Digital Design with Chisel

Chisel数字系统设计基础

Chisel/FIRRTL Hardware Compiler Framework

深圳大学　计算机与软件学院　　罗秋明

Chisel数字系统设计基础

目标

初步掌握用**Chisel**设计数字系统

背景知识

- 1）数字电路
  - 组合逻辑、时序逻辑
- 2）C、Java等编程语言

Digital Design with Chisel

Chapter 1

Chisel简介

罗秋明

2023-08-11

网络资源

官网：

https://www.chisel-lang.org

文档：

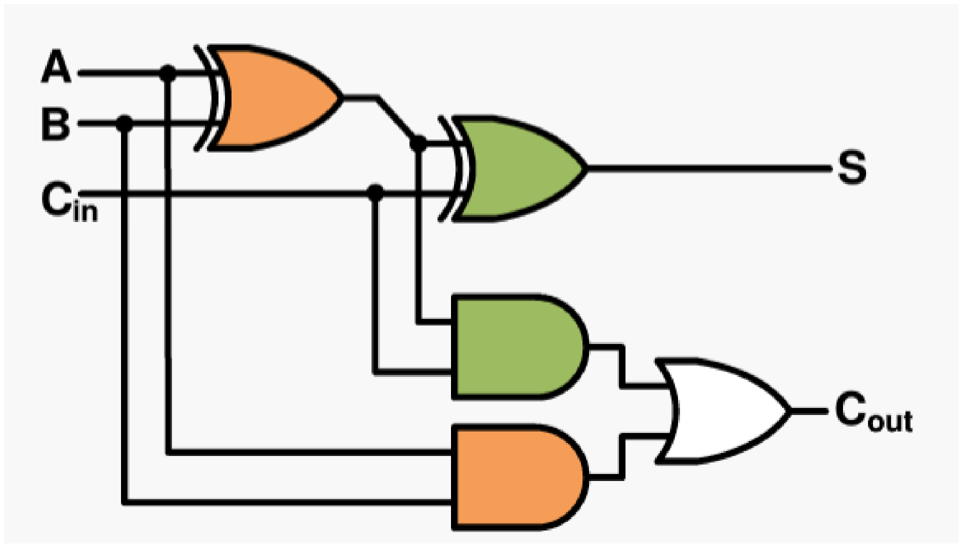
https://www.imm.dtu.dk/~masca/chisel-book.pdf

1.1 硬件描述语言HDL

- HDL (hardware description language)
  - 数字硬件描述语言简称为数字硬件语言
  - 行为描述、结构描述、数据流描述
  - 经EDA工具的仿真、综合后，可由FPGA或ASIC实现
- 常用HDL
  - VHDL (Very High-Speed IC Hardware Description Language)
  - Verilog
- 发展
  - System Verilog、System C、Chisel

1-bit 全加器示例

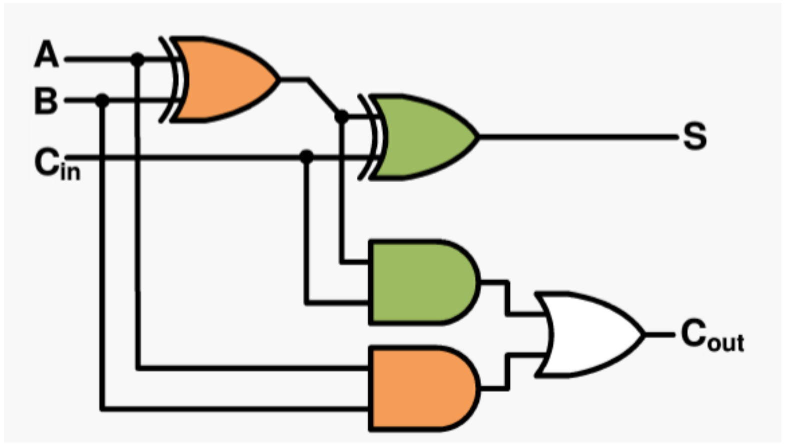
- 两个半加器构成
  - 输入端口: A、B、Cin
  - 输出端口: Sum、Cout



A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

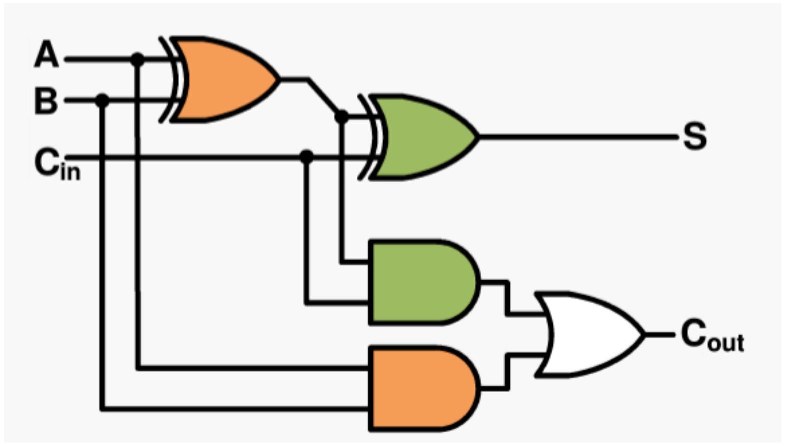
VHDL 版 1-bit 全加器

```
library ieee;
 use ieee.std_logic_1164.all;

entity full_adder is
          Port (     a : in std_logic;
                     b : in std_logic;
                     cin : in std_logic;
                     sum : out std_logic;
                     cout : out std_logic);
end entity;

architecture behavioral of full_adder is
begin
          sum <= a xor b xor cin;
          cout <= (a and b) or (cin  and (a xor b));
end architecture;
```



Verilog 版 1-bit 全加器

```
module full_adder(
        input a,
        input b,
        input cin,

        output sum,
        output cout );

        assign sum = a^b^cin;
        assign cout = (a&b)|((a^b)&cin);

endmodule
```

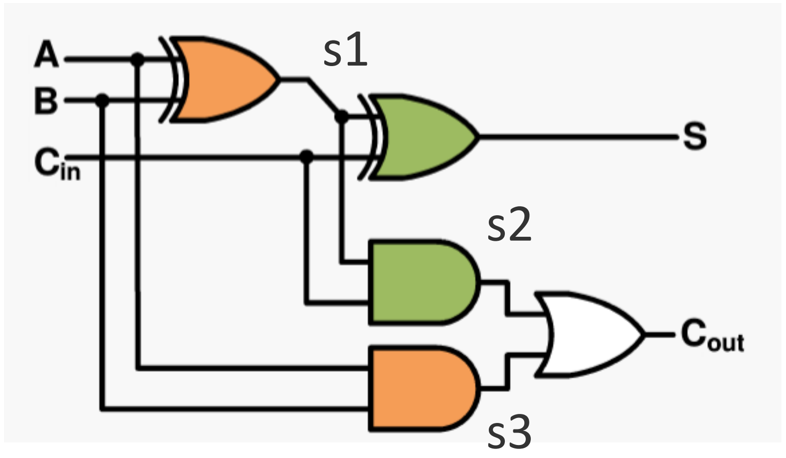


1.2 Chisel

- Chisel from UC Berkeley 2012
  - Constructing Hardware In a Scala Embedded Language
  - 硬件构造语言（ Hardware construction language）
  - 优势：抽象层次高、易于扩展、可读性高、开源支持
- Chisel →Scala →Java on JVM
  - a library of classes and functions
  - 引入了OOP 和函数式编程
- 敏捷开发(Agile Development)
  - 硬件工程师 ←→软件工程师
  - RISC-V：RocketChip、香山处理器

