2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

SystemC extension for power specification, simulation and verification

Mikhail Moiseev, Ilya Klotchkov, Intel Corporation, Hillsboro OR, USA
(*mikhail.moiseev@intel.com, ilya.v.klotchkov@intel.com*)

Kirill Gagarski, Maxim Petrov, St. Petersburg Polytechnic University, St. Petersburg, Russia
(*gagarski@kspt.icc.spbstu.ru, maxim.petrov@kspt.icc.spbstu.ru*)

***Abstract*— SystemC language is widely used to design hardware modules and whole systems on chip. Hardware development for low power applications requires power management techniques like power gating, clock gating and memory power control. SystemC does not support power related features that is a gap between power intentions at architecture level and power specification implemented in RTL. We suggest SCPower extension that allows to inject power specification into SystemC design and automatically generate UPF file. The SCPower extension provides power aware SystemC simulation that is equivalent to RTL simulation with power specification in UPF. The SCPower does verification of some power related rules and provides debugging API to access power element states.**

Keywords—SystemC; UPF; power management; low power design

I. INTRODUCTION

High level synthesis (HLS) approach and SystemC language [1] become more and more popular for development hardware modules and whole systems on chip. Low power hardware designs require power management techniques like clock gating, power gating, and memory power control. These techniques are planned at high level architecture specification and implemented at RTL in unified power format (UPF) [2]. SystemC language does not contains any features to represent power related properties that leads to several issues in verification of low power designs.

The first issue is possible mismatch between SystemC simulation and RTL simulation with power specification in UPF. In RTL simulation with UPF, in accordance with power management specified, some processes may be switched off, some inputs may be isolated and have another values. In the result, difference in the simulations may be significant.

The second issue is memory state which may be kept or reset depending on the memory power mode. There is no special memory class in SystemC language. Memory is represented with C++ arrays, which may be mapped into custom memory modules in the HLS tool. Because of that no memory power modes are supported in SystemC simulation that also causes simulation mismatches.

The third problem is manual creation of the UPF file(s) using the name hierarchy from the generated RTL file. That requires surfing through the generated RTL code, which may be difficult to read. The name hierarchy may be changed during SystemC code modification that requires manual update of the UPF file(s). The last problem is that power control requirements are not verified at SystemC design level. Any mistake is detected by simulation of the generated RTL code only, that makes debugging process longer and more complicated.

To cope with all discussed problems we suggest SCPower extension for OSCI SystemC. The SCPower extension injects power properties into SystemC designs that allows to:

- Describe power specification in SystemC design;
- Perform power-aware simulation, which is equivalent to RTL simulation with UPF;
- Automatically generate UPF file from the SystemC design code;
- Verify power related requirements to ensure correct RTL generation.

The rest of the paper is organized as follows. Section II considers other approaches to power specification, simulation and verification in SystemC designs. Section III describes the SCPower primitives. Section IV shows power-aware simulation details. In Section V we present UPF generation rules. Section VI gives some details of SCPower implementation. The last section concludes the paper.

II. Related Work

There are some works related to power management support in SystemC designs. The paper [3] proposes an approach to describe power management specification at system level. The evolution of that paper is [4] where the authors propose a SystemC extension defining macros and classes to specify power domains and power modes in SystemC. The authors are trying to provide a way to specify power management processes together with logic description, using higher than UPF level concepts.

The paper [5] proposes the PwARCH framework that allows to describe power intent in SystemC TLM using UPF concepts such as power domain, supply net, etc. Also it provides means to describe and verify different classes of contracts, which express properties of power and functional architecture. This framework is trying to abstract out some UPF concepts such as power switches. In [6] the concept of the Power Intent Model is presented which uses some terms of UPF. The power values are stored in a Power Data Model, which is related to the Power Intent Model and to a Connectivity Model. The main focus of this work is power consumption estimation at different model levels.

The work [7] provides a way to define PSM (Power State Machines) and DPMP (Dynamic Power Management Policies). PSM and DPMP are modeled explicitly and are separated from application functionalities and architectures. Binding the power model with the behavior model is performed with mapping and parameter annotations. This work does not use UPF concepts. The work [8] proposes a high level power consumption modeling for hardware IP's, based on SystemC and TLM. This work also does not use UPF concepts and proposes mixing up logic description with explicit updates to power state. The work [9] presents an approach uses functional and timed system-level model, augmented with information about power consumption: power-state models and data traffic models. It provides a way to perform fine and coarse power estimation and also coupled with temperature simulator.

