Providing User-Friendly Interfaces to SystemC/SystemC AMS Models through Simulink, Python and Browser for Enhanced Customer Experience

Peter Alfred Hinteregger-Frießnegger

05.12.2024

Table of contents

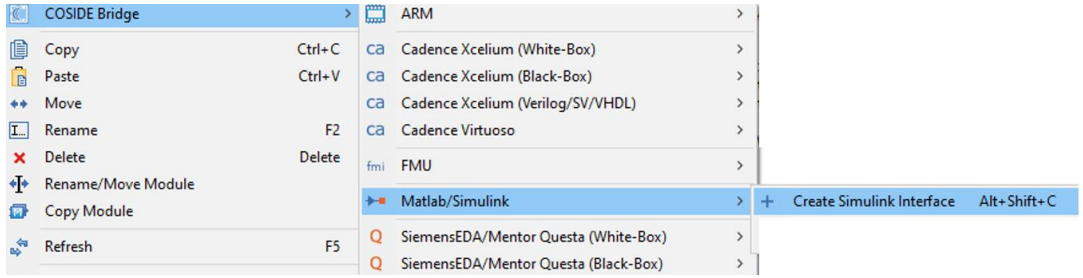
Motivation

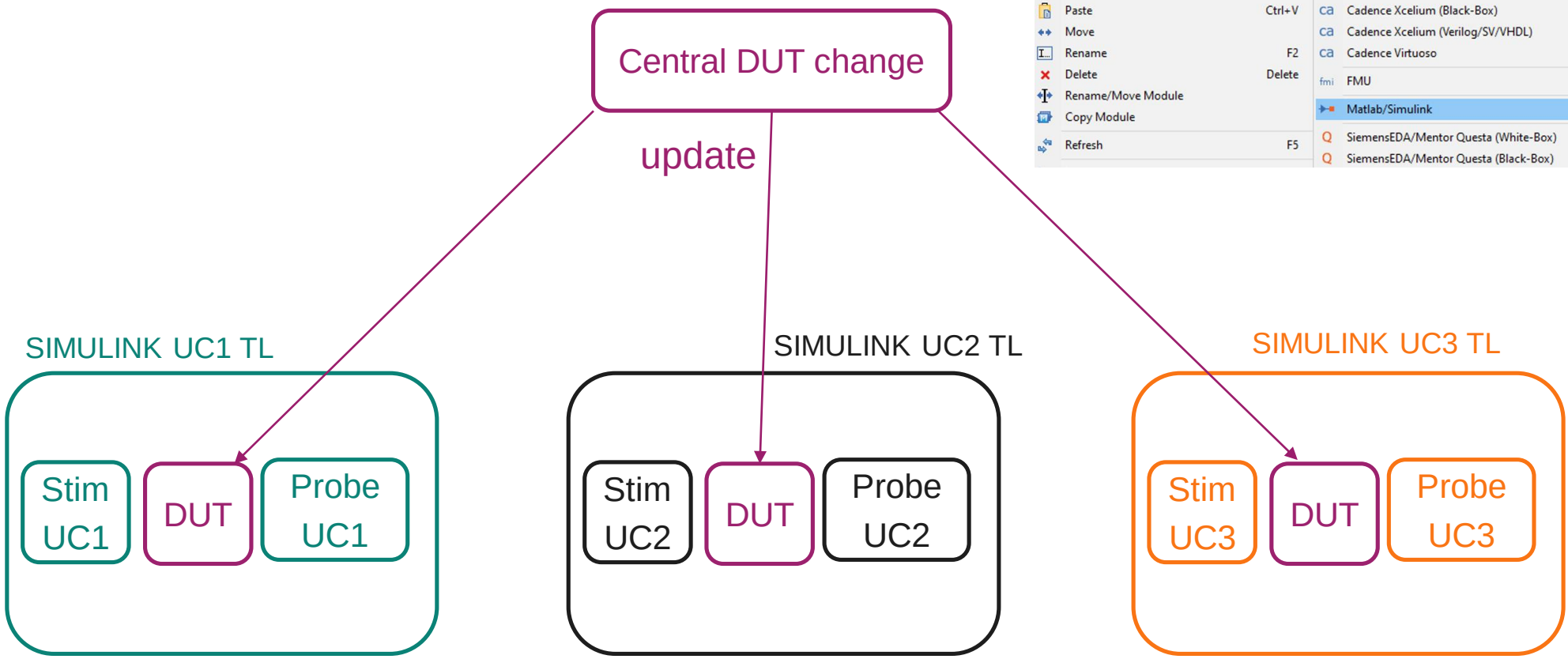
- Provide (customer) model in target environment
  - Allow early integration tests
  - Give customer performance indication already in early development state
  - Empower stakeholder understanding by providing „executable specification“
- Keep maintainable Black/grey box → secure IP and keep effort low
- Reach stakeholders not used to modelling/design environment
- Eliminate additional setup need

