# 搭建你的数字积木
## —数字电路与逻辑设计
## （Verilog HDL&Vivado版）
## — 参考课件PPT

东南大学 & Xilinx大学计划部

# Chap.6.逻辑设计工程技术基础

# 本章学习导言

- 前五章结合最新的可编程逻辑器件综合开发环境Vivado对数字电路（逻辑设计）的基本知识点，包括布尔代数、组合逻辑设计、时序逻辑设计、有限状态机设计等内容做了相应的介绍。这些知识基本上完整覆盖了今后逻辑系统设计的大多数理论和设计基础，但在逻辑系统设计的实际工程应用中，还有很多对于工程设计质量有重要影响的实用技术。
- 本章的教学目的是给出部分对逻辑系统设计质量有影响的主要**实际工程技术**，包括逻辑系统稳定性认识、逻辑系统中的竞争和冒险（俗称“毛刺问题”）、程序的可读性和可扩展性设计、Vivado工具常用辅助功能等内容进行介绍，增加学习者的实践经验和设计能力。

# 主要内容

- 数字电路稳定性
- 组合逻辑与毛刺
- 异步设计与毛刺
- Verilog HDL设计中的编程风格
- Xilinx开发环境中的其它逻辑设计辅助工具

# 数字电路稳定性

- 数字电路稳定性定义
- 竞争与冒险
- 时间抖动
- 数字系统亚稳态

# 数字电路稳定性定义

**系统稳定性**：指当去掉系统干扰后，系统能以足够的精度恢复到初始平衡状态。

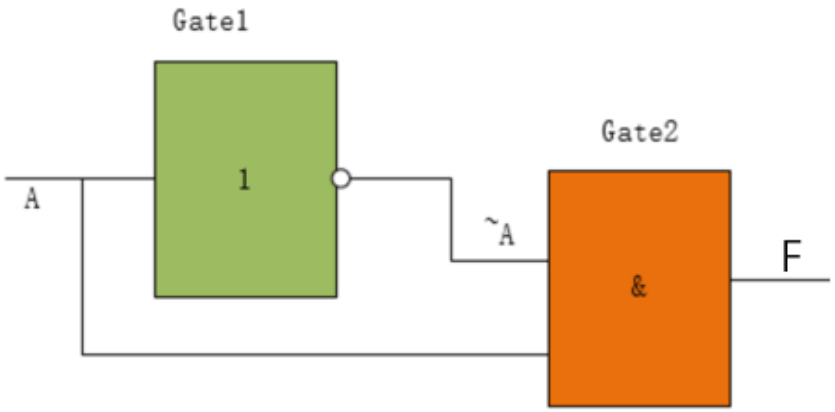
**系统可靠性**：需要系统正常工作时候它能正常工作，可表达为时间的数字函数：

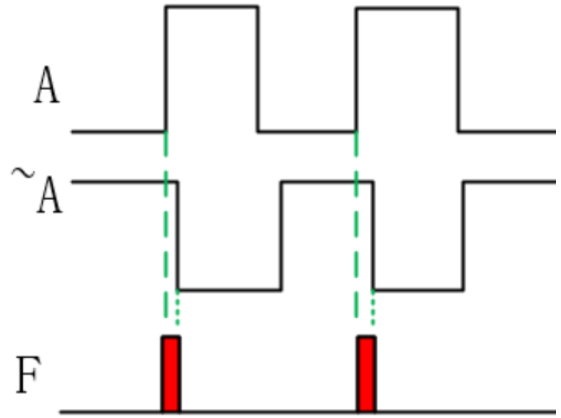
R(t)=系统在时刻t仍能正常工作的概率。

数字系统的稳定性有很多影响因素。有时候，即使设计正确，该系统仍不能正常工作。