In [10] the authors propose a power estimation methodology for SystemC TLM. This work is mainly focused on power consumption of SoC peripherals. It proposes an approach to augment transaction level models to perform power estimation. The work [11] also focuses on the power consumption impact of peripheral IP modules. The power consumption model is extracted from RTL description and used to augment higher level functional model with power model.

The works [12] and [13] focus mainly on power estimation of components of a cycle-accurate SoC platform. The paper [13] discusses power estimation of multiprocessor SoC's. The paper [14] proposes to use UPF specifications to extract power management assertions. This work uses local power intent extracted from UPF and global power intent described in SystemVerilog. Extracted predicates are formally verified using a verification tool. The work [15] uses UPF specification to translate it into an executable hierarchy, which is parallel to the system design. Simulation results from the system and power designs are used to automatically verify the SoC against its specification.

Power estimation, applied at system level of SoC, requires many efforts to consider library and black box components power consumption, dynamic power dissipation, which depends on signal switching activity, and others. Power estimation results, got at this level, cannot be quite precise because there are lots of optimizations and transformations done by an HLS tool, a logic syntheses tool and other tools in the design flow. To estimate power consumption of large designs it is important how the modules are placed and routed in the die. That information is also absent at the system level. Therefore we have no intention to estimate power consumption of SystemC designs.

Unlike the discussed approaches, the SCPower provides high level interface to describe low level power features with details, required by industrial design process. On the other side, our approach prohibits to create

power control logic that is not feasible at the next stages of the design flow. The important part of the approach is support of power management for vendor memory. Fine grained control of memory power consumption is "must have" feature for low power designs.

III. POWER SPECIFICATION PRIMITIVES

The SCPower extension supports main primitives specified in the UPF standard. It includes primitives to represent power domains, power switches, isolation strategies, power supply and logic signals. Also different types of vendor memory with power management are supported.

A. *Domain Structure*

In accordance with the SCPower requirements a SystemC design should have exactly one top module. The top module implementation should be inherited from `sc_top_module` instead of `sc_module`. An instance of `sc_top_module` used to mark the module as design top for UPF generation. Also the top module registers all other power related entities including power domains, memories and power supplies of child modules. The top module is considered to be placed in always on domain. To perform power control over a module, the module should be placed in a power domain.

A power domain is a collection of instances that are powered in the same way. A power domain contains one or more module instances and may contain isolation strategies. The power domain class `sc_domain` may be instantiated in the top module as well as in other modules, but nested domains are not supported. A power domain may be switched on and off with call of the corresponding methods. Power domain switching on/off is provided by a power switch, which is implicitly created for each domain except always on domain.

B. *Isolation and Power Supply*

Inputs and outputs of a module instance should be isolated if they are connected to instances from other power domains. To provide isolation for input and output ports, `sc_iso_in` and `sc_iso_out` classes are provided. These classes have the same interface as SystemC `sc_in` and `sc_out` respectively. Isolation value is specified with isolation strategy class `sc_isolation`. Isolation strategy allows to set a specified values to a group of isolated ports. Such a group is described as string of regular expression with glob syntax [16]. Isolation strategy may be enabled and disabled independently of power domain switching on/off.

The last power primitive is power supply signal `sc_supply_signal`, which specifies voltage value. Every power domain requires two power supply signals: power and ground. Memory modules may require more power supply signals.

C. *Memory*

Memory types differ with functional interface as well as with power modes supported. We consider the following memory types:

- Register file (RF), read/write memory with shutdown non-retention mode;
- SRAM, read/write memory with shutdown non-retention and sleep retention mode;
- ROM, read only memory with shutdown retention mode.

The memory implementation contains memory stub classes that represent the memory types. The memory stub has the memory specific signal interface. It provides power-aware simulation equals to RTL simulation of the memory module. After synthesis the memory stub RTL file is replaced with the memory implementation from a vendor library.

To provide function call interface, memory transactors has been developed. The read and write memory transactors provide simple universal memory read and write interfaces. They allow to make synchronous or asynchronous requests and waiting for the response. The power transactor provides simple memory power control interface. The transactor allows to switch memory on/off and also sleep on/off, if sleep mode is supported for the memory type.

