只从宏观上说下，，细节没啥好说的，代码最清楚了， 你一定能看懂(因为我用的都是基础的东西）

首先环境的建立

我们遵循 GNU 的 custom： 由用户运行(不管用户在哪个目录）一只侦测程序 configure 生成环境（目录的建立，脚本的 copy, link 以及 design data 的 link）, 其中大的版本号即由顶层确定的那个在此阶段选择，之后该版本下的东西将不再由 configure 生成

代码的设计：

分为 3 部分

1. 各阶段顶层文件(init.tcl fp.tcl place.tcl cts.tcl postcts.tcl route.tcl)是一套统一的模板，在 configure 的时候解析 copy 到用户目录下，这就意味着用户可以改----所以我们尽量让顶层文件简单简单再简单----用户要改的也就越少
2. 与项目有关的设定放在 proj_setting 文件夹里，proj_setting 这个文件夹是 link 的，项目后续的什么 write_verilog_lvs.tcl write_gds.tcl，insert_std_fill.tcl 啥的也都放在 proj_setting 里，这样用户无需 update 环境就能拿到最新脚本
3. 用户自己的特殊设定放在 user_setting 文件夹里，该文件夹是 copy 在用户目录下，里面包含 place_setting.tcl cts_setting.tcl 等用户存放特殊配置的文件

说下特殊的地方：

库的配置不再由 design_name 这样的变量控制，而是在 configure 的时候确定掉，比如你是用 9track，那就给你选好 9track 的库，你用到了哪些 memory ip 等那就给你选好相应的库，这样每个用户看到的都是与自己有关的东西，同时我们扔掉 design_name 这个变量。

makefile 是个入口， 你可以从它那单个 vim 浏览整个工程(gf, :bd) 。

其他：

抽 spef –------ 独立的 run_starrc.sh 脚本(你很可能没见过）但是却超容易维护

抽 ETM，FRAM， DEF 等----这个我们提供相应的脚本，由用户手动完成，不利用 makefile 自动做了，这个不是经常需要跑，用户手动更加可控，release 数据的时候出 bug 了的话真的很烦的

tags.sh 快速定位变量在哪里声明的-----参见 wiki

是的，看上去好像太简单了， 没有一点新意。

<完>

2016/01/12