目录

[1. Verilog为什么适合描述硬件设计？ 2](#_Toc57907691)

[2. 阻塞赋值和非阻塞赋值的区别？ 2](#_Toc57907692)

[3. 为什么用于综合的verilog不建议使用for循环？ 2](#_Toc57907693)

[4. 如何防止综合出Latch？ 2](#_Toc57907694)

[5. verilog可综合风格？ 3](#_Toc57907695)

[6. verilog可综合、不可综合语句汇总？ 3](#_Toc57907696)

[7. function和task区别？ 3](#_Toc57907697)

[8. 一段式、两段式、三段式状态机？ 4](#_Toc57907698)

[9. Mealy状态机和Moore状态机区别、优缺点？ 5](#_Toc57907699)

[10. 状态机编码选择？ 5](#_Toc57907700)

[11. 独热码优点？ 5](#_Toc57907701)

[12. DFT的意义及常用DFT技术？ 5](#_Toc57907702)

[13. 竞争和冒险、Glitch及消除办法？ 6](#_Toc57907703)

[14. case/casez/casex的区别？ 7](#_Toc57907704)

[14. 跨时钟域同步方法？（后续会专门推出跨时钟域同步的专题） 8](#_Toc57907705)

[16. 同步设计的优越性、缺点及设计规则？ 8](#_Toc57907706)

[17. 同步电路和异步电路的概念？ 9](#_Toc57907707)

[18. 功能覆盖率、代码覆盖率、断言覆盖率？ 9](#_Toc57907708)

[19. 条件运算符对x和z的处理？ 10](#_Toc57907709)

[20. 乒乓buffer的概念？ 10](#_Toc57907710)

[21. 线与逻辑？ 11](#_Toc57907711)

[22. Modelsim仿真步骤？ 12](#_Toc57907712)

[23. 同步通信和异步通信有什么区别？UART、SPI、I2C、I2S协议对比？ 12](#_Toc57907713)

[24. >>和>>>的区别? 12](#_Toc57907714)

[25. ==和===的区别? 13](#_Toc57907715)

[26. 奇偶检验? 13](#_Toc57907716)

[27. 写一段代码，用半加器组成全加器? 13](#_Toc57907717)

[28. FPGA的基本结构？ 14](#_Toc57907718)

[29. 模块划分的依据以及注意事项？ 14](#_Toc57907719)

[30. 高频设计中，可以采用哪些手段提供系统工作频率？ 15](#_Toc57907720)

[31. 如何提升设计的性能？ 15](#_Toc57907721)

[32. 面积优化方法？ 15](#_Toc57907722)

[33. If-else和case综合出的电路区别？ 15](#_Toc57907723)

Verilog：笔试面试常考易错点整理

[摸鱼范式](javascript:void(0);)

**摸鱼范式**

微信号 icparadigm

功能介绍 研究生在读学生，分享在数字IC学习过程中的一些收获

2020-11-08[原文](https://mp.weixin.qq.com/s/AcBD1rJ1lUXxC76Vu0GEUg)

# Verilog为什么适合描述硬件设计？

* always块间是并行的，符合硬件中电路并行计算的特性
* always时钟触发的特性，符合寄存器的行为
* Verilog是一种自顶向下的层次化设计方法，能够将复杂的大型数字系统划分为规模较小且功能相对简单的单元电路，从而加速大型数字系统的设计、调试等工作
* Verilog的行为级描述方法可以简化硬件电路的设计，可借助于高级语言的精巧结构
* Verilog支持开关级、门级、RTL级、算法级、系统级设计

# 阻塞赋值和非阻塞赋值的区别？

* 阻塞赋值的操作符为=，阻塞的概念是指在同一个always块中，其后面的赋值语句从概念上是在前一句赋值语句结束后在开始赋值的，阻塞赋值在语句结束后立即完成赋值操作，可以认为是顺序执行，用于组合逻辑的设计；
* 非阻塞赋值的操作符为<=，非阻塞赋值的操作可以看作为两个步骤的过程：1)在赋值时刻开始时，计算非阻塞赋值RHS表达式，2)在赋值时刻结束时，更新非阻塞赋值LHS表达式。是一个同时赋值的操作，并行执行，用于时序逻辑的设计