D. Power Primitives Example

There is a piece of top module constructor code that shows SCPower primitive instantiation.

```
explicit soc_top(const sc_module_name& name)
{
  // Register supply signals
  register_supply(vdd);
  register_supply(vdd_core);
  register_supply(vss);

  // Domain creation
  scpower::sc_domain* d = new sc_power::sc_domain("mult_dom_1", vdd, vss, "1 ns");
  register_domain(d);

  // Isolation creation, isolation value is 0
  scpower::sc_isolation* iso = new sc_power::sc_isolation("mut_iso_1", 0);
  d->register_isolation_strategy(iso);
  // SRAM module instantiation
  sram_stub = new sc_mem::sc_sram_stub<ADDR_WIDTH, DATA_WIDTH, SRAM_TRAITS>
                                      ("sram0", vdd, vdd_core, vss);
}
```

IV. Simulation and Verification

The SCPower extension is intended to support power-aware SystemC simulation and RTL simulation with power specification in UPF file(s). In SystemC simulation power related properties of the design are provided by SCPower extension classes. It is important that these two simulations are equivalent. The design using the SCPower extension complies to the SystemC synthesizable subset [17] that allows to generate RTL with an HLS tool. To comply the synthesizable subset, the SCPower has two modes: (1) Simulation mode, used for SystemC power-aware simulation and (2) Synthesis mode, used for RTL synthesis by an HLS tool. These modes are switched by setting `__SYNTHESIS__` preprocessor constant.

A. SystemC Simulation

In SystemC simulation mode design processes can switch on and off power domains, enable and disable isolations, control power mode of the memories via power transactors. In this mode power related properties are taken into account and some power requirements checks are performed. When a domain is switched off, all processes in the domain modules are stopped to avoid changing state. If an isolation strategy is enabled, its isolated ports have the value specified in the strategy. Currently there is no control on relationship between the isolation strategy and the domain power states. The possible errors may be detected with help of SystemC simulation.

An example of the signal diagram for SystemC simulation is shown on Figure 1. There are domain power control signal `pwr_ctrl`, set into 1 when the domain is powered on, and power acknowledge signal `pwr_ack` from the domain power switch. The `ready` is an isolated output, which has isolation value 0. The isolation value is set before domain is powered off (at the time when `pwr_ctrl` becomes 0).

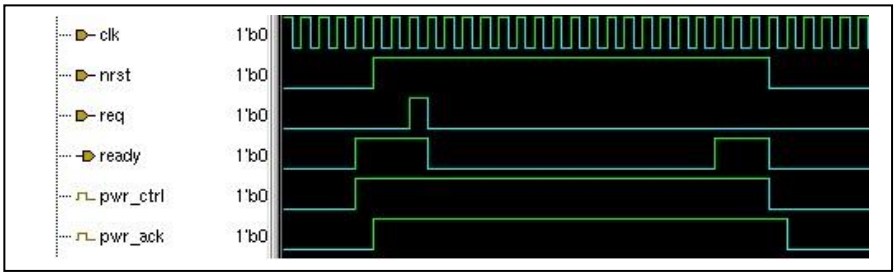

Figure 1. SystemC simulation example

B. RTL Simulation

In SystemC simulation the UPF file for the design is automatically generated (see section V). RTL simulation with the generated UPF file may be performed with one of existing simulation tools which supports UPF. An example of the signal diagram for RTL simulation is given on Figure 2. In this diagram power control signals `pwr_ctrl` and `pwr_ack` as well as isolated output `ready` are the same as on Figure 1. Also the domain power state and voltage are shown, which is provided by the simulation tool.

