**基于Systemverilog实现的AltiVec ALU RTL级功能验证平台**

**V1.0**



**Tianjin University**

**School of Computer Science and Technology**

**Lab of VLSI Design and Application**

**No.92 Weijin Road, Tianjin, China**

**December , 2014**

**Lab of VLSI Design and Application confidential proprietary**

**版本说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修改时间** | **修改内容** | **修改人** |
| V1.0 | 2014.2.21 | 增加VFPU指令的约束 | 王捷 |

目录

[一、平台介绍： 1](#_Toc380767871)

[二、平台实现情况说明： 2](#_Toc380767872)

[三、平台使用方法： 2](#_Toc380767873)

[四、参考文献： 4](#_Toc380767874)

## 

## 一、平台介绍：

本平台基于Systemverilog实现Altivec ALU功能模块（VSFX、VCFX、VFPU、PU）的RTL级功能验证，包括”base+random”软硬件自动比对（如图1所示），断言和覆盖率检测（待完善），其中base test对每条指令进行全零数据和全一数据验证，random test根据所设约束产生随机向量，对待测单元进行验证。

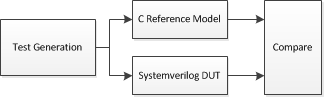


图1.软硬件自动比对验证

平台验证过程如下：首先在Stimulus Generator中根据所设约束产生base或random验证向量（缺省产生10000条验证向量），Driver收集所产生的验证向量，并产生相应的控制信号，这些信号和clock\_reset中产生的时钟、复位信号一同送到DUT和Checker中，在Checker中检测所产生控制信号的正确性，在DUT中例化Altivec ALU功能单元，计算出硬件结果，Monitor将产生的结果和计算结果所用到的输入封装成类，送往Scoreboard，在Scoreboard中调用ISS计算出软件结果，进行软硬件自动比对，同时在Coverage中检测验证覆盖率，如图2所示。



图2.验证平台整体结构

Stimulus Generator和Driver间，Monitor和Scoreboard间通过自定义的altivec类的mailbox进行通信，mailbox是一种fifo数据类型

## 二、平台实现情况说明：

1.支持Altivec ALU全部70条vsfx指令、22条vcfx指令、16条vfpu指令和34条pu指令的base+random软硬件协同验证，在base+random模式下，每条指令都会验证全部base激励，之后会使用随机产生的激励进行验证。在random only模式下，会直接使用随机产生的激励进行验证。

2. Modelsim6.5和Modelsim10.0a使用不同的随机数种子，故两个版本的Modelsim运行本平台的输出结果不同。

3.为了获得更好的Systemverilog的支持效果，推荐使用Modelsim10.0a版本

4.Checker断言检测部分目前仅添加了dut\_busy一条信号的正确性判断逻辑。

5.Coverage覆盖率检测部分根据所检测指令数占总指令数（142条）的比重计算覆盖率。

6.每次只能验证一个Altivec ALU功能单元。

7.根据需要对VFPU指令的输入添加了相应的约束。

8.平台输出（transcript）使用十六进制格式记录数据。

## 三、平台使用方法：

1. 运行该验证平台需要安装Modelsim和mingw(mingw安装方法：运行mingwstudiofullsetup.exe，并在环境变量path一栏中添加MinGWStudio/MinGW/bin文件夹所在路径)，如无需对待测指令的软件部分进行修改，仅安装Modelsim也可进

行验证，需要将./sim.do中

gcc -c gcc\_top.c

gcc -shared -Bsymbolic -o gcc\_top.dll gcc\_top.o两行前添加#注释掉

verilog代码存放位置：

顶层模块文件存放在./src/Verilog/下，

vsfx模块文件存放在./src/Verilog/vsfx下，

vcfx模块文件存放在./src/Verilog/vcfx下，

pu模块文件存放在./src/Verilog/pu下，

vfpu模块文件存放在./src/Verilog/vfpu下，

如需对硬件进行修改，替换相应目录下的文件即可

1. 根据需要验证的功能模块修改./Define.v中20行的CHECKMODE，

0 = vsfx, 1 = vcfx, 2 = vfpu, 3 = pu

根据需要验证的功能模块修改./Define.v中21行的BASE，

0 = random test only, 1 = base + random

根据需要验证的功能模块修改./Define.v中22行的LIMIT，为测试用例数，当测试用例达到sb.limit所规定的值，或者功能覆盖率(在./src/SystemVerilog /test/t\_altivec\_coverage.sv中说明，待完善)达到100%时，验证结束。

