SystemC Community Update

NASCUG 14
October 25, 2010

Thanks to Our Global Sponsors

Outline

- Update on the OSCI organization, events and activities
- Technical Working Group updates
  - Synthesis Working Group (SWG)
  - Language Working Group (LWG)
  - Transaction-Level Modeling (TLM-WG)
  - Analog/Mixed-Signal (AMS-WG)
  - Configuration, Control and Inspection (CCI-WG)
- Video tutorials
- IEEE P1666 Update (outside of OSCI)

OSCI Membership

Corporate Members

ARM
cadence
FORTE DESIGN SYSTEMS
intel
Mentor Graphics
NXP founded by Philips
ST
SYNOPSYS

Associate Members

Actis
ATRENTA
BROADCOM
Canon
carbon design systems
CISC semiconductor
CoFluent Design
DOULOS
Fraunhofer Institut Integrierte Schaltungen
創意電子 GLOBAL UNICHIP CORP.
GreenSocs
infineon
Industrial Technology Research Institute
JEDA Technologies
Maple Design Automation
NEC
OFFIS INSTITUTE FOR INFORMATION TECHNOLOGY
QUALCOMM
STARC
Texas Instruments
UNIVERSITE PIERRE & MARIE CURIE LA SCIENCE A PARIS
XtremeEDA

