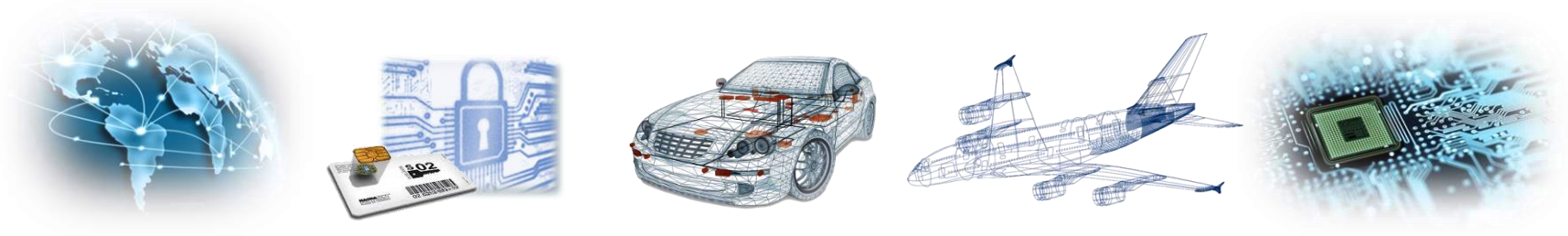
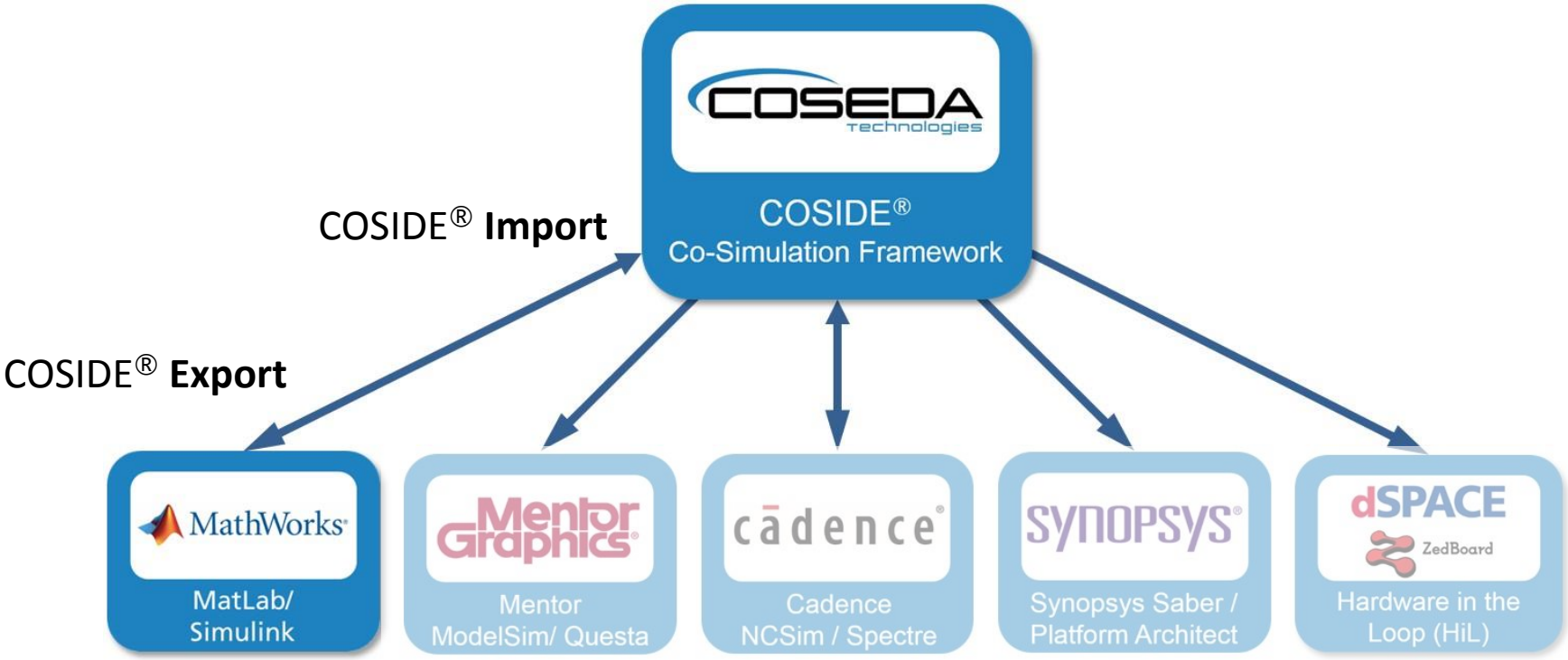
Tutorial: Bridging the Gap to and from Matlab/Simulink

