Accellera Standards
Technical Update

Uwe Simm (Cadence Design Systems)
Sharon Rosenberg (Cadence Design Systems)
Erwin de Kock (NXP Semiconductors)
Philipp A. Hartmann (Intel Corporation)
Overview

- UVM and IEEE-1800.2
- Portable Stimulus
- IP-XACT
- SystemC
Accellera Standards Update
UVM and IEEE-1800.2

On behalf of the working group
Uwe Simm
uwes@cadence.com
What is happening?

UVM1.* Library

Standard / LRM

IEEE 1800.2 WG

.. 2014

Library

2016..
Deliverables

• IEEE 1800.2 WG
  – 1800.2 = SV focused
  – Defines UVM functional API (not an implementation)
  – Produces 1800.2 LRM
  – Allows for multiple implementations

• Accellera WG
  – Delivers UVM Library (SV) Reference Implementation matching 1800.2 LRM
  – Provide bugfixes for UVM library
Contribution options

• IEEE 1800.2 WG
  – For IEEE members every other week @ 9AM PST call
  – Tracking via accellera.mantishub.com → P1800.2

• Accellera WG
  – For Accellera members every other week @ 9AM PST call
  – Tracking via accellera.mantishub.com → UVM
General focus areas 1800.2

- Allow flexibility for future implementations
- Remove implementation artifacts from LRM
- Review API in the light of backward compatibility, consistency, simplification and extensibility
- Fully document and describe API

(Note: Library can support deprecated API)
1800.2 REG Sub WG

- Identification and closure of register related issues
- Definition of necessary API to support future use models e.g. SoC use models
- Further align reg sub system with other relevant standards (IPXACT)
- Currently: issue collection and prioritization
1800.2 TLM Sub WG

• Align UVM TLM with current IEEE 1666-2011 standardized TLM-1 and TLM-2.0 concepts
• Improve current UVM TLM standard documentation to explicitly explain execution semantics and underlying concepts
• Discuss completeness of current UVM TLM API
Questions
Portable Stimulus Working Group

Authored by the PSWG
Presented by Sharon Rosenberg
Senior Architect
Cadence Design systems
Portable Stimulus Working Group (PSWG)

• Our goals
  – Enable value and automation for individual teams
  – Allow sharing and executing scenarios across teams and platforms
  – Define a clear and robust semantic for consistent implementation of multiple tools by multiple vendors

• Group Info
  – Was officially formed on December 15, 2014 by the Accellera board of directors
  – Chair: Mr. Faris Khundakjie, Intel Corporation
  – Vice-Chair: Mr. Tom Fitzpatrick, Mentor Graphics
  – Secretary: Mr. Tom Anderson, Breker
  – Multiple participating companies
PSWG Active Members

Agnisys
Altera
AMD
AMIQ EDA
Analog Devices
Breker
Cadence
Cisco
CVC
Encore Semi

Freescale
IBM
Intel
Mentor Graphics
NVIDIA
Qualcomm
Semifore
Sonics
Synopsys
Vayavya Labs
PSWG Vision

Proposed Portable Stimulus Diagram

Abstract Portable Stimulus Model
- Syntax/Concepts/Semantics
- Use Case Verification
- Visualization
- Runtime Portable Semantics

Tools (Secret Sauce)

Verification Environment

Platform

Scope (Integration)
- Middleware (Graphics, Audio, etc..)
- OS & Drivers
- Bare Metal SW
- System on Chip (HW + SW)
- Sub-System
- IP

User
- Architect
- HW Developer
- Analog Developer
- SW Developer
- Verification Engineer
- SW Test Engineer
- Post-silicon Validation Engineer
Solution Requirements and Scope

User Requirements
- Collected and prioritized by the proposed working group
- A list of 117 unique solution requirements
- Examples include constraints, inheritance, reuse across platforms, coverage and more

Usage examples to define the problem space
- 14 Concrete use-case examples to help scope the requirements and assess the contributions
- Users and vendors collected both typical and critical challenges
- Meant to be universally recognized as important and also representing a whole class of use cases with similar challenges
Covering the Spectrum with Usage Examples

- Multi-core/Multi-master data cache coherency
- Stimulus portability across pad selection connectivity
- Resource allocation for concurrent peripheral activity
- Video processing pipeline
- Exhaustive exercise of power states
- Exercising Ethernet controller

PSS Standard Scope
PSWG Standardization Status and Timeline

- **PSWG Face to Face Oct. 12-14**
  - Complete, Next Step:
  - Cadence, Mentor, Breker and Vayavya (CMBV) to work on exploring options 2 & 3 (and their deltas), consult with user companies
Contributions and Collaboration

