数字部分 大作业

0. 数字实验概述

数字实验总体分为三个部分：1.代码仿真。2.逻辑综合。3.物理设计。

(1) Linux 系统命令

常见的 Linux 命令：

```
$ cd~
```

更改当前工作目录为根目录，根目录名称一般是自己的学号。

```
$ pwd
```

查看当前目录。

```
$ mkdir vlsi
```

建立名为 vlsi 的文件夹。

```
$ cd vlsi
```

更改当前工作目录为 vlsi，可以理解为进入 vlsi 文件夹。

```
$ cp ~eda/course/219004/RTLtoGDSII_3_0.tar.gz .
```

复制文件至当前工作目录。

```
$ cd ..
```

返回上一层工作目录

```
$ tar xvfz RTLtoGDSII_3_0.tar.gz
```

解压文件

```
$ ls
```

列出当前工作目录下的文件

```
$ gedit counter.v
```

使用文本编辑器打开文件，小技巧：如果想在编辑文本的同时进行命令行操作，可以使用下面的命令：

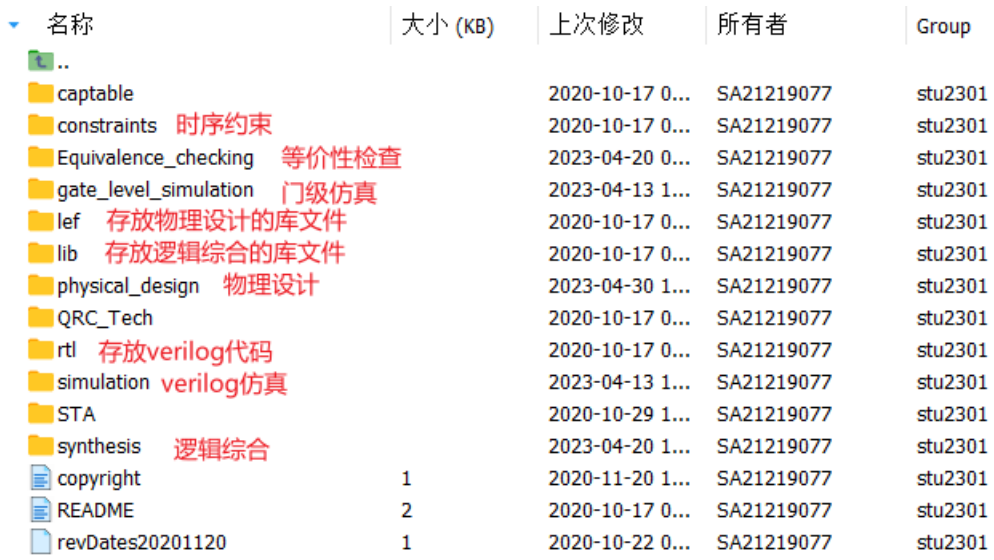
```
$ gedit counter.v &
```

这时文本编辑器会在后台运行，不影响命令行的输入。

(2) 实验文件与目录结构

在实验一中，建立了一个实验目录 **counter_design_database_45nm**，实验一、二、三都是在此目录下对应的目录中进行的。工作目录是否正确很重要，关系到执行脚本时能否读取到正确的设计文件。如果使用 MobaXterm 软件进行实验，可以直接在窗口左侧看到文件目录结构。

