TJ-FPGA仿真平台介绍

2018年12月10日星期一

目 录

[1. 背景 3](#_Toc1951711051)

[2. 仿真平台特性 3](#_Toc866134375)

[3. 仿真平台使用方法 3](#_Toc433170815)

[3.1 平台目录 3](#_Toc1818371266)

[3.1.1 可综合代码目录rtl 3](#_Toc2041730359)

[3.1.1.1 举例说明 4](#_Toc1461242435)

[3.1.2 基本验证文件目录tb/btc 4](#_Toc1718520667)

[3.1.2.1 举例说明 5](#_Toc1351270679)

[3.1.3 仿真执行目录runsimlin 5](#_Toc1770825879)

[3.1.3.1 举例说明 5](#_Toc2096550173)

[3.1.4 验证目录vf 6](#_Toc660820046)

[3.1.4.1 目录介绍 6](#_Toc734465558)

[3.1.4.2 扩展性开发 7](#_Toc1601455298)

[3.2 使用方法 7](#_Toc868722189)

[3.2.1 基于vivadoSim的仿真使用方法 7](#_Toc1002590662)

[3.2.1.1 编写rtl代码和验证文件 7](#_Toc1039354230)

[3.2.1.2 修改tc.cfg配置文件 9](#_Toc119900444)

[3.2.1.3 执行makefile脚本 9](#_Toc1904488865)

## 背景

为方便开发RTL-CNN推理加速器，方便组内人员协同工作，共同完成VLSI规模的集成电路设计工作，保证设计的每个阶段都向着正确方向前进，特此开发了TJ-FPGA仿真平台。

## 仿真平台特性

支持模块单元测试

支持多用例测试

支持系统级别验证

支持工程碎文件自动清理

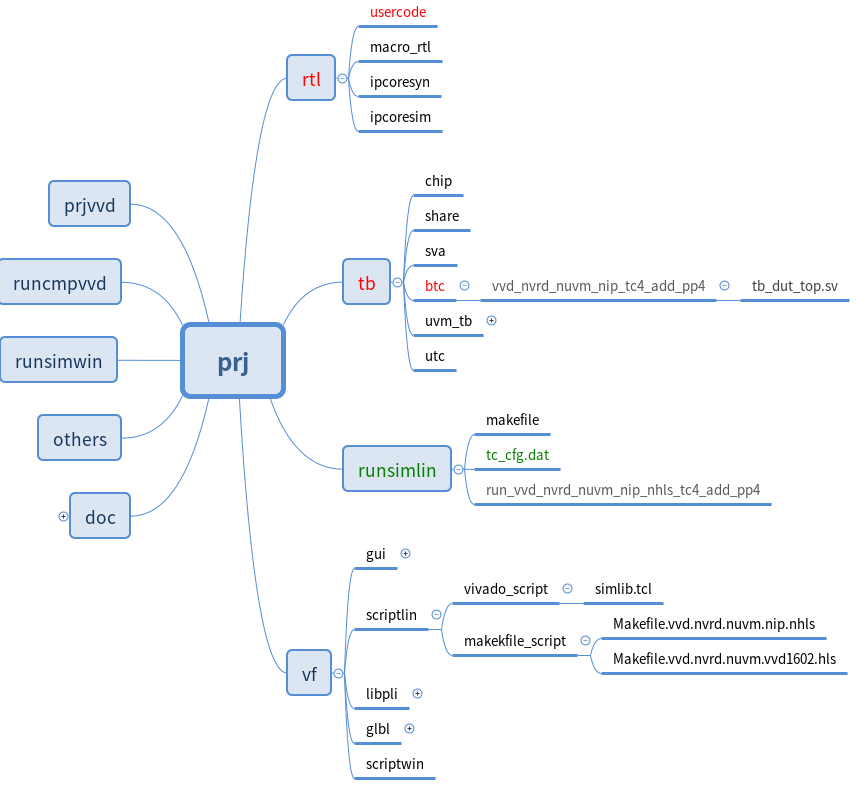
支持git版本管理

支持每日模拟冒烟，并通过jenkins统一管理

## 仿真平台使用方法

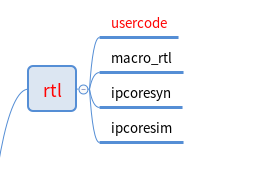
### 平台目录

目录结构主要分为4个，如图所示。



#### 可综合代码目录rtl

该目录下放置可综合的代码，也就是实际要在板子上，经过综合工具能生成电路的代码，具体结构如图所示，usercode表示要实际放的代码文件，比如要设计流水线加法器，那么就可以把流水线加法器替换usercode。

