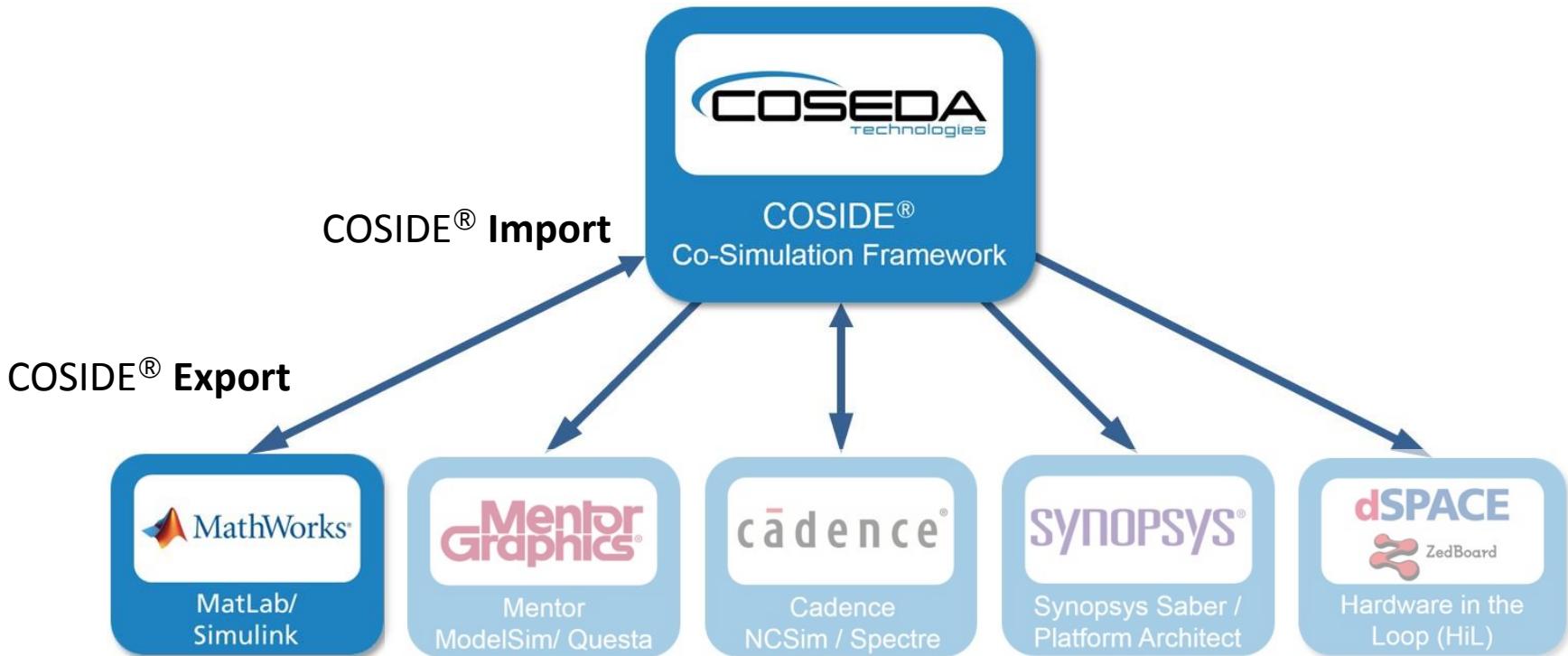


THE ANALOG AND DIGITAL SYSTEM LEVEL COMPANY



COSIDE® - Model Export - Import

Model Export and Import Features



COSIDE® - Model Export - Import

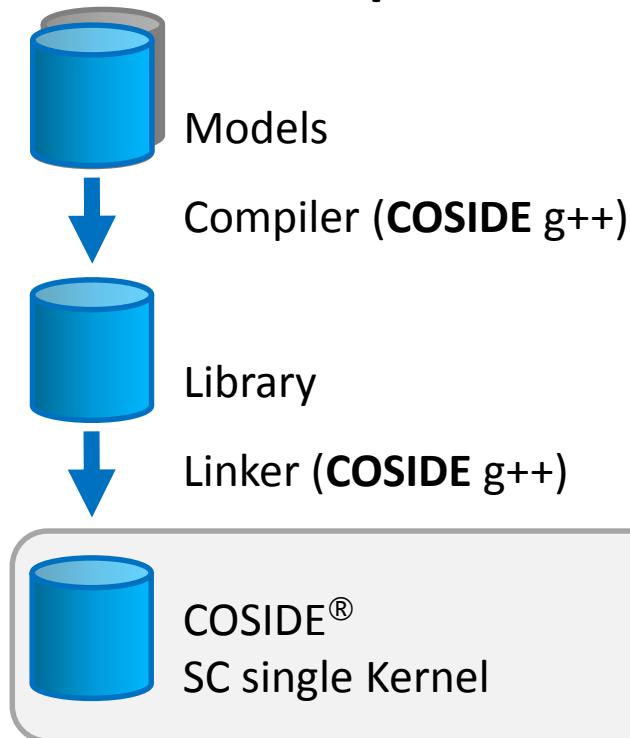
Agenda

1. COSIDE® Import
2. COSIDE® Export
3. COSDIE® Interaction
4. Summary

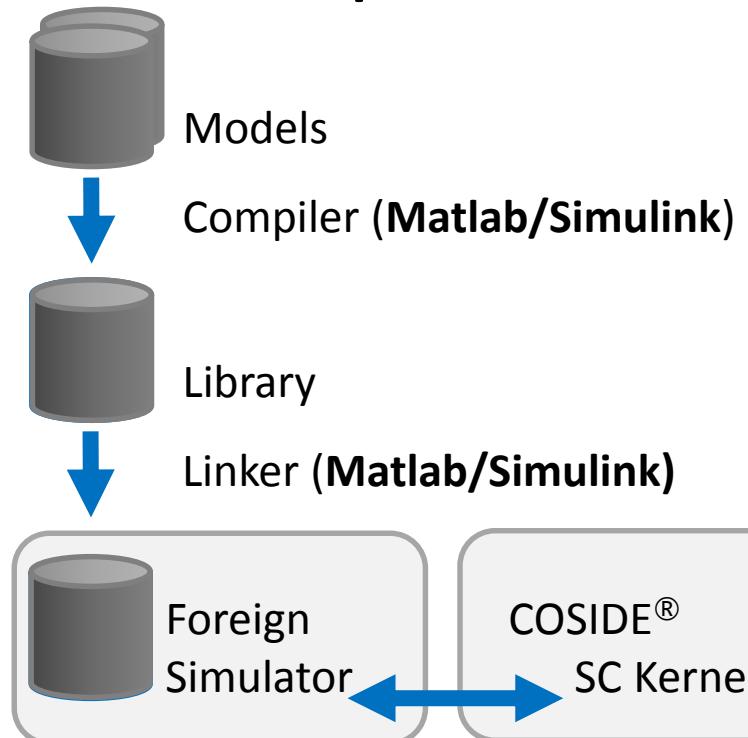
