How to use Synopsys tools?

Preliminaries

At the outset, you need to ensure that your environment is set up to run Synopsys tools. One way is to use the following command in the directory where you want to run a Synopsys tools.
For C-shell, use
```
source /net/ictools/csh/synopsys.csh
```
For bash/sh, use
```
source /net/ictools/sh/synopsys.sh
```

Another (better) way to automate this is to have this line in your **.envrc** file in your UNIX home directory. If this file exists, it is automatically sourced when **.cshrc** or **.bashrc** runs when you open a new terminal.

An easy way to verify if the path for Synopsys tools is properly set is to use any of the following commands:
```
which vcs
```
should return
```
/net/ictools/synopsys/vcs-mx/bin/vcs
```
and
```
which design_vision
```
should return
```
/net/ictools/synopsys/synthesis/bin/design_vision
```
Once this is verified, you are all set to use Synopsys for functional verification (described in part 1 below) or logical synthesis (described in part 2 below).

Part 1. Functional Verification

Let us suppose you have your entire design in one file **DQPSK_rc.vhd** and your simulation testbench in **TB_DQPSK_rc.vhd**. In case you have more than one design files (which is common), e.g., **DQPSK_sub1.vhd** and **DQPSK_sub2.vhd** representing sub-modules inside **DQPSK_rc.vhd**, replace "**DQPSK_rc.vhd**" with "**DQPSK_sub1.vhd DQPSK_sub2.vhd DQPSK_rc.vhd**" in all of the commands below. Note that the ordering of the files should be from lower-level modules to the top-level module. Now, we go back to having **DQPSK_rc.vhd** as the only design file and show the steps you need to carry out for simulating your design.

1. First, you need to analyze the VHDL files using the following command:
```
vhdlan DQPSK_rc.vhd TB_DQPSK_rc.vhd
```

2. The next step is to run VCS to create the simulation executable.
```
vcs -debug_all CFG_TB_DQPSK_RC
```

3. Finally, you run the executable file **simv** to run the simulation.

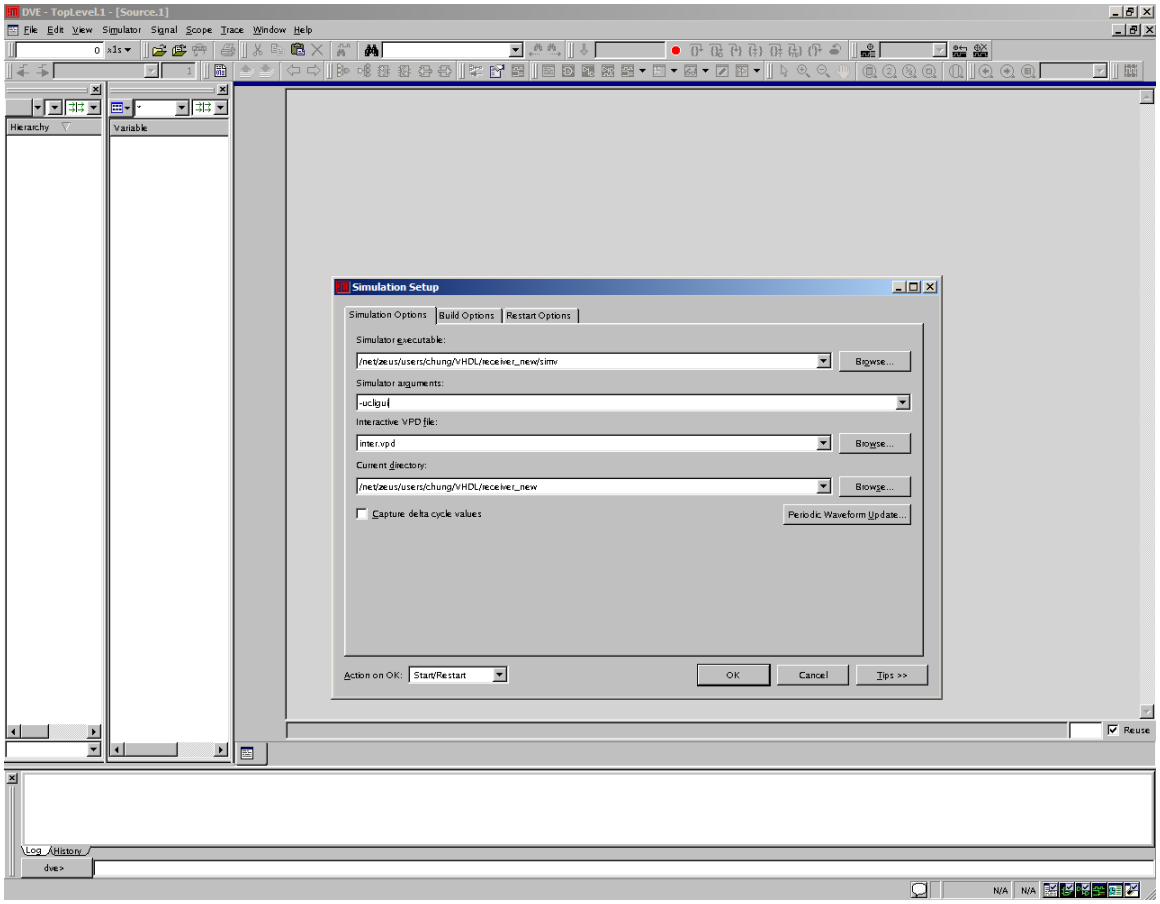
```
./simv
```

4. Now, you want to see your simulation results and debug. For this, you need to use Synopsys Discovery Visual Environment. The detailed user guide is available at: **/net/ictools/synopsys/vcs-mx/doc/UserGuide/pdf/dve_ug.pdf**
The following mentions basic steps required to view and debug your simulation results.

