【笔试解析】2020华为海思校招芯片岗笔试

**数字ICer**

微信号 Studying\_Times

功能介绍 研究生在读，分享数字IC、FPGA、Linux、Python、UVM、SV、信号处理等学习笔记，快乐学习，一起进步。

**1.    表示任意两位无符号十进制数需要（7）位二进制数。**

**2.    时间尺度定义为timescale 10ns/100ps，选择正确答案(b)**

**a. 时间精度10ns**

**b. 时间精度100ps**

**c. 时间单位100ps**

**d. 时间精度不确定**

解析：10ns为时间单位，100ps为时间精度。

时间单位主要用在`timescale定义之后的代码中，直至遇到另一个`timescale指令或`resetall指令为止，用来表示verilog语法中使用到时间的表达式的单位。比如：

*and # (5.22, 6.17 ) Al (Z, A, B);* 里面的5.22和6.17分别代表52.2ns和61.7ns

或者*always@(posedge clk) out <= #3 in;* 里面的3代表30ns。

时间精度一般体现在波形的时间横轴上，最小的一个时间单位跨度即时间精度100ps。

**3. 时序逻辑电路不仅与输入有关，还与原来的状态有关。（正确）**

**4. 同步复位需要进行recovery和removal检查，异步复位不需要进行recovery和removal检查（错误**）

解析：recovery和removal存在于拥有async reset端口的寄存器，因此异步复位需要做这种检查。这种检查的目的是为了保证复位释放的时刻能早于时钟沿并且保持到时钟沿到来（recovery，恢复），并且保证复位置位的时刻能晚于时钟沿一段时间（removal，清除），以此来避免复位释放的时候出现亚稳态。类似于寄存器D端的setup和hold检查。

**5.  在异步FIFO设计中，满信号由写时钟产生，空信号由读时钟产生（正确）**

解析：写时钟驱动写指针增加，和FIFO有效数据的增加，有效数据到达深度时满信号拉高；读时钟驱动读指针增加，和FIFO有效数据的减少，有效数据为0时空信号拉高；

**6.关于false path，正确的是（d）**

**a. 一般异步电路可以设置为false path**

**b. 两个不同频率的接口一定可以设置为false path**

**c. 一般异步复位可以设置为false path**

**d. 一般模拟ip和系统的互联接口都可以设置为false path**

解析：

a是错误的。设置false path的原因是该路径没必要满足特定时序（比如setup/hold,recovery/removal,clock gating，max delay等特定时序）。虽然异步电路时钟之间没有固定的相位关系，工具没法直接做reg-reg的setup/hold检查，但是可以设置max delay等特殊约束对需要满足一定时序关系的异步电路做检查，需要看具体的场景。

b是错误的。频率不同和false path没有任何关系。而异步电路的频率可能不同，但是频率不同不等于异步电路，异步电路的本质是来自不同晶振的时钟源驱动的电路，他们频率甚至也可以相同。

c是错误的。对于异步复位，指的只是寄存器使用了异步复位端口async\_reset，在复位的时候复位信号可以是异步的。但它在释放的时候是需要满足recovery-removal时序检查的，需要做同步释放，因为做了同步，所以STA工具有能力做检查。所以异步复位不能设为false path，需要检查recovery-removal时序。

d是正确的。模拟ip的时钟频率一般远小于数字电路，因此对数字电路的时序不敏感，互联接口可以设置false path。

**7.下面是一个什么电路：(c)**

***always@(posedge clk or negedge rst\_n) begin***

***If(rst\_n == 1’b0)***

***a<=2’b0;***

***elseif(b>2’b0)***

***a<= b***

***end***

**a.    综合为latch**

**b.    带同步复位的d触发器**

**c.    带异步复位的d触发器**

**d.    组合逻辑**

解析：posedge clk决定了这是一个触发器，if(rst\_n == 1’b0)和negedge rst\_n决定了这是一个异步复位的d触发器，复位和时钟无关（异步）。

