vtags user guide

vtags is a gvim plugin, it's function similar like verdi's trace verilog signal function, this plugin writed by python, currently support: trace signal， macro define trace，show module's topology， quick access， save and opend a snapshort, add checkpiont etc.

support function and shortcut key:

shortcut key	function
gi	go into submodule
gu	go upper module
<Space><Left>	trace source
<Space><Right>	trace destination
<Space><Down>	roll back
<Space><Up>	go forward
<Space> + v	view sidebar
<Space> + c	add checkpoint
<Space> + b	add basemodule
<Space> + d	delete
<Space> +h	hold cur window
<Space>	quick access
<Space> + s	save snapshort
gvim/vim	reload snapshort

detial see 2 support feature and shortcut key

attention :

（1）after you generate database use command "vtags" at verilog code dir, the database file will compiled at the first time when you use gvim/vim open a verilog file, so the first time open code need a little time, after that every time open code file will be very quick.

vtags install

(1) tar -zxvf vtags-1.20.tar.gz // unpack the file to the path you want to install. we assuming to "~/"

(2) in "~/.cshrc" add line: alias vtags 'python ~/vtags-1.20/vtags.py'

or "~/.bashrc" add line: alias=vtags 'python ~/vtags-1.20/vtags.py'

(3) "source ~/.cshrc" or "source ~/.bashrc"

(4) in "~/.vimrc" add line： source ~/vtags-1.20/vtags_vim_api.vim

(5) config vtags-1.20/vim_glb_config.py as it decribe in vim_glb_config.py. (can do it in local_config.py see below)

vtags use

1 generate vtags.db database

when finish install vtags, first need generate vtags database for code you want use.

(1) "cd" to code dir

(2) use command "vtags"

then will generate a vtags.db dir, in the dir include module database , log file, and local config files.

2 support feature and shortcut key

after you finish generate vtags, you can use gvim/vim open verilog now, and can use shortcut key below to use the function in vtags, such as trace signal, open topology.

gi : go into submodule

when read verilog code use gvim/vim, if your cursor on a submodule call line, you can use "gi" to enter the submodule.

gu : go upper module

"gu" 's function is opposite to the "gi", go to the upper module of current cursor module.

<Space><Left>: trace source

1）if your cursor on a signal, then just trace the signal's source.

2）if your cursor on a macro name, then just go to the macro define.

<Space><Right>: trace destination

trace the cursor word's destination, if have more than one dest, then cursor go to the first dest , and all dest will show in the report window at bottom of screen, you can go tho other source you <Spece> operation discribed below.

<Space><Down>: roll back

go back to the pre cursor location automatic recording.

<Space><Up>: go forward

contrary to <Space><Down> action.

<Space> + v: view sidebar

1 show the left sidebar, mainly for 3 information:

(1) if cursor current in valid module, show the module's topology(the same to verdi's left sidebar)

(2) checkpoint add by user

(3) all modules treated as base module

2 fold or unfold the sidebar item

(1) if cursor on sidebar and on a topology line, enter <Space> + v again can fold or unfold a submodule's topology.

(2) if cursor on labe "CheckPoint" or "BaseModule" can fold or unfold all the checkpoints or basemodules.

<Space> + c: add checkpoint

this action will save the cursor position and show in sidebar's "CheckPoint" item, any time you want go back to the cursor location, you can but your cursor on the checkpoint and use <Space> action to go direct to the it. you also can use

<Space> + d to delete one checkpoint in sidebar.

<Space> + b: add basemodule

base module is those module instantiation too many times, if all instance shown to the topo will made topo too long to read. so base module will not show every instance in topo, and just show the module name and instance times.

<Space> + b and the cursor word to the base modules, even if the word not a module, the only result is base module instanse will not show in topo one by one.

<Space> + d: delete

when cursor in sidebar's checkpoint or basemodule items, use <Space> + d to delete a checkpoint or base module.

<Space> +h : hold cur window

vtags when need open a new window(such as trace cross modue signal), if current opened window already reach the max open window allow to open(config at vim_glb_config.py or vim_local_config.py). it will close a oldest window and then open the new window.

<Space> + h 's action is hold the cursor window, and then vtags will not close this window untill you close it yourself.

<Space>: quick access

when cursor on sidbar or report window, use <Space> can quick access

corresponding file.

(1) cursor on topo's basemodule or instance, can open corresponding module

(2) cursor on check point, can to to the checkpoint location

(3) cursor on report window and on trace result, use <Space> can to to the trace result line.

<Space> + s: save snapshort

create snapshort for current opened gvim windows, next go to the dir save snapshort and use "gvim/vim"(without other parm), will ask you weather open the snapshort if you enter "y/yes" will open the snapshot.

gvim/vim：reload snapshort

if you use <Space> + s save snapshort before, at the same dir, use "gvim/vim"(without other parm), will ask you weather open the snapshort if you enter "y/yes" will open the snapshot.。

3 vtags vim config

config file：

vtags's config file at two position, global config and local config. when gvim open file and valid open vtags, will use local config first, if not local config then will use the global config.

(1) global config：

at the vtags's install dir, we can config vtags in vtags_glb_config.py.

(2) local config：

at code dir use cmd: "vtags" will generate vtags.db folder, in this folder has a file vtags_local_config.py we can config vtags in this.

config parms：

frame_window_width = 30

set the gvim left sidebar width, this sidebar used to show topo checkpoint and base module.

report_window_height = 10

set gvim bottom report window height, this window used to show the report and signal trace result.

max_open_work_window_number = 1

when vtags need open new windows(such as : quick access, trace crose module ...), this number is the max window num opened, if opened window already reach this number, vtags will close a old window and then open new file.

max_roll_trace_depth = 1000

use <Space> <Down/Up> max history depth.

max_his_check_point_num = 10

max checkpoint can rember, added by <Space> + c

base_module_threshold = 50

when code generate vtags.db, if some module instantiate times bigger than base_module_threshold will auto set it to be base module, when show the topo will not show the base module instance one by one.

support_verilog_postfix = ['v']

all the verilog code file postfix used, default only "v", you can add your self postfix like support_verilog_postfix = ['v', 'your_postfix0', 'your_postfix1' ...], all the file below code dir has postfix in support_verilog_postfix will treat as verilog code.

debug_mode = False

this mode used for develop, will print lots of report, suggest set to False.