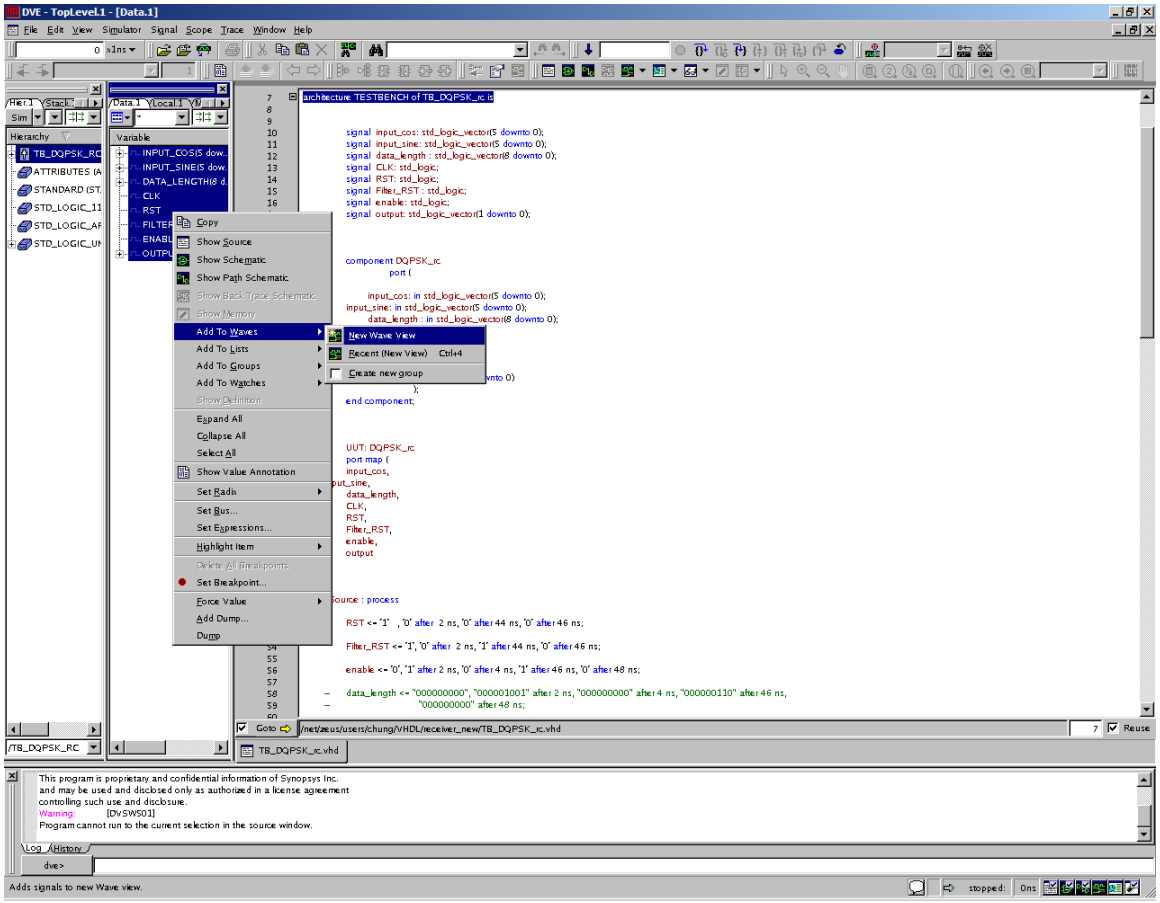
a. To invoke the tool use the command:

```
dve &
```