COSIDE® - Model Export - Import

General Overview: White vs. Black Box Import

COSIDE® White Box Import



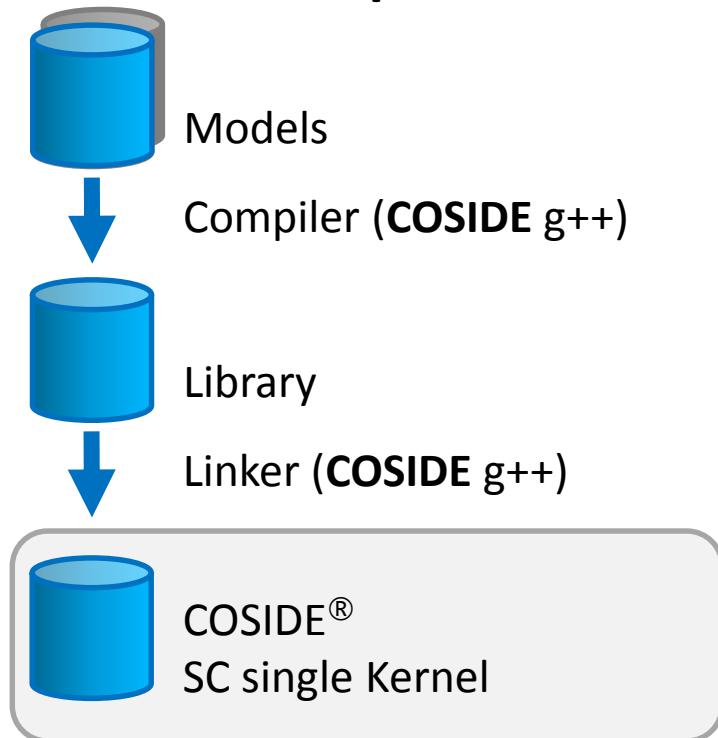
COSIDE® Black Box Import



COSIDE® - Model Export - Import

White Box Import – Simulink Coder – overview

COSIDE® White Box Import

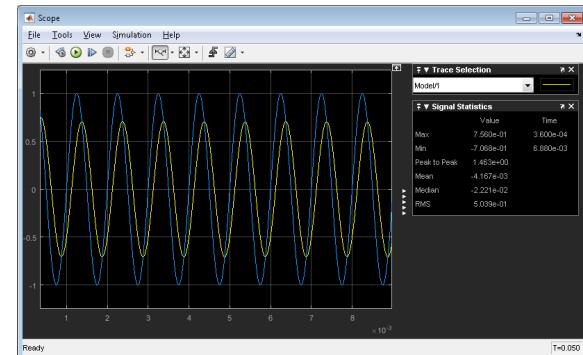
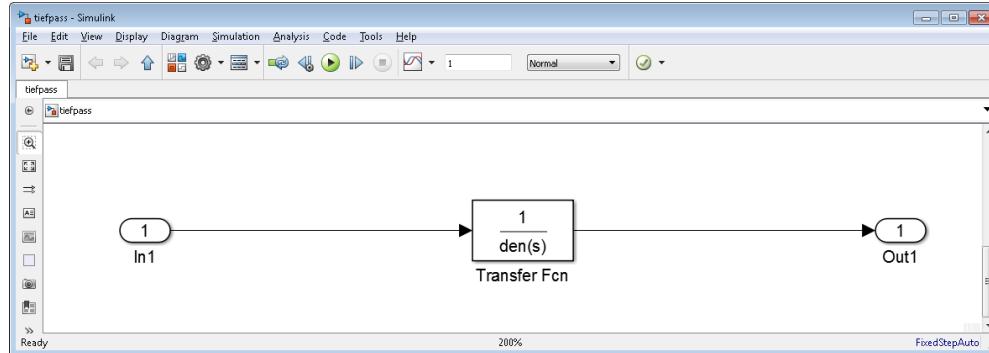