Chisel 版 1-bit 全加器

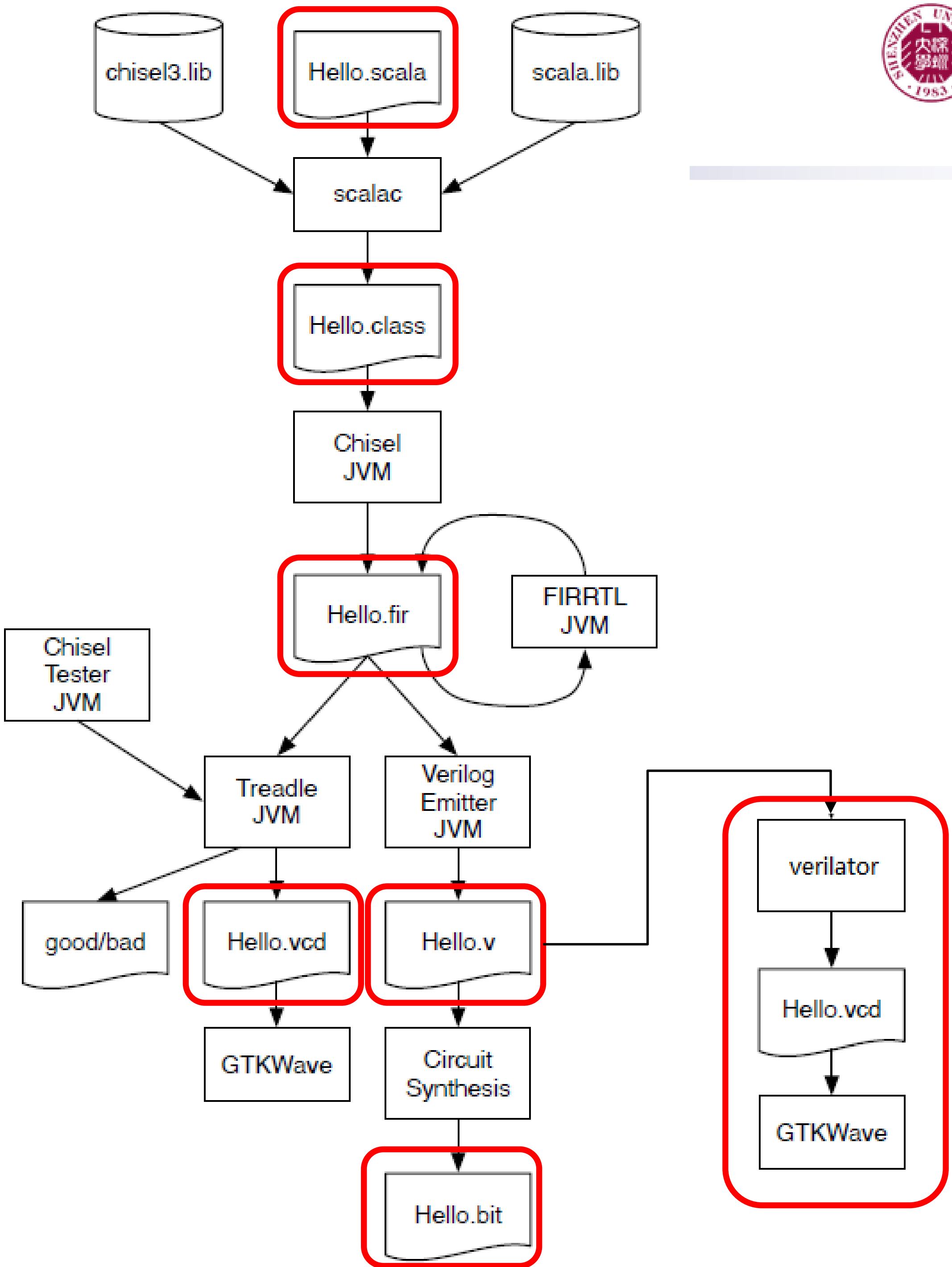
```
package examples
import chisel3._
class FullAdder extends Module {
        val io = IO(     new Bundle {
                val a     = Input(UInt(1.W))
                val b     = Input(UInt(1.W))
                val cin  = Input(UInt(1.W))
                val sum  = Output(UInt(1.W))
                val cout = Output(UInt(1.W))  }        )

        val s1= io.a ^ io.b
        io.sum := s1 ^ io.cin
        val s3 = io.a & io.b
        val s2 = s1 & io.cin
        io.cout := s2| s3
}
```



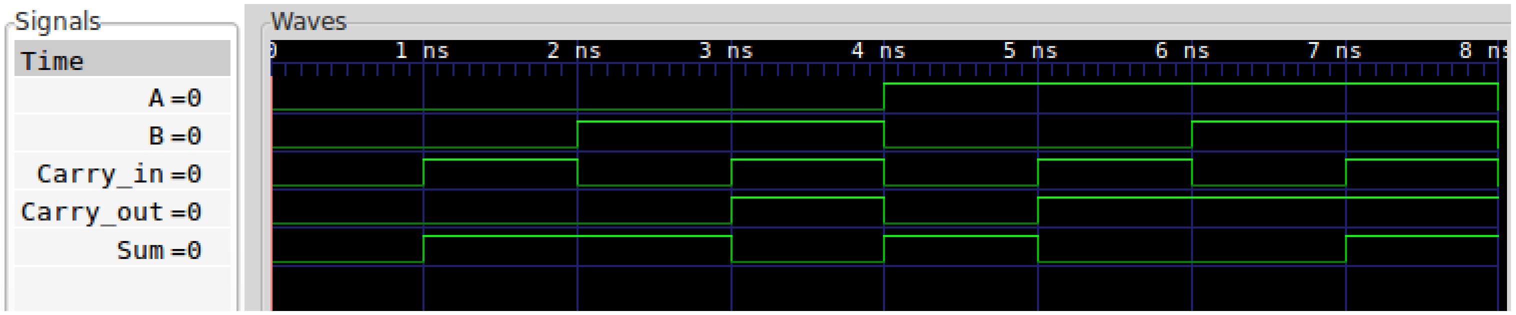
1.3 Chisel/FIRRTL

- Chisel/FIRRTL
  - https://www.chisel-lang.org/

Chisel/FIRRTL Hardware Compiler Framework


chisel3.lib
Hello.scala
scala.lib
scalac
Hello.class
Chisel JVM
Hello.fir
FIRRTL JVM
Chisel Tester JVM
Treadle JVM
Verilog Emitter JVM
verilator
good/bad
Hello.vcd
Hello.v
Hello.vcd
GTKWave
Circuit Synthesis
GTKWave
Hello.bit

仿真波形——Chisel 1-bit 全加器



Chisel生成Verilog/Firrtl示例

```
package my.hello
import chisel3._
class ModuleSample extends Module {
        val io = IO(new Bundle {
                        val in_a = Input(SInt(4.W))
                        val out_b = Output(SInt(4.W))
        })

        io.out_b := io.in_a
}

object MyModule extends App {
  emitVerilog (new ModuleSample(), Array("--target-dir", "generated"))
}
```

```
generated/
├── MyModule.v
├── MyModule.fir
├── MyModule.anno.json
```