# 为什么用于综合的verilog不建议使用for循环？

* DC无法优化for循环的逻辑，只是单纯进行展开和复制，可能增大电路的面积，降低性能

# 如何防止综合出Latch？

* 在if-else和case中判断所有条件分支
* 在组合逻辑进程中，敏感向量列表要包含所有要读取的信号（包括RHS和判断语句中的信号）（注意：对仿真有影响，但是综合工具会自动补全敏感向量列表，所以在综合之后的电路中是不会有latch的）
* 不要出现自己给自己赋值的情况
* 不要出现组合逻辑环路

# verilog可综合风格？

* 时序逻辑用非阻塞赋值，组合逻辑用阻塞赋值，同一个always块中既有时序逻辑又有组合逻辑时用非阻塞赋值，不要在同一个always块中混合使用，不要在两个及以上always块中对同一个变量赋值
* 所有内部寄存器都能复位，通过复位使信号初始状态可预测
* 不混合使用上升下降沿（可以考虑使用倍频时钟来设计）
* 不使用initial，不要使用延时，不使用循环次数不确定的循环语句
* 防止出现Latch（除非目的性的Latch）

# verilog可综合、不可综合语句汇总？

* 所有综合工具都支持的结构

always，assign，begin，end，case，wire，tri，aupply0，supply1，reg，integer，default，for，function，and，nand，or，nor，xor，xnor，buf，not，bufif0，bufif1，notif0，notif1，if，inout，input，instantitation，module，negedge，posedge，operators，output，parameter

* 所有综合工具都不支持的结构

time，defparam，$finish，fork，join，initial，delays，UDP，wait

* 有些工具支持有些工具不支持的结构

casex，casez，wand，triand，wor，trior，real，disable，forever，arrays，memories，repeat，task，while。

* 注：while在某些情况下是可以综合的，例如while(posedge clk)这和always@(posedge clk)作用一样。

# function和task区别？

* 可综合的任务和函数都只能实现组合逻辑

| **比较点** | **任务** | **函数** |
| --- | --- | --- |
| **输入、输出** | 可以有任意多个各种类型的参数 | 至少有一个输入，不能有输出和双向端口 |
| **调用** | 任务只能在过程语句中调用，而不能在连续赋值语句中调用 | 函数可作为赋值操作的表达式，用于过程赋值和连续赋值语句 |
| **触发事件控制** | 任务不能出现always语句；可以包含延时控制语句（#），事件控制@等，但只能面向仿真，不能综合\*\*（可综合的任务只能实现组合逻辑）\*\* | 函数中不能出现（always、#）这样的语句，要保证函数的执行在零时间内完成 |
| **调用其他任务和函数** | 可以调用其他任务和函数 | 只能调用函数，不能调用任务 |
| 返回值 | 没有返回值 | 只有一个返回值 |
| 其他说明 | 任务调用语句可以作为一条完整的语句出现 | 函数调用语句不能单独作为一条语句出现，只能作为赋值语句的右端操作数 |

# 一段式、两段式、三段式状态机？

* 一段式（要避免）：将整个状态机写在一个always块中，将状态转移判断的组合逻辑和状态寄存器转移的时序逻辑混写在一起，既描述状态转移，又描述转移状态的输入和输出。一段式不利于时序约束、功能更改、调试，不利于综合器和布局布线器对设计优化，不能很好的表示米勒FSM的输出，容易写出Latch。
* 两段式：其中一个always模块采用同步时序描述状态转移，另一个模块采用组合逻辑判断状态转移条件，描述状态转移规律。好处是，不仅便于理解维护，更有利于综合器和布局布线器优化设计，利于添加时序约束。缺点是，由于输出直接采用组合逻辑，容易产生毛刺。
* 三段式：一个always块采用同步时序描述状态转移，第二个采用组合逻辑判断状态转移条件，描述状态转移规律，第三个使用同步时序电路描述次态的输出。相比于两段式，虽然代码结构稍微复杂一些，但是消除了组合逻辑输出的不稳定和毛刺，更利于时序路径分组，获得更好的综合及布局布线结果

# Mealy状态机和Moore状态机区别、优缺点？

* Mealy状态机的输出不但取决于当前状态还取决于输入信号

优点：实现一个功能所需要的状态数相比于moore型更少，更利于设计复杂的状态机

缺点：如果不对输出做同步处理，输出容易出现glitch；由于输出不仅取决于状态还取决于输入，因此输出的译码电路会更复杂，可以认为mealy能跑的频率比moore低一些

