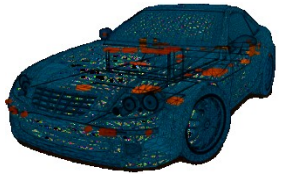
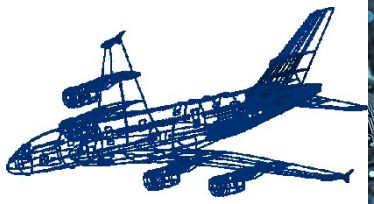
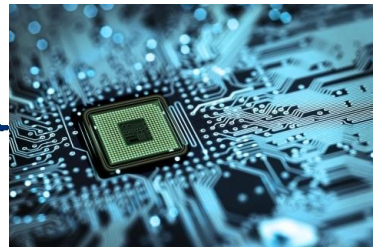
SystemC-AMS Interactive Debug and Tracing Feature

Karsten Einwich

COSEDA technologies

THE ANALOG AND DIGITAL SYSTEM LEVEL COMPANY



Interactive Debug Feature / Extendable and unifed Tracing

Motivation for the talk

- Initiate discussion:
  - Unifying SystemC and SystemC AMS functionality – so far as useful
  - Adding interactive feature
  - Rethinking SystemC / SystemC AMS tracing

Interactive Debug Feature / Extendable and unified Tracing

Motivation / History

- Enabling Interactive feature of Cadence Incisive for SystemC AMS
  - Cooperation: NXP-Cadence-COSEDA
  - -> vendor independent interface for SystemC AMS defined
  - **Can/Should be generalized for SystemC ?**
- Discussion in SystemC AMS WG to make tracing extensible
  - User defined trace formats and data types
  - Unification with SystemC tracing

SystemC-AMS Tracing

Principle

- Each object which is derived from **sca_traceable_object** is traceable
- The sca_traceable_object's will be registered to the corresponding solver
- Pure virtual methods trace_init and trace will be called by the solver to get the initialization and trace data and send them to the SystemC-AMS trace mechanism
- Trace values will be handled by an abstract value handler (a template class derived from a base class with a virtual method for printing)
  - Writing data for tabular trace uses << operator
  - for vcd the type is recognized by dynamic_cast

SystemC-AMS Tracing

Trace mechanism

- SystemC-AMS has no global time
  - Trace data, which are send to the tracing from different signals are not ordered corresponding the time
- -> Tracing contains ordering mechanism to synchronize the different traces
- Not at all signals are sample available at all time points
- Analog trace formats may require sample at all time points
  - Interpolation mechanism required to get missing sample (e.g. for double linear interpolation, otherwise hold previous value)
- SystemC AMS supports AC simulation (complex value over frequency instead time)

SystemC-AMS Tracing

Trace File

- The concrete tracing (e.g. tabular trace or vcd) is implemented by classes derived from sca_trace_file
- sca_trace_file contains pure virtual methods like write_headers and write_waves which will be called by the trace mechanism
- write_waves gets a vector for one time point which contains the corresponding abstract value handlers

SystemC AMS Tracing

Additional Feature

- Mechanism to reduce amount of data
  - Enable / disable tracing
  - Sample / decimate
  - Re-direct to different files / streams

Interactive Debug Feature / Extendable and unifed Tracing

Feature

- Getting current (at SystemC time) value of ELN nodes, TDF and LSF signals, current traceable ELN modules and TDF trace variable
  - As string (using >> operator)
  - As object of the corresponding type
- Callback if a new value is available
- Force / release values for TDF signals (ELN/LSF would structural change the equation system)
- (Deposit value)

Interactive Debug Feature

Interface in sca_traceable_object 1/2

- virtual bool register_trace_callback(sca_trace_callback, void*);
  - typedef void (*sca_trace_callback)(void*);
  - Registers callback, which is executed if a new value has been produced, returns false if the corresponding object does not support tracing – the callback is called with the given void pointer as argument
- virtual const std::string& get_trace_value() const;
  - Returns the current value (as string) of the object at the current SystemC time (sc_core::sc_time_stamp())
  - Returns an empty string, if the object does not support tracing e.g. the signal is non-causal

Interactive Debug Feature

Interface in sca_traceable_object 2/2

- virtual bool force_value(const std::string&);
  - Forces the signal with the value given by the string, the value becomes valid with the start of the next cluster period, the force only influences the read's form the signal – the written and thus traced value remains
  - Returns false, if the value cannot be forced (the string cannot be converted or the object does not support value forcing like electrical elements)
- virtual void release_value();
  - Releases a forced value, the value is released with the start of the next cluster period

Interactive Debug Feature

Additional methods for sca_tdf::sca_signal<T> + ports

- const T& get_typed_trace_value() const;
  - Returns the current value of the object at the current SystemC time (sc_core::sc_time_stamp())
  - Will be used by const std::string get_trace_value()
- void force_typed_value(const T&);
  - Forces the signal with the value, the value becomes valid with the start of the next cluster period, the force only influences the read's form the signal – the written and thus traced value remains
  - The method is used by void force_value(const std::string&)

Interactive Debug Feature

Generalization to SystemC

- Is a generalization also useful for SystemC ?
- sc_core::sc_interface methods
- Combining with sc_variant for the type independent access

Unified and extensible Tracing

Motivation

- One tracing mechanism for SystemC and SystemC AMS
- User definable trace format
- Online tracing
- Tracing „beyond“ VCD and tabular – e.g. transactions, …
- Physical units
- Abstract possibility for adding signals to traces (e.g. for getting the ability to write a function which adds all signals of the current hierarchy)

Unified and extensible Tracing

Unified Format – beyond VCD and tabular

- HDF5 ?
  - + container format should enable tracing of all SystemC /SystemC AMS / TLM data
  - + supported by different tools (e.g. Matlab)
  - - complex container format – tools may do support subset only or content organization must be known (similar to XML)
  - - not (yet) standard for EDA tools ?
  - - performance ?
- Better / Other ideas ?

SystemC-AMS Interactive Debug and Tracing Feature

Summary

- Should the interactive feature generalized (SystemC and SystemC AMS) ? -> integrated in the SystemC standard
- Should SystemC / SystemC AMS tracing unified?
- Is there a need to “rethink” the tracing mechanism
  - Other more powerful format
  - Abstract tracing
  - User defined trace formats