- Simulink coder is used to generate C-Code
- COSIDE® generates wrapper modules
- Wrapped code is included into COSIDE's normal makeflow
- COSIDE® stimulates and runs the wrapped Simulink code

COSIDE® - Model Export - Import

White Box Import – Simulink Coder – C-Code Generation

- Select Matlab Subsystem to be exported



- Generate C Code from your Simulink model (using Simulink Coder)

```
1 // File: tiepass.cpp
2 // Code generated for Simulink model 'tiefpass'.
3 //
4 // Model version          : 1.0
5 // Simulink Coder version : 0.9 (R2015b) 13-Aug-2016
6 // C/C++ source code generated on / Fri Jun 17 14:58:40 2016
7 //
8 // Target selection: crt.cio
9 // Embedded hardware selection: Intel-i386-64 (Windows64)
10 // Code generation options: 
11 //   1. Execution efficiency
12 //   2. RAM efficiency
13 // Validation results: Not run
14 //
15 // System target file: 
16 // Processor: 
17 // Architecture: 
18 // Function: #include "tiefpass.h"
19 // Function: #include "cetnsmp.h"
20 // Function: #include "tiefpass_COMMON_INCLUDES_
21 // Function: #include "tiefpass.h"
22 // Function: #include "tiefpass_COMMON_INCLUDES_
23 // Function: #include "tiefpass.h"
24 // Function: #include "tiefpass.h"
25 // Function: #include "tiefpass.h"
26 // Function: #include "tiefpass.h"
27 // Macros for accessing real-time model data structure
28 #ifndef _TIEPASS_CRTDATAEXTRAS_H
29 #define _TIEPASS_CRTDATAEXTRAS_H
30 
```

```
1 // Class declarations for model tiepass
2 class tiepassModelData;
3 public:
4     static tiepassModelData* Create();
5     void Initialize();
6     void Update();
7     void Stop();
8     void Start();
9     void SetTimeStep(double dt);
10    void SetTime(double t);
11    void SetTimeRate(double rate);
12    void SetTimeMode(TIME_MODE mode);
13    void SetTimeStepMode(TIME_MODE mode);
14    void SetTimeRateMode(TIME_MODE mode);
15    void SetTimeStepRateMode(TIME_MODE mode);
16    void SetTimeStepRateRateMode(TIME_MODE mode);
17    void SetTimeStepRateRateRateMode(TIME_MODE mode);
18    void SetTimeStepRateRateRateRateMode(TIME_MODE mode);
19    void SetTimeStepRateRateRateRateRateMode(TIME_MODE mode);
20    void SetTimeStepRateRateRateRateRateRateMode(TIME_MODE mode);
21    void SetTimeStepRateRateRateRateRateRateRateMode(TIME_MODE mode);
22    void SetTimeStepRateRateRateRateRateRateRateRateMode(TIME_MODE mode);
23    void SetTimeStepRateRateRateRateRateRateRateRateRateMode(TIME_MODE mode);
24    void SetTimeStepRateRateRateRateRateRateRateRateRateRateMode(TIME_MODE mode);
25    void SetTimeStepRateRateRateRateRateRateRateRateRateRateRateMode(TIME_MODE mode);
26    void SetTimeStepRateRateRateRateRateRateRateRateRateRateRateRateMode(TIME_MODE mode);
27    void SetTimeStepRateRateRateRateRateRateRateRateRateRateRateRateRateMode(TIME_MODE mode);
28    void SetTimeStepRateRateRateRateRateRateRateRateRateRateRateRateRateRateMode(TIME_MODE mode);
29 
```

COSIDE® - Model Export - Import

White Box Import – Simulink Coder – exposed Interface

```
Editor - C:\ehrlich\fhg_coder\Paul\tiefpass_ert_rtw\tiefpass.h
EDITOR      VIEW
FILE        NAMIGATE    EDIT    BREAKPOINTS
New Open Save Compare Go To Comment % Find Indent Breakpoints
+-----+
tiefpass.cpp x tiefpass.h x +
155
156 // Class declaration for model tiefpass
157 class tiefpassModelClass {
158     // public data and function members
159     public:
160         // External inputs
161         ExtU rtU;           Inputs
162
163         // External outputs
164         ExtY rtY;          Outputs
165
166     // Model entry point functions
167
168     // model initialize function
169     void initialize();
170
171     // model step function
172     void step();          Step function
173
174     // Constructor
175     tiefpassModelClass();
176
177     // Destructor
178     ~tiefpassModelClass();
179
180     // Real-Time Model get method
181     RT_MODEL * getRTM();
182
183     // private data and function members
184     private:
185         X rtX;              // Block continuous states
`--
```

