2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

What is next for SystemC Synthesizable Subset?

Peter Frey, HLS Technologist, Mentor Graphics, San Jose, USA (peter_frey@mentor.com)

***Abstract*—High-Level Synthesis (HLS) has reached a level of maturity in which it is proven capable of doing very large designs; achieving QofR compared to hand-coded RTL while substantially reducing design and verification time and costs. However, to enable broader HLS adoption, HLS users are looking for standardized modeling guidelines and common synthesis behavior. Towards that end, in March 2016, Accellera approved the SystemC Synthesis Subset Standard. This standard establishes which SystemC/C++ constructs should be supported by a compliant HLS tool and it identifies SystemC/C++ language constructs outside of the scope of HLS. This paper focusses on key aspects of SystemC which are currently not covered by the standard and should be included to achieve truly portable models.**

Keywords---High-Level Synthesis, SystemC, Synthesizable Subset Standard

I. INTRODUCTION

The SystemC Synthesis Working Group (SWG) continues the effort of standardizing supported language constructs. The SWG is looking to add additional language constructs especially in the area of C++11 syntax as well as evaluating restricted support for unions. Standardizing these additional constructs will continue to ensure that the HLS models will be portable among all conforming HLS tools.

Keeping in mind that the end goal of an HLS standard is to enable truly portable synthesizable models, just standardizing on the support of the SystemC/C++ language constructs is not enough. In the following sections we will use three specific examples to illustrate the need to extend the SystemC Synthesizable Subset Standard to include synthesis specific semantics to enable true model compatibility.

II. SOURCE BASED SYNTHESIS CONSTRAINTS

The High-Level Synthesis process is a design process. The user determines how a specific algorithm is implemented in the final hardware. During this process the user is for example required to specify how to deal with parallelism of loops (Figure 1).

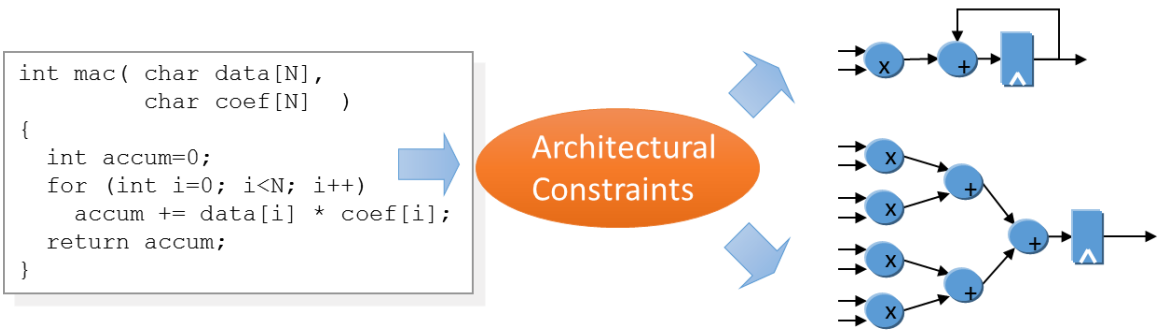

Figure 1. Architectural Transformation

Most synthesis tools provide the capability to specify the handling through synthesis constraints within the tool. However, as loops are a very common modeling construct and the desired implementation is often clear, the designer would like to "hardcode" the HLS transformation in the source. Today these constraints are different between HLS tools making model reuse problematic.

III. DESIGN LATENCY

The behavior of the design is modeled in SystemC either by a SC_METHOD, or through SC_CTHREADs/SC_THREADs. While the latency of a design is well defined for a sequential SC_METHOD (one cycle) or combinational SC_METHOD, it is variable based on wait-statements for SC_CTHREADs/SC_THREADs.

Therefore, from a synthesis perspective a SC_METHODs is simply considered an RTL like, single cycle or combinational, specification. The behavior is translated and only simple sharing opportunities can be exploited by HLS.

On the other hand, wait-statements are used in SC_CTHREADs/SC_THREADs to indicate cycle requirements. In practice, this results in wait-statements to be utilized only to model protocols, leaving the cycle budget for the behavioral description undefined. In general, this is advantageous when it comes to HLS as the tool is provided with sufficient freedom to exploit different architectures. However, if timing relationships between operations such as different protocols or memory accesses need to be upheld, the SystemC language doesn't have any means to express this relationship.

IV. Protocol Modeling for High-Level Synthesis

Despite, SystemC's support for higher abstraction modeling for simulation performance, when it comes to synthesis, protocol definitions will need to be defined through basic sc_signal read- and write- operations together with wait-statements representing the temporal aspect of the protocol. This is clearly defined by the SystemC synthesizable subset. However, the SystemC synthesizable subset doesn't define where a protocol definition starts and ends or how overlapping protocols are modeled.

If we consider the following simple handshaking example as illustrated in Figure 2.

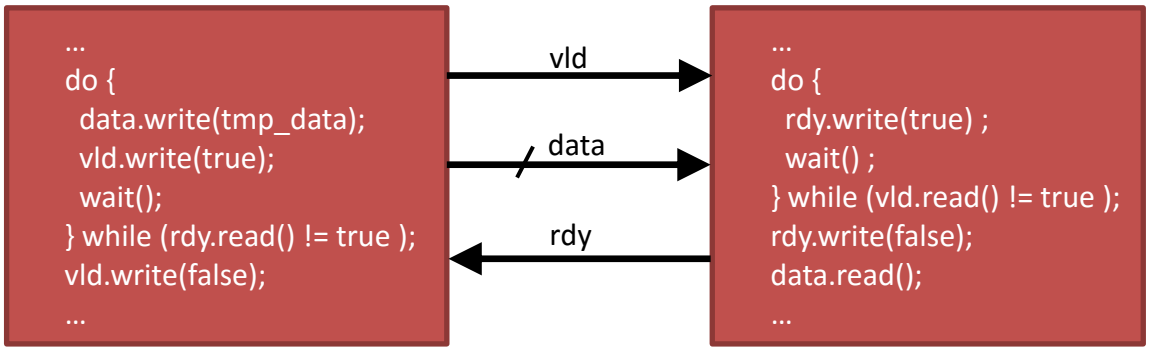

Figure 2. Simple Handshaking Protocol

Any modification to the cycle behavior will make the handshaking protocol break and the synthesis results unexpected. Although, this can be considered a special case with respect to the design latency issue, standardization on protocol specification in particular is needed. This is due to the fact that protocols are present even in simplest designs and that SystemC designers would prefer to model communication as atomic transactions to simplify modeling on different abstraction levels.

V. Conclusion

While the SystemC Synthesizable Subset standard continues to clarify the supported language subset, SystemC semantic aspects need to be standardized/defined for synthesis purposes. In other words, truly portable models as intended by the standard will not be possible without expanding the standard to include modeling semantics.