# V1系列课程实验0

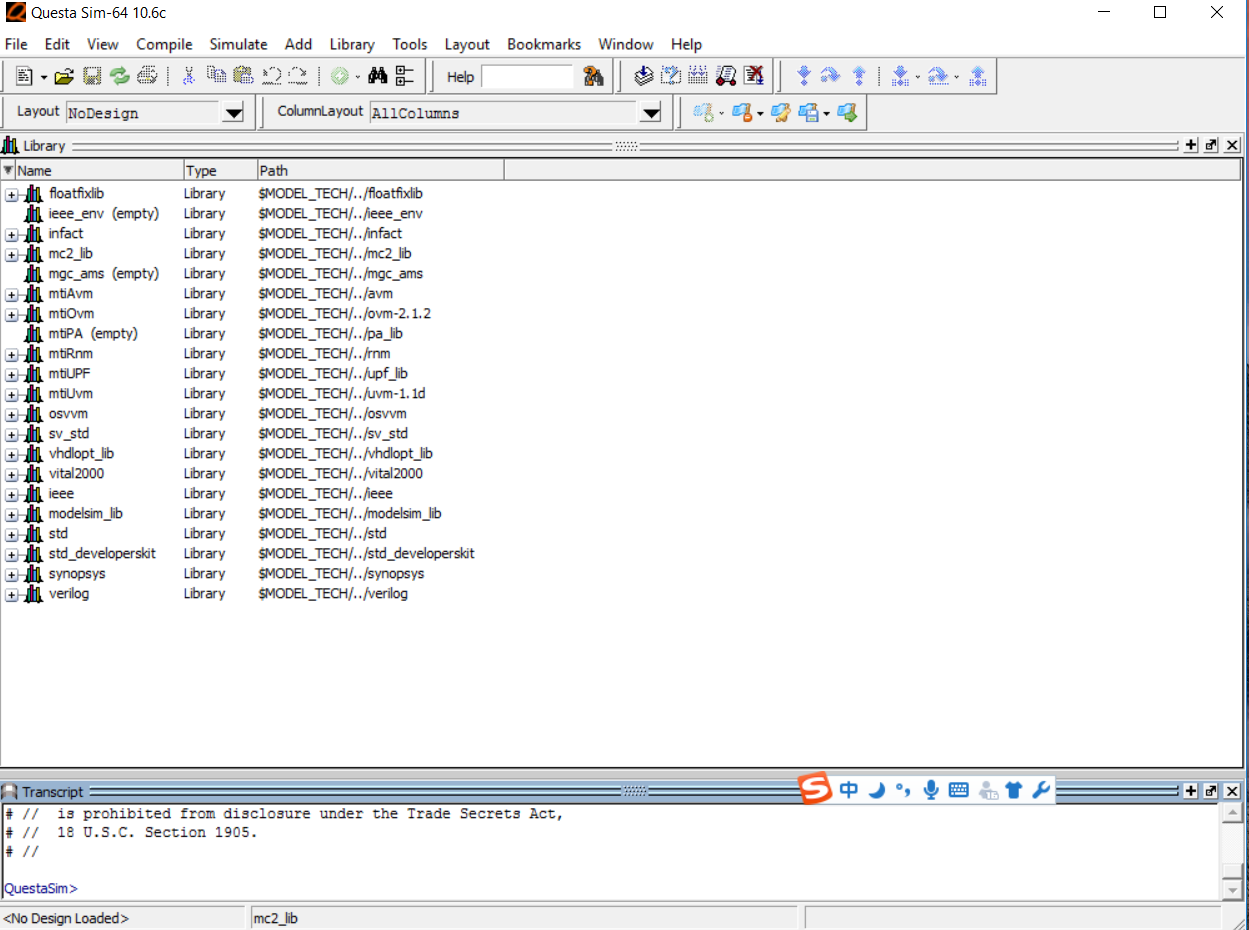
Hi，欢迎进入芯片验证V1系列课程的实验部分。实验0会帮助大家认识什么是设计功能描述文档，让大家在理解设计描述的同时，可以结合设计文件（Verilog实现），在展开实验之前可以一方面回顾Verilog的知识，另外一方面认识MCDF的结构（V2课程及红宝书的待测设计）。本次实验重点在于帮助大家实现这些实验前的要求：

