ECE 425/520 -VLSI Design & Test Automation
Fall 2015

Synopsys-Design Compiler Tutorial

Instructor: Dr. Spyros Tragoudas
Office: E-0202b
Email: spyros@engr.siu.edu

Teaching Assistant: Pavan Kumar Javvaji
Office: E238
Email: pavan.javvaji@siu.edu

Department of ECE
Southern Illinois University Carbondale

1 Introduction

This tutorial aims in giving a brief introduction on how to use Synopsys Design Compiler(DC) to perform design synthesis. **This is neither a full tutorial, nor a full manual about synthesis using the Synopsys Design Compiler**. The user manual of this tool is available in the path **/synopsys/DOCS/dc/dcug.pdf**. Actually this user manual is one of the available manual under synopsys directory.

1.1 Brief Introduction About Synthesis

Synthesis is an automatic method of converting a higher level of abstraction to a lower level of abstraction. In other words the synthesis process converts Register Transfer Level (RTL) descriptions to gate level net-lists. These gate level net-lists can be optimized for area, speed, testability, etc. The synthesis process is shown in Figure 1. The inputs to the synthesis process are RTL HDL description, circuit constraints and attributes for the design, and a technology library. The synthesis process produces an optimized gate level net-list from all these inputs. Synthesizing a design is an iterative process and begins with defining the constraints for each block of the design. In addition to these constraints, a file defining the synthesis environment is also needed. The environment file specifies the technology cell libraries and other relevant information that the tool uses during synthesis.

1.2 Brief Introduction About Design Compiler

Design Compiler is used to generate optimized gate level net-lists from the RTL models, in the following order,

1. Read the technology library into the synthesis database.
2. Read the HDL source code for the design, written in Verilog or VHDL, into the synthesis database.
3. Generate a generic net-list based on the generic library.
4. Map the generic net-list to cells in the technology library.
5. Optimization of the design under different design constraints.

2 Logic Synthesis using Synopsys Design Compiler

1. Before starting the simulation, make sure that you are in correct directory.

   In this tutorial, the current directory is **/grad/pavan.javvaji/ece425/DC**.

2. To initialize synopsys,

   Type: **synopsys** and click **enter** in the terminal.

   The terminal is changed to synopsys mode.

   Note: Synopsys should be initialized before using synopsys tools, whenever a new terminal is opened.

3. To check whether the synopsys Design Compiler is properly setup or not,

   Type: **which dc_shell** and click **enter**.

   If the terminal shows the path as **/synopsys/DC/bin/dc_shell**, then Design Compiler is properly setup and ready for use.

4. In order to know how to work on Design Compiler, now copy a small VHDL source file **FullAdder.vhd** into your DC directory.

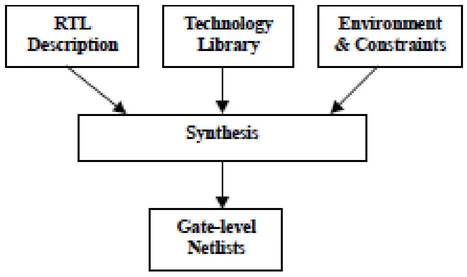


Figure 1. Synthesis

2.1 To invoke Design Compiler

Design Compiler has both the command-line interface and a graphical user interface (GUI). Both provide the same synthesis functions. The tool takes behavioral description of Verilog and VHDL designs and synthesizes it to a gate level net-list.

To launch the Design Compiler in the terminal, Type: **dc_shell -gui** and click **enter**.