THE ANALOG AND DIGITAL SYSTEM LEVEL COMPANY



COSIDE® - Model Export - Import

Model Export and Import Features



COSIDE® - Model Export - Import

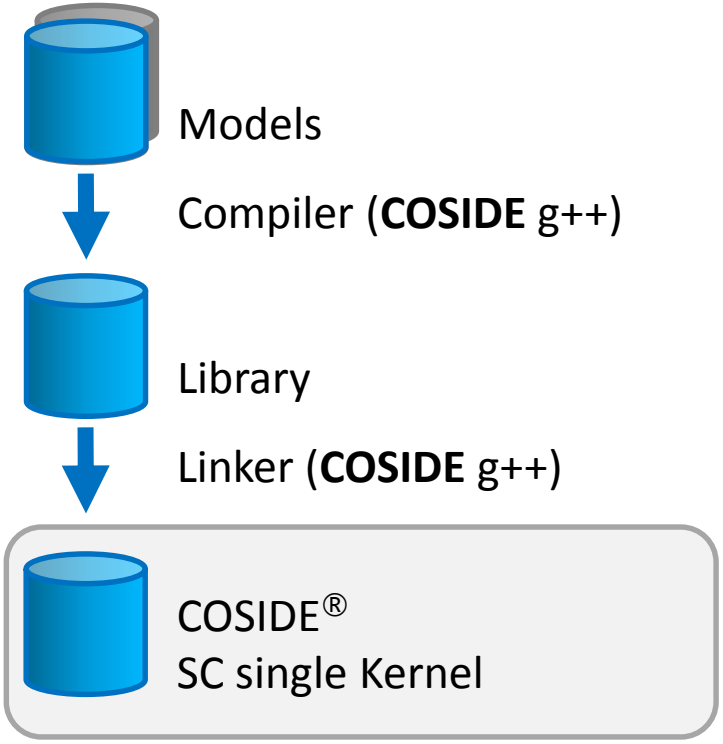
Agenda

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

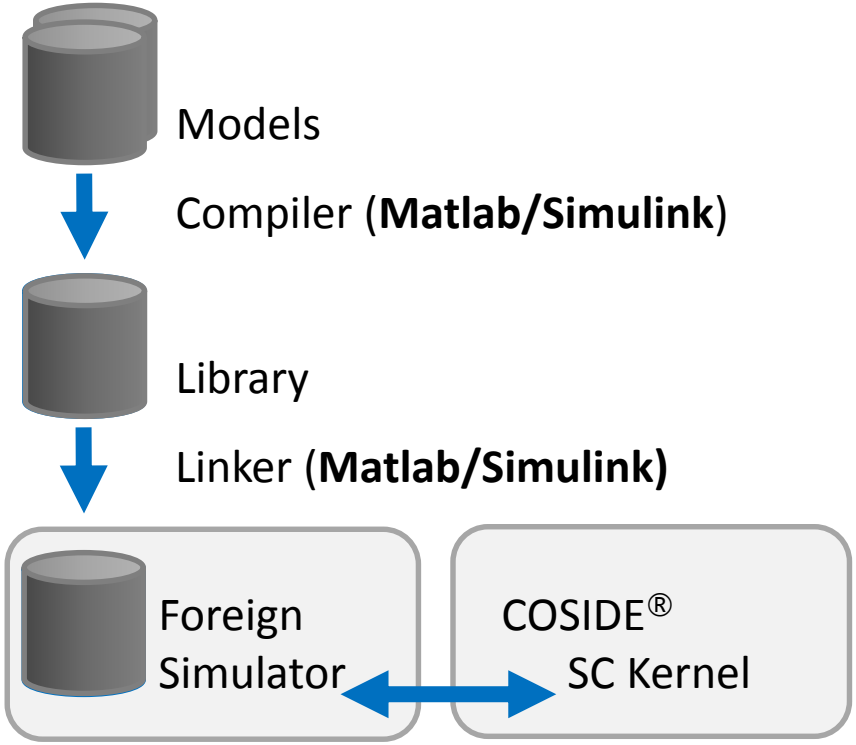
COSIDE® - Model Export - Import

General Overview: White vs. Black Box Import

COSIDE®
White Box Import



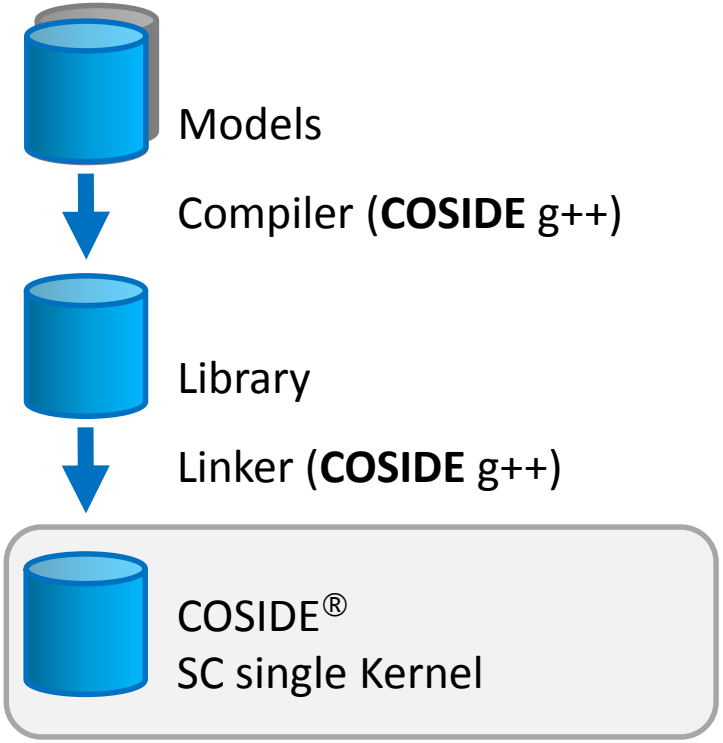
COSIDE®
Black Box Import



COSIDE® - Model Export - Import

White Box Import – Simulink Coder – overview

COSIDE®
White Box Import

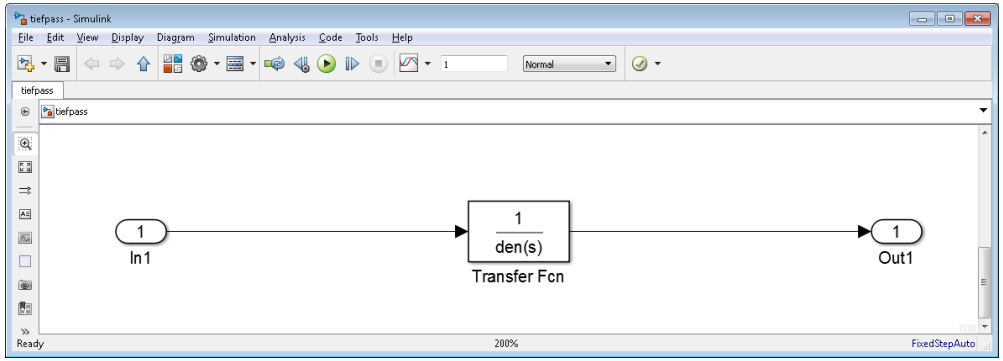


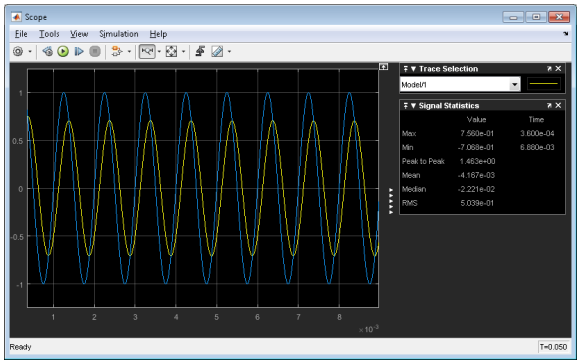
- Simulink coder is used to generate C-Code
- COSIDE® generates wrapper modules
- Wrapped code is included into COSIDE's normal makeflow
- COSIDE® stimulates and runs the wrapped Simulink code

COSIDE® - Model Export - Import

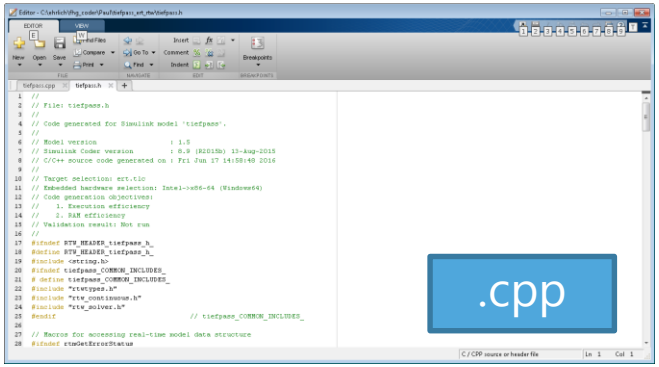
White Box Import – Simulink Coder – C-Code Generation

