CS250 2013年lab content

Lab1:

Overview：

本作业的目标是为了让您熟悉整个学期都将使用的VLSI CAD工具。

您将了解ASIC工具流中的每个阶段，并有机会进行一些Chisel coding.具体来说，您将编写一个最大公约数(GCD)电路的RTL模型，综合、PR，通过在每个阶段的模拟来验证其正确性，并进行功率分析（power analysis）。

Deliverables ：

1. ChiselRTL
2. 由VCS, DC Compiler, Formality, IC Compiler, and PrimeTime PX生成结果和报告
3. 对本文档末尾给出的问题的书面答案作为writeup/report.pdf或writeup/report.txt。

VLSI Toolflflow Introduction

图1说明了您将用于第一个实验室的工具流。您将使用Synopsys公司VCS(vcs)来模拟和调试RTL设计。

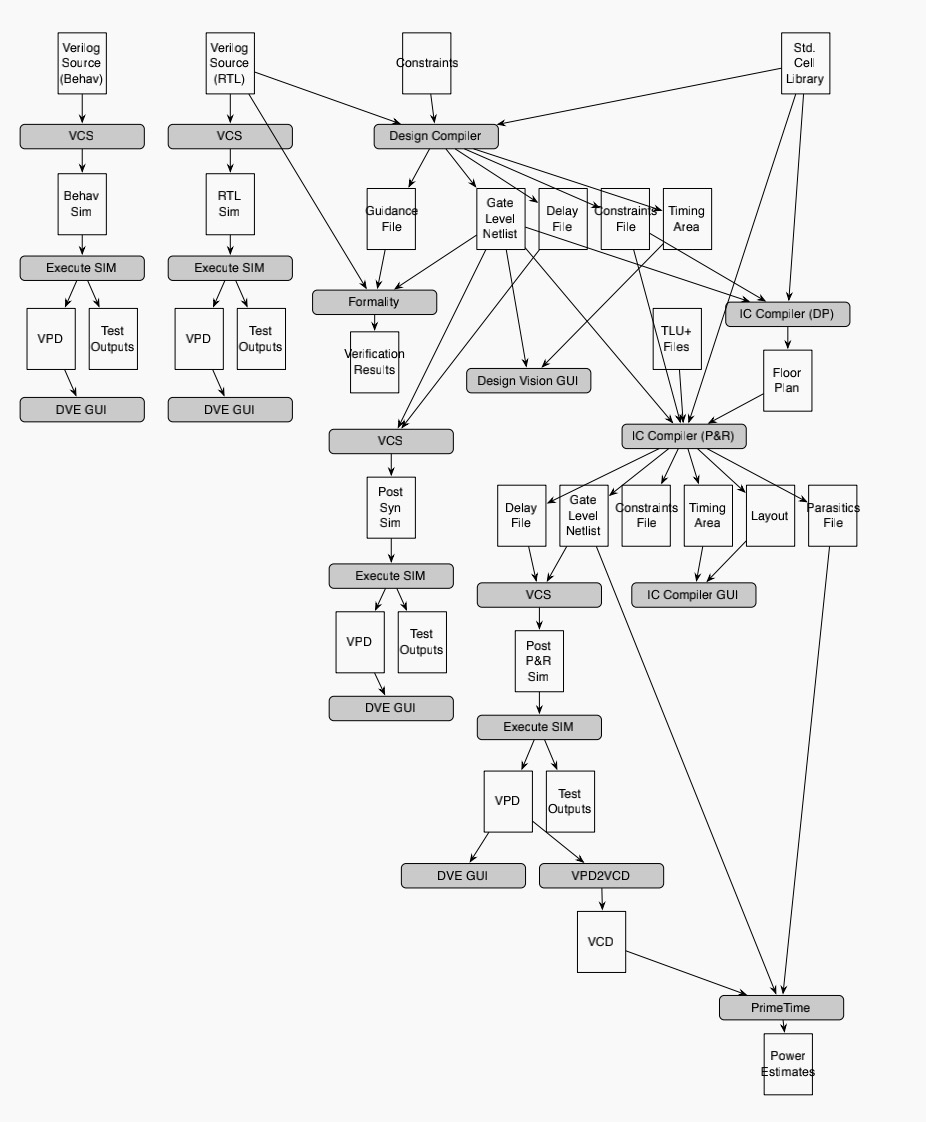
在设计正确后，您将使用Synopsys设计的编译器(dc\_shell-xg-t)来综合您的设计。综合是将RTL转换为门级网列表的过程。

您将使用Synopsys Formality (fm shell) 来正式验证RTL模型和门级模型是否匹配。

然后再次使用VCS来simulate the synthesized gate-level netlist。

在得到可以工作的 gate-level netlist, 将使用 Synopsys IC Compiler (icc shell) to place and route 我们的设计。Placement 是将标准单元放置在芯片上合适位置的处理过程。 routing 包括将各个单元通过金属线连接起来using traces on the various metal layers, with vias providing connections between the metal layers.