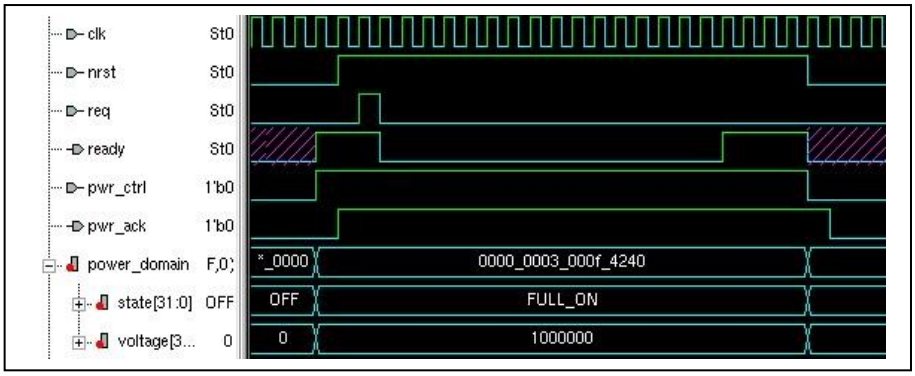

Figure 2. RTL simulation example

C. Power Debugging API

Debugging power related issues in SystemC simulation may require access power states of the SCPower primitives. To provide that, power debugging API has been implemented, it allows to:

- Iterate over domains/memories/power supplies in the top module;
- Get domain/memory/power supply by name from the top module;
- Get domain/memory state string representation;
- Iterate over isolation strategies in the domain;
- Iterate over ports in the isolation strategy.

The code that uses the debugging API is not synthesizable, so it may be used in the testbench or may be hidden in DUT code with `#ifdef`'s. Here is a piece of log which includes domain states and isolation strategies.

```
120 ns ---- power OFF Mult 0
Domain mult_domain_0:
   State: POWER_OFF
Isolation strategies:
   mult_isolations_0: enabled
Domain mult_domain_1:
   State: POWER_OFF
Isolation strategies:
   mult_isolations_1: enabled
130 ns ---- power ON Mult 0
Domain mult_domain_0:
  State: POWER_ON
Isolation strategies:
  mult_isolations_0: disabled
Domain mult_domain_1:
  State: POWER_OFF
Isolation strategies:
  mult_isolations_1: enabled
```

D. Verification of Power Related Rules

The SCPower does run-time verification of several power related rules. When a module is powered off, all its processes must be in reset. When the domain is powered up, each module is required to be in reset by any registered reset signal. If no reset signal is asserted for a module process, the corresponding warning is reported.

There is memory access control. A memory stub checks read/write access and raises warnings if some process accesses powered off or sleeping memory. To detect read before write issues after memory power up, all memory cells are filled with the special value (`0xDEADBEEF`). Figure 3 shows the situation where the domain is powered off and then it is powered up without reset assertion. In this case the SCPower reports warning:

```
150 ns, WARNING: No reset for domain module mult_domain_1 in power up
```

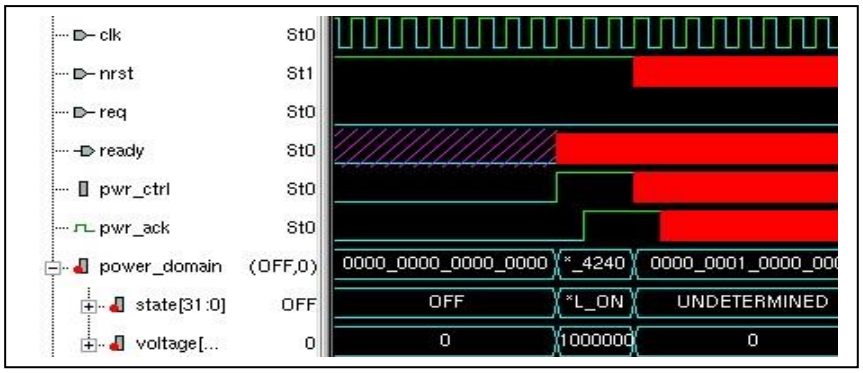

Figure 3. No reset for powered off domain

There is also some power element hierarchy and power property rules checking. The rules include:

- A domain and a memory module should have power supply connected;
- A module should be placed in one and only one power domain;
- All module ports on the domain border should be isolated ports;
- An isolated port should be registered in one and only one isolation strategy.

These rules are checked at the elaboration phase of SystemC simulation.

V. UPF GENERATION

The SCPower extension automatically generates the UPF file that contains all power features specified in the SystemC design code. Let us discuss UPF commands generated for the SCPower primitives.

