Connecting the System to the Chip: Using VHDL/VITAL for Board-level Simulation

Russell E. Vreeland
Free Model Foundation
31901 Valone Ct.
Temecula, CA 92591
russ@vhdl.org

Abstract:

“Systems on Silicon” has been an oft-repeated phrase of the last couple years. The ability to pack more and more gates onto a single IC has made this buzz phrase a reality and brought many issues to the forefront: Intellectual Property concerns, sub-micron design & verification methodologies, speed & size hurdles, entirely new or rejuvenated strategies such as cycle-based simulation and formal methods. This is all fine, but somewhere away from the sexy, leading-edge design arenas tackling million-gate designs, there are still boards being made to put all these Systems-On-Silicon; no one has yet proposed free-floating ASICs holographically connected to the rest of the universe. These boards with their attendant glue logic and other off-the-shelf logical components may still need to be simulated either alone or with their valuable ASIC cargo.

The Free Model Foundation, a not-for-profit corporation, was formed by myself and 3 other engineers, who were working at TRW Inc. at the time. Our motivation was brought about by the lack of easily available board-level VHDL simulation models. Our goals were and are to encourage the use of standards-based board-level simulation by posting our findings, utilities, and simulation models to the public domain. “We” in this paper refers to the FMF, either working on our own, or previously working for TRW Inc.

This paper discusses a few topics in the use of VHDL/VITAL for board-level simulation. These include advantages of standards-based vs. proprietary tools; technical issues with VITAL board-level simulation models tackled by the FMF, and the work of the FMF to provide a resource and repository for public domain board-level simulation models.

VITAL as a standard for board-level simulation:

Why use a digital board simulation methodology based on VITAL? The first reason is that VITAL is a standard. Using proprietary simulators with proprietary formats causes a lack of design portability, and a risky entanglement with the lives and fortunes of CAD/CAE vendors and products. Using a standard like VITAL frees the design from such entanglements. The cumulative efforts of the entire CAD/CAE industry to improve tools tied to the standard are leveraged. Other advantages are gained from using an HDL-based methodology. A commonality of design methodology between chip and board is furthered. The problem of multiple tools’ support and training issues is alleviated.

With the advent and maturing of the VITAL standard, we chose it as the simulation model standard for our board-level component models. Verilog was not an attractive option in the Defense Contractor environment, since our ASIC and FPGA work was done in VHDL. Trying it anyhow (in the late 1994 time frame), we experienced considerable problems using Verilog in a heterogeneous, multi-library environment. This paper does not intend to ignite any religious wars by discussing Verilog vs. VHDL any further – the FMF supports the use of standards in general, including Verilog (we have some activity in supporting the IBIS standard as well) - but our experiences and our needs have so far been in VHDL.

The VITAL standard was written primarily to solve the problem of developing consistent ASIC libraries with VHDL. The use of VITAL for board-level models pushed the standard in some ways that required finding solutions. There are difficulties writing models for ECL parts that have differential inputs and/or clocks. Modeling passive components proved to be no problem except for the interesting case of the bi-directional resistor model. Due to the verbose style of VITAL and its deficiencies in support for "sizeability" we chose to represent most of our sizeable models as single-bit and let the CAD tool's netlist generator expand the sized model into multiple instantiations. (While size or bit width of the model may be passed into the model as a VHDL generic, it's difficult to propagate that "size" generic through the entire model: many VITAL timing check procedures aren't overloaded for vectored parameters, for one example.)[1]

VITAL as a standard for board-level simulation facilitates a complete top-down design methodology, if VHDL is the language used for other levels of simulation. In an idealized design flow, system engineers do behavioral, conceptual analyses and generate simulatable specifications, design engineers plug their increasingly complex RTL and (later) gate-level representations of the same design into the same test benches the system engineers used, and at the end, the whole thing goes "on-the-shelf" and becomes a reusable archive for the next generation design that interfaces or extends the one completed. This is idealized, but realizable if the discipline exists within an organization to pioneer and establish the flow.