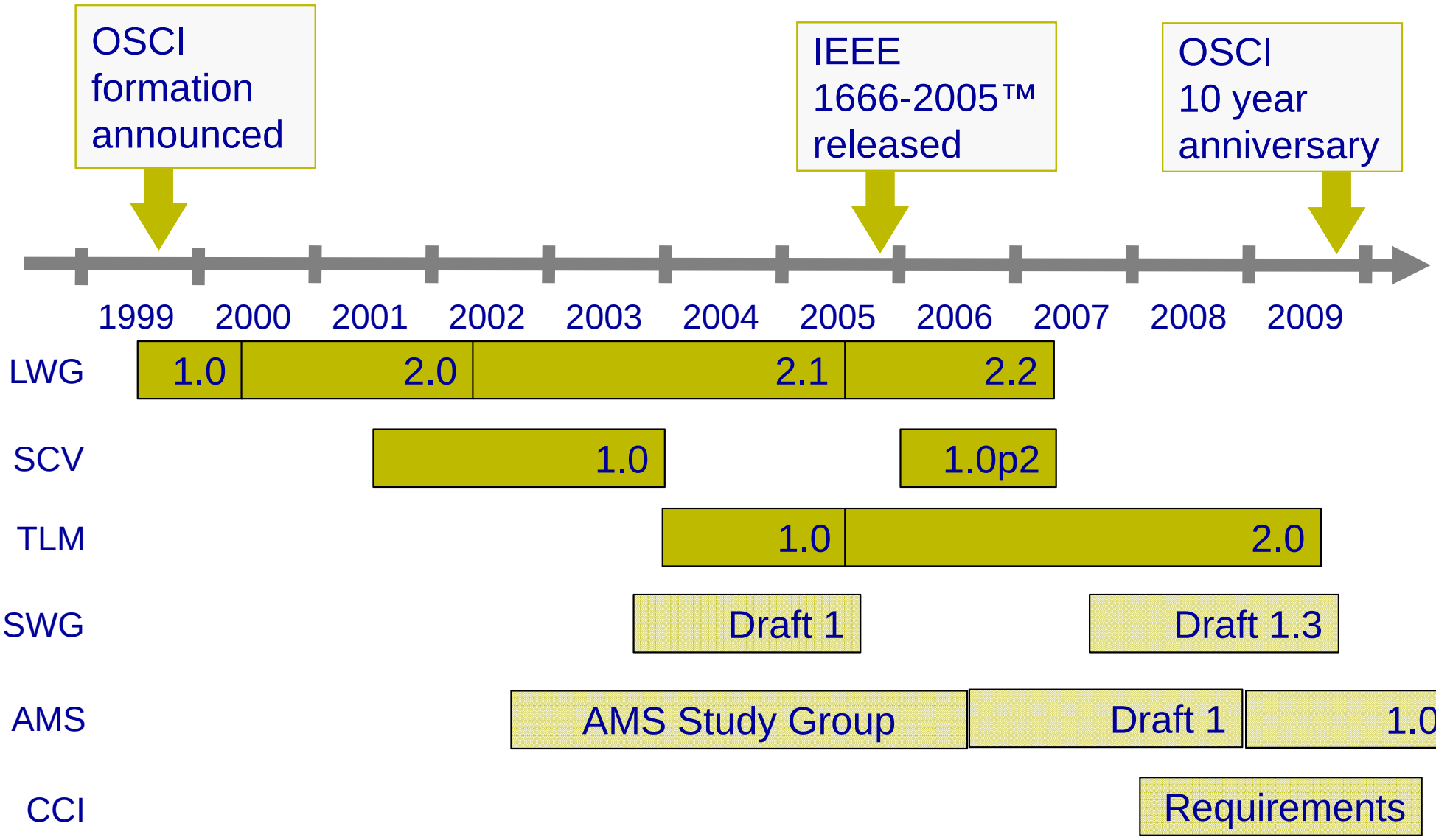
30 members

SYSTEMC™

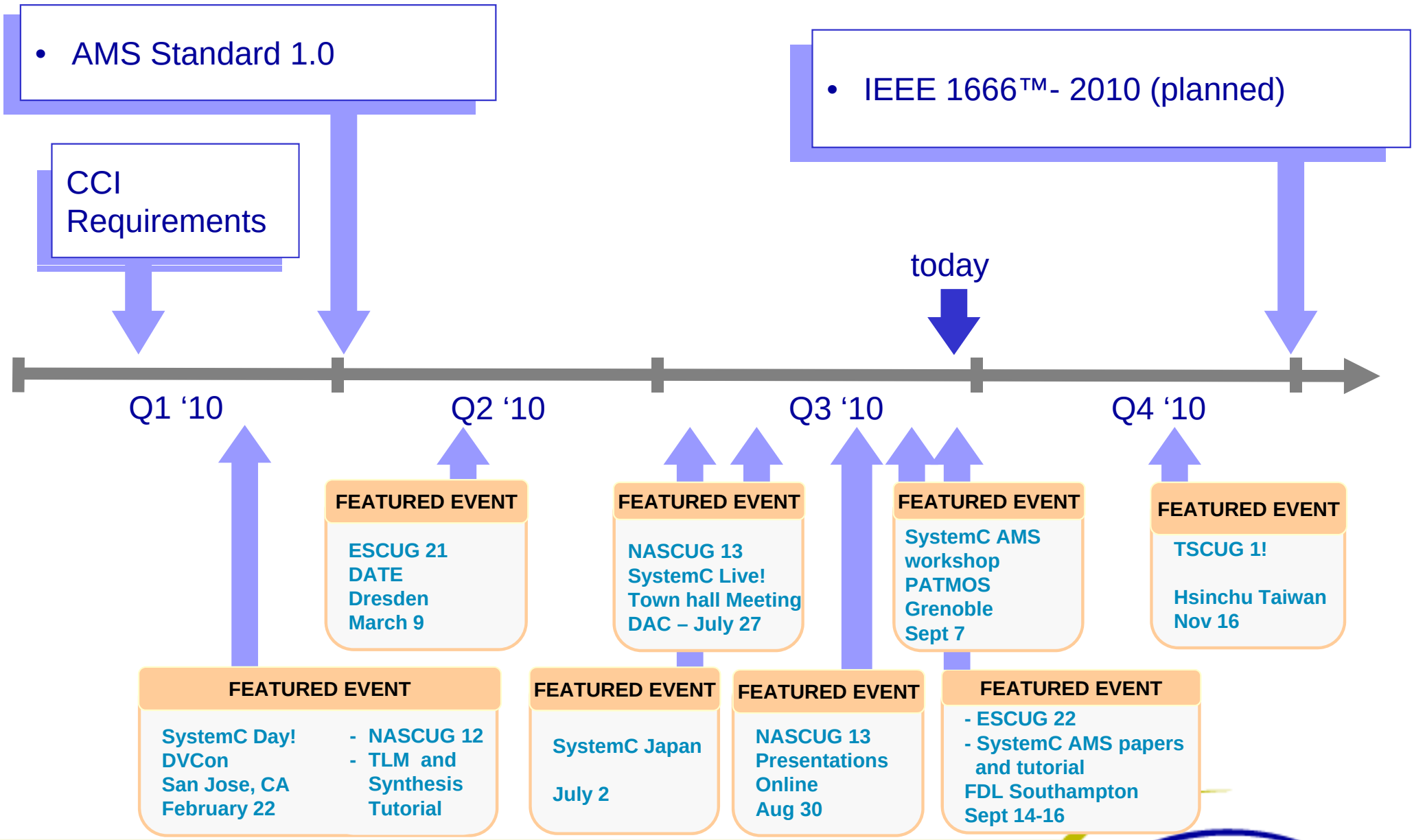
Broad-based User Community

- SystemC User groups are worldwide
  - European SystemC User Group
  - Annual SystemC Users meeting in Japan
    - Previously by JEITA
    - Starting this year by OSCI
  - North America SystemC User Group
    - New venue: ESWeek
  - Latin America SystemC User Group in Brazil
  - India SystemC User Group
  - Taiwan SystemC User Group

OSCI Standards History



SystemC Events And Activities 2010



Synthesis WG Update

- Public review of draft 1.3 completed. Updates include:
  - Supported language constructs established
  - Added section on processes, clocks, resets
  - Added discussion on abstraction levels to introduction
  - Simplified and clarified
- Tutorial at DVCon Feb 22 San Jose CA
  - Video available at: http://media.systemc.org/tlm20andsubset/index.html

Chair: Andres Takach, Mentor Vice-Chair: Mike Meredith, Forte

Language WG Update

- Interest in IEEE 1666 Standard continues strong
  - 27,793 total downloads of IEEE 1666-2005 LRM
  - 4217 downloads Jan 1, 2009 to May 31, 2009

- Current LWG activity
  - Just restarted!

Chair: David Black, XtremeEDA

Transaction-Level Modeling WG Update

- TLM-2.0 released June 9, 2008; experiencing rapid adoption
  - Includes Users Guide, implementation and examples
  - Encompasses:
    - Approximately Timed (AT) & Loosely Timed (LT) modeling
    - Temporal Decoupling
    - Direct Memory Interface
    - Debug Transactions
    - Extensible Base Protocol for modeling memory mapped buses

- Public release of TLM-2.0 Reference Manual, June 2009
  - Accompanied by TLM library: release 2.0.1.