* 安装仿真工具Questasim。不建议安装Modelsim或者Quatus/ISE等仿真工具。一方面Questasim对SV语言的支持很好，另外一方面实验需要同样的工具，保持大家在实验环境的一致性。
* 为了简易期间，路桑不对大家的代码编辑器提出要求，同学们可以在Questasim自带的编辑窗口中编辑代码，或者使用其他代码编辑器。
* 在实验综合练习环节，路桑会带领大家一同阅读可能是你人生中的第一份“硬件设计功能描述”（hardware design function specification）文档。虽然MCDF并不作为V1实验的DUT，但是了解它对于接下来你理解MCDT的功能会有帮助哦。
* 了解MCDF的Verilog设计和各个模块之间的功能、连接和交互，再来了解MCDT的功能。我们接下来的实验也将围绕MCDT运用SV所学知识逐步升级验证环境。

## Questasim仿真器安装

工具安装资源和具体步骤在云协作，在安装时同学们需要注意以下几点：

* 分清楚你的机器是64位机器还是32机器，就最近几年的笔记本而言，64位机器已经是普遍情况，而安装时也需要注意选择64位安装包还是32位安装包。
* 注册机制需要按照云协作中的步骤，重点主要在于WINDOWS环境变量设置环节。
* 正确安装之后，运行Questasim之后不会再出现“注册或者试用”的软件声明，那么恭喜你，实验所需要的唯一软件就已经安装成功喽！



## MCDF功能描述

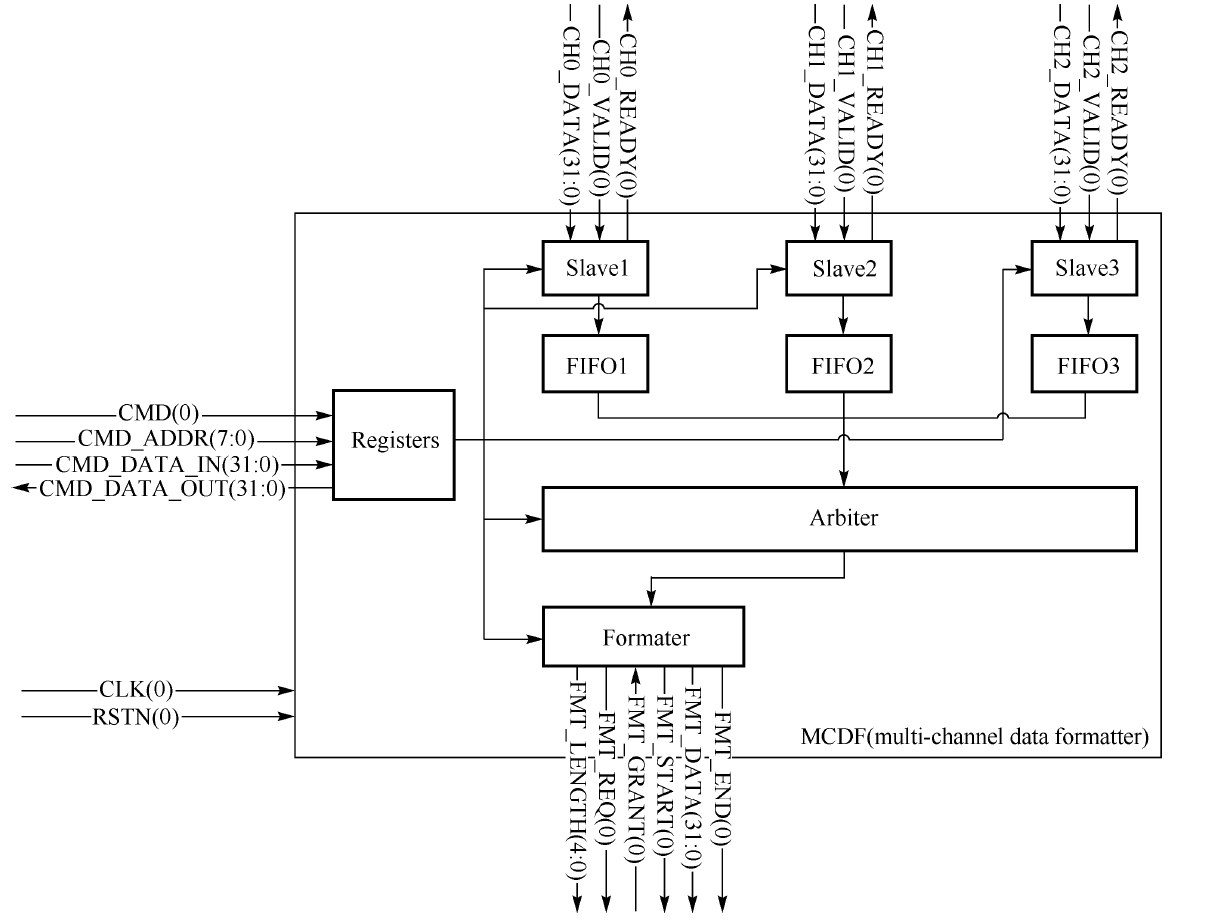
为了模拟实际情景，我们给出贯穿于V1和V2课程及实验的硬件设计MCDF，并且遵循硬件设计描述的方式，介绍它的结构、功能、寄存器和时序。在以后的SV部分中，我们也将围绕这个硬件设计来考虑测试平台的构成。日后对测试平台的构建论述，需要经常引用MCDF的功能描述，请同学们对此注意。同时，熟悉硬件描述的方式，也是进入验证领域的一项基本技能。那么，就从这个规模适中的设计开始了解吧。