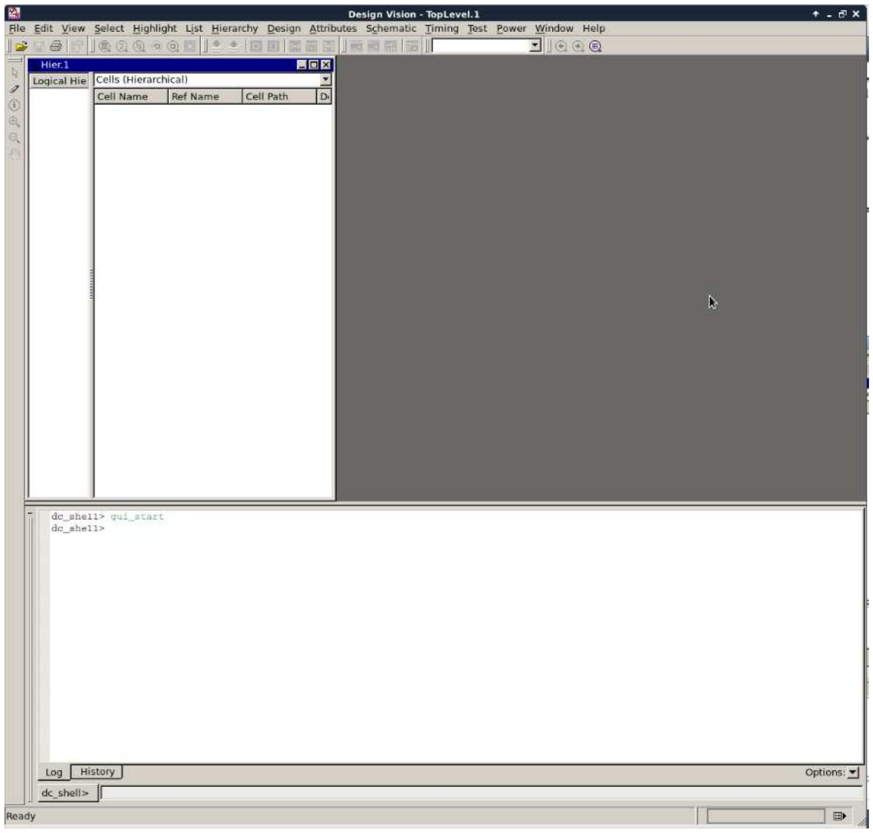
Note: Don't use & in the end of the command.

This command starts Design Vision (GUI) as shown in Figure 2. In the terminal, the path will be changed to **dc_shell**>.

The window has Four areas:

1. The Left and the area adjacent to it are used it see the logical and cell hierarchy of the given design.

2. Schematic viewer (right) displays the design in schematic form. It allows to pan and zoom, display fan-in and fan-out cones, and display critical paths and timing values. It also allows to group instances, dissolve instances, or change the reference point of an instance. Unlike other GUI


Design Vision - TopLevel.1
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power Window Help
Hier.1
Logical Hie
Cells (Hierarchical)
Cell Name
Ref Name
Cell Path
Log
History
Options:
dc_shell>
Ready

```
dc_shell> gui_start
dc_shell>
```

Figure 2. Initial GUI

interfaces, this GUI is the initial window from which *dc_shell* was launched that why it had to be launched in the foreground with out the *ampersand*.

3. Console(below) shows the messages same as in the terminal. Instead of the terminal, commands can be executed from the white space with a tab **dc_shell**> below the Console.

