## 《PLD原理与EDA技术》复习提纲

**一、基本概念**

1-1 EDA技术与ASIC设计和FPGA开发有什么关系?

答：利用EDA技术进行电子系统设计的最后目标是完成专用集成电路ASIC的设计和实现；FPGA和CPLD是实现这一途径的主流器件。FPGA和CPLD通常也被称为可编程专用IC，或可编程ASIC。FPGA和CPLD的应用是EDA技术有机融合软硬件电子设计技术、SoC（片上系统）和ASIC设计，以及对自动设计与自动实现最典型的诠释。

1-2与软件描述语言相比，VHDL有什么特点?

答：编译器将软件程序翻译成基于某种特定CPU的机器代码，这种代码仅限于这种CPU而不能移植，并且机器代码不代表硬件结构，更不能改变CPU的硬件结构，只能被动地为其特定的硬件电路结构所利用。综合器将VHDL程序转化的目标是底层的电路结构网表文件，这种满足VHDL设计程序功能描述的电路结构，不依赖于任何特定硬件环境；具有相对独立性。综合器在将VHDL(硬件描述语言)表达的电路功能转化成具体的电路结构网表过程中，具有明显的能动性和创造性，它不是机械的一一对应式的“翻译”，而是根据设计库、工艺库以及预先设置的各类约束条件，选择最优的方式完成电路结构的设计。

l-3什么是综合?有哪些类型?综合在电子设计自动化中的地位是什么?

什么是综合? 答：在电子设计领域中综合的概念可以表示为：将用行为和功能层次表达的电子系统转换为低层次的便于具体实现的模块组合装配的过程。

有哪些类型? 答：(1)从自然语言转换到VHDL语言算法表示，即自然语言综合。(2)从算法表示转换到寄存器传输级(RegisterTransport Level，RTL)，即从行为域到结构域的综合，即行为综合。(3)从RTL级表示转换到逻辑门(包括触发器)的表示，即逻辑综合。(4)从逻辑门表示转换到版图表示(ASIC设计)，或转换到FPGA的配置网表文件，可称为版图综合或结构综合。

综合在电子设计自动化中的地位是什么? 答：是核心地位（见图1-3）。综合器具有更复杂的工作环境，综合器在接受VHDL程序并准备对其综合前，必须获得与最终实现设计电路硬件特征相关的工艺库信息，以及获得优化综合的诸多约束条件信息；根据工艺库和约束条件信息，将VHDL程序转化成电路实现的相关信息。

1-4在EDA技术中，自顶向下的设计方法的重要意义是什么?

答：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。

1-5 IP在EDA技术的应用和发展中的意义是什么?

答：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。

2-1 叙述EDA的FPGA/CPLD设计流程。

答：1.设计输入(原理图/HDL文本编辑)；2.综合；3.适配；4.时序仿真与功能仿真；5.编程下载；6.硬件测试。

2-2 IP是什么?IP与EDA技术的关系是什么?

IP是什么? 答：IP是知识产权核或知识产权模块，用于ASIC或FPGA/CPLD中的预先设计好的电路功能模块。

IP与EDA技术的关系是什么? 答：IP在EDA技术开发中具有十分重要的地位；与EDA技术的关系分有软IP、固IP、硬IP：软IP是用VHDL等硬件描述语言描述的功能块，并不涉及用什么具体电路元件实现这些功能；软IP通常是以硬件描述语言HDL源文件的形式出现。固IP是完成了综合的功能块，具有较大的设计深度，以网表文件的形式提交客户使用。硬IP提供设计的最终阶段产品：掩模。

2-3 叙述ASIC的设计方法。

答：ASIC设计方法,按版图结构及制造方法分有半定制(Semi-custom)和全定制(Full-custom)两种实现方法。

全定制方法是一种基于晶体管级的，手工设计版图的制造方法。

半定制法是一种约束性设计方式，约束的目的是简化设计，缩短设计周期，降低设计成本，提高设计正确率。半定制法按逻辑实现的方式不同，可再分为门阵列法、标准单元法和可编程逻辑器件法。

2-4 FPGA/CPLD在ASIC设计中有什么用途?