### 功能描述

该设计我们称之为多通道数据整形器（MCDF，multi-channel data formatter），它可以将上行（uplink）多个通道数据经过内部的FIFO，最终以数据包（data packet）的形式送出。

由于上行数据和下行数据的接口协议不同，我们也将在后面的接口描述和时序部分进一步讲解。此外，多通道数据整形器也有寄存器的读写接口，可以支持更多的控制功能。

### 设计结构



从上图的MCDF结构来看主要可以分为如下几个部分：

1. 上行数据的通道从端（Channel Slave），负责接收上行数据，并且存储到其FIFO中。
2. 仲裁器（Arbiter）可以选择从不同的FIFO中读取数据，进而将数据进一步传送至整形器（formatter）。
3. 整形器（Formatter）将数据按照一定的接口时序送出至下行接收端。
4. 控制寄存器（Control Registers）有专用的寄存器读写接口，负责接收命令并且对MCDF的功能做出修改。

### 接口描述

**系统信号接口**

* CLK(0)：时钟信号。
* RSTN(0)：复位信号，低位有效。

**通道从端接口**

* CHx\_DATA(31:0)：通道数据输入。
* CHx\_VALID(0)：通道数据有效标志信号，高位有效。
* CHx\_READY(0)：通道数据接收信号，高位表示接收成功。

