Free Models:

What are they?

How are they used?

How can they be free?

Richard Munden
Free Model Foundry
324 N Murphy Av
Sunnyvale, CA 94085
(408) 774-1884
www.FreeModelFoundry.com

Abstract

The Free Model Foundry (FMF) provides VHDL and Verilog open source component models to the electronics industry. These models are free to end users. This paper will explain what makes these models ideal for board-level verification and how to use them. It will also explain the Free Model Foundry business model and how FMF can provide these models to end users without cost or restrictions.

1.0 Introduction

A critical step in the design of any electronic product is verification. Verification is usually accomplished through a mix of simulation and prototyping. The goal is to find and fix as many of the errors as possible prior to prototyping. Design errors caught before hardware is built are easier and less expensive to fix than errors found after hardware is built.

Most ASICs and FPGAs are simulated. But all too often, they are simulated stand-alone. ASICs and FPGAs are rarely used in a stand-alone environment. Therefore, an important area of verification, correct interfaces with other components, is being missed until prototyping. In the past, these custom parts were not simulated at the board level. Models were not available for the rest of the components to which they connected. That is changing now.

Back in 1995, no one was offering models of any type, except the EDA companies. Their models were proprietary and worked only on their simulators. The component manufacturers found modeling an unattractive proposition because of all the incompatible simulators. That's when Russ Vreeland, Luis Garcia and Richard Munden, with the help of Robert Harrison, founded the Free Model Foundry. The goal in this undertaking was to develop standards for component modeling and to encourage the component vendors that models were as important as data books and should be as easy to get.

The more forward looking IC companies have begun providing simulation models of their products. This is particularly true for companies producing memory components. Micron Technology was one of the first. They started offering VHDL and Verilog models for free download on their website around 1997. IDT followed in 1999 and Spansion now also offers models. The models from these vendors are distributed as source code and should work with any simulator that meets the IEEE language standards for VHDL or Verilog.

A few other companies offer encrypted models. These models will usually run only on the particular simulator for which they were encrypted. When problems are found, the engineer cannot look inside the model to help understand what went wrong.

2.0 FMF Model Features

FMF was founded by three engineers that practiced board-level simulation. They developed a set of requirements for component models. Those requirements are: portability, readability, manageability, and performance.

2.1 Portability

Component models must work in the full range of customer environments, or the component manufacturers will not consider them worth creating. FMF models are based on VHDL and VITAL. VITAL is the VHDL Initiative Toward ASIC Libraries. VITAL provides standard methods and packages for dealing with timing delays and constraints in VHDL models. VHDL and VITAL are two IEEE standards that are supported by every major EDA tool vendor. Ancillary tools such as "mk_sdf" (discussed later) are written in Perl and are open source software.

Verilog is less portable but FMF tries to support the most popular simulators.

2.2 Readability

If simulations always passed without errors, engineers would never need to read component models. Engineers would also quit simulating their designs because they would be perfect on the first pass. But in the real world errors are found. When errors are found, engineers will frequently want to read the model reporting the error to help determine what went wrong. This can only

be done if the source code is available for the model. One of the first decisions at FMF was that models must be distributed as readable source code. Of course, this also helps with the first goal of portability. Source code can be compiled by any standard simulator.

2.3 Manageability

There are *a lot* of components to model and simulate. Fortunately, they seem to run in families. Memories come in different speed grades. Logic components come in different technologies. Using VITAL, one can create a single model that will suffice for multiple components that differ only in timing. Each VHDL model has a companion timing file that defines multiple timings for the model. Currently, 700 VHDL models are enough to provide simulation capability for over 10,000 part numbers. Having fewer models makes both modeling and model maintenance much more practical and affordable. This is possible because VITAL allows all timing values to be specified outside the model. The model is written with generics for each delay or timing constraint. The generics receive their values through Standard Delay Format (SDF) timing annotation at the start of simulation. Each instance of a component model is allowed to have unique timing values.