Cadence, Mentor and Breker

- Submitted a domain specific language that combines C++ and SV intuitions
- Includes scenario specification and coverage

Breker

- Submitted also a C++ only library
- Took the AI to add value constraints and coverage parts later to demo the usage examples

Vayavya

- Contributed a complementary syntax to generate register sequences, firmware and driver routines from a canonical/standard register description
Contributions and Collaboration

Contributors are open minded and already converged into a single approach

- Combine constraint model-based and graph approaches
- Usage of actions to capture the legal space
- Modular actions with attributes that represent pieces of behavior
- Interface to other actions with inputs, outputs, and resource needs
- Value and path constraints to represent model and scenario rules

A focused team is working in collaboration to combine the two solutions into one.
Options to Explore – Food for Thought

**PS as C++ add-on syntax with:**

1) Mostly declarative (procedural where needed)
2) Confined in compilation units isolated from non-PS code
3) Follows 2 stage processing:
   generation of target compilation units by new tool linked with non-PS compilation units by new tool linker or off-the-shelf linker

**PS as declarative new language, improved with:**

1) Large rigorous interoperability gen and run time with non-PS C/C++ modules
2) Improved procedural support within overall declarative theme
3) Improved measures for scalability and reactive stimulus
Summary

• The Portable Stimulus Working Group is assigned to capture portable stimuli, coverage and checking
  – Driving toward milestones to have a prototype on May 2016 and 1.0 release on Jan 2017

• There are solutions out there that demonstrate significant speed-up in tests development and an overall improved quality
  – Try the existing solutions 😊
  – Make sure you are not being locked into a non-standard approach

• Join us in Accellera to ensure that your needs are addressed

• To learn more about the Portable Stimulus Working Group, visit the web site
Accellera Standards Technical Update
IP-XACT

Erwin de Kock, NXP Semiconductors
11-11-15
Vice Chair IEEE P1685 WG
From IEEE 1685-2009 to IEEE 1685-2014
IP-XACT, A Standard Structure ...

- For packaging IP
  - XML document format for IP data sheets

- For integrating IP
  - XML document format for IP instances and their connections

- For reusing IP within tool flows
  - Tool interface format to access XML data

"Electronic Data Sheet"
- Bus interfaces
- Ports
- Registers

"Electronic Interconnect"
- IP instances
- Connections

Tool

Database

XML
What does IP-XACT standardize?

- XML schema
  - Bus and abstraction definitions (interface definitions)
  - Components (IPs) and abstractors (special IPs for abstraction conversion)
  - Designs (component instances and interconnect)
  - Design configurations (configuration of instance views and interconnections)
  - Generators (tool references and tool inputs)

- Semantic Consistency Rules (SCR)
  - Additional rules for XML documents on top of XML schema rules

- Tool interface (TGI)
  - A set of HTML messages for interaction between tools and databases
What changed in IEEE 1685-2014?

• So much that we cannot discuss it all, including
  – Change of XML schema namespace from spirit to ipxact
  – Updates of Semantic Consistency Rules
  – Full tool interface support to create, edit, and access all XML data

• Today, we focus on
  – Parameter propagation
  – Expression language
  – Conditionality
  – View-specific port maps

• And how IEEE 1685-2014 enables top-down design flows
Parameter Propagation in 2009

• Parameters in
  – Components

• Designs instantiate
  – components with parameter values

• One level of parameter value propagation
  – From design (component instance) to component
Parameter Propagation in 2014

• Parameters in
  – Components
  – Designs and design configurations
  – Bus and abstraction definitions

• Designs instantiate
  – Components with parameter values

• Components instantiate
  – Designs and design configurations with parameter values
  – Buses and abstraction definitions with parameter values

• Full parameter value propagation
  – Throughout the whole design hierarchy

Design E
  • Parameter Z

Component C
  • Parameter X
  • Parameter Y
  • Instance of E
    • $Z = X + Y$

Design D
  • Instance of C
    • $X = 3$
    • $Y = 4$
Expression Language in 2009

- XPath 1.0 expression language
  - Design language neutral
- Parameter definition

```xml
<spirit:parameter>
  <spirit:name>X</spirit:name>
  <spirit:value
    spirit:id="X" spirit:resolve="user">
    3
  </spirit:value>
</spirit:parameter>

<spirit:left
  spirit:format="long"
  spirit:resolve="dependent"
  spirit:dependency="id('X') + id('Y')">
  7
</spirit:left>
```

Component C
- Parameter X
- Parameter Y
- Port P
  - Left X+Y
  - Right 0
Expression Language in 2014

