用户指南

vtags 是一款在 gvim 下实现类似 verdi 的信号追踪、显示拓扑等功能的插件。vtags 插件完全使用 python 实现，目前实现功能包括信号追踪、宏定义追踪、显示模块拓扑、快速打开文件、保存和打开 gvim 快照、添加断点等功能。

支持功能和快捷键:

快捷键	功能
gi	进入子模块
gu	回到上层模块
<Space><Left>	追信号源，或宏定义
<Space><Right>	追信号目的
<Space><Down>	回退
<Space><Up>	向前
<Space> + v	显示策略，和展开收回策略条目
<Space> + c	添加记录点
<Space> + b	添加基本模块
<Space> + d	删除记录点或基本模块
<Space> +h	固定当前窗口
<Space>	快速访问
<Space> + s	储存快照
gvim/vim	加载快照

详细信息看《二：开始使用及支持功能和快捷键》

注意：

（1）在 code 目录下通过 vtags 生成 vtags.db 后，第一次打开 verilog code 时需要编译生成的 database，所以第一打开 code 比较慢，之后打开 code 就会非常迅速。

vtags 安装：

（1）下载插件代码，解压并复制到安装路径下。(vtags 无需安装，解压即可)

tar -zxvf vtags-1.20.tar.gz // 解压 vtags 压缩包，假设目前在用户根目录下，则安装路径为： ~/vtags-1.20/

（2）在 linux 配置文件中添加别名。

1）使用 cshrc 的用户：

~/.cshrc 中添加：alias vtags 'python ~/vtags-1.20/vtags.py'

使用 bashrc 的用户：

~/.bashrc 中添加：alias=vtags 'python ~/vtags-1.20/vtags.py'

2）source ~/.cshrc 或 source ~/.bashre 使设置生效。

注： 使用别名的目的是为了在任何位置都可以，直接输入 vtags 来调用安装目录下的 vtags.py 脚本。 所以能达到这个目的的方式都可以，也可以直接通过全局路径调用脚本。

（3）在 vim 配置文件中加入插件。

~/.vimrc 中添加： source ~/vtags-1.20/vtags_vim_api.vim

注 :在 vtags_vim_api.vim 中除了加载脚本外，还可以修改所有支持操作的快捷键。

(4) 本步骤可选， 可以在 vtags-1.20/vim_glb_config.py 中设置插件的一些全局设置， 你也可以使用 vtags 时生成的局部配置文件 vim_local_config.py 中进行设置，详细配置见下文。

vtags 使用：

一： 生成 vtags.db 数据结构

完成安装后，vtags 的使用和 ctags 类似，首先需要生成对应的数据结构。

（1）cd 到代码所在目录。

（2）执行命令：vtags (得益于前面设置的别名，其实是在代码目录下调用 ~/vtags-1.20/vtags.py 脚本)

注：(1) 执行 vtags 生成数据文件，主要分 3 步。

第一步：找到当前目录及子目录下所有支持的 verilog 文件（各自代码使用的 verilog 后缀可以在配置文件中添加，默认只支持.v 文件，可以在 config 中任意添加支持的 verilog 后缀）

