【第一章】1、FPGA 芯片的发展主要体现在哪几个方面？未来的发展趋势是什么？

答：新型芯片的规模越来越大，成本越来越低，低端的 FPGA 已逐步取代了传统的数字元件。

先进的 ASIC 生产工艺已经被用于 FPGA 的生产，越来越丰富的处理器内核被嵌入到高端的 FPGA 芯片中，基于 FPGA 的开发成为一项系统级设计工程。随着半导体制造工艺的不同提高，FPGA 的集成度将不断提高，制造成本将不断降低，其作为替代 ASIC 来实现电子系统的前景将日趋光明。

2、EDA 技术的优势是什么？

答：1.用 HDL 对数字系统进行抽象的行为与功能描述以及具体的内部线路结构描述，从而可以在电子设计的各个阶段、各个层次进行计算机模拟验证，保证设计过程的正确性，可以大大降低设计成本，缩短设计周期。

2.EDA 工具之所以能够完成各种自动设计过程，关键是有各类库的支持。

3.某些 HDL 也是文档型的语言，极大地简化了设计文档的管理。

4.EDA 具有日益强大的逻辑设计仿真测试技术，极大地提高了大规模系统电子设计的自动化程度。

5.基于 EDA 技术的设计，由于用 HDL 表达的成功的专用功能设计在实现目标方面有很大的可选性，它既可以用不同来源的通用 FPGA/CPLD 实现，也可以直接以 ASIC 来实现，设计者拥有完全的自主权。

6.EDA 技术的设计语言是标准化的，不会由于设计对象的不同而改变；它的开发工具是规范化的，EDA 软件平台支持任何标准化的设计语言；它的设计成果是通用性的，IP 核具有规范的接口协议。良好的可移植与可测试性，为系统开发提供了可靠的保证。

7.EDA 技术能将所有设计环节纳入统一的自顶向下的设计方案中。

8.EDA 不但在整个设计流程上充分利用计算机的自动设计能力，在各个设计层次上利用计算机完成不同内容的仿真模拟，而且在系统板设计结束后仍可利用计算机对硬件系统进行完整全面的测试。

3、EDA 的设计流程包括哪几个环节？

ANS: ①设计输入（原理图/HDL 文本编辑）②综合 ③ FPGA/CPLD 适配 ④时序仿真与 功能门级仿真 ⑤FPGA/CPLD 编程下载 ⑥FPGA/CPLD 器件电路硬件检测。

4、硬件描述语言的种类有哪些？

ANS: VHDL 、Verilog HDL、SystemVerilog、System C 等

答：VHDL、Verilog、HDL、System Verilog、System C。

5、自顶向下设计方法的优点是什么？

ANS: 过程大部分由计算机完成，可植性强，便于系统的优化和升级，以及对模型进行及 时的修改，以改进系统或子系统的功能，更正设计错误，提高目标系统的工作速度，减小面 积耗用，降低功耗和成本等。在 EDA 技术应用中，自顶向下的设计方法，就是在整个设计 流程中各设计环节逐步求精的过程。

6、ip 核可分为哪几类？

ANS: ①软 IP 、②固 IP、③硬 IP

7、ip 在 EDA 技术的应用和发展中的意义是什么？

ANS: IP 就是将某些功能固化，而当 EDA 设计也需要这些功能的时候，就可以

直接将植 入了此功能的 IP 拿过来直接用，而不用再重新设计。这样既可以提高效率又可以减少设计 风险。IP 核具有规范的接口协议，良好的可移植与可测试性，为系统开发提供了可靠的保 证。

8、可编程逻辑器件经历哪些发展过程？

答：它大致经历了从 PROM、PLA、PAL、GAL、EPLD、FPGA 和 CPLD 的发展过程。

FPGA 的配置方式有哪些？

PS 被动串行模式、PPS 被动并行同步模式、PPA 被动并行异步模式、PSA 被动串行异步模式、JTAG 方式、AS 主动串行模式。

JTAG：MSEL 都为 0。

9、VHDL 中标识符的命名规则是什么？

答：标识符是设计者在 VHDL 程序中自己定义的，用于标识不同名称的词语。

1. 有效的字符：包括 26 个大小写英文字母，数字包括 0~9 以及下划线；
2. 任何标识符必须以英文字母开头；
3. 必须是单一的下划线，且前后都要有字母或数字；
4. 标识符中的英文字母不区分大小写；
5. 允许包含图形符号，包括空格等。

10、端口模式有哪些？

1. in：输入端口。定义的通道为单向只读模式。规定数据只能由此端口被读入实体中；
2. out：输出端口。定义的通道为单向输出模式。规定数据只能通过此端口从实体向外流出，或可以将实体中的数据向此端口赋值；
3. inout：双向端口。定义的通道确定为输入输出双向端口；

buffer：缓冲端口。功能与 inout 类似，区别在于当需要输入数据时，只允许内部回读输出的信号，即允许反馈。与 inout 模式相比，buffer 回读的信号不是由外部输入的，而是由内部产生、向外输出的信号。

11、VHDL 中有哪些基本的数据类型？

ANS: bit、 bit_vector、 std_logic、 std_logic_vector、 boolean （布尔） natural（自然数） integer 、 、（整数）、signed（有符号）、unsigned（无符号）、array（数组类)、record（记录类型） 、 Subtype（子类型）、用户自定义类型。

