

# **Building an Environment for VHDL Board-Level Simulation**

Richard Munden  
Acuson, A Siemens Company  
1230 Shorebird Way  
Mountain View, CA 94039-7393  
[munden@acuson.com](mailto:munden@acuson.com)

## 1.0 Introduction

Acuson builds premium quality medical ultrasound machines. At up to \$200,000 each, our products must be highly reliable, operate for many years and accept many upgrades. Our product life cycle is about 20 years.

Acuson has traditionally been a verilog house. Verilog was originally used for ASIC design and verification. More recently, it has been used for FPGA synthesis. It has worked fine in these applications. We intend to continue using it as a front end to synthesis and expect it to remain the sign-off language for ASIC vendors.

However, at the board-level, verification is done in VHDL. This is due to code maintainability and model availability. There are very few source level models available for commercial components in verilog. In contrast, there are free, unencrypted models available in VHDL for at least 5000 part numbers just from the Free Model Foundry (FMF). Some IC vendors, such as Micron, provide their own. More are being published every week.

For these and other reasons, it was decided to migrate Acuson to VHDL for board-level simulation. However, all those ASICs and FPGAs are still going to be done in verilog. So, a mixed HDL simulation environment is required. In this environment most component models will be in VHDL while ASIC and FPGA models, and possibly a few component models, will be in verilog. The test bench may be in whichever language the engineer prefers.

The goal of the transition is to make the overall design-simulation cycle easier and faster. We tested different simulators and determined that a single kernel simulator was required. The only single kernel simulator available at the time, 1997, was ModelSim from Model Technologies.

An advantage of HDL simulation is that it is standards based. We always have the option of switching to Cadence (or any other vendor of a single kernel simulator), if we need to. I anticipate that simulators should be fairly interchangeable in our simulation flow.

VHDL was integrated into our Concept SCALD environment. I presented a paper to this group in 1998 titled "Building an Environment for Mixed VHDL and Verilog Board Level Simulation" that described how to set up Concept to use Free Model Foundry models and VHDL netlisting. Since then, Cadence has replaced SCALD with ConceptHDL. Of course the library architecture and the netlisters have changed. So, this paper will focus on how to set up ConceptHDL to work with FMF libraries and VHDL netlisting.

## 2.0 Library Structure

The heart of every CAE process is its libraries. This is particularly true in the board design-simulation process. While Acuson's libraries have been optimized for VHDL simulation, we can still use the Verilog RTL models of our FPGAs in the board level simulations.