1.4 Chisel安装与运行

- 开发环境
  - 操作系统　Linux (Ubuntu 22.04)
  - 语言环境　Scala/Java
    - Chisel只是Scala的一个库
  - 仿真工具　Verilator(Verilog仿真)
  - 波形查看　GTKWave (波形查看)
  - FPGA开发工具　Vivado (Xilinx)

Java>sbt安装

- Java安装：

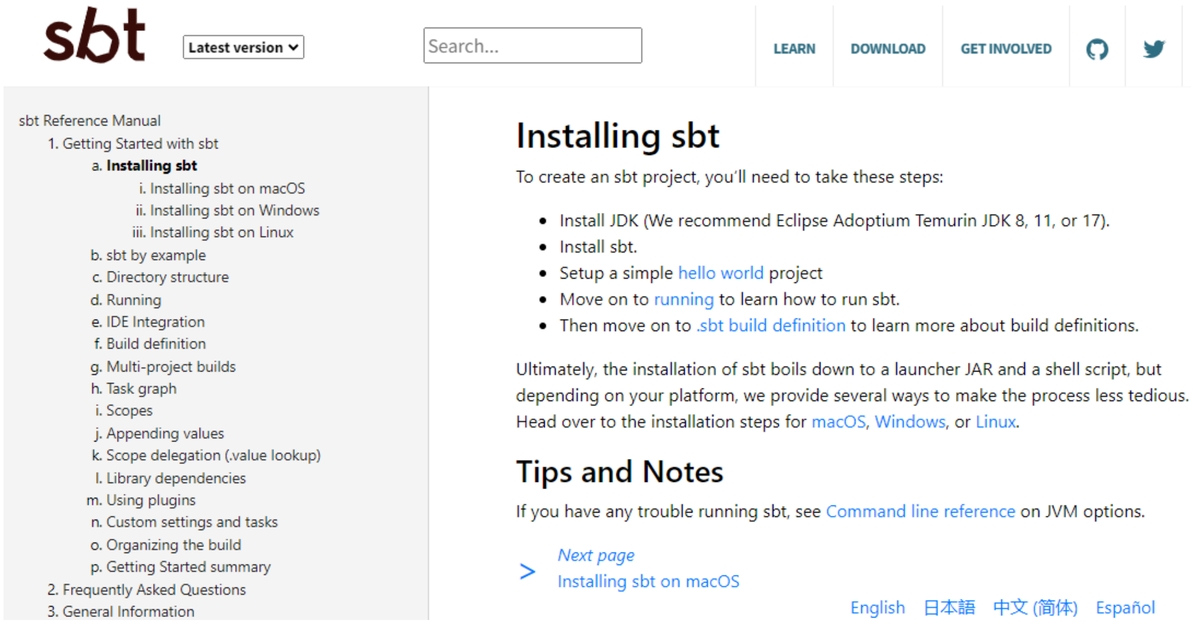
  sudo apt-get install default-jdk

- sbt安装：(https://www.scala-sbt.org/1.x/docs/Setup.html)

  安装必要组件

  添加sbt源

  安装



sbt Reference Manual
1. Getting Started with sbt
   a. **Installing sbt**
      i. Installing sbt on macOS
      ii. Installing sbt on Windows
      iii. Installing sbt on Linux
   b. sbt by example
   c. Directory structure
   d. Running
   e. IDE Integration
   f. Build definition
   g. Multi-project builds
   h. Task graph
   i. Scopes
   j. Appending values
   k. Scope delegation (.value lookup)
   l. Library dependencies
   m. Using plugins
   n. Custom settings and tasks
   o. Organizing the build
   p. Getting Started summary
2. Frequently Asked Questions
3. General Information

Installing sbt

To create an sbt project, you'll need to take these steps:

- Install JDK (We recommend Eclipse Adoptium Temurin JDK 8, 11, or 17).
- Install sbt.
- Setup a simple hello world project
- Move on to running to learn how to run sbt.
- Then move on to .sbt build definition to learn more about build definitions.

Ultimately, the installation of sbt boils down to a launcher JAR and a shell script, but depending on your platform, we provide several ways to make the process less tedious. Head over to the installation steps for macOS, Windows, or Linux.

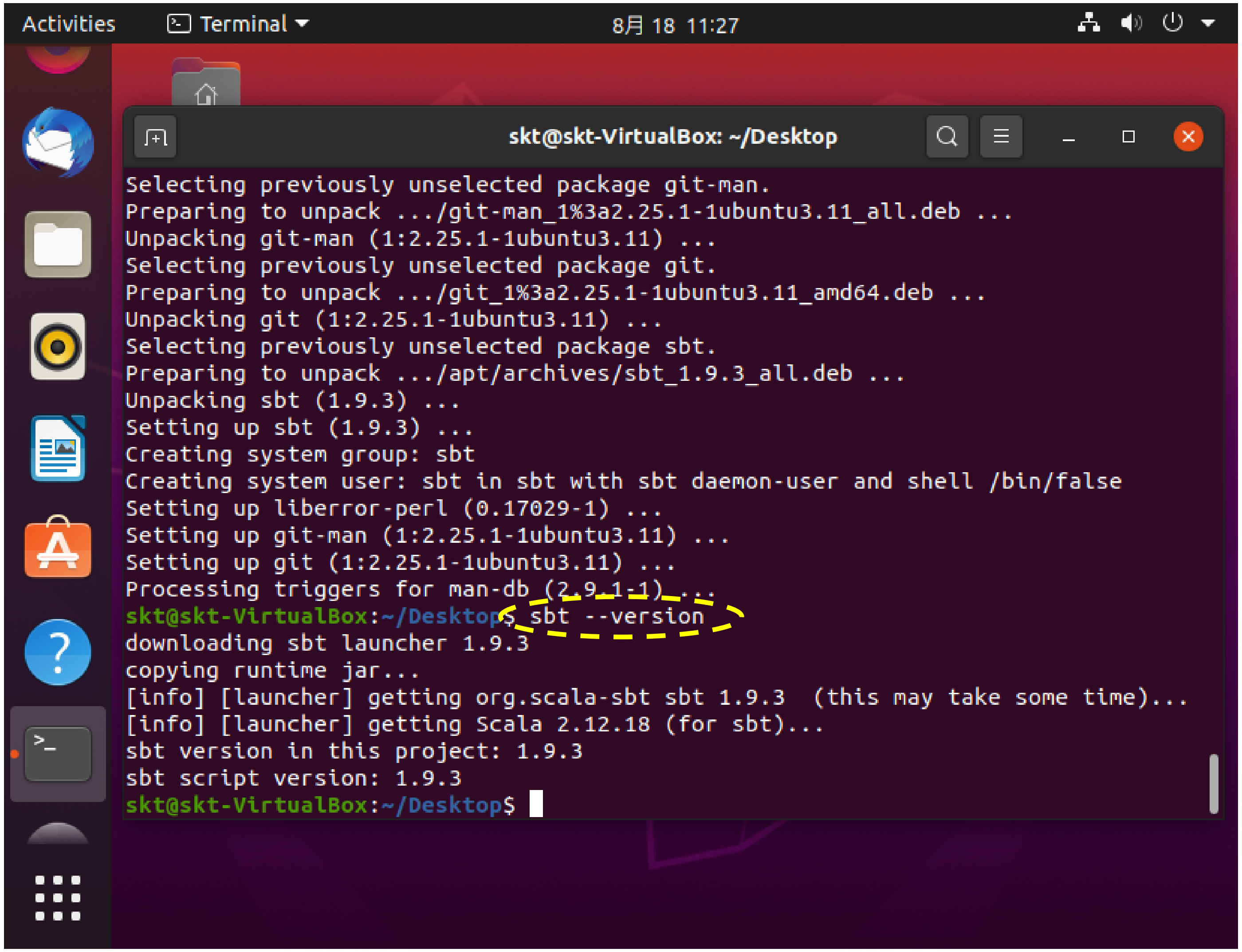
Tips and Notes

If you have any trouble running sbt, see Command line reference on JVM options.