2.2 Setting the library of standard cells

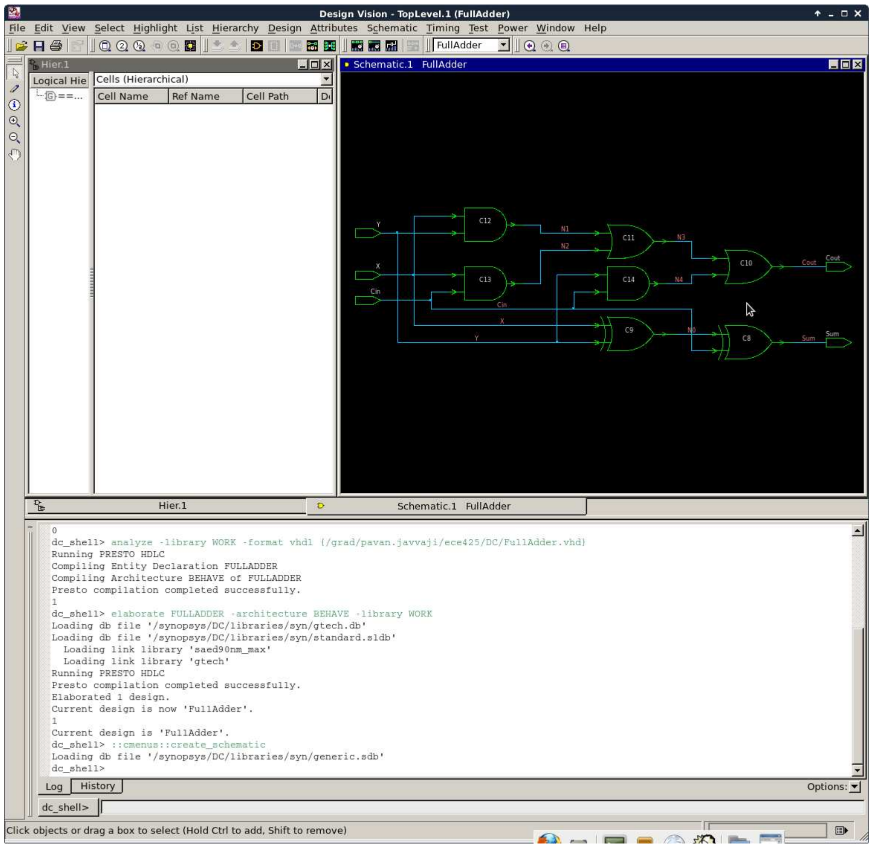
There are many technology libraries available in synopsys. In this course, we use **SAED 90nm** physical and timing libraries. *DC* needs four parameters to set the library.

1. **search_path**: This parameter is used to specify the synthesis tool that it should search this paths when looking for a technology library during synthesis. To do so,

   Type: **set search_path "/synopsys/GPDK/SAED_EDK90nm/Digital_Standard_Cell_Library/synopsys/models"** and click **enter**.

2. **link_library**: This parameter points to the library that contains information on the logic gates in the synthesis technology library. The tool uses this library solely for reference. To do so,

   Type: **set link_library "saed90nm_max.db"** and click **enter**.


Design Vision - TopLevel.1 (FullAdder)
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power Window Help
Hier.1
Schematic.1 FullAdder
Log
History
Options:
Click objects or drag a box to select (Hold Ctrl to add, Shift to remove)

```
0
dc_shell> analyze -library WORK -format vhdl {/grad/pavan.javvaji/ece425/DC/FullAdder.vhd}
Running PRESTO HDLC
Compiling Entity Declaration FULLADDER
Compiling Architecture BEHAVE of FULLADDER
Presto compilation completed successfully.
1
dc_shell> elaborate FULLADDER -architecture BEHAVE -library WORK
Loading db file '/synopsys/DC/libraries/syn/gtech.db'
Loading db file '/synopsys/DC/libraries/syn/standard.sldb'
  Loading link library 'saed90nm_max'
  Loading link library 'gtech'
Running PRESTO HDLC
Presto compilation completed successfully.
Elaborated 1 design.
Current design is now 'FullAdder'.
1
Current design is 'FullAdder'.
dc_shell> ::cmenus::create_schematic
Loading db file '/synopsys/DC/libraries/syn/generic.sdb'
dc_shell>
```

Figure 3. Elaborated Design

3. **target_library**: This parameter specifies the file that contains all the logic cells that should used for mapping during synthesis. In other words, the tool during synthesis maps a design to the logic cells present in this library. To do so,

   Type: **set target_library "saed90nm_max.db"** and click **enter**.

4. **symbol_library**: This parameter points to the library that contains the visual information on the logic cells in the synthesis technology library. All logic cells have a symbolic representation and information about the symbols is stored in this library. To do so,

