



November 24, 2006

Speed hardware development with Model-Based Design

Model-Based Design cuts design time by speeding system-level simulation. It also slashes verification time by providing an executable specification, and by linking this spec to the final HDL models.

By Sudhir K. Sharma, The MathWorks, Inc.

Abstract

It should not come as news or a surprise to engineers that design cycles are short, product cost is an issue, and getting it right the first time is still the goal. It is also well-known that the ever changing standards, increased design complexities, high NRE and design re-spin charges, and lack of seamless tool flows are making it that much harder to meet the time-to-market and cost objectives.

In particular, for implementation of signal processing systems on FPGAs and ASICs, engineers are well aware that there is significant overlap between system design, HDL design, and verification tasks. Yet, tools to seamlessly integrate these tasks are lacking. For example, system engineers and HDL engineers are still engaged in a manual 'bit-true' verification methodology. This is often a laborious, time consuming, and error-prone activity involving file exchanges between the system designer and the HDL designer. Geographically diverse teams face even bigger challenges in this regard since the system designer and the HDL designers are not sitting in the same location.

What is needed, then, is a design methodology that bridges the world of system designers, the HDL designers, and verification engineers to increase productivity and produce 'correct-by-construction' designs that match the system specification.

This paper will illustrate how the process of Model-Based Design provides the benefits of top-down and bottom-up design methodologies to help quickly achieve the design objectives. Using the concept of 'golden design specification,' the paper will discuss how Model-Based Design streamlines both design and verification of HDL implementations.

Some Statistics

As the complexity of designs has increased, the system designers have moved testability to the forefront of the design cycle. Today, most design specifications require verification and test methodology to be included as part of the original design document. This is a good trend.

As Figure 1 shows many of the flaws that are introduced in the design specification are captured at a much later stage in the design cycle. [1]

Finding and fixing design flaws early in the design cycle would clearly lead to much improved designs and the likelihood of designs that reach their target market on time.

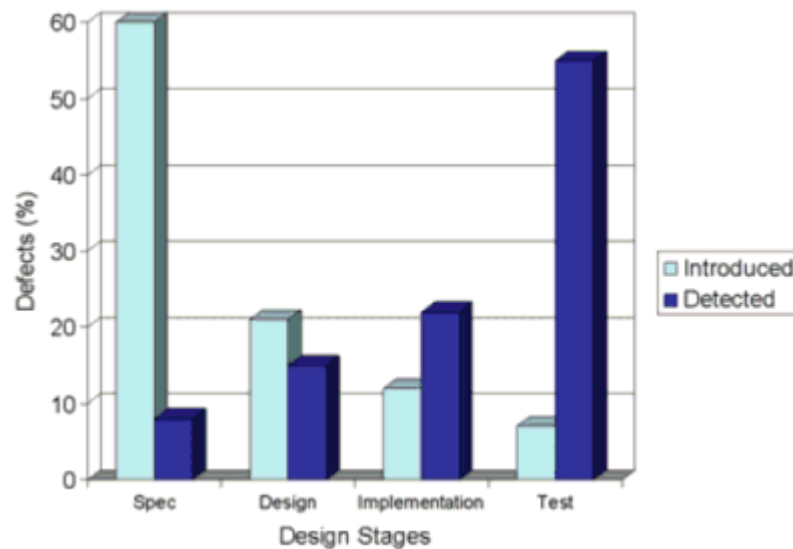


Figure 1. Errors are introduced early in the design process but caught much later. [1]

In a recent EE Times survey, it was noted that system design consumes 18% of the design schedule, followed by 20% time spent in HDL design, and 28% time spent in functional verification. [2] That represents nearly 2/3 of the design schedule. It would seem natural, then, to look for productivity gains early in the design process.

About Design Methodologies

Real productivity gains can be realized by choosing the right design methodology. Software designers quite often use top-down design methodology to quickly create a working model of their design. In the early days of IC design, hardware engineers also used top-down design methodology.

