A UVM-based Approach for Rapidly Verifying Digital Interrupt Structures

Christoph Rumpler, Alexander W. Rath, Sebastian Simon

Prof. Heinz Endres

Infineon

FHWWS
Agenda

1. Motivation
2. Concept
3. Model of Interrupt Structures
4. UVM Testbench and Tests
5. Conclusion and Outlook
Agenda

1. Motivation
2. Concept
3. Model of Interrupt Structures
4. UVM Testbench and Tests
5. Conclusion and Outlook
Motivation

• Many SoCs include processors (e.g. ARM Core)
• Processors are interacting with peripherals
• Increasing number of interrupt sources in SoCs
• Interrupt structures are getting more complex
• Ensure correct functionality

-> Systematic verification of such interrupt structures is needed
Motivation

• Up to now verification of such interrupt structures is done manually
  – Time consuming
  – Not standardized
  – Error prone (No flow)
Motivation

• Our goal: Generate verification code (test sequence + testbench) out of structural information (model) from the interrupt structure

- Standardized (UVM)
- Timesaving
- Low error risk
Agenda

1. Motivation

2. Concept

3. Model of Interrupt Structures

4. UVM Testbench and Tests

5. Conclusion and Outlook
Concept

- Meta model for interrupt structures
- Creating an abstract interrupt structure model of the SoC
- Using parser scripts to create tests and testbench
Concept

• Parser scripts use the information (interrupt model) to split the interrupt structure in separate interrupt paths for verification
  – Applying simplification rules:

\[
((S1 \land S1IE) \lor (S2 \land S2IE)) \land GIE
\]

\[\iff\]

\[
(S1 \land S1IE \land GIE) \lor (S2 \land S2IE \land GIE)
\]

Every interrupt path is verified separately!!!
Agenda

1. Motivation

2. Concept

3. Model of Interrupt Structures

4. UVM Testbench and Tests

5. Conclusion and Outlook
Model of Interrupt Structures

• An interrupt structure might look as follows:

![Diagram of interrupt structure with labeled modules and connections]

Module 1

Module 2

Processor
Model of Interrupt Structures

• A meta model of Interrupt Structures must contain information about
  
  – Number and names of registers and their bit fields
  – Names of ports (source and core nodes)
  – General interrupt structure
Model of Interrupt Structures

- UML diagram of the meta model

Meta model is filled with specific information of the interrupt structure (project)
Agenda

1. Motivation

2. Concept

3. Model of Interrupt Structures

4. UVM Testbench and Tests

5. Conclusion and Outlook
UVM Testbench and Tests

• **UVM Testbench:**
  – One monitor for each source / core node
    • TLM port for communication with Scoreboard
    • Virtual interface for Interface Module
  – One Interface Module for each source / core node
    (connected to design by using SV bind statement)
  – One Scoreboard to evaluate interrupt events collecting functional coverage data
UVM Testbench and Tests

- Block diagram of the UVM Environment:
UVM Testbench and Tests

• UVM Environment:

```plaintext
function void build_phase (uvm_phase phase);
    super.build_phase (phase);
    monitor SOURCE1 = interrupt_monitor::type_id::create
        ("monitor_SOURCE1",this);
    uvm_config_db#(string)::set(uvm_root::get(),"*monitor_SOURCE1",
        "signal_name","SOURCE1");
    uvm_config_db#(int)::set(uvm_root::get(),"*monitor_SOURCE1",
        "sensitive_level",1);
endfunction : build_phase

function void connect_phase (uvm_phase phase);
    super.connect_phase(phase);
    monitor_SOURCE1.item_collected_export.connect
        (interrupt_sb.item_collected_import);
endfunction : connect_phase
```
• Test sequences:
  (One test sequence for each interrupt path)

function new(string name = "seq_SOURCE1CORE_intr ");
  super.new( name );
  path.set_name ("seq_SOURCE1CORE_intr");
  path.set_source_node_name ("SOURCE1");
  path.set_core_node_name ("CORE");
  path.ie_registers.add("interrupt_ctrl","global_en",1);
  path.ie_registers.add ("module1_ctrl","module1_en",1);
  path.status_registers.add ("module1_sts","source1_is",1);
  path.clear_registers.add ("module1_clr","source1_ic",1);
endfunction
UVM Testbench and Tests

- Test scenarios:
  - Pending interrupt
  - Non-Pending interrupt
  - Open Interrupt Enable (for IE register)
  - Not triggered (for ST register)
Agenda

1. Motivation
2. Concept
3. Model of Interrupt Structures
4. UVM Testbench and Tests
5. Conclusion and Outlook
Conclusion and Outlook

• We presented a new approach for verifying interrupt structures in processor based designs

• **Idea:** verification testbench and tests sequences are generated automatically based on a meta model of interrupt structures

• **Advantages:**
  – Tests are standardized (independent of the verification engineer)
  – Code is generated automatically which saves time and has a low error risk
Conclusion and Outlook

• Our future work:
  – Hierarchical reusability of existing interrupt models in projects (model from module verification can be used for system / top level verification)
  – Concept of tagging generated code and user-defined code

• Overall goal: Providing an easy-to-use verification method that is able to verify interrupt structures of any design
Thank you for your attention!