浙江九霄智能科技有限公司

uSim 1.0 用户手册

目录

1 文档信息

1.1 关于本文

发布日期	07/01/2024
版权信息	浙江九霄智能科技有限公司
版本号	1.0
适用范围	支持 uSim 1.0 以及更高版本
使用限制	未经浙江九霄科技有限公司许可，任何形式的复制和分发，均视为违反本文件版权的行为。

表 1 - 关于本文

1.2 修改记录

修订	日期	描述
0.1	03.09.2024	增加仿真器安装使用说明
0.2	04.02.2024	增加仿真流程详细描述，增加初级代码示例 1~20
0.3	05.19.2024	增加后仿、覆盖率和后仿描述
0.4	06.09.2024	删除代码示例章节
0.9	06.15.2024	精简后仿描述
10.	06.25.2024	修改仿真流程描述中的错误，增加部分参数描述

表 2 - 修改历史

2 功能概述

USim 是一款功能强大且高效的仿真工具，可帮助设计人员在整个集成电路设计过程中确保其设计的正确性和可靠性。它提供了一个全面的仿真环境，并全面支持各种硬件描述语言（HDL），如 Verilog 和 SystemVerilog。USim 可以对设计进行功能仿真、时序仿真和功率解析，以确保其正确性和可靠性。

USim 提供了一套全面的特性和功能。它支持各种先进的仿真技术，主要是事件驱动仿真，以满足不同设计的多样化需求。通过优化的编译技术，USim 可提供快速的仿真速度和高效的内存利用率。此外，USim 还支持各种验证方法，例如功能覆盖率解析，从而提高了设计的质量和可靠性。

本章包括以下各节：

- 仿真器的技术支持
- 仿真器的设置
- 仿真器的使用
- 默认时间单位和精度

2.1 仿真器技术支持

USim 支持以下 IEEE 标准：

- 标准 Verilog 硬件描述语言（IEEE Std 1364）中定义的 Verilog 语言。
- System Verilog 语言参考手册中定义的 lEEE Std 1800 语言。

2.2 设置仿真器

USim 的基本环境，其中最重要的一步是获得许可证。

必须拥有许可证才能运行 USim。要获取许可证，请联系当地销售代表。销售代表将需要计算机的主机 ID，并提供许可证文件。把它放在 USim 的同一个目录下。

如果获取了相应的许可证，则在运行时将打印以下信息：

签 名 许 可 证 [DEFAULTClient-sigatureAABs-/lQW-ixA=extra-datalimit_DEFAULT : 1 lic_ver200valid-from xxxx-xx-xx valid-to xxxx-xx-xx（your 许可证的有效期）

主软件的许可证正常

ultrasky 技术

--

请等待加载...

无需自行安装和配置 GCC 等 C 编译器，它将与我们的软件安装包一起交付给，只需按照正确的安装步骤操作即可。

2.3 使用仿真器

USim 使用以下三个基本步骤来编译、链接和仿真任何 Verilog/SystemVerilog 设计:

- 解析设计
- 链接设计
- 仿真设计

解析设计

USim 为提供了内部创建的解析器 ，用于解析的 Verilog 设计代码。它解析设计并将中间文件存储在内存中。有关详细信息，请参见第 2 章 “USim 流”。

链接设计

USim 自动详细设计，并生成目标代码。 生成的文件链接到硬盘中的打包文件，共同生成可执行文件 simv。有关详细信息，请参见第 2 章 “ USim 流”。

仿真设计

通过执行二进制仿真可执行文件 simv 来仿真设计。有关详细信息,请参见第 2 章“USim 流”。

2.4 默认时间单位和时间精度

如果没有特别设定，仿真的默认时间单位为 1 ns。

此编译单元中所有模块/程序的时间刻度如下：

“-unit_timescale=1ns”