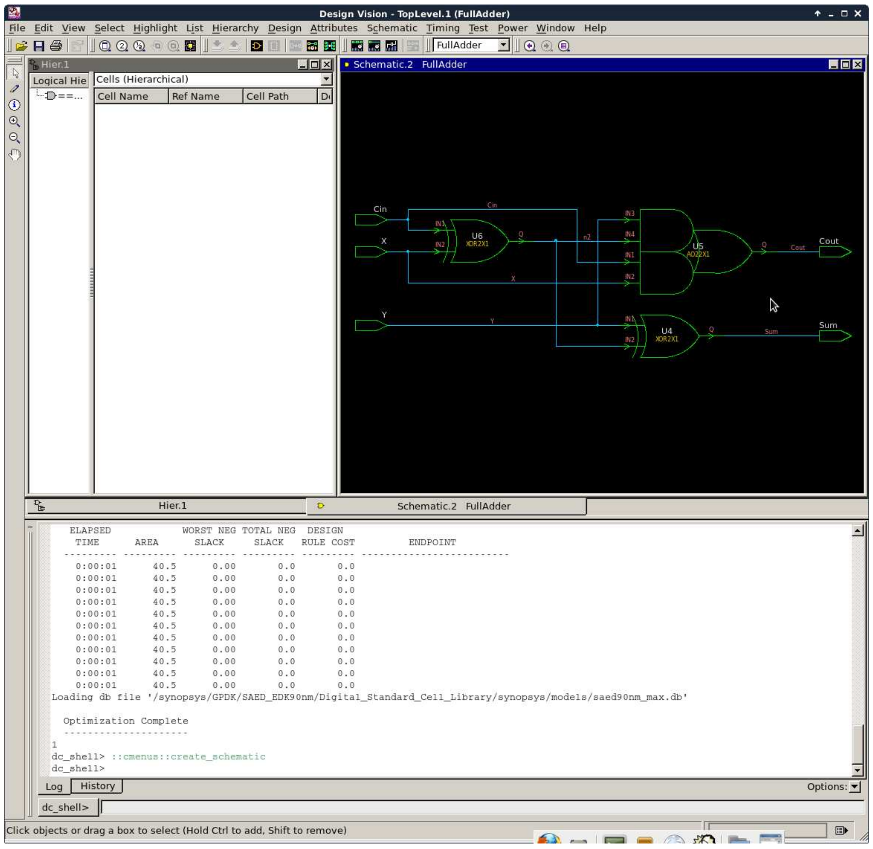
   Type: **set symbol_library "saed90nm_max.db"** and click **enter**.

   Note: In this course, we use same library for all three (link, target, symbol). Use of different library for each of them separately is determined by the application.

The above four commands will set the search path and library of standard cells to be used with the design.

2.3 Loading the HDL file

Design Compiler is not a HDL code compiler, so any errors in the HDL code will not be verified by it. Make sure the HDL code is *error free* and *synthesizable*. The HDL codes should be compiled and simulated using VCSMX or any other HDL simulators, to ensure that they are *error free*. The code


Design Vision - TopLevel.1 (FullAdder)
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power Window Help
Hier.1
Cells (Hierarchical)
Cell Name
Ref Name
Cell Path
Schematic.2 FullAdder
Cin
X
Y
U6
XOR2X1
U5
AO22X1
U4
XOR2X1
Cout
Sum
Log
History
Options:

```
  ELAPSED            WORST NEG TOTAL NEG  DESIGN
   TIME      AREA      SLACK     SLACK   RULE COST         ENDPOINT
 --------- --------- --------- --------- --------- -------------------------
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
   0:00:01      40.5      0.00       0.0       0.0
Loading db file '/synopsys/GPDK/SAED_EDK90nm/Digital_Standard_Cell_Library/synopsys/models/saed90nm_max.db'

  Optimization Complete
  ---------------------
1
dc_shell> ::cmenus::create_schematic
dc_shell>
```

```
dc_shell>
```

Click objects or drag a box to select (Hold Ctrl to add, Shift to remove)

Figure 4. Mapped Design

which contains any time information is referred as *not synthesizable*, in other words all the variables of type *TIME* and all the *after* statements should be removed before the loading the HDL file into Design Compiler. The example file, *FullAdder.vhd* is an *error free* and *synthesizable* code.

To load any HDL file,

Type: **analyze -library WORK -format *vhdl(verilog)*** {***file name with path***} and click **enter**.

