Digital Logic Synthesis and Equivalence Checking Tools

Hardware Verification Group

Department of Electrical and Computer Engineering,
Concordia University, Montreal, Canada

CAD Tool Tutorial

May, 2010

Abstract

This document contains a brief introduction to Synopsys Design Analyzer, Sysnopsys Formality, and Cadence Conformal tools. You would need approximately three hours to finish this tutorial.

Revision History

Version	Authors	Modification	Section	Date
1.0	Naeem Abbasi	Document Creation	-	April 16, 2010
2.0	Naeem Abbasi	Section on Synopsys Formality added An additional appendix added Title modified	3 B -	April 18, 2010
2.1	Naeem Abbasi	Added Figures 1 and 2 and their descriptions	1, 2.1	April 19, 2010
3.0	Naeem Abbasi	One's counter example added (provided by Ted Obuchowicz)		May 3, 2010
3.1	Naeem Abbasi	Screen Captures Updated Remote Access Commands added to the appendix Minor changes in the Appendices		May 3, 2010

Table of Contents

List of Figures

1 Introduction

Logic synthesis usually refers to the process of translation of RTL design into an optimized gate level description. Logic equivalence checking refers to a technique that mathematically (i.e. with out simulation) verifies that the two design descriptions are functionally equivalent.

Section 2 of this tutorial describes how to setup and synthesize an RTL description into a structural netlist of gates using Synopsys design analyzer. During this process, the design is read in, analyzed and elaborated. After that, the designer identifies critical nets in the design such as the clock signal. Then area and timing constraints are specified for guiding the design optimization. Once the design is optimized, it is exported to a gate level netlist. This completes the synthesis step.

Section 3 and 4 of this tutorial describe use of equivalence checking tools. An equivalence checking tool takes two descriptions of a design and verifies if they are functionally equivalent (see Fig. 1). In case if the two descriptions are not equivalent, a counter example is produced.

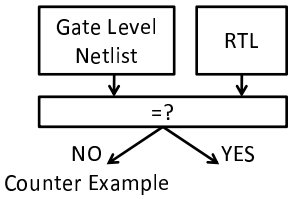

Fig. 1: Equivalence Checking

Section 3 of this tutorial describes how to check if the synthesized design is equivalent to the RTL design using Sysnopsys tool Formality. The process begins by first reading in the two design descriptions. Then those ports and nets are selected for which we wish to verify functional equivalence. The equivalence checker is then run which either verifies the equivalence of the two designs or helps in debugging by identifying the failing points, ports, and nets. Failing points in the reference and implemented design can be viewed side by side in a schematic browser.

Cadence Conformal suite of tools contains a tool called Logic Equivalence Checker or LEC. Section 4 of this tutorial describes how to formally verify that the synthesized design is functionally equivalent to the RTL description using LEC. This section can be skipped if one chooses to use Formality for equivalence checking. During the equivalence checking process, the two design descriptions are first read in, then key mapping points in the two design descriptions are selected, and finally the equivalence checker is run. The generated reports show if the the two designs are equivalent. Similar to Formality, Conformal LEC also has Schematic and Netlist browser features, which are extremely helpful in debugging.

Appendix A, B and C contain the source code for the examples used in this tutorial along with some other useful information regarding these tools.

1.1 Synopsys Design Analyzer

Synopsys Design Compiler (DC) is a logic synthesis and design optimization tool. The synthesis and optimization steps, described in this tutorial, can be easily converted to a script, which can later be modified and run from the command line interface.

More information about Synopsys design compiler (DC) can be found in
[/CMC/tools/synopsys.2005a/syn/doc/syn/tutorial],
[/CMC/tools/synopsys/syn/doc/online/top.pdf], and
[/CMC/tools/synopsys.2005a/syn/doc/syn/examples]

1.2 Synopsys Formality

Formality is an equivalence checking tool. More information about Synopsys Formality can be found in [/CMC/tools/synopsys/fm/doc/fm]

1.3 Cadence Conformal

Cadence Conformal Logic Equivalence Checker (LEC) is a formal logic equivalence checking tool. More information about LEC can be found in
[/CMC/tools/cadence.2007a/CONFRML/doc]

2 Synopsys Design Compiler

2.1 Introduction

Logic synthesis is a process that translates an RTL description of a circuit into an optimized netlist consisting of flipflops, latches, and logic gates. Design engineers provide HDL descriptions and various constraints and bounds on the design to synthesis tools. These constraints reflect the needs that the design must meet. For example minimum area, minimum speed and maximum power dissipation.

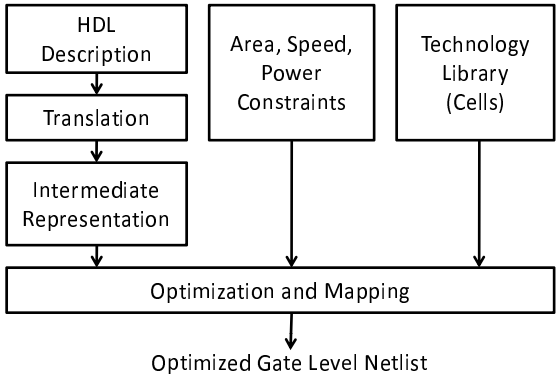

Fig. 2: Digital logic synthesis

Figure 2 shows a typical high level flow used in most logic synthesis tools. Logic synthesis is usually done in three steps. First the RTL description is translated to an unoptimized boolean description. HDL Constructs such as IF, CASE, LOOPs and conditional assignments are converted to their equivalent boolean equivalents consisting of primitive gates such as NAND and NOR gates, flipflops and latches etc.

Such descriptions are functionally correct but are completely unoptimized. This step is followed by the logical optimization step. The boolean optimization algorithms are used to produce an optimized equivalent description. These algorithms utilize logic flattening and logic factoring operations together with fanout and loading constraints to optimize the logic. During flattening operations remove structure while the factoring operations introduce new structure. These operations when applied in conjunction with each other help optimize the logic.

Finally, the optimzed boolean equivalent description is mapped to actual logic gates by making use of a technology library of the target process. This step uses logical and timing information from the technology library to build a netlist. The generated netlist meets the area and speed needs of the user.

At the end of this process several techniques are used to ensure that the optimized netlist is functionally equivalent to the RTL design and also does not violate any of the rules of the technology.

2.2 Design example used in this tutorial

In this tutorial, we will utilize the following design example. This design implements a purely combinational circuit that counts the number of 1s in the 4 bit input vector.

```
-- T. Obuchowicz
-- counts the number of 1s in the 4 bit input vector
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ones_counter is
    port(
        a : in  std_logic_vector(3 downto 0);
        f : out std_logic_vector(2 downto 0));
end;

architecture  rtl  of ones_counter  is
begin
    process(a)
    variable  ones : std_logic_vector(2 downto 0);
    begin
        ones := (others => '0');
        for index in a'range loop
            if a(index) = '1' then
```

```
                ones := ones + "1";
            end if;
        end loop;

        case ones is
            when "000" => f <= not("000");
            when "001" => f <= not("001");
            when "010" => f <= not("010");
            when "011" => f <= not("011");
            when others => f <= not("100");
        end case;
    end process;
end;
```

In the next section, we will synthesize this behavioral description of the design using Synopsys Design compiler.