Table of contents

SIMULINK export

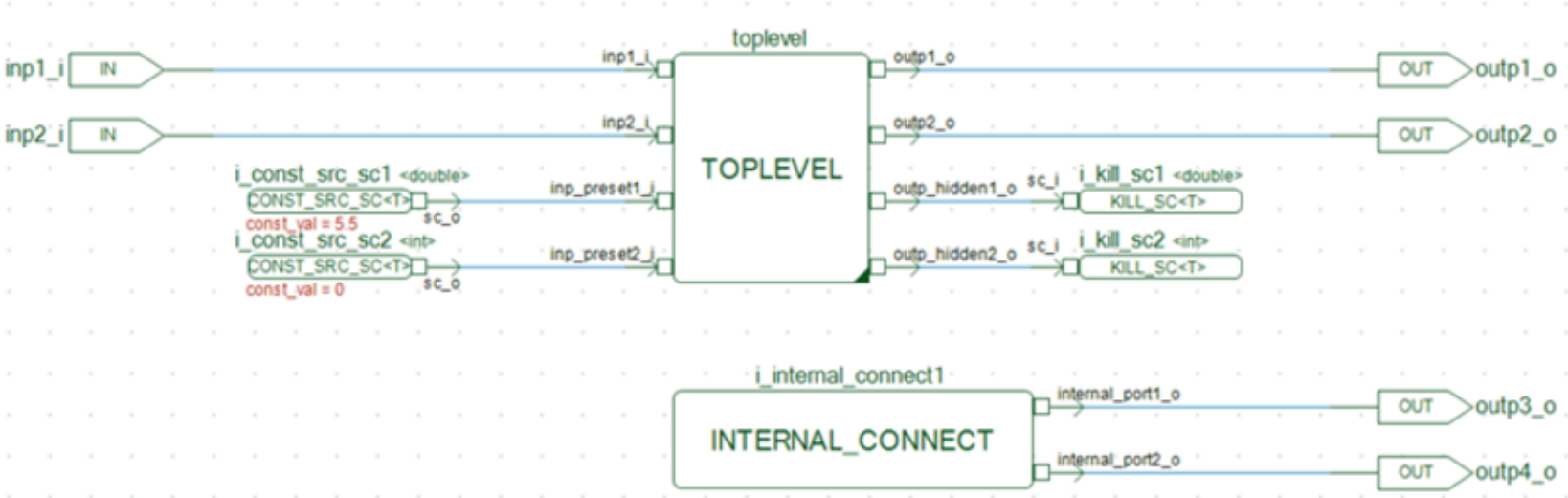
- Setup export TL and apply COSIDE® Coupling functionality on target xml
- Update/Generate export via make all (if the interface does not change, no maintainence necessary)
- Arbitrary number of customized TLs handleable





