SoC-Gulp 构建工具

原则 & 目标

1. 简单，易于部署
2. 易于构建 **flow**
3. 静态化，设计与 **sulp** 无关，每一步的输出都可以被其他工具介
4. 插件化，易于扩展，可读易于维护
5. 发挥 **gulp task** 的优势：分割组合 **task/** 创建替换 **task.....**
6. 统一的 **flow** 规范
7. 报告 **rpt** 做独立收集
8. 规范 **file flow** 中的导入的设计文件

构建工具简介

- C → make/cmake
- Java → ant/maven/gradle
- Scala → sbt/mill
- Js/ts → Grant/Gulp/webpack/angular-cli

- SoC → ??? make ? Wake? Gulp?

构建工具 **sulp** 的作用

- **task** 管理 / 依赖管理
- **init** ， **git** 、 **vpp** 、
  verif 、 **fpga** 、 **asic** 、 **mem** 、 **signoff**
- 自动化，避免重复劳动
- 加速项目开发迭代

Gulp 简介

入门指南

- 创建任务 (task)
- 异步执行
- 处理文件
- Glob 详解
- 使用插件
- 文件监控

npm 包管理

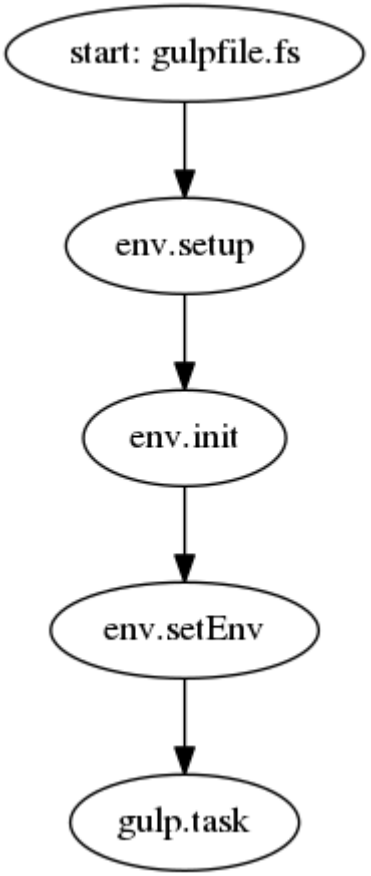
什么是包管理器？

- apt
- yum
- npm
- yarn
- ……

npm中文文档

- npm 是什么
- 安装 npm
- 使用 npm

sulp 执行过程

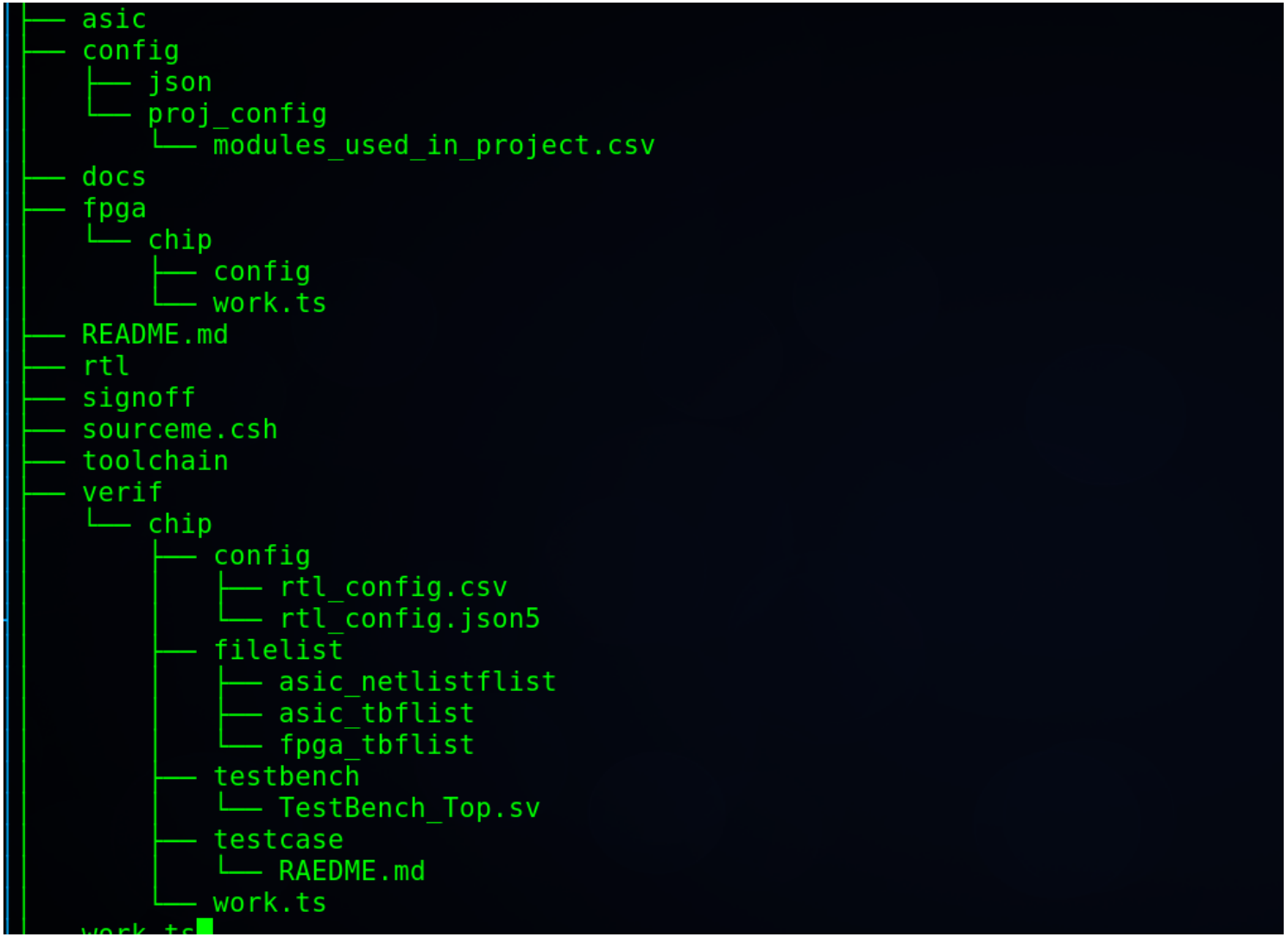



```
export {};                          // isolate the namespace
require('json5/lib/register')       // require the module needed
const gulp     = require('gulp')
....
import {fooInterface} from "../protocol/FooInterface" // import the flow data structure interface

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')
}
```

生命周期

- 初始化
  - Init project
  - Init work
- 配置
  - work.ts
- 执行
  - sulp init:project
  - sulp init:repo
  - …