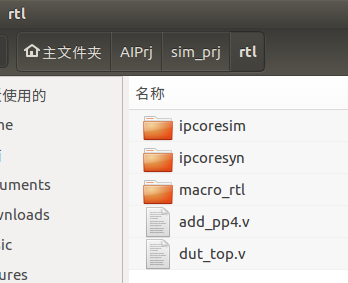


如果代码中有调用xilinx的ip，比如调用fifo，ram之类的ip，若该ipcore要支持仿真，就把它放到ipcoresim目录下，若要支持综合，就放到ipcoresyn目录下。

另外，还有些情况需要设置一些宏参数，我们可以把宏参数统一汇总成一个文件后，放到macro\_rtl目录下。

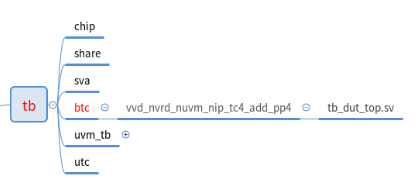
##### 举例说明

比如要设计流水线加法器，直接把写好的文件放到rtl目录下，这里流水线加法器用了2个文件add\_pp4.v和dut\_top.v.



#### 基本验证文件目录tb/btc

目录结构如图所示。



一般的，写完rtl设计，需要再自己写testbench文件，用来测试设计代码是否满足既定的要求，这一点和c/c++中的测试用例非常相似。

验证文件放到tb/btc下面，一般给验证文件起名为tb\_dut\_top.sv这种名字，当然起其他的名字也可以。

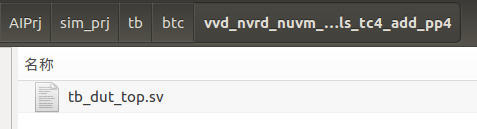
tb(testbench的缩写)

btc(base testbench的缩写，表示基本测试用例)