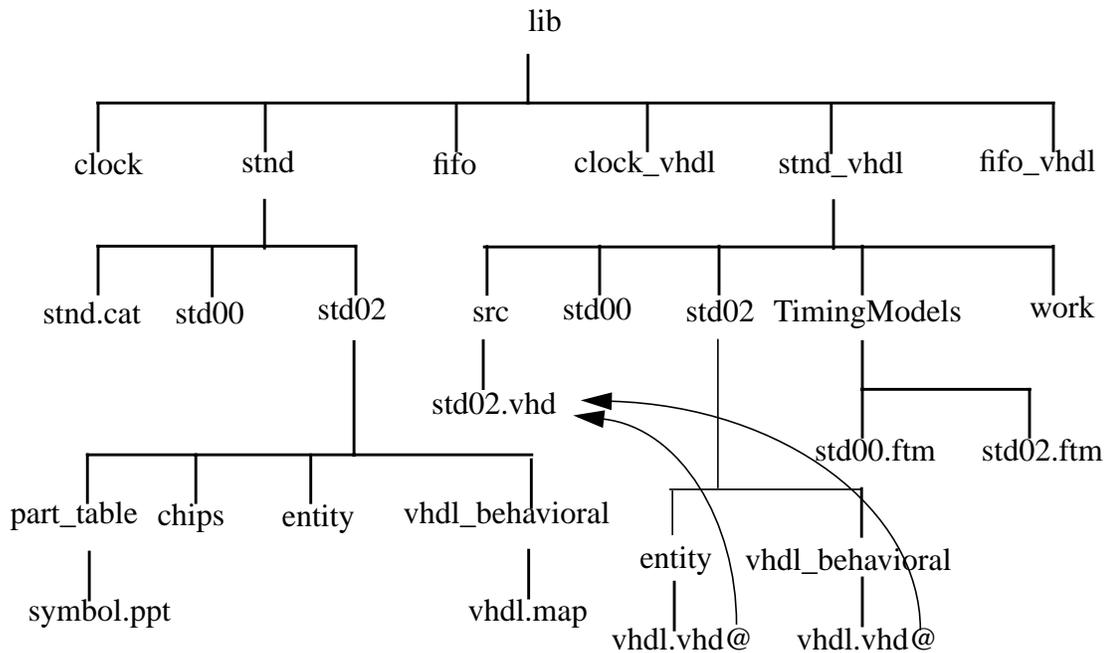
### 2.1 Technology Independence

A key feature of the Acuson library is the separation of functionality from timing. This allows for technology independence and significantly reduces the total number of parts (and models) in the library. The savings varies with part family - for the 7400 series, the savings can be huge; for more specialized components, it may be minimal.

---

## 2.2 Directories

The directory structure starts out looking very similar to the standard ConceptHDL library. For each compo-



**FIGURE 1. Library directory structure**

nent library, there must also be a corresponding VHDL library. That is the ConceptHDL standard. We have added three new directories to the VHDL library. These are *src*, *TimingModel* and *work*.

The *src* directory contains the model VHDL source code. The *TimingModels* directory contains the timing files. And the *work* directory contains the compiled VHDL models.

It should be pointed out here that the directories found in the component library are based on the part body names but the directories found in the VHDL library are based on the model names. There is not a one to one mapping between the body names in the component library and the model names in the corresponding VHDL library. The mapping between the two libraries is controlled by each part's *vhdl.map* file.

Within the *entity* and *vhdl\_behavioral* directories (in the *lib\_vhdl*) there are links named *vhdl.vhd*. They point back to the model source in the *src* directory. These links are required for the netlister to work.

At Acuson, after the librarian creates the component directory with the body files (*symbol.css*) and *chips.prt* file, a perl script called *chp2vhdl* is run. It creates the corresponding directories in the VHDL library. These include the *entity* and *vhdl\_behavioral* directories, a dummy model, and the two links. The dummy model is then usually replaced with a functional model and the map file edited accordingly. It also creates a *vhdl.map* file in the component directory.

## 2.3 Map Files

Map files are required to map pin names from ConceptHDL *chips.prt* files to the VHDL port names listed in the entity. On a good day, they perform a simple function simply. On a bad day, it can take a lot of trial and error to figure out how they work.

```
FILE_TYPE = VHDL_MAP;
PRIMITIVE 'ACT04_DP', 'ACT04_SO';
  DEFAULT_MODEL = 'STD04';
  MODEL 'STD04';
  PIN_MAP
    'A'<0> = '(A)';
    '-Y'<0> = '(YNeg)';
  END_PIN;
  END_MODEL;
END_PRIMITIVE;
END.
```

**FIGURE 2. sample vhdl\_map file**

One of the more interesting things to figure out was what to put in the primitive line of the map file. If a section of the *chips.prt* has only one entry on its primitive line, then the part\_name from that section should be entered on the primitive line of the map file. However, if the section of the *chips.prt* has multiple entries on its primitive line, then each of those entries must be duplicated on the primitive line of the map file.

Map files are explained in greater detail in the “Concept-HDL Digital Simulation User Guide” in cdsdoc, the Cadence on-line documentation library.

## 3.0 Tool Flow

Here is the general design flow through simulation and analysis. It shows the various tools and the order in which they are used.

### 3.1 Libraries

It is assumed at this point that the libraries are in place and are pre-compiled for the user. This would normally be done by the librarian, however, the user may be responsible for setting up any verilog FPGA or ASIC models in the design as described above. Acuson’s component libraries are technology independent and utilize FMF simulation models. This reduces the effort required for library development and maintenance.

### 3.2 Schematics

Schematics are drawn in ConceptHDL as they would be for any board design with some caveats to be discussed below. The schematics may be either hierarchical or flat. Because Acuson uses FMF technology independent libraries, components must be added to the schematics using the component browser set to physical mode. Selecting parts this way causes properties to be added to the schematics that are later used to select the correct timing for each component.

#### 3.2.1 Remove Attribute

Some parts in the design are best left out of the simulation. De-coupling capacitors, for example, will add nothing to the functionality. They will just make the netlist longer. Series termination resistors also will not help your simulation but the pullup and pulldown resistors should stay.