综合和PARA工具会反馈给我们设计的 performance and area characteristics 。来自PAR的结果更贴近现实情况，但需要更长的时间来产生。在PAR后，您将使用VCS生成和模拟最终的门级网列表。You will use this gate-level simulation not only as a fifinal verifification step, but also to generate transition counts for every net in the design. Synopsys PrimeTime PX (pt shell)获取这些转换计数，并将它们与从最终布局中提取的电容值关联起来，以估计设计所消耗的功率。下图说明了这些工具是如何一起工作的。



正如您从图中所看到的，即使是一个简单的设计，从RTL一直到晶体管级的实现，也需要许多不同的工具。每个工具都非常复杂，而且许多工业工程师只专门从事一种工程。

为了在一个学期内产生一个超大规模集成电路设计，我们需要了解每一个。

每个工具都有一个GUI界面。但是，您想要执行的大多数命令在每次设计迭代中都是相同的，并且会重复输入，因此使用TCL脚本来实现自动化过程。

当您使用GUI时，终端窗口将显示与每个鼠标操作或菜单项对应的命令行，并可以将这些命令添加到脚本中。

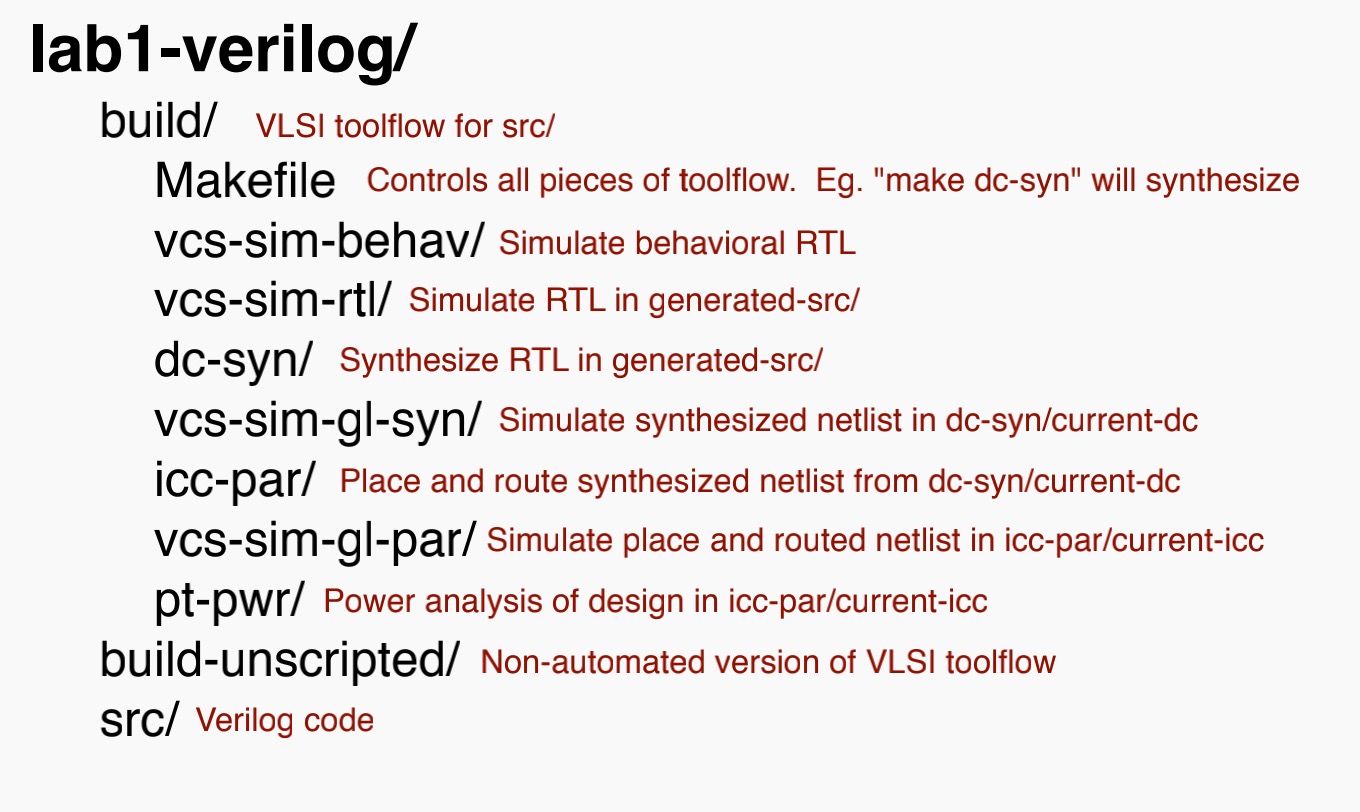
为了保持文件的组织性，工具流的每个部分都有自己的构建目录和自己的Makefile。Make文件初始化程序并将其指向设置脚本。

