

# MCDF 实验 1

## TB1. 从 Verilog 到 SV 的进场

同学们可以先下载 `tb1.v` 文件，该文件同上一次实验 0 的测试文件 `tb1.v` 的内容是一致的。接下来，同学们需要将 `tb1.v` 的文件名修改为 `tb1.sv`，即将文件后缀修改为 `.sv`。接下来就要开始我们的实验要求啦！

### 要求 1.1

在修改为 `tb1.sv` 之后，同学们可以按照之前的步骤，编译仿真，查看仿真行为是否同 `tb1.v` 的仿真行为一致？这说明了什么呢？

### 要求 1.2

同学们可以将 `tb1.sv` 中的信号变量类型由 `reg` 或者 `wire` 修改为 `logic` 类型，再编译仿真，查看行为是否同修改之前的一致呢？这是为什么？

### 要求 1.3

在步骤 2 的基础上，如果同学们将 `rstn` 的类型由 `logic` 修改为 `bit` 类型，再编译仿真，行为是否同步骤 2 的一致呢？这是为什么？

## TB2. 方法 task 和函数 function

同学们在 `tb2.sv` 文件中，可以看到不同于 `tb1.sv` 文件的是，之前产生时钟和发起复位的两个 `initial` 过程块语句都被两个 `task` 即 `clk_gen()` 和 `rstn_gen()` 取代了。接下来同学们请完成实验部分：

### 要求 2.1

不做修改的情况下，对 `tb2.sv` 进行编译仿真，时钟信号和复位信号还正常吗？为什么？

### 要求 2.2

同学们在路桑标记的两个 `initial` 块中分别调用产生时钟和复位的 `task`，再编译仿真查看时钟信号和复位信号，是否恢复正常呢？

### 要求 2.3

为什么要将两个 `task` 在两个 `initial` 块中调用？这是为什么呢？是否可以在一个 `initial` 块中调用呢？如果可以，调用它们的顺序是什么？

### 要求 2.4

同学们是否可以读出目前时钟的周期和频率呢？该如何测量呢？如果我们想进化 `clk_gen()` 方法，使其变为可以设置时钟周期的任务 `ckgen(intperoid)`，那么该怎么修改目前的任务 `clk_gen()` 呢？修改成功后，请在 `initial` 块中调用任务 `cdk_gen(20)`，看看波形中的时钟周期是否变为 20ns 呢？

## 要求 2.5

如果将 `timescale 1ns/1ps` 修改为 `timescale 1ps/1ps`，那么仿真中的时钟周期是否发生变化?这是为什么呢?

## TB3. 数组的使用

同学们在实验 0 环境中有没有对“`data_test`”部分比较好奇呢?譬如为什么要产生这么多的数据?而如果要发起激励的时候，每个 `slave` 的数据之间有什么相同和不同的地方呢?接下来我们会使用 `tb3.sv` 实验文件，对数组类型多加练习吧!

### 要求 3.1

如果路桑现在要求同学们对每一个 `slave` 的数据发出 100 个数，那么大家该怎么实现呢?请按照你的方式实现吧!

### 要求 3.2

如果我们现在要先生成 100 个数，并对它们按照日前的数值规则进行赋值，那么请同学们创建 3 个动态数组，分别放置要发送给 3 个 `slave` 的数据。

### 要求 3.3

接下来我们利用之前生成的数组数据，将它们读取并发送给三个 `channel`。

## TB3. 验证结构

为了实现清晰的验证结构，我们希望将 DUT 和激励发生器(`stimulator`)之间划分。因此，我们可以将激励方法 `chnl_write()`封装在新的模块 `lhn1_initiator` 中，请同学们下载 `tb4.sv`，在接下来开展实验步骤前，同学们可以将“数组的使用”环节中添加的代码部分移柏到 `tb4.sv` 对应的位置上。

从 `tb4.sv` 中同学们可以发现之前的 `initil` 语句块“`channel_write_task`”已经不见了，在其位置上的变为了三个例化的 `shnl_initiator` 实例 `chnl0_init`、`chnl1_init` 和 `chnl2_init`。它们的作用扮演每个 `channel_slave` 通道对应的 `stimulator`，发送激励，因此我们在其模块 `chnl_initiator` 中定义了它的三个方法，即 `set_name()`、`chpl_write()`和 `chnl_idle()`。

### 要求 4.1

`chnl_idle()` 要实现的一个时钟周期的空闲，在该周期中，`ch.valid` 应为低，`ch.data` 应为 0。

### 要求 4.2

`Chnl_write()`要实现一次有效的写数据，并随后调用 `chnl_idle()`，实现一个空闲周期，在实现有效写数据时，请同学们考虑如何使用 `ch_ready` 信号，结合功能描述的 `channel_slave` 接口时序来看，只有当 `valid` 为高且 `ready` 为高时，数据写入才算成功，如果此时 `ready` 为低，那么则应该保持数据和 `valid` 信号，直到 `ready` 拉高时，数据写入才算成功。

### 要求 4.3

`set_name()`即设置实例的名称，在 `initial` 过程块 “`data_test`”中，在发送各个 `channel` 数据前，请设置各个 `channel_initiator` 的实例名称，这样方便在仿真时各个实例的打印信息可以显示它们各自的名称、数据发送时间和数据内容，便于阅读和调试。

### 要求 4.4

最后，同学们进入了本次实验的最后一个步骤了，路桑之所以提出发送更多的数据，并发送更紧凑高速的数据，是为了同学们可以观察到，是否你的二个 `channel_slave` 各自的 `chX_ready` 信号可以拉低呢？如果拉低了，这代表着什么？那么请你试试看，考虑如何发送更多更快的数据，让 `MCDT` 的三个 `chx_ready` 信号可以拉低吧。