2.3 Setting up the Environment and Important Libraries and File

In this tutorial, we will be using standard cell library "class.db". The following steps set-up the Design Compiler (DC) environment:

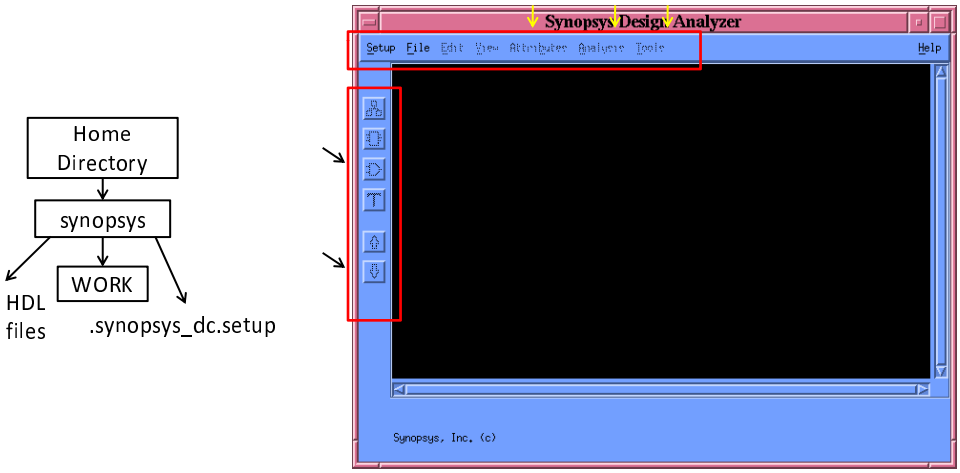

Fig. 3: Directory structure and design analyzer graphical user interface

1. Create a separate directory. `mkdir synopsys`. Change directory to synopsys. `cd synopsys`
2. Source the environment script. `source /CMC/ENVIRONMENT/synopsys.env`
3. Copy the `.synopsys_dc.setup` file into the `synopsys` directory. Copy the HDL files (ones.vhd) to the synopsys directory.

4. Start DC interface by typing `design_analyzer` at the unix prompt. You may run the design_analyzer in command mode by entering: `dc_shell -f ScriptFileName`
5. Analyze RTL descriptions of the design. File ⇒ Analyze. Select Create New Library if it doesn't already exist. Then click OK
6. Load top-level RTL file via File ⇒ Read menu (ones.vhd). Then press OK.
7. Observe DC generated messages carefully to make sure that correct libraries are loaded
8. Select the top-level design module (one_counter) and click on Analysis, and then on Link Design. Observe and make sure that the correct link libraries are loaded. Then press OK.
9. Check your design by selecting Analysis ⇒ Check Design menu. If a value of 1 is returned it means that the design check command executed successfully and that there were no errors in the design. Most warnings can usually be ignored. Design Analyzer user interface allows one to navigate up and down through the design hierarchy. It is also possible to switch between symbol and schematic views (see Fig. 4).

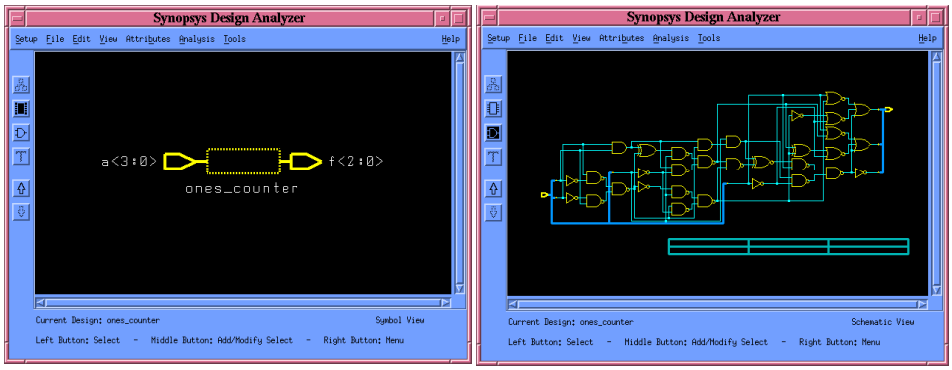

Fig. 4: Symbolic and schematic views

2.4 Setting the Design Attributes

2.4.1 Clock Attributes

Following steps can be used to define the clock signal attributes such as the clock period and skew etc. (You can skip this section as the design used in this tutorial is purely combinational)

1. Select CLK pin in the Symbol View
2. Select Attributes ⇒ Clocks ⇒ Specify (see Fig. 5). Enter clock period in nanoseconds. Un check Don't Touch Network. This will force DC to insert clock tree buffers if needed to meet the design optimization requirements.

3. Click Apply and then close the dialog window.

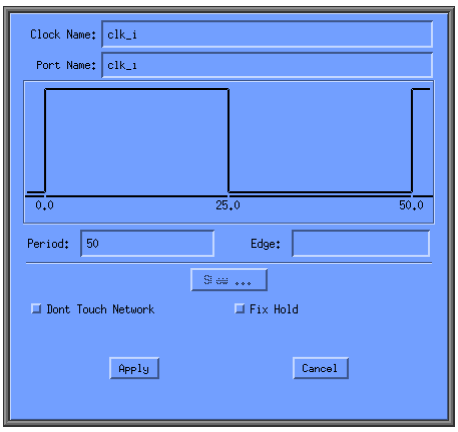

Fig. 5: Defining clock

2.4.2 Specifying the Output Load

Capacitive load on the design outputs can be specified as follows:

1. Select the desired input or output pin
2. Select Attributes ⇒ Operating Environment ⇒ Load (see Fig. 6)
3. Enter capacitive load in pF (e.g., 10pF)
4. Click Apply and close the dialog window.

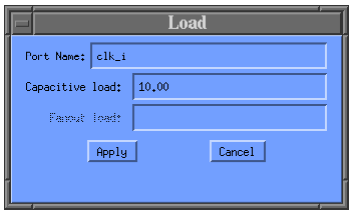

Fig. 6: Specifying capacitive load

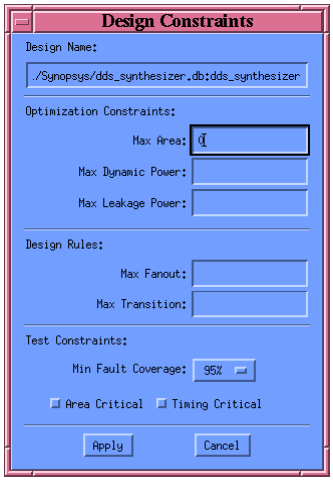

Fig. 7: Area constraints

2.5 Specifying the Design Constraints

2.5.1 The Area Constraints

The Area constraints guide the area optimization process during design synthesis.(you can skip this section as we are not interested in area optimization at this time).

1. Select top level module in Symbol View. Select Attributes ⇒ Optimization constraints ⇒ Design constraints (see Fig. 7)
2. Set Max Area to 0 (μm^2). This will force the design compiler to optimize for smallest possible area.
3. Apply the area constraints and then close the dialog window.