C / CPP source or header file | Ln 16 Col 3

COSIDE® - Model Export - Import

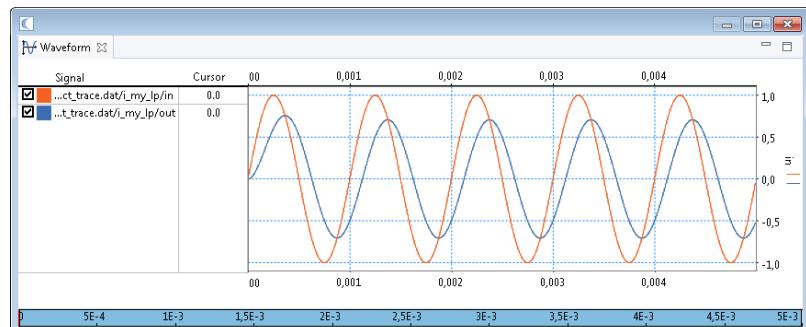
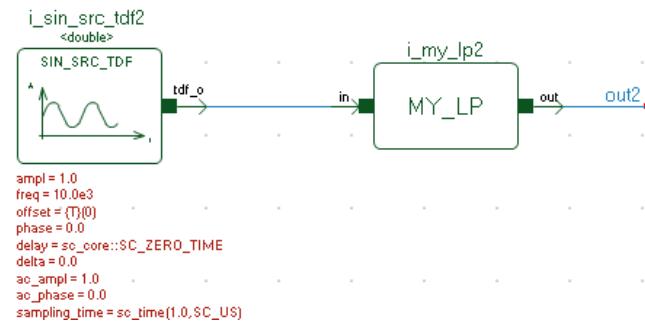
White Box Import – Simulink Coder - Use of generated C Code

- COSIDE® generates wrapper for exported Code



```
61 //////////////////////////////////////////////////////////////////
62 // method processing
63 //////////////////////////////////////////////////////////////////
64
65 void my_lp::processing()
66 {
67     s.rtObj.rtU.In1 = in;
68     rt_oneStep();
69     out = s.rtObj.rtY.Out1;
70
71 }
```

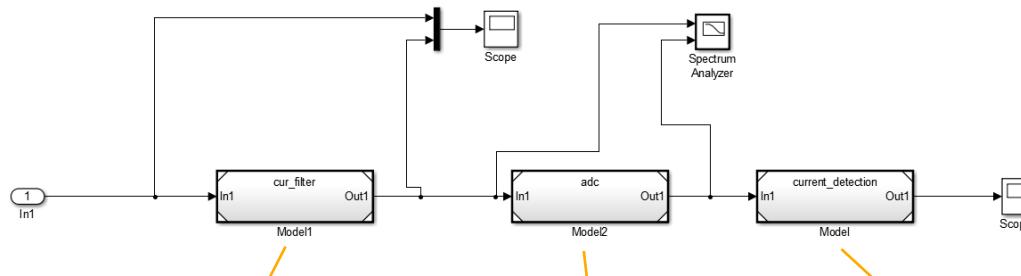
- Code is simulated within COSIDE® context



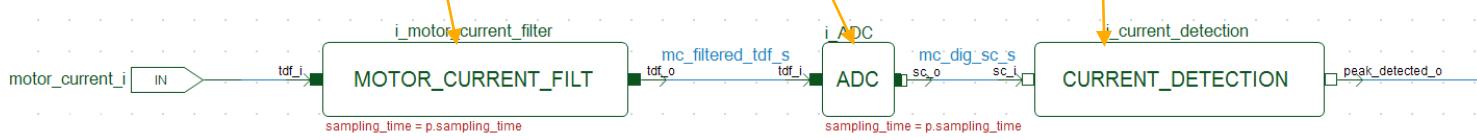
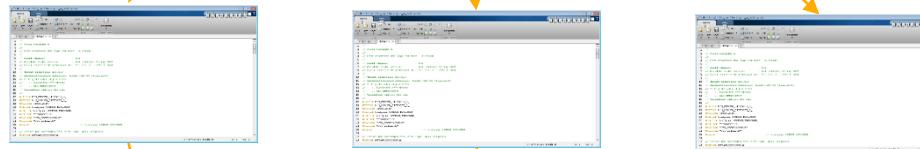
COSIDE® - Model Export - Import