3 USim 工作流程

使用 USim 仿真设计涉及三个基本步骤：

- 解析
- 链接
- 仿真

3.1 解析

解析是仿真设计的第一步。在此阶段，USim 使用内部创建的解析器解析 HDL 文件。 在解析阶段，USim 会检查设计是否存在语法错误。在此阶段，USim 生成详细说明所需的中间文件，并将这些文件保存在内存中。

解析器解析 Verilog 设计文件，并将生成的带有.ast 后缀的中间文件存储在缓存文件 .ultra 中。

解析器使用 LR 算法将预处理的 SV 代码转换为 AST（抽象语法树）。

抽象语法树（AST）是源代码抽象语法结构的树状表示。它是用于描述程序代码结构和语法的编程语言语法结构的图形表示。

它提供了一个中间表示形式，使编译器或解释器更容易处理和操作源代码。

然后，生成语法树并将其转换为自定义的数据结构，HDL 代码被转换为 C++代码并存储在动态库中。

3.2 链接

链接是第二步。在此阶段，USim 使用解析过程中生成的中间文件和已打包在硬盘中的文件部分，构建实例层次结构并生成二进制可执行的 simv 文件。此二进制可执行文件用于后续仿真。

下面列出了一些常用的选项：

- **帮助选项**

  -h 打印出本帮助文件

  注：如果 Verilog 或者 SV 代码中未指明 dump 波形命令，本公司软件提供默认的 dump， 文件名为__[top_module_name].vcd；例如对于名称为 harness 的顶层模块项目，默认的 vcd 文件为__harness.vcd。

- **verilog 选项**

  -verilog

  仅支持 verilog 语法

  +define+MY_MACRO=7

  命令行中的宏定义高于代码中定义的宏（“MY_MACRO”）：

- **指定延迟和 SDF 文件的选项**

  -sdf min|typ|max:instance_name:file.sdf

  启用 SDF 注释。文件中指定的最小值、典型值或最大值。SDF 在实例 instance_name 上进行了注释。

  +maxdelay

  指定在编译 SDF 文件时使用 min: type:max delay 三元组中的最大定时延迟。$sdf_annotate 系统任务的 mtm_spec 参数覆盖了这个选项。

  +mindelay

  指定在编译 SDF 文件时使用 min: type:max delay 三元组中的最小定时延迟。$sdf_annotate 系统任务的 mtm_spec 参数覆盖了这个选项。

  +typdelays

  指定在编译 SDF 文件时使用 min: type:max delay 三元组中的典型定时延迟。$sdf_annotate 系统任务的 mtm_spec 参数覆盖了这个选项。

+nbaopt

消除非阻塞分配中的惯性延迟

❖ **指定块和定时检查选项**

+pathpulse

在指定块中搜索 pathpulse $ specparam。

+nospecify

在指定块中抑制模块路径延迟和定时检查。

+notimingcheck

禁止在指定块中进行时序检查

❖ **脉冲滤波选项**

+pulse_e/number

显示一条错误消息，并为任何宽度小于或等于 number 参数指定的模块路径延迟百分比的路径脉冲传播一个 x 值，但仍然大于 number 参数指定的模块路径延迟百分比到 +pulse_r/number 选项。

+pulse_r/number

拒绝任何宽度小于模块路径延迟百分比的脉冲。number 参数的取值范围是 0 到 100。

+pulse_int_r

与现有的+pulse_r 选项相同，只是它只适用于 INTERCONNECT 延迟。

+pulse_int_e

与现有的+pulse_e 选项相同，只是它只适用于 INTERCONNECT 延迟。

❖ **负时序检查选项**

+neg_tchk

在时序检查中启用负值

❖ **在文件中指定源文件和编译/精化选项的选项**

-F /my_home/filelist.f

添加相对路径的文件列表文件

-f /my_home/filelist.f

添加绝对路径的文件列表文件。注意，该 filelist.f 文件可能包含 -F/-f，以及 +define+MY_MACRO, +incdir+/xx/xxx

❖ **指定时间刻度的选项**

-timescale=1ns/1ps

第一个模块/程序的时间刻度

-vtimescale=1ns/1ps

定义命令行文件的初始时间刻度

-unit_timescale=1ns/ps:

该编译单元中所有模块/程序的时间刻度

-override_timescale=1ns/1ps

覆盖源代码中所有“TimeScale 编译器”指令的时间单位和精度单位

❖ **指定目录**

-top module_name/program_name:

指定顶层模块/程序的名称

+incdir+../xxx/xxx

包含文件的搜索路径

/xxx/xxx

待编译的 Verilog 或 SystemVerilog 文件

-y library_path

指定外部模块库，与 +libext+ 一起使用

+libext+.v+.sv

指定后缀为 .v 或 .sv 的外部模块库文件名

-v module_library_file

指定外部 Verilog 库文件

-libmap library_mapping_file

在分析期间指定库映射文件

❖ **其他选项**

-e inputFile outputFile algorithm

使用 algorithm（des） 加密 inputFile 并另存为 outputFile

-d inputFile key algorithm

使用密钥使用算法解密 inputFile

-compile

仅解析，不详述

-preprocessing

仅预处理，不解析

-compliance vcs/xrun

遵循 VCS 或 XRUN 的规则

-l compile.log

指定日志文件

-errormax 5

指定处理的最大错误数

3.3 仿真

在链接过程中,使用生成的中间文件,USim 创建一个二进制可执行 simv。可以使用 simv 运行。

使用以下命令行对设计进行仿真:

-us

链接完成后，执行 -simv 继续下一步。

下面列出了一些常用的选项以供参考

-svseed 1: 指定种子

-svseed random: 随机种子

+ntb_random_seed =1: 设置要在模拟开始时由顶级随机数生成器使用的种子值。srandom(seed)系统函数调用将覆盖此设置。取值为任意整数。默认的随机种子值是 1。相对于 svseed1，+ntb_random_seed =1 是一个更通用和标准的方法。

+ntb_random_seed_automatic: 工具提供的自动化种子，由时间、主机名和进程 id 组合而成。这确保了没有两个模拟具有相同的起始种子。

4 门级仿真

门级仿真（Gate Level Simulation）是数字电路设计验证过程中的一个重要步骤，它更接近硬件实际工作状态，可以更准确地预测电路的性能。

uSim 仿真器支持不带时序反标两种门级仿真，也支持符合 Systemverilog 标准的 SDF 文件反标的时序仿真。

4.1 SDF注释

OVI 标准延迟文件(SDF)规范提供了用于表示和应用延迟信息的标准 ASCII 文件格式。USim 支持本规范的 OVI 版本 1.0、2.0、2.1、3.0 和 IEEE 4.0。

在 SDF 格式中，工具可以指定固有延迟、互连延迟、端口延迟、时序检查、时序约束和脉冲控制（PATHPULSE）。

当 USim 读取 SDF 文件时，它会将延迟值“反向注释”到设计中，即添加延迟值或更改源文件中指定的延迟值。

以下是对 SDF 文件中指定的延迟进行反向注释的方法:

- **使用统一的 SDF 特性**

使用$sdf_annotate 系统任务

使用统一的 SDF 特性

统一的 SDF 特性允许使用以下编译/链接选项对 SDF 延迟进行反向注释:

-sdf min|typ|max:instance_name:file.sdf

对于 USim 设计流程:

链接

% us -sdf min|typ|max:instance_name:file.sdf

[elab_options] top_cfg/entity/module

仿真

% -simv [run_options]

- **使用$sdf_annotate 系统任务**

可以使用$sdf_annotate 系统任务将延迟值从 SDF 文件反向注释到 Verilog 设计中。

$sdf_annotate 系统任务的语法如下:

$sdf_annotate ("sdf_file"[, module_instance]

[,"sdf_configfile"][,"sdf_logfile"][,"mtm_spec"]

[,"scale_factors"][,"scale_type"]);

位置位于: "sdf_file"