后续开发中，还会有uvm testcase，这个版本暂时不提供，主要是因为vivado工具不支持uvm。

##### 举例说明

比如要设计流水线加法器的验证文件，该文件名字是tb\_dut\_top.sv，我们把它放到tb/btc目录下。



特别注意：放该文件时，需要先创建一个文件夹，该文件夹的名字是

vvd\_nvrd\_nuvm\_nip\_nhls\_tc4\_add\_pp4，这个文件夹名字很奇怪，这个文件夹的名字，就是我们跑仿真时候的用例名称。

解释：

我们仿真平台的用例支持5个维度的功能，每个维度之间用“\_”来分割。

分析“vvd\_nvrd\_nuvm\_nip\_nhls\_tc4\_add\_pp4”名称。

第一维度：仿真器的类型，取vvd字符串，表示vivado sim，意思是我们选择该用例的仿真器是vivado sim。

第二维度：表示是否用到verdi波形查看工具，取nvrd字符串，表示no verdi，意思是我们选择不用verdi工具。

第三维度：表示是否用用到uvm库，取nvum字符串，表示no uvm，意思是我们不用uvm库。

第四维度：表示用到了哪个版本的ip库，取nip，表示no ipcore，意思是我们不用任何ipcore库。

第五维度：表示是否用到hls，取nhls，表示no hls，意思是这个是我们自己写的代码，不是hls工具生成的代码。

第六维度：用例真实的名字，取tc4\_add\_pp4，这个名字我们可以随意起名，这里的意思是，表示它是第4个仿真流水线加法器的用例。

#### 仿真执行目录runsimlin

目录结构如图所示：



设计和验证文件都写完后，需要进行真实的仿真，进入runsimlin目录下，该目录下有makefile和tc.cfg配置文件。

makfefile文件负责执行make命令，进行仿真命令解析执行等功能。

tc.cfg文件是仿真用例配置文件，用来告诉仿真器，需要仿真那些测试用例。

##### 举例说明

比如要仿真流水线加法器，打开tc.cfg文件后，按照如下格式填写用例名称。

我们知道，

要用的仿真器是vivadosim。因此名字前缀1 vvd

我们不用verdi，因此前缀2,nvrd

我们不用uvm,因此前缀3,nuvm

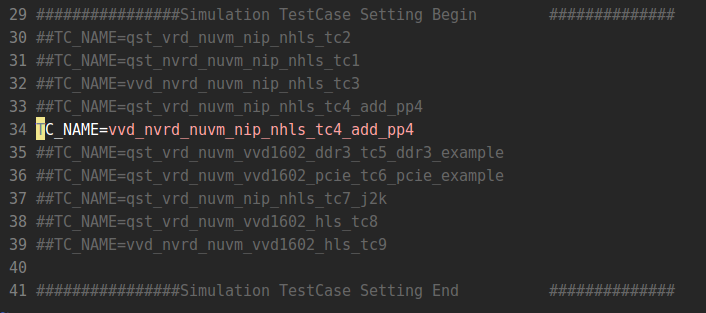
我们不用ip，因此前缀4,nip

我们不用hls，因此前缀5,nhls

我们给用例起名字时tc4\_add\_pp4

每个项之间用\_分割开，

根据这些需求，因此用例全称为：vvd\_nvrd\_nuvm\_nip\_nhls\_tc4\_add\_pp4



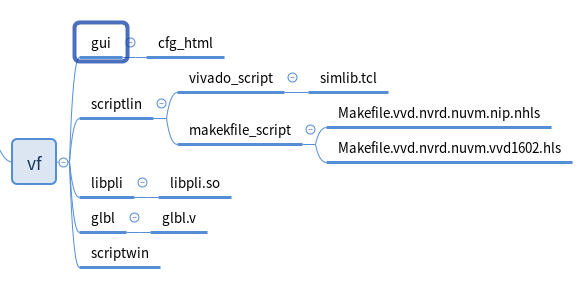
用例格式：

TC\_NAME=xxx

这种方式，如果注销掉这个用例，只需要在行首添加#即可。

#### 验证目录vf

目录结构如图所示，验证目录vf(verification的缩写)，这个目录下存放了仿真执行的脚本信息，以及一些库信息，是仿真工具得以自动化执行的基础，仿真平台开发的新功能以及需求，都是在这个目录下。



说明：一般入门用户不需要关注和修改这个目录，如果在这个仿真平台上开发新需求，需要关注该目录。

##### 目录介绍

gui目录下，存放页面开发需求中相关的html文件，比如仿真环境中，testcase用例的页面配置等等，该需求尚未开发;

scriptlin目录，表示linux脚本目录，很重要，里面的vivado\_script目录下的simlib.tcl文件是自动生成的，用户不用关心。里面的makefile\_script目录很重要，实际建立仿真工程时，新建测试用例中的5个前缀信息，对应了Makfile.xx1.xx2.xx3.xx4.xx5，这5个信息，主makefile会解析测试用例名称，根据名称的5个前缀，在makefile\_script下找到对应的Makefile子文件，进行执行相应配置下的makefile命令，完成仿真功能。

libpli目录存放库;

glbl目录存放xilinx仿真需要的glbl.v文件

scriptwin目录存放windows环境下仿真执行的脚本信息，一般都是bat脚本，预留文件夹，该需求暂未开发。

##### 扩展性开发

比如实际使用过程中，发现目前的5个前缀信息中，都不能很好的支持新的仿真需求，那么可以自定义新的前缀信息，再开发新的Makefile子文件即可，该仿真平台可以满足大部分应用需求。