答：FPGA/CPLD在ASIC设计中，属于可编程ASIC的逻辑器件；使设计效率大为提高，上市的时间大为缩短。

2-5 简述在基于FPGA/CPLD的EDA设计流程中所涉及的EDA工具，及其在整个流程中的作用。

答：基于FPGA/CPLD的EDA设计流程中所涉及的EDA工具有：**设计输入编辑器（作用：**接受不同的设计输入表达方式，如原理图输入方式、状态图输入方式、波形输入方式以及HDL的文本输入方式。**）；HDL综合器（作用：**HDL综合器根据工艺库和约束条件信息，将设计输入编辑器提供的信息转化为目标器件硬件结构细节的信息，并在数字电路设计技术、化简优化算法以及计算机软件等复杂结体进行优化处理**）；仿真器（作用：**行为模型的表达、电子系统的建模、逻辑电路的验证及门级系统的测试**）；适配器（作用：**完成目标系统在器件上的布局和布线**）；下载器（作用：**把设计结果信息下载到对应的实际器件，实现硬件设计**）。**

3-1 OLMC（输出逻辑宏单元）有何功能?说明GAL是怎样实现可编程组合电路与时序电路的。

OLMC有何功能? 答：OLMC单元设有多种组态，可配置成专用组合输出、专用输入、组合输出双向口、寄存器输出、寄存器输出双向口等。

说明GAL是怎样实现可编程组合电路与时序电路的? 答：GAL（通用阵列逻辑器件）是通过对其中的OLMC（输出逻辑宏单元）的编程和三种模式配置（寄存器模式、复合模式、简单模式），实现组合电路与时序电路设计的。

3-2 什么是基于乘积项的可编程逻辑结构?

答：GAL、CPLD之类都是基于乘积项的可编程结构；即包含有可编程与阵列和固定的或阵列的PAL（可编程阵列逻辑）器件构成。

3-3 什么是基于查找表的可编程逻辑结构?

答：FPGA（现场可编程门阵列）是基于查找表的可编程逻辑结构。

3-4 FPGA系列器件中的LAB有何作用?

答：FPGA（Cyclone/Cyclone II）系列器件主要由逻辑阵列块LAB、嵌入式存储器块（EAB）、I/O单元、嵌入式硬件乘法器和PLL等模块构成；其中LAB（逻辑阵列块）由一系列相邻的LE（逻辑单元）构成的；FPGA可编程资源主要来自逻辑阵列块LAB。

3-5 与传统的测试技术相比，边界扫描技术有何优点?

答：使用BST（边界扫描测试）规范测试，不必使用物理探针，可在器件正常工作时在系统捕获测量的功能数据。克服传统的外探针测试法和“针床”夹具测试法来无法对IC内部节点无法测试的难题。

3-6 解释编程与配置这两个概念。

答：编程：基于电可擦除存储单元的EEPROM或Flash技术。CPLD一股使用此技术进行编程。CPLD被编程后改变了电可擦除存储单元中的信息，掉电后可保存。电可擦除编程工艺的优点是编程后信息不会因掉电而丢失，但编程次数有限，编程的速度不快。

配置：基于SRAM查找表的编程单元。编程信息是保存在SRAM中的，SRAM在掉电后编程信息立即丢失，在下次上电后，还需要重新载入编程信息。大部分FPGA采用该种编程工艺。该类器件的编程一般称为配置。对于SRAM型FPGA来说，配置次数无限，且速度快；在加电时可随时更改逻辑；下载信息的保密性也不如电可擦除的编程。

3-7 请参阅相关资料，并回答问题：按本章给出的归类方式，将基于乘积项的可编程逻辑结构的PLD器件归类为CPLD；将基于查找表的可编程逻辑结构的PLD器什归类为FPGA，那么，APEX系列属于什么类型PLD器件? MAX II系列又属于什么类型的PLD器件?为什么?

答：APEX(Advanced Logic Element Matrix)系列属于FPGA类型PLD器件；编程信息存于SRAM中。MAX II系列属于CPLD类型的PLD器件；编程信息存于EEPROM中。

5-1 归纳利用Quartus II进行VHDL文本输入设计的流程：从文件输入一直到SignalTap II测试。

答：1  建立工作库文件夹和编辑设计文件；2 创建工程；3 编译前设置；4 全程编译；5 时序仿真；6 引脚锁定；7 配置文件下载；8 打开SignalTap II编辑窗口；9 调入SignalTap II的待测信号；10 SignalTap II参数设置；11 SignalTap II参数设置文件存盘；12 带有SignalTap II测试信息的编译下载；13 启动SignalTap II进行采样与分析；14 SignalTap II的其他设置和控制方法。

6-1 什么是固有延时?什么是惯性延时?