---

ConceptHDL has an attribute that can help. The attribute name is "REMOVE". There are four possible values. The two we have used are "LINK" and "EXCLUDE". If a body has "REMOVE = EXCLUDE" attached to it, the netlister will remove that component from the netlist all together. This is good for all those de-coupling caps. If a body has "REMOVE = LINK" the two nets attached (I don't know what happens if there are more than two nets) will be aliased, in effect shorting them together. This is good for series terminating resistors.

Beware, that alias can have side effects in a hierarchical design if one of the nets is a port on the block.

### 3.2.2 Hierarchical Bodies

In a hierarchical design, each block becomes a subdesign in the netlist. As such it has its own entity and port list. The direction of the ports is determined by the file `<block_name>/entity/vhdl.vhd`. This file is created and/or modified each time the body of the block is written. If the ports don't come out with the intended direction automatically, the easiest thing to do is simply edit that file.

## 3.3 Setup

Before netlisting, a number of setup steps are required:

- ensure that for every component library in your design, you also reference the corresponding VHDL library.
- under setup -> tools -> simulation, select the simulator type that is correct for your site. At Acuson, we use "Third Party VHDL". Then click on the Setup button.
- enter the VHDL package libraries you will be using. If you use FMF models you will enter "IEEE" and "FMF".
- enter the packages you will use from the above libraries. For FMF models they are "IEEE.std\_logic\_1164.all", "IEEE.VITAL\_TIMING.ALL", "IEEE.VITAL\_PRIMITIVES.ALL", and "FMF.gen\_utils.all".
- if you have turned off "Create Netlist" in ConceptHDL, turn it on and check VHDL. Write all pages of your design.

## 3.4 Netlister

There seems to be problems running the netlister from the command line so, I recommend using the gui.

- From the tools menu in either ConceptHDL or the Project Manager, select simulate.
- From the popup window that appears, enter the path to where you want the VHDL netlist and log files to go.
- Click on "Run".

Eventually, a popup will appear to tell you that netlisting was successful, or not. If it was not successful, a markers file may have been created that you can use in ConceptHDL to find where in your schematic the error occurred. However, it may be easier to read the netassembler.log file that was placed in the same directory you specified for the netlist. The most likely errors that will be reported are port mismatches.

## 3.5 VHDL Compiler

Having produced a VHDL netlist, the next step is to compile it. For us, the compiler is called vcom and the command is "vcom <filename>". Before the netlist can be compiled the first time, a work directory must be established to receive the compilation results. This done with the command "vlib work". If there are compiler errors, they will be written to stdout.

---

### 3.6 SDF Generator

Each model has an associated timing file that describes the internal delays of a component with any required timing constraints. Some of the benefits of external timing files is discussed below in the Models section.

SDF generation produces a file in Standard Delay Format that may be used by the simulator to provide accurate timing for the simulation. Initial simulation runs may not require timing and can skip this step. Without SDF annotation, FMF models default to unit delays. However, some models such as certain memories, may give misleading results when run with unit delays.

The SDF tool is called *mk\_sdf* and may be obtained as a perl script from the Free Model Foundry (at no cost). It uses a command file named *mk\_sdf.cmd* which should reside in the working directory.

```
SET sdffile_suffix .sdf
SET use_global_timing_dir false
SET timingfile_dir TimingModels
SET timingfile_suffix .ftm
SET time_scale 1ns
SET local_path .
SET diagnostics off
SET vhdl_file vhdlldlink.vhd
SET lwb off
```

**FIGURE 3. Sample *mk\_sdf.cmd* file**

---

Some of the lines in the command file that may require some explanation are:

- SET sdffile\_suffix .sdf
  - The SDF file will have the same name as the VHDL netlist with this suffix appended.
- SET use\_global\_timing\_dir false
  - Set this to true if you will have all of your timing files in one local directory. Otherwise, *mk\_sdf* will look in the Cadence libraries for the timing file directories.
- SET timingfile\_dir TimingModels
  - The name of the directory(ies) that contain the timing files.
- SET timingfile\_suffix .ftm
  - Timing files use the same name as their associated models but with this extension instead of .vhd.
- SET diagnostics off
  - If you are having difficulties getting *mk\_sdf* to work, turning diagnostics on will result in some additional output files that may (or may not) help locate the problem.
- SET vhdl\_file design\_name.vhd
  - The name of the netlist. This will typically be the design\_name.vhd. It may also be supplied on the command line.
- SET lwb off
  - *mk\_sdf* understands the lwb file structure. Turn this on only if you are running Logic Work Bench.

Any line in the command file may be overridden from the command line. Normal execution is accomplished by simply entering *mk\_sdf* without any arguments.

