最近在 ZTE onsite 折腾 ICC flow， 翻了下 makefile 文档，可以说蛮巧妙的解决了一个一直以来大家未曾解决的问题 ------ 版本控制与 makefile 的依赖 和谐共存，与大家探讨下

makefile 的设计初衷或者说核心是用来管理一项有依赖关系的工程，即该工程明确规定了先做什么，后做什么。

没有依赖关系的 makefile 不是 makefile，相当于一个 shell 脚本而已，即你用 shell 脚本实现代码是一模一样的。但如果说你用 shell(或者其他语言)去实现 makefile 的依赖功能，那还是相当难的，毕竟这是先贤设计出来的核心功能。

我们来看两套例子：

huabin 留下的 makefile	PCD chenxue 的 flow
做了版本控制 通过 make 的时候输入 input cell 和 output cell， 比如 make 12_fp_to_21_place.place_opt (input cell = 12_fp, output_cell = 21_place,以下类推) make add_bound_fp_to_add_bound_place.place_opt 但代价是丢掉了 makefile 的依赖 导致如果要从 place 跑到 route 的话需要 make fp_to_place.place_opt make place_to_cts.cts make cts_to_postcts.postcts make postcts_to_route.route 这样敲这么多，特别是当你的 cell 名字变了的话（比如跑另外一版）或者想要从 cts 跑到 route 的话，那你又要对上的 target 作出修改， 违背了代码设计的 **closed for modifications** 原则 参见 《head first design patterns》 一书	没有做版本控制，由 block owner 自己管理， 顶层文件 place.tcl cts.tcl postcts.tcl 等与其他与项目相关的文件是 link 的，用户不能修改，好处: block owner 实时得到更新(link 的）不需要重新生成。 与之同时每一步留了一个文件(接口）给用户修改(用户自己目录下)，方便用户私人定制，比如哪里加 bound 啊， cts 做 tree 怎么做啊，以及 input cell 和 output cell 等。 这样用户每跑一版都要记得去该文件里修改 open_mw_cel 用那个 cell 以及 save_mw_cel 存成什么 cell，同样的 violate **closed for modifications**

现在来回顾一下 makefile 的核心功能：举个栗子，比如今天白天我好不容易整好了 floorplan，想利用晚上的时间从 place 一直跑到 route，明早回来看结果，
那我们只需要敲下 make route， 然后 makefile 发现 哦， floorplan 已经做好，那就按 place---> cts ----> postcts ----> route 顺序开跑吧
第二天，白天我 debug 发现 timing 不好，得调调 tree， 好不容易搞了一天把 float pin 设置弄好了，晚上跑一下 看看结果如何，那我们只需敲下
make -t place
make route 然后 makefile 就开始按 cts ----> ----> postcts ----> route 开跑
这就是依赖的强大功能。。。

这就是 init fp place cts postcts route 这些 target 或者说成用户的 interface 简单，定死----- **closed for modifications** (我希望是一万年不变）的好处，比如 Linux 下安装各种软件就是千篇一律的 make install 就 OK 了，比 win 平台的“下一步 下一步”还简单

但定死 target 之后带来的问题是版本控制怎么实现?
我们这里实现起来相当简单（细节见附件，代码相当少）, 大体思路 敲下 make XXX 后由 make 向用户(interactive 的方式)询问你要从那个 cell 开始跑， 以及你要存成什么样的 cell，举个列子，我们以那种日期作为版本号的话 从 place 跑到 starrc 抽 spef
那我们敲下 make starrc， make 会向你所要 input cell 和 version，当你输入 fp_0111 和 _0111 的话那么 cell 的演变顺序类似各位用 huabin 的 makefile 那样
fp_0111 -----> place_0111 -----> cts_0111 -----> postcts_0111 ----> route_0111 ---> run_starrc 抽 spef （版本号作后缀使用）
当然你可以不提共 input cell 和 version， 那么将使用默认的设置，比如你省略版本号，那将