顶级Makefile连续运行每个程序，因此单个命令可以在没有任何用户干预的情况下始终进行RTL设计。

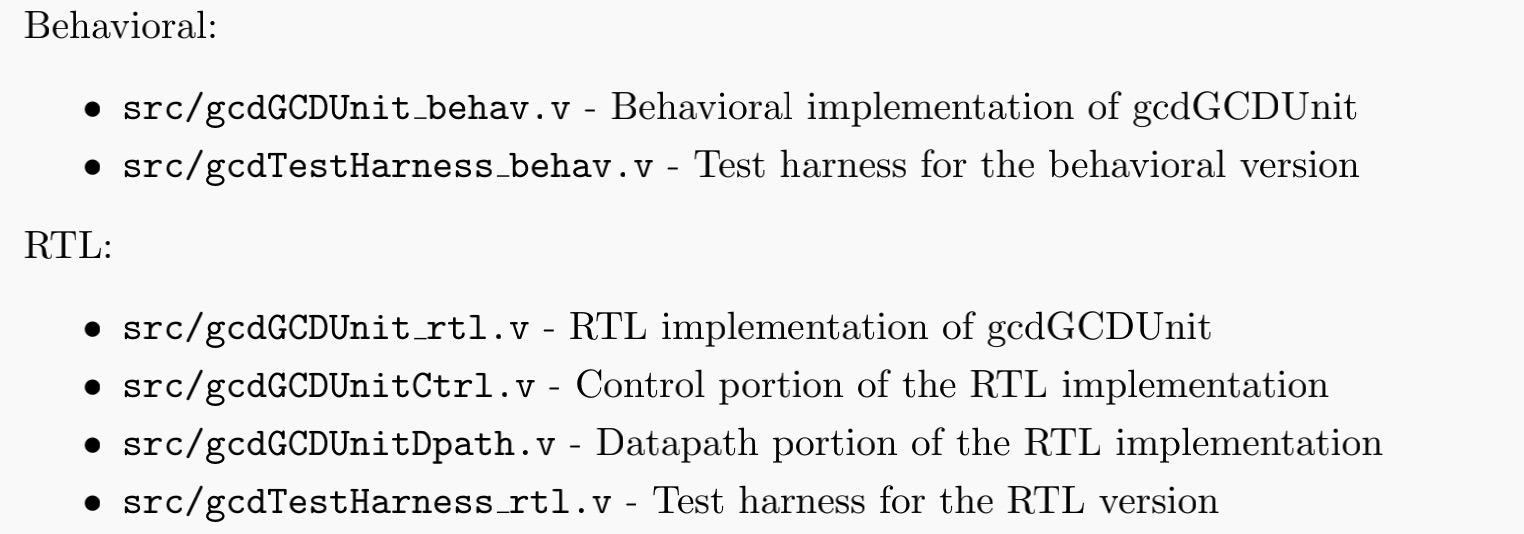
这将有助于设计空间探索，使您可以轻松修改设计的某些部分，并查看这些变化如何影响结果芯片的功率、区域和性能特性。

图2显示了实验提供的资料，并包括关于它们所做的操作的注释。

我们提供了两个Verilog设计，一个“behav”版本，和一个“RTL”版本。behav设计有点像C代码。它只描述了一个模块的行为，而是不打算被综合成硬件中的。testbench通常以行为风格书写。