> *Next page*
Installing sbt on macOS

English 日本語 中文 (简体) Español

```
sudo apt-get update
sudo apt-get install apt-transport-https curl gnupg -yqq

echo "deb https://repo.scala-sbt.org/scalasbt/debian all main" | sudo tee
/etc/apt/sources.list.d/sbt.list
echo "deb https://repo.scala-sbt.org/scalasbt/debian /" | sudo tee
/etc/apt/sources.list.d/sbt_old.list

curl -sL
"https://keyserver.ubuntu.com/pks/lookup?op=get&search=0x2EE0EA64E40
A89B84B2DF73499E82A75642AC823" | sudo -H gpg --no-default-keyring --
keyring gnupg-ring:/etc/apt/trusted.gpg.d/scalasbt-release.gpg --import

sudo chmod 644 /etc/apt/trusted.gpg.d/scalasbt-release.gpg

sudo apt-get update
sudo apt-get install sbt
```



```
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.25.1-1ubuntu3.11_all.deb ...
Unpacking git-man (1:2.25.1-1ubuntu3.11) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.25.1-1ubuntu3.11_amd64.deb ...
Unpacking git (1:2.25.1-1ubuntu3.11) ...
Selecting previously unselected package sbt.
Preparing to unpack .../apt/archives/sbt_1.9.3_all.deb ...
Unpacking sbt (1.9.3) ...
Setting up sbt (1.9.3) ...
Creating system group: sbt
Creating system user: sbt in sbt with sbt daemon-user and shell /bin/false
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.25.1-1ubuntu3.11) ...
Setting up git (1:2.25.1-1ubuntu3.11) ...
Processing triggers for man-db (2.9.1-1) ...
skt@skt-VirtualBox:~/Desktop$ sbt --version
downloading sbt launcher 1.9.3
copying runtime jar...
[info] [launcher] getting org.scala-sbt sbt 1.9.3  (this may take some time)...
[info] [launcher] getting Scala 2.12.18 (for sbt)...
sbt version in this project: 1.9.3
sbt script version: 1.9.3
skt@skt-VirtualBox:~/Desktop$
```

Chisel示例——LED闪烁

- 克隆chisel示例工程
  - git clone https://github.com/schoeberl/chisel-examples.git

```
skt@skt-VirtualBox:~/Desktop/chisel-examples$ ls
bubble.gtkw   hello-world   project    src                  vhdl
build.sbt     LICENSE       quartus    target               vivado
generated     Makefile      README.md  TowardsChisel3.md
skt@skt-VirtualBox:~/Desktop/chisel-examples$ ls hello-world/
build.sbt  Makefile  quartus  README.md  src  verilog  vivado
skt@skt-VirtualBox:~/Desktop/chisel-examples$ tree hello-world/src
hello-world/src
├── main
│   └── scala
│       └── Hello.scala
└── test
    └── scala
        └── HelloTest.scala

4 directories, 2 files
skt@skt-VirtualBox:~/Desktop/chisel-examples$
```

```
import chisel3._

class Hello extends Module {
    val io = IO(new Bundle {
    val led = Output(UInt(1.W))
  })
  val CNT_MAX = (50000000 / 2 - 1).U

  val cntReg = RegInit(0.U(32.W))
  val blkReg = RegInit(0.U(1.W))

  cntReg := cntReg + 1.U
  when(cntReg === CNT_MAX) {
     cntReg := 0.U
     blkReg := ~blkReg
  }
  io.led := blkReg
}

object Hello extends App {
  (new chisel3.stage.ChiselStage).emitVerilog(new Hello(),
              Array("--target-dir", "generated"))
}
```

$\left.\right\}$ 电路/器件
接口声明

$\left.\right\}$ 功能逻辑实现

$\left.\right\}$ 输出Verilog文件

新建自己的工程

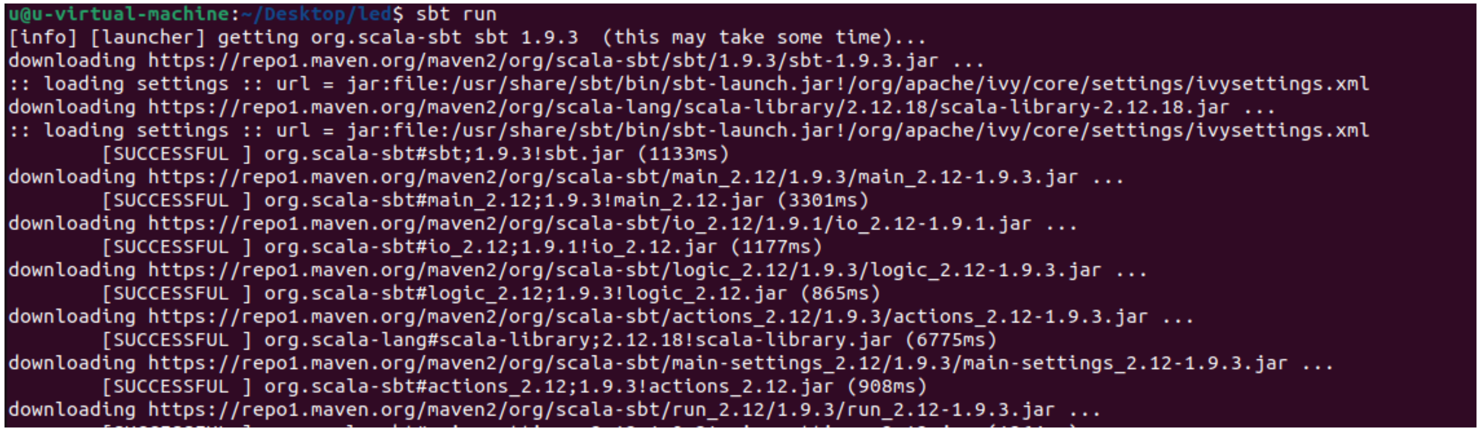
- 创建工程目录、从样例工程拷贝

  build.sbt、Hello.scala和test.scala

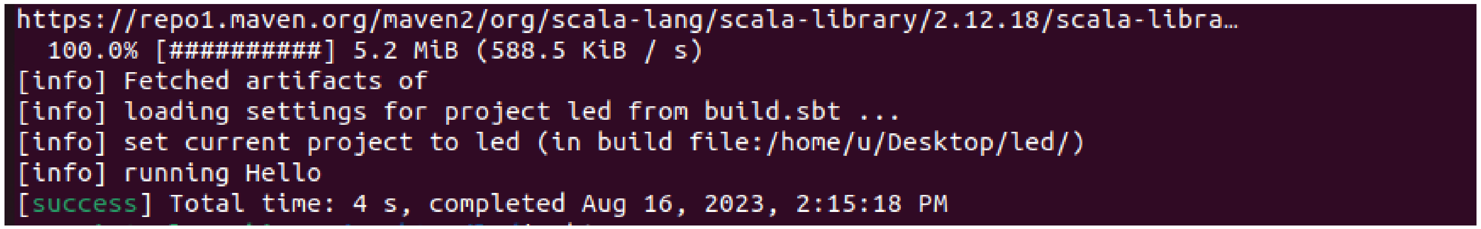
```
skt@skt-VirtualBox:~/Desktop$ tree myLED
myLED
├── build.sbt
└── src
    ├── main
    │   └── scala
    │       └── Hello.scala
    └── test
        └── scala
            └── HelloTest.scala
5 directories, 3 files
skt@skt-VirtualBox:~/Desktop$
```