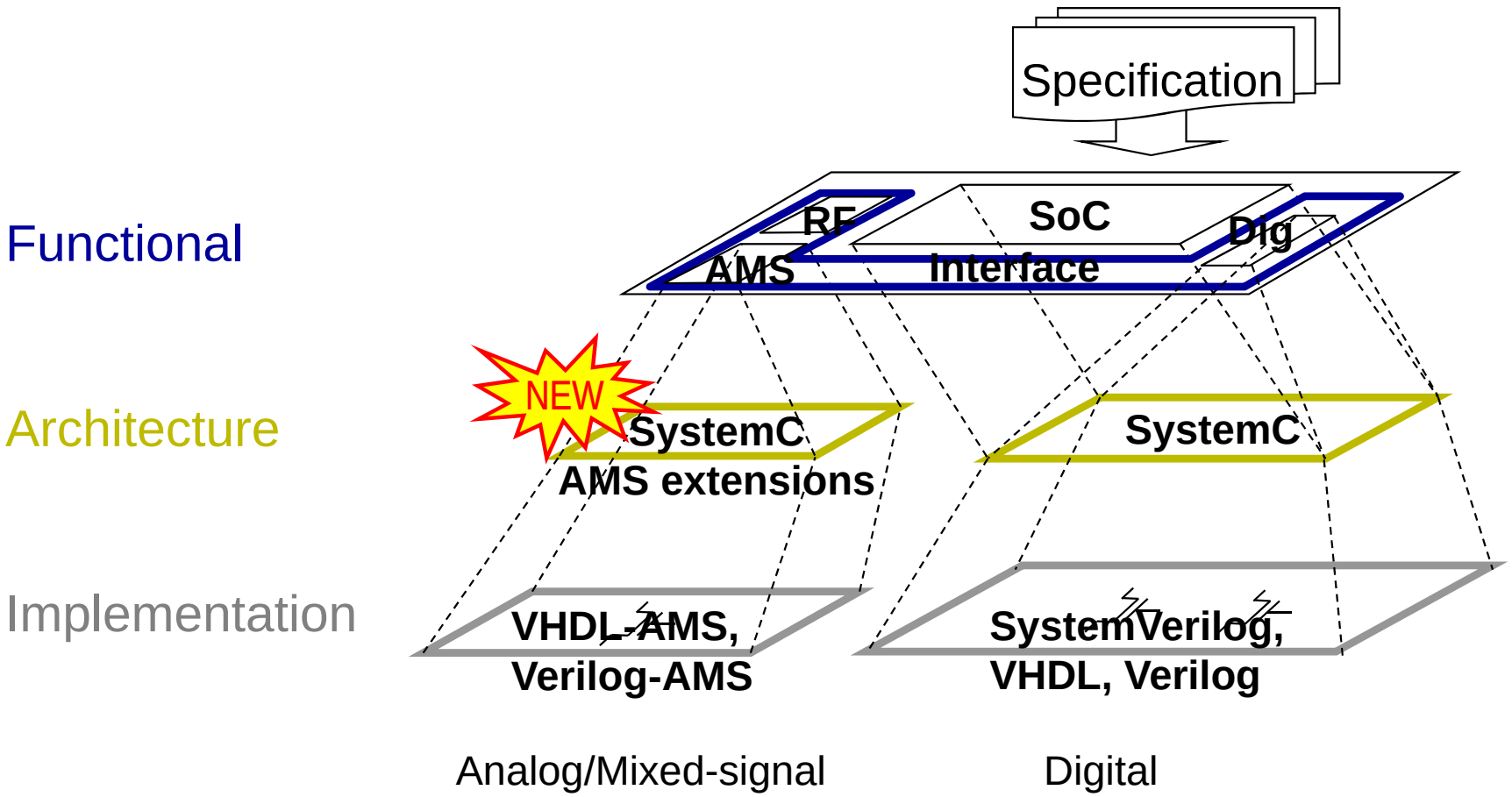
Chair: Bart Vanthournout, Synopsys Vice-Chair: James Aldis, Texas Instruments

Analog/Mixed-Signal WG Update

- Public review of the AMS Draft 1 kit completed in March 2009
  The kit includes:
  - Draft LRM and requirements specification
  - Whitepaper “An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS extensions”
- SystemC AMS 1.0 Standard released in March 2010
  - Updated LRM and requirements specification
  - *NEW*: User’s Guide including guidelines for AMS model creation and modeling strategy
- Many SystemC AMS tutorials and workshops given at conference and special events
  - In 2010: DATE, DAC, PATMOS, FDL
  - Material available via www.systemc.org