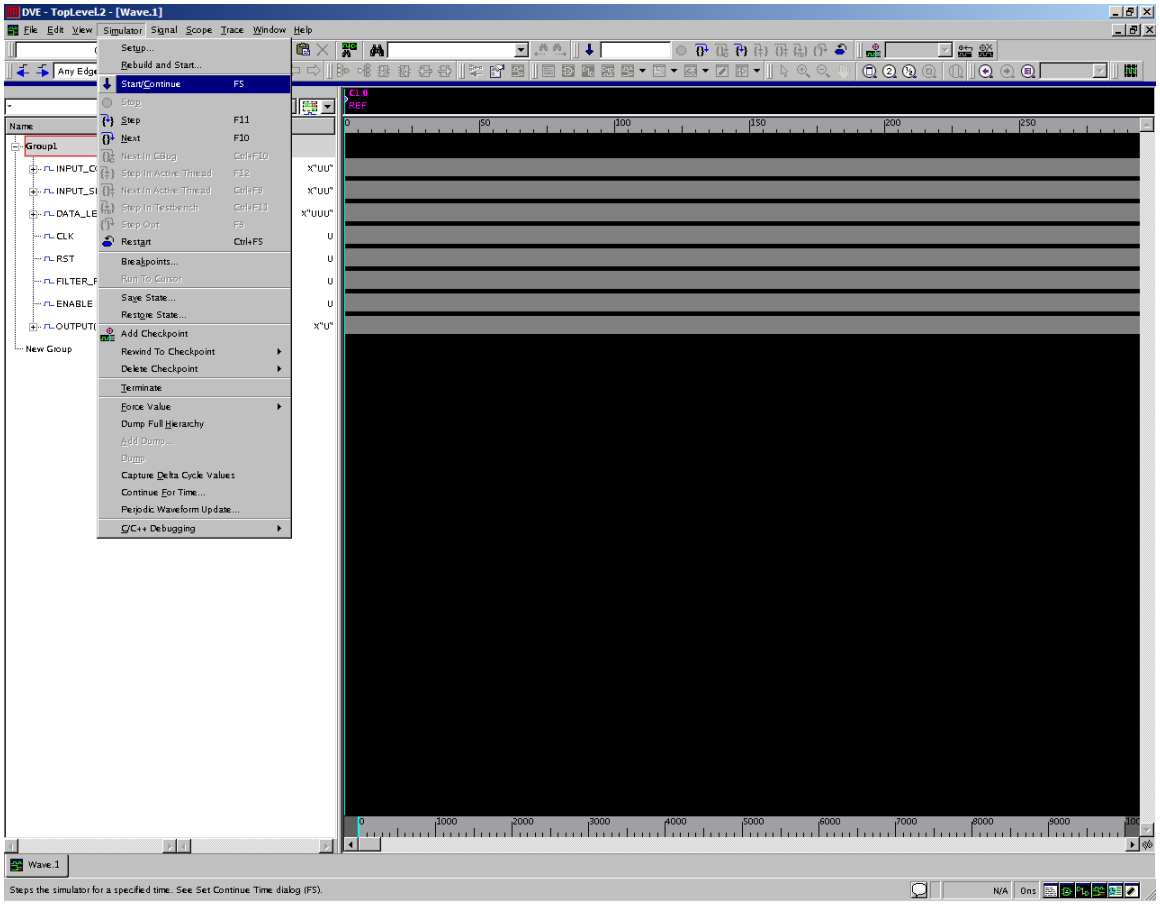
b. Next, go to Simulator → Setup and enter appropriate filenames in the fields. The screenshot below shows an example. Under "Simulator executable", enter the path for the **simv** file. Select the simulator arguments from the drop-down menu. Choose a name for your dump file, for example **inter.vpd**. Set your current working directory. Click "OK".



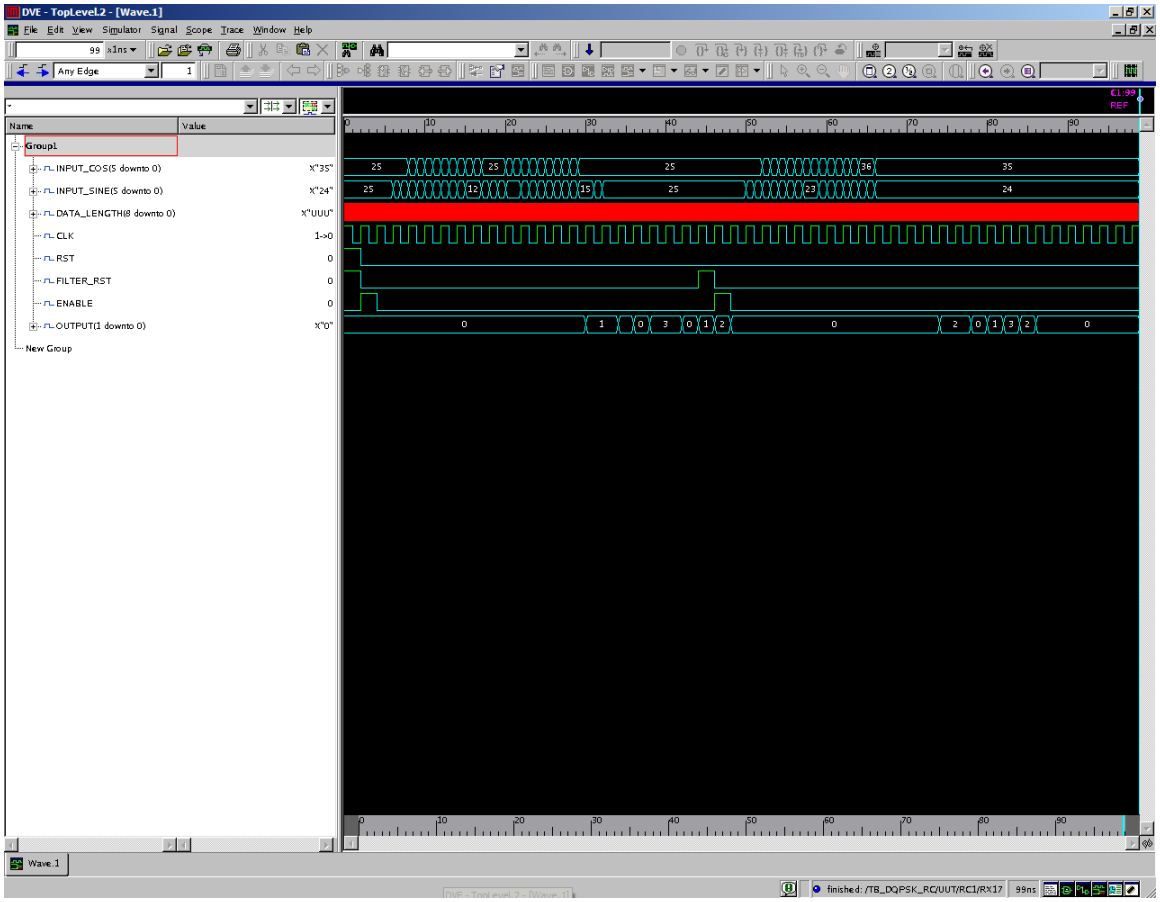
c. This opens up your design with the list of signals on the left panel. Select the signals that you want to observe in your waveform, right click and choose “Add to Waves → New Wave View” from the menu. For an example, see the screenshot below.



