

数字部分 实验二 逻辑综合与等价性检查

一、环境配置

1.1 登录方法

利用 Xmanager 软件登录服务器, 协议选择 SSH, IP 地址为 202.38.81.119, 端口 2122, 登录管理节点, 然后利用 SSH 登录运算节点 c01n01 至 c01n14, 登录时请注意避开用户较多的节点。例如, 登录 c01n10 节点:

```
$ ssh -X c01n10
```

查看各节点的负载情况, 可以浏览如下网址:

<http://202.38.81.119/ganglia/>

1.2 进入实验目录

```
$ cd ~/vlsi
```

本实验继续使用实验一的目录, 若实验一中使用了其他目录名, 请做相应的修改。

二、逻辑综合基本流程

2.1 设置软件环境

设置 Cadence Genus 181 软件环境:

```
$ setdt genus
```

注意: 上述命令中的 `setdt` 是实验中心自定义的脚本, 不是通用命令, 作用是设置软件所需的路径、环境变量等。在其他服务器运行软件时, 请咨询管理员或 CAD 支持人员。

2.2 进入逻辑综合运行目录

```
$ cd counter_design_database_45nm/synthesis
```

注意: 请勿在指定目录之外的位置执行实验操作, 以免文件相对路径错误, 或实验产生的文件相互混在一起。

2.3 定义时序约束

使用时序约束文件(SDC), 可以定义时钟的周期、脉冲宽度、上升和下降时间、不确定度以及各种信号的输入/输出延时。

实验数据中提供了 SDC 文件: `../constraints/constraints_top.sdc`, 请使用文本编辑器打开并认真阅读, 其中的命令定义了最基本的时序约束。以下对每条命令加以简要解释。

```
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clk"]
```

上述命令在端口 `clk` 上定义了一个名为 `clk` 的时钟, 周期 10 ns, 在每周期的 0 ns 处上升, 5 ns 处下降, 即占空比 50%, 前半周期为高电平。

```
set_clock_transition -rise 0.1 [get_clocks "clk"]
```

```
set_clock_transition -fall 0.1 [get_clocks "clk"]
```

上述两条命令分别定义了时钟 `clk` 的上升和下降时间皆为 0.1 ns。

```
set_clock_uncertainty 0.01 [get_ports "clk"]
```

上述命令定义了时钟 `clk` 的不确定度范围为 0.01 ns，这是由时钟抖动(Jitter)和时钟偏斜(Skew)造成的。

```
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"]
```

上述两条命令分别定义了输入/输出信号的输入/输出延时皆为 1.0 ns，将用于剩余时间(Timing Slack)的计算。

2.4 使用命令输入运行逻辑综合

2.4.1 启动软件

启动 Genus 软件，并进入 Genus 的命令状态：

```
$ genus
```

2.4.2 执行逻辑综合

以下每步操作都会产生或多或少的运行信息，请注意观察，这些信息提示了目前的运行状况，并有助于调试可能遇到的问题。

设置单元库的搜索路径：

```
@genus> set_db init_lib_search_path ../lib/
```

设置 HDL/RTL 代码的搜索路径：

```
@genus> set_db init_hdl_search_path ../rtl/
```

载入单元库：

```
@genus> read_libs slow_vdd1v0_basicCells.lib
```

读入设计：

```
@genus> read_hdl counter.v
```

设计若包括多个 HDL 文件，需使用花括号把多个文件括起来。

Elaborate 设计，构建设计层次，连接内部信号：

```
@genus> elaborate
```

读入时序约束：

```
@genus> read_sdc ../constraints/constraints_top.sdc
```

设置综合各步骤的努力量(Effort)，并把设计综合成为通用(Generic)门单元、映射到标准单元库、优化设计：

```
@genus> set_db syn_generic_effort medium
```

```
@genus> set_db syn_map_effort medium
```

```
@genus> set_db syn_opt_effort medium
```

```
@genus> syn_generic
```

```
@genus> syn_map
```


```
@genus> syn_opt
```

浏览上述命令在终端输出的信息，了解这些信息包含的内容。

2.4.3 使用图形界面(GUI)