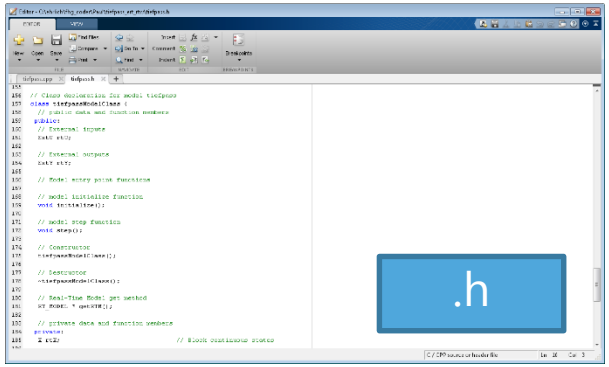
- Select Matlab Subsystem to be exported





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





COSIDE® - Model Export - Import

White Box Import – Simulink Coder – exposed Interface

Editor - C:\ehrlich\fhg_coder\Paul\tiefpass_ert_rtw\tiefpass.h

EDITOR | VIEW

New | Open | Save | Find Files | Compare | Print | Go To | Find | Insert | Comment | Indent | Breakpoints

FILE | NAVIGATE | EDIT | BREAKPOINTS

tiefpass.cpp | tiefpass.h

```
155
156  // Class declaration for model tiefpass
157  class tiefpassModelClass {
158    // public data and function members
159   public:
160    // External inputs
161    ExtU rtU;
162
163    // External outputs
164    ExtY rtY;
165
166    // Model entry point functions
167
168    // model initialize function
169    void initialize();
170
171    // model step function
172    void step();
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
```