**影响因素**：数字电路中的竞争与冒险、时间抖动（jitter）、亚稳态等因素都会影响数字系统的稳定性和可靠性。下面我们分别做详细介绍

# 竞争与冒险



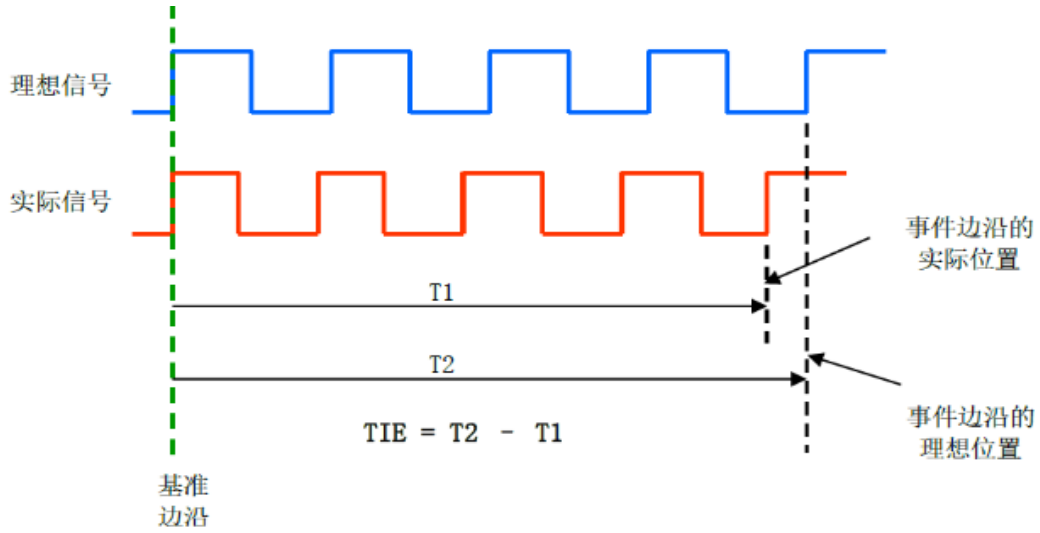




在逻辑电路中，某个输入变量或不同信号通过两条或两条以上的途径传到输出端，由于途径上的延迟时间不同，到达输出门的时间就有先有后，这种现象称为竞争，因此而导致输出干扰脉冲险象的现象称为冒险，如上图F产生“毛刺”。

**非临界竞争**：不会产生错误输出的竞争现象;

**临界竞争**：产生暂时性的或永久性错误输出的竞争现象。

# 时间抖动



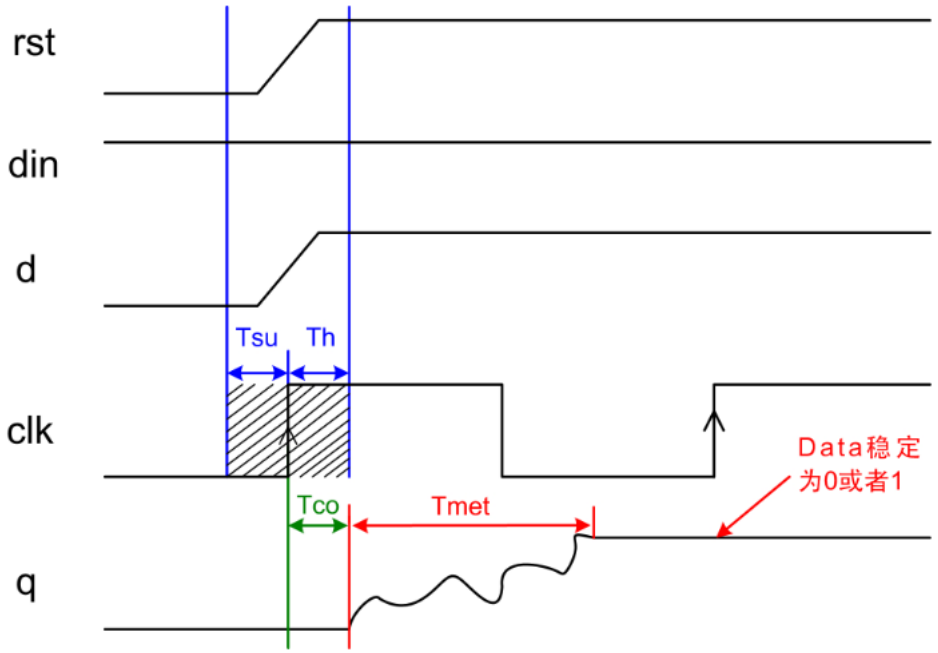

**抖动**：信号的定时事件与其理想位置之间的偏差。

在理想情况下，一个频率固定的完美的脉冲信号（以1MHz为例）的持续时间应该恰好是1us，每500ns有一个跳变沿。但是，这种理想信号并不存在。

实际的信号周期长度总会有一定差异，从而导致下一个沿的到来时间不确定，这种不确定就是**抖动（jitter）**。

抖动是对信号时域变化的测量结果，它从本质上描述了信号周期距离其理想值偏离了多少。

# 数字系统的亚稳态



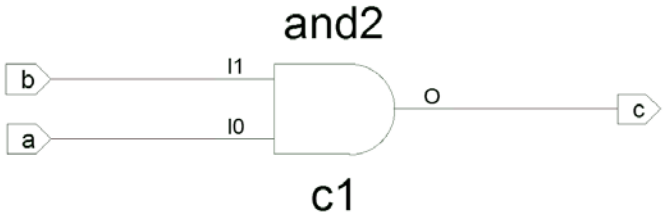

**亚稳定状态：**在同步系统中，如果触发器的建立时间和保持时间不满足，就可能进入亚稳态，此时触发器输出端Q在有效时钟沿之后比较长的一段时间处于状态1和状态0之间的不确定状态，即亚稳定状态。

当其他门电路或触发器接受到**亚稳定的输入信号**之后，有些部件会把这个信号当成0，而另一些则把它当成1，还有一些部件本身也可能产生**亚稳定的输出信号**，从而导致电路工作出现稳定性错误。