In top-down design methodology, individual blocks are designed and verified within the context of the top-level. This is beneficial because the block level interfaces and functionality are both verified at the top-level. In this design flow, there is only one test bench and the number of synthesis and simulation scripts is minimized. Moreover, there is the added benefit of script portability across projects.

However, the nature of top-down design is such that the entire design can only be simulated at the top level and block-level verification done at the top-level is an inefficient way to use verification time. In addition, the HDL simulation speed is very slow and powerful machines are required to achieve decent simulation time.

Driven in part by the desire to speed-up simulation time and part by the desire to verify blocks before they are used, engineers adopted bottom-up design methodology. In bottom-up design methodology, the blocks are designed and verified individually before they are integrated together to create the top level system.

Experienced engineers will recognize that bottom-up design introduces a lot of rework into the design process and, in most cases, the test structures that are built to verify individual blocks are not reusable for top level verification. Moreover, the rework shows up at the most inopportune time. That is, it often becomes necessary to redesign blocks when the top level system is put together and engineers realize that the blocks don't integrate as specified in the original specification.

Two engineers reading the same specification may have different interpretation or the specification may have been incorrect to begin with. Whatever the cause, by the time top level integration takes place, schedule deadlines become increasingly critical. At this point either the offending blocks, which have been painstakingly verified at the block level by the designer, must be redesigned or a 'patch' is put in place. Neither is an optimal solution.

Clearly, to counter this increased system complexity and compressed design cycles, a new methodology is needed that captures the benefits of top-down and bottom-up design methodologies. Recent advances in co-simulation technologies have enabled fast simulation time and verification of individual blocks in a system environment. The rest of this paper discusses how engineers can implement a practical and scalable design environment that offers the combined benefits of top-down and bottom-up design flows using MATLAB, Simulink, and Link for ModelSim.

Model-Based Design

Model-Based Design improves design quality and eases the verification burden by creating a 'golden executable specification' in Simulink. This specification is then used to create and verify the hardware design. The significant advantage of Model-Based Design is that it moves the verification process all the way to the beginning of the design cycle and thus helps detect system specification related errors, design errors, and implementation errors early. (See Figure 1)

The design flow using Model-Based Design captures the ideas of design reuse, design verification, and co-simulation to produce designs and code that are 'correct by construction'. In addition, Model-Based Design provides an abstraction level that is easy to understand. The Model-Based Design flow keeps the design at the algorithm level until it is completely verified. It allows design co-simulation with HDL simulators and allows seamless mapping to existing hardware design flows.

Design Space Exploration

As an example, consider the design of an 802.11n wireless networking system. Using the approach of Model-Based Design, a system designer can capture the specification in Simulink. Each of the blocks that are part of the 802.11n draft specification can be designed and verified in Simulink. As the draft specification changes, the design can be modified easily. The pieces of IP can continue to be reused and modified easily to meet the final 802.11n standard when it is ratified.

More importantly, notice that using Model-Based Design does not involve manual edits to HDL code. By staying at the architecture level, the Model-Based Design approach allows the engineers to simulate designs much faster than would be possible with HDL simulations. This is a very useful capability that significantly enhances productivity. Since the design is captured and simulated at the model level, time consuming code edits and HDL simulations are completely avoided at the system design stage.

Co-simulation

When the Simulink model of the system is completed, the system designer can distribute the Simulink model to the HDL designer as the 'golden executable design specification'. The HDL designer then uses this specification to create the equivalent HDL. Notice that the interaction between the system designer and the HDL designer is no longer input/output files or text specifications. Rather, the system designer is providing a functional model that the HDL designer can execute to understand how the design is expected to function.

Additionally, the Simulink model is also a ready-to-use test-bench that the HDL designer can plug his/her design into and ensure that the HDL produces the same result as the system specification. Furthermore, all the test files that the system designer used to verify the Simulink model can be used by the HDL designer to ensure functional coverage of the HDL code per the system design specification.

At this point in the design cycle, the HDL designer can start inserting the equivalent HDL blocks into the Simulink model and co-simulate the design using the Link for ModelSim tool.

