SYNAPSE design

Position Paper

Abstractions and System Design Concepts

Executive Summary

The design process traverses a number of levels of abstraction, initially starting at a very high level and transitioning to the detailed implementation, as described below:

- User & Marketing Requirements
- Golden Reference Model
- Transaction Level Model (TLM)
- RTL
- Gate/Physical Implementation

Each level in this abstraction hierarchy has various methods to:

- Capture the design intent at that level in an appropriate format or language
- Verify the conversions from the higher level to the current level have been made correctly
- Validate the current level of abstraction is accurate and continues to meet the design requirements

The discussion below describes in some detail the various levels of abstraction as defined above, identifying the formats, languages and methodologies in present use; the methods to verify the conversions; the issues/concerns in traversing from one level to the next in the abstraction hierarchy; plus future trends.

Synapse Design
150 S Almaden Blvd, Suite #1380, San Jose, CA 95113 Tel: 408-850-3640 Fax: 408-516-9416

Discussion

From Requirements to Golden Reference

A typical system design starts by gathering user/market requirements. These requirements have three dimensions –architecture, function and performance. With that in mind an architect converts these requirements into specification. Initial specification, although somewhat cognizant of the three dimensions, does not necessarily fully capture all the requirements. These specification documents represent first abstract representation of the system and may exclude some of the architectural details or suggestions as to what such a design would look like structurally.

In some cases, the specifications could be a pure algorithm, which only captures the functions. Initial abstract model could be purely functional and modeled using any of the high level programing languages or it may use tools (such as Matlab or other) for modeling algorithms for certain classes of design.

For SOC designs that we see today, the next refinements could add more architectural details that could be easily modeled. These models begin to include the structure and organization that can be eventually translated into more detailed models as we go down the abstraction hierarchy. Today the SystemC/TLM2.0 standard allows us to create models that can capture the design at these levels of abstraction. It is possible to create such models that can be incrementally refined to add the third dimension of our requirements, that is, performance. Timing details can be added to these modularly created models. More details mean more accuracy, but slower simulation.

At this stage various architectural options can be explored to optimize function and performance. We need a methodology that allows us to isolate and fix any issues at various levels and propagate at higher level of abstraction. If left unidentified in early stage, it can cause costly re-spins late in design cycle.

The tests and software developed at this stage must conform to the requirements of the specification document. Since this model/test will be our golden reference as we move down the hierarchy of abstraction; a rigorous design review is needed to ensure that these models and test do represent the user requirements as best as it can. It is best to find the

discrepancy at this stage to keep the cost down and allow the golden reference to be truly golden. In practice, however, there are issues that come up late in the design cycle or scope changes that come in late in design cycle. In such cases, we pay the cost and fix the golden reference. Some of it is unavoidable, but we want to keep this to a minimum.

Multiple Abstraction Level in Single TLM Representation (SystemC/TLM2.0)

The ability to navigate multiple levels of abstraction from a purely untimed functional model, to loosely timed, to approximately timed and then finally to cycle accurate models, all in a single representation, is a big plus. The big issue as we refine our models and add details is that we want to keep the design that is already verified at a more abstract level intact. Since, all levels of abstractions are in TLM2.0, all the tests and software that we might have written for the more abstract models should still run on the refined models.

Another practical aspect of modeling in TLM2.0 is that one can mix abstraction models. One block can be cycle accurate; another can be loosely timed and so on. This hybrid approach is very suitable and optimizes the development schedule. It allows for reuse of previously developed and verified models.

An IP library can be developed and readily available to support various abstraction levels.

Same SystemC simulators run on all abstraction levels.

SystemC is not the only representation available today to describe the high level representation. However, if we use a different representation, we have to solve the verification problem when we move to different abstraction level.

RTL: Toward the final Implementation

In fact, as we move to RTL representation, where full cycle accuracy is present, it's the most accurate and closest to implementation. Typically, this phase is often directly translated from detailed microarchitecture specification from which a golden reference is developed. To bridge the gap between tools that utilize the golden reference and that run RTL, a common suite of tests is used. The common suite needs to be as complete as possible.

If golden reference is in C++ or SystemC, results produced running them are compared to results produced running the same tests in RTL. Otherwise, golden reference models are

sometimes created using UVM-like approach in System Verilog. In that case the language of the testbench and RTL are both System Verilog; again the same representation. This dynamic simulation is supplemented by presence of assertions and formal tools that ensure compliance to protocols at RTL level or some standards as defined by microarchitecture.

In any case, for these reasons, test plans and coverage become very critical. They represent all the compliance to specification. Again, all golden reference model/tests must be critically reviewed by the design and architecture teams to ensure that they do represent the intent.

Beyond RTL

Beyond RTL, as we move toward gate implementation there are tools that can provide static equivalence checking (formal) and static timing analysis. This suffices to verify that the new details added have not violated the functions implemented at the RTL level. Key is to ensure that "*nothing is lost in translation*". Dynamic simulation still plays a part as a more thorough check for portions of some designs, but since simulations are much slower static tools provide the necessary guarantee.

RTL is often translated into FPGA and emulation, where more speed is available and one is able to run the software that might have been created at TLM level. Some of our customers even create hybrids of TLM and emulation in a single platform to leverage best of both worlds.

What's missing in the flow today?

While Dynamic Simulation can run common software and common tests all the way the various levels of abstraction; there are limited ways to ensure the equivalence at the TLM model and RTL. For certain classes of design, a synthesizable model is created at TLM level (SystemC) that needs to be equivalence checked against the RTL it produces. There are tools coming into the market that help here as well. However, the classes of designs that can be synthesized are limited. Common assertions can also provide some compliance checks between two abstract models. As this area matures in years to come, TLM

abstraction may become as mainstream as RTL, but these tools need many more success stories before everyone jumps onto the bandwagon.

Synapse Design Capabilities

Synapse Design assists our clients to navigate the most optimal way to complete a system design from specification to implementation. We help them create specifications, abstract models as needed and provide verification methodology that is right for each design. We have designed many chips from spec all the way through to packaged, tested parts, with an impeccable history of silicon working right the first time.