# 组合逻辑与毛刺

- “毛刺”的产生
- “毛刺”的分类
- “毛刺”的处理

# “毛刺”的产生_1

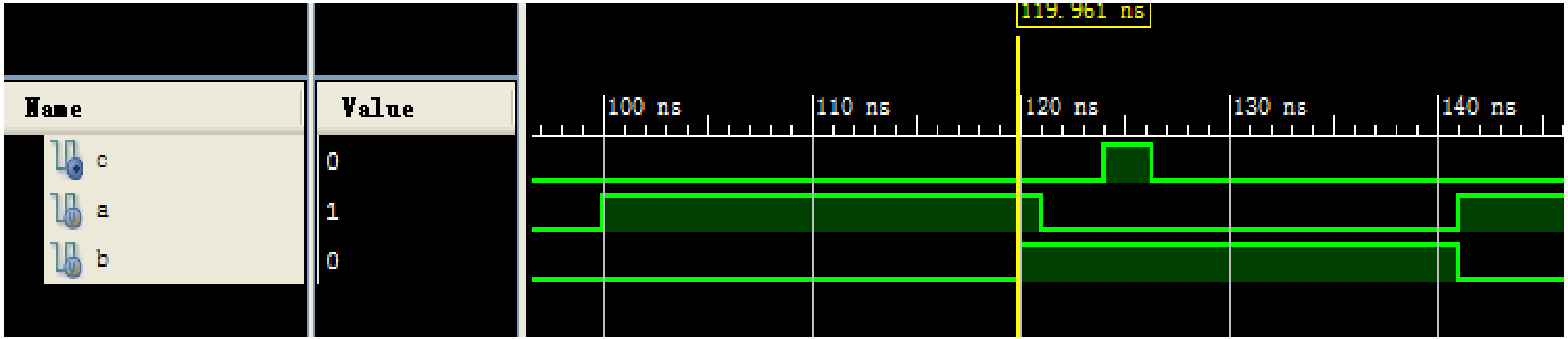



**延时**：信号在器件内部通过逻辑单元和连线时，都有一定的延时，分别称为器件延时和路径延时。

用软件预估出a、b到c的两条最差路径的延时分别是7.567ns和5.436ns，这个总延时等于互连线路径延时加上与门的延时。

由于延时的影响，加之高低电平转换也需要一定的过渡时间，在信号变化的瞬间，组合逻辑的输入信号并不是同时变化，由此往往会使得组合逻辑的输出出现一些不正确的尖峰信号，这些尖峰信号称为“毛刺”。具体分析参见下一页的波形图

