Exploring SystemC / SystemVerilog Interaction

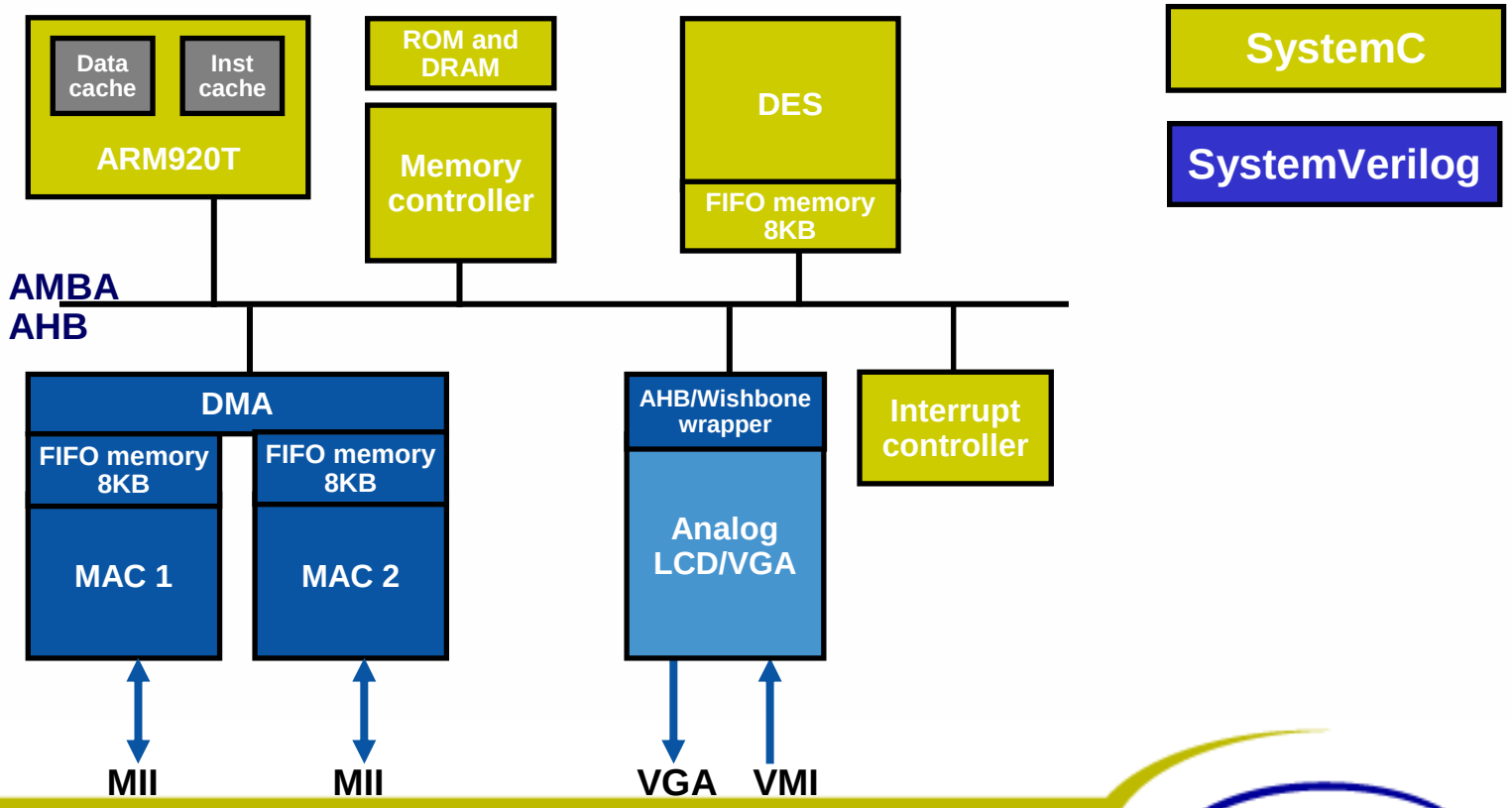
Stuart Swan
IEEE SystemC Technical Working Group Chairman
Senior Architect, Cadence Design Systems, Inc.

Introduction to SystemC

- IEEE 1666-2005
- Excels at system design and verification
- Cleanly supports
  - Transaction level modeling and verification
  - HW/SW co-design
  - SOC architectural analysis and optimization
- Based on C/C++
  - Ideal for integrating models written in C/C++
  - Easy to integrate embedded SW components
  - Easy adoption for people familiar with C++ and system design.
- Widely used

Need for SC-SV Interoperability

- Both SC and SV are being widely adopted
  - Mixed language usage is already common
  - As IP reuse increases, need for mixed language interoperability increases



Module / Signal Level Interoperability

- SC modules can instantiate SV and vice-versa
  - Single module hierarchy with mixed language components
  - Module parameters passed across languages
- SC signals can bind to SV ports and vice-versa
  - Datatypes need to be converted as they cross the language boundary
  - Standard set of conversions useful
  - Desirable to support large datatypes (structs, arrays)
- SC and SV schedulers need unified semantics
  - Mixed language systems should work as if they were in a single language
- Benefit: provides simple, clean solution for mixing RTL blocks in SC and SV
  - Same integration scheme can also be useful at higher abstraction levels

Transaction Level Interoperability

- TLM communication uses function calls, message passing
  - burst_read(char* buf, int addr, int len);
- SystemC has TLM 1.0 standard API, and upcoming TLM 2.0 standard
  - Goal is interoperability between SystemC TLM components
- Desirable to have a similar TLM API in SV
  - Goal would be TLM interoperability within SV and with SC
- Can leverage cross language fifo for transaction message passing
- SV DPI provides good foundation for cross language function calls
  - Needs to be extended to support C++/SystemC
  - Remove restriction that all calls must originate in SV
  - Possibly allow events to be passed to support mixed language TLM API
  - etc..

Some Difficult Things

- Mix arbitrary SC & SV code
- Reference SC header file directly from SV
- Reference SV package directly from SC
- SC class inheritance from SV class and vice-versa
- Manipulate arbitrary SC classes directly from SV and vice-versa

Conclusion

- Mixed language SC – SV flows will be common
  - Usage of these flows is already starting
- Several approaches to integrating SC and SV
- Standardization efforts for some of these approaches may start soon