//|本文总结于 2020/9/1

新手小白进入项目的第一步，一定是要好好读一遍小乌龟里各类 spec，知道如何 start a project，不然会无法正常使用工具。

进行项目验证的第一步是制定 VO 并且 review 完成，然后写 hvp；第二步使用 ATB 工具搭建 TB；第三步调用 ICRun，选定 hvp，然后 generate tc，产生对应的空 cases 文件夹；第四步在 tc/cascs 相应文件夹下 building case。

# 1. ATB

uGenHier.py -y name.yaml

**问题：**生成失败，报 error

**解决方法：**注意这里起名字不能跟 RTL 中现有的重名；

vip 部分不能为空，可以保留一个 ioc；

版本用 ATB_v0.5。

//| 一定要熟悉 makefile 脚本，知道如何使用。

//| ATB 需要自己在 th.sv 里把信号连线，将用到的 vif 使用 uvm_config_db

//|set 在 env 路径的 resource pools 里。

//| DO NOT！不要重复 create，否则很有可能配置的和连接的是不同的空间。

//| MUST！uvm_config_db::get()放在 if-else 判断中，用 uvm_fatal 停

//|止仿真。

//| MUST！所有的例化空间，连接都要放在 env 里。

# 2. GenRegModel

uGenRegModel.py –y name.yaml

**问题：**由于 GenRegModel 与 ATB reg 目录文件版本不一致，导致编译失败。

**解决方法：**目前未更新 ATB 的 regmodel，所以需要用 GenRegModel 生成一版 regmodel，然后覆盖 ATB 的版本。

# 3. VMS

搭建好的 TB 环境要按照 VMS 建立对应的 hierarchy 目录。

HVP 编写按照 hvp spec 规则一步步走，并且参考其他目录下的例子。

# 4. TCs

使用 regmodel 内建 sequence 测试寄存器有如下步骤要注意，并且会遇到下述几点问题。

**问题：** *_reg_hdl.sv 文件里面 `define 的名字与 RTL 不匹配，未按照命名规则来。

**解决方法：** 目前需要手动修改定义名，使后门路径与绝对路径匹配。

RegModel 工具生成的 register model，不光包含 name_reg_model.sv 文件，还有 ivt_reg_reset_seq 、ivt_reg_access_seq 、ivt_reg_bit_bash_seq，三个“内建”sequence， name_reg_info.sv 是 ivt_reg_access_seq 用到的文件，吃设置好的关联数组 masks。

1. 在 case 里做一个 rw sequence，rw_seq 例化 ivt_reg_reset_seq 、ivt_reg_access_seq 两个“内建”seq，在 body 函数里 new 一下，并连接 model，然后调用 seq.start()启动 sequence;

```
--| ivt_reg_access_seq name_regacc_seq;
--| timer_reg_info reg_info;
--| …
--| name_regacc_seq = new("name_regacc_seq");
--| name_regacc_seq.info = reg_info;
--| name_regacc_seq.model = p_sequencer.model;
--| name_regacc_seq.start(null);
```

//| 由于 seq 的父类 virtual_sequence 里指明 p_sequencer 的 sqr，所以
//|可以直接使用 p_sequencer。

//| 注意这里的 sqr 首先得把 model 和 create/build 在 env 里的 model 连
//|接起来。

2. rw_seq 在 rw test 里例化，然后用 m_vsequencer (ATB_v0.5 里的 virtual sequencer 例化名字) start，就 OK 啦；

```
--| timer_reg_info reg_info;
--| …
--| reg_info  = new("reg_info");
--| …
--| seq.reg_info = reg_info;
--| seq.start(m_vsequencer);
```

//| reg_info 在 build_phase 里 new 一下，让 sequence 能引用到 info。

3. name_reg_hdl.sv 定义的名字跟 design 脚本工具不匹配，所以还要手工修改，使名字定义的内容跟后门路径索引内容一一匹配；

   **问题：** design 工具生成的 name_bac_register_block，绝对路径指明的是一个 parameter 参数，在 reg_access_seq 中的 RO 类型 task 里，后门写（poke）前门读，后门无法写（parameter 无法 force），mirror 镜像比对报错。