# “毛刺”的产生_2




a和b两条路径存在时间差，b比a提前2ns左右，因此虽然a、b两个信号只在1ns内为同高，但是c输出的高电平持续时间达到了2.2ns左右。

# “毛刺”的分类_1

按输出信号是否应该变化可分为静态险象和动态险象。

**静态险象：**如果在输入变化而输出不应发生变化的情况下，输出端产生了短暂的错误输出，则称为静态险象。

**动态险象：**如果在输入变化而输出应该发生变化的情况下，输出在变化过程中产生了短暂的错误输出，则称为动态险象。

# “毛刺”的分类_2

按错误输出脉冲信号的极性可分为“0”型险象与“1”型险象。

**“0”型险象**：错误输出信号为负脉冲。

**“1”型险象**：错误输出信号为正脉冲。

# “毛刺”的分类_3

按险象出现的原因可分为逻辑险象与功能险象。

**逻辑险象**：由于某个变量与其反变量作用时间不一致，出现瞬时同态造成的险象。

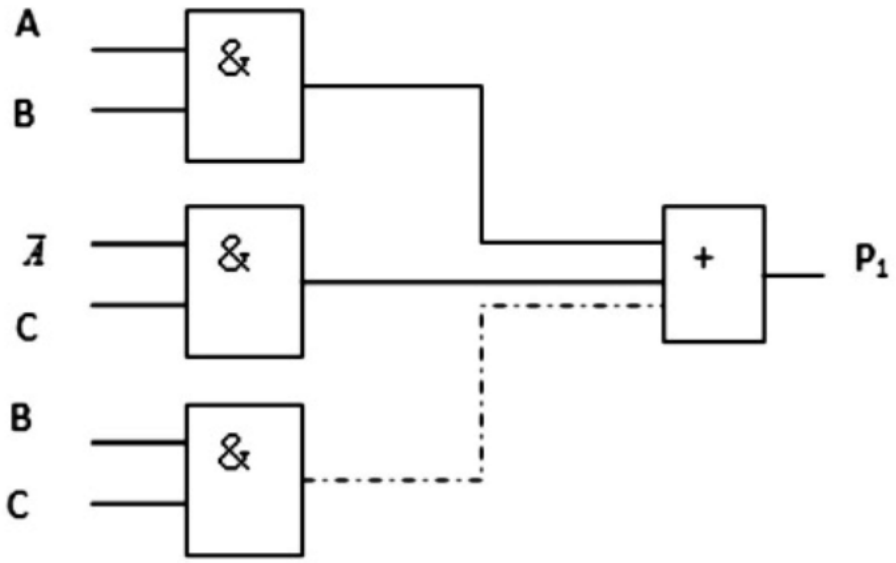
**功能险象**：由于两个或两个以上输入信号同时变化时出现的险象。

# “毛刺”的处理_1：增加冗余项

增加冗余项的方法是：

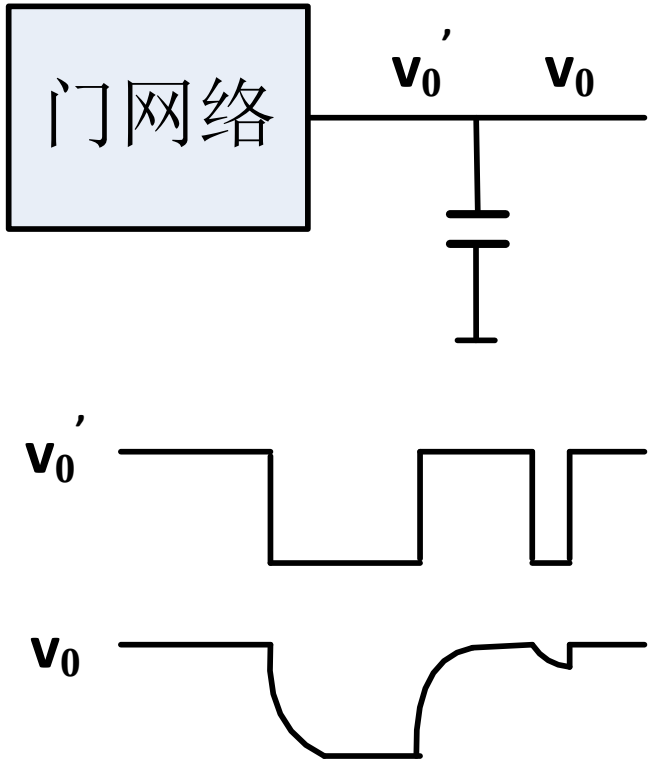
1. “或”上冗余的“与”项
2. “与”上冗余的“或”项

可以消除可能产生的险象。但是此法只能用于消除逻辑险象，功能险象无法消除。




如图，在原来表达式基础上加上”B*C”项，可消除B=C=1所产生的冒险。

# “毛刺”的处理_2：增加小电容

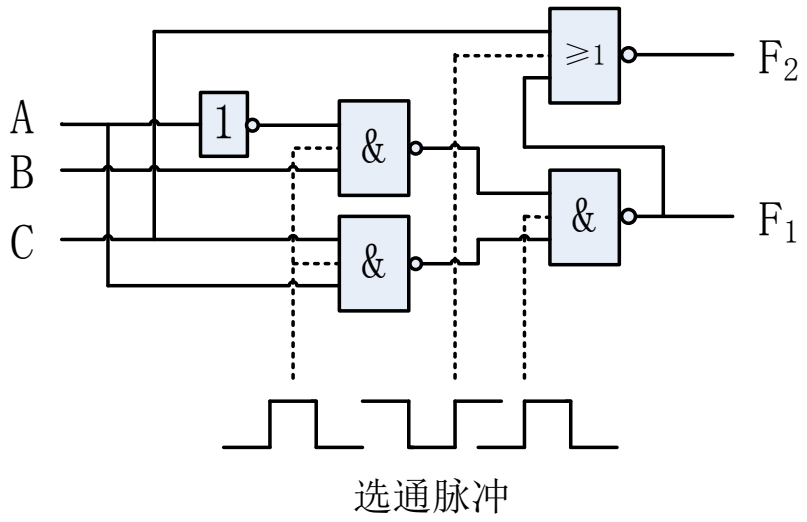



## 在组合逻辑芯片的管脚输出端增加小电容

由于竞争引起的险象都是一些频率很高的尖脉冲信号，在电路的输出端加一个小电容，如图所示。利用电容对电压变化延迟的特性来过滤掉宽度极窄的毛刺信号。

但由于电容将使得输出波形的边沿加宽，降低了电路的速度。

# “毛刺”的处理_3：选通法




真正在设计中对于消除毛刺常用的是这个选通法。通过选通脉冲对电路的输出门加以控制，令选通脉冲在电路稳定后出现，可使输出避开险象脉冲，送出稳定输出信号。

**分类：**对输入信号选通

对输出信号选通

采用此方法后，门网络的输出不再是连续的，它只在选通信号作用期间有效。

如果选通信号采用系统全局的时钟，能够使得系统各部分同步工作。

# 异步设计与“毛刺”

- 异步时序电路中的“毛刺”现象
- 异步时序电路中“毛刺”的处理

# 异步时序电路中的“毛刺”现象

当异步时序电路在状态瞬变期间，若存在多于一个的状态变量同时发生改变，则说此电路处于竞争之中。

**临界竞争**：若电路所趋向的最终稳态与状态变量的变化次序有关,则此电路的竞争是临界的。

**非临界竞争**：若电路所趋向的最终稳态与状态变量的变化次序无关,此电路的竞争是非临界的。

如果异步时序电路存在临界竞争，输出状态会出现“毛刺”现象，影响电路的稳定性。

# “毛刺”处理_1:延迟元件法

| $q^2q^1$ | | $x_1x_2$ | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| A | 00 | 00 | 11* | 10 | 11* |
| B | 01 | 00 | 01 | 11 | 11 |