* Moore状态机的输出只取决于当前状态

优缺点：互换一下；moore型最重要的特点就是将输入和输出隔离开（但是对于三段式的设计来说，mealy型也是隔离开的）

# 状态机编码选择？

* 二进制编码和格雷码使用较多的组合逻辑，使用最少的触发器
* 而独热编码则相反，独热编码的优势在于比较时仅需比较一个bit，简化了比较逻辑，提高了状态转换的速度，同时也减少毛刺产生概率，增强稳定性，缺点是需要更多的触发器资源
* 小型设计用二进制编码和格雷码更有效，而大型设计用独热编码更高效。

# 独热码优点？

* 可以将特征向量映射到欧式空间，让特征之间的距离计算更合理
* 状态机使用独热码，不需要再译码，可以简化相关组合逻辑
* 减少毛刺产生概率，增强稳定性

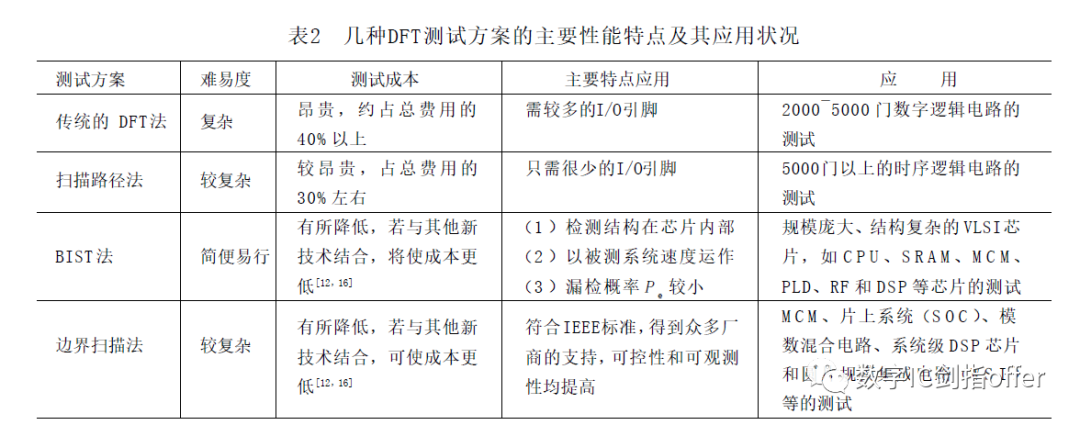
# DFT的意义及常用DFT技术？

**意义：**

* 缩短产品进入市场的时间(TTM)
* 降低测试成本（COT）
* 提高产品质量

**技术：**

* 扫描设计（SCAN）: 将电路中的普通触发器（flip-flops）替换为具有扫描能力的扫描触发器，然后连成扫描链
* 内建自测试（BIST）：靠自身逻辑进行判断，需要管脚大大减少，用于RAM和ROM
* 边缘扫描（Boundary Scan）：每一个输入输出引脚上增加一个存储单元，然后再将这些存储单元连成一个扫描通路，从而构成一条扫描链



# 竞争和冒险、Glitch及消除办法？

* 竞争和冒险：组合逻辑中，由于门的输入信号经过不同的通路会有不同的延时，导致到达的时间不同的现象叫做竞争，由此带来毛刺信号叫做冒险。
* 产生原因：组合逻辑延时导致当输入信号发生变化时，其输出不能同步跟随输入变化，经过一段过渡时间才到达原先期望的状态，这时会产生小的寄生毛刺信号。
* 危害性：毛刺信号并不是对所有输入都有危害，如触发器D端，只要毛刺不出现在时钟上升沿并且满足数据的建立保持时间，就没影响；但当毛刺信号成为系统的控制信号、握手信号、复位信号、时钟信号时就会导致逻辑错误。
* 冒险分类：冒险分逻辑冒险和功能冒险。逻辑冒险是只有一个信号发生变化产生的冒险，是由逻辑表达式导致的，可以通过代数式或卡诺图判断出来；功能冒险是当两个或者两个以上的输入信号发生变化导致的输出端产生毛刺，无法通过修改逻辑表达式来消除。
* 消除办法：

