SystemC Transaction Level Models And RTL Verification

Stuart Swan
Cadence Design Systems, Inc.
2670 Seely Ave.

San Jose, CA 95134

stuart@cadence.com

ABSTRACT

This paper describes how systems companies are adopting SystemC transaction level models for system on chip design and verification, and how these transaction level models are being reused for RTL verification. The paper discusses how the task of system verification is changing as systems become more complex and it discusses how companies are striving to eliminate fragmentation within their design and verification flows by leveraging SystemC transaction level models.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids – *simulation, verification.*

General Terms

Standardization, Languages, Verification.

Keywords

SystemC, Transaction Level Model, TLM, RTL Verification, Hardware/Software Co-Design, Hardware/Software Co-Verification.

1. INTRODUCTION

The large scale and complexity of modern system-on-chip designs are forcing changes in the way that chips are designed and verified. Today's SOCs commonly contain multiple heterogeneous processors, multiple on-chip busses and caches, custom-designed hardware accelerators for dedicated functions, and multiple peripheral control devices. Embedded software is now an intrinsic part of such systems, and often the software design effort associated with such an SOC is larger than the hardware design effort. [1]

It has been apparent for some time that such SOCs cannot be designed and verified solely using traditional RTL HDL modeling methodologies. This is because RTL HDL models take too much effort to develop, simulate too slowly, and aren't available early

enough in the design flow to enable activities such as architectural exploration and early hardware/software integration. [3]

Today we are seeing increasing adoption of SystemC transaction level models for advanced SOC design and verification to address the limitations of pure RTL modeling methodologies. This paper will explain how and why major systems companies are adopting SystemC TLM methodologies, and how they are leveraging SystemC transaction level models for RTL HDL verification.

2. SYSTEMC TRANSACTION LEVEL MODELS

Transaction level models use software function calls to model the communication between blocks in a system. This is in contrast to hardware RTL and gate level models, which use signals to model the communication between blocks. For example, a transaction level model would represent a burst read or write transaction using a single function call, with an object representing the burst request and another object representing the burst response. An RTL HDL model would represent such a burst read or write transaction via a series of signal assignments and signal read operations occurring on the wires of a bus.

It is possible to create transaction level models that are fully cycle accurate with respect to the real hardware and yet which simulate much faster than RTL HDL models. [2] However in many cases designers find it advantageous to forego full cycle accuracy in order to reduce modeling effort and increase model speed.

Transaction level models have been used in various forms for many years. What is different today is the scale and complexity of systems that need to be modeled, and the emergence of widely used standard modeling languages and APIs such as SystemC TLM.

Standard transaction level modeling approaches based on SystemC are beginning to enable model interoperability and exchange within companies and between companies. This is crucial since modern SOCs rely heavily on IP reuse.

3. THE CHANGING NATURE OF VERIFICATION

Because of the size and complexity of today's advanced SOCs, systems companies are increasingly relying on IP reuse and software programmability. These changes, combined with time to market pressures, are changing the nature of chip verification efforts to some degree.

In many cases the reused IP blocks are assumed to be free of bugs, and the verification effort is instead focused on checking the reused blocks' interaction with other blocks in the system. For example a processor or bus bridge block that has been successfully used many times in the past does not need to be extensively re-verified, but the interactions between such blocks does need to be fully verified. Transaction level models and analysis tools can be very effective in verifying such interactions between blocks.

Many SOCs are targeted to embedded system applications which have hard real-time constraints. In many cases it is necessary to verify that the real-time performance goals of the SOC are met by simulating complex scenarios that involve the interaction of many hardware and software components in the SOC. For example a video encoding or decoding function within a multimedia SOC will have hard real-time constraints related to the video frame rate, and will likely involve the interaction of a diverse set of hardware and software components. Verifying such scenarios becomes an important part of the overall SOC verification process since a performance flaw in such a chip is just as costly as a logical design flaw. High speed transaction level models with the appropriate level of timing accuracy can be an excellent way to verify that performance requirements are met in such scenarios.

The intense time to market pressures and emphasis on software programmability mean that in some cases logical design flaws in hardware can be fixed via software changes and the silicon can still be shipped. So the primary verification effort may be focused on detecting hardware design flaws that would prevent silicon from being shipped at all. Hardware design flaws that might occur but which can be fixed via software might not be fully verified prior to silicon.

Even as the nature of verification for today's SOCs changes in some respects, nevertheless verification of newly designed or configured RTL blocks remains a crucial, time-consuming part of the overall SOC design flow. While newly designed RTL blocks may only constitute a fraction of the overall SOC, the complexity of such blocks and the potentially high costs of uncaught design bugs mean that these blocks must still be extensively verified. For such blocks we continue to see increasing adoption of advanced verification techniques such as coverage driven verification, constrained random stimulus, assertion based verification, formal verification, and hardware-assisted simulation acceleration and emulation.