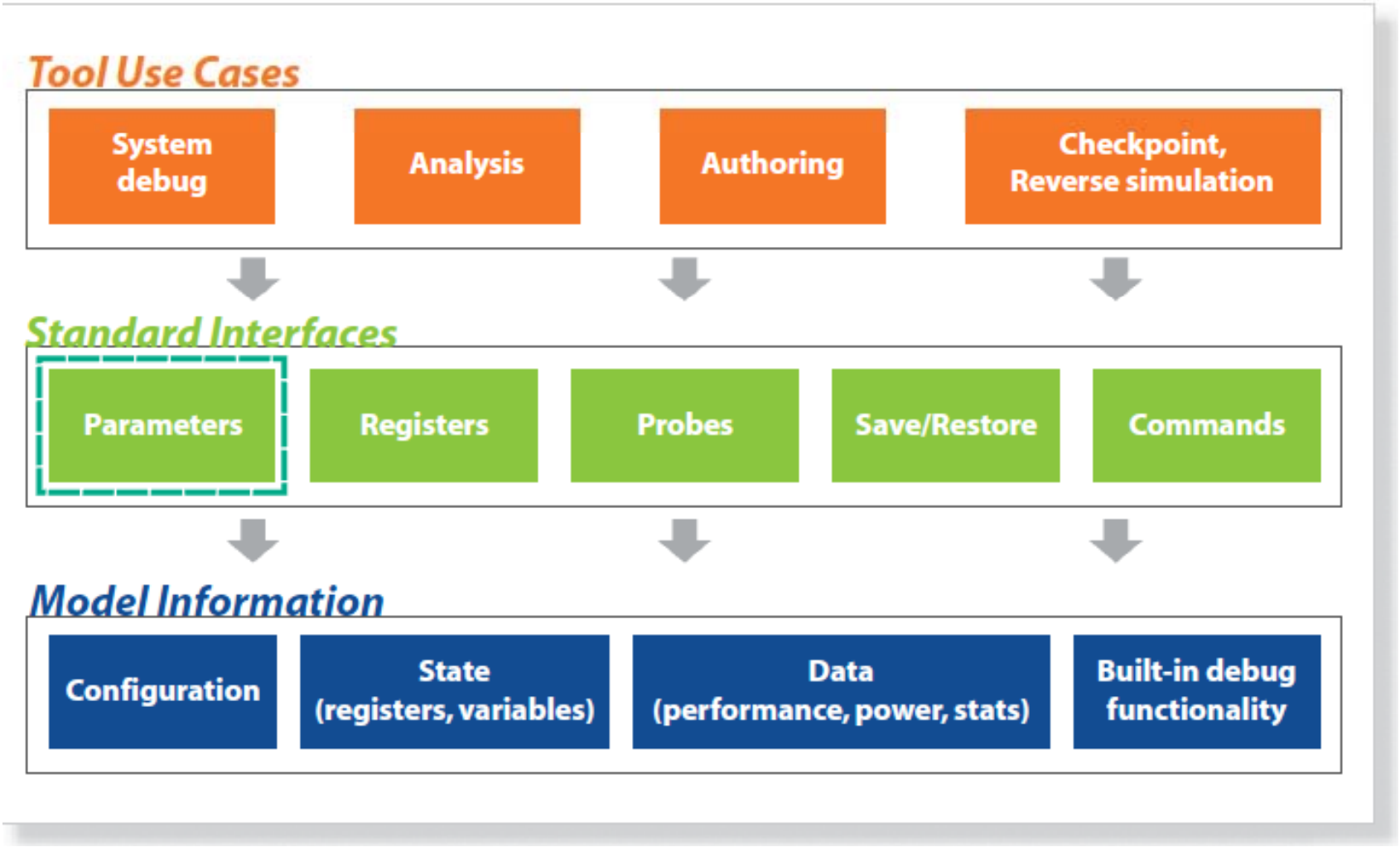
Chair: Martin Barnasconi, NXP Semiconductors
Vice-Chair: Christoph Grimm, Vienna University of Technology

SystemC-AMS Will Supply The Missing Abstraction



Configuration, Control & Inspection WG

Standardizing interfaces between models and tools



Chair: Trevor Wieman, Intel. Vice-Chair: Bart Vanthournout, Synopsys

Configuration Parameter Requirements

- Available for public review Feb. 22, 2010
- A sampling of key requirements:
  - Parameter information (type, description, default value, value's origin, etc.)
  - Parameter lookup
  - Parameter types (immutable, elaboration-time, mutable)
  - Parameter lifetimes
  - Precedence for overriding parameter values
  - Notification of parameter value changes
  - Locking parameter values
  - Parameter access control (read-only, hidden, etc.)
  - Portable parameter value encoding
- Feedback and discussion highly encouraged via the cci_forum

Video Tutorials

- OSCI TLM-2.0 Standard and Synthesizable Subset Video Tutorial. Topics:
  - The TLM-2.0 Standard
  - Current Trends in ESL/HLS
  - Why SystemC for Synthesis
  - SystemC Synthesizable Subset Draft V1.3
  - Lessons Learned Using SystemC Synthesis
  - How SystemC HLS Fits in the Design Flow

John Aynsley Doulos | Michael McNamara Cadence | Michael Meredith Forte

- NASCUG 12 Presentations
- NASCUG 11 Presentations
- Using TLM-2 Extensions for Bus Locking and Snooping
- TLM-2 In Action Tutorial
- More coming soon!