Inputs

Outputs

Step function

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

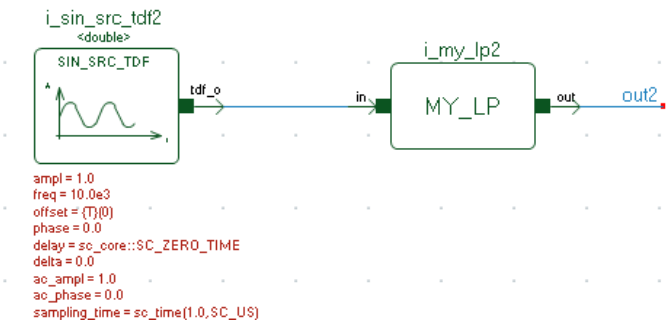
COSIDE® - Model Export - Import

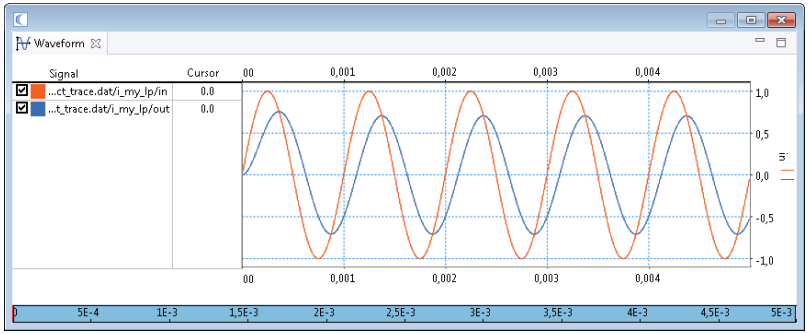
White Box Import – Simulink Coder - Use of generated C Code

- COSIDE® generates wrapper for exported Code


```
82 ////////////////////////////////////////////////
83 // method processing                          //
84 ////////////////////////////////////////////////
85 void my_lp::processing()
86 {
87     s.rtObj.rtU.In1 = in;
88     rt_oneStep();
89     out = s.rtObj.rtY.Out1;
90
91 }
```