- SystemVerilog expression language
  - End user friendly
- Parameter definition

  ```
  <ipxact:parameter
      parameterId="X" resolve="user">
    <ipxact:name>X</ipxact:name>
    <ipxact:value>3</ipxact:value>
  </ipxact:parameter>
  ```

- SystemVerilog expression

  ```
  <ipxact:left>
    X + Y
  </ipxact:left>
  ```

- Other expression conversion examples
  - Example 1: 0x1F becomes ’h1F
  - Example 2: concat(id(‘S’),id(‘T’)) div 2 becomes {S,T} / 2

Component C

- Parameter X
- Parameter Y
- Port P
  - Left X+Y
  - Right 0
Conditionality – New in 2014

- Conditional presence of data in
  - Components
  - Design and design configurations
  - Abstraction definitions

- Implication on Semantic Consistency Rules
  - Some rules can be checked only after IsPresent values are resolved
  - Example 1: Registers R1 and R2 overlap only if they are both present
  - Example 2: Output ports O1 and O2 drive the same input port through connections C1 and C2 only if O1, O2, C1, and C2 are all present

Component C

- Parameter X
- Parameter Y
- Register R
  - IsPresent if X=3
- Port P
  - IsPresent if X+Y=5
Port Maps in 2009

- **Component bus interface**
  - Bus type [MyBus]
  - Abstraction type [MyBusAbs_rtl]
  - Port maps
    - MyCLK → clk
    - MyRESETn → rstn
    - MyRDATA → rdata
    - MyWDATA → wdata
  - Ports
    - clk
    - rstn
    - addr[3:0]
    - rdata[31:0]
    - wdata[31:0]
Port Maps in 2014

- Component bus interface
  - Bus type [MyBus]
  - Abstraction type [MyBusAbs_rtl]
  - Port maps
    - MyCLK <-> clk
    - MyRESETn <-> rstn
    - MyRDATA <-> rdata
    - MyWDATA <-> wdata
  - Ports
    - clk
    - rstn
    - addr[3:0]
    - rdata[31:0]
    - wdata[31:0]
Enabling Mixed RTL and ESL Design!

Component instance I

MyBus
MyBusAbs_rtl
MyBusAbs_tlm

Component instance J

Bus interface
Port map rtl
Port map tlm

Port map tlm

C

Port map tlm

Port map rtl

View rtl

View tlm

View rtl

View tlm

Design config rtl: I \rightarrow \text{rtl} \quad J \rightarrow \text{rtl}

Design config tlm: I \rightarrow \text{tlm} \quad J \rightarrow \text{tlm}

Design config rtl2tlm: I \rightarrow \text{rtl} \quad C \rightarrow A[\text{rtl2tlm}] \quad J \rightarrow \text{tlm}

Design config tlm2rtl: I \rightarrow \text{tlm} \quad C \rightarrow B[\text{tlm2rtl}] \quad J \rightarrow \text{rtl}
# Four Netlists Out of One Design 😊

## Design config rtl: I.clk → J.clk
- I.rstn → J.rstn
- I.rdata[31:0] → J.rdata[31:0]
- I.wdata[31:0] → J.wdata[31:0]

## Design config tlm: I.socket → J.socket

## Design config rtl2tlm:
- I.clk → A.clk
- I.rstn → A.rstn
- I.rdata[31:0] → A.rdata[31:0]
- I.wdata[31:0] → A.wdata[31:0]

## Design config tlm2rtl:
- I.socket → B.socket
- B.clk → J.clk
- B.rstn → J.rstn
- B.rdata[31:0] → J.rdata[31:0]
- B.wdata[31:0] → J.wdata[31:0]
Top-Down Design Flows in 2014

IEEE 1685-2014 enables parameterizable design hierarchies with conditional elements and mixed abstraction levels

- **Block diagram**
  - Views without ports
  - Register abstraction layer generation, e.g. UVM, CMSIS SVD
- **ESL design**
  - Views with transactional ports
  - SystemC TLM2 interconnect and register bank generation
- **RTL design**
  - Views with wire ports
  - Verilog / VHDL RTL interconnect and register bank generation
Summary IEEE 1685-2014

• Major update of IEEE 1685-2009 introducing amongst others
  – Parameter propagation
  – SystemVerilog expression language
  – Conditionality
  – View-specific port maps
  – Full tool interface support

• Enables parameterizable design hierarchies with conditional elements and mixed abstraction levels

• Enables flow automation for such design hierarchies
Acknowledgement

IEEE P1685 WG members

- David Courtright (Secretary)
- Joe Daniels (Technical Editor)
- Edwin Dankert
- Jean-Michel Fernandez
- Jeffery Griffiths