链接 ：% us [elab_options] top_cfg/entity/module

仿真 ：% -simv [run_options]

4.2 SDF配置文件

可以使用配置文件在模块类型和全局基础上控制以下内容：

- 最小值：典型值：最大选择值
- 缩放（scale）
- MIPD（模块输入延迟）近似策略，适用于同一输入端口的“重叠”注释情况。

SDF 配置文件中使用以下命令及其语法：

- INTERCONNECT_MIPD 命令
- MTM 命令
- SCALE 命令

4.3 延迟和时间安排

延迟可分为传输延迟和惯性延迟，以 DELAY 关键字开头的时序规范应将延迟值与输入到输出路径、输入端口、互连和设备输出相关联。它们还可以为输入到输出路径提供窄脉冲传播数据。

有两种类型的时序控制：延迟和事件表达式。延迟控制只是在模拟器遇到语句和实际执行语句之间添加 delay 的一种方式。 事件表达式允许语句延迟到某个模拟事件发生之前，这些模拟事件可以是网络上的值更改，也可以是可用的（隐式事件）或在另一个过程中触发的显式命名事件。

❖ 惯性延迟实现

对于原语[门、开关和用户定义原语(UDP)]、连续分配、mipd、模块路径延迟和 INTERCONNECT 延迟，从 SDF 文件到网络的反向注释，惯性延迟实现是相同的。还有第三种实现，用于模块路径和 INTERCONNECT 延迟和脉冲控制，参见“脉冲控制”。

惯性延迟的实现如下：

考虑一个由脉冲前沿调度的事件,该事件要么被调度到稍后的模拟时间,要么尚未发生。该事件在指定延迟结束时和新的模拟时间由后缘调度的事件取代。所有的窄脉冲都被滤除。

❖ 允许传输延迟

传输延迟不是默认延迟。可以使用+transport_path_delay 编译选项指定模块路径延迟上的传输延迟。要使这个选项起作用，还必须包含+pulse_e/number 和+pulse_r/number 编译选项。

可以使用+transport_int_delay 编译选项指定网络上的传输延迟,并对其反向注释 SDF INTERCONNECT 延迟。要使这个选项起作用，还必须包含+pulse_int_e/number 和 +pulse_int_r/number 编译选项。

+pulse_e/number,+pulse_r/number,+pulse_int_e/number 和+pulse_int_r/number 选项定义了脉冲宽度的特定阈值，这允许告诉 USim 只过滤掉一些脉冲，而让其他脉冲传播。

4.4 使用配置文件

可以使用 USim 配置文件来禁用模块路径延迟、指定模块以及指定模块定义的所有实例的块和时序检查。 定时的属性关键字如下:

noIopath ： 在指定的模块实例中禁用模块路径延迟。

Iopath ：在指定的模块实例中启用模块路径延迟。

noSpecify ：禁用指定模块实例中的指定块。

noTiming ：在指定的模块实例中禁用定时检查。

Timing ：在指定模块实例中启用定时检查。

5 覆盖率

USim 在仿真过程中监控设计的功能覆盖，以便验证工程师可以将精力集中在那些没有被覆盖到的功能点，以尽快实现 100% 的覆盖率。USim 提供功能覆盖技术来测试 HDL 代码的功能覆盖率。

功能覆盖率检查实现的整体功能。要执行功能覆盖，必须定义 DUT 中要覆盖的功能的覆盖点。USim 支持 Covergroup。

6 关于 UVM

UVM 是一个以 SystemVerilog 为主体的验证平台开发框架，为使用 SystemVerilog 的代码实现提供了支持库。

USim 自带 UVM-1.2 库，位于：$USim_HOME/CodeGen/uvm-1.2 中。

下面列出了一些常用的选项

-ntb_opts uvm-1.2 ： 编译 uvm-1.2

-uvm ：编译 uvm-1.2，更通用的开关

-uvmhome CDNS-1.2 ：查找 UVM 安装的位置

7 附录

NA