### 使用方法

以流水线加法器为例，介绍仿真器的具体使用方法，其他开发方式也类似。

#### 基于vivadoSim的仿真使用方法

仿真平台本身不支持编辑器环境，也没有gui界面，用户可以根据自己的喜好使用编辑器，常见的比如vscode，gvim等等，都很好。

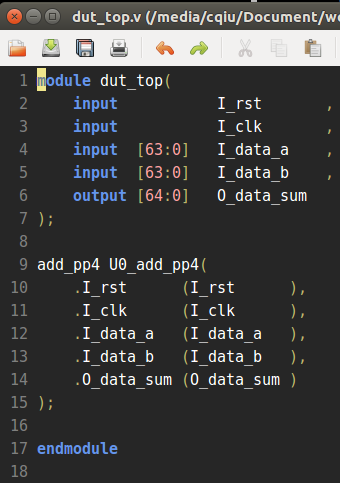
##### 编写rtl代码和验证文件

仿真平台本身不支持编辑器环境，也没有gui界面，用户可以根据自己的喜好使用编辑器，常见的比如vscode，gvim等等，都很好。

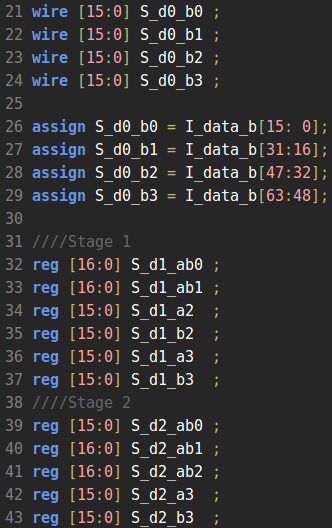
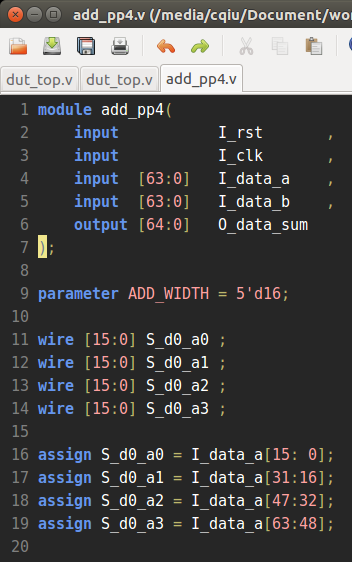
###### 编写rtl代码

dut\_top.v顶层

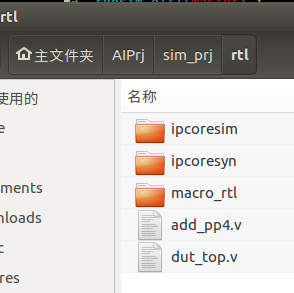
代码见下，这个文件就是实例化了add\_pp4加法器。



add\_pp4.v部分代码

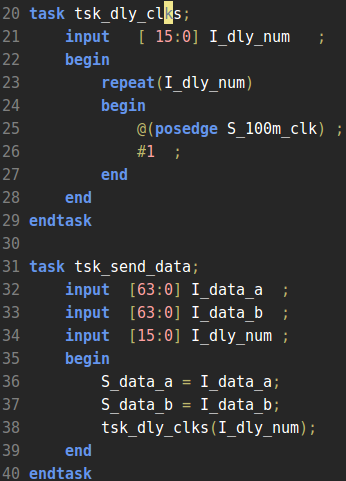
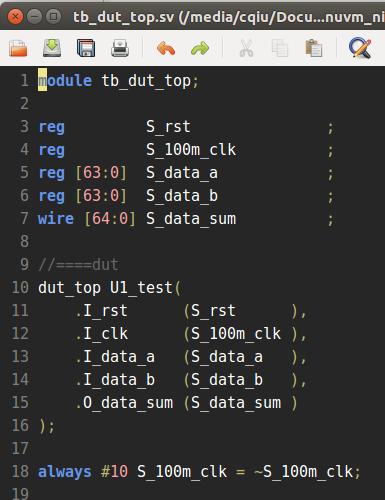