**整形器接口**

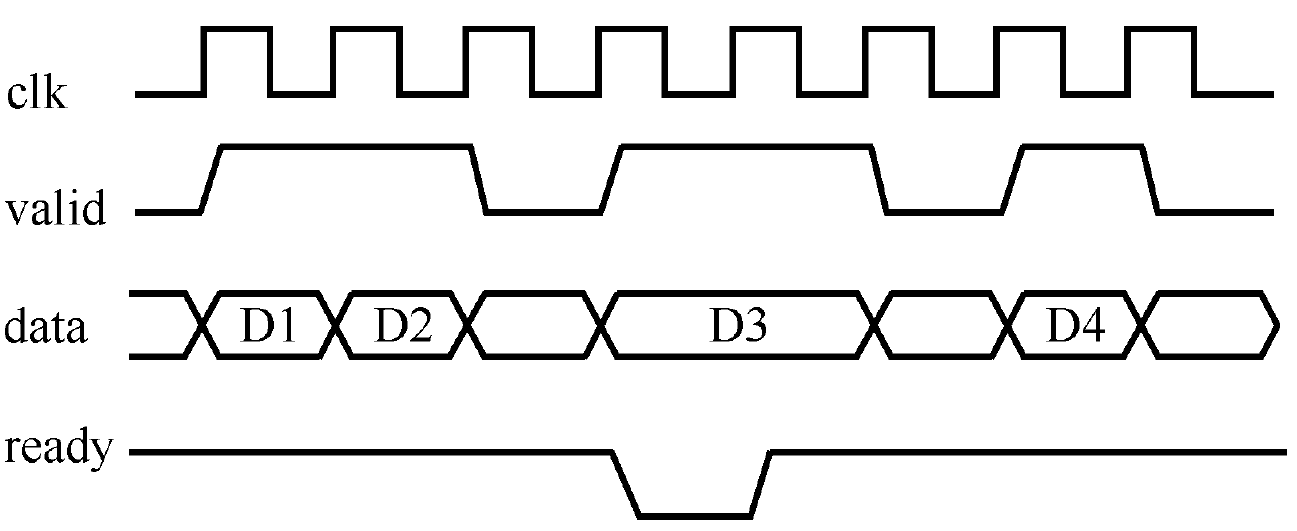
* FMT\_CHID(1:0)：整形数据包的通道ID号。
* FMT\_LENGTH(4:0)：整形数据包长度信号。
* FMT\_REQ(0)：整形数据包发送请求。
* FMT\_GRANT(0)：整形数据包被允许发送的接受标示。
* FMT\_DATA(31:0)：数据输出端口。
* FMT\_START(0)：数据包起始标示。
* FMT\_END(0)：数据包结束标示。

**控制寄存器接口**

* CMD(1:0)：寄存器读写命令。
* CMD\_ADDR(7:0)：寄存器地址。
* CMD\_DATA\_IN(31:0)：寄存器写入数据。
* CMD\_DATA\_OUT(31:0)：寄存器读出数据。