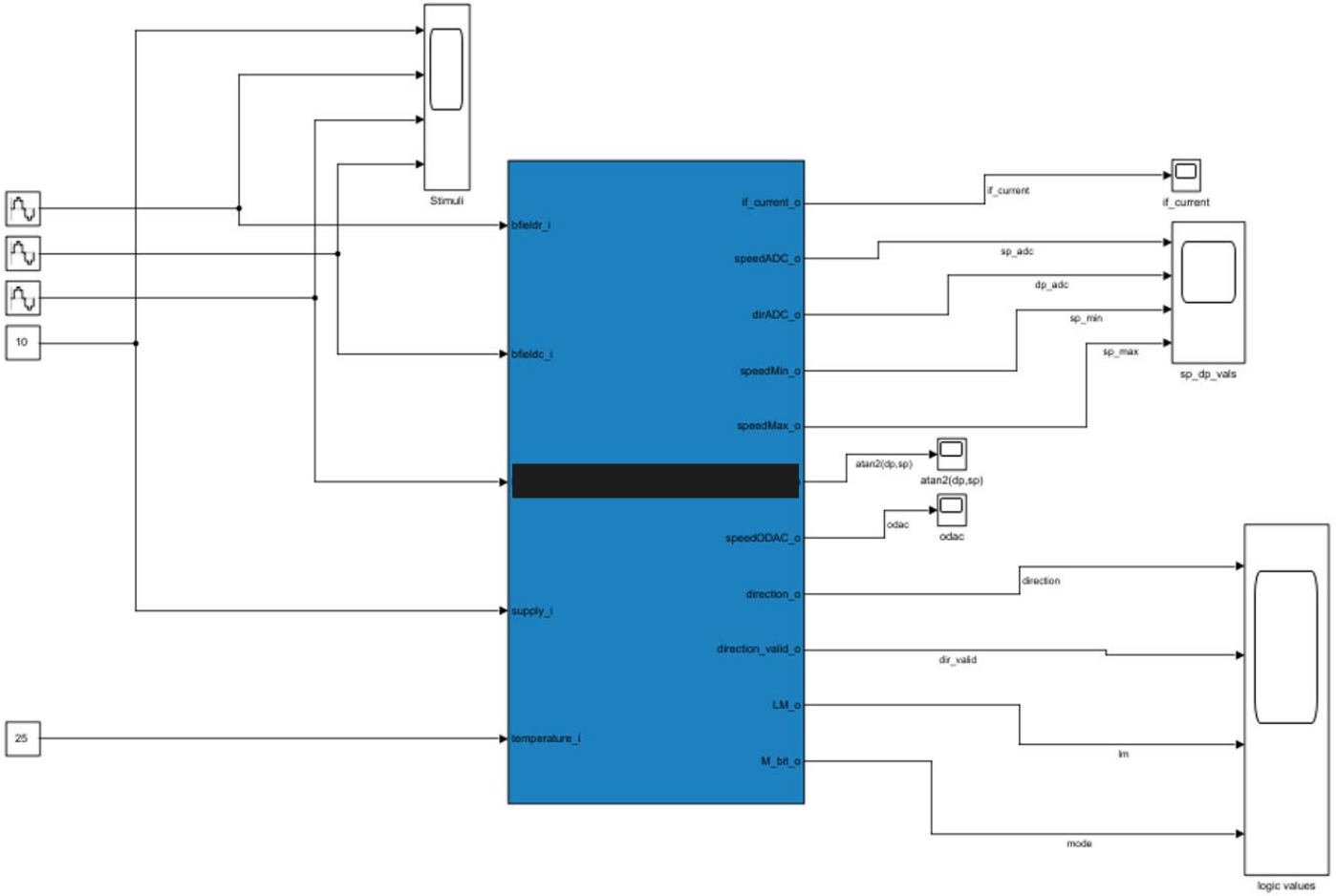
Maintain different IFs via schematics/instantiation in modules

- Provide useful interface adaptations to ease model handling & provide „spys“ on lower levels
  - preset inputs which the user shall not change/ do not need to care
  - Hide outputs which the user is not allowed to see/ do not need to care
  - Add internal signals to interface
- COSIDE® provides the functionality (cos_assign, hierarchical access, …) to connect an outside block to signals within another block on arbitrary hierarchic levels - No need to adapt model interfaces ... Only modification of wrapper schematic necessary! (even global signal can be reached)