More detailed instructions for *mk\_sdf* can be downloaded with the script from [http://www.eda.org/fmf/fmf\\_public\\_models/tools/](http://www.eda.org/fmf/fmf_public_models/tools/).

---

### **3.7 Simulator**

The ModelSim simulator is invoked from the command line with arguments for SDF backannotation and name of the SDF file and the design name: “vsim -sdfmax mydesign.sdf mydesign”. If timing is not required, the “-sdfmax mydesign.sdf” argument may be omitted for unit delay simulation. Or, multiple SDF files may be read to include interconnect delays from Allegro, timing files for ASICs and FPGAs, etc.

Based on the results of the simulation, there may be a loop back to schematic capture from this point to refine the design.

### **3.8 Layout**

After satisfactory simulation results are obtained, the design goes to PCB layout. If timing margins were determined to be tight, anticipated interconnect delays based on manhattan distances between pins may be computed and backannotated through SDF to check if timing constraints are likely to be met by the proposed layout.

### **3.9 Signal Analysis**

Subsequent to layout and routing of the PCB, signal integrity analysis may be performed. The same physical parts tables that were used to select the correct timing file for simulation also provide the name of the signal integrity model to be used in SigNoise. SigNoise is capable of computing values for various physical effects such as crosstalk and noise margins. It can also compute accurate interconnect delays based on the characteristics of a driver’s output buffer, receiver thresholds, propagation delays calculated for the board stackup, and transmission line analysis of the traces.

### **3.10 Timing Backannotation**

Accurate interconnect delays, including transmission line effects, can be extracted from Allegro with the writesdf utility. Instructions for running writesdf appear on the popup informing you of successful netlisting.

### **3.11 Simulator**

Once the interconnect SDF file is generated, a final full timing simulation can be run. It would be more efficient at this point to run a static timing analyzer. Unfortunately there still do not seem to be any good candidate tools on the market. All the data needed appears to be in the SDF files but the few static timing analyzers available read only proprietary model formats. Perhaps this is a market opportunity for someone.

## **4.0 Special Considerations**

While the design-simulation process described above should be relatively easy and straight forward, it is possible to make it difficult. Below are some techniques and caveats to keep in mind to avoid unnecessary complications.

### **4.1 Schematic Considerations**

Schematics are the primary means of capturing board design intent. Minor details in schematics can have substantial downstream effects.

#### **4.1.1 Net Names**

Try to use legal VHDL names for all nets. Names that are not legal VHDL will be changed by the netlister into legal, but less readable, names.

---

### 4.1.2 Unused Bits

Some people like to run busses into hierarchical blocks but not connect all bits of the bus inside the block. The netlister may have difficulties with this. Either connect to the block only the number of bits actually used or, attach a port body to each unconnected net inside the block.

### 4.1.3 Mixed Busses

Sometimes, an engineer will create a bus for which some bits are inputs to a block and other bits are outputs from the block. Doing this will successfully prevent netlisting. All bits on a bus must go in the same direction.

### 4.1.4 Port Bodies

The netlister occasionally has difficulties determining the correct mode of a port on a hierarchical body. Usually port mode can be forced to the correct value by attaching the appropriate port body to the net inside the block and/or manually editing the entity/vhdl.vhd file for the block. There are three usable port bodies. They are named INPORT, OUTPORT, and IOPORT. There is a BUFPORT that I have not tried. Never use the one named LNKPORT.

## 5.0 Models

Acuson uses FMF style models. This modeling style has several important benefits.

- Technology independent models allow a reduction of the number of elements in the component library.
- New speed grades or technologies of a component can be added without duplicating anything already in the library.
- Model timing can be customized without touching the model.
- Simulations can be run with just unit delays, just component delays, or with component delays and timing checks.
- Interconnect delays can be backannotated from Allegro.
- Models are available.

## 6.0 FMF

The Free Model Foundry is a not-for-profit corporation. Its goal is to improve the availability and usability of simulation models of off-the-shelf electronic components. It is working to accomplish this goal by helping the component vendors understand the benefits of providing such models. It provides services to the component vendors to assist them with technical issues. If staffing is a problem for the vendor, FMF can provide outsourcing arrangements for model creation.

FMF encourages end users that have to create their simulation models to share those models with others.

The Free Model Foundry offers all of its models for download from its website at [www.eda.org/fmf/](http://www.eda.org/fmf/).

---