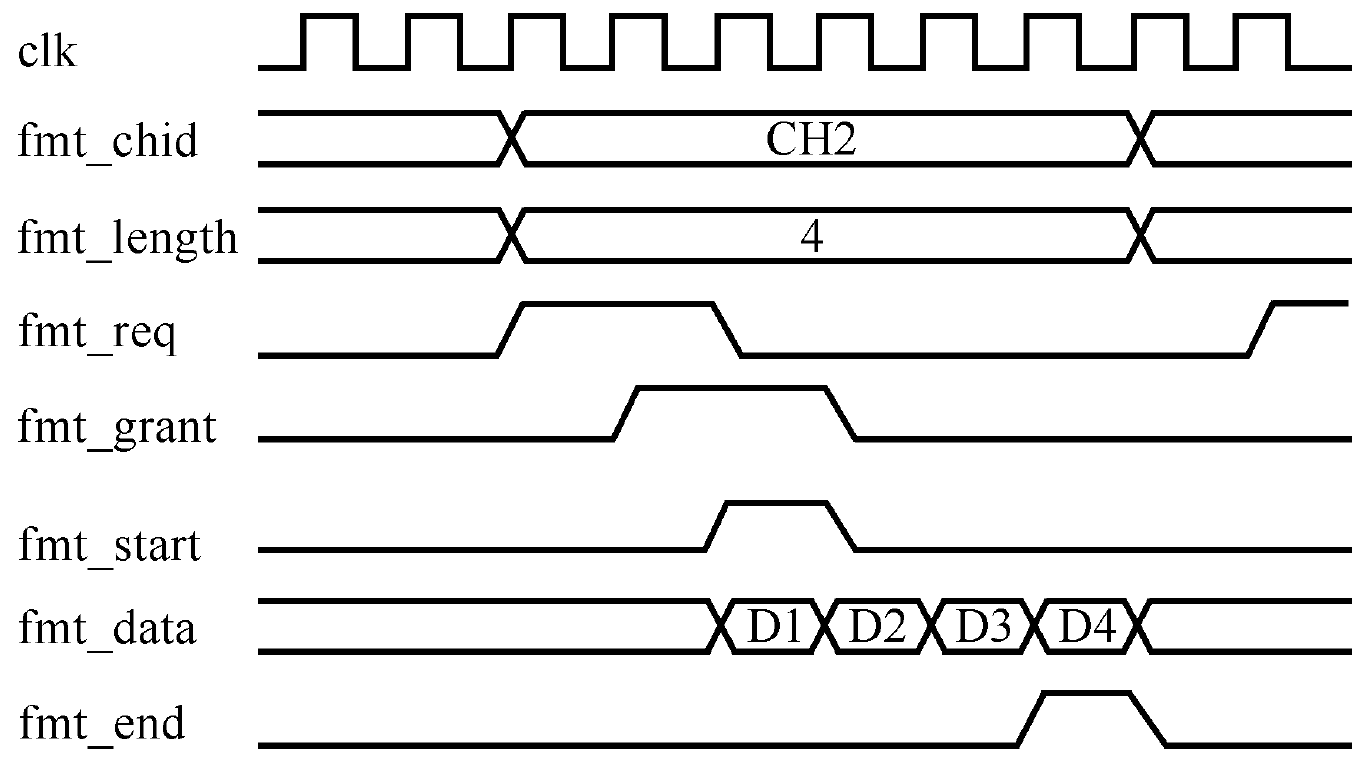
### 接口时序

**通道从端接口时序**



当valid为高时，表示要写入数据。如果该时钟周期ready为高，则表示已经将数据写入；如果该时钟周期ready为低，则需要等到ready为高的时钟周期才可以将数据写入。