Available at: http://media.systemc.org

Summary

- OSCI continues to advance the ESL industry with SystemC standards
- Synthesis Subset draft 1.3 in public review completed
- TLM-2.0 is seeing rapid adoption across the ecosystem
  - Multiple TLM-2.0 video presentations and tutorials available at http://media.systemc.org
  - TLM-2.0 OSCI Reference Manual is now available
- SystemC AMS 1.0 released in March
- Configuration, Control & Inspection WG released requirements
  - Still time to get in on the ground floor!

Visit www.systemc.org for details!

IEEE P1666 Update

John Aynsley, Doulos

The IEEE P1666 Working Group

www.eda.org/systemc

IEEE
Celebrating 125 Years of Engineering the Future

P1666 SystemC WG

In Coordination With OSCI

Welcome to the IEEE P1666 website

P1666 Charter

The general charter of this project is to provide a C++-based standard for designers and architects who need to address complex systems that are a hybrid between hardware and software.

The specific charter of this standard is to provide a precise and complete definition of the SystemC class library including a Transaction Level Modeling library so that a SystemC implementation can be developed with reference to this standard alone. This standard is not intended to serve as a users guide or to provide an introduction to SystemC, but does contain useful information for end users.

Important Links

- IEEE Patent Policy
- IEEE SA Advanced Memebrship
- P1666 Policies and Procedures
- P1666 Roster & Meeting Attendance
- Meeting Minutes
- Meeting Schedule
- Email Reflector Archive
- Technical Email Reflector Archive
- Document Repository
- How to Join the P1666 Email Reflector

IEEE

IEEE P1666 Goals And Charter

- Charter is to update the 1666-2005 SystemC Standard
  - Errata to 1666-2005
  - Clarifications to 1666-2005 standard
  - Better formalization of TLM 1.0 Message Passing Interface
  - Add OSCI TLM-2.0 LRM
  - Selected new features
- Goal is to
  - Create/ballot draft P1666-2011 standard in during 2010
  - Send the approved standard to RevCom in early 2011
- Chair: Stan Krolikoski (stanleyk@cadence.com)

P1666 Member Entities

Accellera
Cadence
Freescale
Intel
JEITA
Mentor
NXP
OSCI
ST Micro
STARC
Synopsys
Texas Instruments