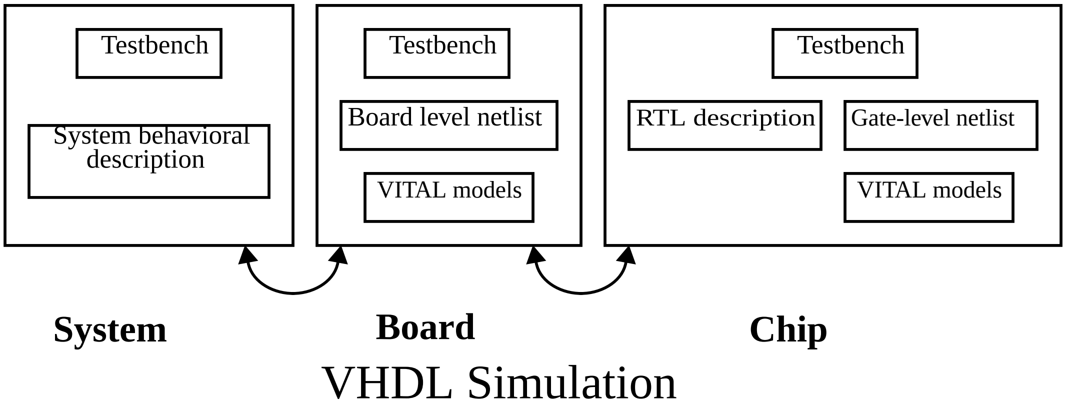


Figure 1. Completing the connection from system to chip using VITAL for boards

Technology Independent Models

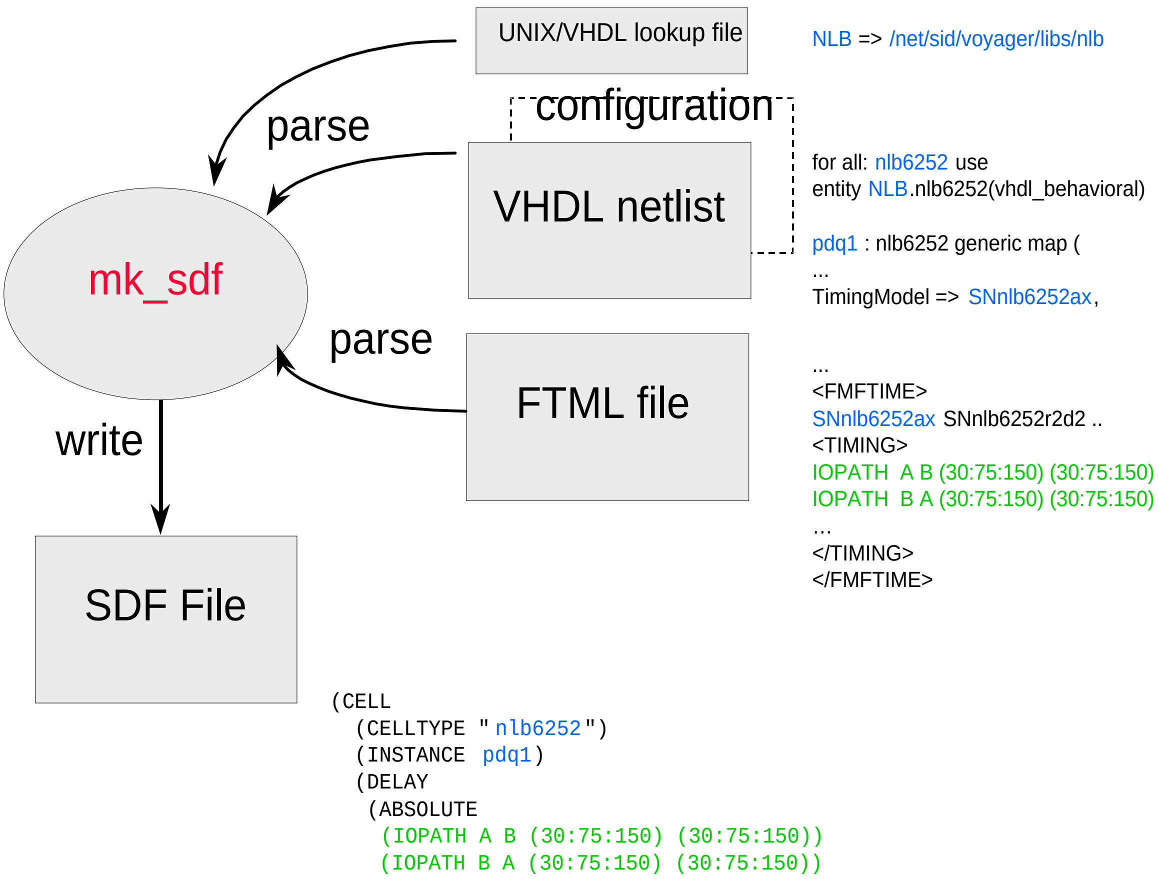
In board design, there often are many vendors and technologies available for a component. The FMF developed a technology independent method of keeping timing information separate from the model and using the SDF backannotation capability of VITAL to import the timing of a particular component into a generic model. This considerably reduces the number of models required to be developed and maintained. Total component counts in the tens of thousands of