**整形器接口时序**

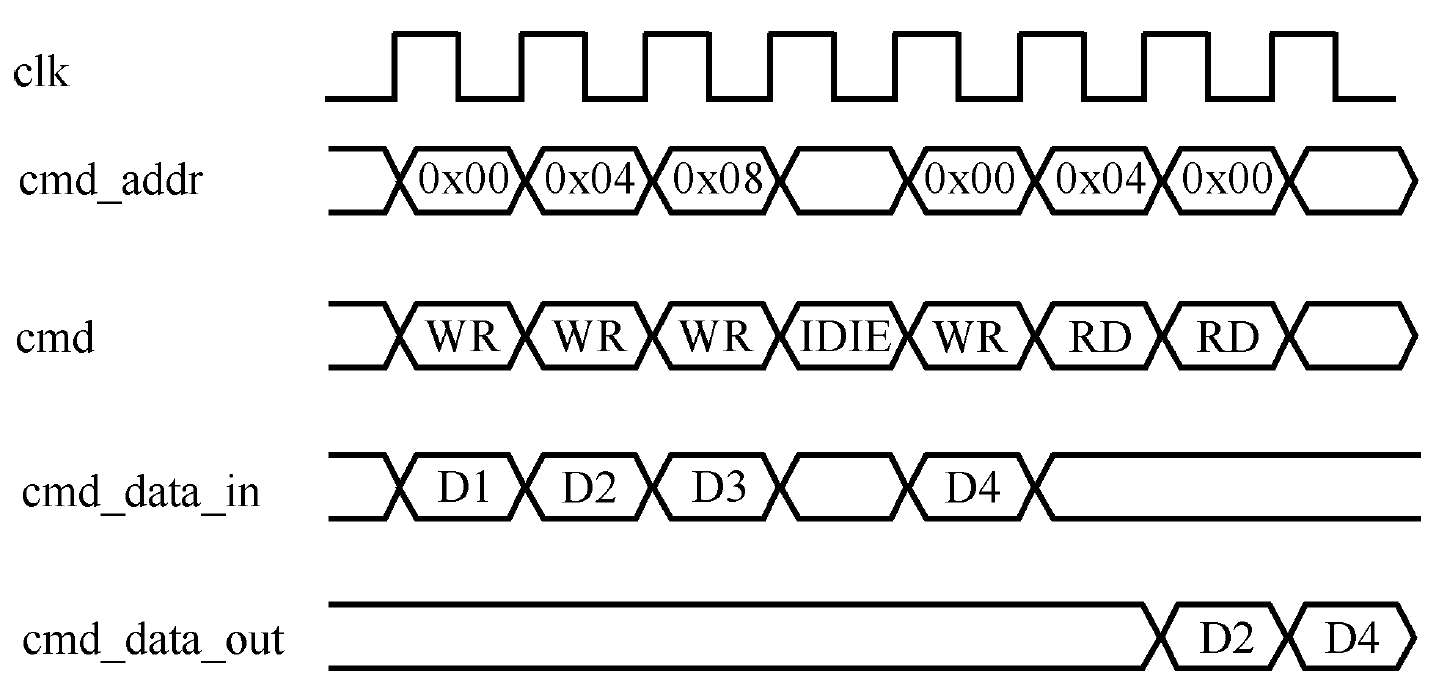


整形器发送数据是按照数据包的形式发送的，可以选择数据包的长度有4、8、16和32。整形器必须完整发送某一个通道的数据包后，才可以转而准备发送下一个数据包，在发送数据包期间，fmt\_chid和fmt\_length应该保持不变，直到数据包发送完毕。

在整形器准备发送数据包时，首先应该将fmt\_req置为高，同时等待接收端的fmt\_grant。当fmt\_grant变为高时，应该在下一个周期将fmt\_req置为低。fmt\_start也必须在接收到fmt\_grant高有效的下一个时钟被置为高，且需要维持一个时钟周期。在fmt\_start被置为高有效的同一个周期，数据也开始传送，数据之间不允许有空闲周期，即应该连续发送数据，直到发送完最后一个数据时，fmt\_end也应当被置为高并保持一个时钟周期。

相邻的数据包之间应该至少有一个时钟周期的空闲，即fmt\_end从高位被拉低以后，至少需要经过一个时钟周期，fmt\_req才可以被再次置为高。

**控制寄存器接口时序**



在控制寄存器接口上，需要在每一个时钟解析cmd。当cmd为写指令时，需要把数据cmd\_data\_in写入到cmd\_addr对应的寄存器中；当cmd为读指令时，即需要从cmd\_addr对应的寄存器中读取数据，并在下一个周期，将数据驱动至cmd\_data\_out接口。