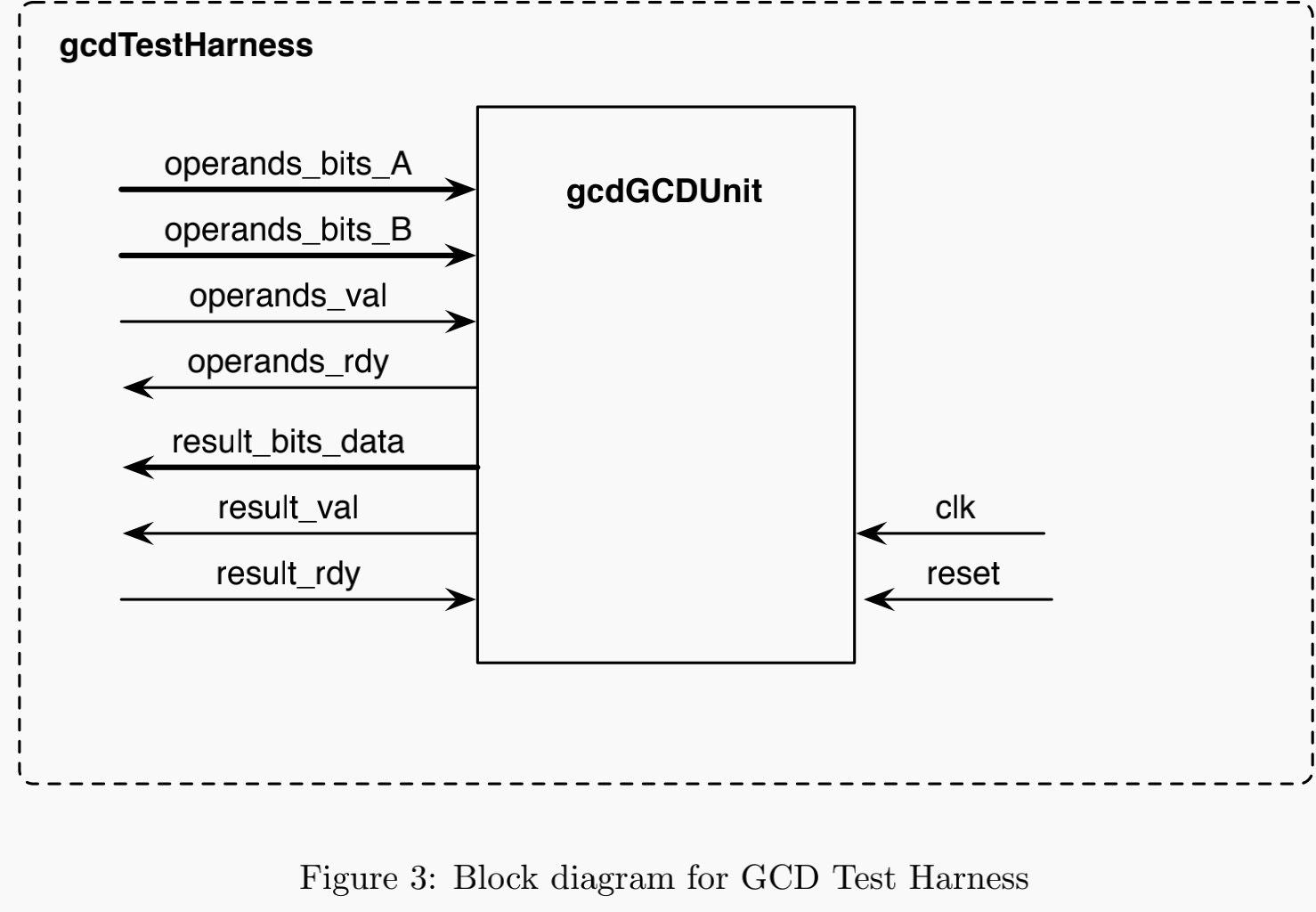


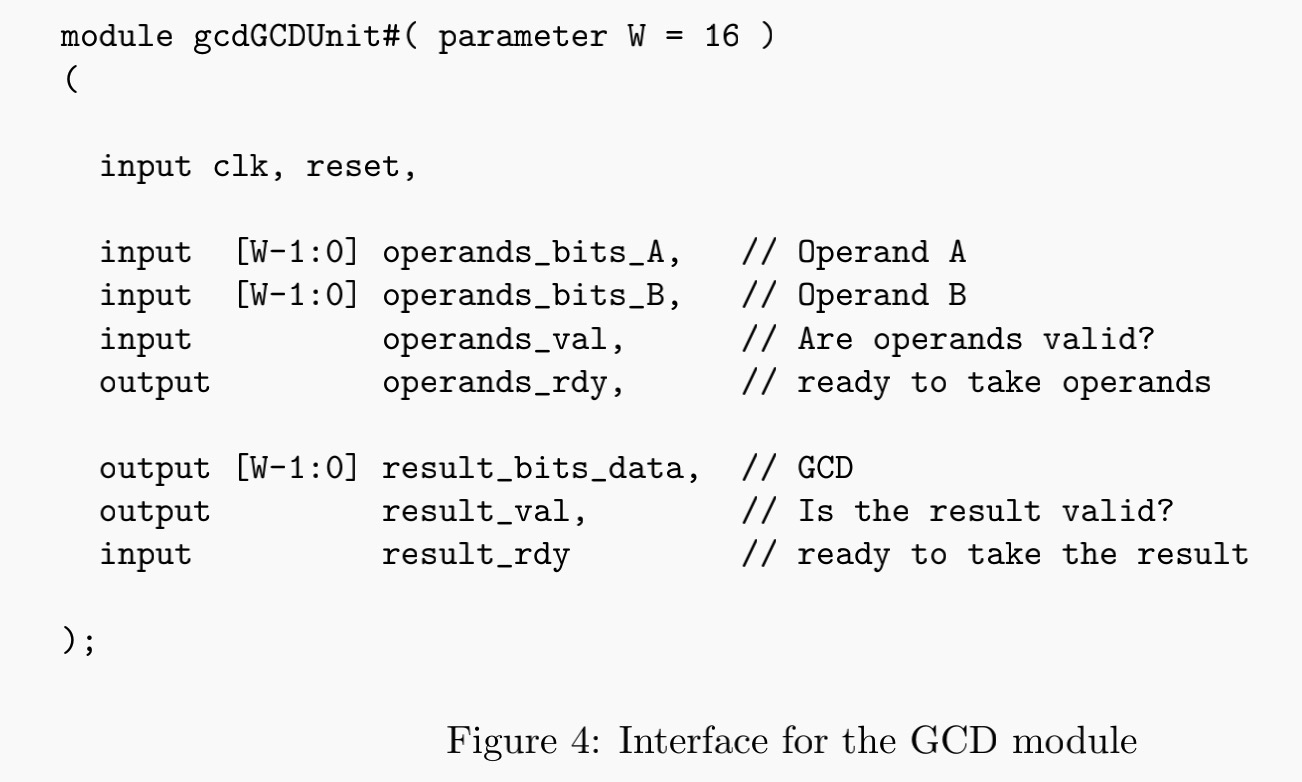
RTL设计是对设计的寄存器传输级别（组合逻辑和寄存器）的描述。这个RTL级别的描述将被合成成一个逻辑门和状态元素的集合，并最终用于生成一个完整的ASIC布局。