1. 根据需要验证的功能模块修改./src/SystemVerilog/test/t\_altivec\_env\_pkg.sv中的

class altivec\_c，修改或添加相应约束，如指令类型,缺省设置为：

vsfx:constraintc\_ins {ins inside {[ 1: 70]};}

vcfx:constraintc\_ins {ins inside {[ 71: 92]};}

vfpu:constraintc\_ins {ins inside {[ 93:108]};}

pu:constraintc\_ins {ins inside {[109:142]};}

1. 根据需要在./src/SystemVerilog/test/t\_altivec\_env\_pkg.sv第171行添加base测试向量。

对于VFPU指令，格式为tmp.setbase\_vfpu(vra,vra,vrb,vrc,ins\_index);

其他指令的格式为：tmp.setbase(vra,vrb,vrc,ins\_index,rc);

填写相应的vra，vrb，vrc，rc值(注：ins\_index不需修改)

并修改./src/SystemVerilog/test/t\_altivec\_env\_pkg.sv第36行为每条指令的总base测试向量数

5. 运行sim.bat，结果自动保存在./transcript文件中

transcript文件记录了系统产生的全部测试向量，格式如下：

# ->vra = ..., vrb = ..., vrc = ..., ins = ...(name of the instruction)

并且，当软硬件结果不匹配时，会打印如下信息：

# SBD ------------Ins ... (name of the instruction) MISMATCH!------------

# vra = ...

# vrb = ...

# vrc = ...

# software result: vrt = ...

# hardware result: vrt = ...

如图3所例

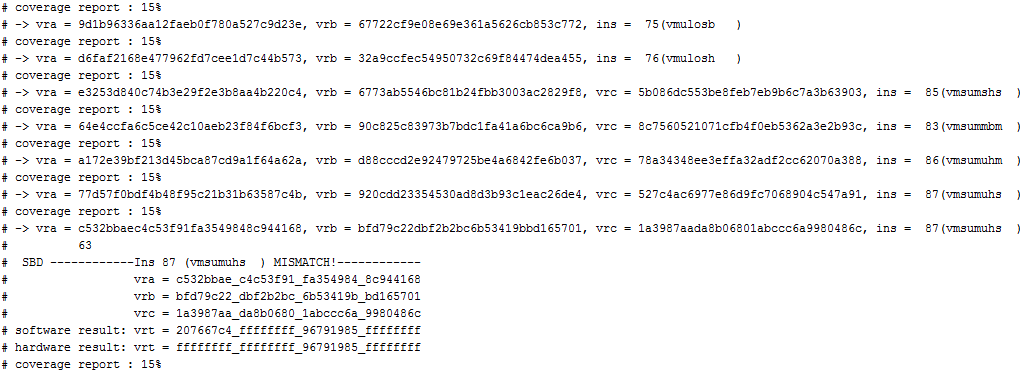


图3.打印信息举例

表1.文件说明：

|  |  |  |
| --- | --- | --- |
| 文件名称 | 文件描述 | 包含部分 |
| t\_top\_class\_based.sv | 顶层模块 | module top\_class\_based |
| t\_altivec\_clock\_reset.sv | 时钟信号和复位信号产生模块 | module altivec\_clock\_reset |
| t\_altivec\_coverage.sv | 覆盖率检测 | class altivec\_coverage |
| t\_altivec\_driver.sv | 获取stimulus generator产生的激励，产生相应控制信号，并将激励和相应的信号送到dut | class altivec\_driver |
| t\_altivec\_dut.sv | 例化硬件功能单元，产生硬件结果 | module altivec\_dut |
| t\_altivec\_dut\_wrapper.sv | 将dut封装，以增强其层次性 | module altivec\_dut\_wrapper |
| t\_altivec\_env.sv | 设置相关配置信息 | class altivec\_env |
| t\_altivec\_env\_pkg.sv | 将待测数据封装成类，描述激励产生和软硬件自动比对 | class altivec\_c  class stimulus\_generator class altivec\_scoreboard |
| t\_altivec\_import.sv | 与C语言的接口，导入ISS功能函数 |  |
| t\_altivec\_interface.sv | 定义driver与dut，dut与monitor之间的接口 | interface altivec\_clk\_if interface altivec\_dut\_pins\_if |
| t\_altivec\_monitor.sv | 获取硬件计算的结果，将其封装成类并传送给scoreboard | class altivec\_monitor |
| t\_altivec\_sva.sv | 断言：systemverilog assertion | module sva\_unit |

## 四、参考文献：

[1]. “Power ISA Version 2.06 Revision B”.

[2]. Martin S. Schmookler, Michael Putrino,“A Low-power, High-speed Implementation of a PowerPC Microprocessor Vector Extension”, IEEE Symposium on Arithmetic, 1999