vtags-2.0 的改变

vtags-2.0 几乎完全重构了 vtags-1.x 的所有功能和实现，无论在功能、速度和用户体验上都有了质的飞跃，主要的改变体现在：

1. 极大的扩充了对 verilog 多种语法的支持.

2. 实现完善快速的数据库动态更新机制，vtags 会随着设计改变动态跟新.

3. 使用了新的识别算法，极大的提升了识别的准确性和速度.

4. 使用了去中心化和用时加载技术，极大的节省了 vtags 运行占用的资源.

5. 提供了 function 和 API 方便用户在不打开 vim 时使用 vtags 的 function 和数据库.

6. 对 vtags 的加载进行了优化, 使的对 vim 打开非 vtags 支持文件几乎没有延迟.

1 vtags 无法识别的 verilog 语法

（1）无法识别 "/*...*/" 的注释形式。

（2）无法识别别编译选项如： 'ifdef xxx ... 'else ... 'endif

2 命令行 function 集

vtags-2.0 中提供在命令行访问 vtags 产生的 database（放在 vtags.db 文件夹中）中一系列 function（显示拓扑，显示模块调用路线，打开模块...），以方便用户在不打开 vim 时使用 vtags 数据结构。

主要 function 如下表：

function	描述

open_module_file(module_name)	使用 gvim 打开 module_name 所在文件并将光标跳到 module 定义位置。
print_module_filelist(module_name)	递归的显示输入模块以及所有子模块的文件路径, 可用来单独提取某个模块的 RTL code.
print_module_io(module_name)	格式化的打印输入模块的 io 信息。
print_module_topo(module_name, depth, mask)	用来打印输入模块拓扑结构, depth 用来指定打印深度, mask 用来屏蔽例化次数过多的基础模块(比如 dff 等), 当例化次数大于 mask 时在打印 topo 时只显示数量不显示具体实例名(对指定 vtags db 中默认指定为 basemodule 的模块也只显示数量)
print_module_trace(to_module, from_module)	在整个 rtl 中查找 to_module 的调用关系, 显示所有从最上级 module 到 to_module 整条调用轨迹。 如: 执行: vtags -func -db ... print_module_trace(lru) 显示结果: cpu -> core -> l2 -> lru 表示 lru 的一条调用路线是从 cpu 到 core 最后到 lru

3 vtags python api

vtags-2.0 中提供了供用户调的 vtag python api, 以方便大家使用 vtags 的数据结构和函数实现自己特点应用的脚本。

提供的 api 在 vtags 安装目录下 vtags_custom_api.py

使用方法见 vtags 安装目录下的 custom_api_example.py