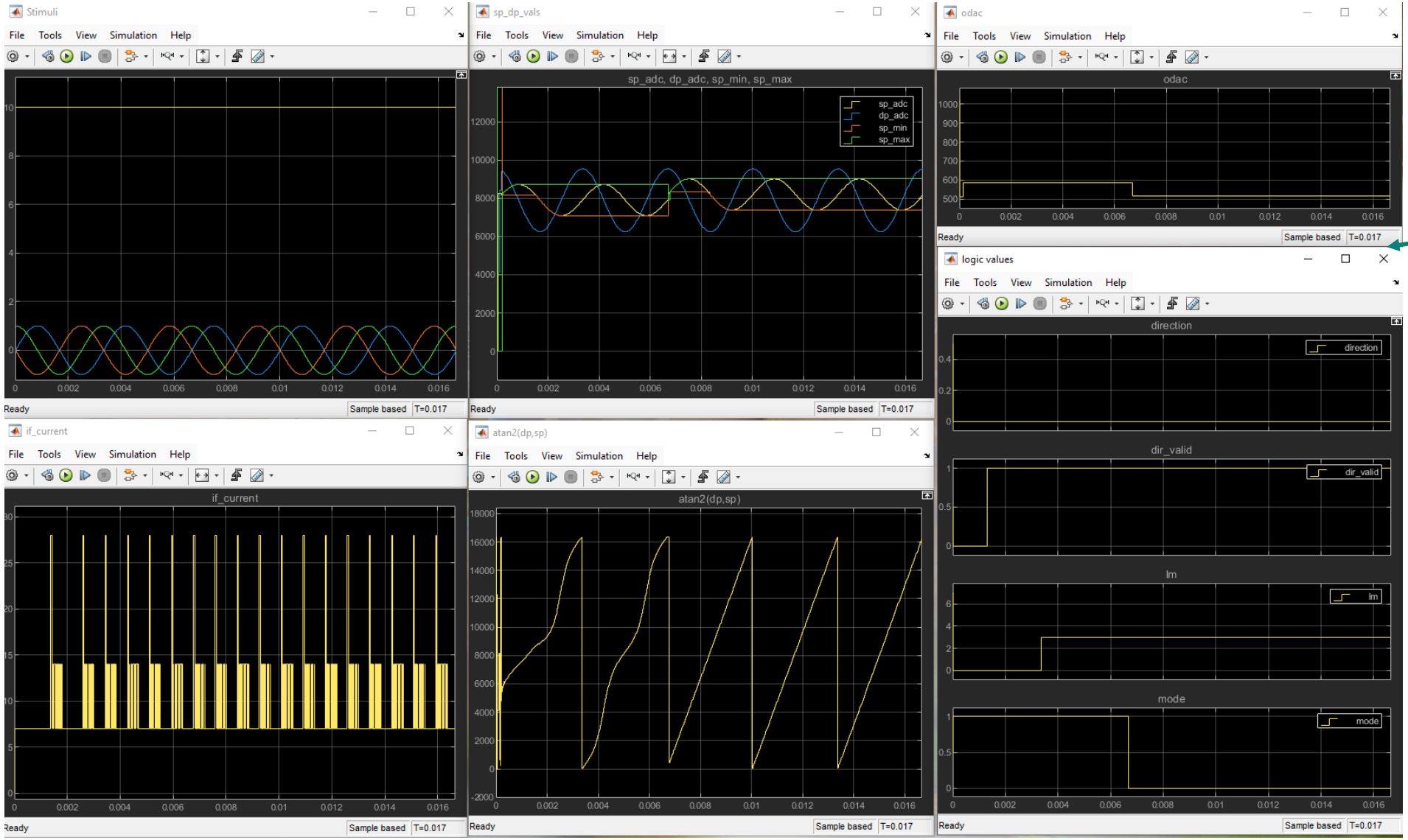


Deliver SIMULINK export to customer

- Easy drag and drop integration into simple TB
- Black Box attribute prevents later modifaction of provided IF
- Deliverable only requires mdl, mex (and opt. mingw) files
- Customer receives a meaningful executable OotB



Deliver SIMULINK export to customer example



These figures are just one click of the play button away

Parametrization supported by TL block

- TL parameter of the export entity will be present in the SIMULINK block GUI
  - Allows static configuration
  - Already present when module constructor is called

```
/// parameters
const struct params
{
    // 34_BEGIN_USER_CODE
    // you can add own code here.
    // END_USER_CODE_34
    std::string fuse_str; /** fuse file id */

    params()
    {
        // 35_BEGIN_USER_CODE
        // you can add own code here.
        // END_USER_CODE_35
        fuse_str = "";
    }
} p;
```