The design top module (`sc_top_module` instance) is represented in UPF with `set_design_top` command. Always on domain is created for the top module with `create_power_domain` command. Supply ports and signals are created in the always on domain for all registered `sc_supply_signal` instances using `create_supply_port`, `create_supply_net`, and `connect_supply_net` commands. Primary supply signals are connected to the always on domain with `set_domain_supply_net` command.

For every `sc_domain` instance a domain in UPF is created, the domain modules are added into `-elements` option. The module hierarchical names are used. For each domain one power switch is created using `create_power_switch` command. Logic net with the corresponding name is created with `create_logic_net`, and connected to the power switch with `connect_logic_net`. The power switch output supply signal and ground signal are set as the supply signals for the power domain.

Isolation strategies for all `sc_isolation` instances are created in UPF with `set_isolation`. The isolated ports are in `-elements` option and the isolation value in `-clamp_value` option. Isolation strategies are also created for memory instances if that is necessary.

A. *UPF Example*

There is an example of UPF file generated by the SCPower.

```
# Create domain
create_power_domain mult_dom_1 -elements {mults_1}
# Create and connect power supplies
create_supply_net vdd_top -reuse -domain mult_dom_1
create_supply_net vss_top -reuse -domain mult_dom_1
create_supply_net mult_dom_s1 -domain mult_dom_1
set_domain_supply_net mult_dom_1 \
   -primary_ground_net vss_top \
   -primary_power_net mult_dom_s1
# Create domain power switch
create_power_switch mult_dom_1_sw \
   -off_state {state_off !a} \
   -on_state {state_on vcc_in a} \
   -ack_delay {o 1ns} \
   -ack_port {o mult_dom_1_ack a} \
   -control_port {a mult_dom_1_ctrl} \
   -input_supply_port {vcc_in vdd_top} \
   -output_supply_port {gtdout mult_dom_s1} \
   -domain mult_dom_1
# Add isolation for output port @ready
set_isolation mult_iso_1 \
   -elements {mults_1/ready} \
   -clamp_value 0 \
   -location self -isolation_sense high \
   -isolation_signal mult_iso1_en \
   -isolation_power_net vdd_top \
   -domain mult_dom_1
```

VI. IMPLEMENTATION DETAILS

The SCPower is implemented as a set of C++ classes, which extends the SystemC kernel. To gather information of the design structure and get access to the instance properties, we exploit “end of elaboration” and “before end of elaboration” callbacks. These callbacks are called for each module in SystemC module hierarchy from top to bottom. “End of elaboration” and “before end of elaboration” callbacks are called after the elaboration phase is completed, that provides constructors for all SystemC and SCPower objects are called and the required instances (e. g. power supplies or domains) are properly registered.

For `sc_domain` instances we exploit “before end of elaboration” callback to create the isolation strategies. In this callback a `sc_isolation` instance constructor gets the list of ports for which the isolation strategy is applied. The list of port is represented with a string, which contains elements to include and elements to exclude from the isolation strategy. The element list string supports TCL-style glob patterns and module instance names. The callback implementation traverses the modules hierarchy to find out `sc_iso_in/sc_iso_out` ports that satisfy the elements list string. We use “end of elaboration” callback to run the UPF generation routine, because it is called after “before end of elaboration” that provides the isolation strategies are already created.

Being an external to SystemC kernel, the SCPower has some issues. One of this issues is related with checking reset state for modules which are powered up. For checking reset state it is necessary to have a reset signals/ports for each process in the module, having reset event is not enough. This information is protected in SystemC kernel and cannot be obtained via SystemC API. To cope with this issue class `sc_power_module` has been added to the SCPower primitives. It should be used as a base class for SystemC modules instead of `sc_module`. The `sc_power_module` class overrides `async_reset_signal_is` and `reset_signal_is` methods to get reset list for the created process. Implementation of power features in SystemC kernel would provide more convenient API as well as faster simulation.

The other side of the SCPower implementation is support of an HLS tool. That requires the SCPower has to generate UPF file with assumptions of RTL generated by the HLS tool. An HLS tool may perform some optimization, but generally preserves the module hierarchy and intra-module signal interfaces. The important thing is support the naming convention of the HLS tool. The names of modules, signals, ports and other instances in the RTL should be determined from SystemC codes in unambiguous manner. So, here the SCPower depends on the HLS tool, and that requires some adjustment the SCPower for every HLS tool. Unfortunately, no naming convention is stated by the SystemC synthesizable subset standard. Having such rules in the standard would make easy developing and support the SCPower and other tools.