- Code is simulated within COSIDE® context

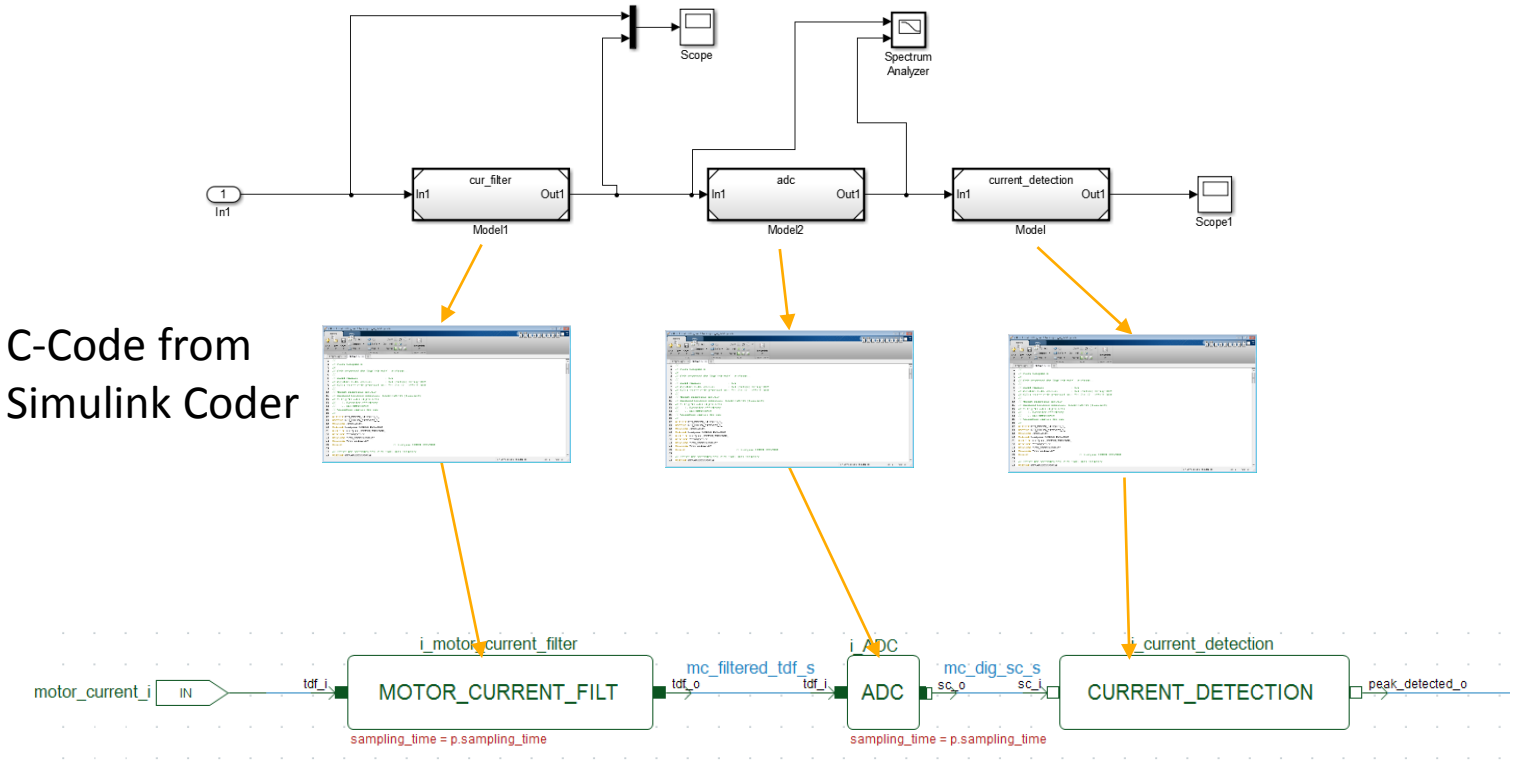




COSIDE® - Model Export - Import

White Box Import – Simulink Coder – multiple blocks

- COSIDE® wrapper modules for multiple Simulink blocks

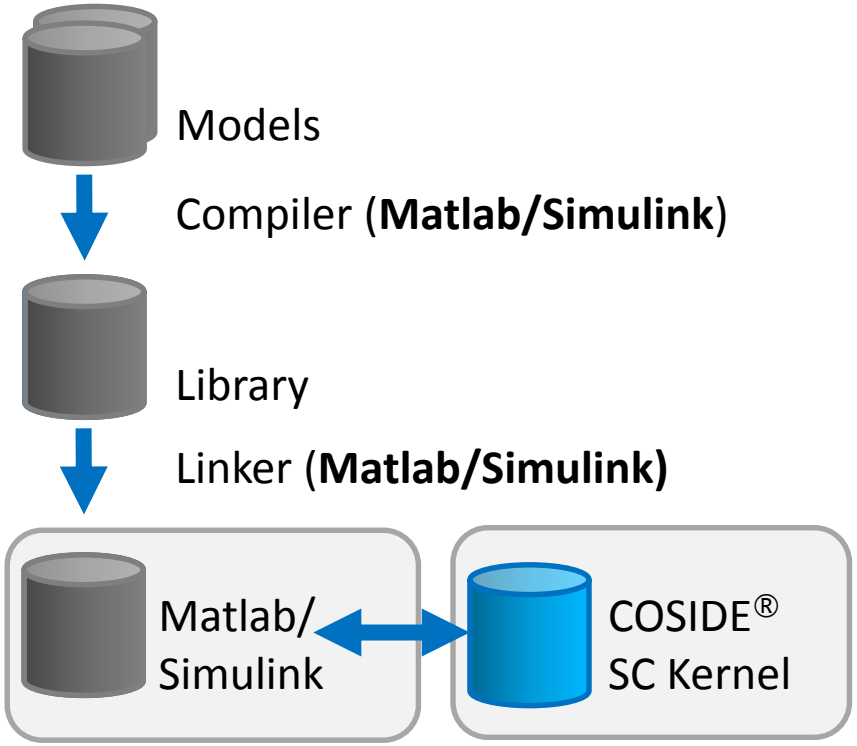


COSIDE® - Model Export - Import

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

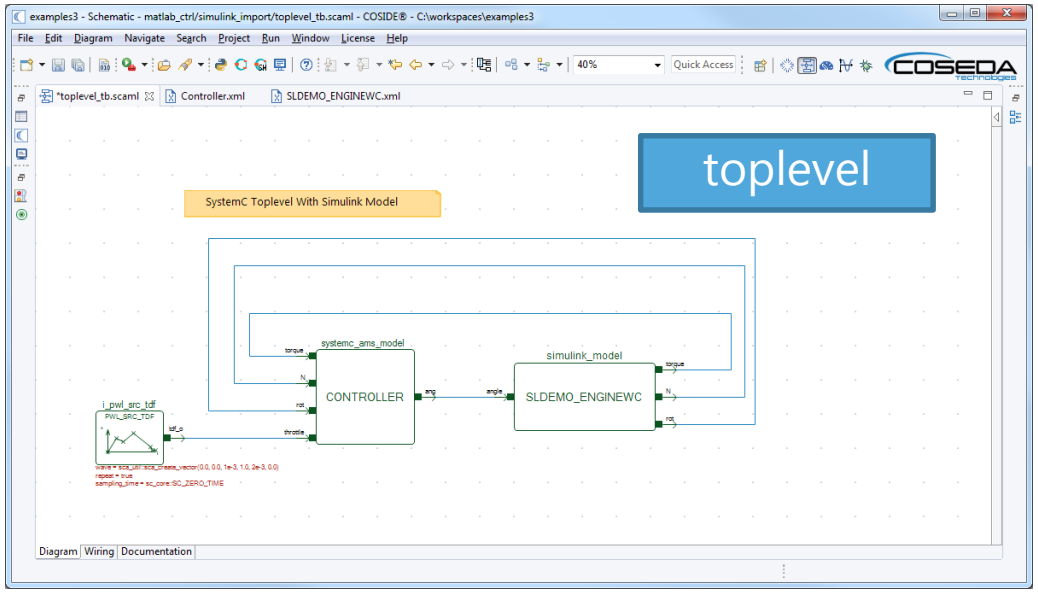
COSIDE®
Black Box Import



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

