DC Synthesizer Lab 1 Summary

For the ***Lab 1 Demo***: Invoke the synthesizer this way:

```
dc_shell  -f  Intro_Top.sct
```

There will be some text messages; then, you should see the `dc_shell` prompt. A `.sct` (Synopsys script) file holds a list of commands to be run before `dc_shell` presents its interactive prompt. We use **`.sct`** in this course to indicate Tcl syntax; also common is a command file name ending in `.scr` ("script").

We'll have plenty of time later to dwell on these messages. For now, you should scroll back and find where you invoked `dc_shell`; then just quickly read forward, making sure there was no error message. The warnings are OK.

What the commands in `Intro_Top.sct` did, was this:

A. They specified the technology and graphical symbol libraries for the synthesizer;

B. they associated the design logical library (in memory) with a directory;

C. they then analyzed and elaborated the design as disc files into that library; they also established operating conditions (`NCCOM` = nominal-case, commercial temperature range) and generic delay parameters (the wire-load model).

After this setup, the commands then set several specific constraints (max allowed area on chip, max allowed delays, estimated loads and drives) and then came to a stop in interactive mode. This design is named `Intro_Top`.

To the `dc_shell` prompt, enter these commands, waiting for each to complete:

```
dc_shell> compile
dc_shell> report_area
dc_shell> report_timing
```

Briefly look over the results. The slack messages mean that the timing requirements were met ("positive") or not ("negative"). Areas are approximate transistor counts, for this library. Now, save the result to disc by writing out a verilog netlist file:

```
dc...> write -hierarchy -format verilog -output Intro_Netlist.v
dc...> exit
```

Next, repeat the run of the DC shell to flatten the netlist hierarchy and improve the speed. Flattening typically is done to improve the timing optimization and the area of the synthesized netlist. Again, invoke

```
dc_shell -f Intro_Top.sct
```

When the setup commands have run, again scroll up briefly to check the messages. Then, assuming no error,

```
dc_shell> ungroup -all -flatten
dc_shell> compile -map_effort high
dc_shell> report_timing
```

The timing has been improved, and, if it did not before, it now should fulfill all constraints in the `.sct` file. Save the synthesized and ungrouped netlist, which no longer has any hierarchy, but don't exit yet:

```
dc_shell> write -hierarchy -format verilog -output Intro_TopFlat.v
```

We'll do one more thing, while we have the synthesized netlist in memory: Write out a Standard Delay Format (SDF) file which contains the netlist timing computed from the library used by the synthesizer. We'll look at this file later in the day.

Be sure to give the new file a name with an SDF extension:

```
dc_shell> write_sdf Intro_TopFlat.sdf
dc_shell> exit
```

A note on "flatten" terminology: To "flatten" the hierarchy in a design, the DC command is **`ungroup`**. In DC, "flatten" refers to boolean logic structure, not to the design hierarchy.

In DC command terminology, a designer may use `set_flatten` to flatten combinational layers of logic to a two-layer boolean sum-of-products. There is a `set_structure` command which factorizes the logic and in a sense is the inverse of `set_flatten`.

---------- ***Optional*** (Step 6 of *Notes* instructions): ----------

Both of the netlists are verilog, so they may be examined in a text editor. However, because they are small and simple, we can convert them to schematics and view them using either VCS or the synthesis GUI interface.

VCS. Exit DC and VCS (if still running) and reinvoke VCS as before. Then, as before, use `[Load session]` and `{F5}` to run the simulation. [*These commands are for old VCS; the new VCS commands are similar.*]

This time, in the initial (control) window, pick "`TestBench (TestBench)`" in the hierarchy window; then, click on the "and gate" icon just above the hierarchy window. This will cause VCS to create a schematic at the `TestBench` level of the design. The testbench and the top level of the design will be shown, with all their connections.

After this, put your cursor on the `Topper01` block of the `TestBench` schematic to highlight it and use the right mouse-button popup menu to select `[Move down to definition]`. This will create a schematic of the top of the design.

You can hover the cursor on a block to see its structural location or contents.

NOTE: A `VCS` hierarchical schematic has been provided in Textbook Fig 1.11.

Synthesis GUI. At the shell prompt, enter `design_vision`. Both DC and `design_vision` use the *TcL* (*T*ool *C*ommand *L*anguage) scripting interface; however, in this Step we shall use only GUI commands.

When the GUI has appeared, use the `[File]/[Read]` command to read in `Intro_Netlist.v`, which you wrote out from DC.

Select `[Intro_Top]` from the drop-down menu. Then, view the schematic by picking the "*and* gate" icon near the middle of the upper menu bar.

Ignoring disproportionate sizes of the blocks *vs*. elementary gates, notice that there isn't much change from the original design schematic in the Course Notes or Textbook, because by default DC preserves design structure and leaves hierarchy intact.

Double-click a block to see its substructure; pick the up-arrow to return.

After viewing the schematic, use `design_vision` **`File/Exit`**.

> NOTE: A `design_vision` hierarchical schematic has been provided in Textbook Fig 1.12.

Then, after exitting DC, view the flattened and optimized netlist schematic: Invoke the synthesis GUI again: `design_vision`

Use the `[File]/[Read]` command to read in the new `Intro_TopFlat.v`.

After viewing the schematic, which now has no hierarchy, pick **`[File]/[Exit]`** to finish the lab.

Run the `Clean` script provided to delete clutter from the lab exercise directory.