SoC-Gulp 文件结构

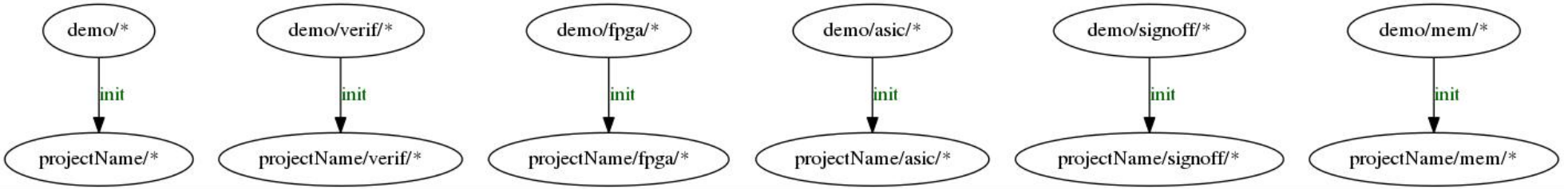


SoC-Gulp flow 介绍

- Init
- git
- verif
- fpga
- mem
- asic

init flow 设计思路

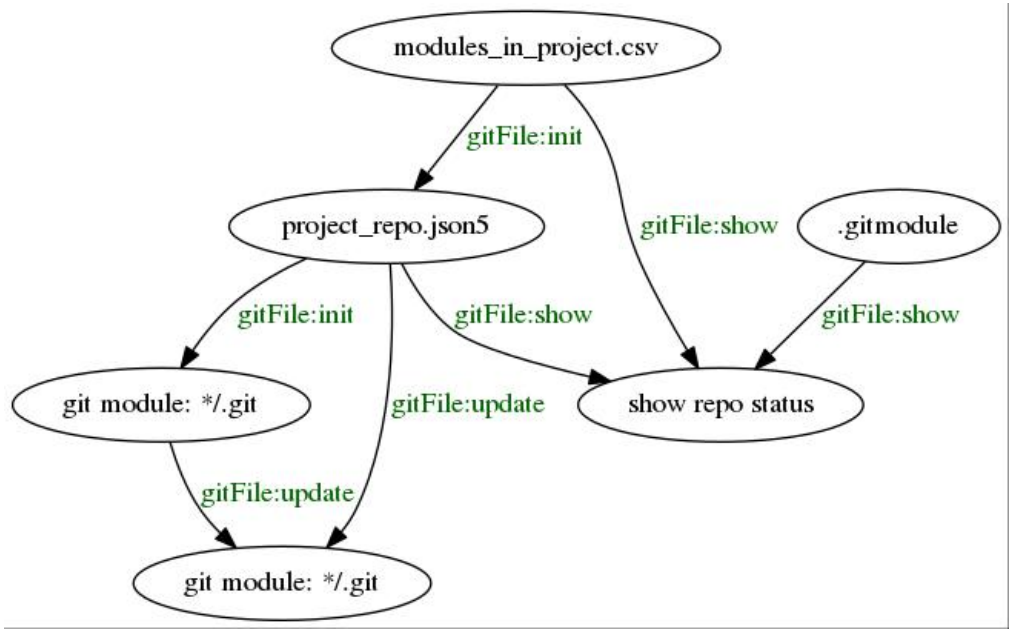
- 设置 sulp 路径
- 初始化项目
  - ➔ 初始化 work.ts 模板
  - ➔ 复制文件
  - ➔ 初始化 git 项目
- 根据项目配置初始 submodule
- 初始化子 work
- 完成项目初始化