是
fp_0111 ----> place -----> cts -----> postcts ------> route ---> run_starrc 抽 spef，(如果你觉的覆盖以前存在的 cell 没问题的话)
那么现在重要的事情来了，我们这样跑的过程不需要用户改任何脚本 也不是重新生成（等效于更改）新的脚本，强大吧，而且我们的脚本相当简单直观易维护哦， 当然用户的私人定制还是要用户弄好的。 比如 我想看看 keepout margin 设置成不同的值的结果，那么你可以在 place_setting.tcl 里面设置好（开放的，在用户自己目录下，用户私人定制的地方）那么你可依这样写

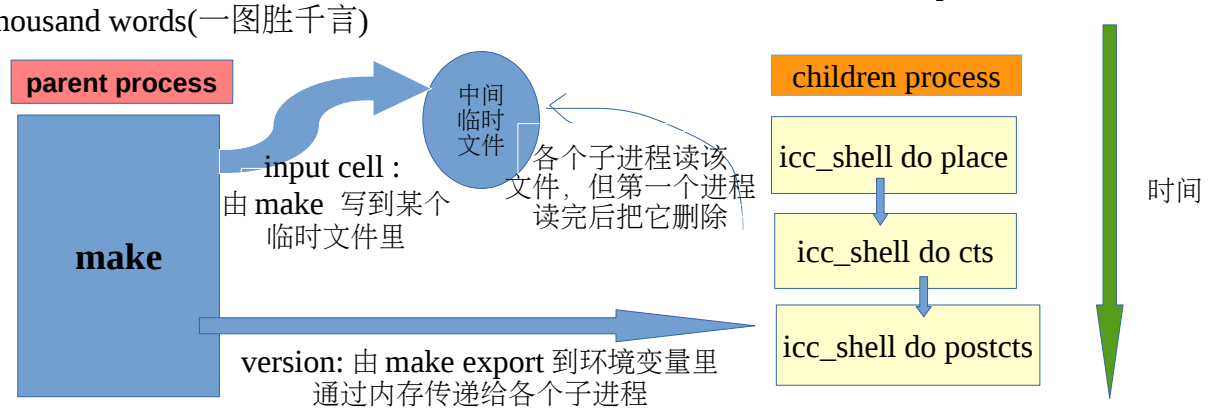
if { $version == “keepout0.5” } { set_keepout_margin …..... }
if { $version == “keepout1.0” } { set_keepout_margin …..... }
完整的保留了代码修改日志，方便日后追踪， 毕竟光一个 version = keepout0.5 描述能力是有限的,至少没有说明在那些 cell 上设了 keepout， AOI？ NAND？ 上下设？ 左右设？，

再来看一个有(gao)难(bi)度(ge)的例子，还是上面那两个不同 keepout margin 版本，我们可以同时并行跑， 开两个 terminal tab

make route (跑 keepout0.5 的)	make route (跑 keepout1.0)
make 索要 input cell (fp_0111) version (_keepout0.5 之后 cell 演变顺序为 fp_0111 --> place_keepout0.5 ----> cts_keepout0.5 ----> postcts_keepout0.5 ----> route_keepout0.5	make 索要 input cell (fp_0111) version (_keepout1.0 之后 cell 演变顺序为 fp_0111 --> place_keepout1.0 ----> cts_keepout1.0 ----> postcts_keepout1.0 ----> route_keepout1.0

两个 job 吃的是相同的 相同的 相同的 一套文件，但因为进程不一样而分道扬镳，互不干扰。

说了这么多，来看一下具体是怎么实现的，废话不说，直接上图，a picture is worth a thousand words(一图胜千言)



核心就是如何传递 version 和 input cell， 这也是大家一直在努力的吧，具体步骤如下：
1. make 向用户询问你要从那个 input cell 跑，以及后续存 cell 的版本号 version，用户不给将使用 default 值
2. make 将 version 装载到自己的环境变量里，那么各个子进程都将继承该环境变量(涉及操作系统知识，可翻阅<modern operating systems>一书)，另外则将 input cell 写到某个独一无二的临时文件里
3. icc_shell 子进程起来后，将接收到 version，同时读取那个含有 input cell 的临时文件，第一个 icc_shell 子进程读取之后将该临时文件删除，那么后续的 icc_shell 子进程则读不到该文件

