0903-莫海-周报

✓ GenerateSimFiles

该命令主要是拷贝编译所所需要的文件到目标文件夹，调用的是`generators\utilities\src\main\scala\Simulator.scala`中的`GenerateSimFiles`函数，该函数混入了`HasGenerateSimConfig`特质，该特质决定传入的传输选项。

- 「sim/simulator」

  通过模式匹配决定使用仿真的软件是`vcs`还是`simulator`

- 「td/target-dir」

  需要拷贝到的目标文件夹

- 「df/dotFName」

  生成的dot-f文件的名称

参数的使用的是`scala`中的`scopt.OptionParser`对象进行判断调用，同时也通过该对象的`parser`成员函数判断是否参数匹配，对匹配的参数进行下一步操作，即调用`GenerateSimFiles`对象的内部关系

- 首先调用`writeFiles`函数，此函数为该类中的入口函数，其他的函数都通过该函数进行初始的调度。
- 在writeFiles中首先创建一个`boorrom`文件夹，并拷贝一些img文件到该文件夹
- 创建传入参数的目标文件夹
- 将`resources`中需要的依赖文件拷贝到目标文件夹

- 使用 writeDotF 函数保存成 dot-f 文件

✓ Generator

命令中的 GENERATOR_PACKAGE 变量很多都是为 chipyard ，所以我主要关注了该package下的 generator 函数。该函数主要作用是设置 FIRRTL 的一些编译选项，控制编译的时候的一些行为，所在目录为 generators\chipyard\src\main\scala\Generator.scala ，通过 StageMain 对象传入一个 ChipyardStage 对象，我主要关注了ChipyardStage中的一些编译选项，而在StageMain中则主要是一些FIRRTL的状态转换以及行为控制、消息传递，没有仔细去进行分析。

ChipyardStage

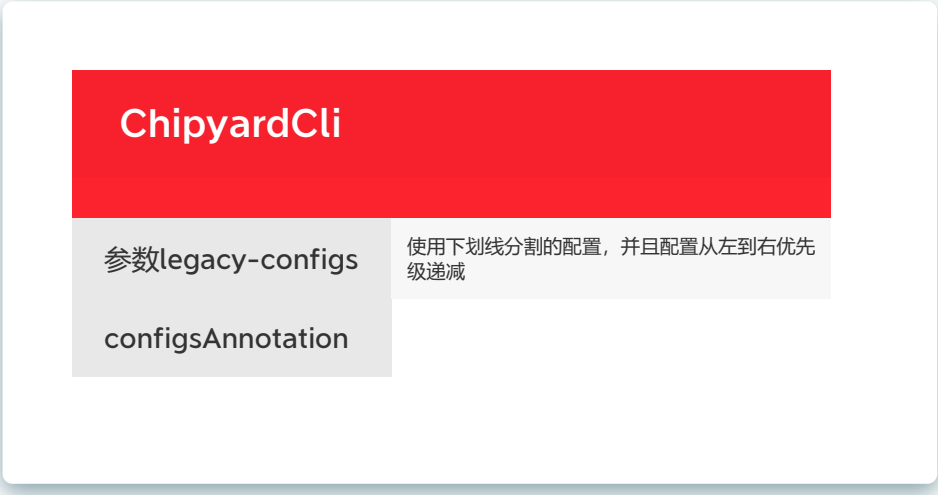
在ChipyardStage中主要有两个比较重要的成员，一个是 Shell 成员另外一个是 target 成员，其中Sell成员主要是控制编译器的行为，而target成员主要调用FIRRTL中的 Depedency 对象，目的是找到一些依赖的库与生成编译过程中的log。

「shell」

该成员通过混入了 ChipyardCli ， RocketChipCli ， ChiselCli ， FirrtlCli 四个特质来接守输入的参数，同时控制编译器的行为。

- 「ChipyardCli」

  该特质接收chipyard的生成选项，接收参数为 legacy-configs ，接受一串以下划线界定的参数，参数的优先级从右到左递减。参数切分处理之后还会进一步调用 ConfigsAnnotation 生成FIRRTL接收的 Annotation 类，在接下来的篇幅中会进一步介绍，该特质的思维导图如下所示。