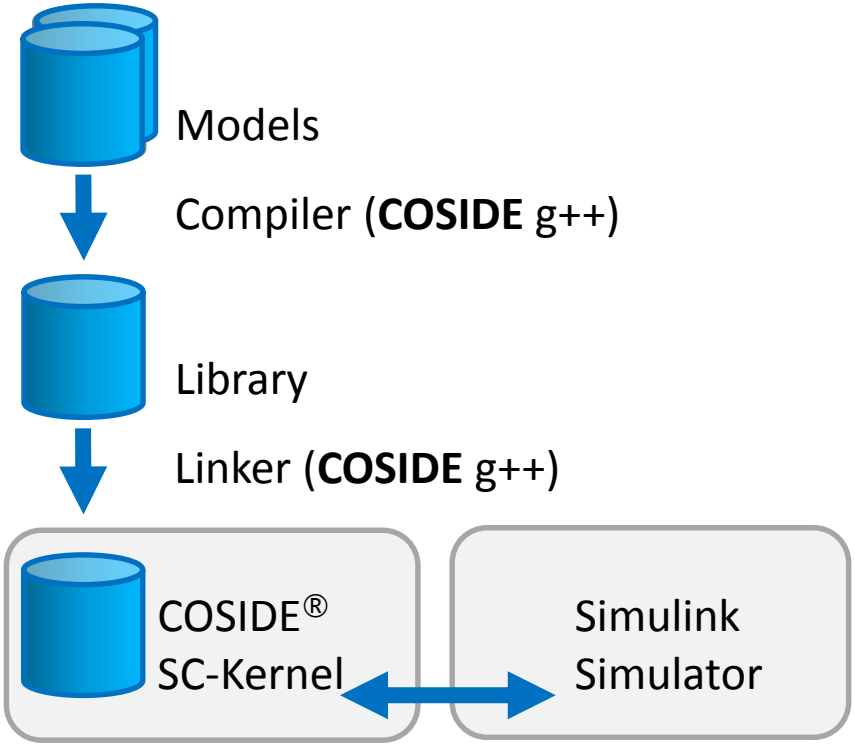
Agenda

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

COSIDE® - Model Export - Import

Black Box Export Matlab

Black Box Export

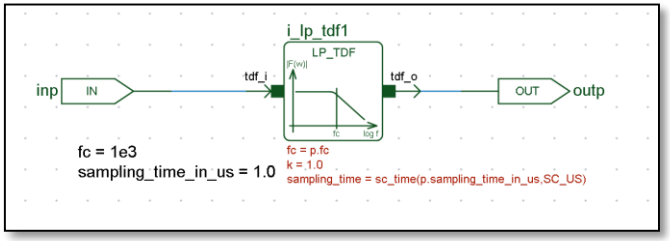


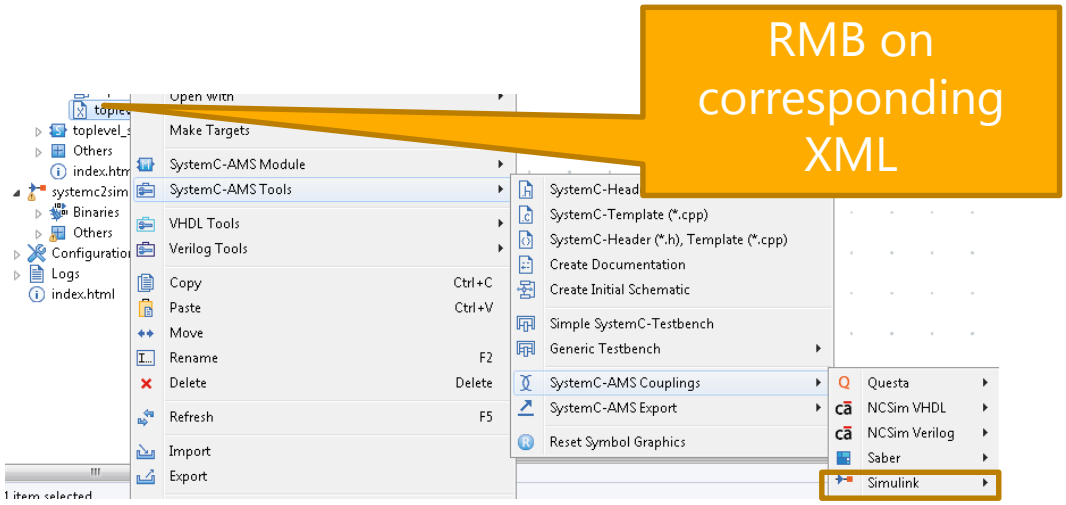
- COSIDE® compiles SystemC AMS modules to an shared object
- Additionally a wrapper (.mex file) loads generated to load it into the Simulink simulation
- Communication is possible via: Shared Memory, Shared Object or TCP/IP