编译运行并输出Verilog

- 在工程根目录下启动终端执行：sbt run



```
u@u-virtual-machine:~/Desktop/led$ sbt run
[info] [launcher] getting org.scala-sbt sbt 1.9.3  (this may take some time)...
downloading https://repo1.maven.org/maven2/org/scala-sbt/sbt/1.9.3/sbt-1.9.3.jar ...
:: loading settings :: url = jar:file:/usr/share/sbt/bin/sbt-launch.jar!/org/apache/ivy/core/settings/ivysettings.xml
downloading https://repo1.maven.org/maven2/org/scala-lang/scala-library/2.12.18/scala-library-2.12.18.jar ...
:: loading settings :: url = jar:file:/usr/share/sbt/bin/sbt-launch.jar!/org/apache/ivy/core/settings/ivysettings.xml
        [SUCCESSFUL ] org.scala-sbt#sbt;1.9.3!sbt.jar (1133ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/main_2.12/1.9.3/main_2.12-1.9.3.jar ...
        [SUCCESSFUL ] org.scala-sbt#main_2.12;1.9.3!main_2.12.jar (3301ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/io_2.12/1.9.1/io_2.12-1.9.1.jar ...
        [SUCCESSFUL ] org.scala-sbt#io_2.12;1.9.1!io_2.12.jar (1177ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/logic_2.12/1.9.3/logic_2.12-1.9.3.jar ...
        [SUCCESSFUL ] org.scala-sbt#logic_2.12;1.9.3!logic_2.12.jar (865ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/actions_2.12/1.9.3/actions_2.12-1.9.3.jar ...
        [SUCCESSFUL ] org.scala-lang#scala-library;2.12.18!scala-library.jar (6775ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/main-settings_2.12/1.9.3/main-settings_2.12-1.9.3.jar ...
        [SUCCESSFUL ] org.scala-sbt#actions_2.12;1.9.3!actions_2.12.jar (908ms)
downloading https://repo1.maven.org/maven2/org/scala-sbt/run_2.12/1.9.3/run_2.12-1.9.3.jar ...
```



```
https://repo1.maven.org/maven2/org/scala-lang/scala-library/2.12.18/scala-libra…
  100.0% [##########] 5.2 MiB (588.5 KiB / s)
[info] Fetched artifacts of
[info] loading settings for project led from build.sbt ...
[info] set current project to led (in build file:/home/u/Desktop/led/)
[info] running Hello
[success] Total time: 4 s, completed Aug 16, 2023, 2:15:18 PM
```

- 输出的verilog文件在generated目录下

```
skt@skt-VirtualBox:~/Desktop/myLED$ ls
build.sbt  generated  project  src  target
skt@skt-VirtualBox:~/Desktop/myLED$ ls generated/
Hello.anno.json  Hello.fir  Hello.v
skt@skt-VirtualBox:~/Desktop/myLED$
```

功能仿真/验证

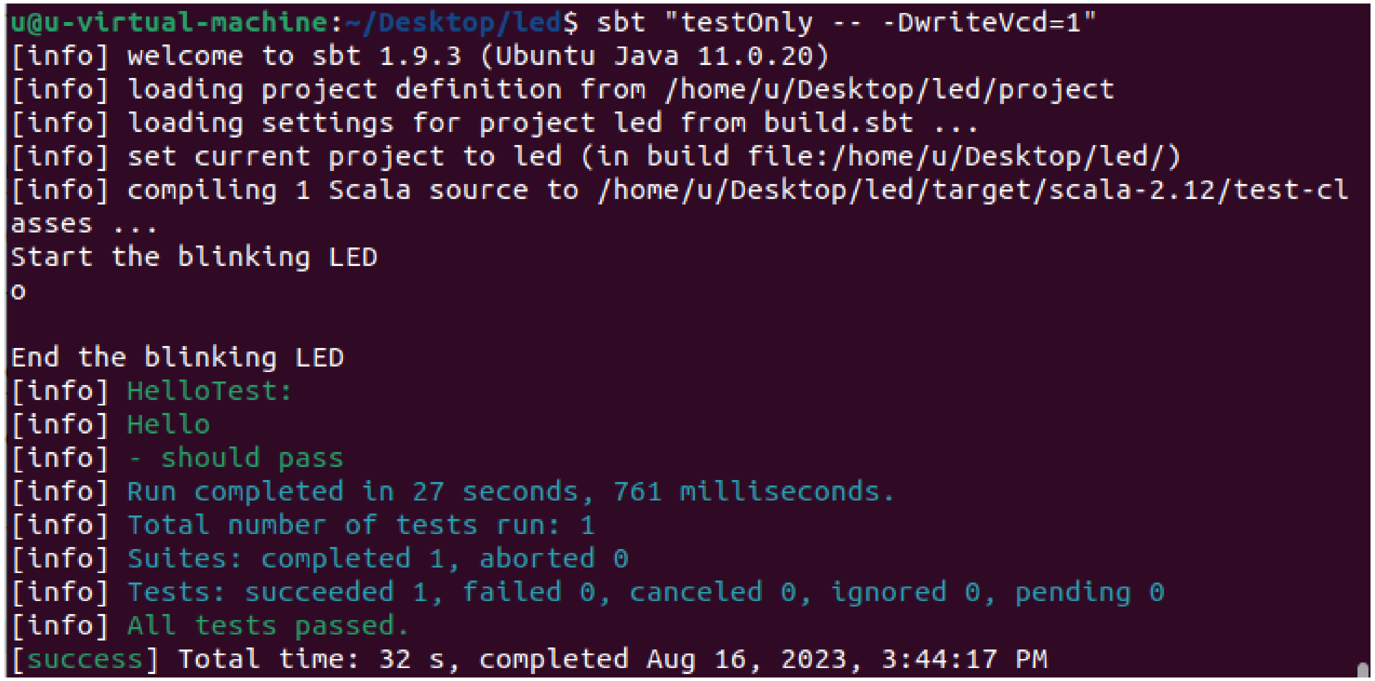
- 工具：Chiseltest
- 仿真过程：
  - 编写测试程序
    - 创建scr/test/scala/目录
    - 编写测试源代码
  - 运行测试程序并输出.vcd波形文件
  - 用gtkwave软件观测波形

测试代码示例

```
import chiseltest._
import org.scalatest.flatspec.AnyFlatSpec
class HelloTest extends AnyFlatSpec with ChiselScalatestTester {
    behavior of "Hello"
    it should "pass" in {
        test(new Hello).withAnnotations(Seq(WriteVcdAnnotation)) { c =>
            c.clock.setTimeout(0)
            var ledStatus = BigInt(-1)
            println("Start the blinking LED")
            for (_ <- 0 until 100) {
                c.clock.step(10000)
                val ledNow = c.io.led.peek().litValue
                val s = if (ledNow == 0) "o" else "*"
                if (ledStatus != ledNow) {
                    System.out.println(s)
                    ledStatus = ledNow
                }
            }
            println("\n End the blinking LED")
        }
    }
}
```