[1] See Vreeland, R.E. "Board Level Component Modeling Using VITAL" – Proceedings of the Spring '97 VIUF for a detailed discussion of the topics in this paragraph, including ECL modeling with VITAL. The paper is available at http://vhdl.org/vi/fmf/wwwpages/vhdl_papers.html

components are not uncommon for the libraries accessed by a typical board design engineering organization.

The FMF technology independent libraries use SGML-based timing files that are parsed by a "C" software program developed by the FMF, "mk_sdf". The program writes out an SDF file for the design according to the timing selections made by the designer. The typical way that is done is for the designer to attach a property to the component on a schematic that specifies the exact Timing Model to be used. This can be done automatically by selecting approved parts from a catalog that has the properties already attached to the parts.

The format of the timing file, "FTML", is based on SGML (Standard Generalized Markup Language).[1] For each generic simulation model, there is one FTML file but that file may contain scores of entries for different electronic parts that utilize that same simulation model. Inside the <TIMING> section of an FTML file, the timing data for the model is listed in the form of a chunk of an SDF file's "DELAY" entry. The only difference between FTML syntax and the snippet of SDF text within the <TIMING> tags, is parentheses may be omitted, and the SDF keywords TIMING, DELAY, ABSOLUTE, and TIMINGCHECK may be omitted (mk_sdf automatically adds these to the SDF file).



```
NLB => /net/sid/voyager/libs/nlb
```

```
for all: nlb6252 use
entity NLB.nlb6252(vhdl_behavioral)

pdq1 : nlb6252 generic map (
...
TimingModel => SNnlb6252ax,
```

```
...
<FMFTIME>
SNnlb6252ax SNnlb6252r2d2 ..
<TIMING>
IOPATH  A B (30:75:150) (30:75:150)
IOPATH  B A (30:75:150) (30:75:150)
...
</TIMING>
</FMFTIME>
```

```
(CELL
  (CELLTYPE "nlb6252")
  (INSTANCE pdq1)
  (DELAY
   (ABSOLUTE
    (IOPATH A B (30:75:150) (30:75:150))
    (IOPATH B A (30:75:150) (30:75:150))
```

Figure 2. mk_sdf algorithm

[1] The exact format specification of a derivative of SGML is given by a DTD (Documented Type Description) The FTML DTD is available at: `ftp://vhdl.org/vi/fmf/fmf_public_models/tools/ftml.dtd`

Packages: IEEE or Brand X IEEE?