打开/显示图形界面：

```
@genus> gui_show
```

点击“加号”按钮打开原理图(Schematic)，浏览综合得到的电路结构。打开其他视图，浏览感兴趣的信息。

关闭/隐藏图形界面：

```
@genus> gui_hide
```

2.4.4 生成设计报告

使用 `report_*` 形式的一系列命令生成设计数据报告。

生成时序报告：

```
@genus> report_timing
```

生成功耗报告：

```
@genus> report_power
```

生成结果质量(QOR)报告：

```
@genus> report_qor
```

将报告输出的主要数据记录在思考题表格中，详见思考题部分。

2.4.5 输出设计数据

综合完成后，把设计输出为网表文件、延时文件(SDF)、时序约束文件(SDC)，用于后面的设计流程。

输出网表文件：

```
@genus> write_hdl > counter_netlist.v
```

生成 SDC 文件：

```
@genus> write_sdc > counter_sdc.sdc
```

输出 SDF 文件：

```
@genus> write_sdf -timescale ns -nonegchecks -recrem split -  
edges check_edge -setuphold split > delays.sdf
```

打开并浏览上述文件，了解设计数据的内容。

2.4.6 退出软件

```
@genus> exit
```

2.5 使用脚本运行逻辑综合

2.5.2 脚本文件

在实际设计工作中，需要反复多次运行仿真、综合、物理设计等软件，因此，通常把这些操作和命令序列保存在脚本文件中，每次运行时，只需要调用脚本文件，即可完成一系列操作，提高工作效率。

当前目录下有两个 Tcl 语言脚本文件。

`genus_script.tcl` 逻辑综合脚本文件

`genus_dft_script.tcl` 逻辑综合及可测性设计脚本文件

打开文件 `genus_script.tcl`，可以发现其内容就是 2.4 节中输入的命令序列。

脚本文件的使用主要有两种方式，分别以下两节介绍。

2.5.3 启动软件并执行脚本

启动 Genus 软件，并自动运行脚本文件中的命令：

```
$ genus -f genus_script.tcl
```

运行结束后，在 Genus 命令状态等待进一步的命令输入。

2.5.4 启动软件后执行脚本

启动 Genus 软件，并进入 Genus 的命令状态：

```
$ genus
```

执行脚本文件：

```
@genus> source genus_script.tcl
```

运行结束后，在 Genus 命令状态等待进一步的命令输入。

三、带有可测性设计的逻辑综合流程（选做）

3.1 流程简介

在基本逻辑综合流程基础上，增加了可测性设计(DFT)流程、增量综合、输出结果和数据、输出自动测试模式生成(ATPG)流程所需文件。

3.2 启动软件

启动 Genus 软件，并进入 Genus 的命令状态：

```
$ genus
```

3.3 逻辑综合与扫描插入

执行逻辑综合的基本设置：

```
@genus> set_db init_lib_search_path ../lib/
```

```
@genus> set_db init_hdl_search_path ../rtl/
```

```
@genus> read_libs slow_vddl1v0_basicCells.lib
```

```
@genus> read_hdl counter.v
```

```
@genus> elaborate
```

```
@genus> read_sdc ../constraints/constraints_top.sdc
```

设置扫描触发器的类型，用于取代非扫描触发器：

```
@genus> set_db dft_scan_style muxed_scan
```

设置插入的 DFT 逻辑单元名称前缀，以便在电路和网表中加以区分：

```
@genus> set_db dft_prefix dft_
```

定义测试使能控制信号：

```
@genus> define_shift_enable -name SE -active high -create_port SE
```

检查 DFT 规则：

```
@genus> check_dft_rules
```

建议在设计过程中多次检查 DFT 规则是否存在违例。

执行逻辑综合步骤：

```
@genus> set_db syn_generic_effort medium
```

```
@genus> syn_generic
```

```
@genus> set_db syn_map_effort medium
```

```
@genus> syn_map
```

```
@genus> set_db syn_opt_effort medium
```

```
@genus> syn_opt
```

```
@genus> check_dft_rules
```

定义扫描链的数量，此处为 1 条：

```
@genus> set_db design:counter .dft_min_number_of_scan_chains 1
```

定义扫描链的扫描输入和扫描输出端口：

```
@genus> define_scan_chain -name top_chain -sdi scan_in -sdo  
scan_out -create_ports
```

连接扫描链，将所有扫描触发器连接起来：

```
@genus> connect_scan_chains -auto_create_chains
```

执行增量综合：

```
@genus> syn_opt -incr
```

输出扫描链报告：

```
@genus> report_scan_chains
```

输出网表文件、延时文件(SDF)、时序约束文件、扫描设计文件(DEF)：

```
@genus> write_hdl > counter_netlist_dft.v
```

```
@genus> write_sdf -nonegchecks -edges check_edge -timescale ns  
-recrem split -setuphold split
```

```
@genus> write_sdc > counter_sdc_dft.sdc
```

```
@genus> write_scandef > counter_scanDEF.scandef
```

生成 ATPG 流程所需文件，输出文件存放在 `test_scripts` 目录中：

```
@genus> write_dft_atpg -library ../lib/slow_vdd1v0_basiccells.v
```