运行测试程序

- 执行：sbt test 或 sbt “testOnly -- -DwriteVcd=1”



```
u@u-virtual-machine:~/Desktop/led$ sbt "testOnly -- -DwriteVcd=1"
[info] welcome to sbt 1.9.3 (Ubuntu Java 11.0.20)
[info] loading project definition from /home/u/Desktop/led/project
[info] loading settings for project led from build.sbt ...
[info] set current project to led (in build file:/home/u/Desktop/led/)
[info] compiling 1 Scala source to /home/u/Desktop/led/target/scala-2.12/test-cl
asses ...
Start the blinking LED
o

End the blinking LED
[info] HelloTest:
[info] Hello
[info] - should pass
[info] Run completed in 27 seconds, 761 milliseconds.
[info] Total number of tests run: 1
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 1, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 32 s, completed Aug 16, 2023, 3:44:17 PM
```

- 输出的波形文件目录：
  - test_run_dir/ ModuleSample_should_pass

GTKWave观测波形

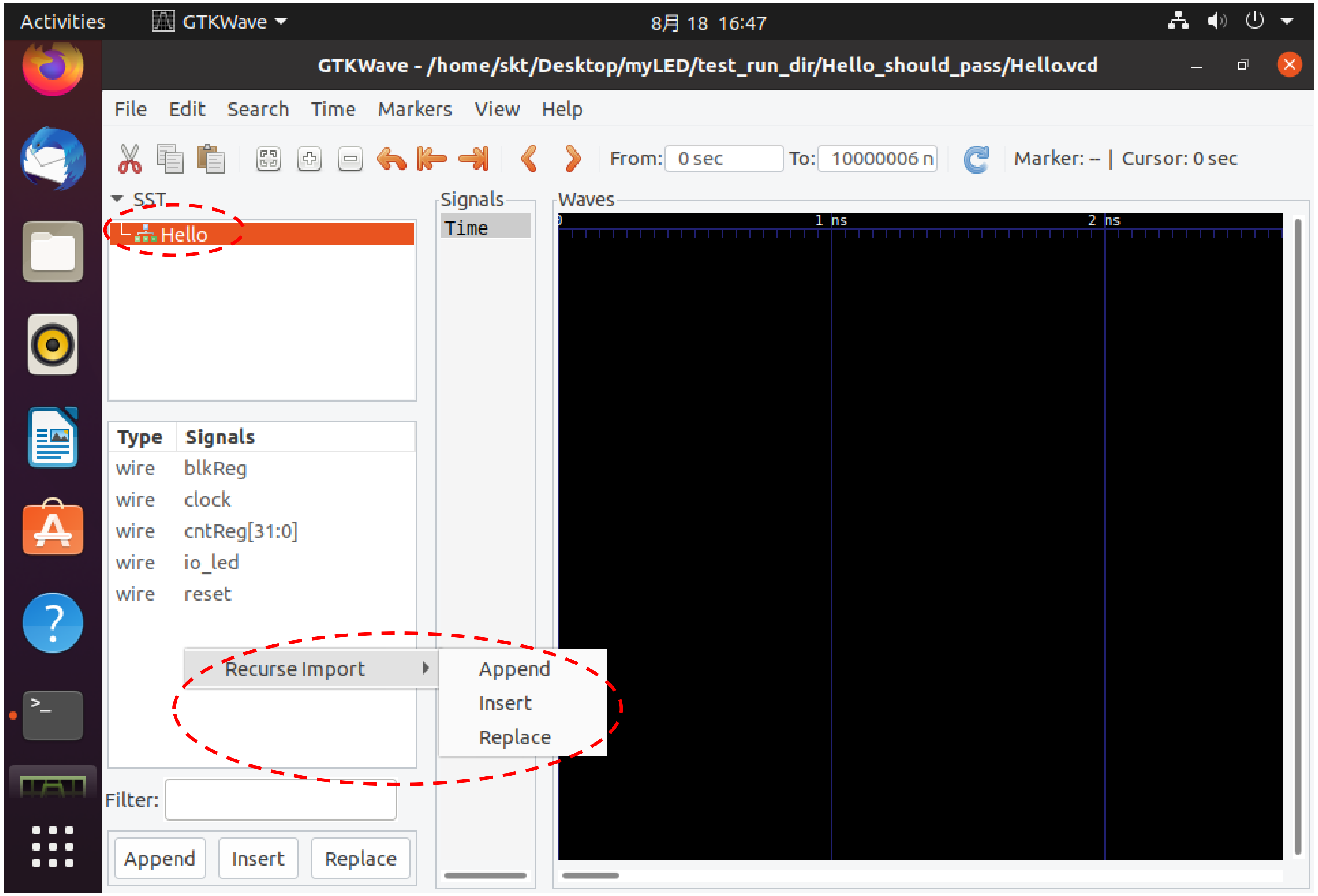
- 安装GTKWave

```
sudo apt-get install gtkwave
```