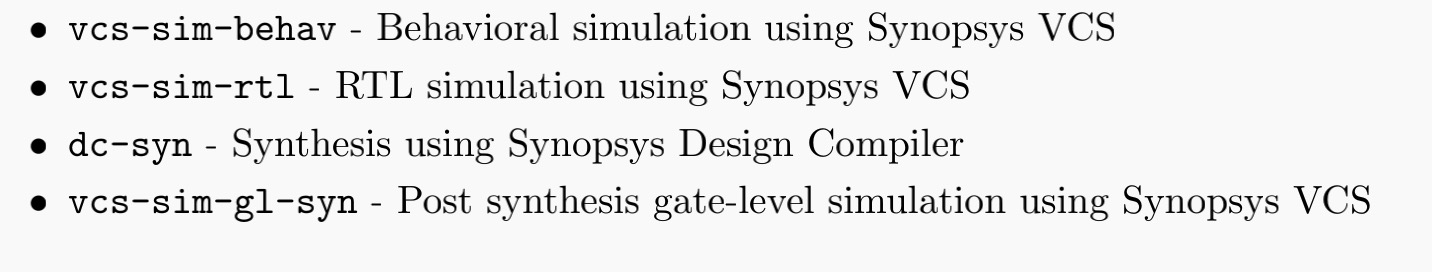
该设计的方框图如图3所示。顶层模块名为gcdGCDUnit，其界面如图4所示。

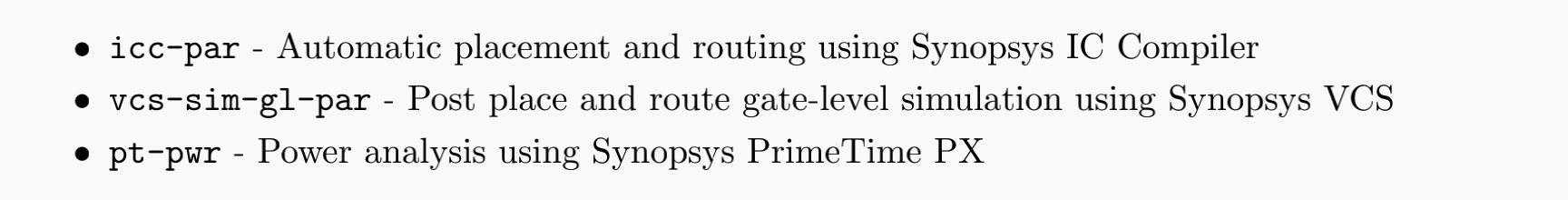
我们已经为您提供了一个a test harness，它将驱动输入和验证您的设计的输出





bulid和build--unscripted 目录包含以下子目录，您将在构建芯片时使用这些子目录。列出它们的顺序是它们在工具流中出现的顺序。





每个子目录都包含一个Makefile，有些还包含脚本文件。在子目录中运行“make”命令将启动相应的工具。For example, to perform synthesis using Design Compiler (DC):

注：您必须按照上面列表中给出的顺序进行操作。

一旦您验证了流中的每个步骤成功，您就可以使用构建目录中的顶层标记文件按顺序执行流中的多个步骤。

例如，一旦所有设计文件和脚本都设置正确，您应该能够使用以下命令synthesize, flfloorplan, and place and route your design.