2.5.2 The Timing Constraints

Following steps can be used to specify the rise and fall delay time constraints for the design. (you may skip this step as well, as we are not interested in timing optimization.)

1. Select the top-level module in Symbol View. Select Attributes ⇒ Optimization Constraints ⇒ Timing Constraints (see Fig. 8)
2. Specify the timing constraints by selecting appropriate Input/Output ports in the schematic view
3. Set the Rise and Fall time option as desired. Equal rise and fall times option is most commonly used.
4. Click Apply and update the timing constraints and then close the dialog window.

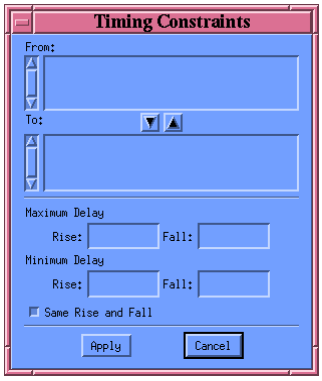

Fig. 8: Timing constraints

By default, Design Compiler optimizes for timing first and then for the area when presented with conflicting goals.

2.6 Synthesis and Optimization

The steps for synthesizing and optimizing the design are as follows:

1. Click Tools ⇒ Design Optimization (see Fig. 9)
2. Select Map Effort level (Low, Medium, High) and then click OK.
3. Observe the log file for errors and warnings. If the synthesis and optimization process completes successfully a value of 1 is returned. Most warnings can usually be ignored.
4. If there were no errors then close the dialog window.

The synthesis and optimization steps can take a long time to finish if high map and verify effort options are selected.

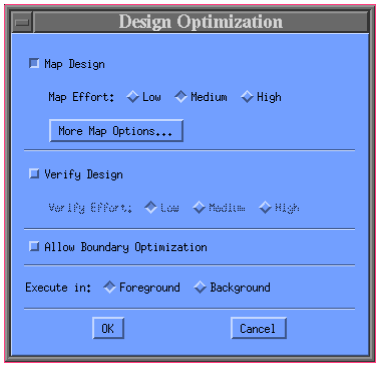

Fig. 9: Design optimization

2.7 Analysis Report

To produce analysis report for the design:

1. Select Analysis ⇒ Report (see Fig. 10). Customize the report by choosing the desired options.
2. Analysis report can be saved to a file by selecting the "File" option in "Send Output To" field (see Fig. 10)
3. Finally, click on Apply to generate the synthesis and optimization report.
4. The critical path can be viewed by selecting Analysis ⇒ Highlight ⇒ Critical Path, in the Schematic View.

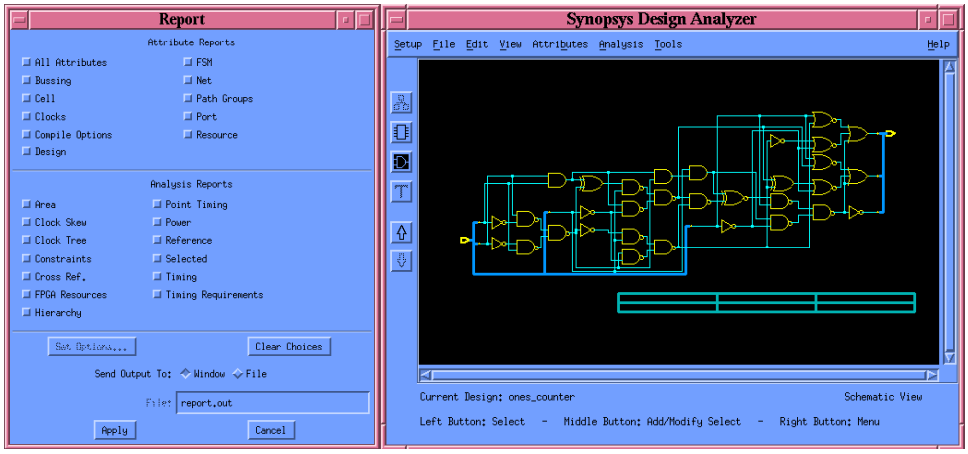

Fig. 10: Analysis report

2.8 Exporting the Design Files

The results of DC synthesis and analysis can be saved as a VHDL or a verilog netlist:

1. Select File ⇒ Save As. Enter file name. Select VHDL or Verilog under File Format and then press OK. [Lets save the synthesized design in VHDL format. Name the synthesized design as "ones_syn.vhd"]
2. To export timing constraints, click on Files ⇒ Save Info ⇒ Constraints. Select a filename, and then press OK.

   Further explore the Setup menu. (1) Open a command window (Setup ⇒ Command Window) and go through the synthesis steps one more time. This time observe more carefully different design analyzer commands being executed. (2) Also explore (Setup ⇒ Defaults), (Setup ⇒ Variables) and (Setup ⇒ Execute Script)
3. Close the dialog window and exit Design Analyzer.

Main steps involved in the synthesis and optimization of RTL designs were described in this section. For more information, please see
[/CMC/tools/synopsys.2005a/syn/doc/syn/tutorial],
[/CMC/tools/synopsys/syn/doc/online/top.pdf], and
[/CMC/tools/synopsys.2005a/syn/doc/syn/examples]

Next two sections of this tutorial will describe two equivalence checking tools, namely, the Cadence Conformal Logic Equivalence Checker, and the Synopsys Formality.

3 Synopsys Formality

3.1 Introduction

Formality is a functional equivalence checking tool from Synopsys. The equivalence checking process flow is shown in Fig. 11. After initial environment setup, separate containers for the reference and implemented design are created. Libraries and the design files are then loaded. Verification is run after top level design modules and compare points have been identified. In case the two designs are not equivalent, Formality identifies the problem areas and displays them in both schematic and HDL source views. Formality can be run in both command line and gui modes.

More information can be found in [/CMC/tools/synopsys/fm/doc/fm]

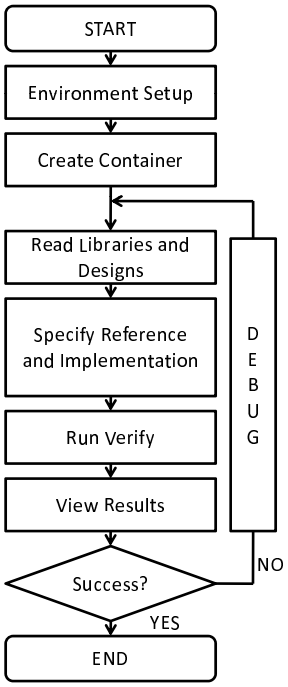

Fig. 11: Formality equivalence checking flow

3.2 Setting up the Environment

Before you begin, setup the environment as follows:

1. Create a separate work directory. `mkdir formality`. Change directory to formality. `cd formality`
2. Source the formality environment script. `source /CMC/ENVIRONMENT/formality.env`
3. Copy the HDL files ("ones_syn.vhd" and "ones_syn1.vhd") into the formality directory. The files used in this tutorial can be found here:

[http://users.ece.concordia.ca/~tahar/coen7501/coen7501.proj.html].