d. Then, to run your simulation, select “Simulator → Start/Continue”, as shown below.



e. Finally, you should see something like this.



Part 2. Logical Synthesis

The tool used for logical synthesis is Design Vision. For details, refer to the Design Vision user guide. The steps mentioned below should serve as beginner's guidelines.

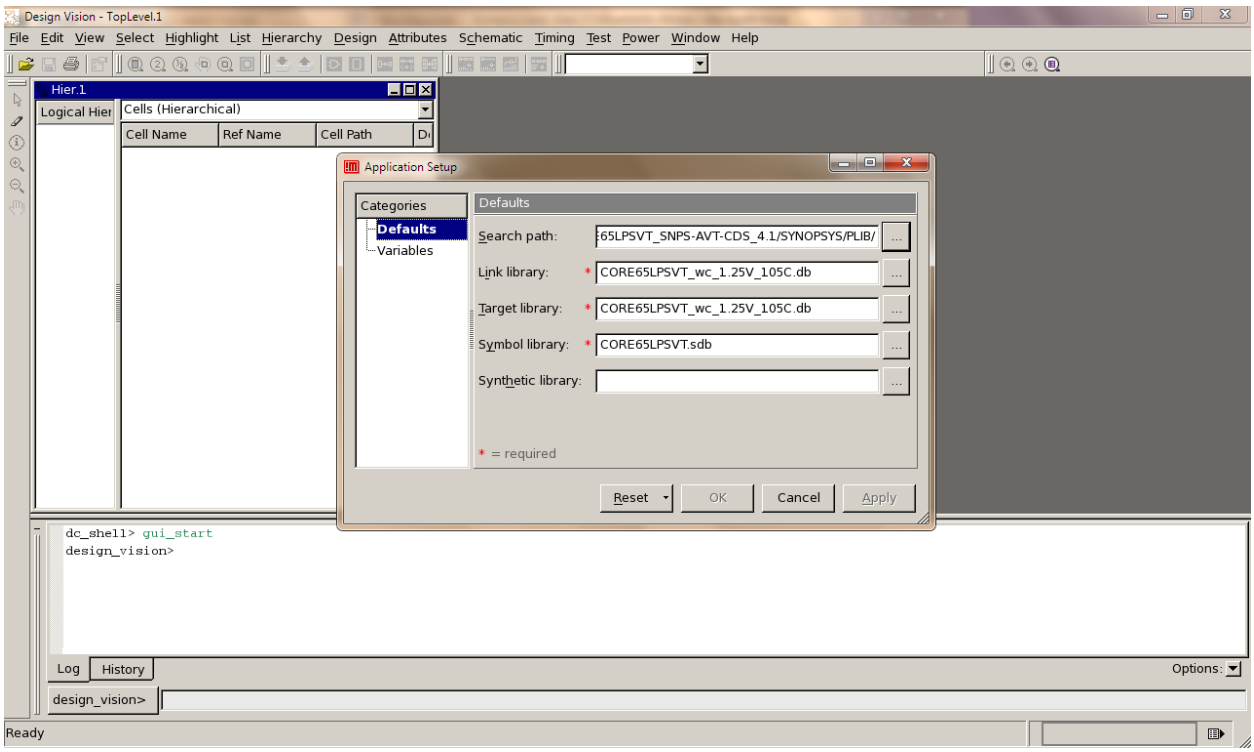
You need to have a setup file called **.synopsys_dc.setup** in your UNIX home area. The following is an example showing the bare minimum the file should contain.

```
set designer "Jacob Murray"
set company "Washington State University, School of EECS"

set search_path "$search_path \
/net/ictools/pdk/CMOS065/CORE65LPSVT_SNPS-AVT-CDS_4.1/libs/ \
/net/ictools/pdk/CMOS065/CORE65LPSVT_SNPS-AVT-CDS_4.1/SYNOPSYS/PLIB/"

set physical_library "CORE65LPSVT.plib"
set link_library "CORE65LPSVT_wc_1.25V_105C.db"
set target_library "CORE65LPSVT_wc_1.25V_105C.db"
set symbol_library "CORE65LPSVT.sdb"
```