% cd $LABROOT/build

% make icc-par

**Pushing a Design Through the VLSI Tools**

首先，在学习如何使用脚本自动化该过程之前，您将手动调用几个工具，并以交互方式输入命令。You will perform the manual build in the *build-unscripted* directory, and the automated build using makefiles and scripts in the *build* directory.

**Synopsys VCS: Simulating your Verilog**

VCS将Verilog源文件编译为本机二进制文件，以实现对Verilog设计的模拟。VCS可以同时模拟行为级别和RTL级别的验证模块。在行为模型中，模块的功能是通过使用更高级层次的抽象来更容易地描述的。在RTL描述中，一个模块的功能在一个可以映射到寄存器和门的集合的级别上被描述了。

Verilog行为模型可能无法合成，但它们在构建测试台和模拟与设计接口的外部设备时很有用.

我们为这个实验室提供的 test harness是如何使用behavioral Verilog的一个很好的例子。您将首先模拟以behavioral style实现的GCD模块。

**% cd $LABROOT/build-unscripted/vcs-sim-behav**

**% vcs -full64 -PP +lint=all +v2k -timescale=1ns/10ps \**

**../../src/gcdGCDUnit\_behav.v \**

**../../src/gcdTestHarness\_behav.v**

默认情况下，VCS会生成一个名为simv的模拟器二进制文件。

* -PP命令行选项启用了对使用VPD跟踪输出格式的支持。
* +lint=all 参数打开所有Verilog警告。由于编写不符合预期行为的合法Verilog代码非常容易，所以您应该始终启用所有警告来帮助您捕捉错误。例如，如果您尝试连接 两个 具有 不同位宽的nets 或 有些模块上的端口没有连接，VCS会警告您。始终尽量消除所有VCS编译错误和警告.
* +v2k命令行选项告诉VCS要启用Verilog-2001语言特性.
* Verilog允许设计者使用“‘timescale ”来指定其设计中的抽象延迟单元如何映射到实时单元

为了便于更改此参数，您将在命令行上指定它，而不是在Verilog源代码中指定它。

在这些参数之后，您将列出Verilog源文件。-v标志用于指示哪些Verilog文件是库的一部分（因此只在需要时编译）以及哪些文件是实际设计的一部分(因此应该始终被编译)。

运行此命令后，您应该会看到文本输出，这表明VCS正在解析Verilog文件并编译模块。

注意，VCS实际上生成C++代码，然后使用gcc编译。当VCS完成时，在构建目录中应该有一个simv可执行文件。现在尝试运行它：

**./simv**

...

