# Verilator使用教程

# 简介

Verilator是一款开源的支持Verilog和System Verilog仿真工具，它支持代码质量检查等功能，能够将给定的电路设计（由Verilog或System Verilog编写）编译成C++或者System C的库等中间文件。使用者可以通过C++编写testbench程序，来调用这些库，以完成Verilog的仿真。

# 安装

编者提供了多种安装方式，感兴趣的同学可以尝试不同的方法进行安装。

## 通过文件中安装包进行安装

编者已经将Verilator的仓库放在了文件夹下，可以通过下面的命令进行安装。值得注意的是，如果不能通过编译，可能是因为缺少一些库文件，请自行根据错误信息进行debug。

#通过下面的命令进行安装，首先确保你当前的位置在解压后的文件夹中

unsetenv VERILATOR\_ROOT # 如果这条命令出现了错误，那么请忽视这条命令

unset VERILATOR\_ROOT # 清理环境变量

cd verilator # 进入verilator的仓库

autoconf # 生成配置文件

./configure # 运行配置文件

make # 进行编译

sudo make install # 安装

#检查安装是否成功

verilator --version

#输出下面的信息表示安装成功(版本信息可能不同）

Verilator 5.012 2023-06-13 rev v5.012-34-g4f13c4d1b (mod)

## 通过官网的指南安装

Verilator的官网地址是[https://verilator.org/guide/latest/](inline)。在第一章节的Installation小节可以找到安装指南，具体内容如下：

#Git Quick Install

#安装依赖

sudo apt-get install git help2man perl python3 make autoconf g++ flex bison ccache

sudo apt-get install libgoogle-perftools-dev numactl perl-doc

# 仅ubuntu用户被要求执行下面的命令

sudo apt-get install libfl2 libfl-dev zlibc zlib1g zlib1g-dev

#拉取仓库，以下两条命令选择一条进行拉取。

#其中gitee是国内网站，比较容易建立链接。

git clone https://gitee.com/mirrors/Verilator.git #gitee仓库

git clone https://github.com/verilator/verilator #github仓库

#每次编译之前需要配置一些环境变量

unsetenv VERILATOR\_ROOT # For csh; ignore error if on bash

unset VERILATOR\_ROOT # For bash

cd verilator # 进入verilator文件夹

autoconf # 生成配置文件

./configure # 进行配置

make # 编译代码

sudo make install # 安装

#检查安装是否成功

verilator --version

#输出下面的信息表示安装成功(版本信息可能不同）

Verilator 5.012 2023-06-13 rev v5.012-34-g4f13c4d1b (mod)

# 使用

你可以在Verilator的官网找到详细的使用指南和例程，这里编者也提供一个例程供大家学习。这个例程就在count文件夹中。

## 文件结构

首先介绍这个例程的结构。其中count.v是verilog文件，count.cpp是使用C编写的testbench文件。至于Makefile文件，它是一种用来进行自动编译的文件，感兴趣的同学可以了解一下。

count

|--Makefile

|--vsrc

| `--count.v

`--csrc

`--count.cpp

## 编译方法

这里提供两种编译方法。

### 通过在终端中输入命令进行编译

首先确保你在count目录下，接着执行下面的命令进行编译。这种编译方法的好处是简单直观，缺点是每次编译都要输入大量的参数。

verilator --top-module count --cc --build --Wall --exe --trace vsrc/\*.v csrc/\*.cpp

# 简单介绍一下参数的作用

# --top-module：这个参数的作用是确定顶层模块，如果你有多个模块的话，需要使用这个参数确定顶层模块

# --Wall：开启全部警告，这个参数将尽可能多的输出编译时产生的警告信息，以帮助使用者debug

# --trace：这个参数的作用是打开追踪功能，如果你想要在cpp程序中生成波形信息，需要添加这个参数。

### 通过Makefile进行编译

编者在文件夹中提供了一个Makefile文件供大家学习和使用。如果你想要学习Makefile工具，请在网上搜索相关的资料进行学习；如果你只是想要使用该工具，你可能需要注意Makefile中第一行的变量。

# 第一行的变量确定了顶层模块

# 你可以通过将等号后面的字符修改成你自己的顶层模块名称来进行快速编译

# 值得注意的是，如果你在使用过程中发生了一些错误，需要你独自去解决

TOPNAME = count

### CPP文件讲解

为了大家能够更好的理解和使用C语言编写Testbench文件，我将CPP文件添加了注释供大家参考。

#include <verilated.h > // verilated.h库文件

#include "verilated\_vcd\_c.h" // 波形追踪的库文件

#include "Vcount.h" /\* 编译生成的中间库文件，

如果你的顶层模块名为xx则该库函数应为Vxx.h\*/

VerilatedContext\* contextp = NULL;

VerilatedVcdC\* tfp = NULL;

static Vcount\* top; //同样的V后面的字符应该被替换为你的顶层模块的名称

/\*执行函数，在这个函数中，时钟信号将翻转，并更新其他信号状态\*/

void step\_and\_dump\_wave() {

top->clk = !top->clk; /\*翻转时钟信号，注意，该语句应该在最前面，

否则可能会输出错误的波形\*/

top->eval(); // 更新模型状态，通过输入信号的改变来更新其他信号的值

contextp->timeInc(1); /\* 将时间推进1单位，这条语句和生成波形有关，

通过这条语句，你可以在任意地方改变信号的值，

而不只是在上升和下降沿\*/

tfp->dump(contextp->time()); //将信号信息输入波形文件

}

/\*初始化函数\*/

void sim\_init() {

contextp = new VerilatedContext;

tfp = new VerilatedVcdC;

top = new Vcount; //同理，需要修改count为你当前的顶层模块名

contextp->traceEverOn(true); //开启波形追踪

top->trace(tfp,10); //波形追踪的深度

tfp->open("count.vcd"); //count.vcd是输出的波形文件的名称

}

/\*关闭函数，用来关闭已经打开的指针\*/

void sim\_exit() {

step\_and\_dump\_wave();

tfp->close();

}

int main() {

int i = 10;

sim\_init(); // 初始化

top->rst\_n = 0; // 将复位信号拉低

while(i != 0) { // 通过循环执行10次执行函数，时模型中的信号复位

step\_and\_dump\_wave();

i--;

}

top->rst\_n = 1; // 将复位信号拉高

while(i < 10000) { // 通过循环执行10000次执行函数

i++;

step\_and\_dump\_wave();

}

sim\_exit(); // 关闭指针

}

## 观察波形

我们可以使用GTKWave工具来观察生成的波形进行调试。它的官网如下：

[https://gtkwave.sourceforge.net/](4m85wt)

### 安装GTKWave

我已经将GTKWave的仓库放在了文件夹中，像安装Verilator一样，我们也需要通过一些命令来进行安装。

# 首先确保你的位置在gtkwave文件夹下

./configure

make

sudo make install

### 通过GTKWave观察波形

进行编译之后，在count文件夹下会生成count.vcd文件，这个文件就是生成的波形文件。你可以通过输入以下命令使用GTKWave进行观察。

# 通过命令行命令进行观察

gtkwave count.vcd

#此外，你也可以使用Makefile工具进行观察

make wave

本教程到此结束，希望大家能够顺利使用Verilator进行Verilog的仿真。此外，强烈推荐大家通过阅读Verilator官网的使用手册来学习Verilator更多的功能和例程。