Signal Processing Design & Verification

By using the Signal Processing Toolbox, Signal Processing Blockset, systems engineers can quickly architect and verify their signal processing algorithms in MATLAB and Simulink. The Link for ModelSim (Figure 2) tool now enables the systems engineers to co-simulate their system model with its equivalent HDL representation.

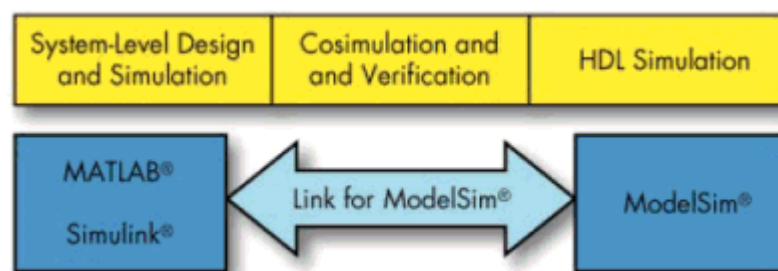
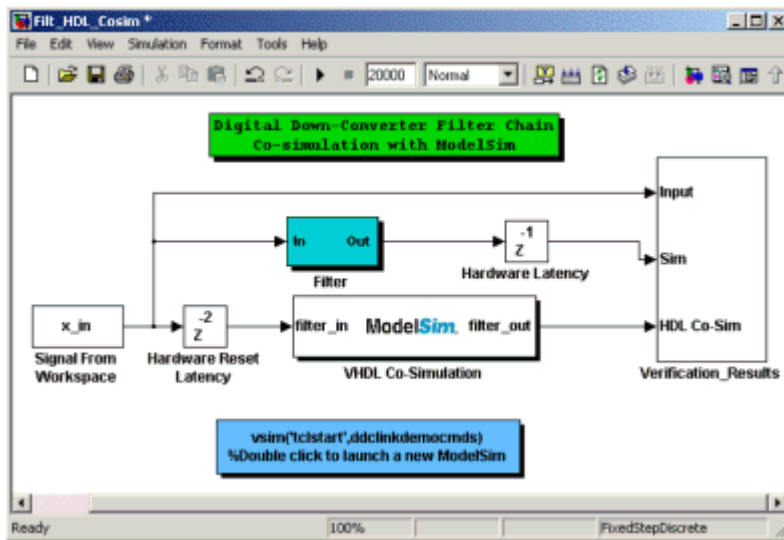


Figure 2. Link for ModelSim Co-Simulation Environment

The Link for ModelSim tool provides a high speed, bi-directional interface between the popular ModelSim HDL Simulation engine from Mentor Graphics and MATLAB & Simulink system modeling tools from The MathWorks.

A Simulink Example

The Simulink model of a Digital-Down-Converter (DDC) Filter below illustrates the co-simulation and top-down design methodology being proposed in this paper.



[\(Click to enlarge\)](#)

Figure 3. Simulink model of a Digital Down Converter (DDC) Filter Chain

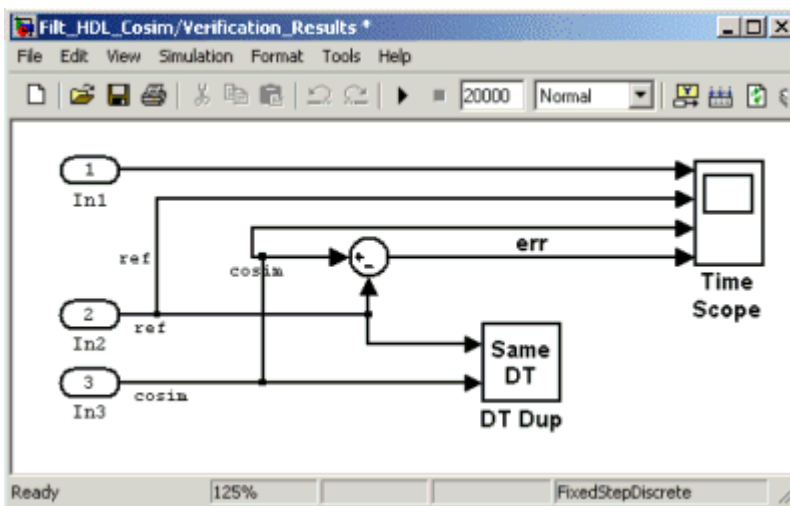


Figure 4. Inside the Verification_Results block.