4. LEVERAGING TRANSACTION LEVEL MODELS FOR RTL VERIFICATION

Since any model development activity requires significant effort, it is desirable to leverage models to the maximum extent throughout the design flow. Naturally designers wish to reuse the transaction level models they have developed for an SOC to assist the process of RTL verification. For example, the designer might wish to insert the RTL HDL block into its SOC context in the SystemC TLM model to verify proper operation in the system level context. Alternatively the designer might wish to use individual transaction level components such as transactors, stimulus generators, response checkers, monitors, and reference models to construct a specific test bench for an RTL HDL block.

There are two keys to enabling SystemC TLM models to be leveraged for RTL HDL verification.

First, a simulator which supports co-simulation and co-debug of mixed SystemC and HDL models must be used. Today all major simulators support mixed SystemC and HDL simulation in order to enable this type of model reuse.

Second, proper transactors must be developed which enable the RTL HDL model to be interfaced with the SystemC TLM model. The modeling effort needed to produce such transactors will be dependent on the nature of the design and the level of abstraction of the TLM test bench which the RTL block needs to interact with. In general we have seen that design teams that have developed SystemC transaction level models have been successful in developing such transactors to enable RTL HDL blocks to be co-simulated with the SystemC models. As always, good model coding practices and a focus on developing TLM components with an eye toward modularity and reuse is helpful.

A key advantage of SystemC TLM models is their simulation speed. It is typical for SystemC programmer's view TLM models of SOCs to execute in excess of 1 megahertz, and even properly coded cycle accurate models of SOCs using SystemC TLM can execute in the range of 100 kilohertz. RTL HDL models of even modest sized blocks, on the other hand, typically execute at less than 1 kilohertz when using software simulators.

Of course when RTL HDL models are co-simulated with SystemC TLM models using software simulation the significantly slower simulation speed of the RTL HDL model will usually cause the overall execution speed to be in the range of 1 kilohertz or less. In some cases this is acceptable, and effective RTL verification can be achieved at these speeds. In other cases users have a very strong desire for higher simulation speeds.

Today we are seeing increasing use of hardware-assisted simulation acceleration for RTL HDL models which are simulated together with SystemC TLM models. By leveraging industry standard interfaces such as SCEMI to integrate SystemC TLM test benches with RTL HDL blocks that are simulated within hardware accelerators, users are able to achieve overall simulation speeds in the range of 100 kilohertz.

5. UNIFYING THE DESIGN FLOW

Perhaps the greatest opportunity to optimize the overall design and verification flow for modern SOCs is to enable the exchange of executable models between different types of engineers. In the past, system architects, embedded software engineers, and hardware engineers often did not exchange executable models. Often the only form of communication between such groups involved paper specifications, which were frequently ambiguous or incomplete. Delays and bugs resulting from miscommunication and misunderstandings were common.

The adoption of SystemC TLM as a means to exchange executable models between these diverse sets of engineers is helping to eliminate these problems and has been proven to significantly reduce the overall SOC design time. [3] In a sense, the exchange of executable models forces different types of engineers to work closely together early in the design flow to insure that the SOC will work.

Companies striving to eliminate fragmentation within their SOC design flow are seeing the benefits of using design tools that provide unified modeling capabilities across different languages and abstraction levels. For example, tightly integrated simulation tools supporting mixed SystemC and HDL modeling enable different types of engineers to understand and resolve problems across abstraction levels and language boundaries. Mixed language debugging and analysis capabilities are proving to be essential tools in such unified design flows. Such tools enable different engineers to use the best tools and languages for their particular jobs while still fully leveraging each others' work.

6. CONCLUSION

This paper has introduced SystemC transaction level models and shown how companies are adopting these models in their advanced SOC design flows. Transaction level models are enabling systems companies to perform early architectural exploration and hardware/software integration, and to unify their overall SOC design and verification flow. This paper has also shown how transaction level models are being leveraged for RTL verification.

It is clear that in the future SOCs will continue to increase in complexity, and that we will see increasing IP reuse and reliance on software programmability. These trends are likely to drive increasing adoption of SystemC transaction level modeling techniques.

7. REFERENCES

[1] Chang, Cooke, Hunt, McNelly, Martin, Todd. *Surviving the SOC Revolution: A Guide to Platform-Based Design*. Kluwer Academic Publishers, Boston, MA, 1999.

[2] Grotker, Liao, Martin, Swan. *System Design with SystemC*. Kluwer Academic Publishers, Boston, MA, 2002.

[3] Ghenassia, F*., Transaction Level Modeling with SystemC*. Springer, Dordrecht, Netherlands, 2005.