| C | 11 | 00* | 10 | 11 | 11 |
| D | 10 | 00 | 10 | 11 | 11 |

在输入信号(*x* 1 *x* 2) = (01) 的情况下, 电路的状态从(*q*1 *q*2) = (00) 到(11) 的转换中含有临界竞争。通过对组合电路插入一个适当量的延迟,使*q*2值从0到1的变化总比*q*1从0到1的变化快, 这样临界竞争就变成非临界竞争, 从而得到正确的结果。

# “毛刺”处理_2：多次转换法

| $q^2q^1$ | | $x_1x_2$ | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| A | 00 | 00 | 11* | 10 | 11* |
| B | 01 | 00 | 01 | 11 | 11 |
| C | 11 | 00* | 10 | 11 | 11 |
| D | 10 | 00 | 10 | 11 | 11 |

假定（A→C)含有一个临界竞争，如果可以找到一个多次转换(A→D→C)，那么，若能使D和A相邻，则用状态D代替C后，将消除转换中的临界竞争，并且维持“外部性能不变”。

例如表中在(01)输入下，(11)与(10)有相同的次态且(11)与(10)相邻，则可用(10)代替(11)，并不影响外部性能，从而使存在临界竞争的转换(00) → (11) 变为无竞争的转换(00) →(10)。

# “毛刺”处理_3：非临界竞争赋值法

| $q^2q^1$ | | $x_1x_2$ | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| A | 00 | 00 | 11* | 10 | 11* |
| B | 01 | 00 | 01 | 11 | 11 |
| C | 11 | 00* | 10 | 11 | 11 |
| D | 10 | 00 | 10 | 11 | 11 |

| $q^3q^2q^1$ | | $x_1x_2$ | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| A | 000 | 000 | 011 | 010 | 011 |
| B | 101 | 000 | 101 | 011 | 011 |
| C | 011 | 000 | 010 | 011 | 011 |
| D | 010 | 000 | 010 | 011 | 011 |
| E | 001 | 000 | 010 | 011 | 011 |
| F | 100 | 000 | -- | -- | -- |
| G | 111 | -- | -- | 011 | -- |

即允许状态编码有非临界竞争出现。例如左边表格中给定的四个状态，利用临界竞争编码得到的编码表如右边表格。当增加E、G、F 三个新状态后，就可以使它们之间的竞争全部化为非临界竞争。

# “毛刺”处理_4:同步电路改造法

**同步电路优点：**

a. 能够有效地避免毛刺的产生，提高设计的稳定性。
b. 能够简化时序分析过程。同步电路便于电路错误分析，加快设计进度。
c. 可以减少工作环境对设计的影响。同步电路受工作温度、电压等影响，器件时延变化小。对于时钟和数据沿相对稳定的电路，时序要求较为宽松，因此对环境的依赖性较小。