1）利用冗余项消除毛刺：在卡诺图的两圆相切处增加一个圆，可以消除逻辑冒险，但对计数器型产生的毛刺无法消除

2）取样法（选通法）：待信号稳定后加入取样脉冲，就只有在取样脉冲作用期间输出信号才有效（触发器其实也是类似的）

3）吸收法：在输出端增加小电容消除毛刺，但会影响波形，可再加整形电路，该方法不宜使用在中间级

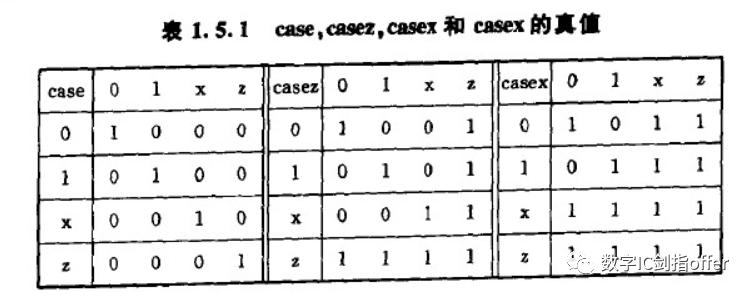
4）锁存法：在输出端加D触发器加以消除，但有时会影响时序，最好采用修改电路等其他办法来彻底消除

5）计数器后接的组合逻辑的glitch可以通过将二进制编码转换成格雷码来完全消除

# 14. case/casez/casex的区别？

三种语句表达式的值是按从上到下的顺序来与分支条件的比较，如果相等，则不再与下面的分支相比较而直接执行该分支的语句

* case语句的表达式的值有4中情况：0、1、z、x。4种是不同的，故表达式要严格的相等才可以操作分支语句。
* casez语句中的表达式情况有三种：0、1、x。不用关心z，z可以和任何数值相等，即z =0.z= 1,z=x;
* casex语句的表达式情况有二种：0、1.不用关心x和z。即x=z=0,x=z=1



* 综合时的差别：（case/casez/casex其实都是可综合的，这一点要记住）

1）case（不是casez/casex的时候）的index列表里面的x和z，都被综合工具认为是不可达到的状态就被去掉了

2）casez和casex里面的x/z都被认为是don't care，所以综合出的电路会是一致的，因此这两个在综合来看没有孰优孰劣

* 注意：要明确的是在case/casez/casex中'?'代表的不是don't care，而是'z'

# 跨时钟域同步方法？（后续会专门推出跨时钟域同步的专题）

* 对于1bit信号，一般来说都是控制信号，方法有

a)如果是慢时钟域到快时钟域，采用两级触发器同步来抑制亚稳态传播，如果快时钟域只要求输出一个时钟周期的有效信号，那么可以采用边沿检测电路；

b)如果是快时钟域到慢时钟域，需要将快时钟域的信号展宽保证慢时钟可以才到，可以采用计数、状态机、握手协议来完成，但这些的前提都是快时钟域的控制信号是不连续的，一般都能满足（如果不行就采用FIFO）；也可以采用窄脉冲检测电路或脉冲同步器。

* 对于多比特的信号，一般来说是数据信号，方法有

a)如果是非连续信号（数据变化速率低于接收时钟），可以采用握手协议或者DMUX

b)如果是连续信号，则必须采用FIFO

# 16. 同步设计的优越性、缺点及设计规则？

**优越性：**

* 可以利用先进STA工具，简化时序分析过程，时序收敛更方便
* 可以更方便的组织流水线，提高芯片速度
* 可以有效避免毛刺，提高设计可靠性
* 工艺兼容性好，可移植性强

**缺点：**

* 由于需要做时钟树的平衡，需要更多逻辑资源，且过高的信后翻转率使得功耗大于异步设计

**设计规则：**

* 尽可能整个设计只使用一个主时钟，同时只使用同一个时钟沿
* 当全部电路不能用同步电路设计时，可以分成若干局部同步电路，局部同步电路之间的接口当作异步接口处理
* 电路实际频率不能大于理论最大频率，留有余量，保证芯片可靠
* 电路中所有寄存器、状态机在复位时都应处于已知的状态