*Entering Test Suite: exGCD\_behav*

*[ passed ] Test ( gcd(27,15) ) succeeded, [ 0003 == 00000003 ]*

*[ passed ] Test ( gcd(21,49) ) succeeded, [ 0007 == 00000007 ]*

*[ passed ] Test ( gcd(25,30) ) succeeded, [ 0005 == 00000005 ]*

*[ passed ] Test ( gcd(19,27) ) succeeded, [ 0001 == 00000001 ]*

*[ passed ] Test ( gcd(40,40) ) succeeded, [ 0028 == 00000028 ]*

*[ passed ] Test ( gcd(250,190) ) succeeded, [ 000a == 0000000a ]*

在命令行上键入所有Verilog源文件可能非常乏味，因此我们将使用Make文件来自动化编译模拟器二进制文件的过程。

**% cd $LABROOT/build/vcs-sim-behav**

**% cat Makefile**

**...**

**vsrcs = \**

**$(srcdir)/gcdGCDUnit\_behav.v \**

**$(srcdir)/gcdTestHarness\_behav.v \**

**...**

**% make**

**% make run**

**您可以利用相同的生成文件来为设计的RTL级别的虚拟化版本构建模拟器。**

*% cd $LABROOT/build/vcs-sim-rtl*

*% cat Makefile*

*...*

*vsrcs = \*

*$(srcdir)/gcdGCDUnitCtrl.v \*

*$(srcdir)/gcdGCDUnitDpath.v \*

*$(srcdir)/gcdGCDUnit\_rtl.v \*

*$(srcdir)/gcdTestHarness\_rtl.v \*

*...*

*% make*

*% make run*

*./simv +verbose=1*

*...*

*+ Running Test Case: gcdGCDUnit\_rtl*

*[ passed ] Test ( vcTestSink ) succeeded, [ 0003 == 0003 ]*

*[ passed ] Test ( vcTestSink ) succeeded, [ 0007 == 0007 ]*

*[ passed ] Test ( vcTestSink ) succeeded, [ 0005 == 0005 ]*

*[ passed ] Test ( vcTestSink ) succeeded, [ 0001 == 0001 ]*

*[ passed ] Test ( vcTestSink ) succeeded, [ 0028 == 0028 ]*

*[ passed ] Test ( vcTestSink ) succeeded, [ 000a == 000a ]*

*[ passed ] Test ( vcTestSink ) succeeded, [ 0005 == 0005 ]*

*[ passed ] Test ( vcTestSink ) succeeded, [ 0000 == 0000 ]*

*[ passed ] Test ( Is sink finished? ) succeeded*

*...*

如果一个设计没有通过你所有的测试，你应该从哪里开始？

答案是使用发现可视化环境(DVE)GUI来调试RTL代码，以在设计中生成信号的波形视图。

模拟器已经将您设计中每个net的活动跟踪写入到vcdplus.vpd文件中。

DVE可以读取vcdplus.vpd文件并可视化

**% ls**

**csrc Makefile simv simv.daidir timestamp vcdplus.vpd**

**% dve -full64 -vpd vcdplus.vpd &**

要将信号添加到波形窗口（见图5），您可以在层次结构窗口中选择它们，然后右击选择“*Add To Waves > New Wave View*.

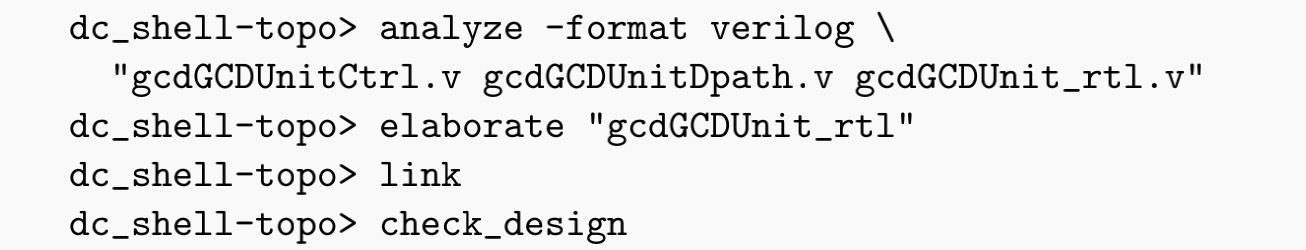
### Synopsys Design Compiler: RTL to Gate-Level Netlist

**Design Compiler**执行硬件综合。一个综合工具以一个RTL硬件描述和一个标准的单元库作为输入，并生成一个gate-level网表作为输出。

由此产生的门级网表是设计的完全结构描述，在设计层次结构中只有标准单元格(后面可能是SRAM micro)的叶子。To cut and past commands from this lab into your Design Compiler shell

To cut and past commands from this lab into your Design Compiler shell and make sure Design Compiler ignores the dc shell-topo> string, we will use an alias to ”undefifine” the dc shell-topo> string.

这些命令指向您的Verilog源目录，创建一个Synopsys work directory，并指向您将用于该类的标准库。setsvf命令用于设置同步系统形式性所使用的Synopsys Formality。现在，您可以加载您的Verilog设计到Design compiler with the **analyze, elaborate, link, and check design** commands.



在综合设计之前，必须指定一些约束条件；最重要的是，您必须告知工具您的目标时钟周期。下面的命令告诉Design Compiler，名为clk的引脚是clock，你的目标时钟周期是1纳秒。

dc\_shell-topo> create\_clock clk -name ideal\_clock1 -period 1

现在，您准备使用**compile ultra**命令实际将您的设计综合到一个门级网列表。

**-no autoungroup** 的指定是为了在综合期间保存层次结构。

*dc\_shell-topo> compile\_ultra -gate\_clock -no\_autoungroup*

*...*

*Beginning Delay Optimization*

*----------------------------*

*0:00:04 3113.2 0.02 0.1 0.0*

*0:00:04 3142.7 0.00 0.0 0.0*

*0:00:04 3142.7 0.00 0.0 0.0*

*0:00:04 3222.8 0.00 0.0 0.0*

*0:00:04 3222.8 0.00 0.0 0.0*

*0:00:04 3222.8 0.00 0.0 0.0*

*0:00:04 3222.8 0.00 0.0 0.0*

**compile ultra**命令将报告设计是如何被优化的。您应该看到DC执行技术映射、延迟优化和面积缩小。上面**compile ultra**的片段显示了the worst negative slack，这表明在设计中的关键路径和指定的时钟约束之间有多少空间。