COSIDE® - Model Export - Import

Black Box Export Matlab - Setup

- Single step export of COSIDE® subsystem

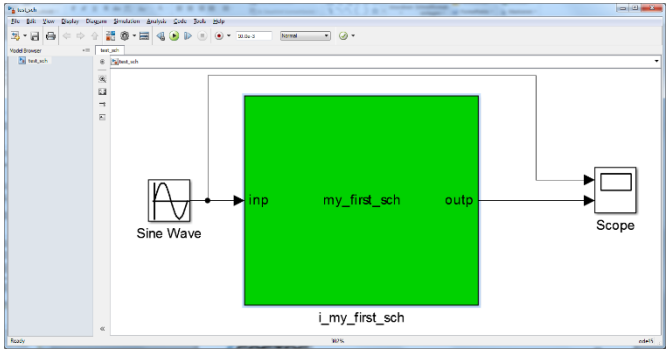


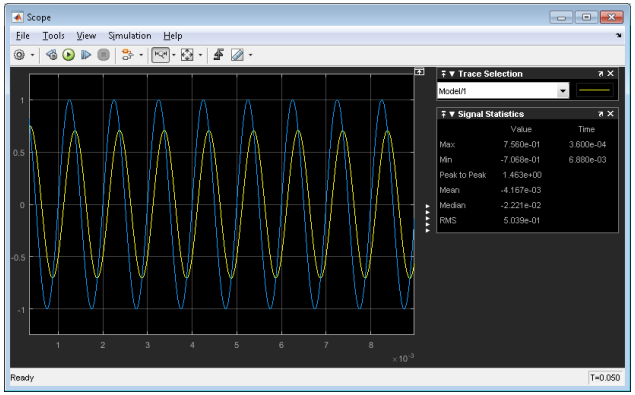


COSIDE® - Model Export - Import

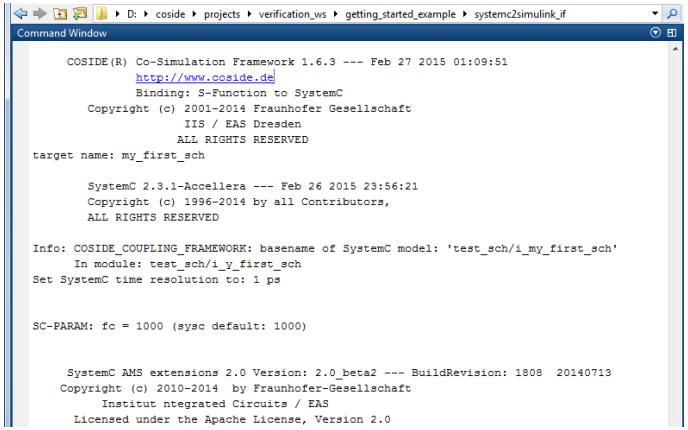
Black Box Export Matlab - Simulation

- Run your Matlab simulation as usal





- SystemC kernel outputs are directed to Matlab console



COSIDE® - Model Export - Import

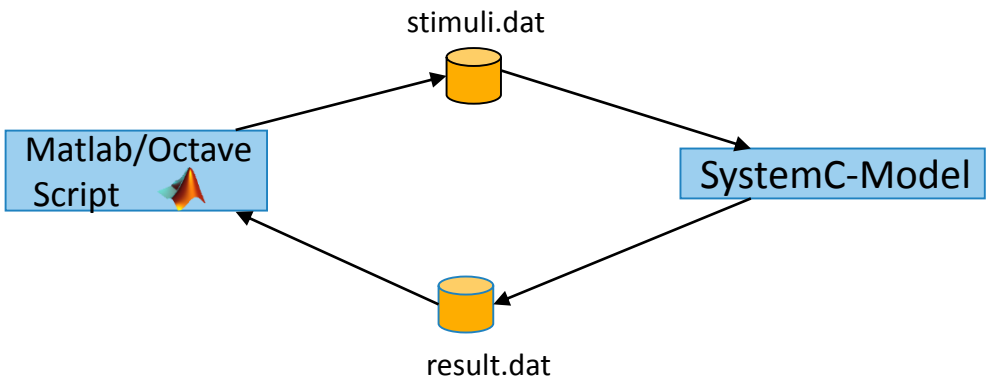
Agenda

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

COSIDE® - Model Export - Import

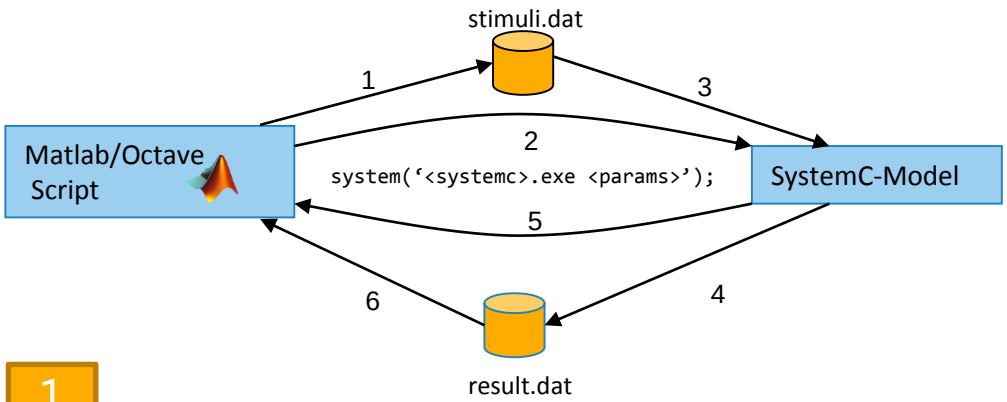
Interaction - Matlab optimization runs and regression testing