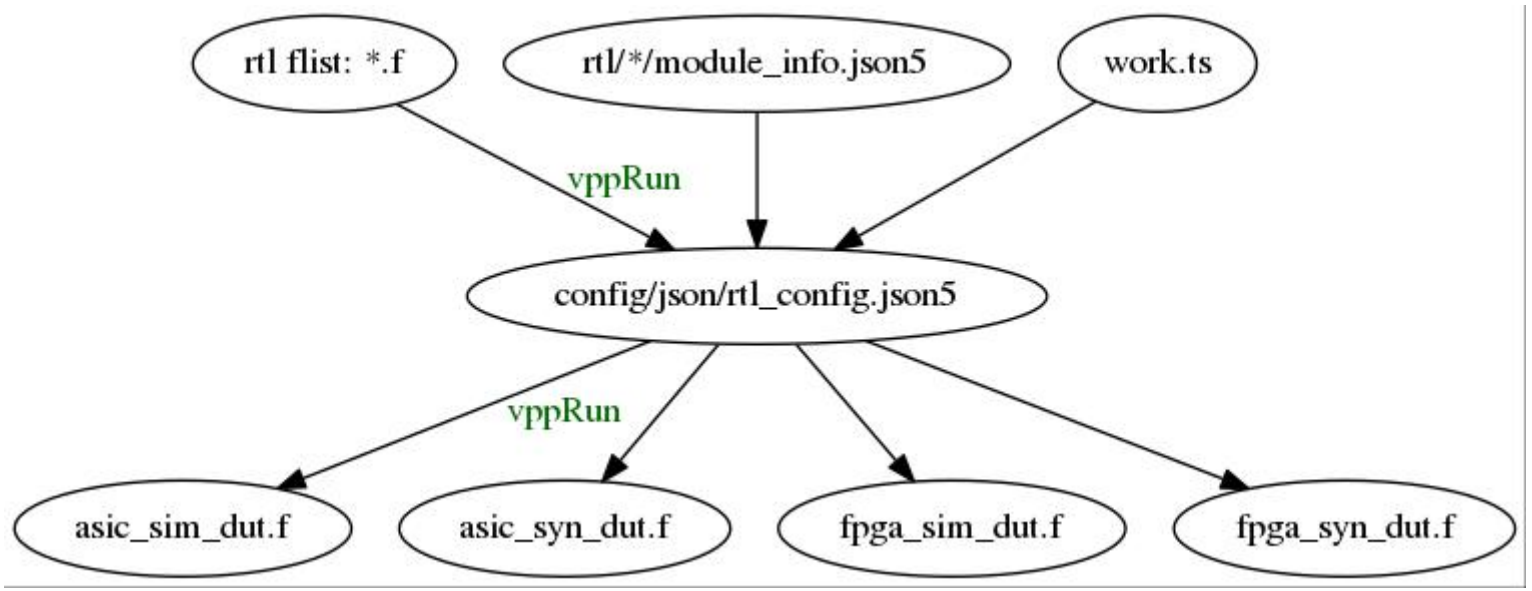
git flow 设计思路

使用 sulp 管理项目中的 submodules

