使用 **Sulp** 构建你的 **flow**

初衷

1、简化 **flow**，

单一的 flow

插件式的 flow

瀑布流式的数据传递

2、统一 **sim**、 **syn**、 **fpga** 的 **flow** 平台

Verif demo 的基本操作

1 、 clone 仓库

```
tmp $ git clone git@gitlab.cn.sifive.com:project/SFC_BIC.git
```

2 、切换到 sulpur-sharp 分支

```
mp $ cd SFC_BIC/
```

```
r) $ git checkout sulpur-sharp
```

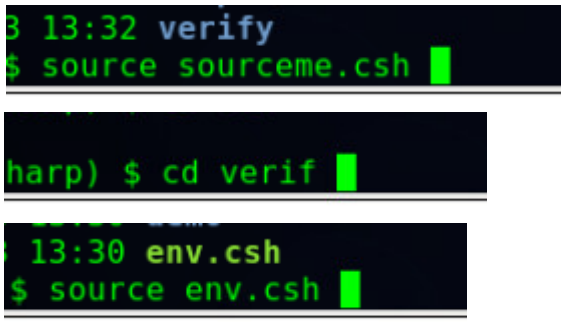
```
) $ git pull origin sulpur-sharp
```

3 、更新 submodule

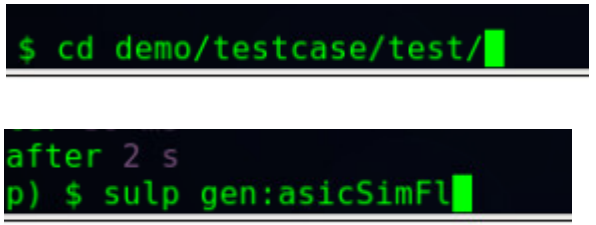
```
arp) $
arp) $ git submodule update --init --recursive
```

Verif demo 的基本操作

4 、 source 环境变量

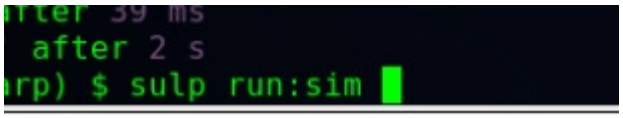


5 、生成 sim 使用的 flist

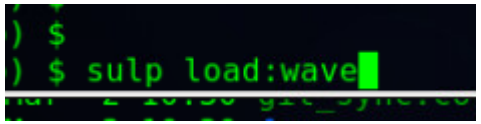


Verif demo 的基本操作

6、执行仿真



7。、打开波形



8、背后的 task

```
sulp --tasks

sulp gen:asicSimFl

# sulp build:sccaseApi

# sulp add:sccase --case test
cd testcase/test
# sulp build:sccase

# sulp build:sclib
# sulp run:elab
sulp run:sim
```