For synthesizing ASICs and FPGAs, it’s safe to say that Synopsys Inc. has the lion’s share of the market. When developing RTL level (synthesizeable) VHDL descriptions of designs, the Synopsys extensions of the IEEE std_logic_1164 package std_logic_arith, std_logic_signed, and std_logic_unsigned are generally used. Embedded in these Synopsys packages (as well as Synopsys’ own std_logic_1164 package) are all sorts of synthesis tool information in the form of directives or “pragmas” (which are comments in the VHDL source code). The Synopsys packages have names for many conversion functions that are specific to Synopsys. For example, std_logic_arith from other vendors has a function to convert UNSIGNED types to integer types “TO_INTEGER()”. In the Synopsys package the function is “CONV_INTEGER()”. The new IEEE Synthesis package “IEEE.NUMERIC_STD” uses the “TO_INTEGER” notation.

This jumble of packages causes problems when a design flow combines an RTL level synthesis design with a board design where VITAL is used. Unless the VITAL packages and models using them are compiled with the Synopsys versions of IEEE 1164 packages, the board design will not simulate with the RTL or synthesis design. But it doesn’t make a lot of sense for large, board-level libraries to be compiled with a package tailored for synthesis – especially if the design environment using the board level libraries is large enough to include non-users of Synopsys. Those would include users of other synthesis tools, users who explicitly call (perhaps in test benches) the bona fide IEEE std_logic_1164 conversion functions that are given in NUMBERIC_STD that do not match those from Synopsys, or users who integrate other models which are compiled with the bona fide IEEE version of the packages.

In short, the existence of non-standard IEEE packages is a big headache that must be managed in a multi-library, mixed synthesis and non-synthesis environment. The preference of the FMF is to support the use of the bona fide IEEE packages, std_logic_1164 and numeric_std (which are the standard), and recognize that, when simulating a Synopsys-synthesized ASIC or FPGA with a board, either the board must be compiled using the Synopsys packages or the chip and test benches must be recompiled using the bona fide IEEE packages.

A new paradigm: models with or instead of data books

Engineers are used to getting lots of necessary information from off-the-shelf IC vendors in the form of data books, or data sheets. Usually, the timing specifications of an SSI or MSI part combined with timing diagrams is sufficient to write a simulation model for the part in VITAL. This raises the question as to why the IC vendors don’t simply supply the model as well as the data sheet.

Well, they ought to. The model creation work would represent a small investment for a large IC vendor, but is a big job for the CAD staff of the typical systems design house. CAD vendors have traditionally supplied many digital libraries but have never been very keen on the task – the quality of most CAD vendors’ libraries speaks for itself. The reasons usually given for IC vendors not supplying their own model libraries are summarized (and debunked) as follows:

It gives away IP to competitors.

Not so. Information in the behavioral model just reflects in useful form what is already in the data sheet.

IC vendors would be liable for inaccuracies in the models, or if not liable, harassed.

They’re not liable for inaccuracies in the data sheets, and ought to appreciate feedback regarding any inaccuracies – misrepresentations of the product could sour customers on the vendor. The same reasons ought to be valid for models provided as “simulatable data sheets”.

No one is asking for them.

That can be changed. The vendors should be pressured to provide the model. An expectation of receiving support in the form of simulatable data sheets needs to be developed among designers.

Of course, the large, expensive models to create, such as Pentium processors and DSP chips, would still be significant IP for IP houses to market and sell: where the line should be drawn between these and the smaller-scale models still needed for simulation is up to the market to determine.

Perhaps when one IC vendor does supply its own models, markets that as a sales differentiator, and gets positive results from doing so, then the paradigm will begin to shift.

Status of the FMF Web site

Primarily because of the previous “day jobs” of the members of the FMF at TRW Inc., most of the models created to date have been of the high-speed design variety. Much of the CAD support that we were involved in at TRW was for ECL and GaAs high-speed boards with lots of relatively small-scale, small gate count components. We developed many ECL simulation models for the Motorola ECLinPS family, as well as some very high-speed models from the NLB family from NEL, a Japanese company. These are posted on the web site.

In some respects, these high-speed boards are reminiscent of the boards done 10 to 15 years ago during the “Golden Age” of TTL design. Back then, boards were loaded with lots of 7400 series ICs. Today, it’s more ASICs and special purpose glue-logic (bus interface ICs, transceivers), but to do board simulation, they all must be modeled. Since it’s impossible for the FMF alone to develop a comprehensive set of simulation models that would cover all categories of designs, we open our site as a repository of models developed and donated by others. Donated models can be of any shape and form – we’ll categorize them as best we can – the only criteria are all models become public domain subject to the GNU “copyleft” agreement which is used in all our models, and the models must be posted in unencrypted source code.