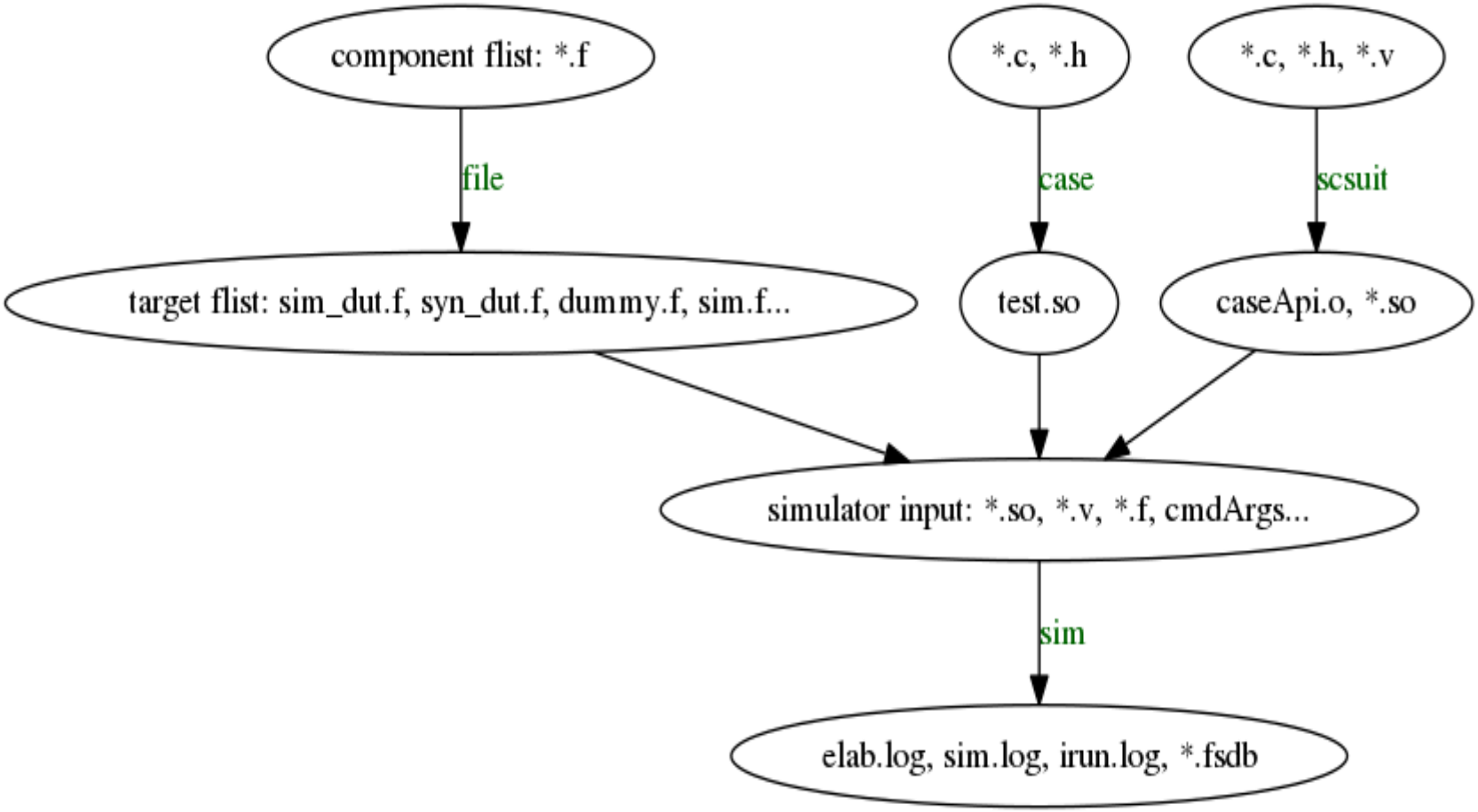
- init　Sulp init:repo
- update　Sulp update:repo
- show　Sulp show:repo