进入 **vlsi** 文件夹，再进入 **counter_design_database_45nm** 文件夹。



不同文件夹有不同作用，在实验一的课件中也有提到。

工作目录非常重要，关系到运行程序或脚本时能否找到相应的文件。

有关 Linux 文件与目录管理的更多知识，可以参考 Linux 文件与目录管理 | 菜鸟教程

1. 实验一 代码仿真

(1) 代码仿真在文件夹 **simulation** 中进行。

4.3 中的测试平台：

```
module counter_test;

  localparam WIDTH=5;

  reg  clk  ;
  reg  rst  ;
  reg  load ;
  reg  enab ;
  reg  [WIDTH-1:0] cnt_in;
  wire [WIDTH-1:0] cnt_out;

  counter
  #(
    .WIDTH ( WIDTH )
   )
  counter_inst
   (
    .clk      ( clk     ),
    .rst      ( rst     ),
    .load     ( load    ),
    .enab     ( enab    ),
    .cnt_in   ( cnt_in  ),
    .cnt_out  ( cnt_out )
   );

  task expect;
    input [WIDTH-1:0] exp_out;
    if (cnt_out !== exp_out) begin
      $display("TEST FAILED");
      $display("At time rst=%b load=%b enab=%b cnt_in=%b cnt_out=%b",
                $time, rst, load, enab, cnt_in, cnt_out);
      $display("cnt_out should be %b", exp_out);
      $finish;
    end
    else begin
      $display("At time rst=%b load=%b enab=%b cnt_in=%b cnt_out=%b",
                $time, rst, load, enab, cnt_in, cnt_out);
    end
  endtask

  initial repeat (7) begin #5 clk=1; #5 clk=0; end

  initial @(negedge clk) begin
    rst=0; load=1; enab=1; cnt_in=5'h15; @(negedge clk) expect (5'h15);
    rst=0; load=1; enab=1; cnt_in=5'h0A; @(negedge clk) expect (5'h0A);
    rst=0; load=1; enab=1; cnt_in=5'h1F; @(negedge clk) expect (5'h1F);
    rst=1; load=1; enab=1; cnt_in=5'h1F; @(negedge clk) expect (5'h00);
    rst=0; load=1; enab=1; cnt_in=5'h1F; @(negedge clk) expect (5'h1F);
    rst=0; load=0; enab=1; cnt_in=5'h1F; @(negedge clk) expect (5'h00);
    $display("TEST PASSED");
    $finish;
  end

endmodule

```

此测试（testbench）除了给定了 counter 模块的输入输出，还设置了 task，用于把 counter 的输出和某一个给定的值进行比较，task 的输出会在批处理之后在 Shell 中显示。如果合理使用 task，可以在批处理时判断电路模块的工作情况。大作业的 testbench 可以借鉴这个文件的相关内容。有关 task 的使用方法可以参考 6.1 Verilog 函数 | 菜鸟教程 。

(2) 批处理和图形模式：

批处理的结果包括代码的一些基本信息，包括编译和建模的结果，也可以在测试平台（testbench）中使用 $display 等函数，将运行过程中的数据在命令行中输出（类似于 c 语言中的 printf 函数）。图形模式是将程序中的变量（wire，reg 等）通过波形的形式进行观察，方便时序的调试。

大作业的实验报告需要包含波形以及对比运算结果的部分。

2. 实验二 逻辑综合

逻辑综合在 **synthesis** 文件夹中进行。

逻辑综合的目的是将已有的 rtl 代码综合为门级单元的连接，生成对应的网表（netlist）文件，同时根据给定的时序约束生成 SDC 文件，网表以及时序约束文件在后续的物理设计中也会用到。

（1）时序约束

在 **constraints** 目录中存在一个时序约束文件，每条命令作用在实验课件中有解释，时序约束文件中需要对设计的输入输出端口，以及时钟信号 clk 进行约束：

```
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clk"]
set_clock_transition -rise 0.1 [get_clocks "clk"]
set_clock_transition -fall 0.1 [get_clocks "clk"]
set_clock_uncertainty 0.01 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "rst"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "count"] -clock [get_clocks "clk"]
```

rst和count是计数器的输入和输出
大作业中需要进行更改

如果在逻辑综合之后的 **report_timing** 结果中时序不符合要求，可以放宽时序约束的条件。

※：为了解决物理设计中可能出现的扫描链问题，需要在 SDC 文件中加入：

```
set_dont_use S*
```

（2）逻辑综合脚本

建议使用逻辑综合脚本进行实验。

实验数据中给出了两个实验脚本，这里对 **genus_script.tcl** 脚本中的部分命令进行解释说明：

```
set_db init_lib_search_path ../lib/
set_db init_hdl_search_path ../rtl/   设置代码路径，之后读入rtl代码时，程序会到../rtl路径下寻找
read_libs slow_vdd1v0_basicCells.lib

read_hdl counter.v   读入rtl代码，此时程序会在../rtl路径下寻找名为counter.v的文件
elaborate
read_sdc ../constraints/constraints_top.sdc  读入时序约束文件

set_db syn_generic_effort medium
set_db syn_map_effort medium
set_db syn_opt_effort medium

syn_generic
syn_map
syn_opt

write_hdl > counter_netlist.v  在当前路径下生成网表文件
write_sdc > counter_sdc.sdc  在当前路径下生成时序约束文件
write_sdf -timescale ns -nonegchecks -recrem split -edges check_edge  -setuphold split > delays.sdf
```