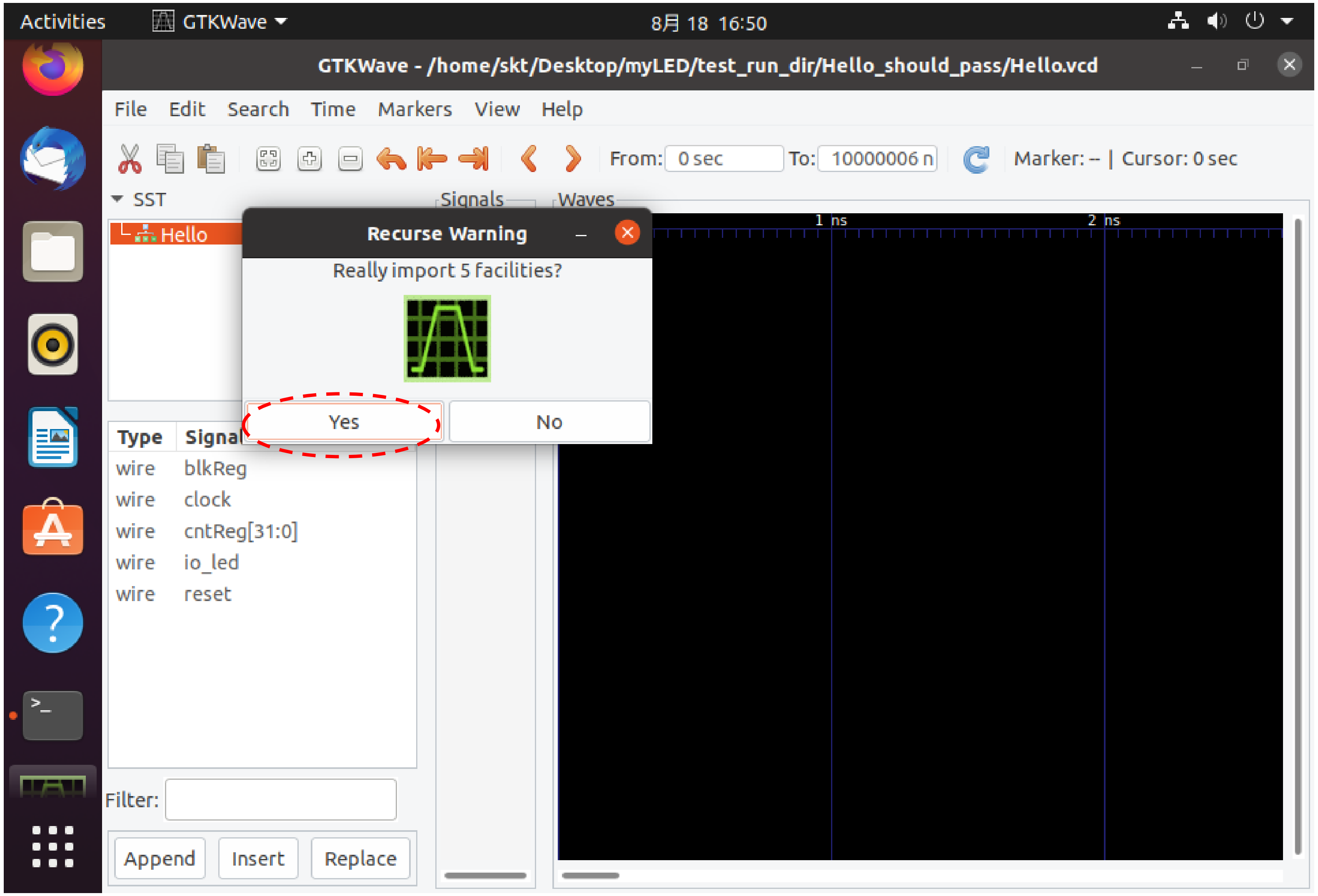
- 载入波形文件

  File -> Open New Tab 选择Hello.vc

- 选择所需观察的信号

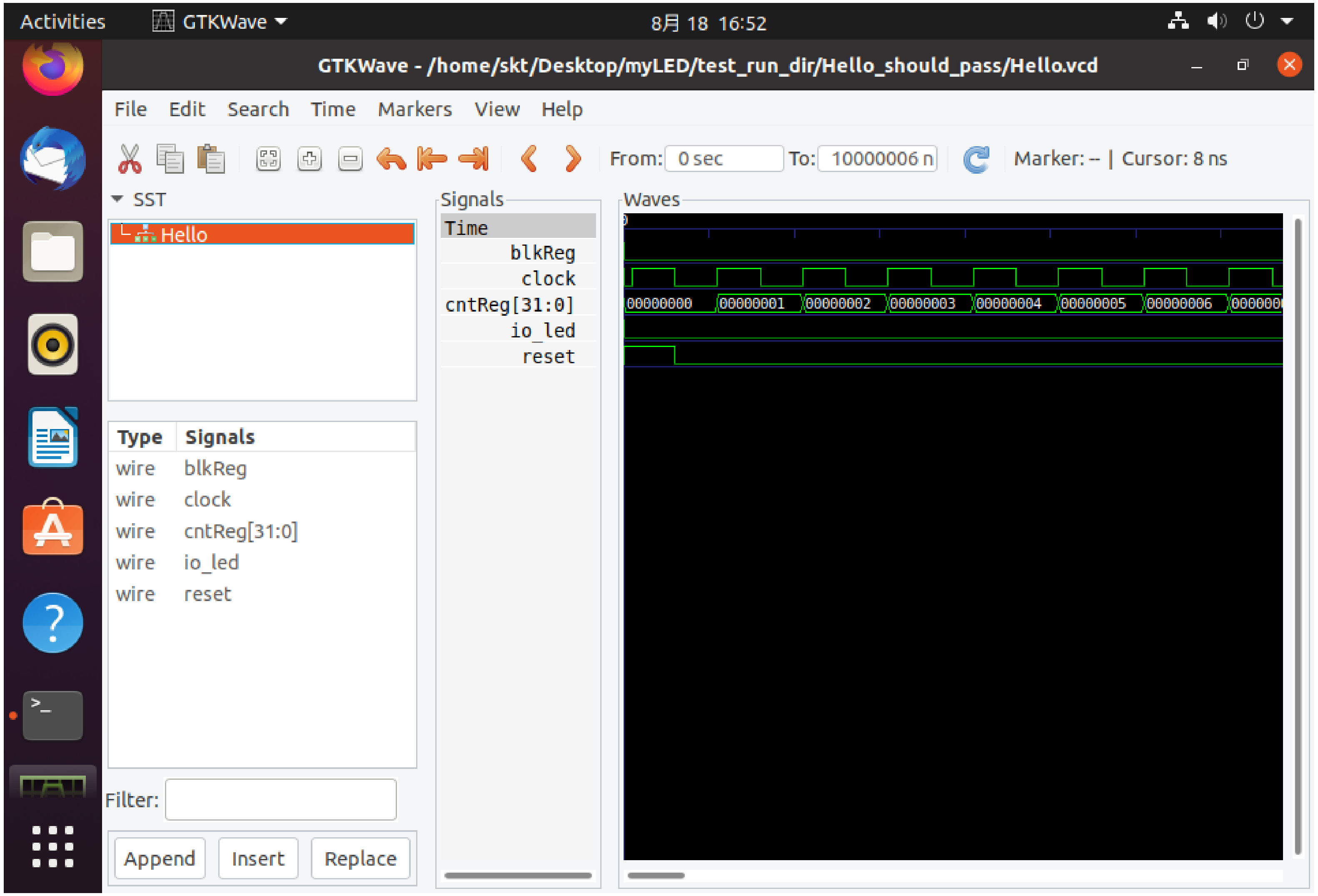

Activities
GTKWave
8月 18 16:47
GTKWave - /home/skt/Desktop/myLED/test_run_dir/Hello_should_pass/Hello.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 10000006 n
Marker: -- | Cursor: 0 sec
SST
Hello
Signals
Time
Waves
1 ns
2 ns
Recurse Import
Append
Insert
Replace
Filter:
Append Insert Replace

Type	Signals
wire	blkReg
wire	clock
wire	cntReg[31:0]
wire	io_led
wire	reset



Really import 5 facilities?

