

Formal Verification: by Formality

EE291F

Speaker Yuxuan Li (liyx22025@shanghaitech.edu.cn)

VSPLab@SIST



上海科技大学

ShanghaiTech University

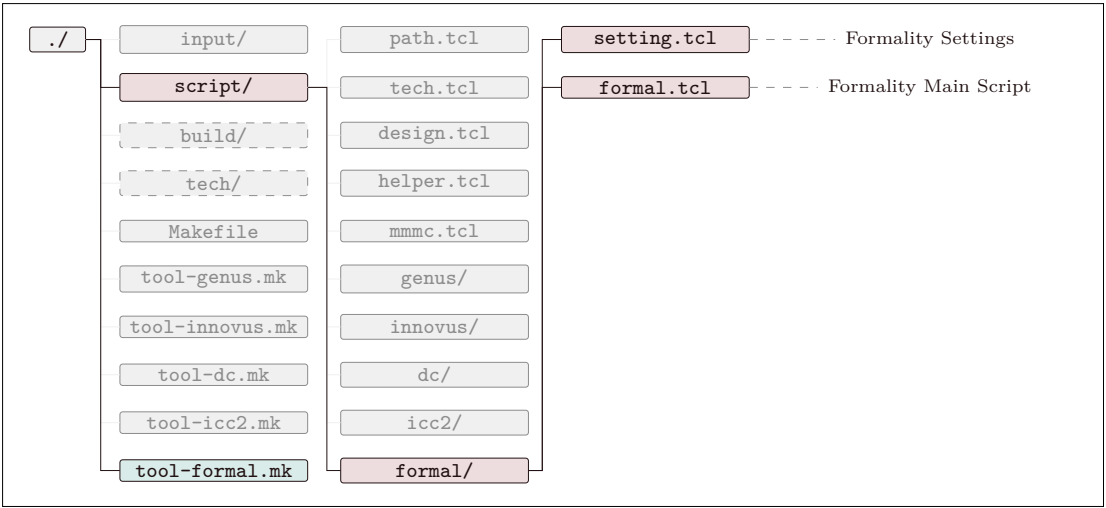
Contents

目录

1 Formality：环境准备

2 Formality：形式化验证

Formality 文件组织总览



Formality：环境准备

Formality 的启动

Formality 可以通过以下方式从命令行启动，`fm_shell` 是 Formality 的交互式 Shell

```
1 fm_shell
```

tool-formal.mk

创建一个独立的 tool-formal.mk，并在主 Makefile 中包含它

```
include tool-formal.mk
```

定义 Formality 的命令名称

```
FORMAL:=fm_shell
```

定义 Formality 的构建目录和日志文件，**所有日志文件都用时间戳命名**

```
FORMAL_BUILD:=${BUILD_DIR}/formal
FORMAL_CMD:=${FORMAL_BUILD}/${RUN_STAMP}-formal.cmd
FORMAL_LOG:=${FORMAL_BUILD}/${RUN_STAMP}-formal.log
FORMAL_COMMIT:=${FORMAL_BUILD}/${RUN_STAMP}-git-commit.txt
```

实现构建目录的创建和删除

```
${FORMAL_BUILD}: | ${BUILD_DIR}
    mkdir -p $@
```

```
formal-clean:
    rm -r -v -I ${FORMAL_BUILD}
```

定义 Formality 的交互启动目标 `formal-shell`

```
formal-shell: | ${FORMAL_BUILD}
    echo "${GIT_COMMIT}" >> $(abspath ${FORMAL_COMMIT})
    echo "${RUN_COMMAND}" >> $(abspath ${FORMAL_COMMIT})
    cd ${FORMAL_BUILD} && ${FORMAL} \
        -x "set_app_var sh_command_log_file {$(abspath ${FORMAL_CMD})}" \
        2>&1 | tee $(abspath ${FORMAL_LOG})
```

- `tee` 命令用于将管道 `|` 接收到的内容同时输出至屏幕和文件。
- `tee` 命令在这里用于产生日志，用于补偿 Formality 本身日志功能不完善的问题。

定义 Formality 的运行目标 formal

```
FORMAL_MAIN:=${FORMAL_SCRIPT}/formal.tcl
```

```
formal: | ${FORMAL_BUILD}
    echo "${GIT_COMMIT}" >> $(abspath ${FORMAL_COMMIT})
    echo "${RUN_COMMAND}" >> $(abspath ${FORMAL_COMMIT})
    cd ${FORMAL_BUILD} && ${FORMAL} \
        -x "set_app_var sh_command_log_file {$(abspath ${FORMAL_CMD})}" \
        -file $(abspath ${FORMAL_MAIN}) < /dev/null \
        2>&1 | tee $(abspath ${FORMAL_LOG})
```

formal.tcl

Formality 的流程都写在 formal.tcl 中，它不需要像 Genus 和 DC 那样分阶段

```
set TOOL formal
source [file join [file dirname [info script]] .. path.tcl]
source [file join ${SCRIPT_DIR} tech.tcl]
source [file join ${SCRIPT_DIR} design.tcl]
source [file join ${FORMAL_SCRIPT} setting.tcl]
set_host_options -max_cores 8
```

Formality：形式化验证

等价性验证

形式化验证用数学方法证明两个设计的**功能等价**，它并不验证功能是否符合设计预期

- **参考设计**（Reference）：通常是综合前的 RTL，作为参照。
- **实现设计**（Implementation）：通常是综合后的门级网表。

形式化验证在两者间寻找**比较点**（Compare Point），并证明每一对比较点的功能完全一致。

立志成才 报国裕民

formal.tcl

创建输出目录，${OUTPUT_DIR} 在 setting.tcl 中定义

```
1 file mkdir ${OUTPUT_DIR}
```

```
1 set OUTPUT_DIR [file join ${FORMAL_BUILD} output]
```

开启与 Synopsys 综合工具适配的相关设置

```
1 set synopsys_auto_setup true
```

formal.tcl

读取 DB 文件，形式化验证不关心时序，使用 FF 或 SS 都可以

```
foreach DB_FILE ${DB_LIST_FF} {
    read_db ${DB_FILE}
}
```

设置 SVF 文件（SVF 文件记录了综合过程中做的变换）

```
set_svf -ordered ${DC_SVF_01} ${DC_SVF_02}
```

formal.tcl

读入参考设计，-container r 指定放入参考容器

```
read_verilog -container r -libname WORK ${RTL_LIST}
set_top r:/WORK/${DESIGN}
```

读入实现设计，-container i 指定放入实现容器

```
read_verilog -container i -libname WORK ${DC_NETLIST}
set_top i:/WORK/${DESIGN}
```

formal.tcl

`verify` 执行形式化验证，**这是整个流程的核心命令**

```
1 verify
```

查看 SVF 的处理情况

```
report_guidance > [file join ${OUTPUT_DIR} report_guidance.rpt]
```

查看失败的比较点

```
report_failing_points > [file join ${OUTPUT_DIR} report_failing_points.rpt]
```

查看未匹配的比较点

```
report_unmatched_points > [file join ${OUTPUT_DIR} report_unmatched_points.rpt]
```

查看黑盒

```
report_black_boxes > [file join ${OUTPUT_DIR} report_black_boxes.rpt]
```

保存会话，便于下次启动时直接恢复匹配和比较结果

```
1 save_session -replace [file join ${FORMAL_BUILD} ${DESIGN}_formal_session]
```

Thank You

谢谢观看