Lab 1 - Logic Equivalence Check

1. Objective

LEC is an important part of the front-end design verification flow because Synthesis tools often optimize the design to meet constraints, and might change the logic inadvertently during this process. The goal of this lab is to check the logical equivalence between RTL and corresponding gatenetlist. There are two parts to this lab. The first part consists of verifying the logical equivalence and fixing bugs in the write back stage of the ARM Amber Core. In the second part, there is an SoC with amber core along with clock gating and scan inserted netlist of the design to be verified for logical equivalence (Does not have bugs).

2. Tool Tutorial

Please refer to the Conformal LEC tutorial from Canvas. If you need more information about the tool, refer to the **help** in the tool.

3. Files

All necessary files except the library files for part B are in compressed file 'lab1.tar.gz'.

The following are its contents.

i. Part A:

1. **partA/lib/gscl45nm.lib** is the library used for synthesizing the RTL of the write back module.
2. **partA/rtl/a25_write_back.v** is the RTL description to be used as golden model for the write back module.
3. **partA/gate/a25_write_back_netlist.v** is the netlist to be used as the revised model for the write back module.
4. **partA/lec/lec.do** is the lec dofile to be completed.

ii. Part B:

1. **partB/scripts/common/rtl.list** is the list of RTL files (golden model). This does not include the file names called-in an RTL file with **`include** Verilog directive.
2. **partB/scripts/lec/lec.do** is the lec dofile to be completed.
3. **partB/scripts/rc/system_cg_scan_netlist.v** is the netlist to be used as the revised model.
4. **partB/scripts/rc/system_dft_setup** is the file which provides the details of scan insertion. This will help you identify the scan signals and the details of the scan chain structure. This information is useful to set the pin constraints in the dofile.
5. **partB/scripts/rc/system_scandef** is the industry standard format which provides the scan chain details. This file is redundant; however, this will help you understand the underlying

scan chaining information.

6. **partB/verilog/*** includes the folders containing the RTL files of the Amber core SoC.

The SoC RTL in the part B is synthesized using multiple libraries enlisted below:
/usr/local/packages/synopsys_2015/SAED32_EDK/lib/stdcell_hvt/db_nldm/saed32hvt_ss0p95vn40c.lib
/usr/local/packages/synopsys_2015/SAED32_EDK/lib/stdcell_rvt/db_nldm/saed32rvt_ss0p95vn40c.lib
/usr/local/packages/synopsys_2015/SAED32_EDK/lib/stdcell_lvt/db_nldm/saed32lvt_ss0p95vn40c.lib

4. Tasks

Part A

1. Using the Conformal LEC tool, perform equivalence checking on the RTL a25_write_back.v (as the reference model) and the gate level netlist a25_write_back_netlist.v (as the implementation model).
2. Note that the provided tutorial gives an introduction into this part. It is strongly suggested to follow the tutorial before you start fixing the bugs.
3. If your verification process fails, do the diagnosis and find the mismatch(es). Describe the bugs you find and fix them in the netlist a25_write_back_netlist.v and run the LEC again to make sure that the problems are resolved.
4. Please note that, for partA your verification results should not contain unmapped points. If you observe such points in your verification results, you need to fix them. You can get help from the User Guide or using help and man commands in the Conformal LEC tool.
5. When you are debugging your design, start from the points (mismatching points) that have some logic cones. Some of the mismatching points do not show any logic to compare and they may not be helpful.

Part B

1. In this part, RTL of an SoC with an Amber core is given to you along with its synthesized netlist. The synthesis tool has performed the scan insertion (for scan testing) and clock gating (to reduce dynamic power) which are not present in the RTL (golden model) as evident from the netlist (revised model).
2. Unlike part A, there are no bugs in the netlist. Your task is to just complete the lec dofile.
3. Since the scan chain and clock gating was not present in the RTL, add suitable pin constraints to the clock gate enable and scan pins to disable the added scan logic. Adding a pin constraint may be as simple as telling LEC to hold a constant value on an enable pin while checking for equivalence.
4. Large memory arrays and analog modules are black boxed while doing LEC because they are usually full custom designs already verified using other techniques. There are 17 RAMs in the SoC which do not have the RTL/library definition. You must set these undefined cells as black boxes in the SETUP mode before reading the golden design. Also, you might have to map some of them manually to the corresponding RAMs the golden model while in LEC mode (before compare) if the tool classifies them as unmapped points.
5. Please note that, for part B your verification results may contain unmapped mapped due to several reasons such as extra pins added for scan chain creation, inserted clock gates, elements removed due to logic optimization during synthesis etc.