In this example to load *FullAdder.vhd*,

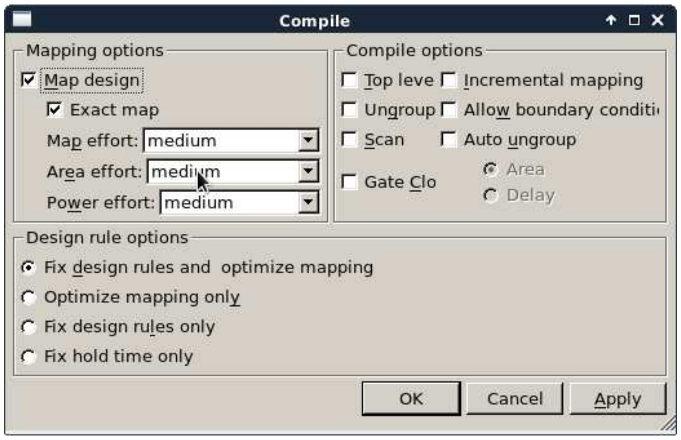
Type: **analyze -library WORK -format vhdl** {**/grad/pavan.javvaji/ece425/DC/FullAdder.vhd**} and click **enter**.

This command will load the *FullAdder.vhd* file into **DC**. If there is any errors, the terminal will show the errors and warnings.

2.4 Elaboration

Elaboration is the step to convert the design from the HDL description into a synopsys specific format that is required for synthesis.

For Elaboration,


Compile

Mapping options

[✓] Map design

[✓] Exact map

Map effort: medium

Area effort: medium

Power effort: medium

Compile options

[ ] Top leve [ ] Incremental mapping

[ ] Ungroup [ ] Allow boundary conditi

[ ] Scan [ ] Auto ungroup

[ ] Gate Clo [✓ Area] Delay

Design rule options

[✓ Fix design rules and optimize mapping]

Optimize mapping only

Fix design rules only

Fix hold time only

Figure 5. Compile

Type: **elaborate *design name* -architecture *architecture name(verilog)* -library WORK** and click **enter**.

In this example to elaborate FullAdder,

Type: **elaborate FULLADDER -architecture BEHAVE -library WORK** and click **enter**.

After elaborating the net-list of the design can be seen in the GUI window like as shown in Figure 3. Also the GUI window allows to freely navigate through the net-list, by zooming in and out, selecting components or even inspect designs at lower levels by double-clicking on them. The elaboration in **DC** synthesis the design with *gtech* library, the default generic library of synopsys.

To see schematic diagram, *click* $\rightarrow$ *Schematic* $\rightarrow$ *New Design Schematic View*.

2.5 Link

Now the FullAdder circuit is ready for link to the target library specified.

To do so Type: **link** and click **enter**.

After the above command the circuit is the same as before, because till now, we link the technology library, but we did not map the design to the that library. To do so,

Type: **compile -exact_map** and click **enter**.

The design will be mapped to the specified technology defined in the library attribute that was set before. Now the net-list of selected component will be totally different as shown in Figure 4.
Since the **compile** can be executed with different options, the above command does the mapping with the default settings. The default settings of the **compile** is shown in Figure 5. The above mentioned steps are standard procedures to load the HDL files, synthesize and map to particular technology Library.

2.6 Export the results

The synthesized design can be exported after this synthesis process in a Verilog description(which is suitable for importing the design to other tools). For the given example,

To get the synthesized HDL code from **DC**:

Type: **write -hierarchy -format** ***vhdl(verilog)*** **-output** ***file name with path*** and click **enter**.

In this Example, Type

write -hierarchy -format verilog -output /grad/pavan.javvaji/ece425/DC/FullAdderMap.v and click **enter**.

Note: To know more about the DC commands and their usage click *Help* → *Man Pages*. In *DC*, when you use the *elaborate, link or compile* command, the memory is updated and previous information will be removed. To get the code of the intermediate files, you need use the above command before proceeding to the next step.

2.7 Inspection of Results, Redirecting Reports, and removing the Designs

The *DC* GUI, visualizes some of the various reporting and design analyzing features of the tool. Start by clicking the *Design* and *Timing*, you will see lot of report options on various attributes. Take some time to explore the various reported values and information.
The report command sends the output to stdout by default; you can redirect stdout information to a file.
To write the *area report* of the synthesized *FullAdder* to a file called *areaFA.txt*
Type, **report_area > areaFA.txt** and click **enter**.