Figure 3 above, shows two implementations of a DDC filter block. The fixed-point Simulink model of the DDC filter block is shown as 'filter'. The equivalent HDL representation is the block labeled as VHDL Co-Simulation ModelSim block. Using the same input, both Simulink and VHDL representations produce output that is then compared in Simulink via the Verification_Results block, which produces an error vector indicating the difference between the fixed-point and VHDL results. (See Figure 4)

Figure 5 shows the input, the fixed-point filter output, the VHDL filter output, and the error between the fixed-point output and the VHDL output. The error is the flat-line indicating no differences between the two outputs. Finally, Figure 6 highlights the point that signals are also visible in ModelSim waveform window.

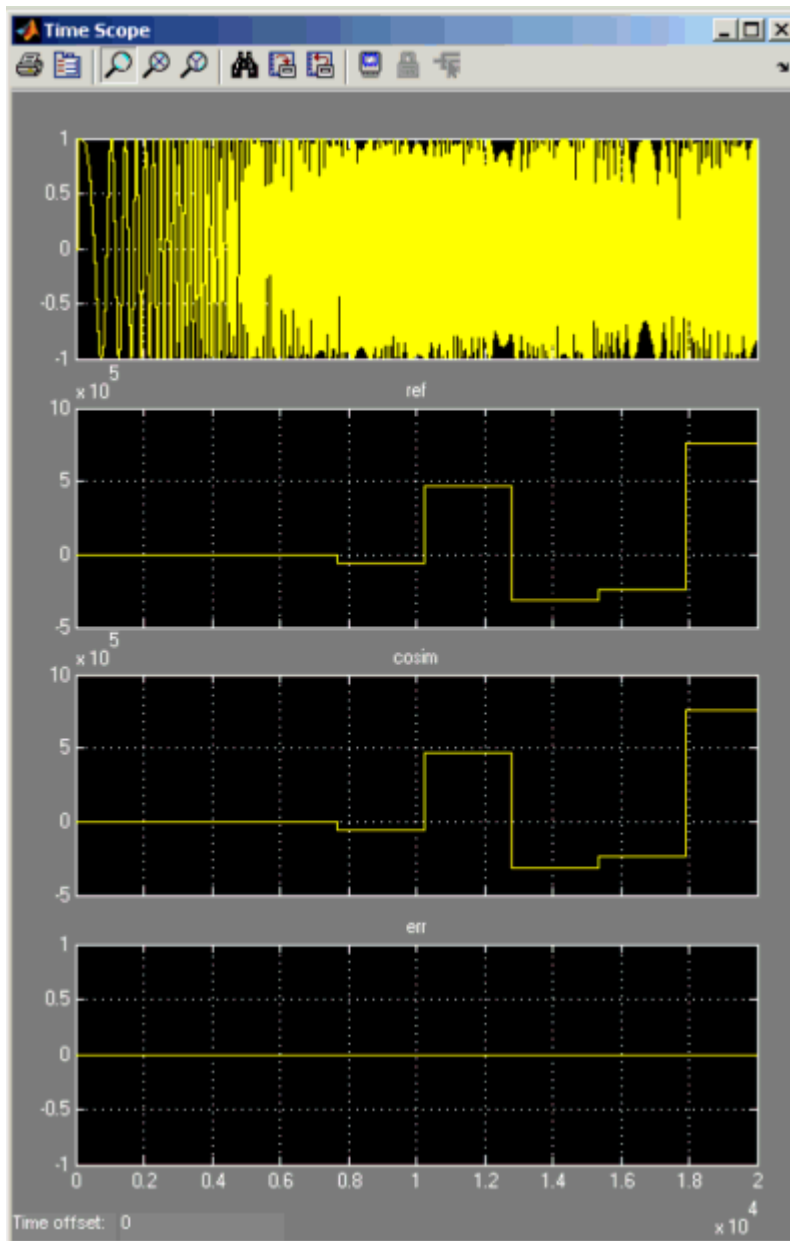
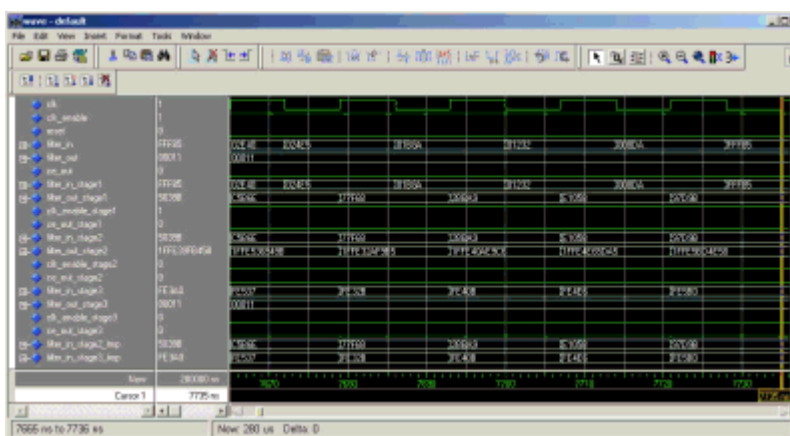