verdi

```
sulp load:wave
```

使用 **sulp-verif** 做独立模块的验证

1、指定 **flist**

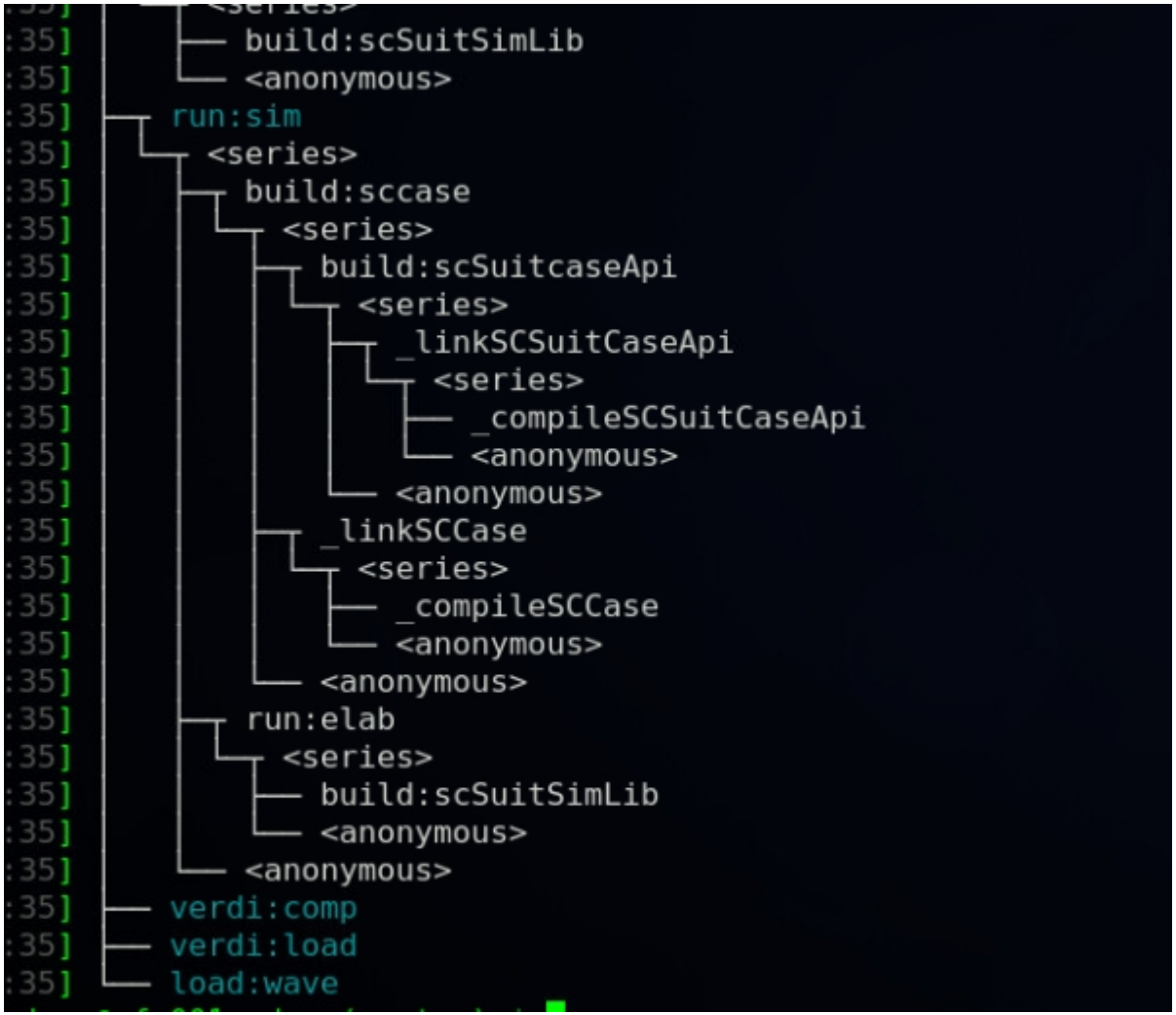
2、指定与编译 **testcase**

3、编译 **scSuit lib**

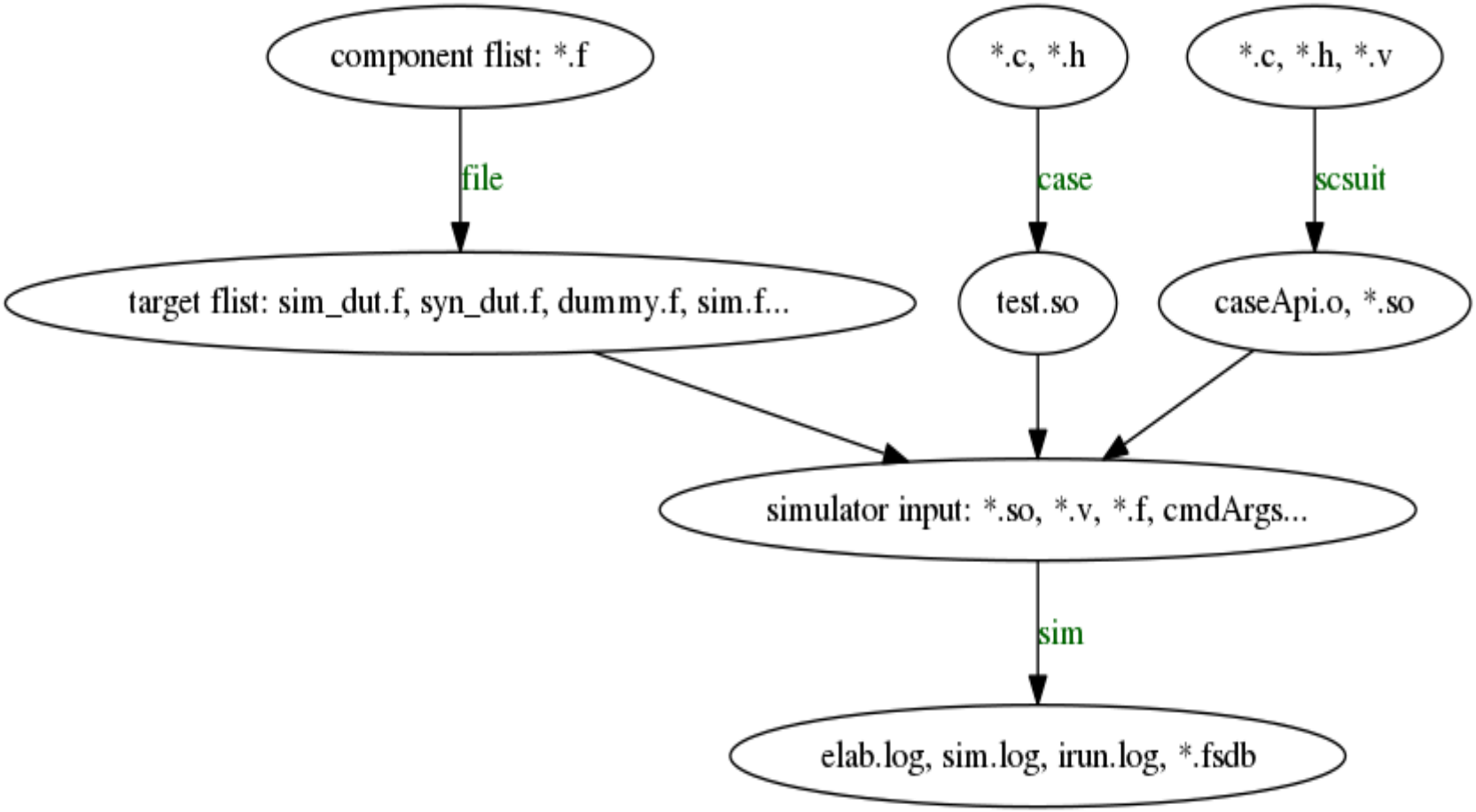
4、 **vlog/elab**

5、 **sim**

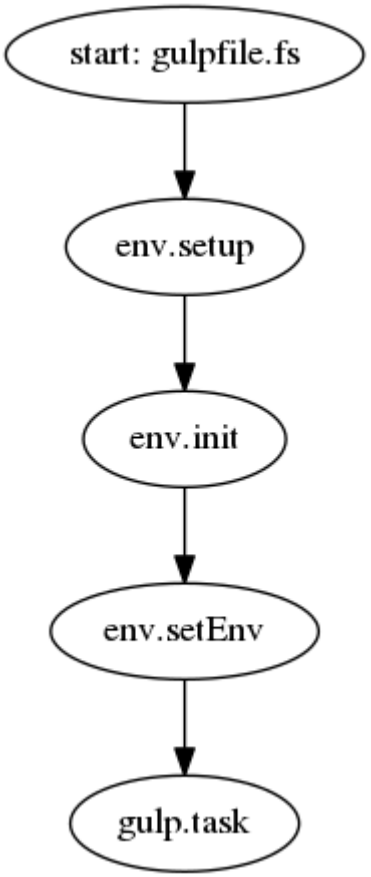
verif 相关的 tasks



Verif flow 数据流与设计思路



sulp 执行过程




```
export {};                          // isolate the namespace
require('json5/lib/register')       // require the module needed
const gulp     = require('gulp')
....
import {fooInterface} from "../protocol/FooInterface" // import the flow data structure interface

let flow:fooInterface// declare the flow scope data structure
let profile:any       // declare the profile variable if needed
....
// local function definition
// gulp task definition
...

module.exports.flowName = 'foo'  // export the flow name, must exist

module.exports.init = ()=>{  // command line arguments definition, must exist
  program.allowUnknownOption()
    .option('--option ') // the option can be find in env.getOpt('foo.option')

module.exports.setEnv = (env)=>{ // fullfill flow data structure, must exist
                                 // env is the top structure include
                                 // the all flow and global
  profile=env.useProfile('foo')  // instance the profile
  flow = env.getFlow('foo');  // create the flow data structure instance
  flow.aaa = env.flow.other.bbb    // fullfill the flow data structure
  flow.bbb = env.getOpt('foo.option')
}
```

文件导入，路径处理层，参数注入 :flow/

参数捕获与命令生成执行： profile

开发你的 **flow**

1 、基于 **ts** 开发 **gulp** 的 **task**

2 、加入已有的脚本