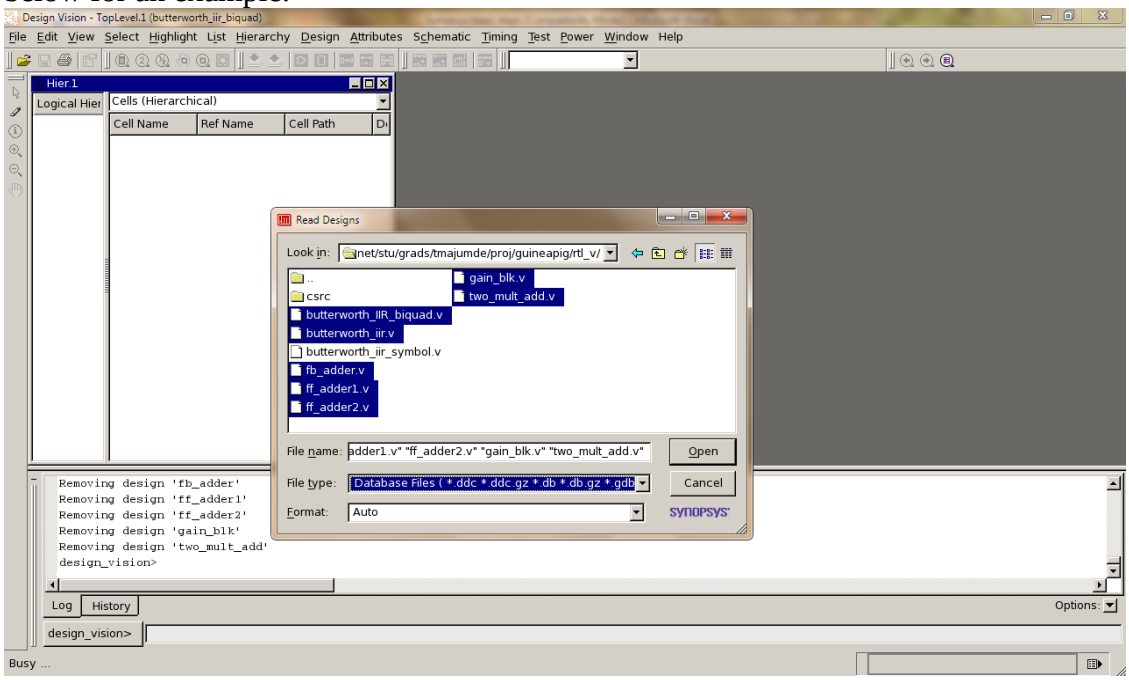
Now, you're all set to start synthesis using Design Vision. To invoke the tool, use the following command:

```
design_vision &
```

Once the GUI comes up, you can go to File → Setup and modify any of the default libraries for your particular project, as shown in the screenshot below.



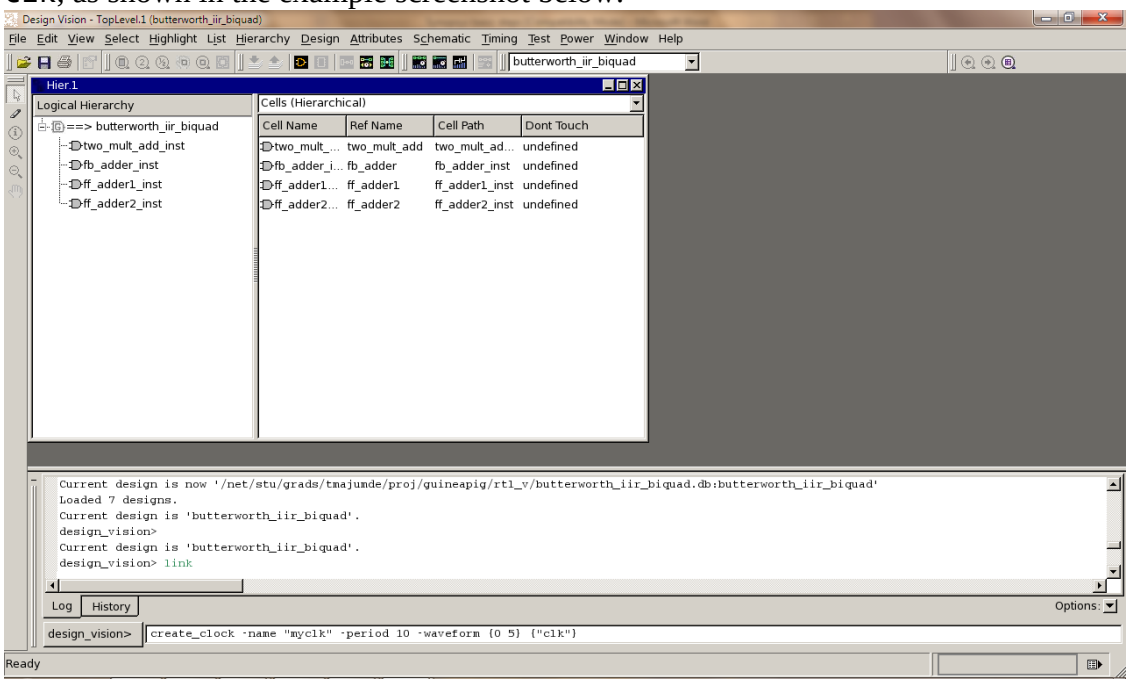
Read the design using “File → Read”. Select all the design files. See the screenshot below for an example.



