Reporting API

Proposal for SystemC 4

accellera SYSTEMS INITIATIVE™

SYSTEMC™ EVOLUTION FIKA

Why?

- There are a lot of proposals for ‘common’ models and components
- BUT
- Each one has a different reporting mechanism,

- Conclusion, SystemC’s reporting mechanism, as it stands, is not to everyone’s (anyones?) taste.

- Lets Modernise what we have, and make it better !

Goals

The goals are:

- Support “{std::format}” style syntax (C++20): (“hello {}”,”world”)
- Support streaming syntax : << “Hello “<<“world”.
- Provide an interface to allow run-time enabling of logging (e.g. via CCI or other mechanism)
- Be **Efficient** (e.g. a single ‘if’ that guards the reporter, and the computation of the message.
- Be independent of SystemC, but work seamlessly with SystemC.

Implementation

- Have to use (nasty) macro's in order to get the __FILE__ and __LINE__ information and provide the single **EFFICIENT** 'if' mechanism, while being easy to use.
- Provide a common macro that can take any "level"
- Provide some convenience macros for 'common' levels

- Reporting should **NOT** have any (SystemC specific) side effects.
- e.g. If you wish to sc_throw, or sc_stop, you should call those yourself.

NAMES !!!

- Some of the names we would like to use exist already (SC_LOG)
- The names should be distinct from the SystemC report macros, as they side effects associated with the report macros will NOT be applicable for the new macros
- The ‘verbosity’ levels in SystemC are confusing (SC_LOW/MEDIUM/HIGH, where HIGH is the highest verbosity, but conversily therefore the least often printed message. LOG_HIGH could mean the opposite of what most people might imagine)

Proposed names

- CRITICAL
- WARN
- INFO
- DEBUG
- TRACE

- E.g. the macros should all be prepended (to avoide conflice) e.g. LOG_
- E.g.
  LOG_WARN(()) << “A Warning”.

Inside the brackets

- LOG_WARN() : Log at the default level of logging (not run-time changeable)
- LOG_WARN(“name”) : Log with the tag “name”. You may switch “name” at run time. Note this will use a hash table lookup and may be expensive. (can be e.g. LOG(name()))
- LOG_WARN((logger)) : logger is a special object which can be instanced ‘locally’. It will cache whether the logger is enabled or not. This provides a single if, based on an constant and a locally assessible variable. The special case of (()) uses the default logger in the current class.

Some examples

```
LOG_TRACE() << "My trace message";
LOG_TRACE("top.mymodel")(”Answer is {}.", 42);
LOG_LOGGER((my_logger),”top.mymodel”);
LOG_TRACE((my_logger)) << "hello";
LOG_TRACE((my_logger),"string")("hello");
```

What Next?

- Currently being refined in the CPS working group – please join in.
- Expected to move to the Language Working Group very shortly!