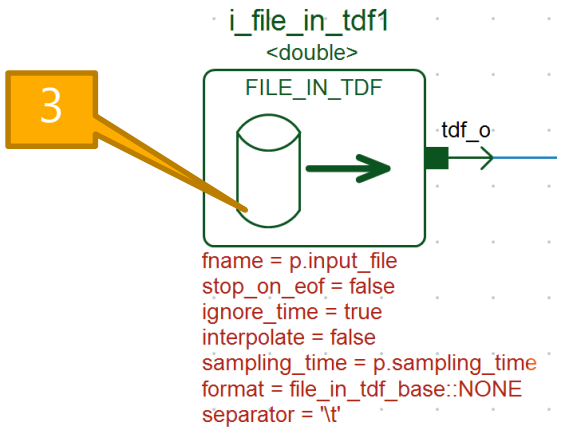
- COSIDE® exchanges data via files with Matlab/Simulink and therefore can:
  - Generate input files (Stimuli.dat)
  - Start the simulation of the SystemC model
  - Perform post-processing after reading back of the results (result.dat)



COSIDE® - Model Export - Import

Interaction - Matlab optimization runs and regression testing - steps





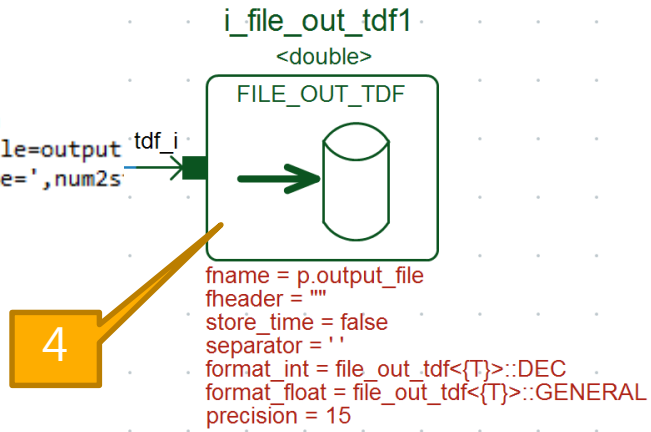
1

```
%write input stimuli to file
save('input.dat','input','-ascii');

%create command line with parameter of the SystemC simulation
cmd = 'my_system_simple_tb --input_file=input.dat --output_file=output
cmd = [cmd, '--sampling_time=', num2str(stepsize),' --sim_time=',num2s

disp([num2str(it),' : ',cmd])

%start systemc simulation
[ret,text]=system(cmd);

%read result file
load output.dat;
```

2

5

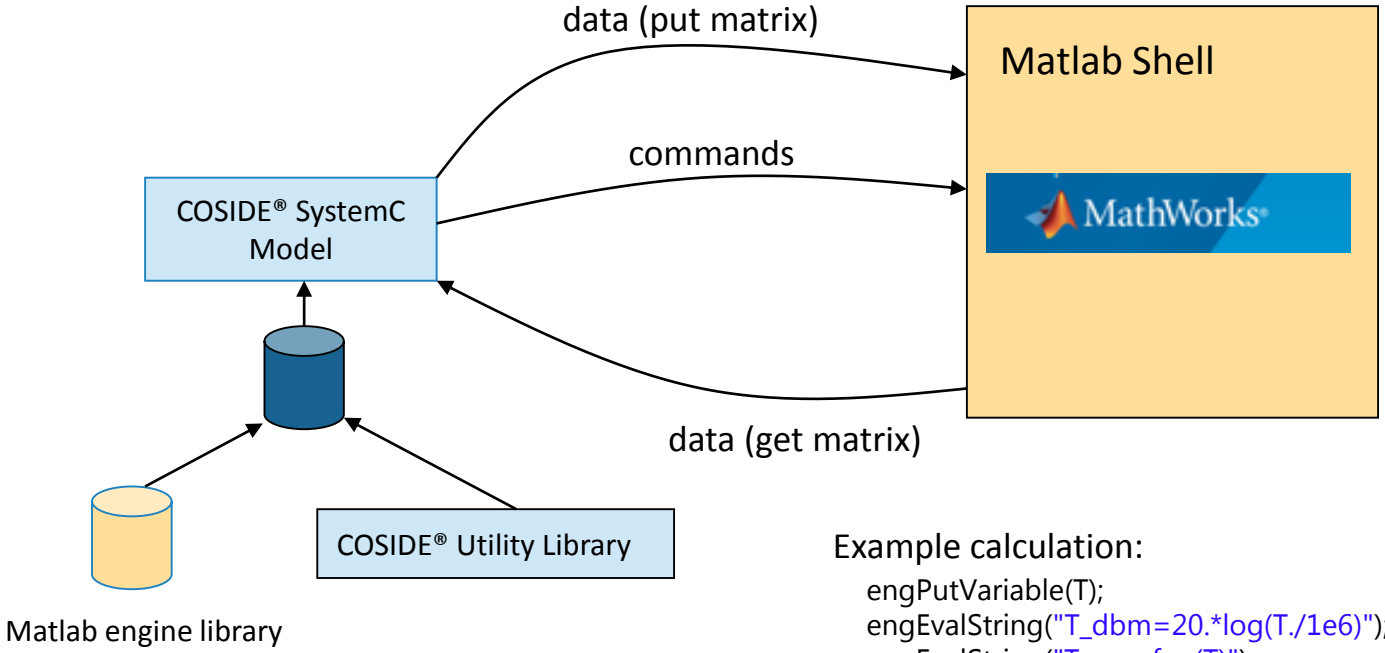
6



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



Example calculation:

```
engPutVariable(T);
engEvalString("T_dbm=20.*log(T./1e6)");
engEvalString("T=my_fun(T)");
T=engGetVariable("T");
```

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