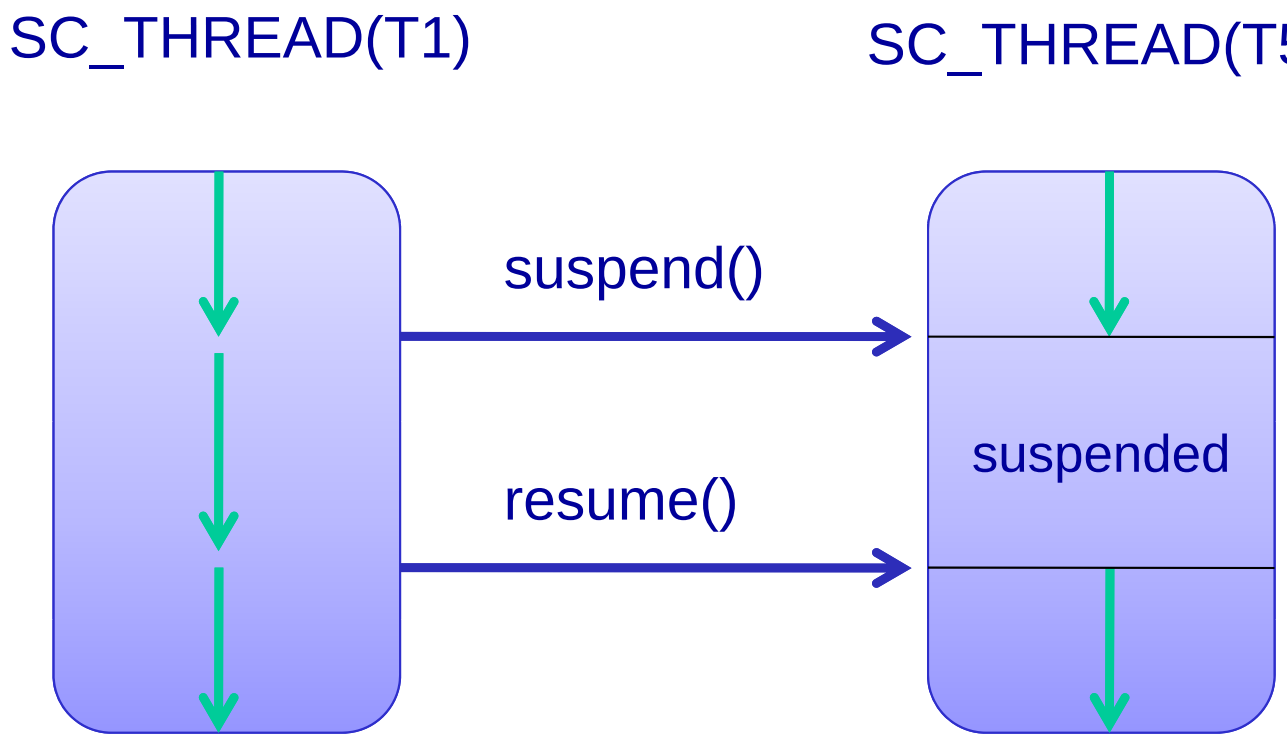
Sample Results From Technical Meeting

	Issue	+ve	-ve	Don't care
POSITIVE				
(A)	(Cadence) The Process Control Extensions, including multiple resets and asynch resets. This is the most substantial enhancement. We seem to have a broad consensus to go ahead with this proposal pretty much as-is.	12	0	0
(B)	(Jerome) Add macros to get the version number of the SystemC standard being implemented. TLM-2.0 provides such macros, but IEEE 1666-2005 only standardizes the function sc_version and sc_release.	11	0	0
(C)	(Dave Long) Add a representation of the maximum value of type sc_time. Currently there is no standard reliable way	10	0	1
(D)	(Tor) sc_event_and_list and sc_event_or_list. There is a consensus behind the idea of adding public constructors to the event list classes such that event lists representing dynamic sensitivity can be created and passed around as regular objects.	9	0	0
(E)	(Jerome) Proposal to make the bind() operators of sc_port, sc_export, and the TLM-2.0 sockets virtual, and with associated changes to the class structure to make this possible. This would make it easier to overload the bind() operators in user-defined po	9	0	2
(F)	Review the status of having multiple processes write to the same sc_signal. Strictly this is illegal in 1666-2005, but there are ambiguities concerning writes to signals from atypical places like sc_main, and the OSCI implementation does not necessarily e	8	0	3
(G)	(Philipp) Proposal to add a container class sc_vector to make it easy to create arrays- of- modules, ports, channels, etcetera.	7	0	5
POSITIVE BUT WITH SOME DISAGREEMENT				
(H)	(Tor) Adding an sc_pause function, similar to sc_stop, but from which it is possible to resume simulation by calling sc_start again. There is a related proposal on the table to add sc_get_status() function, which returns a flag to indicated whether is was	8	2	0
(J)	(David Black) Proposal to make request_update thread-safe. This would enable interaction between SystemC and external software.	7	2	3
(K)	(Jerome) Naming sc_events. Bishnupriya/Cadence has made a detailed proposal under which named events are lightweight objects that may be created during elaboration or dynamically during simulation.	7	1	2
CONTROVERSIAL OR NEGATIVE				

SystemC Enhancements

- Process Control Extensions
- Process id
- sc_pause
- sc_get_status
- Other enhancements

Process Control Extensions



Once a process is suspended, it will not run again until resumed

suspend/resume

```
module(sc_module_name  _name)  {
    SC_THREAD(T1);
}

void T1()  {
    sc_process_handle t5 =
      sc_spawn(sc_bind(&Top::T5, this));
    wait(25, SC_NS);
    t5.suspend();
    wait(20, SC_NS);
    t5.resume();
}
```

```
void T5()
{
    for (int i = 0; i < 8; i++)
    {
      wait(10, SC_NS);
      cout << ...
    }
}
```