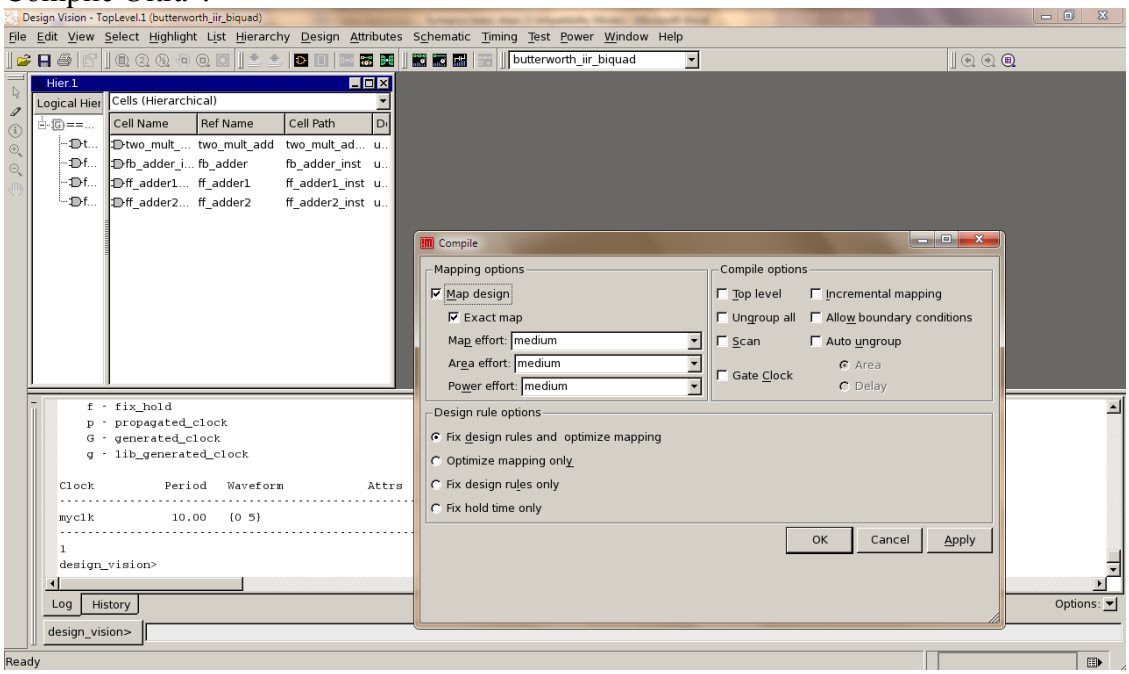
Then, you need to **specify the clock**. The best way to do that is to use the following command in the command-line.

```
create_clock -name "myclk" -period 10 -waveform {0 5} {"clk"}
```