**8. 对于信号定于语句：reg[0:4] always,a;说法错误的是（b）**

**a. 不能使用关键字定义信号名**

**b. 信号定义为reg型，只能使用在时序电路的赋值中**

**c. bit定义顺序应该从高到低**

**d. 每个信号应单独用一行来声明**

解析：a和c违背verilog语法，是错的。d只是建议，并不违背verilog语法。相比之下，b是绝对错的，因为reg型可以用于组合逻辑赋值，比如always(\*)。

**9. 下列说法错误的是（d）**

**a. 条件语句，如果无优先级关系，尽量采用case，避免多级else if，影响时序；**

**b. If/else语句配对使用，case语句加default项；组合逻辑中在所有条件下都要对信号幅值，如果要保持步便用a=a方式赋值；**

**c. 两个向量进行比较操作或加减操作或赋值操作时，两个向量的位宽要相等，避免隐式扩展。**

**d. 可综合代码中，除了for语句中的循环变量可以定义为integer型以外，所有变量和信号都只能为wire或reg型，不能定义为整型，实数型，无符号型，realtime型和time型。**

解析：

a是正确的，优先级条件语句会综合成串联的选择逻辑，时序较差，而case可以综合成并行的选择逻辑，n个条件逻辑级数为log2n。

b是正确的，但不严谨。if/else可以不配对使用，在时序逻辑下，可以只有if没有else，这时候else分支保持寄存器原值，当然有else使代码更清晰。有case语句，如果是组合逻辑，只要有条件没给赋值，会综合出latch。

c是正确的。两个向量位宽不匹配会隐式拓展高位，如果有操作数定义为wire类型但是作为有符号数使用，隐式拓展会错误高位补0，拓展成无符号数，功能出错。

d是错误的，integer可以被定义成信号，会综合出32bit的register。但因为位宽固定，一般不这么使用。另外在verilog-2001中加入了无符号型和有符号型的reg/wire定义，也属于可综合类型。其他类型不可综合。

**10.关于异步设计的危害，下面说法错误的是：(b)**

**a.  信号的时延随着每次布局布线的不同而不同，随着pvt的改变而改变，因此可靠性很差，而且不容易移植**

**b. 异步设计会产生毛刺**

**c.  异步设计不能做静态时序分析**

**d. 异步设计会带来很大的同步翻转噪声**

解析：

a是正确的。现在大部分的设计都是同步设计，小部分高性能低功耗电路可能会考虑使用异步设计。异步设计的好处之一便是可以最大限度利用时间窗的margin。有点类似latch的timeborrow，无需等待同步打拍，并且没有了时钟信号，功耗可以大大降低。坏处是需要特殊的握手机制来保证时序，而握手机制又很大程度上依赖于信号的时延，像a所说，时延不可控型较强，因此可靠性较差，而且不容易移植。

b是错的，毛刺并非异步设计导致，即便是同步设计，只要信号传递过程中中间结果和最终输出不同（卡诺图可以清楚看到），也会有毛刺，只不过异步设计的毛刺比较容易导致功能错误，因为没有时序检查，没有setup/hold来保证毛刺消除。

c是对的。现在的静态时序分析都是基于同步设计，异步设计没有时钟的概念，所有信号没有固定的相位关系，工具无法建模分析。

d是对的。这里的翻转噪声主要来自于信号线之间的耦合电容。如果两个异步信号之间有耦合电容，那么他们的噪声比同步信号之间大得多，因为他们的电平跳变相互影响的时间窗口更大。