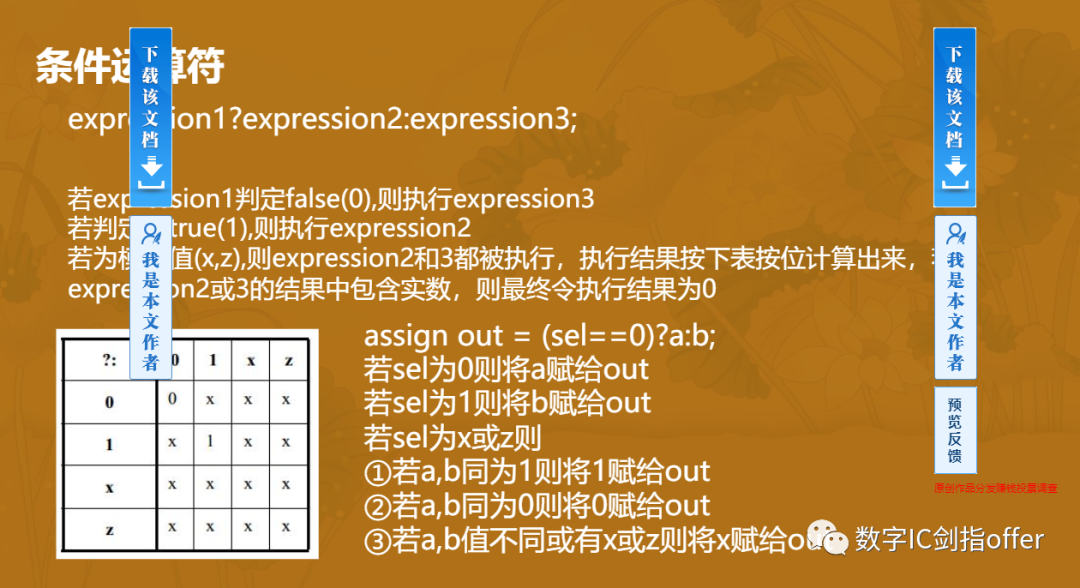
# 17. 同步电路和异步电路的概念？

* 同步电路：电路中存储元件的状态只在时钟沿到来时才能同步发生变化，且是同一个时钟（或者是相位固定的同步时钟），占据了当前数字芯片的绝大多数部分。
* 异步电路：存储元件的状态随输入信号的变化立刻发生变化，信号之间的传递通过握手协议来完成，异步电路时序很复杂，规模通常无法做大，限制了其用途。

# 18. 功能覆盖率、代码覆盖率、断言覆盖率？

* 功能覆盖率：用来衡量哪些设计特征已经被测试程序测试过的一个指标，首要的选择是使用更多的种子来运行现有的测试程序；其次是建立新的约束，只有在确实需要的时候才会求助于定向测试，改进功能覆盖率最简单的方法是仅仅增加仿真时间或者尝试新的随机种子。验证的目的就是确保设计在实际环境中的行为正确。设计规范里详细说明了设备应该如何运行，而验证计划里则列出了相应的功能应该如何激励、验证和测量。功能覆盖率是和设计意图紧密相连的，有时也称为”规范覆盖率“，而代码覆盖率则是衡量设计的实现情况。
* 代码覆盖率：衡量验证进展的最简易的方式是使用代码覆盖率。这种方式衡量的是多少行代码已经被执行过。代码覆盖率衡量的是测试对于设计规范的“实现”究竟测试得有多彻底，而非针对验证计划。多少行代码已经被执行过（行覆盖率），在穿过代码和表达式中的路径中有哪些已经被执行过（路径覆盖率），单比特变量的值是0或1（翻转覆盖率），状态机中有哪些状态和状态转换被访问过（有限状态机覆盖率）。代码覆盖率包括：语句覆盖率（行覆盖率）、分支覆盖率（判定覆盖）、条件覆盖率、路径覆盖率、循环覆盖率、有限状态机覆盖率、翻转覆盖率。（分支覆盖率和条件覆盖率的区别：条件覆盖不是将判定中的每个条件表达式的结果进行排列组合，而是只要每个条件表达式的结果true和false测试到了就OK了。因此，我们可以这样推论：完全的条件覆盖并不能保证完全的判定覆盖）
* 断言覆盖率测量断言被触发的频繁程度。

# 19. 条件运算符对x和z的处理？



# 20. 乒乓buffer的概念？

* ping-pong buffer 也叫双缓存 double buffer, (必须是两个)就是一个缓存在写入的时候, 另一个缓存同时在处理的结构. 用来提高计算机运行速度, 在显示数据处理中常常用到。可以提高系统的数据吞吐率，两个存储器是并行处理的。
* 乒乓buffer是一种常用的提高数据通路带宽的技术，是一种面积换性能的经典设计。个人理解：乒乓buffer并不能提高频率，但是可以提高数据吞吐率。
* 应用场景：