FMF models are intended to be read and are written in a common style. This was originally done for maintainability. Since then, FMF has realized that it also makes the models more readable and reduces the effort required to create them. The strictly adhered to style has at times allowed global edits of models when standards changed. Once a user becomes familiar with the style, it becomes very easy to navigate through the models as they all have similar organizations.

2.4 Performance

There are two measures of performance for component models: simulation speed and memory footprint. Simulation speed is maximized by writing the models at as high a level of abstraction as possible while still maintaining accuracy. Accuracy is the first priority, but using the highest level of abstraction makes the model run faster and makes it easier to write and to understand. Another benefit of modeling at a high level of abstraction, from the perspective of the component vendor, is that FMF models need not disclose any intellectual property and are not synthesizable. FMF models can usually be written based entirely on the information published in the manufacturer's datasheets.

The other measure of performance, memory footprint, is how much computer memory is consumed by the model during simulation. Footprint is a problem primarily for models of large memory devices. Many of the models that are available from the manufacturers treat memories as arrays of std_logic_vector. This causes each bit of modeled memory to consume one byte of simulation memory. With memory devices topping out above one gigabit today, this causes very large memory footprints. The FMF approach is to treat a memory as an array of integers. This way, each word (usually 9 bits) of modeled memory consumes only four bytes of simulation memory.

3.0 How FMF Models Are Used

The purpose of FMF models is not to verify components, it is to verify designs that use components. They are written to be easy to use and to integrate into a schematic capture system. All models can be precompiled in libraries and installed to support multiple engineers.

It is assumed (though not required) that board designs are captured using schematics. A design simulation flow is shown in figure 1.

The schematics are drawn in the desired schematic capture system. In doing so, certain attributes placed in the schematic are mapped to their corresponding generics in the VHDL netlist. In particular, the schematic attribute "TimingModel", which is placed on a schematic symbol, is expected to be mapped to the same named generic in the model instance corresponding to that symbol. The TimingModel generic controls the timing that will be applied to the model, and in some cases, modify its behavior. "TimingModel" is used by a tool named "mk_sdf" to find the desired timing for the instance in the model's timing file and write it into an SDF file that will later be read by the simulator. The mk_sdf program and instructions can be found on the FMF website. Another generic, "mem_file" can be used to name a file that contains data to be pre-loaded into a memory instance.

When the schematic is complete, it is netlisted to VHDL. Then, the mk_sdf script may be run to create an SDF file.

Many modern board designs are centered around one or more ASICs or FPGAs. These customized parts are almost always simulated. However, they are often simulated in isolation. Having models of the board-level components allows one to simu-