Type	Signal
wire	blkReg
wire	clock
wire	cntReg[31:0]
wire	io_led
wire	reset

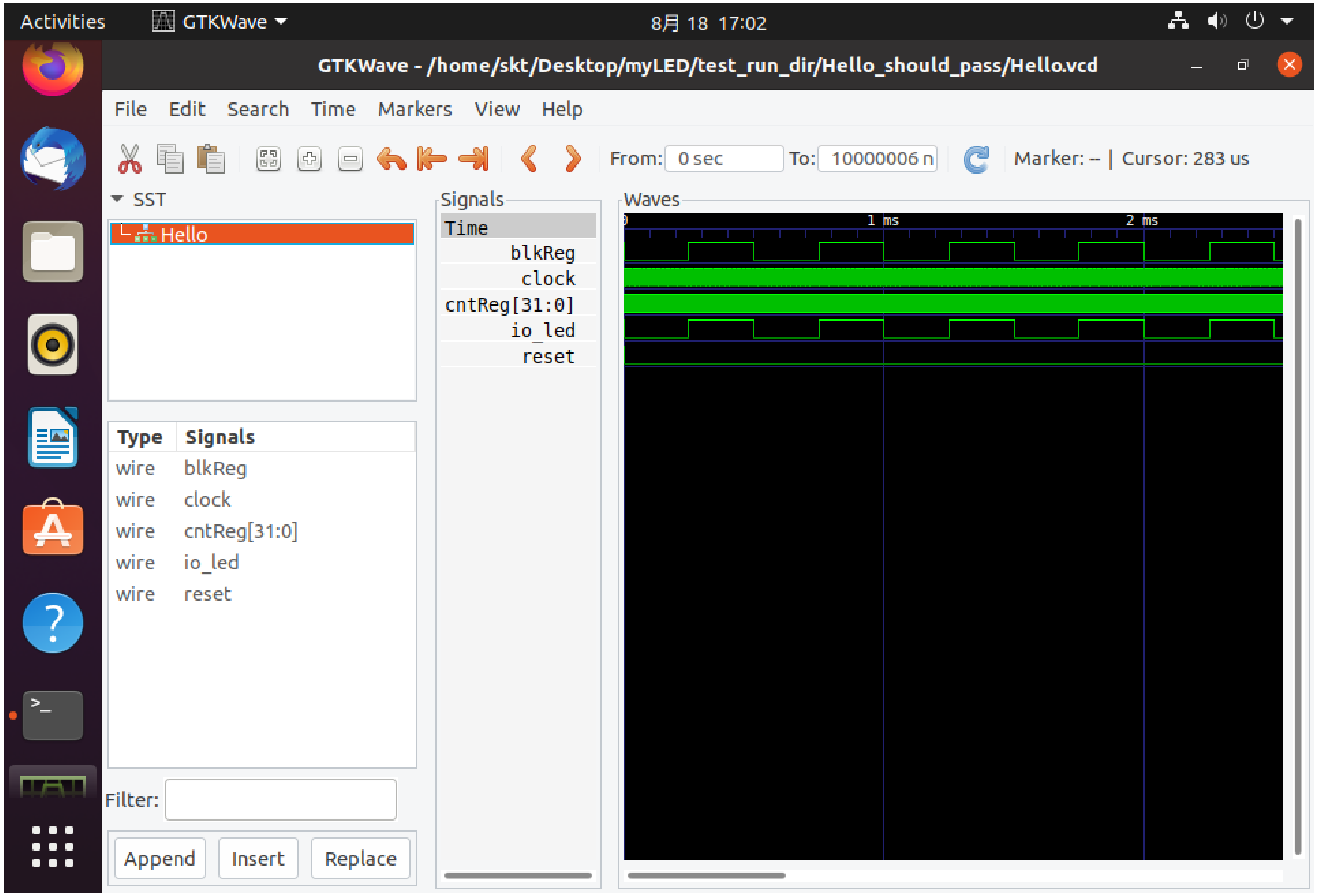


Type	Signals
wire	blkReg
wire	clock
wire	cntReg[31:0]
wire	io_led
wire	reset

```
Time
blkReg
clock
cntReg[31:0]   00000000 00000001 00000002 00000003 00000004 00000005 00000006 0000000
io_led
reset
```

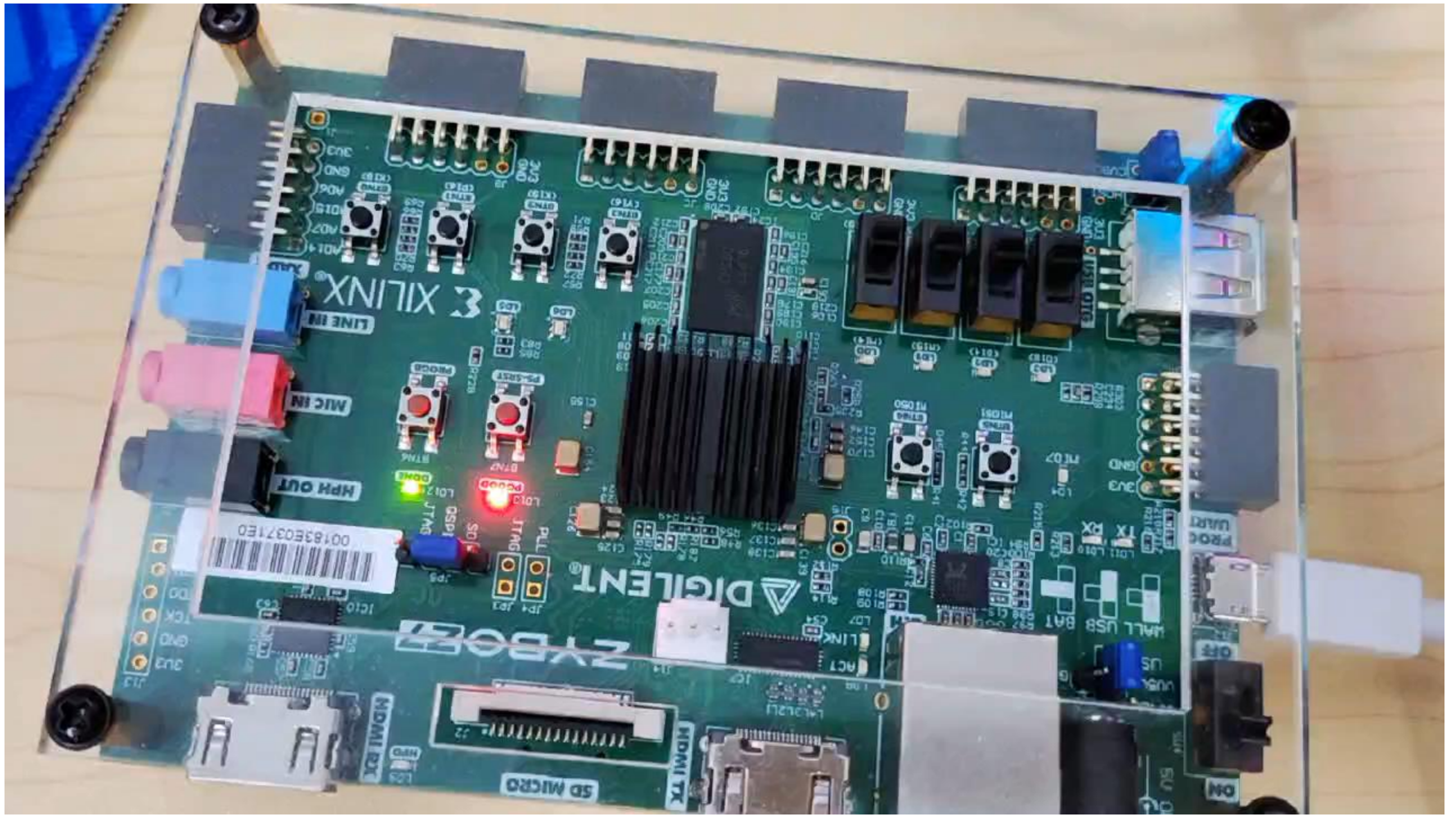
```
class ModuleSample extends Module {
        val io = IO(new Bundle {
        val led = Output(UInt(1.W))
        })

        val CNT_MAX = (50000000 / 2 - 1).U;
        val cntReg = RegInit(0.U(32.W))
        val blkReg = RegInit(0.U(1.W))
        cntReg := cntReg + 1.U
        when(cntReg === CNT_MAX) {
                cntReg := 0.U
                blkReg := ~blkReg
                }
        io.led := blkReg
}
```

```
class ModuleSample extends Module {
        val io = IO(new Bundle {
        val led = Output(UInt(1.W))
        })

        val CNT_MAX = (50000 / 2 - 1).U;
        val cntReg = RegInit(0.U(32.W))
        val blkReg = RegInit(0.U(1.W))
        cntReg := cntReg + 1.U
        when(cntReg === CNT_MAX) {
                cntReg := 0.U
                blkReg := ~blkReg
                }
        io.led := blkReg
}
```



Type	Signals
wire	blkReg
wire	clock
wire	cntReg[31:0]
wire	io_led
wire	reset

FPGA实现


XILINX
DIGILENT
ZYBO Z7
LINE IN
MIC IN
HPH OUT
HDMI RX
HDMI TX
SD MICRO