White Box Import – Simulink Coder – multiple blocks

- COSIDE® wrapper modules for multiple Simulink blocks



C-Code from
Simulink Coder

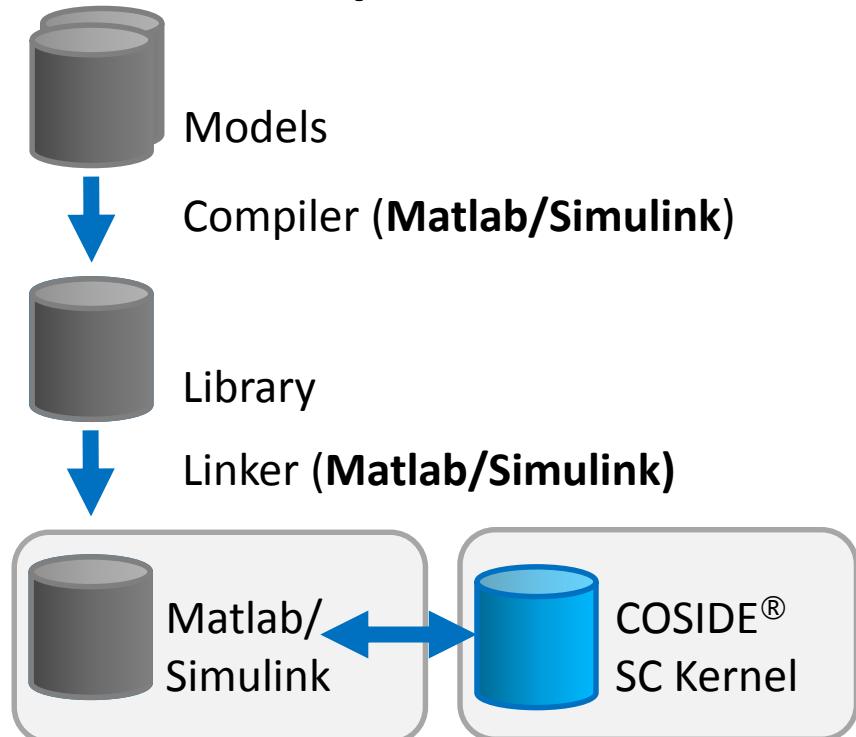


COSIDE® - Model Export - Import

Black Box Import - Simulink Model Import principles

- COSIDE® starts Matlab during testbench execution
- Matlab runs the Simulink/ Matlab part of the model, therefore it has to run during simulation (license required)
- Both share data via Shared Memory or Socket connection

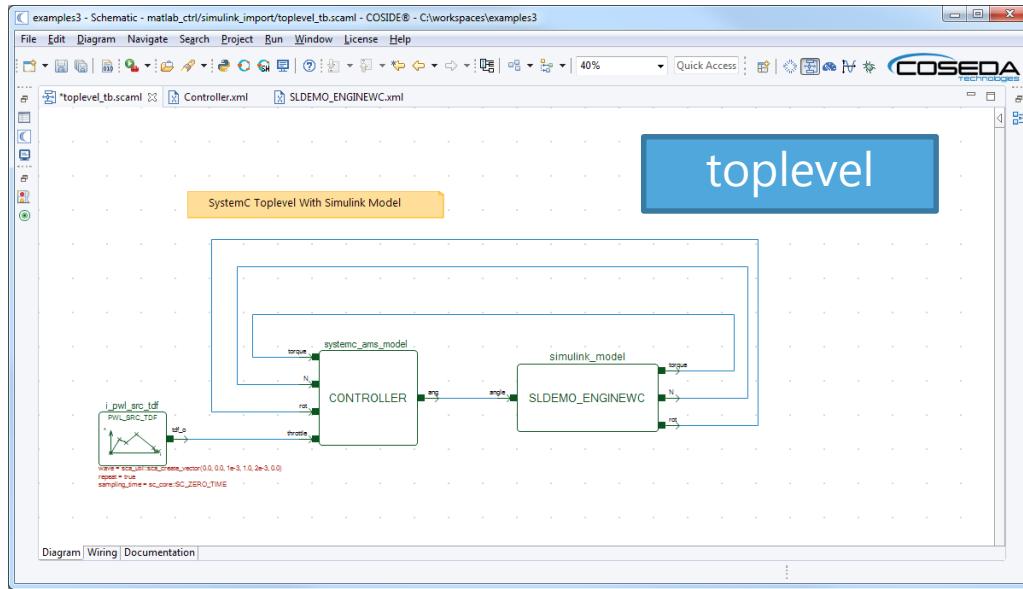
COSIDE® Black Box Import



COSIDE® - Model Export - Import

Black Box Import – CO-Simulate SystemC AMS and Simulink Model

- Create Testbench and run SystemC model as usual



- In the background Matlab/Simulink is started running the Simulink part of the model