- 「RocketChipCli」
  该特质是 `RocketChip` 的编译选项，其中有三个类来处理该编译的参数，分别为 `TopModuleAnnotation`，`OutputBaseNameAnnotation`，`ConfigsAnnotation`，

  - TopModuleAnnotation

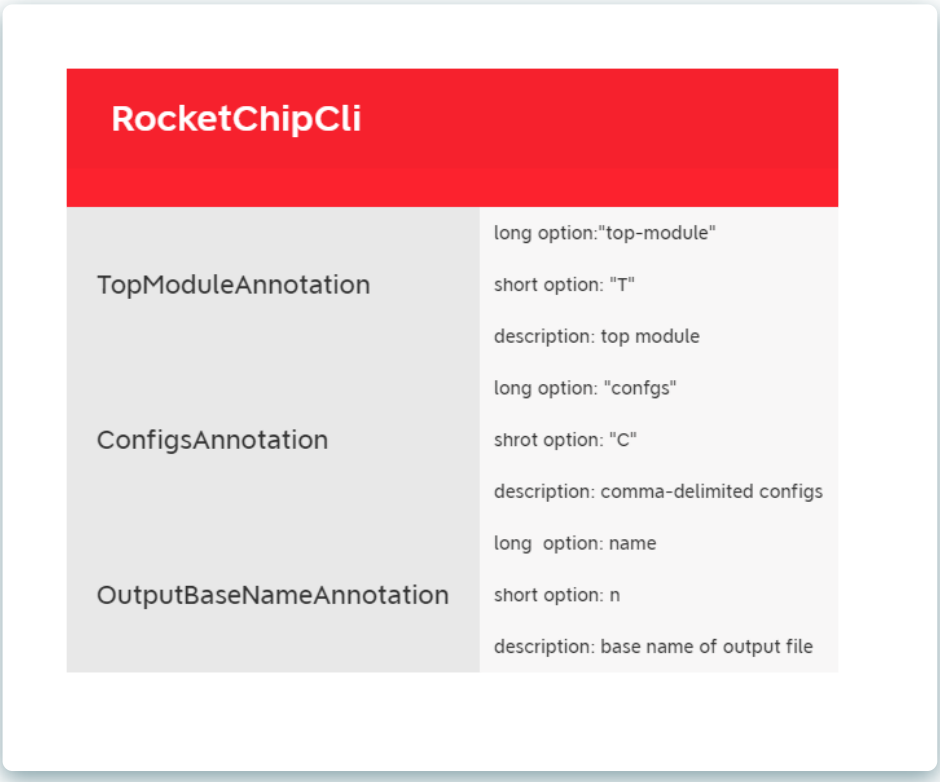
    接收参数 `top-module/T`，将输入的顶层模块参数转化为Annotation传递给 `FIRRTL`

  - ConfigsAnnotation

    接收参数 `configs/c`，配置参数以句点字符串进行配置，切分之后以 `Annotation` 的形式传递给 `FIRRTL`

  - OutputBaseNameAnnotation

    接收参数 `name/n`，设置基础的输出文件名



- 「ChiselCli」

  该特质是`chisel`的编译选项，选择是否使用`FIRRTL`，跟踪打印在`transform`过程中发生异常的信息，chisel生成Annotaion的选项等，他通过三个类来实现上述的目标，分别为`NoRunFirrtlCompilerAnnotation`，`PrintFullStackTraceAnnotation`，`ChiselGeneratorAnnotation`。

  - NoRunFirrtlCompilerAnnotation

    接收参数`no-run-firrtl/chnrf`，使用了该选项就不使用`FIRRTL`进行编译，直接由chisel得到硬件电路。

  - PrintFullStackTraceAnnotation

    接收参数`full-stacktrace`，即当发生异常的时候，将会以栈的形式全打印出来

  - ChiselGeneratorAnnotation

    接收参数`module`，通过给出类的名字跟路径搭一个chisel module，且不能带参数，要是找不到将会抛出异常。

- 「FirrtlCli」

  该特质是FIRRTL编译器的编译选项，主要通过 FirrtlFileAnnotation ， OutputFileAnnotation ， InfoModeAnnotation ， FirrtlSourceAnnotation ， CompilerAnnotation ， RunFirrtlTransformAnnotation ， EmitCircuitAnnotation ， EmitAllModulesAnnotation ， NoCircuitDedupAnnotation ，这些特质实现对编译选项的识别与对应函数的执行