12、常用的 VHDL 程序包有哪些？

ANS ： STD_LOGIC_1164 、 STD_LOGIC_ARITH STD_LOGIC_UNSIGNED

答：std_logic_1164、std_logic_arith、std_logic_signed、std_logic_unsigned

13、verilog 中两种基本的数据类型 net（wire）和 reg 的区别。

reg 型主要用于定义特定类型的变量，寄存器变量

wire 型应用于 assign 语句中，且 assign 语句中必须要用 wire 网线型变量。

reg 相当于存储单元，wire 相当于物理连线

wire 表示直通，即只要输入有变化，输出马上无条件地反映；reg 表示一定要有触发，输出才会反映输入

wire 对应于连续赋值，如 assign

reg 对应于过程赋值，如 always，initial

reg 型保持最后一次的赋值，而 wire 型则需要持续的驱动。

14、verilog 中的时钟过程表述的特点和规律

1.某信号被定义成边沿敏感时钟信号，则 posedge A 或 negedge A 放敏感表中，always 结构块中不能再出现信号 A 了。

2.若 B 被定义成对应于时钟的电平敏感异步控制信号，则除 posedge B 或 negedge B 放敏感表中，always 块中必须给出逻辑描述，即表述上是边沿敏感，性能上是电平敏感。
3.若某信号对于时钟同步，则不能出现在敏感信号表中。
4. 敏感表中边沿敏感信号和电平敏感信号不能同时出现。
15、阻塞式赋值和非阻塞式赋值的区别
阻塞式赋值是顺序执行符号为“=”是时钟触发，非阻塞时赋值是并行执行的符号为“<=”是边沿触发。在组合逻辑建模中应使用阻塞赋值； 在时序逻辑建模中应使用非阻塞赋值。
16、verilog 语言有哪几种描述风格？
1. 行为描述；
2. 数据流描述；
3. 结构描述。
17、任务和函数语句的区别
任务就是一段封装在“task…endtask”之间的程序。任务可以彼此调用，而且任务内还可以调用函数。任务调用语句只能出现在过程块内； 任务的输出端口必须和寄存器类型的数据变量对应。
函数的调用也是通过函数名来完成的，而且它在函数结构体内代表一个内部变量，函数调用的返回值就是通过函数名变量传递给调用语句的。a.函数定义只能在模块中完成，不能出现在过程块中；b. 函数至少要有一个输入端口，但不能包含输出和双向端口;c. 在函数结构中，不能使用任何形式的时间控制语句（#、wait 等），也不能使用 disable 中止语句；d. 函数定义结构体总不能出现过程块语句；e. 函数内部可以调用函数，但不能调用过程。
18、状态机的优点
1. 高效的顺序控制模型；
2. 容易利用现成的 EDA 优化工具；
3. 性能稳定；
4. 设计实现效率高；
5. 高速性能。
19、状态机的状态编码有哪几种？各自的优缺点是什么？
1. 直接输出型编码：这种编码最典型的应用就是计数器。直接输出型编码方式就是所谓的用户自定义编码方式，它的优点是输出速度快，不太可能出现毛刺现象。缺点是程序的可读性差，用于状态译码的组合逻辑资源比其他以相同触发器数量触发器构成的状态机多，而且控制非法状态出现的容错技术要求比较高。
2. 顺序编码：优点是这种编码方式最为简单，在传统设计技术中最为常用，其使用的触发器最少，剩余的非法状态也最少，容错技术较为简单。
缺点也很多，如常常会占用状态转换译码组合逻辑较多的资源，特别是有的相邻状态或不相邻状态的状态转换时涉及多个触发器的同时状态转换，因此将耗费更多的转换时间，而且容易出现毛刺现象。
4. 一位热码状态编码：一位热码状态编码虽然占用了较多的触发器，但其简单的编码方式大为简化了状态译码逻辑，提高了状态转换速度，增强了状态机的工作稳定性，这对于含有较多的时序逻辑资源、相对较少的组合逻辑资源的 FPGA 器件是最好的解决方案。

16 分频

```
module clk_div_16( clk_in, rst_n, clk_out);
 input clk_in;
 input rst_n;
 output clk_out;
 reg [2:0] cnt;
 reg clk_out_t;
 always @(posedge clk_in)
      begin
     if (!rst_n)
          begin
                                    cnt <= 0;
               clk_out_t <= 0;
                              end
                         else
                              begin
                                 if (cnt == 3'b111)
                                   begin
                                        cnt <= 3'b000;
                                        clk_out_t <= ~clk_out_t;
                                   end
                                 else
                                    begin
                                        cnt <= cnt + 3'b001;
                                    end
                               end
          end
   assign clk_out = clk_out_t;
endmodule
```

奇数分频 3 分频

```
module clk_div_3(clk_in, rst_n, clk_out);
input clk_in;
input rst_n;
output clk_out;
reg [1:0] cnt, cnt1;
reg clk_1to3p, clk_1to3n;
always @(posedge clk_in) begin //上升沿 3 分频，占空比为 1:2？
if(!rst_n) begin
 cnt <= 0;
clk_1to3p <= 0;
 end
 else
begin
 if(cnt == 2'b10) begin
```

```
    cnt <= 0;
clk_1to3p <= clk_1to3p;
end
else begin
    cnt <= cnt + 1;
clk_1to3p <= !clk_1to3p;
end
  end
end
always @(negedge clk_in) begin //下降沿 3 分频，占空比为 1:2
if(!rst_n) begin
  cnt1 <= 0;
clk_1to3n <= 0;
  end
  else begin
  if(cnt1 == 2'b10) begin
    cnt1 <= 0;
clk_1to3n <= clk_1to3n;
end
else begin
    cnt1 <= cnt1 + 1;
  clk_1to3n <= !clk_1to3n;
end
  end
end
assign clk_out = clk_1to3p | clk_1to3n; //错位相或
endmodule
```