   影响寄存器名： CHIPID 、 MODULEID 、 CHECKSUM

   **解决方法：** 使用 uvm_resource_db，内建 seq 里有相应的

get_by_name，如果!null, return 不继续执行 task。

```
--| uvm_resource_db#(bit)::set({"REG::",
model.name_bac_block.CHIPID.get_full_name()},
"NO_REG_ACCESS_TEST", 1);
--| uvm_resource_db#(bit)::set({"REG::",
model.name_bac_block.MODULEID.get_full_name()},
"NO_REG_ACCESS_TEST", 1);
--| uvm_resource_db#(bit)::set({"REG::",
model.name_bac_block.CHECKSUM.get_full_name()},
"NO_REG_ACCESS_TEST", 1);
--| uvm_resource_db#(bit)::set({"REG::",
model.name_kernel_block.NAMELOAD.get_full_name()},
"NO_REG_BIT_BASH_TEST", 1);
```

//| 注意这里白皮书里有加“.*”,工具会把.识别成路径，造成没 get 到。

4. sequence 会报错，比如 access_seq 中，*_SET 类型的 reg，如果[0]位配 1，*_STATUS 寄存器会受影响，此时需要在 access_seq 中的 name_reg_info 文件中添加 masks 信息：

```
--| masks["REGISTER_BANK_CLOCK_ENABLE_SET"] = 32'hFFFF_FFFE;
```

//| 注意赋值要在 task 或者 function 里做。

**问题：** RW 类型的 reg，前门写前门读，但是由于没有 clk，写不进入，会导致 check 比对错误。

**解决方法：** 使用 uvm_resource_db

**问题：** bit_bash seq 寄存器测试 RW 类型报错，line136，val 与 exp 不一致，check 比对错误。

**解决方法：** 原先 debug 时候误认为是没有配 kernel clk，写不进去造成的。但是在另一个环境中发现有 clk 也会遇到这个问题。bit_bash_seq Line127, rg.get()获取期望值，这里获取的是上一次 write（）的期望值。实验发现，在 get（）前，加上#1，才会 get 到目前 write（）的期望值。

# 5. ICRun

//|Module IP 为例　2020.8.20

TCs 跑通后就需要 ICRun 跑 regression，有下述几点流程需要注意。

1. 如图 F.1 首先在 design_repo/rtl/MODULE NAME/ 路径下 mkdir verif 文件夹，然后在 verif 文件夹下创建 tc/ atb/ etc/ scripts/ vplan/文件夹，

   - atb/下放置如下图 F.2 hierarchy 所示 ATB 生成的文件目录；
   - etc/放 ICRun.cfg test_icrun_tmp.cfg 等文件；
   - vplan/放置编写好的 hvp；