FirrtlCli

FirrtlFileAnnotation，FIRRTL的输入文件	long option: input file short option: i description: FIRRTL输入文件
OutputFileAnnotation，FIRRTL的输出文件	long option: output-file short option: o description: FIRRTL输出文件
InfoModuleAnnotation，如何进行消息处理	long option: info-module description: 消息处理配置 helpvaluename: use / gen / ignore / append
FirrtlSourceAnnotation	long option: firrtl-source description: An input FIRRTL circuit string
CompilerAnnotation，编译成啥样的温江	long option: compiler description: FIRRLT使用的编译方式 helpValueName: none — 不编译 / hilgh — HighFIrrtlCompiler / low — LowFirrtlCompiper / middle — MiddleFirrtlCompiler / verilog — verilogCompiler / mverilog — MinimumVerilogCompiler / sverilog — SystemVerilogCompiler
RunFirrtlAnnotation，在编译过程中转换	long option: — custom-transforms short option: — fct description: — 在编译过程中执行转换
EmitCircuitAnnotation，run the specified module emitter (one file per module)	long option: emit-circuit short option: E description: run a specified circuit emitter (all module in one file) helpValueName: none — 不编译 / hilgh — HighFIrrtlCompiler / low — LowFirrtlCompiper / middle — MiddleFirrtlCompiler / verilog — verilogCompiler / mverilog — MinimumVerilogCompiler / sverilog — SystemVerilogCompiler
EmitAllModuleAnnotation，也是编译成啥文件的选项，但是与EmitCircuitAnnotation相对应	long option: emit-module short option: e description: run the specified module emitter (one file per module) helpValueName: none — 不编译 / hilgh — HighFIrrtlCompiler / low — LowFirrtlCompiper / middle — MiddleFirrtlCompiler / verilog — verilogCompiler / mverilog — MinimumVerilogCompiler / sverilog — SystemVerilogCompiler

NoCircuitDedupAnnotation	long option: no-dedup description: Do not dedup Module

✓ GenerateTopAndHarness

该函数也是FIRRTL的一些编译选项，但与上面不同的是，此处控制的主要是编译成的目标代码的电路结构，上面主要是FIRRTL编译器编译之前的一些选项，如输入输出文件啦，中间过程如何转化等。

同时生成了很多

同时他不直接在`GenerateTopAndHarness`中处理信息，这只是一个接口，他通过继承`GenerateTopAndHarnessApp`，然后调用GenerateTopAndHarnessApp中相应的执行函数进行。

该函数主要通过继承`HasFirrtlOptions`与`HasTapeoutOptions`来处理参数选项。

其中`HasTapeoutOptions`主要是用来控制生成RTL的代码结构的特质，而`HasFirrtlOptions`我个人觉得还是关于FIRRTL中参数传递，中间转换的一些编译控制选项，与之前的有一部分是重叠。