COSIDE® - Model Export - Import

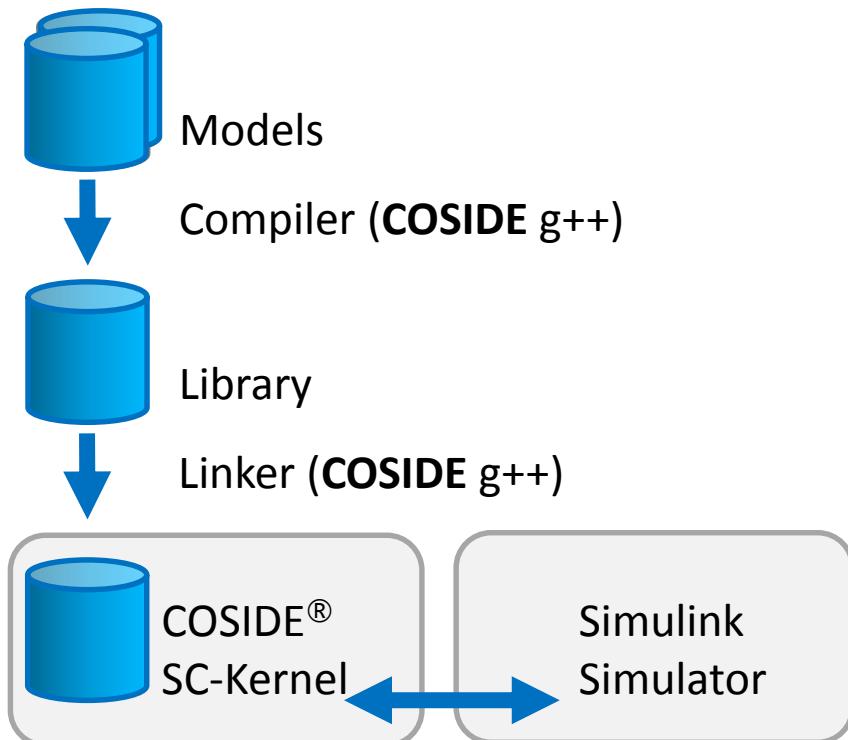
Agenda

1. COSIDE® Import
2. **COSIDE® Export**
3. COSDIE® Interaction
4. Summary

COSIDE® - Model Export - Import

Black Box Export Matlab

Black Box Export

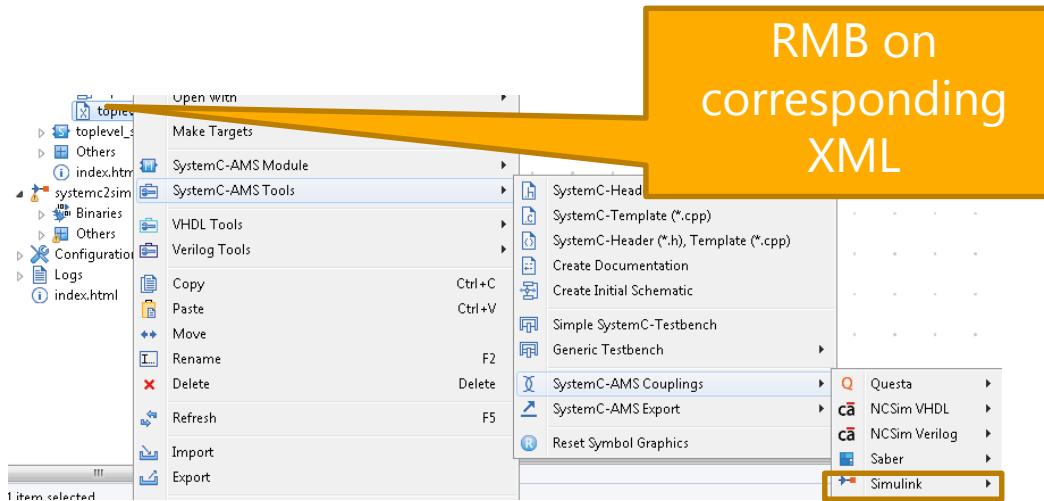
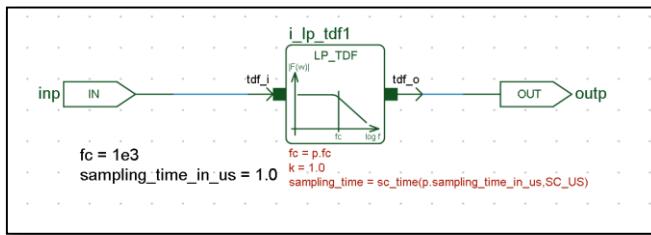


- COSIDE® compiles SystemC AMS modules to an shared object
- Additionally a wrapper (.mex file) loads generated to load it into the Simulink simulation
- Communication is possible via: Shared Memory, Shared Object or TCP/IP

COSIDE® - Model Export - Import

Black Box Export Matlab - Setup

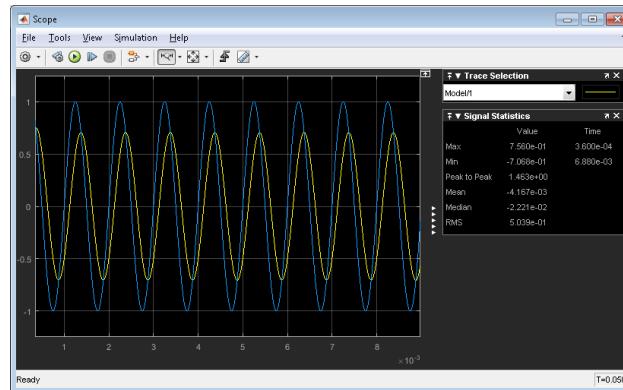
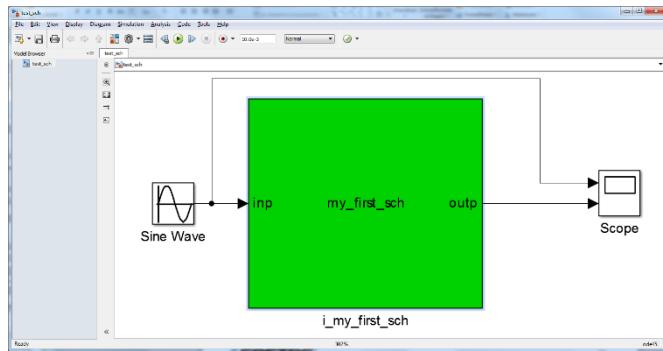
- Single step export of COSIDE® subsystem



COSIDE® - Model Export - Import

Black Box Export Matlab - Simulation

- Run your Matlab simulation as usual



- SystemC kernel outputs are directed to Matlab console

The diagram shows a Matlab Command Window titled 'D:\coside\projects\verification_ws\getting_started_example\systemc2simulink_if'. The window displays the startup logs for the COSIDE framework, SystemC, and SystemC AMS extensions. Key log entries include:

```
INFO: COSIDE_COUPLING_FRAMEWORK: basename of SystemC model: 'test_sch/i_my_first_sch'  
INFO: In module: test_sch/i_my_first_sch  
INFO: Set SystemC time resolution to: 1 ps  
INFO: SC-PARAM: fc = 1000 (sysc default: 1000)  
INFO: SystemC AMS extensions 2.0 Version: 2.0_beta2 --- BuildRevision: 1808 20140713  
INFO: Copyright (c) 2010-2014 by Fraunhofer-Gesellschaft  
INFO: Institut Integrated Circuits / EAS  
INFO: Licensed under the Apache License, Version 2.0
```