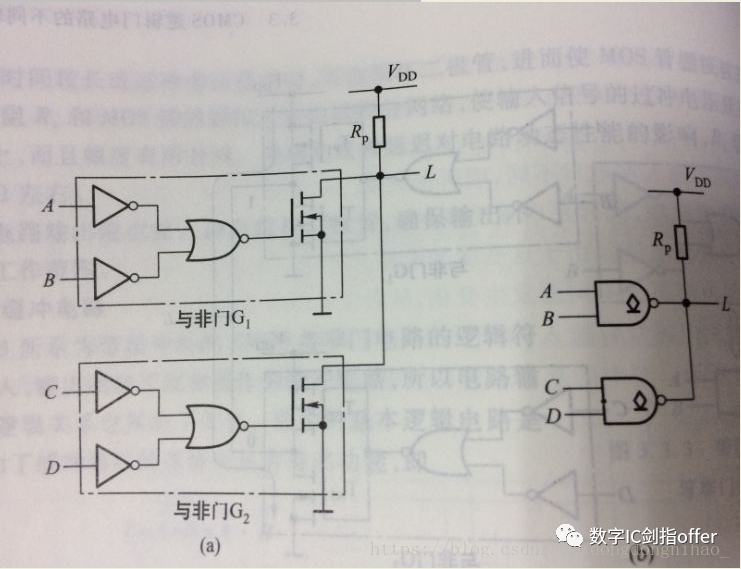
1）下游必须等到上游数据全部写完或者积累到某个程度才能开始读

2）上游必须等到下游数据全部读完或者读到某个程度才能开始写

* 如果没有以上约束，就没必要用乒乓buffer，直接使用普通的buffer即可。乒乓buffer要解决的关键性能点就在于上下游的互相等待这段latency

# 21. 线与逻辑？

* OC（Open Collector）门，又称集电极开路，OD（Open Drain）漏极开路，通过名称就可以判断，OC门是针对三极管来说的，而OD门是针对MOS管而言
* 所谓漏极开路门（OD门）是指CMOS门电路的输出只有NMOS管，并且它的漏极是开路的。使用OD门时必须在漏极和电源VDD之间外接一个上拉电阻（pull-up resister）RP。如图所示为两个OD与非门实现线与，将两个门电路输出端接在一起，通过上拉电阻接电源
* 线与逻辑：即两个输出端（包括两个以上）直接互连就可以实现“AND”的逻辑功能。如下图，当两个与非门的输出全为1时，输出为1；只要其中一个输出为0，则输出为0，所以该电路符合与逻辑功能，即L=(AB)'(CD)'



# 22. Modelsim仿真步骤？

* vlib->vmap->vlog->vsim->run

# 23. 同步通信和异步通信有什么区别？UART、SPI、I2C、I2S协议对比？

* 同步通信需要相同频率的时钟，逐字符发送接收，发一个收一个，收一个发一个，不能有间隙。异步通信可以任意间隙，接受端随时准备，发送端任意时刻发送 ，需要加停止位和开始位。
* UART

（1）两线（TX、RX） （2）异步进行，有标准波特率 （3）收发独立，全双工 （4）通信双方地位平等 （5）无应答机制，可配置奇偶校验位 （6）速度较慢 （7）数据传输低位在前，高位在后

* SPI

（1）四线（SCLK、MOSI、MISO、CS） （2）同步进行 （3）收发独立，全双工 （4）主从通信机制，由主设备产生SCLK （5）无应答机制 （6）速率可达几Mbps （7）数据传输高位在前，低位在后

* I2C

（1）两线（SCL、SDA） （2）同步进行 （3）同一时间只能由一方发送或接受数据，半双工 （4）多主控总线，任何一个设备都能像主控器一样工作，并控制总线 （5）具有应答机制 （6）传输速率有标准模式（100 kbps）、快速模式（400 kbps）和高速模式（3.4 Mbps） （7）数据传输高位在前，低位在后

* I2S

主要用来传输音频信号 （1）三线（BIT\_CLK、SDATA、WS），WS用来选择左右声道 （2）同步进行 （3）半双工 （4）主从通信机制，由主设备产生BIT\_CLK、SDATA、WS （5）无应答机制 （6） （7）数据传输高位在前，低位在后