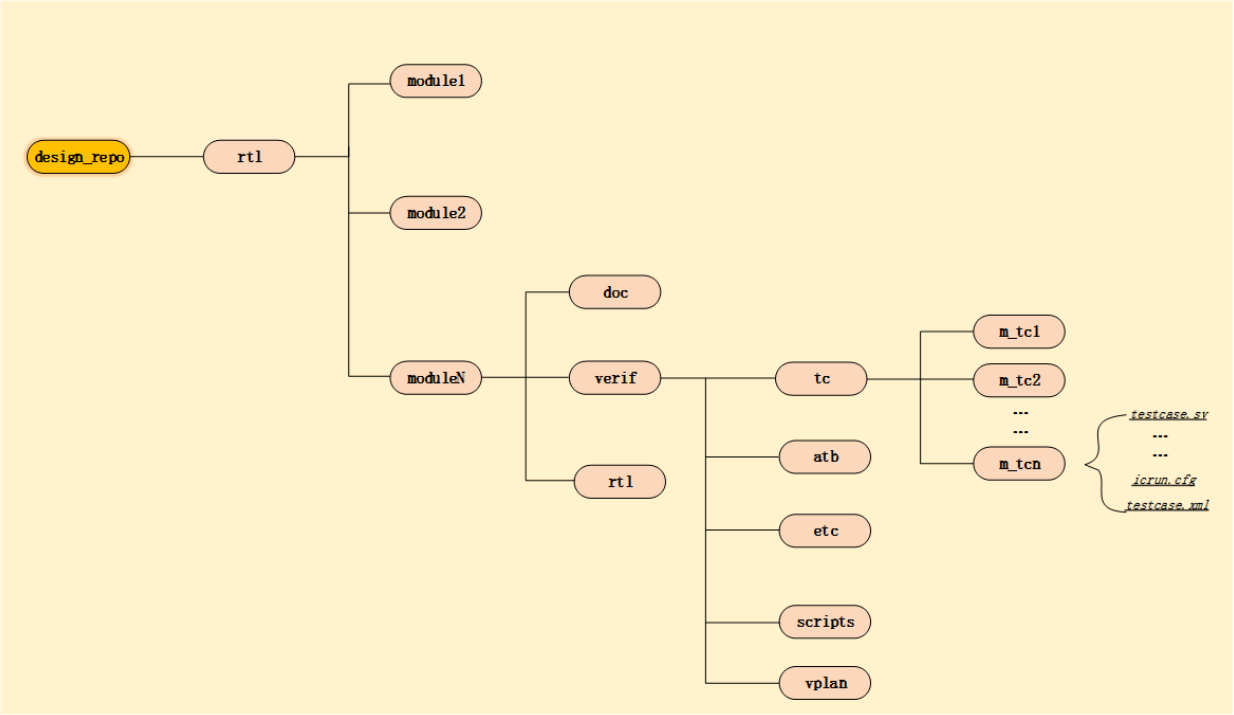


Figure 1 VMS hierarchy

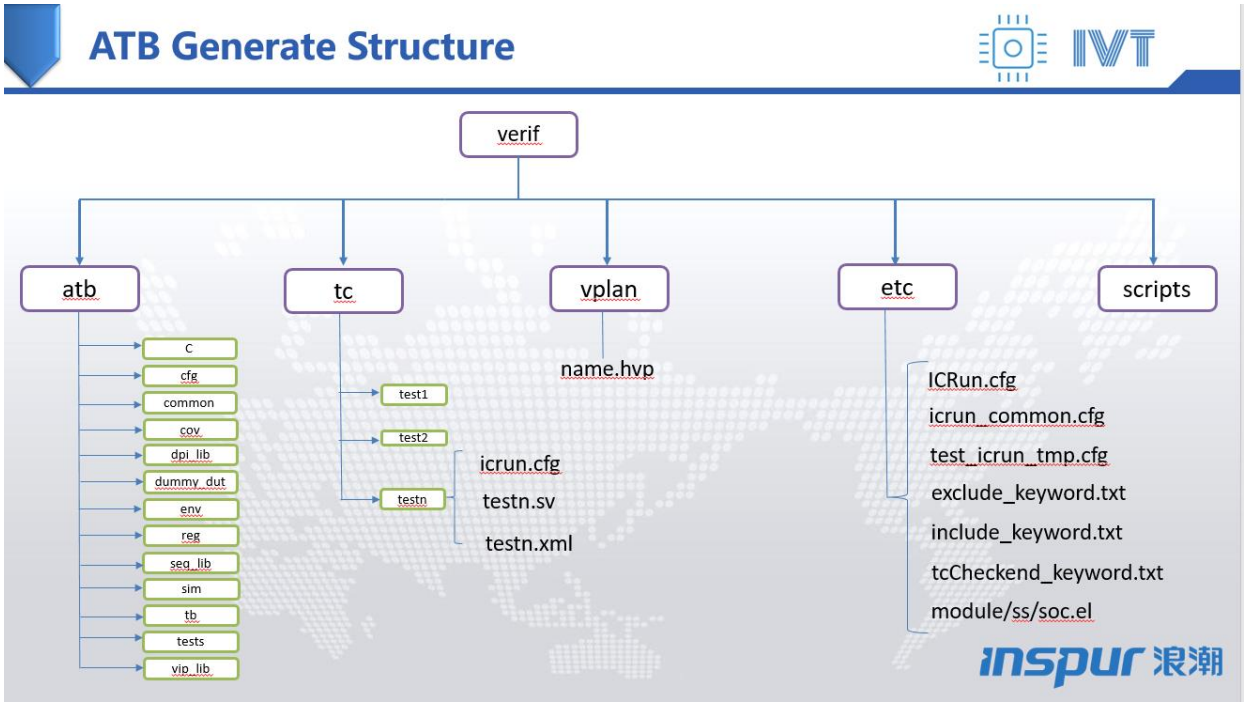


Figure 2 ATB hierarchy

2. 配置 ICRun.CFG 文件,

   - GL_SIM_DIR 路径为创建 simulation 文件夹的目录;
   - GL_LOG_DIR 路径为 simulation/logtree，regression 生成的所有 case log 目录;
   - GL_TARGET_DIR 路径为 elab 完成后 WORKAREA/playground/project_lib 生成的编译文件目录;
   - GL_USR_CFG_TPL 路径为用户自己写的 test_icrun_tmp.cfg 路径;

3. test_icrun_tmp.cfg

   **Target_name 为 $WORKAREA/playground/project_lib/vcsobj/ 路径下编译产生的文件名字;**

   scripts/ 用不到脚本的话，可以注释掉 copy_file_list 相关行;

   link_file_list 中/project_lib/vcsobj/+elab 生成的文件名+*;

   run_opts 写自己的 run options， **PS：+TESTNAME={{testname}};**

4. hvp

   - `annotation string module_hier = “ahb_timer.verif”;`
   - `annotation string module_hier = “ahb_spi.verif”;`

   对于 module IP 级别，只填写上述这部分就行;

5. RUN

   Command: `$WORKAREA/tool_repo/ICRun/ICRun_V0.1/bin/ICRun -i ../verif/etc/ICRun.CFG &`

   **问题：** TC 无法根据 test_icrun_tmp.cfg 模板 generate 出对应的 cfg。

   **解决方法：** 不要使用 module load icrun。

6. Coverage

   elab 命令加上 COVERIGE=1;

   test_icrun_tmp.cfg 文件中, run options 加上 coverige 相关命令-cm,

   如果直接设置覆盖率, 可以把 target_name 后面加上.cov, 相当于下面 gui 界面的操作，仿真结果会放入 name.cov.vdb/ 目录里;