4. Also copy the "class.db" primitive cell library into the formality directory. It can be found here:

   [/CMC/tools/synopsys/syn/libraries/syn/class.db]

3.2.1 Required Libraries and Files

We will be using following files in this tutorial:

1. The reference design (RTL design): `ones_syn.vhd`
2. The implemented design (gate level design): `ones_syn1.vhd`
3. The primitive cells library: class.db

3.3 Starting Formality

To start formality, type `formality &` at the unix prompt. The main formality window will appear after a few second (see Fig. 12). Formality can be run in command mode by entering: `fm_shell`

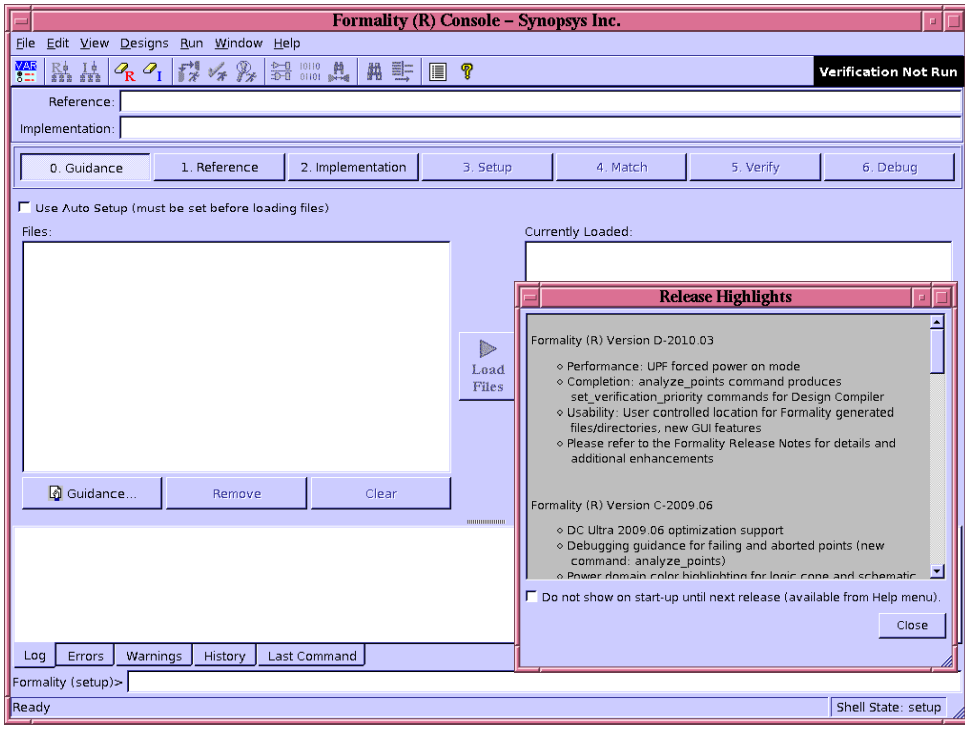

Fig. 12: Starting formality

Please see the Formality Users Guide (user.pdf) for a detailed description of the graphical user interface [/CMC/tools/synopsys/fm/doc/fm].

3.4 Design Input

3.4.1 Reading the Reference Design

Click on the "1. Reference" button.

1. Read Design Files: In VHDL tab, click on VHDL button. Select `ones.vhd`, then click open. Select the `ones.vhd` file and then click on Load Files button to load the reference design file. Repeat the same steps until all reference design files are loaded.

2. Read DB Files: Observe that GTECH library is selected by default. Read in the "class.db" file.

3. Set Top Design: Choose `ones_counter` and click on Set Top Design. You will now see a green check mark on "1. Reference" button. In the formality window, observe the log of executed commands. Click on Set Reference button to add the design files to the reference container.

3.4.2 Reading the Implemented Design

Click on the "2. Implementation" button.

1. Read Design Files: In VHDL tab, click on VHDL button. Select `ones_syn.vhd`, then click open. Select the `ones_syn.vhd` file and then click on Load Files button to load the reference design files. Repeat the same steps until all reference design files are loaded.

2. Read DB Files: Observe that GTECH library is selected by default. Read in the "class.db" file.

3. Set Top Design: Choose `ones_counter` and click on Set Top Design. You will now see a green check mark on "2. Implementation" button. In the formality window, observe the log of executed commands. Click on Set Reference button to add the design file to implementation container.

Note: You may have to read in certain libraries and packages, in some cases, before you read the reference and/or implemented design netlists.

Figure 13 shows Formality window when reference and implemented designs have been successfully loaded into two separate containers and are ready for equivalence checking.

3.5 Setup

Click on "3. Setup" button. Constants, design parameters, and equivalences between ports and nets can be set here. In this tutorial, we will skip this step. By default Formality will check for port equivalence. For more information about this step please consult Formality users guide (user.pdf) [/CMC/tools/synopsys/fm/doc/fm].

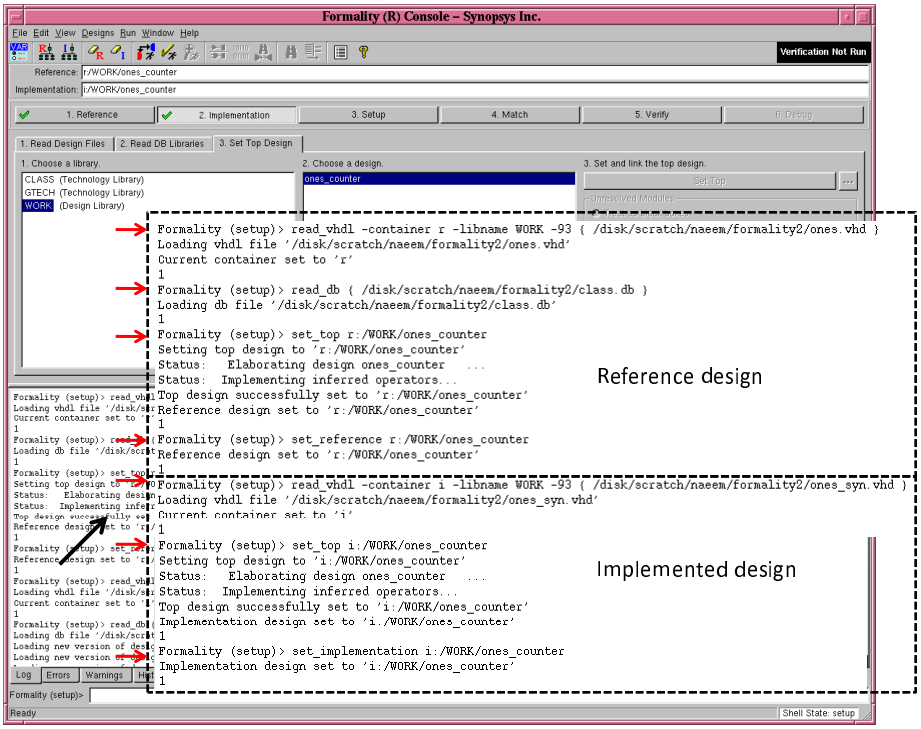

Fig. 13: Reference and implemented designs loaded and ready for equivalence checking

3.6 Match

Click on "4. Match" button. Click on Run Matching button. You will see match command execution and its results in the Formality console (see Fig. 14).