同步设计时钟信号的质量和稳定性决定了同步时序电路的性能，而在FPGA内部有专用的时钟资源，如全局时钟布线资源、专用的时钟管理模块DUL、PLL等。

目前商用的FPGA都是针对同步的电路设计而优化的，同步时序电路可以很好地规避毛刺，提倡在设计中全部使用同步逻辑电路。特别注意，不同时钟域的接口需要进行同步。

# Verilog HDL设计中的编程风格要点

- 强调代码编写风格的必要性
- 强调编写规范的宗旨
- 变量及信号命名规范
- 编码格式规范

# 强调代码编写风格的必要性

- 所有软件设计公司都是强调代码规范的。逻辑设计也是这样：如果不按规范做的话，过一个月后调试时发现有错，回头再看自己写的代码，估计很多信号功能都忘了，更不要说检错了。
- 遵循代码编写规范书写的代码，便于阅读、理解、维护、修改、跟踪调试、整理文档。
- 相反，对于编写风格随意的代码，通常晦涩、凌乱，会给开发者本人的调试、修改工作带来困难，也会给合作者带来极大麻烦。

# 强调编写规范的宗旨

1. 缩小篇幅
2. 提高整洁度
3. 便于跟踪、分析、调试
4. 增强可读性，帮助阅读者理解
5. 便于整理文档、交流合作

# 变量及信号命名规范_1

## 1. 系统级信号

系统级信号指复位信号，置位信号，时钟信号等全局信号；

系统信号以字符串Sys开头，如SysClk；时钟信号以clk开头，并在后面添加相应的频率值；

复位信号一般以rst或reset开头；置位信号为st或set开头。如下：

```
wire [7:0] sys_dout, sys_din;

wire clk_32p768MHz;

wire reset;

wire st_counter;
```

# 变量及信号命名规范_2

## 2. 低电平有效的信号

信号后一律加下划线和字母n。如：SysRst_n；Dram_WrEn_n

## 3. 经过锁存器锁存后的信号

信号后加下划线和字母r，与锁存前的信号区别。如Data_Out信号，经锁存后应命名为Data_Out_r。

低电平有效的信号经过锁存器锁存后，其命名应在_n后加r。如Data_Out_n信号，经锁存后应命名为Data_Out_nr

多级锁存的信号，可多加r以标明。如Data_Out信号，经两级触发器锁存后，应命名为Data_Out_rr。

# 变量及信号命名规范_3

## 4. 模块的命名

将模块英文名称的各个单词首字母组合起来，形成3到5个字符的缩写。若模块的英文名只有一个单词，可取该单词的前3个字母。各模块的命名以3个字母为宜。例如：

Central Processing Unit模块，命名为CPU。

## 5 . 模块之间的接口信号的命名

命名分为两个部分，第一部分表明数据方向，第二部分为数据名称，两部分之间用下划线隔离开。

第一部分全部大写，第二部分所有具有明确意义的英文名全部拼写或缩写的第一个字母大写，其余部分小写。举例：CPUMMU_WrReq；

模块上下层次间信号的命名也遵循本规定。若某个信号从一个模块传递到多个模块，其命名应视信号的主要路径而定。

# 变量及信号命名规范_4

## 6. 模块内部信号

模块内部的信号由几个单词连接而成，缩写要求能基本表明本单词的含义；

单词除常用的缩写方法外，一律取该单词的前几个字母。如：Frequency->Freq, Variable->Var 等；

每个缩写单词的第一个字母大写。举例：FlashAddrLatchEn;

若遇两个大写字母相邻，中间添加一个下划线。举例：SdramWrEn_n;

# 编码格式规范_1

## 1.分节书写

书写时，各节之间加1到多行空格。每节基本上完成一个特定的功能，即用于描述某几个信号的产生。在每节之前有几行注释对该节代码加以描述，至少列出本节中描述的信号的含义。