###### 放到rtl目录下



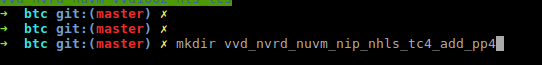
###### 编写testbench代码

代码文件是tb\_dut\_top.sv，部分代码见下



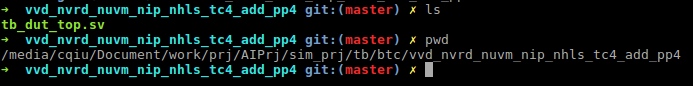
###### 为测试用例起名字，在tb/btc下新建文件夹

进入tb/btc目录下，新建用例名称对应的文件夹，界面和终端都可以。



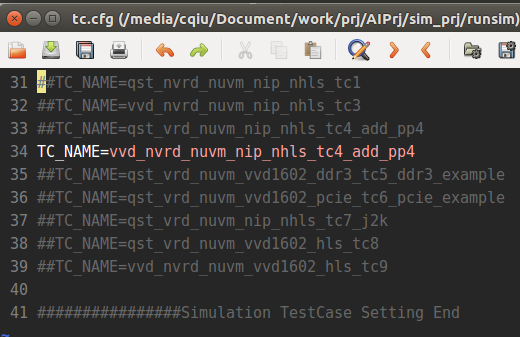
###### Testbench代码文件放到新建文件夹下

代码文件是tb\_dut\_top.sv，放到vvd\_nvrd\_nuvm\_nip\_nhls\_tc4\_add\_pp4文件夹下面，放置完毕后，终端界面上看如下图所示。



##### 修改tc.cfg配置文件

在工程目录下，进入到runsim目录下，打开tc.cfg配置文件，修改内容如下，可以只使能一个testcase，这个testcase的名称就是testbench验证文件中，新起的文件夹名称，改完后退出关闭。



##### 执行makefile脚本

打开终端，在runsim目录下，方可执行makefile脚本。

###### 执行make查看帮助命令

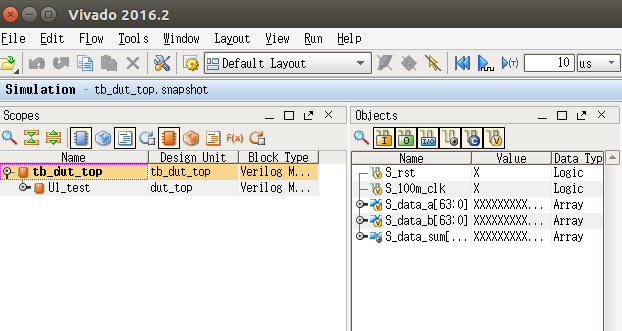


如图所示，执行make后，可以看出目前makefile创建命令并不多，最常用的就是

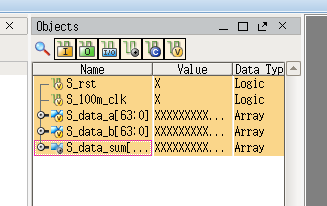
make run，执行仿真。

###### 执行make run进行仿真

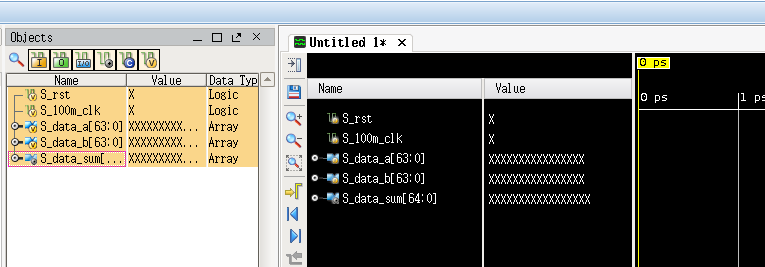
执行make run后，终端会自动打开vivado工具，如图所示，此时可以看到仿真顶层以及一些内部信号。