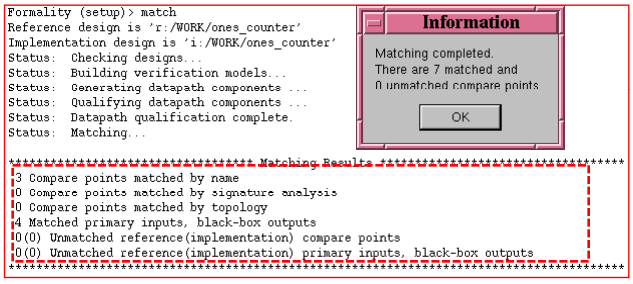

Fig. 14: Match command execution results

3.7 Verify

Click on "5. Verify" button. As the verification process runs you will see its progress and when it finishes the verification results are printed in the Formality console window (see Fig. 15).

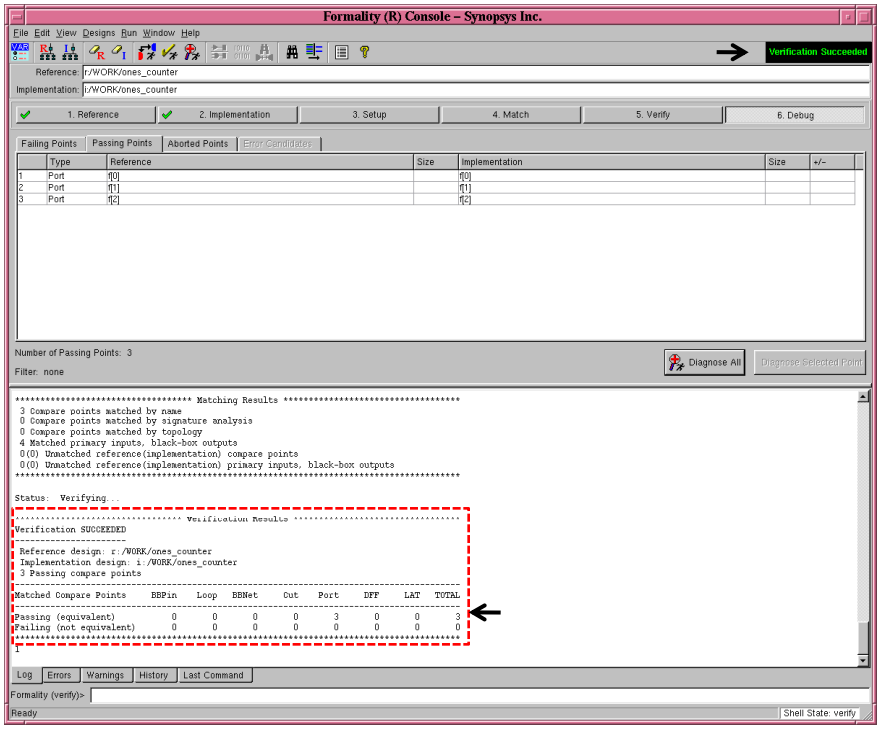

Fig. 15: Equivalence successfully verified

3.8 Debug

In case if equivalence checking fails, one can run the "diagnose" command to locate points which did not match. To trace the possible problem, we first determine possible error candidate ports or nets. We then select one of many possible options for debugging. These options contain viewing logic cones, signal patterns, schematic and source views etc.

Now lets take an example where we debug an equivalence problem. You will require the following design (see Fig. 16) and library files for this example.

1. Reference design (ones_syn.vhd)
2. Implemented design (ones_syn1.vhd)
3. primitive cell library (class.db)

Now using the steps described in previous sections, load the reference (ones_syn.vhd) and implemented (ones_syn1.vhd) designs. Next run the verification as before. The verification will fail this time (Fig. 17). (1) Now select a failing point. Right click on the failing point and click "diagnose". You will notice that the "Analyses" tab is enabled. Click on the "Analyses" tab. (3) Select an error candidate. Right click on "Show Logic Cones". A schematic window (Fig. 18) containing logic cones starting from the compare points all the way back to the inputs will appear. You will see the reference design in the top window and the implemented design in the bottom window. (4) Click on the "Isolate Error Candidates" button in the tool

bar or press F8 function key. You will see a cone appear in the implemented design schematic window. Notice that the output f[1] is not equivalent because the implemented design contains a two-input XOR gate, whereas, the reference design has a two-input NOR gate. (5) Fix the implemented design and re run the verification. (6) You may also use the Apply Pattern feature to further diagnose the compare points. Select an error candidate. Right click ⇒ Show Patterns. The show patterns window shows the reference and implemented design inputs side by side, together with input patterns which cause the outputs under investigation to be unequal. Click on columns numbered 1 through 5 one by one and observe the reference and implemented compare point values. You will see that they are not equal.

For more details consult chapter 8 of the Formality users guide (user.pdf) [/CMC/tools/synopsys/fm/doc/fm].

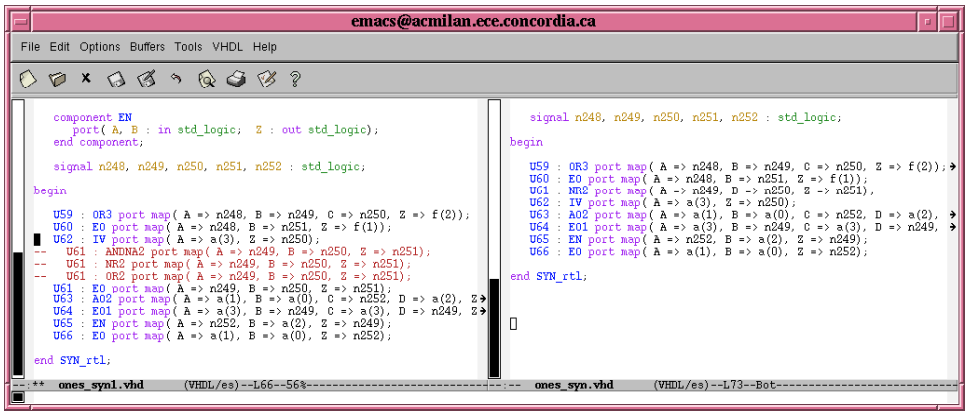

Fig. 16: Reference and Implemented design netlists

At this point you can close Formality GUI. It is possible to save the state of the session at any point in the verification process and reload it at a later stage (File ⇒ Save Session, File ⇒ Restore Session).

3.9 How to create a command script

1. A list of all the commands executed in any session is kept in the `fm_shell_command.log` file. This script file can later be modified and re run from `fm_shell`.
2. A transcript of the session can be saved as a `Tcl` script (File ⇒ Save Transcript).