## 2. 对齐

行首不要使用空格来对齐，而是用Tab键，Tab键的宽度设为4个字符宽度。行尾不要有多余的空格。

## 3. 注释

注释有两种方法：

① 使用//进行的注释行以分号结束，例：//SDRAM Data bus 16 Bits

② 使用/* */进行的注释，/*和*/各占用一行，并且顶头。

# 编码格式规范_2

## 4. 空格的使用

① 不同变量，以及变量与符号、变量与括号之间都应当保留一个空格。

② Verilog关键字与其它任何字符串之间都应当保留一个空格。

③ 使用大括号和小括号时，前括号的后边和后括号的前边应当留有一个空格。

④ 逻辑运算符、算术运算符、比较运算符等运算符的两侧各留一个空格，与变量分隔开来；单操作数运算符例外，直接位于操作数前，不使用空格。

⑤ 使用//进行的注释，在//后应当有一个空格；注释行的末尾不要有多余的空格。

## 5. 代码块

在endmodule，endtask，endcase等标记一个代码块结束的关键词后面要加上一行注释说明这个代码块的名称。

# 编码格式规范_3

## 6. 同层次书写

同一个层次的所有语句左端对齐；Initial、always等语句块的begin关键词跟在本行的末尾，相应的end关键词与Initial、always对齐；这样做的好处是避免因begin独占一行而造成行数太多。如下：

```
always @(negedge SysRst or posedge SysClk) begin
  if ( !SysRst ) Sd_Cnt = 6'b111111;
  else begin
    if (Go == 0)
     Sd_Cnt = 0;
    else
     if (Sd_Cnt < 6'b111111) Sd_Cnt = Sd_Cnt + 1;
  end
end
```

