UNIVERSITY OF RHODE ISLAND

Simultaneous VHDL and Schematic Simulation with the Mentor Graphics Tools - A Tutorial -

Version 1.0

February 23, 1997

Gus Uht

1 Introduction

This is an overview of the necessary steps to simulate digital designs composed of both compiled VHDL descriptions, and schematic-based circuits. The main Mentor tool that accomplishes this feat is QHPro. It is assumed herein that the highest level component (root component) of the design is schematic-based.

QHPro invokes both the Quicksim II simulator, for the schematics, and the QuickHDL simulator, for the VHDL modules. It supplies the necessary communications between the two simulators to simultaneously simulate all components of a hybrid design.

This document assumes that the reader is familiar both with Mentor schematic entry and simulation, and Mentor VHDL entry, compilation, and simulation. Two relevant documents are:

> Introduction to the ELE UNIX Environment and the Mentor Graphics Design Tools - (Project 1 of ELE 405, Spring 1997)
>
> Using VHDL on the Mentor Graphics Tools - A Tutorial- Version 1.01, 12/4/96.

The three general steps needed to simultaneously simulate VHDL and schematic-based designs are: (1) generate a symbol for the VHDL component, using Design Architect; (2) use the symbol in the schematic; and (3) use QHPro to simulate the complete design. I now go through these steps in detail, after briefly giving the framework for the example.

2 Setup

My running example will be the design of a one bit register, composed of a 2 to 1 multiplexer and a 74ls74a D-type positive edge triggered flip-flop. The final design is shown in Figure 1. The multiplexer is described in VHDL; see Figure 2.

I assume that the design data is held in `~user/proj1`. I'll refer to this as the "`proj1`" directory. Within this directory, the VHDL source should be in the `src` subdirectory, and the compiled VHDL code should be in the `work` subdirectory.

3 VHDL Component Preparation

In order to use the VHDL MUX description as a component in a schematic, we have to generate a symbol for it. Before we actually do this, make sure that you have compiled the VHDL description with a special flag set (as far as I know, this can only be done via a UNIX command line invocation of the VHDL compiler `qvhcom`). The command is:

```
qvhcom -qhpro_syminfo <path to entity VHDL source file>
```

In our case the path is: `proj1/src/mux2to1.vhdl_4`, where "`4`" is the source file's version number (just look at the `src` directory to get this).

Now you can actually generate the symbol. Do this by starting Design Architect (DA) in the `proj1` directory, and then selecting "Generate/Symbol..." from the File drop-down menu. As the source, choose "Entity". Once the dialog box expands, you have to enter (select) four pieces of information:

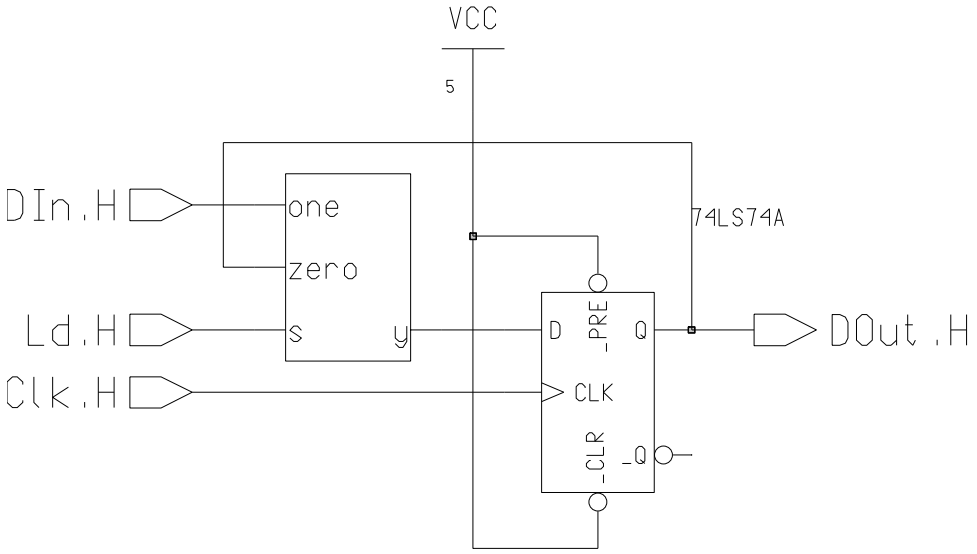

Figure 1: One bit register using a VHDL-modelled multiplexer.

```
--one bit wide 2 to 1 MUX

library ieee;
use ieee.std_logic_1164.ALL;
entity mux2to1 is
port(
        signal s,zero,one:      in      std_ulogic;
        signal y:               out     std_ulogic);
end mux2to1;

architecture first of mux2to1 is
begin
        y <= one after 30 ns when (s='1') else zero after 45 ns;
end first;
```

Figure 2: VHDL code for the multiplexer (including delay).

QHDL InitFile: This should already point to the "quickhdl.ini" file in your `proj1` directory; if not, select it with the navigator. If for some reason you can't find it, select:

`/usr/local/packages/MGraphics/lib/quickhdl.ini`

Library Logical Name: Choose "work", "./work" from the list. This is the `work` directory in `proj1`.

Entity Name: Choose the entity name of the multiplexer (say "mux2to1").

Default Architecture: Choose the architecture from the presented (one-entry) list.

Then perform the other usual selections (shape, replace existing symbol, etc.) and click "OK". If all is well, a picture of the new symbol should appear on the screen.

The next two tasks are only necessary until Mentor corrects an "issue" :-).

1. Change (manually edit) the MODEL property of the symbol **to** "`qvpro`" (from "`qhpro`"). Yes, that's to v from h. (Recall to do this, select the symbol, then press the right mouse button, and select "Properties/Modify..."; the rest is straightforward.)

2. Perform a Mentor incantation by executing the following command on a UNIX command line when in the `proj1` directory:

   `reg_model <path_to_component> -bp qvhdl -la qvpro`

   or in our example:

   `reg_model mux2to1 -bp qvhdl -la qvpro`

4 Use It

You may now call up the symbol for the VHDL MUX for use in any schematic design. Do so in your design of the one bit register ("`onebitreg`").

5 Simulate

Now, to simulate `onebitreg` (or similarly for any schematic with VHDL components), invoke QHPro (instead of Quicksim II directly), by typing on a UNIX command line:

```
qhpro onebitreg
```

Both Quicksim II and QuickHDL will be invoked by QHPro. Enter your test traces for the register as usual in the QHPro(Quicksim II) window (NOTE: to do many Quicksim operations, you may have to select "Quicksim II" from the "Solver" drop down menu; this includes setting it for non-zero delays). If you like, you can open up a Wave window in QuickHDL, select the MUX's signals, and you will be able to see the simulation of the MUX part.

Now "Run" the simulation as usual, and voila! There it is...

6 Summary and Caveats

So to use both schematic and VHDL-based designs, generate a symbol for the VHDL component, use it in the schematic, and run QHPro.

Note that there may be multiple components in VHDL form. HOWEVER: VHDL components may only invoke other VHDL components, NOT schematic-based components.

The above procedure is for designs whose top level is schematic-based. It is also possible to handle designs whose top level is VHDL-based, but the procedure is somewhat different and will not be covered here.

Good luck!