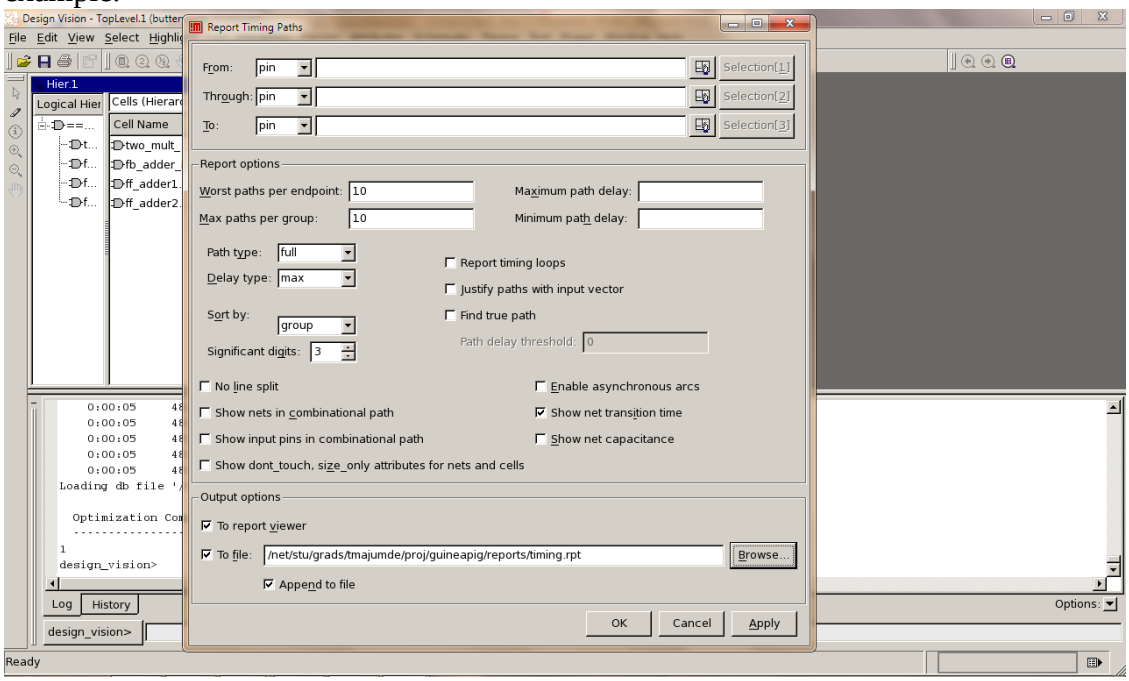
Here the name of the clock is `myclk`, it has a period of 10 ns with rising and falling edges at 0 and 5 ns respectively (i.e. 50% duty cycle) and the clock is applied at the input port `clk`, as shown in the example screenshot below.



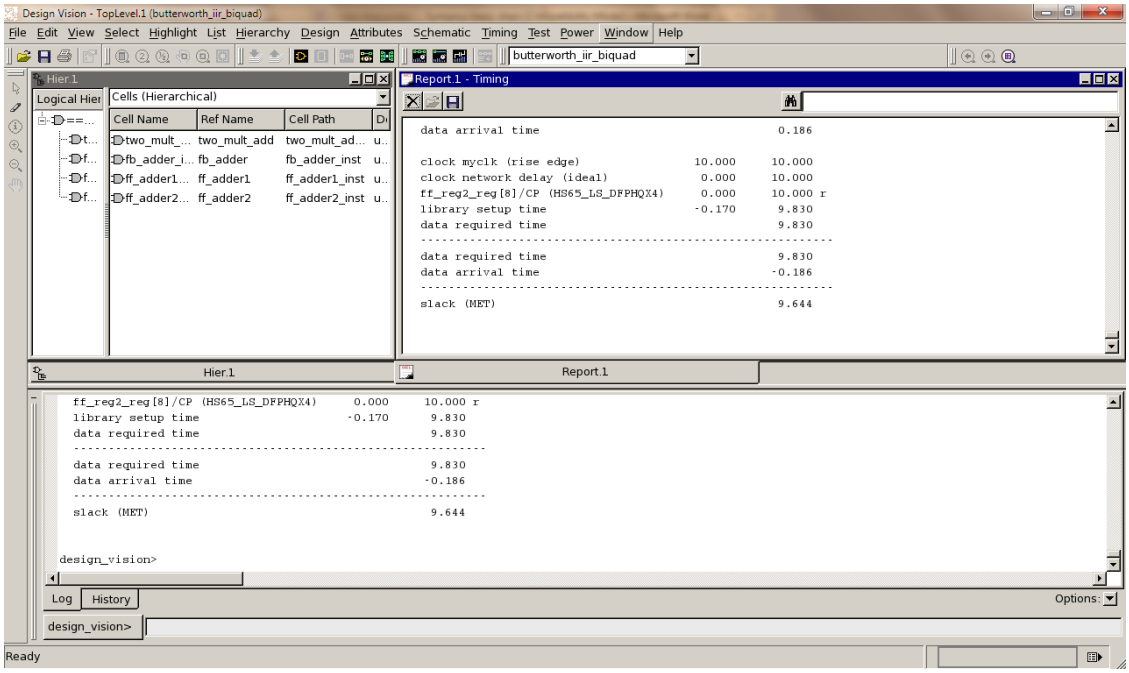
Select “Design → Compile Design” to **map the design to logic gates**. The following screenshot shows the options available. You can change the mapping, area or power efforts as per the requirements of your design. Also, you can check “Gate Clock” to enable clock gating in your design. For more optimized mapping, you use “Design → Compile Ultra”.