### 寄存器描述

**地址0x00 通道1控制寄存器 32bits 读写寄存器**

bit(0)：通道使能信号。1为打开，0位关闭。复位值为1。

bit(2:1)：优先级。0为最高，3为最低。复位值为3。

bit(5:3)：数据包长度，解码对应表为， 0对应长度4， 1对应长度8，2对应长度16，3对应长度32，其它数值（4-7）均暂时对应长度32。复位值为0。

bit(31:6)：保留位，无法写入。复位值为0。

**地址0x04 通道2控制寄存器 32bits 读写寄存器**

同通道1控制寄存器描述。

**地址0x08 通道3控制寄存器 32bits 读写寄存器**

同通道1控制寄存器描述。

**地址0x10 通道1状态寄存器 32bits 只读寄存器**

bit(7:0)：上行数据从端FIFO的可写余量，同FIFO的数据余量保持同步变化。复位值为FIFO的深度数。

bit(31:8)：保留位，复位值为0。

**地址0x14 通道2状态寄存器 32bits 只读寄存器**

同通道1状态寄存器描述**。**

**地址0x18 通道3状态寄存器 32bits 只读寄存器**

同通道1状态寄存器描述**。**

## Questasim基本使用技巧

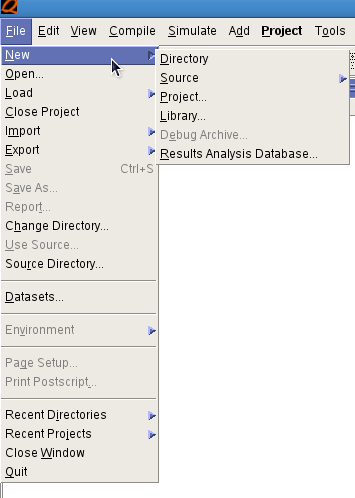
在了解了MCDF的设计之后，我们接下来的几次实验将首先围绕着MCDF中的子模块channel和arbiter的组合MCDT着手，暂时不考虑formatter的数据打包功能和寄存器的配置作用。因此我们所选择的MCDF的“片段”MCDT即由三个slave channel和一个arbiter构成。从之前介绍的各个模块功能来看，这个组合的作用类似于一个高级的“MUX”多路选择器，只不过这个选择器的无需外部的选择配置，而是会根据slave channel的数据发起请求来做选择授权，继而将数据送出。

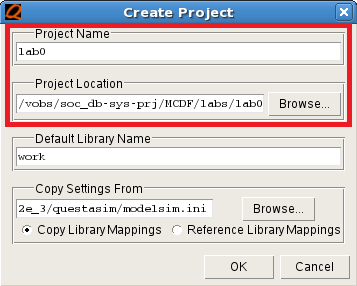


接下来我们就将文件库中的文件进行编译和仿真，伴随着同学们熟悉的Verilog定向激励，一起回顾一下在Questasim中如何建立一个项目、添加设计文件和验证文件、完成编译和开始仿真吧。

### 建立项目

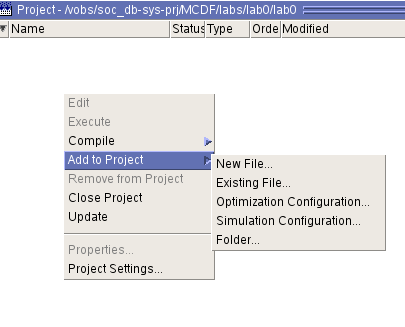
* 新建项目名称File -> New -> Project，在project name处填入名称例如lab0



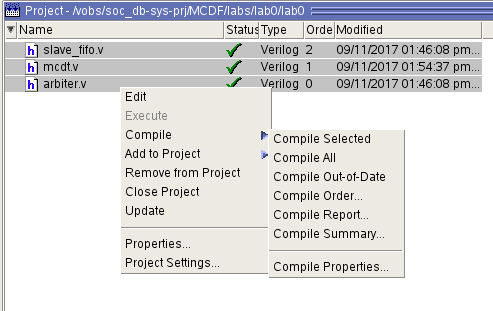


### 添加设计文件和编译

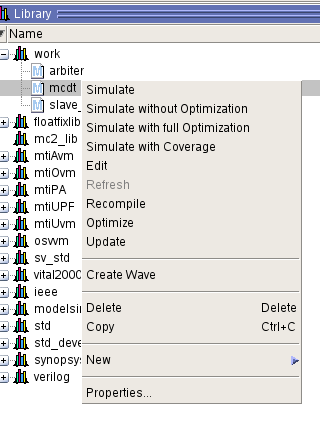
* 在project窗口单击右键，选择Add to Project -> Existing File，找到下载的lab1实验文件\*.v。