###### 添加波形信号

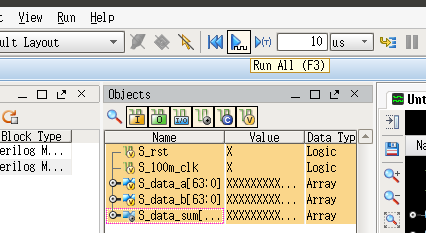


选中内部信号，鼠标右键，在下拉菜单中选择”Add wave to window”，界面如下图所示，此时还没有仿真信号。

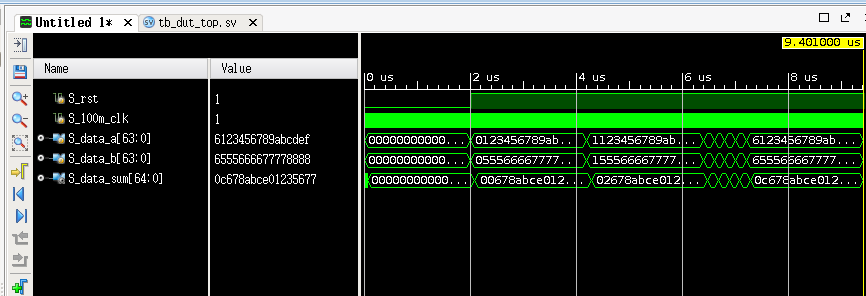


###### 选择蓝色带时钟波形并Run All

选择方法如下图所示



点击后，即可显示仿真后的波形。



至此，基于vivado sim的仿真介绍完毕。

###### 删除仿真工程

执行make clean即可删除vivado sim建立的仿真工程，但是rtl设计，和仿真验证文件是保留的，这种方法可以将仿真平台进行瘦身操作，节约磁盘空间。

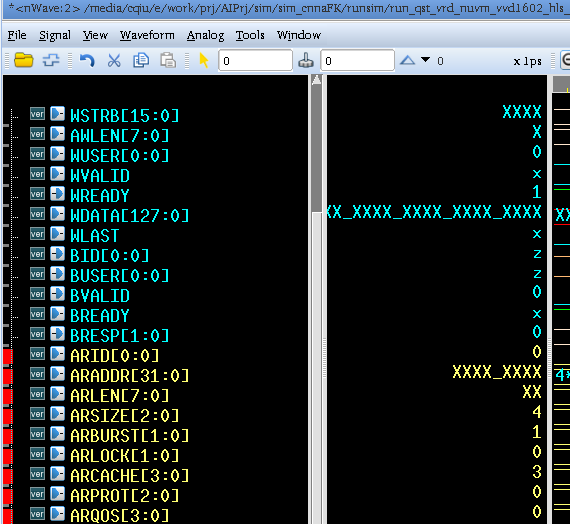
###### 多用例仿真

如果要新建多个测试用例，则只需要新建多个testbench即可，并且再tc.cfg文件下，添加多个TC\_NAME即可。

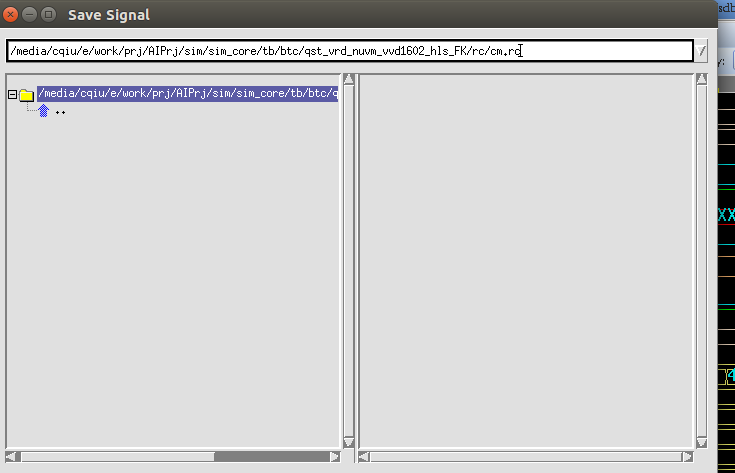
#### 基于Questasim的仿真使用方法

##### Verdi如何保存波形信号

S1：从nWave中，找到File-->Save Singal



S2:将波形信号配置文件，命名为cm.rc，将其放到对应testbench文件目录中，rc子目录下，如果rc不存在，可以手工创建



S3：关闭verdi 再次 make verdi打开后，就自动加载了之前查看的波形信号。