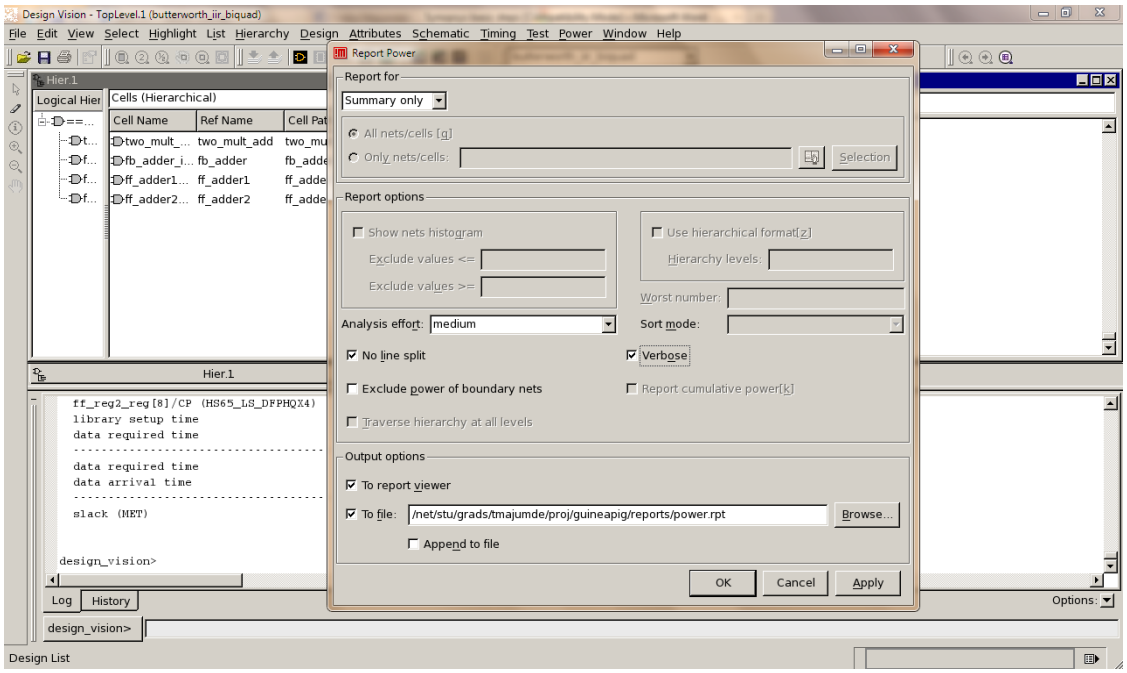
For **reporting timing**, select “Timing → Report Timing Path”. The following shows an example.



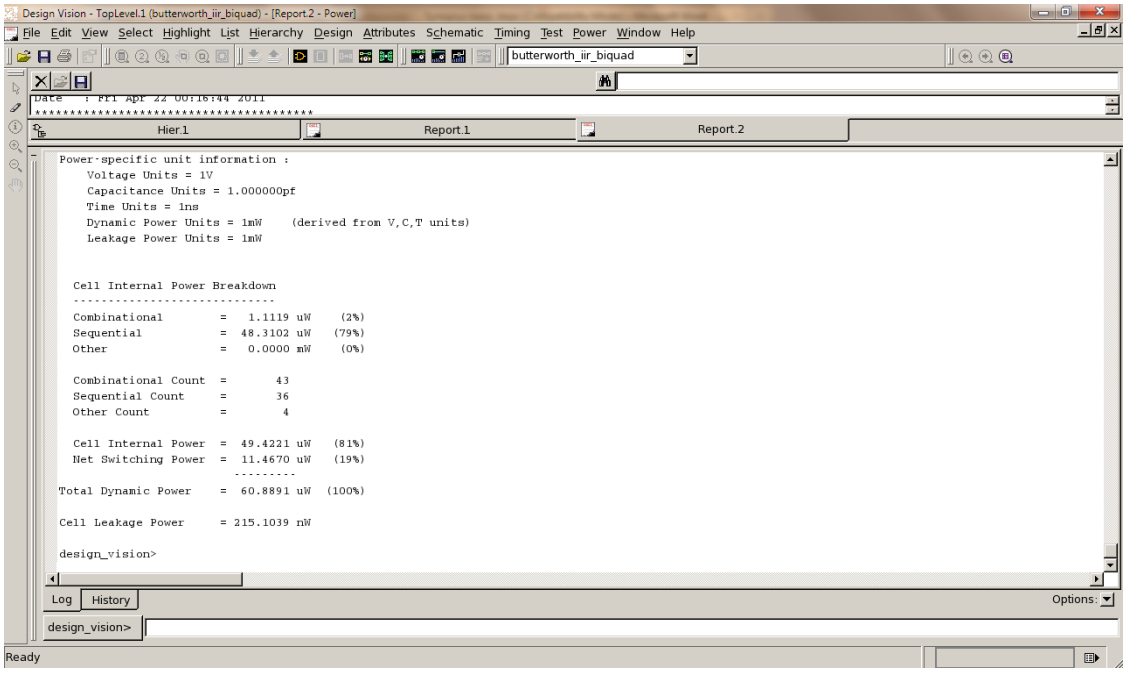
This is how a timing report would look like. If it shows `slack (MET)`, it means you're meeting timing requirements. Otherwise, it will show `slack (VIOLATED)`. You can analyze each timing path in the report to find out what went wrong.



To **report power**, use "Design → Report Power". You should see a dialog box like this.



This is what the result would look like.



Finally, to **save your netlist**, use "File → Save as" and select the filename, as shown below.