解释中主要分为两个阶段，第一个阶段是收集在`synTop`之下的模块，第二阶段是删除所有的module然后生成测试工具。

具体的编译选项跟描述如下表所示。

GenerateTopAndHarness，主要是设置编译成的目标RTL代码的结构，与一些FIRRTL的编译选项，有一部分跟上面有重叠

HasFirrtlOptions firrtl的编译选项，与上一个编译命令有些许重叠

- input-file
  - opt: input-file
  - abbr: i
  - description: use this to override the default input file name , default is empty
- output-file
  - opt: output-file
  - abbr: o
  - use this to override the default output file name, default is empty
- annotation-file
  - opt: annotation-file
  - abbr: faf
  - description: Used to specify annotation files (can appear multiple times)
- force-append-anno-file
  - opt: force-append-anno-file
  - abbr: ffaaf
  - description: do nothing, not use anymore
- output-annotation-file
  - opt: output-annotation-file
  - abbr: foaf
  - description: use this to set the annotation output file
- compiler
  - opt: compiler
  - abbr: X
  - description: 选择编译成何种形式得RTL代码
- info-mode
  - opt: info-mode
  - description: 与之前的一致，就是ignore，use 灯这四种形式
- custom-transforms
  - opt: custom-transforms
  - abbr: fct
  - description: runs these custom transforms during compilation
- inline
  - opt: inline
  - abbr: fil
  - description: Inline one or more module (comma separated, no spaces) module looks like "MyModule" or "MyModule.myinstance
- repl-seq-mem
  - opt: repl-seq-mem
  - abbr: repl-seq-mem
  - description: Replace sequential memories with blackboxes + configuration file
- list-clocks
  - opt: list-clocks
  - abbr: clks
  - description: List which signal drives each clock of every descendent of specified module
- split-modules
  - opt: split-modules
  - abbr: fsm
  - description: Emit each module to its own file in the target directory
- no-check-comb-loops
  - opt: no-check-comb-loops
  - description: Do NOT check for combinational loops (not recommended)
- no-dce
  - opt: no-dce
  - description: "Do NOT run dead code elimination
- no-dedup
  - opt: no-dedup
  - description: Do NOT dedup modules

HasTapeoutOptions 这主要是设置harness相关的一些结构，电路的结构

- harness-output
  - opt: harness-o
  - abbr: tho
  - description: use this to generate a harness at <harness-output>（使用他来生成线束）
- syn-top
  - opt: syn-top
  - abbr: tst
  - description: use this to set syntop
- top-fir
  - opt: top-fir
  - abbr: tsf
  - description: use this to set topFir
- top-anno-out
  - opt: top-anno-out
  - abbr: tsaof
  - description: use this to set topAnnoOut
- top-dotf-out
  - opt: top-dotf-out
  - abbr: tdf
  - description: use this to set the filename for the top resource .f file
- harness-top
  - opt: harness-top
  - abbr: tht
  - description: use this to set harnessTop
- harness-fir
  - opt: harness-fir
  - abbr: thf
  - description:use this to set harnessFir
- harness-anno-out
  - opt: harness-anno-out
  - abbr: thaof
  - description: use this to set harnessAnnoOut
- harness-dotf-out
  - opt: harness-dotf-out
  - abbr: hdf
  - description: use this to set the filename for the harness resource .f file
- harness-conf
  - opt: harness-conf
  - abbr: thconf
  - description: use this to set the harness conf file location

✓ MacroCompilerc

这个函数比较复杂，调用了两次，大概是生成不同层次的memory，接下来还需要配合文档在仔细阅读，两次调用此一次大概是生成`TOP_SMEMS`，第二次生成的是`HARNESS_SMEMS`，具体如何生成的我先不去理解，这里先列举他的参数配置相关的内容，具体如下表所示。

MacroCompiler

-n, --macro-conf	The set of macros to compile in firrtl-generated conf format (exclusive with -m)
-m, --macro-mdf	The set of macros to compile in MDF JSON format (exclusive with -n)
-l, --library	The set of macros that have blackbox instances
u, --use-compiler	Flag, whether to use the memory compiler defined in library
-v, --verilog	Verilog output
-f, --firrtl	FIRRTL output (optional)
-hir, --hammer-ir	Hammer-IR output currently only needed for IP compilers
-c, --cost-func	Cost function to use. Optional (default: \" default\")
-cp, --cost-param	Cost function parameter. (Optional depending on the cost function.). e.g. -c ExternalMetric -cp path /path/to/my/cost/script
--force-compile [mem]	Force the given memory to be compiled to target libs regardless of the mode
--force-synflops [mem]	Force the given memory to be compiled via synflops regardless of the mode

✓ 总结&&Todo

「总结」这只是单纯的看了`common.mk`得到的相应的部分编译配置，在全局搜索看那些模块调用了common.mk发现，其实很多都是独立的模块，包括在前面文档中编译过的VCS或者Verilator其实都有调用，其实都调用了common.mk，但是肯定不止是这几个文件就可以生成对应的仿真工具的。还有就是一些编译选项具体所

指有点不明白，需要进一步去看文档，比如所在`MacroCompiler`中的`HARNESS_SMEMS`这个选项具体到底是想要生成的那一部分的menory需要具体再去细看。

「todo」接下来还需要去看下文档，包括他的具体的设计结构，也就是相关的设计的层次结构，然后再回来看这个编译文档可能会得到更多的新搜获