答：固有延时(Inertial Delay)也称为惯性延时，**固有延时的主要物理机制是分布电容效应**。

6-2 δ是什么?在VHDL中，δ有什么用处?

δ是什么? 答：在VHDL仿真和综合器中，默认的固有延时量（它在数学上是一个无穷小量），被称为δ延时。

在VHDL中，δ有什么用处?答：在VHDL信号赋值中未给出固有延时情况下，VHDL仿真器和综合器将自动为系统中的信号赋值配置一足够小而又能满足逻辑排序的延时量δ；使并行语句和顺序语句中的并列赋值逻辑得以正确执行。

6-4 说明信号和变量的功能特点，以及应用上的异同点。

答：变量：变量是一个局部量，只能在进程和子程序中使用。变量不能将信息带出对它做出定义的当前结构。变量的赋值是一种理想化的数据传输，是立即发生的，不存在任何延时行为。变量的主要作用是在进程中作为临时的数据存储单元。

信号：信号是描述硬件系统的基本数据对象，其性质类似于连接线；可作为设计实体中并行语句模块间的信息交流通道。信号不但可以容纳当前值，也可以保持历史值；与触发器的记忆功能有很好的对应关系。

6-5 在VHDL设计中，给时序电路清零(复位)有两种力方法，它们是什么?

解：设Q定义成信号，一种方法：Q<=“000…000”； 其中“000…000”反映出信号Q的位宽度。第二种方法：Q<=(OTHERS=>‘0’)；其中OTHERS=>‘0’不需要给出信号Q的位宽度，即可对Q清零。

6-7 什么是重载函数?重载算符有何用处?如何调用重载算符函数?

答：（1）什么是重载函数? 根据操作对象变换处理功能。

（2）重载算符有何用处? 用于两个不同类型的操作数据自动转换成同种数据类型，并进行运算处理。

（3）如何调用重载算符函数?采用隐式方式调用，无需事先声明。

7-2 LPM\_ROM、LPM\_RAM、LPM\_FIFO等模块与FPGA中嵌入的EAB、ESB、M4K有怎样的联系?

答：ACEXlK系列为EAB；APEX20K系列为ESB；Cyclone系列为M4K

二、 **EDA名词解释**

1．CPLD：复杂可编程逻辑器件

2．HDL：硬件描述语言

3．LUT：查找表

4．ASIC：专用集成电路

5．SOC：单片电子系统（或者片上系统）

6．VHDL：超高速集成电路硬件描述语言

7．FPGA：现场可编程门阵列

8．RTL：寄存器传输级

9．JTAG：联合测试活动组织

10．EAB：嵌入式阵列块

11：SOPC：片上可编程系统

12．ISP：在系统可编程技术

13．ICR：在线可重配置技术

14．IP：知识产权核

15．IEEE：电子电气工程师协会

16．LPM：参数可定制宏模块库

17．EDA：电子设计自动化

18．FSM：有限状态机

**三、简答题：**

1．简述VHDL程序的基本结构？

答：VHDL程序是由以下几个部分组成的：

（1）LIBRARY （库）部分

（2）PACKAGE （程序包）部分

（3）ENTITY （实体）部分

（4）ARCHITECTURE （结构体）部分

（5）CONFIGURATION （配置）部分

2．CPLD和FPGA有什么差异？在实际应用中各有什么特点？

答：差异：（1）CPLD：复杂可编程逻辑器件，FPGA：现场可编程门阵列；（2）CPLD：基于乘积项技术的确定型结构，FPGA：基于查找表技术的统计型结构;（3）CPLD：5500 ～ 50000门，FPGA：1K ～10M 门 。

实际应用中各自的特点：CPLD适用于逻辑密集型中小规模电路，编程数据不丢失，延迟固定，时序稳定； FPGA适用于数据密集型大规模电路，需用专用的 ROM 进行数据配置，布线灵活，但时序特性不稳定。