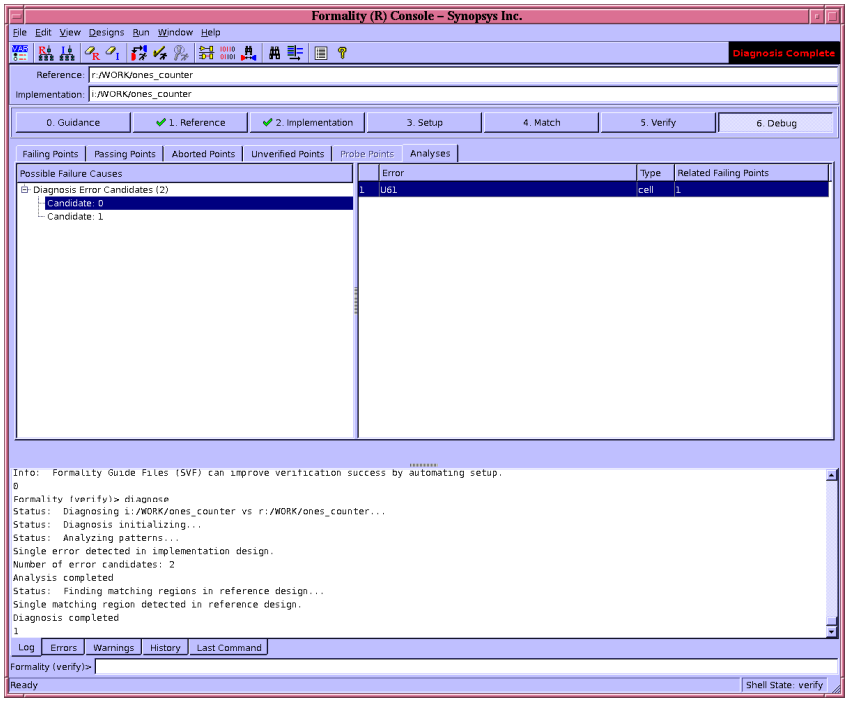

Fig. 17: Error candidates

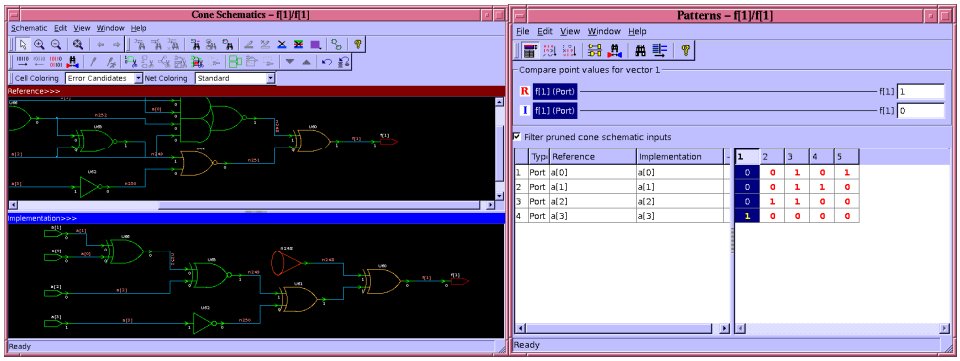

Fig. 18: Logic cones and patterns

4 Cadence Conformal

4.1 Introduction

Conformal LEC is a powerful equivalence checking tool. It can provide a formal proof that the output from Synthesis matches the original RTL code. It can do all of that without having to run a single simulation. In this section of the tutorial, we will learn how to read in a RTL and a synthesized design and how to prove that they are functionally equivalent. The original RTL netlist is usually referred to as the "golden" design. It serves as the reference

for the comparison. The synthesized gate-level netlist is also called the "Revised" design.

4.2 Setting up the Environment

Before you begin, setup the environment as follows:

1. Create a separate work directory. `mkdir conformal`. Change directory to conformal. `cd conformal`
2. Source the conformal environment script. `source /CMC/ENVIRONMENT/conformal.env`
3. Copy the HDL files (both golden and revised) and the primitive cell library file into the conformal directory.

The files used in this tutorial can be found here:
[http://users.ece.concordia.ca/∼tahar/coen7501/ones_syn.vhd], and
[http://users.ece.concordia.ca/∼tahar/coen7501/ones_syn1.vhd] Also copy the "class.db" primitive cell library into the formality directory. It can be found here:
[/CMC/tools/synopsys/syn/libraries/syn/class.db]

4.2.1 Required Libraries and Files

Following files are used in this tutorial:

1. The RTL design: `ones.vhd`
2. The gate level design: `ones_syn.vhd`
3. The primitive cells library: class.lib
4. The conformal script: `lec.do`

4.3 Starting Conformal

To start the tool, type `lec -XL &` at the unix prompt. The main Conformal LEC window will appear (see Fig. 19). Conformal can be run in command mode by entering: `lec -nogui`

4.4 Design Input

4.4.1 Reading the RTL netlist

Read the RTL design: Click on "File ⇒ Read Design". In the Design option: Select VHDL format. Click on "ones.vhd". Set the "Type" to "Golden". Click on Add Selected. Then load the file by clicking "OK" (see Fig. 19)

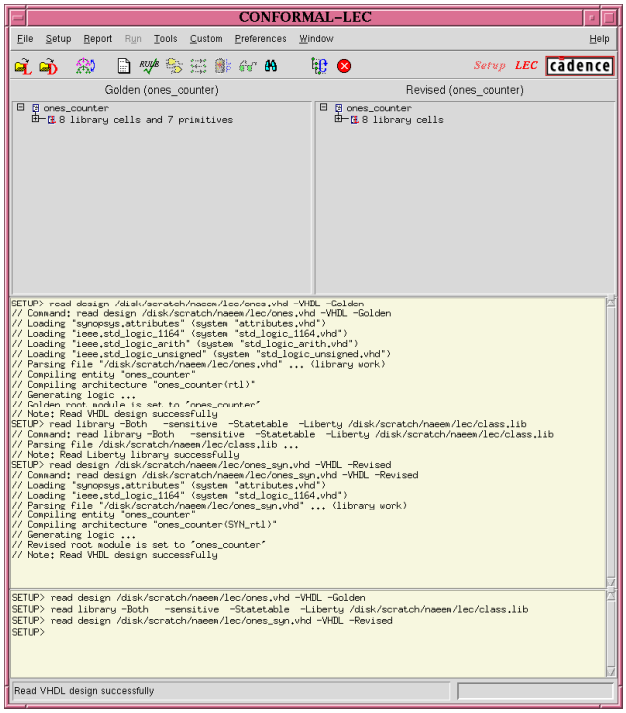

Fig. 19: Confomral LEC

4.4.2 Reading the Gate-level Netlist

If your gate level design uses a standard cell library, you may have to load the library files before you load the design files. In case of hierarchical designs, the rule is to always load from bottom up. File ⇒ Library. Select “class.lib”. Read the synthesized design: Click on File ⇒ Read Design. In the Design option: Select VHDL format. Select ”ones_syn.vhd”. Set the ”Type” to ”Revised”. Click on Add Selected. Then load the file by clicking OK (see Fig. 19).
Note: You may have to read in certain libraries and packages in some cases, before you read the golden and/or revised netlists.

4.5 Design Preparation and Key point Mapping

Since both the designs have been successfully loaded, we can now start the verification process. Conformal has 2 operating modes, the ”Setup” and the ”LEC” mode. Switch to the LEC mode by Clicking on the ”LEC” icon in the upper right hand corner of the window (see Fig. 20).