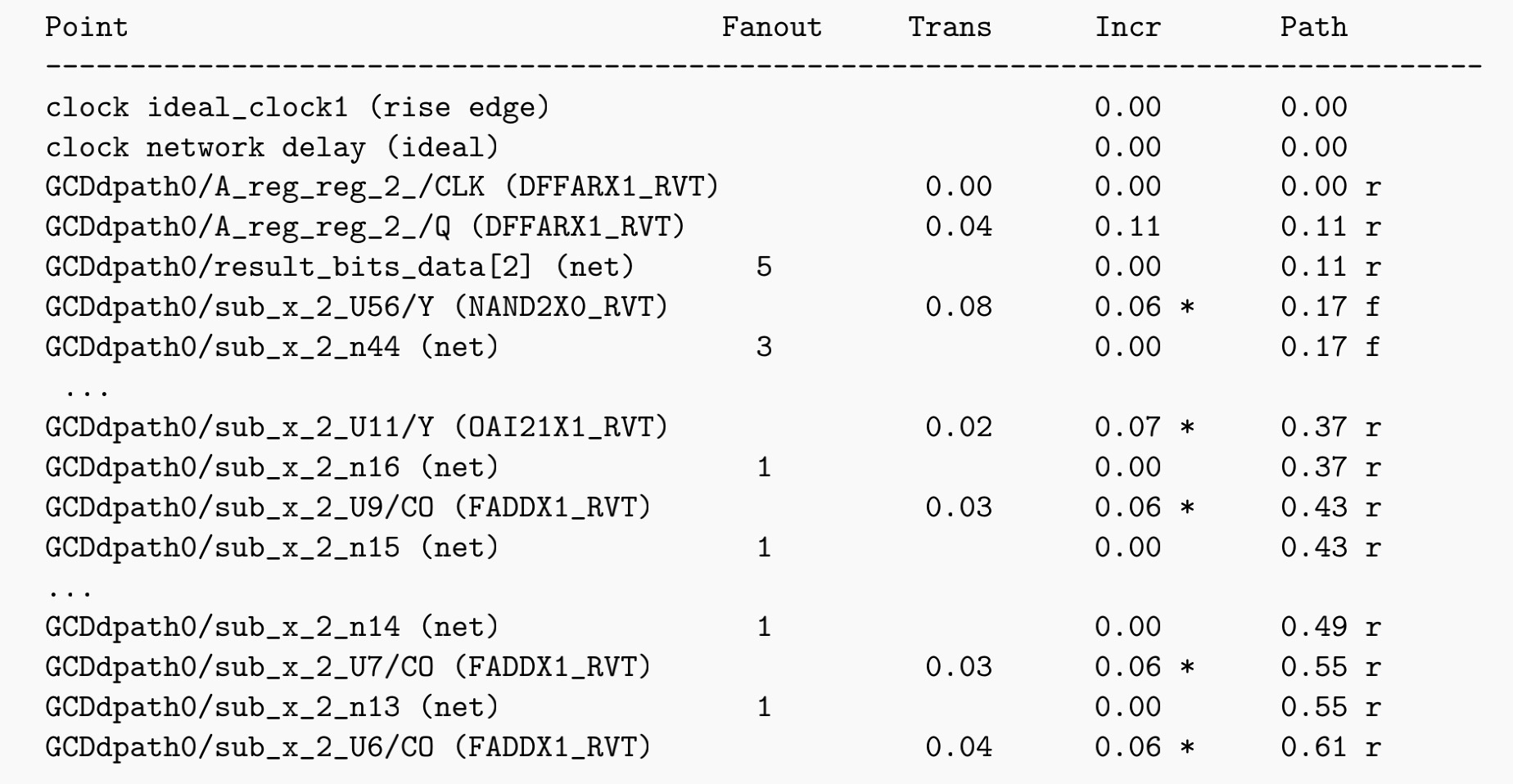
较大的negative slack values ， 因为这意味着您的设计大量缺少所需的时钟频率。Total negative slack是设计中所有端点（寄存器输入或顶级输入/输出端口）上的所有negative slack

的总和。

现在，您可以生成formal verifification tool所需的指导信息，并生成the synthesized gate-level netlist 和导出的约束文件。

让我们来看看描述综合结果的各种报告吧：

*dc\_shell-topo> report\_timing -transition\_time -nets -attributes -nosplit*



本报告显示了设计的关键路径。关键路径在设计中任意两个寄存器之间的传播延迟最长，因此在设计的操作点上设置了一个上限输入频率。在本报告中，我们可以看到关键路径从数据路径中操作数A寄存器的第2位的输出开始，到数据路径中的几个分包器，到输出规则结束ter 15.临界路径总共需要0.91ns，这小于1ns的时钟周期约束。这反映在最后一行宣布，积极的松弛的时机已经成熟了。