3． 设计项目的验证有哪几种方法？它们具体含义？

答：包括功能仿真、时序仿真和定时分析。

功能仿真又称前仿真，是在不考虑器件延时的理想情况下的一种项目验证方法，通过功能仿真来验证一个项目的逻辑功能是否正确。

时序仿真又称模拟仿真或后仿真，是在考虑设计项目具体适配器件的各种延时的情况下的一种项目验证方法。时序仿真不仅测试逻辑功能，还测试目标器件最差情况下的时间关系。

定时分析可以分析各个信号到输出端的时间延迟,可以给出延迟矩阵和最高工作频率，还可分析信号的建立、保持时间。

4．结构体的描述方式有几种，各有什么特点？

答：结构体的描述方式有: Structure（结构）描述、 Date Flow（数据流）描述 、Behavior Process（行为）描述 。

Structure描述——描述该设计单元的硬件结构。

Date Flow描述——它是类似于寄存器传输级的方式,描述数据的传输和变换。

Behavior Process描述——只描述电路的功能或者电路行为（输入输出间转换的行为）。

5．解释编程与配置这两个概念？

答：编程：基于电可擦除存储单元的EEPROM或Flash技术。CPLD一股使用此技术进行编程。CPLD被编程后改变了电可擦除存储单元中的信息，掉电后可保存。电可擦除编程工艺的优点是编程后信息不会因掉电而丢失，但编程次数有限，编程的速度不快。

配置：基于SRAM查找表的编程单元。编程信息是保存在SRAM中的，SRAM在掉电后编程信息立即丢失，在下次上电后，还需要重新载入编程信息。大部分FPGA采用该种编程工艺。该类器件的编程一般称为配置。对于SRAM型FPGA来说，配置次数无限，且速度快；在加电时可随时更改逻辑；下载信息的保密性也不如电可擦除的编程。

6．简述层次结构设计的优点？答：层次化设计是一种模块化的设计方法，设计人员对设计的描述由上至下逐步展开，符合常规的思维习惯；由于顶层设计与具体的器件和工艺无关，因此易于在各种可编程逻辑器件中间进行移植。层次化的设计方法可以使多个设计人员同时进行操作。有利于对设计任务进行合理的分配并用系统工程的方法对设计进行管理。

7． 简述 CPLD的结构？

答：CPLD的基本结构由可编程逻辑阵列（LAB）、可编程I/O控制模块和可编程内部连线（PIA）等三部分组成。

1）可编程逻辑阵列（LAB）

可编程逻辑阵列又若干个可编程逻辑宏单元（Logic Macro Cell，LMC）组成， LMC内部主要包括与阵列、或阵列、可编程触发器和多路选择器等电路，能独立地配置为时序或组合工作方式。

2）可编程I/O单元（IOC）

CPLD的I/O单元（Input/Output Cell，IOC），是内部信号到I/O引脚的接口部分。根据器件和功能的不同，各种器件的结构也不相同。由于阵列型器件通常只有少数几个专用输入端，大部分端口均为I/O端，而且系统的输入信号通常需要锁存。因此I/O常作为一个独立单元来处理。

3）可编程内部连线（PIA）

可编程内部连线的作用是在各逻辑宏单元之间以及逻辑宏单元和I/O单元之间提供互连网络。各逻辑宏单元通过可编程连线阵列接收来自输入端的信号，并将宏单元的信号送目的地。这种互连机制有很大的灵活性，它允许在不影响引脚分配的情况下改变内部的设计。

8． 简述 FPGA的结构？

答：FPGA由可编程逻辑块（CLB）、输入/输出模块（IOB）及可编程互连资源（PIR）等三种可编程电路和一个SRAM结构的配置存储单元组成。

1）可编程逻辑块（CLB）

CLB主要由逻辑函数发生器、触发器、数据选择器等电路组成。

2）输入/输出模块（IOB）

IOB主要由输入触发器、输入缓冲器和输出触发/锁存器、输出缓冲器组成，每个IOB控制一个引脚，它们可被配置为输入、输出或双向I/O功能。

3）可编程互连资源（PIR）

PIR由许多金属线段构成，这些金属线段带有可编程开关，通过自动布线实现各种电路的连接。实现FPGA内部的CLB和CLB之间、CLB和IOB之间的连接。

9． 什么是边界扫描技术?其原理是什么? 与传统的测试技术相比，边界扫描技术有何优点?