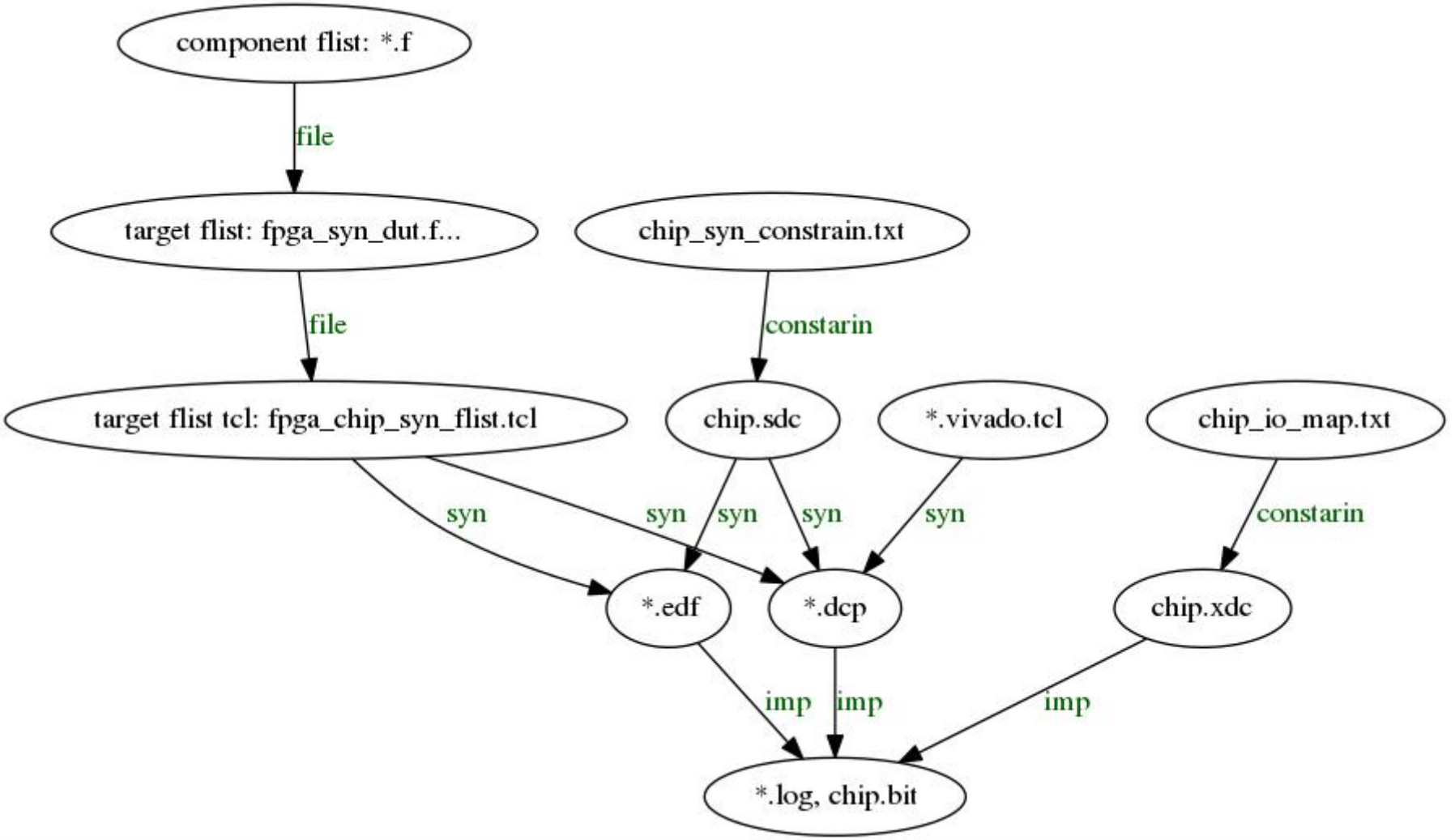
vpp flow 数据流与设计思路



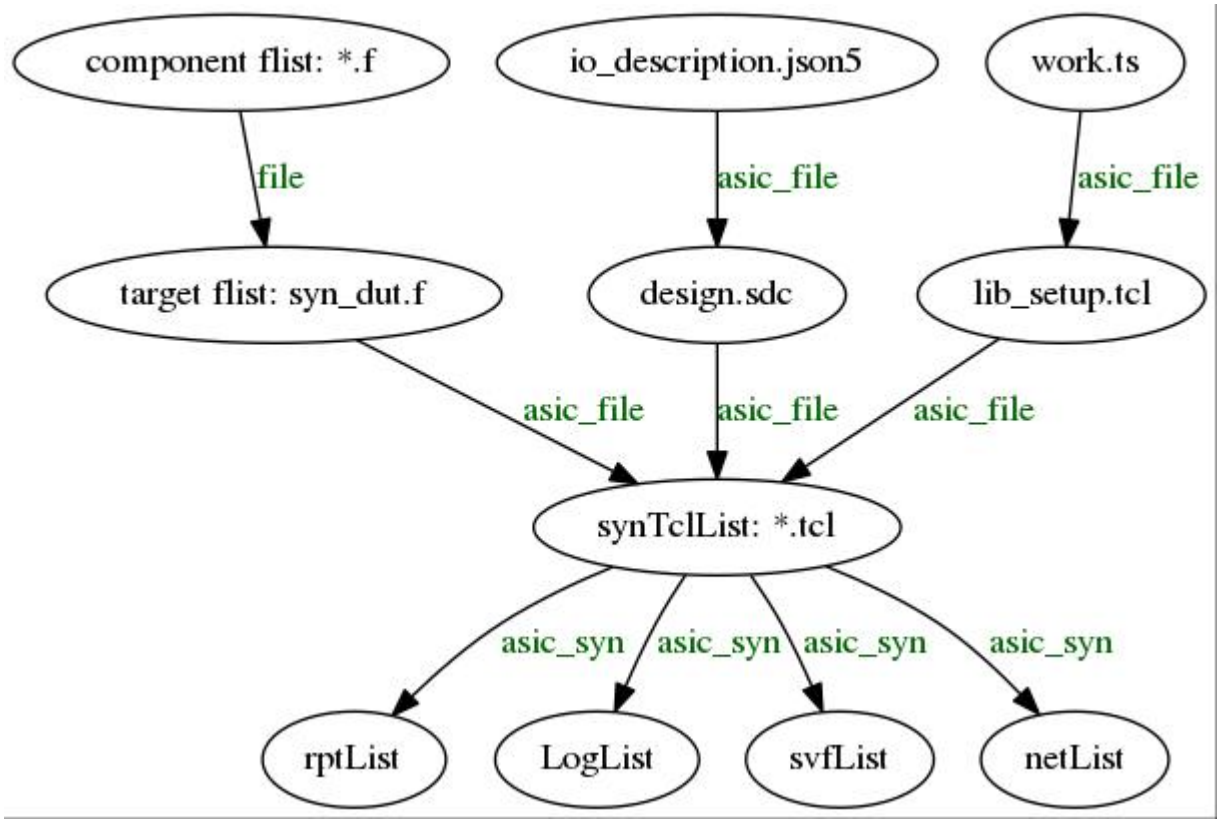
Verif flow 数据流与设计思路



fpga flow 数据流与设计思路

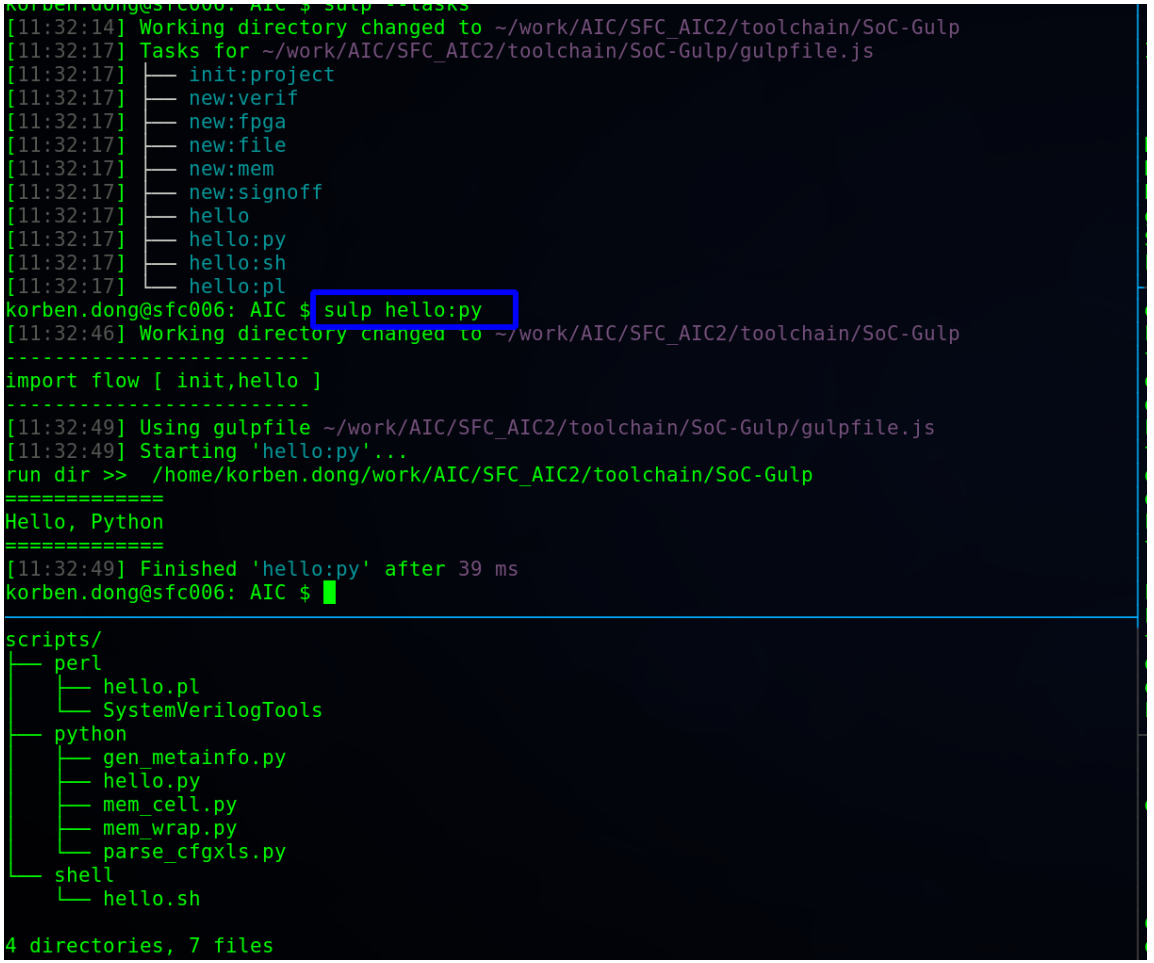


asic flow 数据流与设计思路



集成 perl/python/shell

参考　: flow/hello.ts



flow 结构

推荐结构：

- file – 归档
- run/build – 执行的 flow 任务
- report – 收集报告，生成信息
- regression – 生成 run/build 与 report 的 pattern

推荐规范：

- 编写易于维护的可读的代码

贡献你的 **flow**

推荐编码规范：

- 编写易于维护的可读的代码
- 编写简单的功能单一的 function
- 清晰的命名

总结

新建工程

Sulp - Gulp – npm

Flow 结构

task 管理

插件编码开发

测试

参考资料

Gulp

Typescript

npm

nodejs

常见的构建工具及对比

What is a build tool?