COSIDE® - Model Export - Import

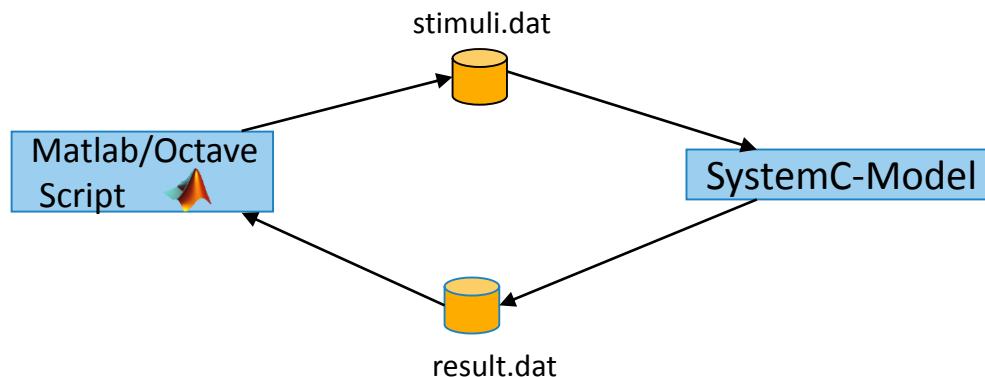
Agenda

1. COSIDE® Import
2. COSIDE® Export
3. **COSDIE® Interaction**
4. Summary

COSIDE® - Model Export - Import

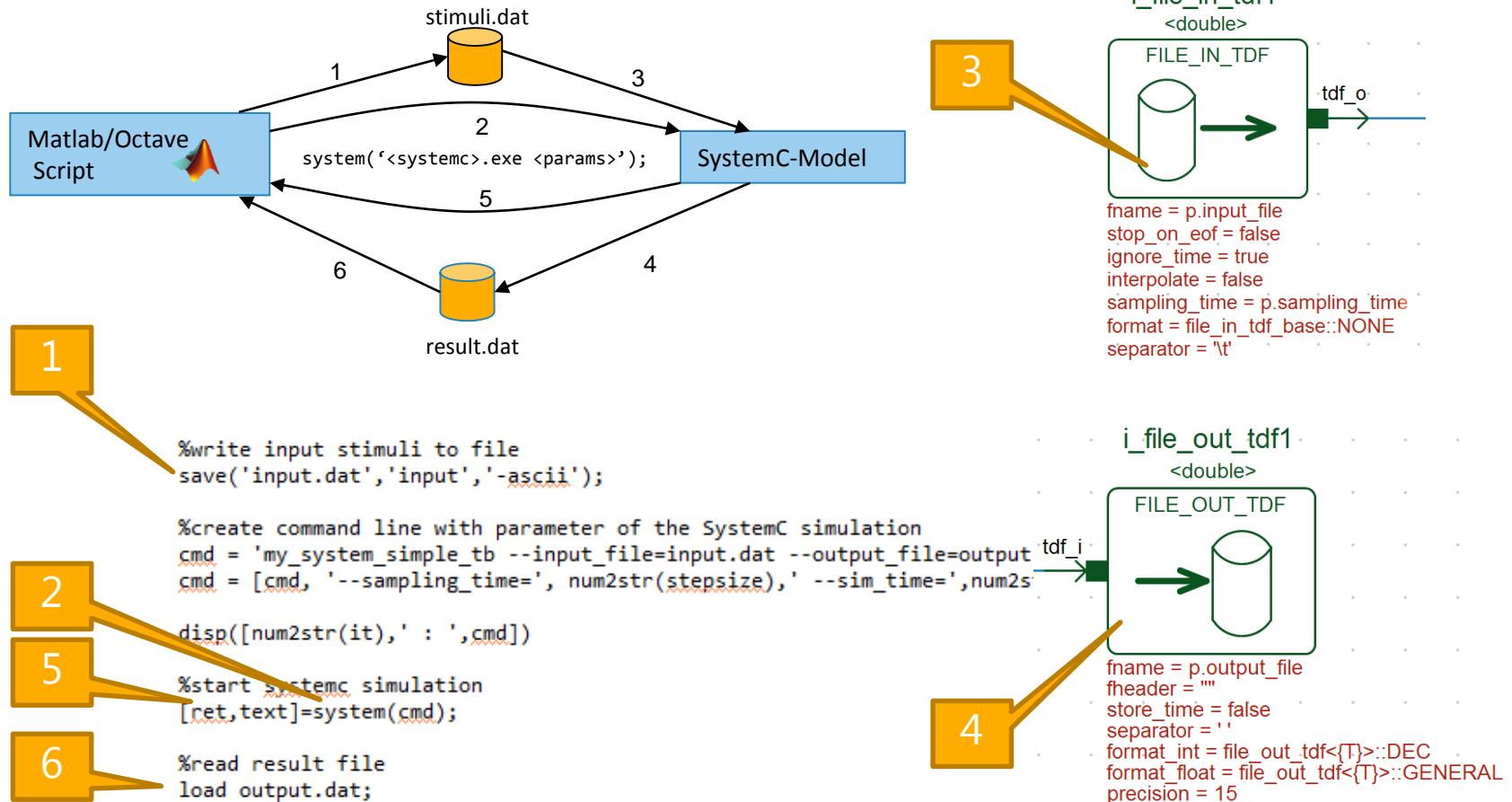
Interaction - Matlab optimization runs and regression testing