以上标注的命令在完成大作业的设计时需要做相应的更改（文件名）。

※：注意，逻辑综合时的参数 N 应该为 16 。

在逻辑综合时，需要注意 shell 中的 Warning 信息，代码中可能会出现不可综合的结构，或者由于代码不规范导致的警告信息，应尽量避免。

网表文件（netlist）和时序约束文件（SDC）都是在当前路径下生成的，在进行实验三物理设计时需要注意文件位置。脚本中没有包含以下三行命令，需要在脚本运行结束后手动输入。

```
@genus> report_timing
@genus> report_power
@genus> report_qor
```

(3) 等价性检查

等价性检查在路径 **Equivalence_checking** 中完成，脚本 **counter.do** 的内容如下：

```
set log file counter_lec.log -replace
read library ../lib/slow_vdd1v0_basiccells.v -verilog -both
read design ../rtl/counter.v -verilog -golden
read design ../synthesis/counter_netlist_dft.v -verilog -revised
add pin constraints 0 SE  -revised
add ignored inputs scan_in -revised
add ignored outputs scan_out -revised
set system mode lec
add compare point -all
compare
```

等价性检查对比rtl文件夹下counter.v文件和synthesis文件夹下counter_netlist_dft.v完成大作业时需要修改为正确的文件路径和文件名

脚本中最后是compare命令没有输出报告，可以手动输出报告

这三条命令设计扫描链大作业不需要完成，可以注释掉

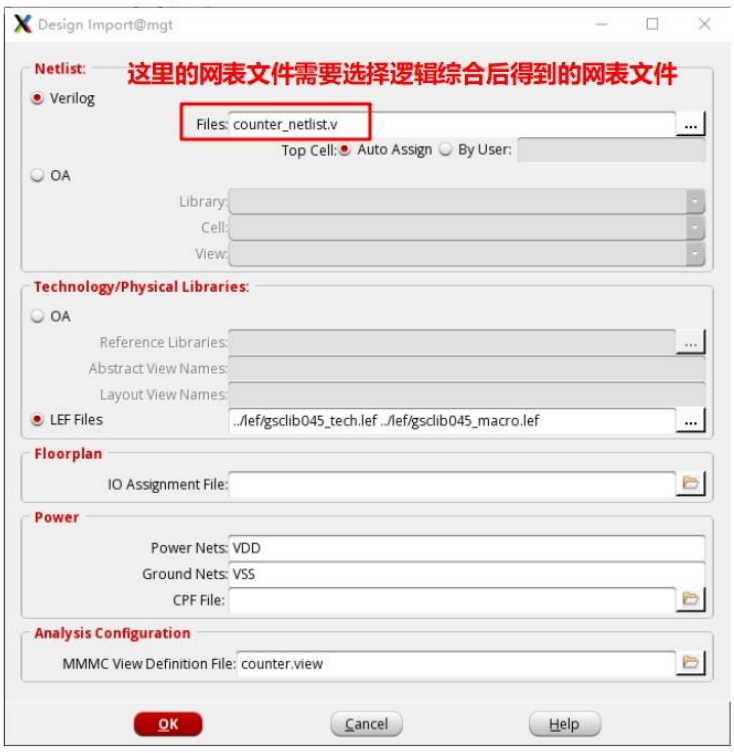
最后可以使用 **report verification** 命令输出验证报告。

3. 实验三 物理设计

1. 导入设计，涉及到门级网表（netlist）文件，工艺单元库文件，MMMC View Definition File，其中门级网表文件在逻辑综合中生成。MMMC view 文件中需要定义设计的时序约束，如果在后续时序分析中出现了过多的违例路径，则可能是 MMMC view 文件中没有定义正确的时序约束（在.view 文件里）：

```
create_constraint_mode -name sdc_cons\
  -sdc_files\
   counter_sdc.sdc
```

上面的命令定义了时序约束文件，在完成大作业时需要更改为对应的 SDC 文件，即逻辑综合生成的 SDC 文件。



2. 后续按照布局，电源规划，时钟树，布线，时序分析的顺序进行操作即可，注意不要选错金属层。有关扫描链的两条命令需要跳过。
3. 版图的面积信息，用标尺测量正方形边长即可。
4. 时序分析的结果，最好在 timing debug 窗口中用柱状图展示
5. 物理验证结果包括 DRC 检查和连接性检查。
6. 完成物理设计后记得保存设计文件：

```
@innovus> write_db signOff
```