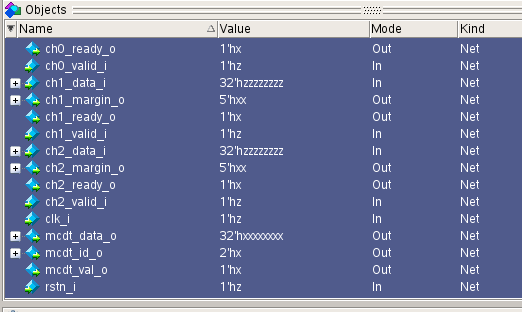
* 编译设计文件。



* 在Library -> Work中找到顶层设计mcdt，选择仿真。



* 选中设计mcdt，在objects一栏中可以看到它的端口和内部信号。

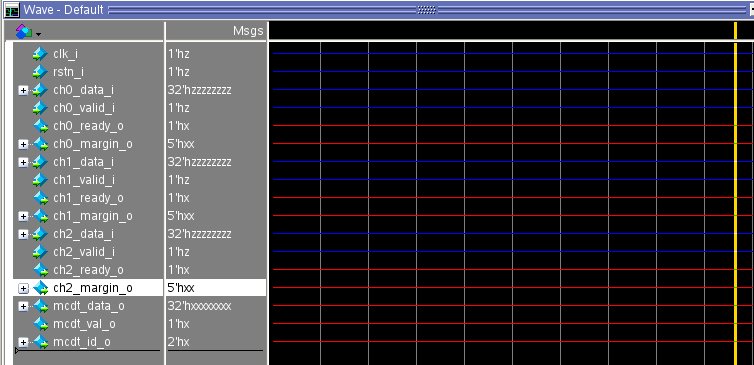


* 接下来在instance -> mcdt上点击右键，选择Add Wave，这是为了在仿真时存储波形。  
  在命令窗口中敲入命令“run 1us”或者在工具栏中运行1us。





* 观察在波形窗口中的，信号是否发生了变化？  
  如果没有变化，那么在哪些端口上是z值，哪些端口是x值？这些值与端口之间是否存在联系？



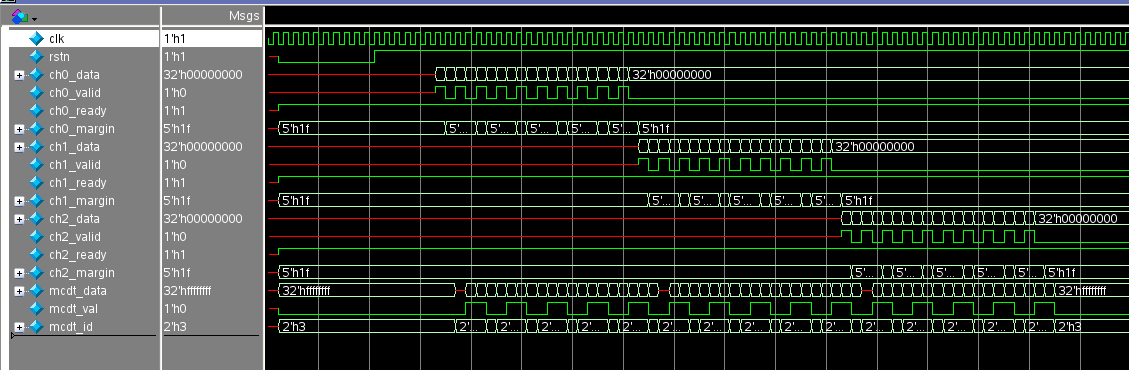
### 添加验证文件tb1.v和仿真

接下来，我们需要开始回顾之前的Verilog典型测试方式，展开基本测试：

* 发起复位操作，观察mcdt的输出端口变化
* 分别在channel0、channel1、channel2写入数据，观察mcdt的输出数据

我们可以按照之前的步骤，将文件**tb1.v**添加到project中，进行编译，并且在Library中选中它，点击右键选择“Simulate without Optimization”，这种仿真模式是为了消除仿真器可能会加入的一些优化处理。在work库中选择了tb1进行仿真，同时添加波形，让仿真进行1us，然后再观察波形，考虑以下问题：

* 各个channel中的数据在输出端口是否完整？
* channel的输入数据是否有何特点？如何区分不同channel的数据内容呢？
* arbiter的输出数据有何特点？如何区分不同channel的数据内容呢？
* 在tb1.v中的测试代码中，哪一个方法是数据激励的基本方法？



### 延伸问题

* 是否可以每个channel的数据连续发送？如何处理呢？
* 是否可以让三个channel的数据同时向arbiter发送？如何处理呢？

## 恭喜你！顺利通关喽！