## 7. 不同层次书写

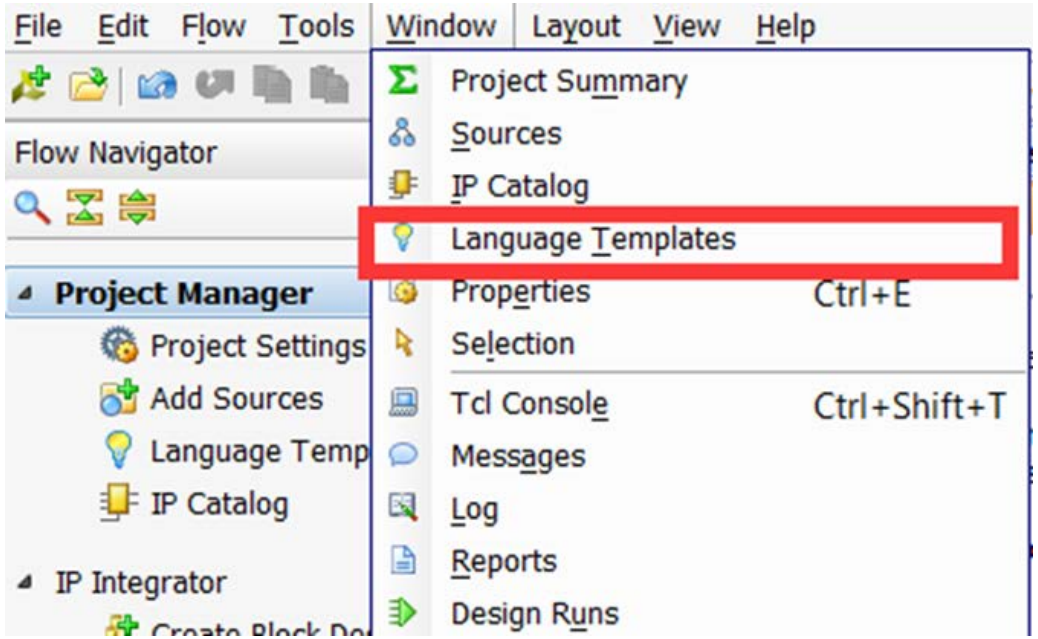
不同层次之间的语句使用Tab键进行缩进，每加深一层缩进一个Tab；

# Xilinx开发环境中的其它逻辑设计辅助工具_1

在VIvado中，软件提供了一些模板和原语给使用者们直接调用FPGA内的资源。

**使用步骤如下：**

1. 打开Vivado Window一栏，选择Language Templates。



# Xilinx开发环境中的其它逻辑设计辅助工具_2

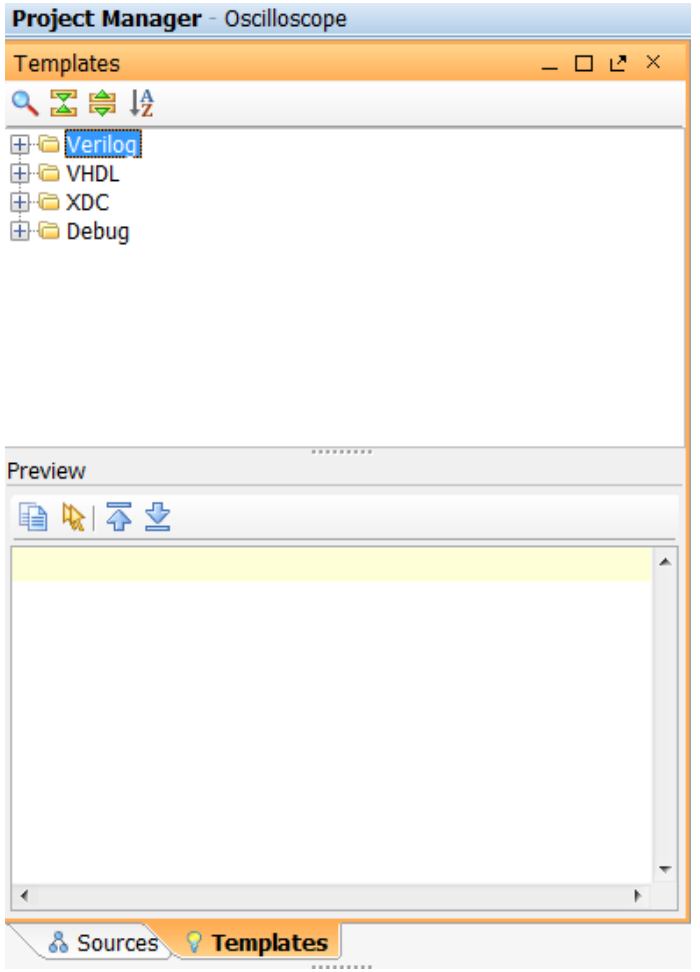
2. 在源程序Sources窗口一栏，选择Templates。这样就可以看到原语列表。

**Verilog：** 主要是Verilog的原语以及程序模板以及仿真模板。

**VHDL：** 主要是VHDL的原语以及程序模板以及仿真模板。

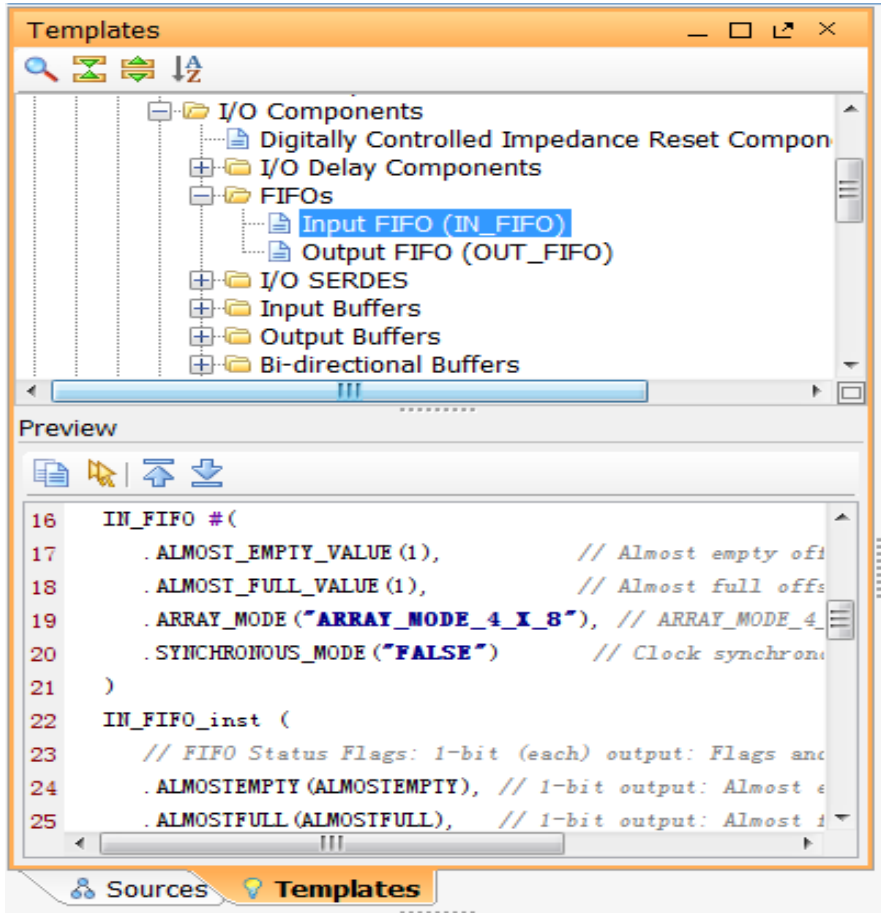
**XDC：** 主要是约束文件模板。

**Debug：** 主要是调试模板。



# Xilinx开发环境中的其它逻辑设计辅助工具_3

3. 调用方法是选择某一个具体的原语或模板，则会在Preview一栏显示出程序。将程序复制到程序代码里，便可以直接调用。




XILINX
UNIVERSITY PROGRAM

xup@xilinx.com