启动 Genus 图形界面，浏览扫描插入后的原理图，观察扫描链的结构。

退出软件。

上述操作过程也可以使用脚本文件 `genus_dft_script.tcl` 批处理执行。

四、逻辑等价性检查

4.1 设置软件环境

设置 Cadence Conformal 171 软件环境：

```
$ setdt cfm
```

注意：上述命令中的 `setdt` 是实验中心自定义的脚本，不是通用命令，作用是设置软件所需的路径、环境变量等。在其他服务器运行软件时，请咨询管理员或 CAD 支持人员。

4.2 进入等价性检查运行目录

```
$ cd ~/vlsi
```

```
$ cd counter_design_database_45nm/Equivalence_checking
```

注意：请勿在指定目录之外的位置执行实验操作，以免文件相对路径错误，或实验产生的文件相互混在一起。

注意：本实验需要先完成第二节逻辑综合基本流程实验。

4.3 启动软件

启动 Conformal 软件，并进入 Conformal 的 SETUP 命令状态：

```
$ lec -XL -nogui -color -64
```

4.4 使用命令方式执行等价性检查

设置日志文件，若同名文件已存在，则覆盖取代：

```
SETUP> set log file counter_lec.log -replace
```

读入 Verilog 格式单元库，同时用于参考(Golden)设计和修订(Revised)设计：

```
SETUP> read library ../lib/slow_vdd1v0_basiccells.v -verilog -both
```

读入 RTL 代码作为参考设计，即检查比对的标准：

```
SETUP> read design ../rtl/counter.v -verilog -golden
```

读入综合网表作为修订设计，即需要检查的对象：

```
SETUP> read design ../synthesis/counter_netlist.v -verilog -revised
```

操作模式由设置模式(SETUP)切换为检查模式(LEC)：

```
SETUP> set system mode lec
```

添加比较点，并比较参考设计和修订设计：

```
LEC> add compare point -all
```

```
LEC> compare
```

观察运行结果，可能有 Equivalent, Non-equivalent, Aborted, Not Compared 等情况。

生成验证报告：

```
LEC> report verification
```

将上述输出的主要数据记录在思考题表格中，详见思考题部分。

打开图形界面：

```
LEC> set gui on
```

自行探索图形界面的功能，完成后关闭软件：

```
LEC> exit
```

4.5 使用脚本文件执行等价性检查

与综合实验类似，逻辑等价性检查的命令也可以保存在脚本文件中，用于多次批处理运行。实验数据中提供了脚本文件 `counter.do`，用于对第三节实验得到综合和可测性设计网表文件进行检查，其中增加了以下三条命令对网表进行约束。

将扫描使能端口 SE 约束为 0，保持网表处于功能模式，而非扫描模式：

```
add pin constraints 0 SE -revised
```

忽略网表中的扫描输入和扫描输出端口：

```
add ignored inputs scan_in -revised
```

```
add ignored outputs scan_out -revised
```

如果没有完成第三节综合和可测性实验，可以将上述三条命令删除，并把网表文件名称由 `counter_netlist_dft.v` 改为 `counter_netlist.v` 即可。

使用脚本文件执行等价性检查：

```
$ lec -XL -nogui -color -64 -do counter.do
```

五、思考题

(1) 从 2.4.4 节生成的报告中，找出以下数据并填入表格中。

`report_timing` 命令运行结果中的部分数据

Critical Path	Value
Group	
Start Point	
End Point	
Clock Edge (ps)	
Output Delay (ps)	
Require Time (ps)	
Data Path Delay (ps)	
Slack (ps)	

report_power 命令运行结果中的部分数据

Item	Value
Instance	
Cells	
Leakage Power (nW)	
Dynamic Power (nW)	
Total Power (nW)	

report_qor 命令运行结果中的部分数据

Item	Value
Clock Period (ps)	
Critical Path Slack (ps)	
Total Negative Slack (TNS) (ps)	
Sequential Instance Count	
Combinational Instance Count	
Total Area (um ²)	
Max Fanout	
Min Fanout	
Average Fanout	

(2) 从 4.4 节生成的报告中，找出以下数据并填入表格中。

set system mode lec 命令运行结果

Mapped points: SYSTEM class				
Mapped points	PI	PO	DFF	Total
Golden				
Revised				

compare 命令运行结果

Compared points	PO	DFF	Total

report verification 命令运行结果

Verification Report	
Category	Count
1. Non-standard modeling options used:	

2. Incomplete verification:	
3. User modification to design:	
4. Conformal Constraint Designer clock domain crossing checks recommended:	
5. Design ambiguity:	
6. Compare Results:	