比如你从 place 跑到 postcts，各个阶段的 icc_shell 的关键状态如下表：

		place	cts	postcts
临时文件		存在	不存在	不存在
input cell		从临时文件里取	空	空
version		$version	$version	$version
copy mw cel to current mw cel		如果 input cell 为空(用户放弃输入)则 current_mw_cel =fp$version 否则 copy_mw_cel -from $input_cel -to place$version(current_mw_cel)	input cel 为空, 显然这个阶段该用前一步做好的 place$version cell copy_mw_cel -from place$version -to cts$version(current_mw_cel)	input cel 为空, 显然这个阶段该用前一步做好的 cts$version cell copy_mw_cel -from cts$version -to postcts$version(current_mw_cel)
打开 cell		open_mw_cel $current_mw_cel	open_mw_cel $current_mw_cel	open_mw_cel $current_mw_cel
做完之后存		save_mw_cel	save_mw_cel	save_mw_cel

最后, 说了这么多。。。大家关心的来了，使用起来复杂吗? So easy!!!

结合 makefile 常用的 3 个选项 -i -n -t ， 部分演示如下,

-i 选项 ： 忽略跑得过程的错误，makefile 的默认机制是跑得过程中出错的话就不往下跑了 因为后续的步骤依赖前面的，前面的数据出错了，后面的结果也十有八九是错的，除非你能确定这些是假错，那就用 -i

-n 用的最多，又叫 --just-print 顾名思义，就是预览你要跑的命令(不是实际开跑)

```
pengmingguo@belogin4: ~ $ make -n starrc
# ======= place =========
bsub -Ip /pub/tools/synopsys/icc_2014.sp2/bin/icc_shell  -f ./scripts/place.tcl | tee log/place.
tcl
check_log.pl place
touch place
# ======= cts =========
bsub -Ip /pub/tools/synopsys/icc_2014.sp2/bin/icc_shell  -f ./scripts/cts.tcl | tee log/cts.tcl
check_log.pl cts
touch cts
# ======= postcts =========
bsub -Ip /pub/tools/synopsys/icc_2014.sp2/bin/icc_shell  -f ./scripts/postcts.tcl | tee log/post
cts.tcl
check_log.pl postcts
touch postcts
# ======= route =========
bsub -Ip /pub/tools/synopsys/icc_2014.sp2/bin/icc_shell  -f ./scripts/route.tcl | tee log/route.
tcl
check_log.pl route
touch route
# ======= starrc =========
run starrc.sh cmax 125 route
```

醒目的红字显示当我 make starrc 的时候是从 place 跑到 starrc， 但如果我 place 已经做好了，不需要再跑了，那就用-t 告诉 makefile 说到 place 已经做好了， 它会懂你的

```
pengmingguo@belogin4: ~ $ make -t place
touch place
pengmingguo@belogin4: ~ $ make -n starrc
# ======= cts =========
bsub -Ip /pub/tools/synopsys/icc_2014.sp2/bin/icc_shell  -f ./scripts/cts.tcl | tee log/cts.tcl
check_log.pl cts
touch cts
# ======= postcts =========
bsub -Ip /pub/tools/synopsys/icc_2014.sp2/bin/icc_shell  -f ./scripts/postcts.tcl | tee log/post
cts.tcl
check_log.pl postcts
touch postcts
# ======= route =========
bsub -Ip /pub/tools/synopsys/icc_2014.sp2/bin/icc_shell  -f ./scripts/route.tcl | tee log/route.
tcl
check_log.pl route
touch route
# ======= starrc =========
run starrc.sh cmax 125 route
```

实际敲下 make starrc:

```
pengmingguo@belogin4: ~ $ make starrc
Please type your input cell, you can Press Enter directly to pass this step
Waiting for you for 1 minutes, or default action (pass) will be taken
Input cell = place_keepout0.5

Please type your version, you can Press Enter directly to pass this step
Waiting for you for 1 minutes, or default action (pass) will be taken
version = _keepout0.5^C
```

Easy 吧

以一条软件设计原则结尾

Open-Closed Principle： open for extension, but closed for modification.

<完>

2016/01/10