5. General Guidelines

1. The golden model or RTL files must not be modified for both Part A & B.
2. The folder hierarchy must not be changed. The location of the lec.do files must remain intact for both Part A & B. That is, for part A, the lec.do dofile must be executed in the folder **partA/lec**. And for part B, the lec.do dofile must be executed in the folder **partB/scripts/lec.**
3. For part B, try to read-in the **partB/scripts/common/rtl.list** file in the lec.do file using the command *source ../common/rtl.list*. This will create a tcl variable of the type list with name $RTL_LIST. If you are not comfortable with the file format you can modify this file 'rtl.list' and it must be included in the submission.
4. In conformal LEC, the commands can be in 'tclmode' or 'vpxmode' mode. The only difference is that, in 'tclmode', the words in a command are concatenated to form a single string as shown in the example below:
    - tclmode: set_system_mode lec
    - tpxmode: set system mode lec

   The lec.do file in both part A & B must only be in 'tclmode'.

6. Lab Deliverables

Report: 10%

A pdf file with the name format LastName_FirstName_lab1.pdf with the following contents:

- Part A:
    1. A brief report describing the netlist bugs and your solution.
    2. Final snapshot of the mapping manager and lec console.
- Part B:
    1. Brief description of the scan structure of the SoC.
    2. The schematic of a clock gating cell in the netlist. (Search for the module with name 'RC_CG_MOD' in the netlist.
    3. What is the significance of the following commands in the lec.do file?

       set_flatten_model -seq_constant -seq_constant_x_to 0
       set_flatten_model -gated_clock
       set_analyze_option -auto
    4. Final snapshot of the mapping manager and lec console.

Lab 1 submission: 90%

A zipped file with the name format LastName_FirstName_lab1.zip with the following contents:

- Part A: 30%
    1. The corrected Verilog netlist: partA/a25_write_back_corrected_netlist.v.
    2. The completed dofile: partA/lec.do file.
- Part B: 60%
    1. The completed dofile: partB/lec.do file.
    2. The 'rtl.list', if modified: partB/rtl.list.

7. Bonus Policy

1. If you finish and submit your work earlier than the deadline, you obtain bonus in grade for early submission: +5% per day, Maximum +10%

2. If you submit your work later than the deadline, there will be a penalty in grade for late submission: -5% per day, Maximum -25% (Zero credit after the maximum penalty submission)

8. Appendix: List of macros in SoC (Part B)

List of macros in RTL (Golden model) useful for manual mapping of compare points.

u_boot_mem_wrapper/u_boot_mem/myram7
u_boot_mem_wrapper/u_boot_mem/myram6
u_boot_mem_wrapper/u_boot_mem/myram5
u_boot_mem_wrapper/u_boot_mem/myram4
u_boot_mem_wrapper/u_boot_mem/myram3
u_boot_mem_wrapper/u_boot_mem/myram2
u_boot_mem_wrapper/u_boot_mem/myram1
u_boot_mem_wrapper/u_boot_mem/myram0
u_eth_top/wishbone/bd_ram/myram
u_amber/u_mem/u_dcache/rams[3].myram
u_amber/u_mem/u_dcache/rams[2].myram
u_amber/u_mem/u_dcache/rams[1].myram
u_amber/u_mem/u_dcache/rams[0].myram
u_amber/u_fetch/u_cache/rams[3].myram
u_amber/u_fetch/u_cache/rams[2].myram
u_amber/u_fetch/u_cache/rams[1].myram
u_amber/u_fetch/u_cache/rams[0].myram

List of corresponding macros in netlist (Revised model)

u_boot_mem_wrapper_u_boot_mem/myram7
u_boot_mem_wrapper_u_boot_mem/myram6
u_boot_mem_wrapper_u_boot_mem/myram5
u_boot_mem_wrapper_u_boot_mem/myram4
u_boot_mem_wrapper_u_boot_mem/myram3
u_boot_mem_wrapper_u_boot_mem/myram2
u_boot_mem_wrapper_u_boot_mem/myram1
u_boot_mem_wrapper_u_boot_mem/myram0
u_eth_top/wishbone/bd_ram/myram
u_amber/u_mem/u_dcache/rams[3].myram
u_amber/u_mem/u_dcache/rams[2].myram
u_amber/u_mem/u_dcache/rams[1].myram
u_amber/u_mem/u_dcache/rams[0].myram
u_amber/u_fetch/u_cache/rams[3].myram
u_amber/u_fetch/u_cache/rams[2].myram
u_amber/u_fetch/u_cache/rams[1].myram
u_amber/u_fetch/u_cache/rams[0].myram