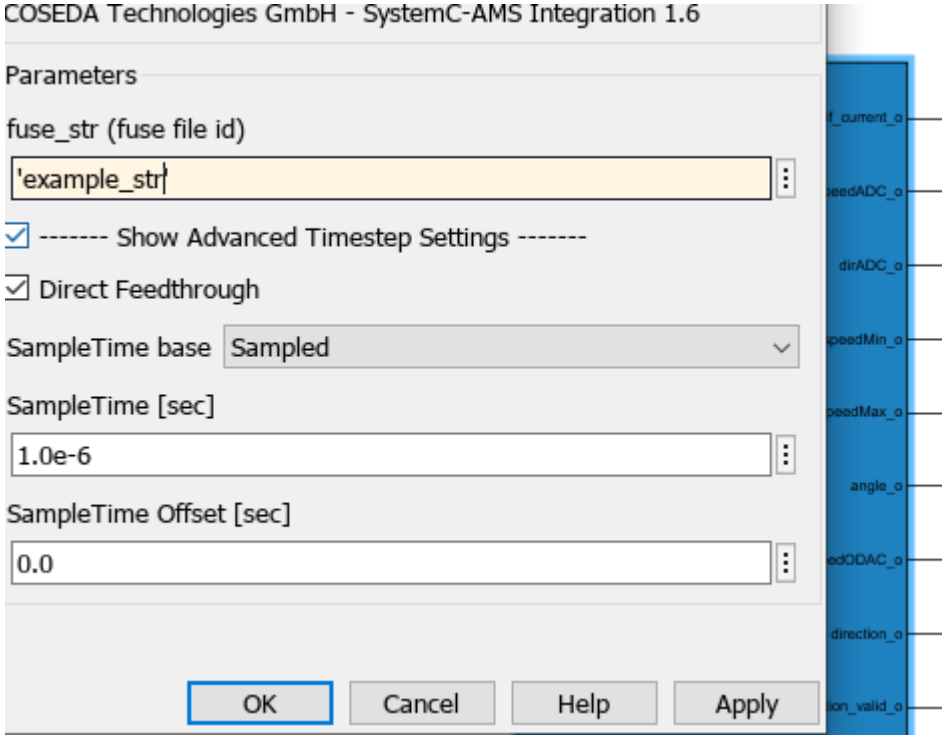
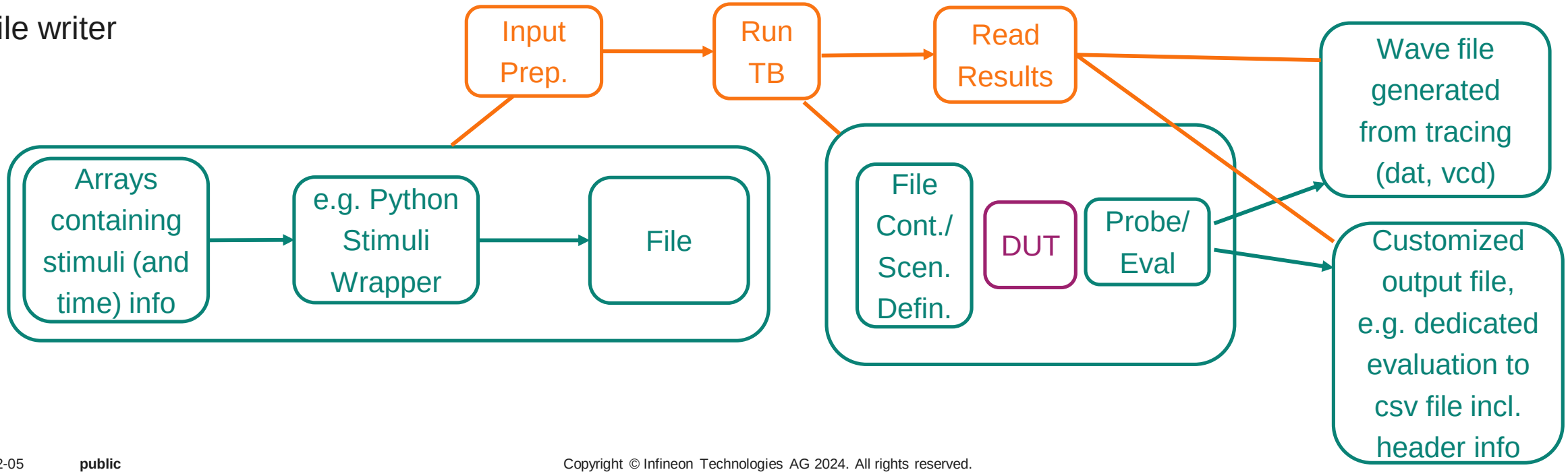