- COSIDE® exchanges data via files with Matlab/Simulink and therefore can:
 - Generate input files (Stimuli.dat)
 - Start the simulation of the SystemC model
 - Perform post-processing after reading back of the results (result.dat)



COSIDE® - Model Export - Import

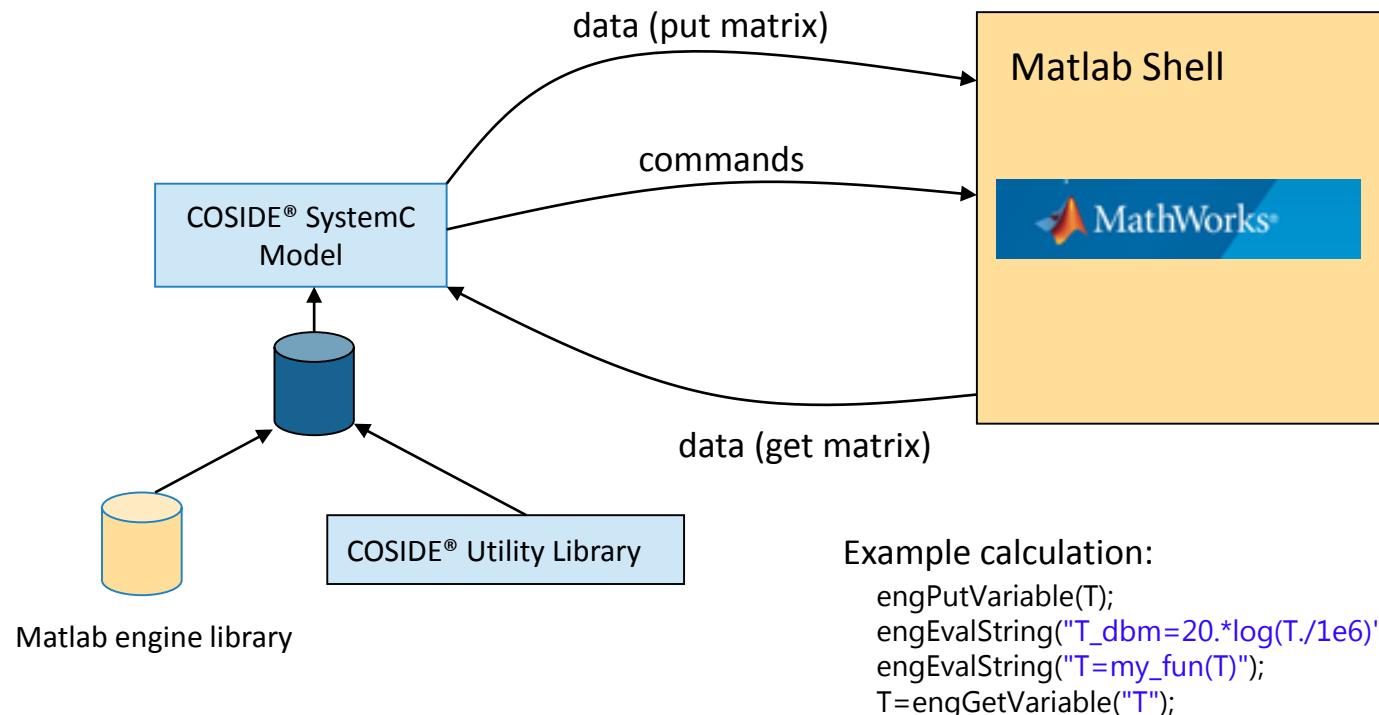
Interaction - Matlab optimization runs and regression testing - steps



COSIDE® - Model Export - Import

Interaction - Calling Matlab Function from SystemC Models

- Matlab provides the Engine API in C/C++ for interacting with a Matlab shell via IPC
- COSIDE® provides easy to use utility functions and examples



COSIDE® - Model Export - Import

Agenda

1. COSIDE® Import
2. COSIDE® Export
3. COSDIE® Interaction
4. Summary

COSIDE® - Model Export - Import

Ways to close the Gap

1. White Box Import – Import C-Code from Simulink coder into COSIDE®
 - Reuse legacy Simulink modules
2. Black Box Import – Simulink is started and connected during COSIDE® run
 - Reuse legacy Simulink modules not capable of being exported with Coder
3. Black Box Export – Shared object to run SystemC AMS modules within Matlab
 - IP Protected customer export

COSIDE® - Model Export - Import

Ways to close the Gap

4. Matlab Interaction – exchange data with Matlab via files (e.g. import stimuli or export for post processing)
 - Regression testing and optimization runs
5. Matlab Interaction – controlling Matlab from SystemC using the engine API of Matlab
 - Reuse of complex Matlab functions