**11.  多bit总线信号可以通过格雷码转换进行异步处理，例如：8bit的数据总线进行格雷码转换，然后通过双触发器法实现异步处理（正确）**

解析：格雷码将多bit的翻转转化为单bit的翻转，解决了多bit异步信号传播过程中因时延不同而导致的数据采集端采集信号错误的问题。

**12. System Verilog中，下面哪种数组在使用前需要执行new操作（c）**

**a. 压缩数组**

**b. 联合数组**

**c. 动态数组**

**d. 多维数组**

解析：动态数组通过new（）函数预先分配存储空间。

**13. 电路设计中，只要采用静态时序分析就可以保证电路设计的准确性，不需要再进行动态时序分析了。(错误)**

解析：现在大部分的同步电路设计流程都是rtl验证+formality形式验证+sta静态时序分析。通过rtl验证保证功能完备，通过formality保证rtl和网表一致性。因为有了formality保证网表和rtl的一致性，因此如果rtl验证完备，一般情况下就可以保证网表功能验证完备。然后通过对网表进行静态时序分析（sta）来保证没有时序风险。动态时序分析（后仿）主要是跑一些典型场景，保证在典型场景下不会有时序问题，增强设计信心，对于同步电路，只要sta阶段检查仔细，问题一般不大。但是设计中难免会有一小部分异步电路无法进行静态时序分析，因此必须需要通过后仿来进行排查。

**14. 对于initial语句，说法错误的是（d）**

**a. 在仿真过程中只执行一次**

**b. 这是一种过程结构语句**

**c. 在模拟的0时刻开始**

**d. 可用于给实际电路赋初值**

解析：initial是不可综合语句，无法变成实际电路。

**15. 线型信号必须显性定义（错误）**

解析：1bit的线型信号不需要显性定义也可以被仿真工具识别，综合工具自动生成。但是最好还是都显性定义，好处主要有2个。一是在定义的时候将需要的信号定义全，帮助提前建立整个电路通路的框架。二是养成好习惯，避免多bit信号也没定义，那么多bit信号会被自动转化为单bit信号，功能不符合预期。

**16. 以下名称不属于verilog关键字的是（d）**

**a. genvar**

**b. endcase**

**c. cmos**

**d. cnt**

**17.  当功能覆盖率达到100%，可以说明：（b）**

**a. 功能覆盖率对应的DUT响应是正确的**

**b. 某些令人关注的情况已经得到测试覆盖**

**c. DUT的功能点已经100%覆盖**

**d. 意味着验证的完整性**

解析：功能覆盖率是通过cover语句来收集电路中信号的翻转情况，保证在大规模随机回归的情况下令人关注的信号是否翻转到期望的值。一些特定信号翻转到期望的值表示一种场景的出现，比如fifo的满信号从0到1表示fifo满的场景被覆盖到。因此b是正确的。

a是错误的，因为被覆盖到不意味着电路的响应是正确的，只是被测试到。

c是错误的。功能覆盖的cover语句是验证人员根据自己的测试点提取来写的，并不一定是所有的功能点，只能说验证人员关注的功能点被覆盖。

d是错误的，功能点都不能保证是100%，更不用说验证完整了。验证完整这句话不是随便可以说的，即便是真的做足了各种验证工作，都要对可能的疏忽保持敬畏之心。芯片的流片成功并不容易。

**18.  下面表达式正确的是（c）**

**a.    A=4’df**

**b.    C=3’d8**

**c.    B=5’h1**

**d.    D=3’b2**

**19.  RTL代码是测试点的一个重要来源（正确）**

解析：测试点的提取可以大致分为黑盒测试，白盒测试和灰盒测试。

黑盒测试直接通过对产品功能定义的解读，完成端到端测试，通过输入导出预期输出，无需了解中间的RTL代码。

白盒测试则是通过对RTL代码的解读，分解出代码实现过程中可能出的问题，但是黑盒测试又不容易测试出来的点，利用断言进行检测。因此RTL代码是白盒测试点的一个重要来源。

灰盒测试则介于黑盒和白盒之间，通过定向用例测试一些RTL内部实现的细节，观察其输出正确性，辅以断言检测。

**20.  时序电路中，既可以使用阻塞语句，也可以使用非阻塞语句（错误）**

解析：时序电路使用非阻塞语句。这点可以作为既定规则记住并使用。如果明白其背后verilog语言的编译机制很好，但如果不明白也没太大关系。毕竟我们是用这门语言设计硬件的，更多的应该关注如何使用它。就像老外可能搞不懂英语背后各种单词的属性，但不妨碍他们流利地使用英语交流。