- Stephane Guntz
- Prashant Karandikar
- Mark Noll (Chair)
- Kamlesh Pathak
- Richard Weber

And all other Accellera IP-XACT Schema WG members
More Info

• The IEEE 1685-2014 standard (free copy)
  – http://accellera.org/downloads/ieee

• The IEEE 1685-2014 XML schema

• Additional Accellera IP-XACT material

• Accellera IP-XACT forum
  – http://forums.accellera.org/forum/30-ip-xact/
Accellera *SystemC* Standards Technical Update

Philipp A. Hartmann (Intel Corporation)
Accellera SystemC Working Groups

• Language Working Group (LWG)
  – Chairs: Philipp A. Hartmann (Intel), Andy Goodrich (Cadence)

• Transaction-Level Modeling Working Group (TLMWG)
  – Chair: Bart Vanthournout (Synopsys)

• Configuration, Control & Inspection WG (CCIWG)
  – Chairs: Trevor Wieman (Intel), Bart Vanthournout (Synopsys)

• Synthesis Working Group (SWG)
  – Chairs: Andres Takach (Mentor), Mike Meredith (Cadence)

• Analog/Mixed-Signal WG (AMSWG)
  – Chairs: Martin Barnasconi (NXP), Christoph Grimm

• Verification Working Group (VWG)
  – Chairs: Stephan Schulz (Fraunhofer), Bas Arts (NXP)
SystemC Overview
SystemC Overview (II)
SystemC Language Working Group

• **Charter:** Responsible for the definition and development of the **SystemC core language**, the foundation on which all other SystemC libraries and functionality are built.

• **Current status**
  – SystemC/TLM 2.3.1 released in April 2014
  – Currently collecting, addressing, refining proposals and errata towards IEEE 1666-201x
  – Adding extensions to the core language (e.g. as needed by other SystemC-related WGs)

• **Plans for 2015/2016**
  – Continue work on necessary standards extensions for parallelization of SystemC simulations
SystemC TLM Working Group

• **Charter:** The Transaction-level Modeling Working Group (TLMWG) is responsible for the definition and development of methodology and add-on libraries for transaction-level modeling in SystemC.

• **Current status**
  – Accellera TLM-2.0 became part of IEEE 1666-2011, PoC implementation 2.0.3 bundled with SystemC 2.3.1

• **Plans for 2015/2016**
  – Work on TLM interfaces, extensions, and guidelines to improve modeling of protocols beyond memory-mapped I/O
  – „TLM signals“; serial, bi-directional communication, ...
SystemC Synthesis WG

• **Charter:** To define the SystemC synthesis subset to allow synthesis of digital hardware from high-level specifications.

• **Current status**
  – Public review of SystemC Synthesizable Subset 1.4 completed in **July 2015**
  – Processing feedback from review

• **Plans for 2015/2016**
  – Release of standard targeted for **Q4 2015**
  – Start work on new topics for the second version of the standard
Configuration, Control & Inspection WG

• **Charter:** The CCIWG is responsible for developing standards that allow **tools to interact with models** in order to perform activities such as setup, debug and analysis.

• **Current status**
  – Requirements specification, available on Accellera web site
  – Proof-of-Concept Implementation, educational examples
  – Key improvements identified

• **Plans for 2015/2016**
  – Working to prepare a 2016 draft standard public review
  – Make identified improvements
  – Complete the Library Reference Manual (LRM)
SystemC Analog/Mixed-Signal WG

• **Charter:** The SystemC AMSWG is responsible for the standardization of the SystemC AMS extensions, defining and developing the language, methodology and class libraries for analog, mixed-signal and RF modeling in SystemC.

• **Current status**
  - IEEE P1666.1 SystemC AMS Working Group completed draft for ballot in IEEE-SA (October 2015)
  - New features under development (e.g. piece-wise-linear modeling, tracing customization, analog solver parameters)

• **Plans 2015/2016**
  - Approval of IEEE 1666.1-2016 by IEEE-SA
  - Publish User’s Guide update based on SystemC AMS 2.0
SystemC Verification WG

• **Charter:** The VWG is responsible for defining verification extensions to the SystemC language standard, and to enrich the SystemC reference implementation by offering an add-on libraries to ease the deployment of a verification methodology based on SystemC.

• **Current Status**
  – Released version 2.0 of SystemC Verification library in April 2014
  – Working on native UVM implementation in SystemC

• **Plans for 2015/2016**
  – Standardization of coverage APIs (coverage groups, bins, etc.)
  – Further explorations of needs regarding SystemC/TLM
//Taf!

Questions?

Join **SystemC Birds-of-a-Feather Meeting** today!
18:30 – 19:30, Forum 8