第二步：分析所有 verilog 文件找到所有 verilog module。

第三步：分析所有 verilog 文件找到每个模块中子模块调用信息。

(2) vtags 执行完成后会在当前目录下生产 vtags.db 文件夹，里面存放的是前面分析的数据，以及当前目录 ctags 的局部配置文件，配置详见下文。

二：开始使用及支持功能和快捷键

在生成 vtags 后就可以通过 gvim 或 vim 开始查看对应代码了。

gvim 打开代码后支持的功能和快捷键如下：

gi : go into submodule

使用情形： 在阅读代码时光标处在子模块调用的语句上，使用 gi 快捷键可以进入子模块中。

gu : go upper module

使用情形： gu 作用同 gi 相反，返回到上一级模块。

空格 + 左方向键： trace source

1）如果光标处在信号名上追踪该信号的源。

2）如果光标处在宏名（如`TEST_MODE）上显示该宏的申明（define 行）。

空格 + 右方向键: trace dest

追踪光标所处信号的目的，多个目的存在时调到找到的第一个目的位置，并将其它目的显示在底部的 report 窗口中。

空格 + 向后方向键: roll back

回退到上一个记录点(一般发生一些需要改变光标位置或窗口的时候插件会自动添加记录点)。

空格 + 向前方向键: go forward

和 roll back 对应，当回滚后恢复到前面的记录点。

空格 + v: view sidebar

功能一： 显示左侧导航栏，主要显示 3 方便信息：

1. 如果光标处在有效模块中，显示当前模块拓扑结构（即类似 verdi 左侧的模块层次结构)。

2. 自己添加的 checkpoint（看 “空格 + c” 操作描述)。

3. 显示所有被认为是基础模块的模块名（看 “空格 + b” 操作描述)。

功能二：展开和收回导航栏显示条目

1. 如果光标在导航栏，模块拓扑结构上，执行 “空格 + v” 可以继续展开或收回对应行模块的拓扑。

2. 在 “CheckPoint" 行可以展开或收回所有 checkpoint。

3. 在 “BaseModule” 行可以展开或收回所有显示的基本模块名。

空格 + c: add checkpoint

将光标位置储存起来，添加到 sidebar 的 checkpoint 栏中，可以在任何时候讲光标移动到 checkpoint 行使用 “空格” 键直接跳转到添加 checkpoint 位置，也可以通过 “空格” + d 删除一个 checkpoint

空格 + b: add basemodule

basemodule 是指一些调用很多的模块，比如 dff, latch 等，如果在显示 topo 时全部显示，会像 verdi 一样拓扑很长不便于阅读。

将光标所在模块添加到 sidebar 的 basemodule 栏中，标移动到 basemodule 行使用 “空格” 键直接跳转到添加 basemodule 位置，也可以通过 “空格” + d

删除一个 basemodule。

空格 + d: delete

当光标在 sidebar 上的 checkpoint 或 basemodule 上时删除对应的 checkpoint 或 basemodule。

空格 + h: hold

vtags 在操作过程中会需要切换或打开新的窗口（如追踪到新的文件，topo 打开新的文件等）这个打开的最大窗口数量在配置文件中有限制，当已经打开的窗口达到最大值后，会自动关闭较老的窗口来打开新的文件。

而 “空格 + h” 操作就是 hold 住光标所在窗口使得，除非手动关不不然 vtags 不会自动关闭该窗口。

空格： quick access

当光标在 sidebar 或 report 窗口中时，按 “空格” 可以快速访问对应的文件。

如:

（1）在 topo 或 basemodule 上会打开对应模块。

（2）在 checkpoint 上时跳转到 checkpoint 添加的位置。

（3）在 report 窗口光标放在 trace 结果的行时，按 “空格” 直接跳转到 trace 的位置。

空格 + s： save snapshort

将当前 gvim 当前打开窗口等创建快照，关闭后，下次使用 cd 到当前 vtags.db

文件夹，使用 “vim/gvim” (注意：不要其他参数)，会提示是否打开快照，输入 y 后，就会打开上次存储的快照。

gvim/vim： reload snapshort

cd 到当前 vtags.db 文件夹，使用 “vim/gvim” (注意：不要其他参数)，会提示是否打开快照，输入 y 后，就会打开上次存储的快照。

三： vtags vim config

配置文件：

vtags 的配置可以在两个位置配置，全局配置和局部配置。

一、全局配置：

在 vtags 的安装目录下，在 vtags_glb_config.py 中添加配置。该配置是所有使用 vtags 的默认配置。

二、局部配置：

在 code 文件夹使用 vtags 生成 vtags.db 后在 vtags.db 中会生成一个局部配置文件 vtags_local_config.py。

当使用 gvim 打开文件时，会首先选择局部配置中的配置，如果没有局部配置才会使用全局的默认配置。

可配参数：

frame_window_width = 30

设置 gvim 窗口最左侧 sidebar 的宽度，该窗口主要显示模块拓扑，checkpoint 和

basemodule

report_window_height = 10

设置 gvim 窗口最底部 report window 的高度，该窗口主要显示提示信息和信号追踪结果。

max_open_work_window_number = 1

当使用 vtags 机制（如追踪到新的文件，topo 打开新的文件...）打开新的文件时，打开窗口的最大数量，当已经打开的窗口数量到达该值时，会自动关闭一个最早打开的文件窗口在打开新的文件。

max_roll_trace_depth = 1000

使用 “空格 + 前/后方向键” 能够倒退/前进的最大深度。

max_his_check_point_num = 10

使用 “空格 + c” 能够保存的最近最多 checkpoint 个数。

base_module_threshold = 50

在生成 code 的数据文件 vtags.db 时，对所有模块如果该模块在整个文件夹下的 code 中被实例化的次数到达该参数的数量时，将该模块设置为 basemodule 当使用 “空格 + v” 显示模块拓扑时不显示该 basemodule 模块的调用细节，只显示模块名+实例次数。

support_verilog_postfix = ['v']

vtags 将当前目录下所有后缀在 support_verilog_postfix 中的文件认为可识别的 verilog 文件。默认只有 ‘v’ , 可以自己添加, 如：support_verilog_postfix = ['v', ‘V’ , ‘sv’]等

debug_mode = False

该模式是开发过程用于调试的选项，打开后会在 gvim 运行过程中，在 vtags.db 中生成运行过程的 log，会减慢执行速度，不建议打开。