A table is now printed in the conformal LEC window. It lists the primary inputs (PI) and primary outputs (PO) in both the revised and golden designs. They are equal if the golden and revised designs have the same number of inputs and outputs. To run the equivalence checker, select ”Run ⇒ Compare” and click ”OK” (see Fig. 20). The equivalence checker

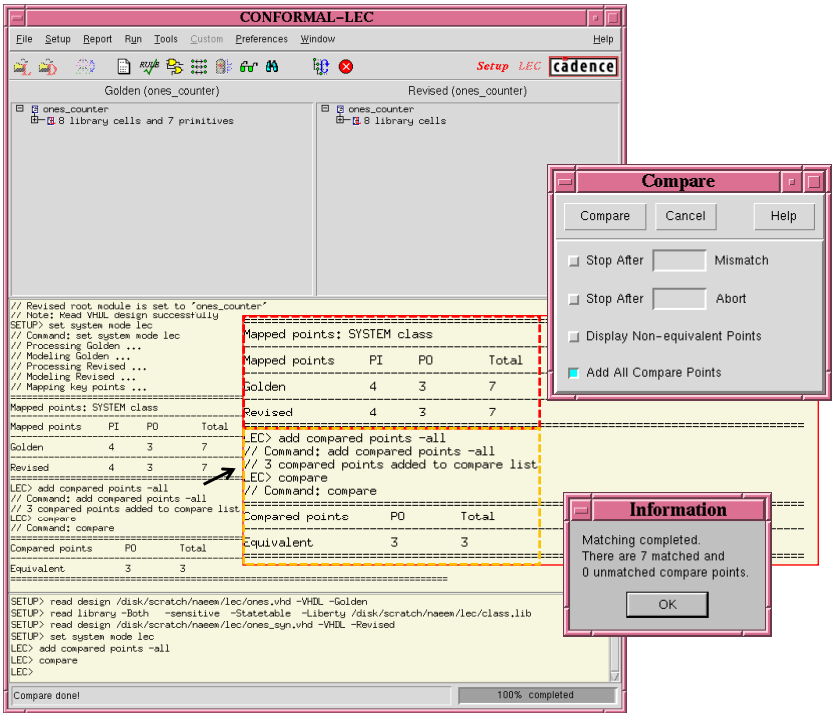

Fig. 20: Conformal map points and equivalence results

now reduces the two designs into canonical representations and then checks to see if they are equal. In the example shown in Fig. 20, all 3 outputs are equivalent. Or in other words, the RTL and the synthesized designs are functionally equivalent.

4.6 Reporting Results

4.6.1 Schematic and Source Views

This feature is very helpful in debugging the design. In the "Golden" or "Revised" column of the main conformal LEC window, right click on a file name or a cell name and then select either "Schematic" or "Source" as desired.

Lets take the same example that we used in the previous section and debug it with Conformal LEC. You will require the following files in this example.

1. The golden design: `ones_syn.vhd`
2. The revised design: `ones_syn1.vhd`
3. The primitive cells library: class.lib

First load the golden (`ones_syn.vhd`) and revised (`ones_syn1.vhd`) designs as described in the earlier sections. Then run verification as before. The verification will be partially

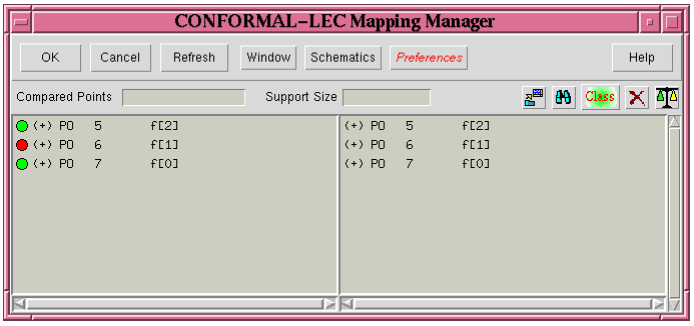

Fig. 21: Conformal mapping manager window

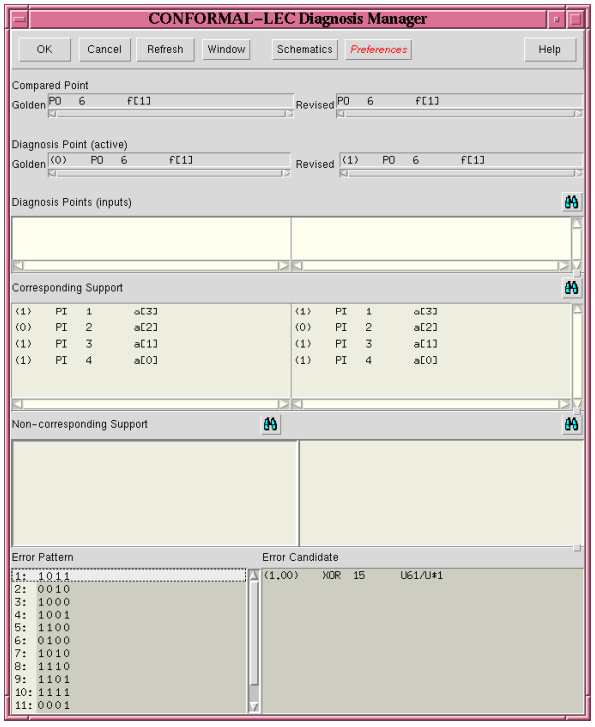

Fig. 22: Conformal diagnosis manager

successful. To see the details of the equivalence checking results, click Report ⇒ Compared Points. Now we see that there is one non equivalent point (red circle, See Fig. 21). Select the unequal compare points, right click and then select "Diagnosis". In the Diagnosis Manager window (Fig. 22) click on the "Refresh" button. Then in the "Schematic" pull down menu, select the "Show Inverted Buffer Inverter Gate" and "Show Schematic Full Fanin Core" options. Then open the golden and diagnosed revised designs (Schematic ⇒ open). You will now see the golden and revised schematics (See Fig. 23). The problem gates and nets will be highlighted in red. Select the problem gate (the XOR gate) in the revised design schematic window, right click and then select "source code". Fix the source code in the revised design. Save the revised design, and re run the verification.