   gui 界面左侧 RLs 下面 name.hvp 右键出现 rl.set_cov，可以与 target name 匹配。

   Command: `bsub -Is dve -full64 –covdir $WORKAREA/playground/project_lib/vcsobj/shashasha.cov.vdp/`

# 6. Coverage

推荐使用 Verdi 查看覆盖率:

`verdi –cov –covdir simv.vdb/ &`

function coverage 写在 class coverage 里，covergroup 语法参考绿皮书，有详细解释。

功能覆盖率调用 sample 有两种方法:

1. 在coverage里使用task run_phase调用sample函数使用analysis port;
2. 在 monitor 里有例化 uvm_analysis_port，coverage 的父类 uvm_subscriber 里定义了 analysis_export，env 这个容器类里把 ap 和 exp 连接(connect)起来，就可以在 monitor 里调用 ap.write()。如果 monitor 的 ap connect 了多个 export/imp，例如 analyzer、collector，那么每次调用 ap.write()，每一个连接的 export 对应的 class 都会执行 task write。

有些 illegal 场景才能覆盖 code coverage，这时候就需要 merge coverage。注意重新跑 case 会用新的 covergae 覆盖旧的，所以需要将 vdb/ 文件夹另存一份。

merge 命令如下所示：

```
bsub –Is urg –full64 –dbname xxx.vdb –dir x1.vdb x2.vdb x3.vdb ... -map spi_top_tb -noreport
```

其中 -dbname xx 是 -dir x1.vdb -dir x2.vdb 两个文件夹 merge 一起后产生的文件名字 xx，

-map xx_xx 是加上指定 merge 的 RTL module 名字。

举一个例子，其中 el 文件为指定 exclude coverage 信息文件。

```
urg -full64 -metric line+tgl+cond+fsm+assert+branch -warn none -dbname
<merge_coverage_name>.vdb \
-dir ${vdb_path}/simv.vdb \
-elfile ${path}/${el_file_name.el}
```