Use **Man Pages** to see all the possible reports to be produced by this tool.

If you want to remove any designs from your *DC* memory.
Type, **remove_*design name*** and click **enter**. **Note: *design name* is the entity or module name in the HDL file**.

2.8 Large Designs

For larger designs with many HDL files, the procedure of loading and synthesize procedure from the command line becomes very time-consuming and error-prone. The Design Compiler provides an easy and efficient way of combining commands into *script* files. An example file, called *FullAdder.script* has been given. These lines are nothing more than all the commands given in the previous procedure. Go through the commands in this *script* file, before executing. To execute the file, go to the GUI window and click on menu File → Execute Script. Then select the *FullAdder.script* file and click OK. This will performed all the steps described above.

3 Obtaining Minimum Area with Timing Constraints Using Design Compiler

Design Compiler is used for optimizing Area, Timing, Power and many other design parameters of the given design. Design constraints specify the goals for the design. They consist of area and timing constraints. Depending on how the design is constrained the **DC** tries to meet the set objectives. This section aims in giving a brief introduction on how to use Synopsys Design Compiler to obtain feasible small area for the given design in the presence of some timing constraints. **This is neither a full tutorial, nor a full manual about Area/Time optimization using the Design Compiler**.

The following steps needs to be done in the order specified to obtaining feasible small area for the design with some timing constraints using **DC**. The clock should be specified prior to any other attributes because all other timing constraints can be provided only with respect to the clock.

create_clock: This command is used to define a clock object with a particular period and waveform. The **period** option defines the clock period, while the **waveform** option controls the duty cycle and the starting edge of the clock. This command is applied to a pin or port, object types.

Example: create_clock −period 40 −waveform {0 20} {CK}
This command means that a port named **CK** is of type **clock** that has a period of *40 ns*, with 50% duty cycle. The positive edge of the **CK** starts at time *0 ns*, with the falling edge occurring at *20 ns*. By changing the falling edge value, the duty cycle of the **CK** may be altered.

There are many timing constraints or operating environment conditions can be applied to a design during synthesis process in **DC**, we see just four of them.

1. **set_input_delay**: It specifies the input arrival time of a signal in relation to the clock. It is used at the input ports, to specify the time it takes for the data to be stable after the clock edge. The following two commands tells the minimum and maximum delay for input ports.

   Example: set_input_delay 0.1 −min −clock CK [remove_from_collection [all_inputs] [get_port CK]] and

   Example: set_input_delay 0.4 −max −clock CK [remove_from_collection [all_inputs] [get_port CK]]

   This command means that the data in all input ports except CK port takes *0.1 to 0.4 ns* to become stable after change in the **CK**.

2. **set_output_delay**: This command is used at the output port, to define the time it takes for the data to be available before the clock edge.

   Example: set_output_delay 0.4 −max −clock CK [all_outputs] and

   Example: set_output_delay 0.1 −min −clock CK [all_outputs]

   This command means that the data in the all output ports available *0.1 to 0.4 ns* before the **CK**.

3. **set_wire_load_model**: Design Compiler calculates the Interconnect wiring and transition delays using the wire load model defined in the target library.

   Example: set_wire_load_model −name 8000 −library saed90nm_max.

This command means that the delays values specified in the **saed90nm_max** library were going to be used for the wires in the design, while synthesizing this design.

4. **set_max_area**: This constraint specifies the maximum area a particular design should have. The value is specified in units used to describe the gate-level macro cells in the technology library.

   Example: set_max_area 0 .

   Specifying a **0** area might result in the tool to try its best to get the design as small as possible.

Once we apply the above operating conditions are applied and complied with no errors. Then, Check the timing using the command **check_timing** and check if the slack is MET(worst NEG slack =0). Then, you obtained the design with feasible minimum area meeting the timing constraints. If not then, modified your clock values and compile, until the slack is MET with minimum area overhead.