Intel Corporation has donated many memory models. These models are not written in conformance to any particular standard (VITAL support of memory modeling is still in the works), but we are glad to post them and disseminate the information to whomever might find them useful, either designers using Intel parts, or those wishing to adapt them to do their own memory modeling.

In addition to the high-speed models, the FMF has also posted many discrete models, including a fairly sophisticated attempt to model the bi-directional resistor model in VHDL. There are several utility packages: one having constants and tables useful for ECL modeling, another having VITAL state tables for modeling most types of flip-flops and latches. Utilizing the flip-flop package would be particularly useful to modelers – the work done in reducing the pessimism of the flip-flops (making them a lot more like real flip-flops in a design) is often overlooked by a casual model writer. The mk_sdf program described in this paper is posted in source code. Of course, it was written and tested utilizing the GNU gcc “C” compiler.

We have been at work on a VITAL Modeling Style Guide for some time, and the beginnings of one have been posted. The key indicator of our “style”, however, is the models themselves. The format used for the models has been developed over more than 2 years of working with VITAL and we think it is sufficiently advanced to be emulated and copied.

Summary

Irrespective of the evolution of board designs from large carriers of 7400 series MSI components to smaller holders of million-gate ASICs (where most of the design effort occurs), board simulation requires models of all those board components which are not ASICs. The emergence of the VITAL standard for ASIC gate-level modeling can be taken advantage of for modeling board components as well. Doing so produces a seamless simulation environment which connects the systems (boards as well as system behavioral models and test benches) to the chip (ASIC or FPGA).

The Free Model Foundation has tried to spark interest in pursuing this approach by researching the methodology of using VITAL for board level simulation, and posting a growing number of source code models to the public domain. The FMF web site serves as a repository for simulation models contributed by designers who wish to contribute to alleviating the problem of a lack of easily available, standards-based board-level simulation models.

A paradigm shift to IC vendors providing the simulation models for their off-the-shelf components would be good thing for encouraging board-level and system level simulation. Board simulation models, for the most part, are proprietary to the CAD vendors' tools, and don't fit well into the ideal of a top-down design and verification environment. If a way could be found to encourage better availability of models, some of the stumbling blocks to this environment would be removed.

Acknowledgments

I would like to acknowledge the work of my colleagues in the Free Model Foundation: Richard Munden (currently with Acuson Inc, formerly with TRW), Luis Garcia (currently with IKOS Systems, formerly with TRW), and Bob Harrison (currently with TRW). In addition, Gordon DeSmet (TRW) and Raymond Steele (TRW) contributed in a big way. TRW Inc. has been incredibly progressive and generous in donating its board-level VHDL models to the FMF and thereby to the public domain, as has Intel Corp. Diagonal Systems and Veda Design Automation have donated tools and encouragement to the FMF.

References

IEEE Standard VHDL Reference Manual

IEEE VITAL Standard ASIC Modeling Specification

Vreeland, R.E., “Tricks and Techniques for Writing VITAL 3.0 Compliant ECL Models”, Cadence International User's Group, Mentor User's Group, both Fall 1995

Vreeland, R.E., “Board Level Component Modeling Using VITAL”, Proceedings of the Spring 1997 VIUF

High Performance ECL Data, ECLinPS and ECLinPS Lite, Motorola Inc.

Biography

Russ Vreeland is a Sr. Consulting Engineer with IKOS Systems Consulting Services in Carlsbad, CA, specializing in the verification of complex digital ASIC designs. He has a BSEE from the Florida Institute of Technology and has been working with simulation tools and CAD systems for 10 years. His other hat has been as Director of Engineering of the Free Model Foundation, a not-for-profit corporation dedicated to promulgating the use of standards-based freely available simulation models.