Table of contents

Connect to Python
(or any other environment able to run executables)

- SystemC(-AMS) code compiled to binary is by its own nature a standalone Black Box export
- Straight forward ways to incorporate complex stimuli
  - Provide mapping from native description in target environment to generate a file structure, which is read in SystemC domain OR
  - Predefine stimuli by setting up scenarios by abstract definition or file structure and only allow selection via command line parameter / param reader
- Visualization and automated post-processing of behavior / more abstract results by wave files or customized file writer



Connect to Python (the interactive way)

- COSIDE® provides functionality to provide sim control to Python
- This includes e.g. Reading from and writing to signals as well as running the model for an arbitrary time
- Provided sequences can be a function of read back results
- Tracing can be adapted to the scenario
- Overall more freedom: Less effort to adapt IF, ...
- **Attention: IP protection might get violated**

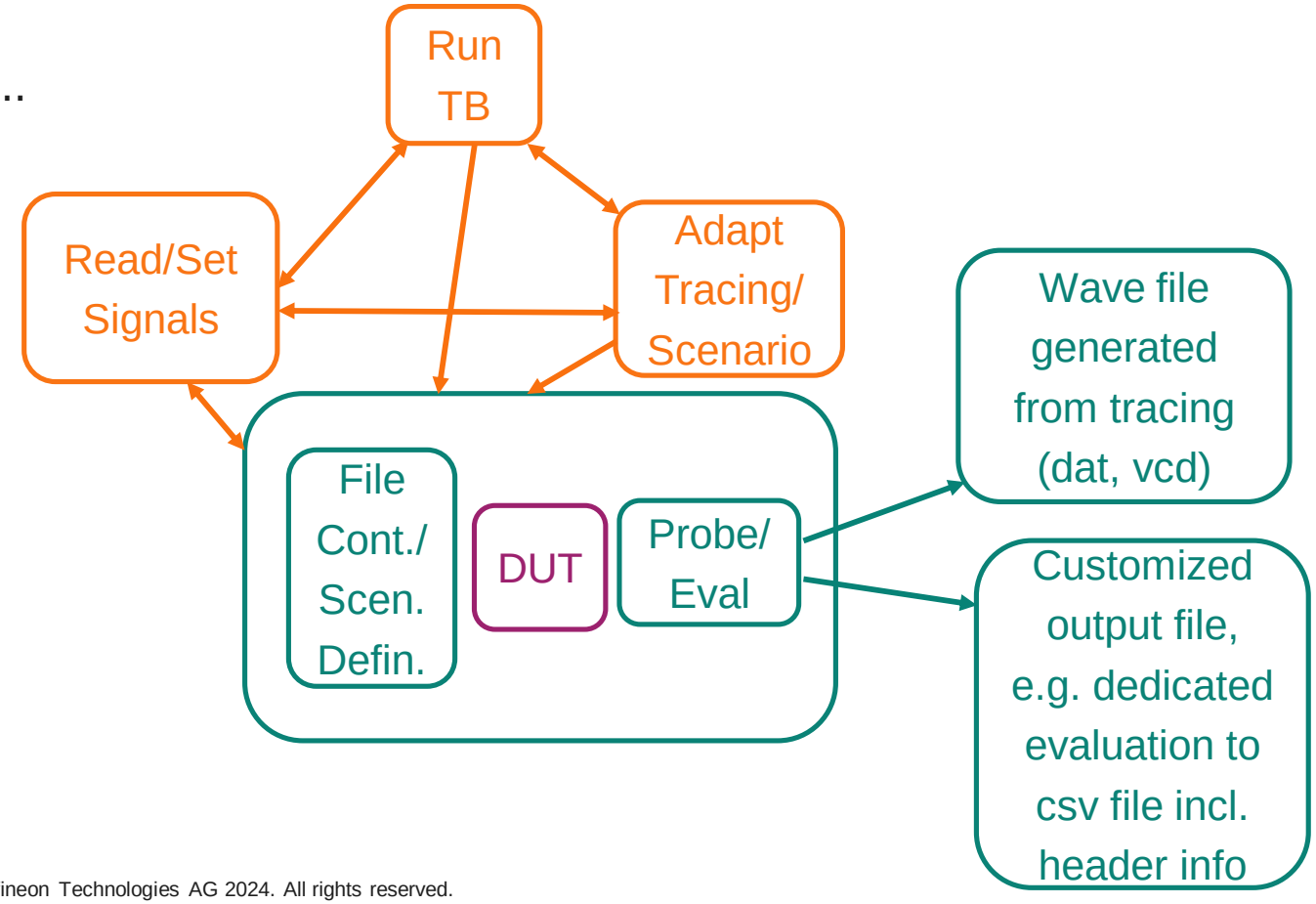


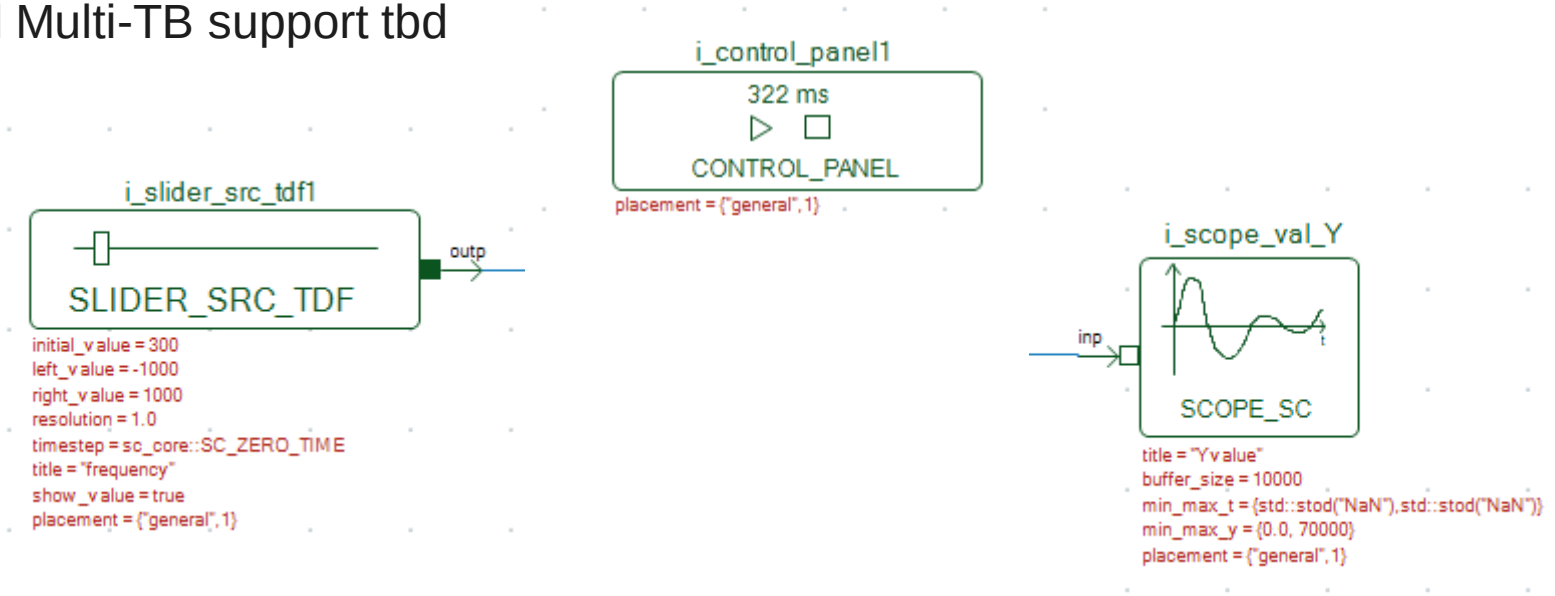
Table of contents

SystemC(-AMS) GUI w/o Setup? → use your browser

- Insert special COS Panel modules to be used as IF
- Run local server (=run executable)
- Full control on I/O kept (and therefore IP can be protected)
- Based on Apache eCharts → Huge Visualization potential
- Customize the GUI by providing own HTML/CSS
  - Integrate your documentation/discalimer into the executable
  - Stick to corporate design
- Parallel Multi-TB support tbd

