MetaPSS: An Automation Framework for Generation of Portable Stimulus Model

Jaimini Nagar, Thorsten Dworzak, Sebastian Simon, Ulrich Heinkel, Djones Lettnin
Agenda

• Introduction
• Model-driven PSS generation flow
• Code Generation
• Conclusion
Introduction

• Portable stimulus model is a single representation of stimulus and test scenarios usable across many levels of integration
• A well-defined meta-model based framework for automated generation of portable stimulus representation
• Inspired from Model-Driven-Architecture (MDA) principle for the code generation
• Meta-model is visually symbolized with UML class diagram
Problem statement

- Writing a PSS model manually costs in terms of human effort
Taxonomy of PSS model

• Component - encapsulating reusable model units

• Action - unit of behavior

• Flow objects - transmit data
Model-driven PSS generation flow (1)

- Capture high-level attributes

- Platform independent way of describing PSS model

- Enables the developer to consider the view that must be generated
Model-driven PSS generation flow (2)

- MetaPSS: A meta-model to signify PSS
- Backbone of the automation framework
- One-time effort
- Define the structure and elements of PSS
Code Generation (1)

- Run <make gui>

- Graphical User Interface to fill the XML Data Structure
Code Generation (2)

• Steps for each PSS model generation
  1. Review the design specification
  2. Identify key elements of PSS
  3. Fill the required data in GUI
  4. Save as *.xml
  5. Run < make pss >

• Generate Portable Stimulus model as *.pss

• Runtime for the generation of code ≈ 1 sec

```cpp
resource uart_r { }

enum uart_mode_e {mode0, mode1, mode2, mode3};
enum baud_rate_e {b2m, b500k, b250k, b115k, b20k, b19k, b9600};
enum uart_nr_e {uart0, uart1};

component uart_c {
    pool [2] uart_r uart_p ;
    bind uart_p * ;

    //import package here //

    action uart_txn {
        rand int tx_data ;
        rand bit [8] tb ;
        // start of user code here for action uart_txn //

        // end of user code here for action uart_txn //
    }
}
```
Advantages

• Re-usable multiple time
  • An example of re-usability

• Give a quick start for working on the Portable Stimulus methodology

• Defined space to add dynamic and design specific user code
Conclusion

• Reduce effort to utilize Portable Stimulus methodology reinforced verification
• An automation framework contains major key constructs of PSS
• Generated PSS representation is an abstract model and platform independent
• Not included executable that defines the targeted verification platform
Acknowledgement

This work has been developed in the project VE-VIDES (project label 16ME0243K) which is partly funded within the Research Programme ICT 2020 by the German Federal Ministry of Education and Research (BMBF).
Questions