数字逻辑与部件设计 实验部分-03.开发流程（下）：仿真

`Software debug.`

大家在上一节课已经体验了从硬件代码到上板的流程，感受到了生成硬件的`bitstream`文件很慢。所以我们需要更高效的调试方法，也就是这节课的内容：仿真。

助教录了一个Vivado仿真流程的视频，视频链接

本文档主要介绍仿真专用的一些语法（写电路用的语法在仿真文件中都可以用）。**这些语法没有对应的电路，只能用于仿真！**

仿真文件架构

仿真文件以一个无输入输出端口的模块作为顶层（假设命名为`sim`）：

```
module sim();

endmodule
```

在这个模块里，例化一个（或很多个）我们待调试的模块，然后生成待调试模块的输入激励信号，检查模块内部信号和输出。

#：延时

我们希望在各个时刻给我们写的数字模块不同的输入激励。`#`语法的作用是延时，可以满足我们的这个要求。

`#`作用于verilog语句前。

initial语句

initial语句的语法：

```
initial begin

end
```

在initial语句块中，软件会**顺序**地执行每一条语句。

不用`#`延时语法的语句被认为在同一时刻执行。

例：

```
logic b;
initial begin
    b = 1'b1;
    #10 b = 1'b0; // delay 10 ns
end
```

可以有多个initial block，用于不相关的信号集合的生成。

系统函数

常用的系统函数有三个：

- `$display`，语法和c语言的`printf`一致，用于在`Tcl Console`输出信息，提供了一种波形图以外的调试方法。
- `$time`，配合`$display`适用，获取当前时刻。
- `$finish`，中止仿真。

task

有些task可以映射到电路，但不推荐写电路时用task。

task的语法和module类似：

```
task taskName();

endtask
```

task内部的语句会被顺序执行。

在仿真中，常用的task是比较答案与模块的实际信号：

```
task check(input logic ans, input logic get);
    #10
    if(ans != get) begin
       // do something
    end
endtask
```