**21. 数字电路设计中，下列哪种手段无法消除竞争冒险现象（d）**

**a.    加滤波电容，消除毛刺**

**b.    增加冗余项消除逻辑冒险**

**c.    增加选通信号，避开毛刺**

**d.    降低时钟频率**

**解析：在同步电路中，降低时钟频率可以让寄存器采集数据的时候避开毛刺，因为假定时序路径延时不变，频率降低使时钟沿后移，setup time不变的情况下给了更多的timing margin使毛刺恢复。但与前3个选项相比，这不是从根本上消除毛刺，该毛刺信号还是会被电路传递，只不过降低了setup违例的风险。而在异步电路中，不论时钟频率是多少，毛刺都有被下游寄存器采集的风险。**

**22. 异步时钟数据采样的方法错误的是（d）**

**a.    单bit高频时钟脉冲展宽后给低频时钟进行采样**

**b.    握手信号后再采样**

**c.    使用fifo隔离进行多bit的采样**

**d.    高频时钟直接采样低频时钟的多bit数据**

**解析：多bit数据存在各bit到达采样点的时刻不同的问题，直接采样可能会采集到错误的数据。要通过握手或者fifo或者格雷码转换单bit等方式进行多bit数据传输。**

**23. 以下不能对多bit的数据总线的时钟异步处理的是：（c）**

**a.    Dmux synchronizer**

**b.    Gray-code**

**c.    寄存器同步**

**d.    FIFO**

**解析：同上一题，多bit数据总线不能直接用寄存器同步方式进行异步处理。**

**24. 一个4bit位宽的有符号数a=4’b1001取绝对值后的值为abs(a)=4’b0110（错误）**

**25. 对于两个位宽相同的无符号数a和b, a>b和a正确）**

**26. 组合逻辑电路的特点是输出信号只是该时刻输入信号的函数，它是无记忆功能的。时序逻辑电路的特点是任何时刻产生的稳定输出信号不仅与该时刻输入信号有关，而且与它过去的状态有关（正确）**

**27. 高的功能覆盖率意味着高的代码覆盖率（错误）**

**解析：功能覆盖率和代码覆盖率没有直接的关系。功能覆盖率主要表征测试点关注信号的翻转情况。代码覆盖率包括了：信号翻转覆盖率，条件覆盖率和代码行覆盖率。可以想象一个极端场景：DUT是一个CPU，测试点只观察CPU里的加法器。加法器测全了，功能覆盖率高了，但是代码只测了冰山一角。**

**如果反过来表达，高的代码覆盖率也同样不意味着高的功能覆盖率。假设DUT是一个8bit无符号加法器，测试两个向量：a=b=8’b1111\_1111，和a=b=8’b0理论上就能基本将代码覆盖率打满，但是这只是所有操作数组合的其中两个，很明显功能覆盖率不可能高。**