T2 awakes at 10ns, 20ns, 45ns, 55ns, ...
Beware races

suspend/resume versus disable/enable

```
void T1()  {
    ...
    t6.suspend();
    ...
    t6.resume();
}
```

```
void T2()  {
    ...
    t6.disable();
    ...
    t6.enable();
}
```

```
void T6()
{
    wait(ev);
    ...
}
```

- Suppose sensitivity is satisfied while a process is suspended
  - On resume() process becomes immediately runnable
  - On enable() process does not become runnable

kill versus reset

```
void T1()  {
    ...
    t7.kill();
    ...
}
```

```
void T2()  {
    ...
    t7.reset();
    ...
}
```

```
void T7()
{
    // Reset behavior
    ...
    for (;;) {
        wait(...);
        // Normal behavior
        ...
    }
}
```

- kill() → Never run again
- reset() → Start again from the top
- Immediate!

sc_unwind_exception

```
void T2()  {
    ...
    t7.reset();
    ...
}
```

```
SC_CTOR(M)  {
    SC_THREAD(T7);
}
void T7()  {
    try {
      ...
    }
    catch (const sc_unwind_exception& ex) {
      if (ex.is_reset())
        ...
      else
        ...
      throw ex;
    }
    ...
```

- kill and reset throw an sc_unwind_exception

IEEE

SC_CTHREAD versus SC_THREAD

```
SC_CTHREAD(CT1, clock);
    reset_signal_is(reset, true);


void CT1()
{
    if (reset)
        Q1 = 0;
    for (;;)
    {
        wait();
        Q1++;
    }
}
```

```
SC_THREAD(T8);
    reset_signal_is(reset, true);
    sensitive << clock.pos();

void T8()
{
    if (reset)
        Q8 = 0;
    for (;;)
    {
        wait();
        Q8++;
    }
}
```

sync_reset_on/off

```
void T2()  {
    ...
    t7.sync_reset_on();
    wait( clock.posedge_event() );
    t7.sync_reset_off();
    ...
}
```

```
SC_THREAD(T7);
    sensitive << clock.pos();

void T7()
{
    // Reset behavior
    ...
    for (;;) {
        wait();
        // Normal behavior
        ...
    }
}
```

Priorities

1. kill() reset() throw_it(exception)
2. disable() enable()
3. suspend() resume()
4. sync_reset_on() sync_reset_off()

- reset_signal_is(reset, active)
- async_reset_signal_is(reset, active)

Other Goodies

- SC_CTHREAD and SC_THREAD unified
- Multiple resets
  - reset_signal_is
  - async_reset_signal_is
  - sync_reset_on
  - reset()
- handle.suspend (SC_INCLUDE_DESCENDANTS);
- handle.resume (SC_INCLUDE_DESCENDANTS);
- ...
- handle.throw_it(my_exception);

Unique Process Id

```
sc_process_handle a, b, c;
a = sc_spawn(...);
b = sc_spawn(...);
c = b;



std::map<sc_process_handle, int> m;
m[a] = 1;
m[b] = 2;
m[c] = 3;
```

a != b
b == c
(a < b) || (b < a)
!(b < c) && !(c < b)

m[a] == 1
m[b] == 3
m[c] == 3

- sc_process_handle::operator<
- sc_process_handle::swap()

sc_pause

```
struct M: sc_module {
    ...
    void my_process
    {
        ...
        sc_pause();
        ...
    }
};
```

```
int sc_main(...)
{
    ...
    sc_start();
    ...
    if (sc_get_status() == SC_PAUSED)
        sc_start();
    ...
}
```

- sc_pause() is like sc_stop() except sc_start() can be called again

sc_get_status

```
if (sc_get_status() & (SC_PAUSED |
                       SC_STOPPED |
                       SC_END_OF_SIMULATION) )
   ...
```

SC_ELABORATION
SC_BEFORE_END_OF_ELABORATION
SC_END_OF_ELABORATION
SC_START_OF_SIMULATION
SC_RUNNING
SC_PAUSED
SC_STOPPED
SC_END_OF_SIMULATION

Other Possible Enhancements

- Version number macros
- Container class to support array-of-ports etc
- Event and/or lists as proper objects
  - wait(or_list)
- Named sc_events
- Making primitive channels thread-safe