VII. Conclusion

In this paper we present the SCPower extension that allows to inject power specification into synthesizable hardware designs in SystemC language. The SCPower provides automatic generation of power specification in UPF and supports power-aware SystemC simulation. SystemC simulation with the SCPower extension gives the same results as RTL simulation with UPF. The SCPower meets the SystemC synthesizable standard, so a design with the SCPower may be synthesized with an HLS tool. The SCPower has been used in the development of a real world low power system on chip.

References

[1] "IEEE standard systemc language reference manual," http://standards.ieee.org/about/get/index.html#get1666.

[2] "IEEE standard for design and verification of low-power, energy-aware electronic systems," https://standards.ieee.org/findstds/standard/1801-2015.html.

[3] D. Macko and K. Jelemensk, "Managing digital-system power at the system level," in AFRICON, 2013, Sept 2013, pp. 1–5.

[4] D. Macko, K. Jelemensk, and P. Cick, "Power-management specification in systemc," in Design and Diagnostics of Electronic Circuits Systems (DDECS), 2015 IEEE 18th International Symposium on, April 2015, pp. 259–262.

[5] O. Mbarek, A. Pegatoquet, and M. Auguin, "Using unified power format standard concepts for power-aware design and verification of systems-onchip at transaction level," IET Circuits, Devices Systems, vol. 6, no. 5, pp. 287–296, Sept 2012.

[6] J. Karmann and W. Ecker, "The semantic of the power intent format upf: Consistent power modeling from system level to implementation," in Power and Timing Modeling, Optimization and Simulation (PATMOS), 2013 23rd International Workshop on, Sept 2013, pp. 45–50.

[7] Y. Xu, R. Rosales, B. Wang, M. Streub ¨ uhr, R. Hasholzner, C. Haubelt, and J. Teich, Architecture of Computing Systems– ARCS 2012: 25th International Conference, Munich, Germany, February 28 - March 2, 2012. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–49.

[8] H. Lebreton and P. Vivet, "Power modeling in systemc at transaction level, application to a dvfs architecture," in Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual, April 2008, pp. 463–466.

[9] T. Bouhadiba, M. Moy, and F. Maraninchi, "System-level modeling of energy in tlm for early validation of power and thermal management," in Proceedings of the Conference on Design, Automation and Test in Europe, ser. DATE '13. San Jose, CA, USA: EDA Consortium, 2013, pp. 1609–1614.

[10] V. Narayanan, I. c. Lin, and N. Dhanwada, "A power estimation methodology for systemc transaction level models," in 2005 Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'05), Sept 2005, pp. 142–147.

[11] N. Bansal, K. Lahiri, and A. Raghunathan, "Automatic power modeling of infrastructure ip for system-on-chip power analysis," in 20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07), Jan 2007, pp. 513–520.

[12] I. Lee, H. Kim, P. Yang, S. Yoo, E.-Y. Chung, K.-M. Choi, J.-T. Kong, and S.-K. Eo, "Powervip: Soc power estimation framework at transaction level," in Asia and South Pacific Conference on Design Automation, 2006., Jan 2006, pp. 8-15.

[13] R. B. Atitallah, S. Niar, and J. L. Dekeyser, "Mpsoc power estimation framework at transaction level modeling," in 2007 Internatonal Conference on Microelectronics, Dec 2007, pp. 245–248.

[14] A. Hazra, S. Mitra, P. Dasgupta, A. Pal, D. Bagchi, and K. Guha, "Leveraging upf-extracted assertions for modeling and formal verification of architectural power intent," in Design Automation Conference (DAC), 2010 47th ACM/IEEE, June 2010, pp. 773–776.

[15] C. Trummer, C. M. Kirchsteiger, C. Steger, R. Wei, D. Dalton, and M. Pistauer, "Simulation-based verification of power aware system-on-chip designs using upf ieee 1801," in NORCHIP, 2009, Nov 2009, pp. 1–4.

[16] "Glob syntax," http://www.tcl.tk/man/tcl8.4/TclCmd/glob.htm.

[17] "SystemC synthesizable subset version 1.4.7," http://accellera.org/downloads/standards/systemc.