**28. 为了发现逻辑缺陷，要想到什么测什么，一定要遍历所有可能（错误）**

**解析：头脑风暴挖掘测试点是必须的，但是要保证测试点是符合规格说明和约束的，并且是可能会发生的场景。**

**29. 在verilog中，关于函数，不正确的是（a）**

**a.    函数可以调用其他函数**

**b.    函数只能返回一个值**

**c.    函数必须至少带有一个输入**

**d.    不能包含任何时延或时延控制**

**解析：函数是可综合语句，可以综合成组合逻辑。b,c，d是正确的，a是错误的，都是语法规定，没有什么原因可以追溯，当成公理应用即可。**

**30. 在systemverilog中，任务task可以调用函数function（正确）**

**31. 下面哪个语句是可以综合的（c）**

**a. initial过程块**

**b. forever 循环语句**

**c. for循环语句**

**d. fork join**

**解析：for循环语句多用在多bit信号的同类型操作或者递归操作上。可以放在组合逻辑always块中，或者与generate结合使用。Initial, forever和fork join主要用在验证环境的进程控制上。**

**32. 同步时序电路和异步时序电路比较，其差异在于后者（c）**

**a. 没有触发器**

**b. 没有稳定状态**

**c. 没有统一的时钟脉冲控制**

**d. 输出只与内部状态有关**

**解析：异步时序电路是一种没有全局时钟的设计，所以c是正确的。异步时序电路有多种设计方式，有的设计方式会使用到触发器。并且是有稳定状态的。输出与内部状态和输入都有关系。因此a,b,d都错误。**

**33. 在模块实例化语句中，悬空端口可通过将端口表达式表示为空白指定为悬空。模块的输入端悬空值为（d）**

**a. 0**

**b. x**

**c. 1**

**d. z**

**解析：模块输入不允许有悬空值。悬空则没有驱动，为高阻态z。**

**34. 下列电路中属于时序逻辑电路的是（b）**

**a.    译码器**

**b.    计数器**

**c.    编码器**

**d.    数据选择器**

**解析：一般来说计数器每个周期统计一次进行累加，需要依赖于前一个周期的状态。其他三种基本电路都是组合逻辑，不需要依赖于过去的状态。**

**35. 以下说法错误的有（a）**

**a. 为了避免wire信号出现x态，最好是在声明时，赋初始值0**

**b. 一个模块例化多次，可以使用generatefor循环减少代码**

**c. 模块例化时，需要将端口显式列出，即使某个端口未连接信号**

**d. 循环表达式的循环次数必须为常数**

**解析：如果声明时将wire信号先赋初始值0，而后使用assign语句，相当于多驱动电路，很明显是错的。只有在组合逻辑的always块中可以先将输出信号赋初始值0。**

**36. 某个状态下，不关心某个寄存器的输出值，那么将其设计为输出0，可以降低功耗（错误）**

**解析：比起设计为输出0，降低功耗更好的做法是保持寄存器原值。因为功耗来自于信号toggle，如果在上一状态寄存器输出为1，下一状态下输出为0，即便0不使用，也产生了1到0的跳变，同样会有功耗，既然不关心，还不如保持输出为1。**

**37. 高频时钟域的数据总线（每时钟周期都变化）传递给低频时钟域，哪种同步方式正确？（b）**

**a.    使用握手信号进行同步**

**b.    使用异步fifo**

**c.    使用同步fifo**

**d.    使用打2拍进行同步**

**解析：c和d很明显错误。多bit数据不能使用打两拍同步。这道题的难点在于a选项。一般而言握手信号可以用于高频时钟域到低频时钟域的同步，但是为了保证数据不被漏采，从高频到低频握手信号valid\_tx需要做信号宽度拓展，至少拓展为1.5个周期的rx时钟，因此相应的数据data\_tx也要保持多个n个tx周期不变，n=1.5\*clk\_rx/clk\_tx。这与题目条件数据总线每周期都变化相矛盾。因此只有b是正确的。**

**38. 同步电路设计中出现setup time不满足，不可以采用下面哪种措施解决（d）**

**a.    减小信号延迟**

**b.    Pipeline**

**c.    降低时钟频率**

**d.    增加时钟频率**

**解析：setup time是由于时序路径延迟td+tsetup > tclk，而增加时钟频率则减小tclk，保持前两项不变，相当于恶化了setup time的满足。**

**39. 编写verilog程序时，变量的定义不可与关键字冲突（正确）**

**40. 关于亚稳态描述正确的是（d）**

**a. 异步逻辑不会产生亚稳态**

**b. 采用格雷码能消除亚稳态**

**c. 亚稳态不会传递，所以不会导致相关逻辑处于不确定态**

**d. 在时钟有效延的时候外部数据未稳定，导致触发器不能判断数据电平的状态**

END