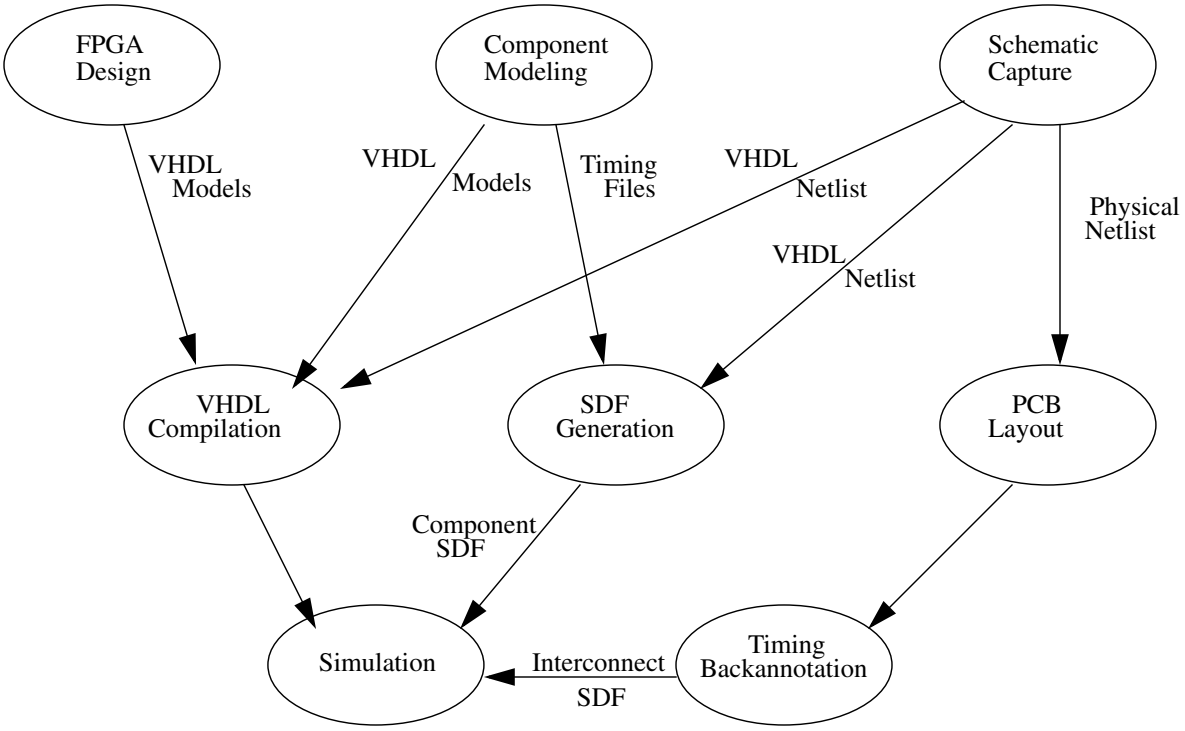

Figure 1.

late the FPGA's interfaces as well as the FPGA itself. Since the other components on the board are becoming increasingly complex, the FPGA interfaces require more attention than they did in the past. FMF models are created specifically for such use. The FPGA becomes another component in the schematic and can be simulated at the board level using either RTL code or the gate-level netlist.

As components become faster, interconnect delays consume a larger share of overall timing budgets. An important feature of FMF component models is that they facilitate backannotation of printed circuit board (PCB) wire delays. Interconnect delays are calculated by the PCB layout or signal integrity tool. They are backannotated to the simulation through an SDF file.

After netlist compilation and SDF file generation, the simulator is started, the compiled testbench and netlist are loaded and the various SDF files are read. A full timing simulation may now be executed.

4.0 How Models Can Be Free

When the Free Model Foundry was founded in 1995, the goal was to solve the model availability problem. Making money was of secondary importance. In addition, the requirement that all models be distributed as open source meant that it would be impossible to restrict model usage to paying end users.

Most software and EDA companies operate under the premise that they will write an application once and sell it over and over. This is not an unreasonable business model. If an application is expensive to develop, the cost can be spread over a large number of customers. Each customer licenses the application for less than the cost of development. However, the developing company must put in controls to enforce payment by companies that want to run the software. Distributing source code eliminates control.

FMF takes a quite different approach. The developers have to eat, just like everyone else. There is no free lunch. To paraphrase Richard Stallman, "The 'free' in 'free models' is like the 'free' in 'free speech', not the 'free' in 'free beer'". In the FMF business model, FMF develops simulation models for component suppliers and sometimes, end users. They charge for their servic-

es. That allows the developers to be paid. The models are relatively small projects. Compared to the other costs of designing and documenting a new component, the model is inexpensive. Therefore, the company marketing the new component can afford to include a simulation model as part of their marketing plan. Since FMF does not believe there will ever be a shortage of new parts to model, they can get by with charging only for development.

Anyone interested, is allowed to download and use the models without further charge. This also serves the needs of the component suppliers. They want to keep the barrier to designing-in their products as low as possible. If someone chooses to further distribute a model, all the better, as they are distributing an executable form of the datasheet and contributing to the vendor's marketing campaign.

Additional Information

The FMF website can be found at: http:/www.FreeModelFoundry/. There you can download models, packages and tools. No registration is required.

Details on how FMF models are written and used can be found in the book ASIC & FPGA Verification, A Guide to Component Modeling by Richard Munden, Morgan Kaufman Publishing, ISBN: 0-12-510581-9. It is available from Amazon.com.