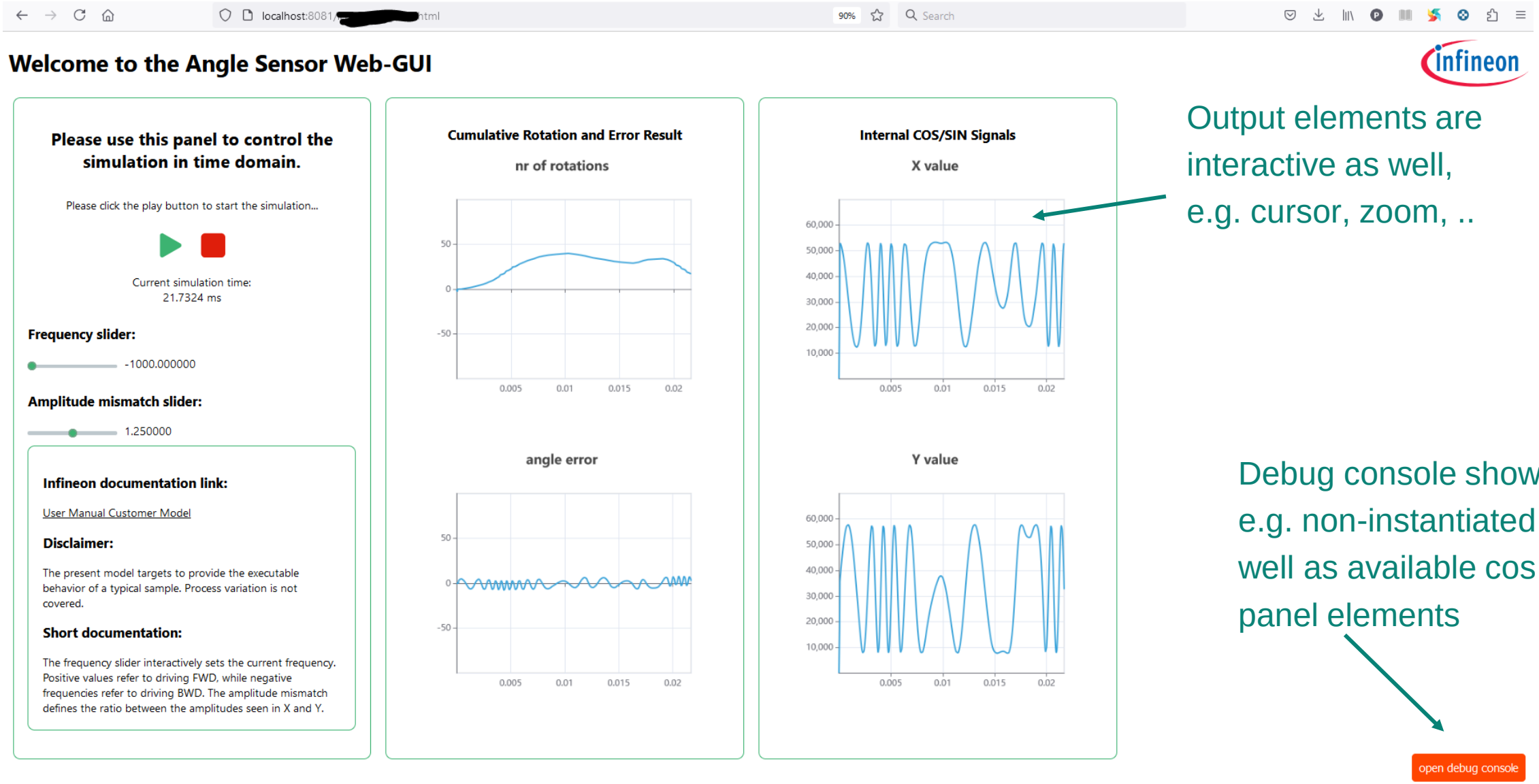
Link FE/BE by linking
via string comparison

```
<div class="cos-grid-item">
<h3>Displaying the simulation results...</h3>
<div id="cos-panel:*.i_scope_val_*" style="width: 100%;"></div>
</div>
```




```
Info: cos_panel_controller:
Open:
        http://localhost:8081
in web browser
```

COS Panel FE example



Output elements are interactive as well, e.g. cursor, zoom, ..

Debug console shows e.g. non-instantiated as well as available cos panel elements

Questions & answers


Infineon