# 24. >>和>>>的区别?

* >>是逻辑右移，左边补0
* >>>是算术右移，考虑符号位，如果是0，则左边补0，如果是1，则左边补1

# 25. ==和===的区别?

* 两者都是相等或比较运算符。“==”检查二值逻辑相等，而“===”运算符测试四值逻辑相等。
* 使用“==”比较二值逻辑，如果出现X或者Z，则结果为X。
* 使用“===”比较四值逻辑，如果出现X或Z，则结果为0或1，能够正确的进行比较。

# 26. 奇偶检验?

* 奇偶校验位是在一串二进制码的最后添加的一位，它使得整个二进制串的1的个数为奇数或者偶数。因此奇偶校验分为两种，奇校验和偶校验。
* 计算校验位需要对二进制码中的1进行计数。如果1的数量为奇数，并且使用偶校验，则校验位为1，使得整体1的个数为偶数。如果1的数量为偶数，并且使用偶校验，则校验位为0，使得整体1的个数为偶数。奇校验类似。奇偶校验位可以通过对所有的比特位进行异或得到。

# 27. 写一段代码，用半加器组成全加器?

module half\_adder(input\_0, input\_1, sum, carry);

input input\_0, input\_1;

output sum, carry;

assign sum = (input\_0)^(input\_1);

assign carry = (input\_0)&(input\_1);

endmodule

module full\_adder(input\_0, input\_1, input\_2, sum, carry);

input input\_0, input\_1, input\_2;

output sum, carry;

reg sum\_intermediate, carry\_intermediate\_0, carry\_intermediate\_1;

half\_adder ha1(input0,input1,sum\_intermediate,carry\_intermediate\_0);

half\_adder ha2(sum\_intermediate,input2,sum,carry\_intermediate\_1);

assign carry = (carry\_intermediate\_0)|(carry\_intermediate\_1);

endmodule

# 28. FPGA的基本结构？

FPGA由6部分构成，分别为可编程输入/输出单元、基本可编程逻辑单元、丰富的布线资源、嵌入式块RAM、底层嵌入功能单元、内嵌专用硬核

* 可编程输入/输出单元：可通过配置适应不同的电器标准和I/O物理特性；可调整阻抗特性、输出驱动电流等。
* 基本可编程逻辑单元：由LUT和寄存器构成，通过LUT来实现组合逻辑
* 丰富的布线资源：包括全局性的专用布线资源、长线资源、短线资源等；
* 嵌入式块RAM：可配置为单端口RAM、双端口RAM、CAM、FIFO等存储结构
* 底层嵌入式功能单元：指通用程度较高的嵌入式功能块，如PLL、DSP等；
* 内嵌专用硬核：通用性较弱的硬核，并不是每个FPGA都有的

# 29. 模块划分的依据以及注意事项？

**划分依据：**

* 按照逻辑功能划分：如IO、Memory、PLL、CORE、JTAG等，CORE中继续按照功能划分，这样做的好处，一个是可以很好的复用资源，另一个是更利于综合工具优化时序
* 按照设计大小和规模划分：模块大小适中，设计大小一般设为一晚上的运行时间，白天人工处理和处理结果。模块太大，对综合工具和计算机配置要求高，模块太小，会造成层次过多，不利于综合工具优化
* 按照时钟域划分：除了跨时钟域同步模块，其余模块每个仅有一个时钟，有利于时序约束和综合工具优化

**注意事项：**

* 顶层模块应只是各个子模块的连接，不能有胶合逻辑
* 寄存模块的输出
* Pad与core分离，I/O、三态都应在顶层描述
* 结构层次不能太多

# 30. 高频设计中，可以采用哪些手段提供系统工作频率？

* 流水线
* 寄存器配平
* 关键路径优化
* 串转并（如RCA改成超前进位加法器）
* 树形结构
* 迟滞信号后移
* DC逻辑优化中的flatten
* 采用先进工艺

# 31. 如何提升设计的性能？

性能可以由带宽、吞吐率、时延来评估

* 空间并行：面积换性能（更宽的总线等等）、DC逻辑优化中的flatten（将复杂的逻辑运算转换成积之和的形式）
* 时间并行：流水线（充分调度资源，使得尽可能少的处于闲置状态）、行波进位加法器改成超前进位加法器
* 算法和硬件的优化，硬件优化比如工艺、关键路径优化

# 32. 面积优化方法？

* 资源共享
* 串行化
* 后端：对正slack的路径，移除buffer或者downsize

# 33. If-else和case综合出的电路区别？

* if-else，assign a = c? b : d这种写法会综合出MUX
* case，assign a = c&b | ~c&d这种写法会综合出AOI