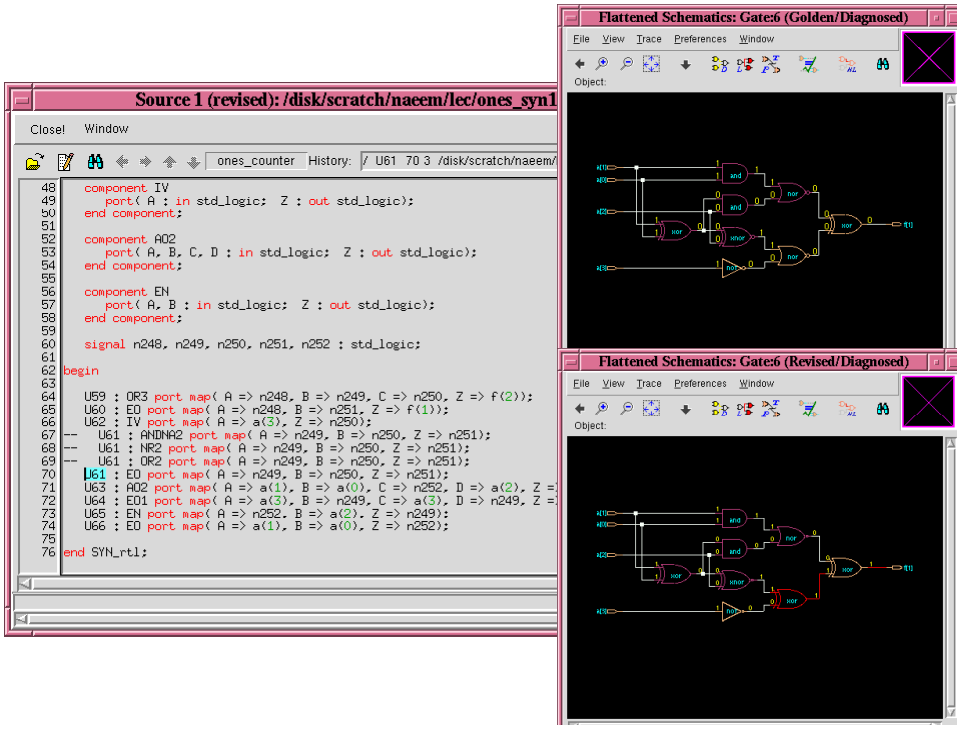

Fig. 23: Conformal schematic and source view

4.7 How to create a command script

1. The DO file script can be generated from the LEC GUI by:
   File ⇒ Save dofile

2. The DO file script can be run from the LEC GUI by:
   File ⇒ Do dofile

3. The DO scripts can also be run directly from the unix prompt by:
   `lec -nogui -do lec.do`

In this tutorial, we learned the basic steps involved in the digital logical synthesis. We also learned how to use two industry standard equivalence checking tools, namely, the Synopsys Formality and the Cadence Conformal Logical Equivalence Checker or LEC.

References

[1] M. Moris Mano, C. R. Kime, *Logic and Computer Design Funcdamentals*, Second Edition, ISBN 0-12-012468-0

[2] D. R. Perry, *VHDL*, McGraw-Hill series in Computer Engineering, Second Edition, ISBN 0-07-049434-7, TK 7885.7.P47 1993

A Synopsys Design Analyzer

A.1 Design Analyzer Setup Files

```
/* This is the .synopsys_dc.setup file required              */
/* for use with the class.db  synthesis technology library   */
/* This library is used by Synopsys for synthesis training   */
/* courses.                                                  */
/*                                                           */
/* Ted Obuchowicz                                            */
/* June 11, 2003                                             */


search_path =  {. /CMC/tools/synopsys/syn/libraries/syn } ;

/*  assumes that the link /CMC/tools/synopsys/syn exists and points */
/*  to the most recent Synopsys synthesis install                   */


target_library = class.db
link_library = class.db
symbol_library = class.sdb


compile_fix_multiple_port_nets = true

/* the bus naming syle determines how one refers to elements of */
/* a bus in a dc_shell script  such as count<2>  if "%s<%d>" is the style */
/* or count_2 if "%s_%d" is used as the style */
/* the <%d>  causes SDF problems since the SDF file refers to buses as count_reg_0 */
/* and does not use <> */

bus_naming_style = "%s<%d>"

/* if you want to back annotate Synopsys  SDF files onto Synopsys gate-level netlists */
/* use the %s_%d bus naming style... but make sure that in your script */
/* you refer to elements of buses as count_2, count_1 , etc !!!   */
/* For Xilinx generated netlists and SDF files there is no problem */
/* can use the standard %s<%d> style     */

/* bus_naming_style = "%s_%d" */


bus_inference_style = "%s<%d>"
bus_dimension_separator_style = "><"

/* to write out EDI netlists, this must be set to "true" */

edifout_netlist_only = "true"
```

The "class.db" library file used in this tutorial is located in:
[/CMC/tools/synopsys/syn/libraries/syn/class.db]

A.2 Design example files

"ones.vhd" can be found here [http://users.encs.concordia.ca/∼tahar/coen7501/ones.vhd]

A.3 Documentation

The Synopsys tool documentation is here: [/CMC/tools/synopsys/syn/doc/online/top.pdf]

B Synopsys Formality

B.1 The reference and Implementation designs

"ones_syn.vhd" and "ones_syn1.vhd" are here:
[http://users.encs.concordia.ca/∼tahar/coen7501/ones_syn.vhd], and
[http://users.encs.concordia.ca/∼tahar/coen7501/ones_syn1.vhd]

B.2 Other useful information

The Synopsys tool documentation is here: [/CMC/tools/synopsys/syn/doc/online/top.pdf]
Some of the Formality user and reference documents are here: [/CMC/tools/synopsys/fm/doc/fm]
If you are interested in reading more about the tool, a good starting point is the user.pdf document.

One can also find many good tool tutorials on the web. For example,
see [http://www.chiptalk.org]

C Cadence Conformal

C.1 The golden and revised design files

"ones_syn.vhd" and "ones_syn1.vhd" are here:
[http://users.encs.concordia.ca/∼tahar/coen7501/ones_syn.vhd], and
[http://users.encs.concordia.ca/∼tahar/coen7501/ones_syn1.vhd]
The "class.db" and "class.lib" library files used in this tutorial are located in:
[/CMC/tools/synopsys/syn/libraries/syn/] directory.

C.2 The command script

```
//************************************************
// design input
//************************************************
Read Design -golden  ones_syn.vhd
Read Library -both class.lib
Read Design -revised ones_syn1.vhd

report design data -both

//************************************************
// design preparation and key points mapping
//************************************************
set system mode lec

//************************************************
// compare mapped key points
//************************************************
add compare point -all
compare

//************************************************
// reporting result
//************************************************
report compare data -summary
```

C.3 Remote access

Following information was very kindly provided by our CAD tool GURU (Ted Obuchowicz).
If you are connecting remotely using 'ssh' take note of the following:

1. On a solaris host use : /usr/bin/ssh to connect to a solaris 10 remote host
2. On a solaris host use : /common/bin/ssh -X to connect to a solaris 10 remote host
3. On a Linux host use : /usr/bin/ssh -X to connect to a solaris 10 remote host

C.4 Commonly used UNIX commands

A good unix tutorial is here: [http://www.ee.surrey.ac.uk/Teaching/Unix/]
Have you ever used these UNIX commands?

Description	UNIX command
Memory use	top
Process list	ps
Kill process	kill -9 pid
Workstation information	prtdiag

where pid is the process identification number.

C.5 Other useful information

User and reference documents are in [/CMC/tools/cadence.2007a/CONFRML/doc/]

Complete documentation is here [/CMC/tools/cadence/CONFRML/doc]

If you are interested in reading more about the tool, a good starting point is the Conformal_User.pdf document.

Currently we have Version 6.1 of conformal installed on our workstations.

One can also find many good tool tutorials on the web. For example,
see [http://www.chiptalk.org]