Figure 5. Simulink scope window showing zero errors between Simulink Fixed Point Model and VHDL Model of the DDC Filter



[\(Click to enlarge\)](#)

Figure 6. VHDL signals are viewable in Simulink and ModelSim

Methodology Once Again

The co-simulation approach using Model-Based Design that is being proposed has significant benefits for the ASIC and the FPGA implementations. For engineers who use FPGAs to verify their designs and then target ASIC as the final

product, the management of multiple representations of the HDL netlist becomes an issue. In other words, both the ASIC and FPGA teams start with the same netlist but the netlists diverge because the FPGA team may need to modify the HDL to map it correctly to a given FPGA. This can be due to I/O constraints or speed constraints. Once the FPGA is successfully verified, the engineers often find it difficult to ensure that all the changes that were made to the FPGA netlist are correctly reflected in the ASIC netlist. The big question, then, becomes: "how do you know that what you verified via FPGA is what you will be taping out?"

Using the suggested methodology of Model-Based Design, the engineers can insert the FPGA netlist or the subset of FPGA netlist into the Simulink Model and verify that it still passes the 'golden executable design specification'

Summary

In summary, Model-Based Design captures the benefits of both top-down and bottom-up design approaches to provide a design flow that maps system specification into hardware.

While the DDC filter design example is a single function that was used to illustrate the concept of Model-Based Design using Link for ModelSim, the approach can be used to create entire systems and the ModelSim block can easily encompass multiple levels of HDL hierarchy.

Model-Based Design permits system engineers and HDL engineers to collaborate using functional models of the design specification, thus speeding design and verification of activities significantly. Using the Link for ModelSim tool, engineers can ensure

- That the HDL is functionally equivalent to the system specification.
- That the HDL has correct signal interfaces to adjacent blocks.

References:

- Yanik, Paul. March 11, 2004. Migration from Simulation to Verification. EDA Tech Forum, Newton, MA.
- 2005 EETimes EDA Survey – Chip Survey http://i.cmpnet.com/eetimes/eda/edasurvey_chip.pdf

About the author:

Sudhir Sharma is a Signal Processing and Communications Product Marketing Manager at the MathWorks, Inc. He has over 20 years of experience bringing semiconductor IC products to domestic and international markets.

Prior to joining the MathWorks, Sudhir was cofounder of Engim, Inc. where he hired and led the engineering team to design wideband 802.11 products. Previously, Sudhir served in key technical and management roles at Texas Instruments, Cirrus Logic, and Compaq Computer Corp.

Sudhir has authored 18 US Patents in the area of computer networks, image processing, and memory design.

Sudhir received a Masters' degree in Electrical Engineering from Southern Methodist University in Dallas, Texas in 1991. He can be reached at sudhir.sharma@mathworks.com