答：边界扫描测试技术（Boundary Scan Testing，BST），主要用于解决可编程逻辑器件芯片的测试问题。

这种测试可在器件正常工作时捕获功能数据。器件的边界扫描单元能够迫使逻辑追踪引脚信号，或是从引脚或器件核心逻辑信号中捕获数据。强行加入的测试数据串行地移入边界扫描单元，捕获的数据串行移出并在器件外部同预期的结果进行比较。

标准的边界扫描测试只需要五根信号线，即TDI（测试数据输入）、TDO（测试数据输出）、TRST（测试复位输入）TMS（测试模式选择）和TCK（测试时钟输入），TRST能够对电路板上所有支持边界扫描的芯片内部逻辑和边界管脚进行测试。应用边界扫描技术能够增强芯片、电路板甚至系统的可测试性。

使用BST（边界扫描测试）规范测试，不必使用物理探针，可在器件正常工作时在系统捕获测量的功能数据。克服传统的外探针测试法和“针床”夹具测试法来无法对IC内部节点无法测试的难题。

**四、判断题**

1.IEEE库使用时必须声明。 （ **√** ）

2.实体（ENTITY）不是VHDL程序所必须的。 （**×** ）

3.一个实体只能有一个结构体。 （ **×** ）

4.结构体内部定义的数据类型、常数、函数、过程只能用于该结构体。（**√** ）

5．VHDL语言运算符没有优先级。（ **×** ）

6．CPLD被编程信息，掉电后信息仍保存。 (**√** )

7．使用BST（边界扫描测试）规范测试，不必使用物理探针。 (**√** )

8．FPGA被编程信息，掉电后信息仍保存。 (**×** )

9．CPLD编程信息保存在SRAM中。 (**×** )

10．FPGA可编程资源主要来自逻辑阵列块LAB。 (**√** )

11．“信号”具有延迟、事件等特性，而变量则没有。 （ √ ）

12．记录类型中可以包含“存取型”和“文件型”的数据对象。 （ × ）

13．“+”和“-”运算符只能用于整形数运算，移位操作符则只能用于BIT型和BOOLEAN型的运算。 （ √ ）

14．目前，在可综合的VHDL程序中，乘方运算符（\*\*）的右操作数可以是任意的整数。 （ × ）

15．“=”和“/=”运算比“>”和”<”综合生成的电路规模要小。( √ )

16．数100在VHDL语言中既可以表示整数又可以表示实数。（**×**）

17．在进程中任意交换语句的顺序，其执行结果不变。（**×**）

18．WAIT FOR 语句后面要求接敏感信号量。（**×**）

19．在IF语句的条件表达式中只能使用关系运算操作及逻辑运算操作的组合表达式（**√**）

20．在使用进程时，敏感信号量必须跟在PROCESS（）的括号中。（**√**）

**五、VHDL程序分析处理**

4-1 画出与以下实体描述对应的原理图符号元件:

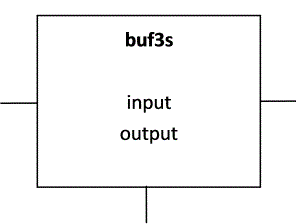
1） ENTITY buf3s IS --实体1:三态缓冲器

PORT(input:IN STD\_LOGIC; --输入端

enable:IN STD\_LOGIC; --使能端

output:OUT STD\_LOGIC); --输出端

END buf3s ;

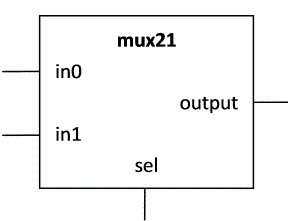


2） ENTITY mux21 IS --实体2: 2选1多路选择器

PORT(in0, in1,sel: IN STD\_LOGIC;

output:OUT STD\_LOGIC);

END mux21；



6-6 哪一种复位方法必须将复位信号放在敏感信号表中?给出这两种电路的VHDL描述。

解：边沿触发复位信号要将复位信号放在进程的敏感信号表中。

（1）边沿触发复位信号

……………………. ARCHITECTURE bhv 0F DFF3 IS SIGNAL QQ:STD\_LOGIC;

BEGIN

PROCESS(RST)

BEGIN

IF RST’EVENT AND RST=‘1' THEN

QQ<=(Others=>‘0’);

END IF;

END PROCESS;

Q1<=QQ;

END;

………………………

（2）电平触发复位信号

……………………. ARCHITECTURE bhv 0F DFF3 IS SIGNAL QQ:STD\_LOGIC;

BEGIN

PROCESS(CLK)

BEGIN

IF RST=‘1' THEN

QQ<=(Others=>‘0’);

END IF;

END PROCESS;

Q1<=QQ;

END;

………………………

6-8 判断下面三个程序中是否有错误，若有则指出错误所在，并给出完整程序。

程序1:

Signal A,EN : std\_logic;

…………………

Process(A, EN)

Variable B: std\_log ic;

Begin

if EN=l then B<=A; end if; --将“B<=A”改成“B:=A”

end process;

程序2:

Architecture one of sample is

variable a，b，c:integer;

begin

c<=a+b; --将“c<=a+b”改成“c:=a+b”

end;

程序3:

library ieee;

use ieee.std\_logic\_1164.all;

entity mux21 is

PORT(a,b:in std\_logic; sel:in std\_loglc;c:out std\_logle;); --将“;)”改成“)”

end sam2; --将“sam2”改成“entity mux21”

architecture one of mux2l is begin

--增加“process(a,b,sel) begin”

if sel= '0' then c:=a; else c:=b; end if; --应改成“if sel= '0' then c<=a; else c<=b; end if;”

--增加“end process;”

end two; --将“two”改成“architecture one”

**六、阅读下列VHDL程序，画出结构体的原理图（RTL级）**

例1：课本习题3.7

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY DLATCH IS

PORT (D,CP :IN STD\_LOGIC;

Q,QN :BUFFER STD\_LOGIC) ;

END DLATCH ;

ARCHITECTURE one OF DLATCH IS

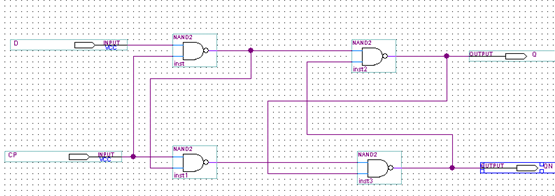
SIGNAL N1,N2 :STD\_LOGIC;

BEGIN

N1<=D NAND CP;

N2<=N1 NAND CP; Q<=QN NAND N1;

QN<=Q NAND N2;END one;



例2．

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY TRIS IS

PORT ( CONTROL : IN STD\_LOGIC; INN : IN STD\_LOGIC; Q : INOUT STD\_LOGIC; Y : OUT STD\_LOGIC );END TRIS;

ARCHITECTURE ONE OF TRIS IS

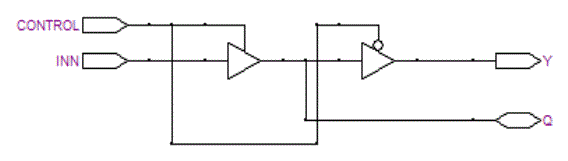
BEGIN

PROCESS (CONTROL, INN, Q)

BEGIN

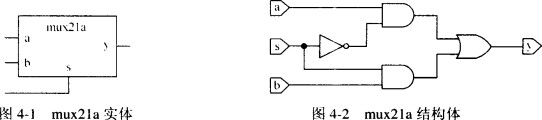
IF (CONTROL = '0') THEN Y <= Q; Q <= 'Z'; ELSE Q <= INN; Y <= 'Z'; END IF; END PROCESS;

END ONE;



**七、应用设计**

1. 2选1多路选择器



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY mux21a IS PORT(a,b,s:IN BIT;

y:OUT BIT);

END ENTITY mux21a; ARCHITECTURE one 0F mux21a IS BEGIN

PROCESS(a,b,s)

BEGIN

IF s=‘0’ THEN y<=a;

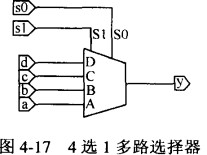
ELSE

y<=b;

ENDIF;

END PROCESS;

2. 图4-17所示的是4选1多路选择器,试分别用IF\_THEN语句和CASE语句的表达方式写出此电路的VHDL程序,选择控制信号s1和s0的数据类型为STD\_LOGIC\_VECTOR;当s1=’0’,s0=’0’；s1=’0’,s0=’1’；s1=’1’,s0=’0’和s1=’1’,s0=’1’时,分别执行y<=a、y<=b、y<=c、y<=d。



--解1：用IF\_THEN语句实现4选1多路选择器

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY mux41 IS

PORT (a,b,c,d: IN STD\_LOGIC;

s0: IN STD\_LOGIC;

s1: IN STD\_LOGIC;

y: OUT STD\_LOGIC);

END ENTITY mux41;

ARCHITECTURE if\_mux41 OF mux41 IS

SIGNAL s0s1 : STD\_LOGIC\_VECTOR(1 DOWNTO 0);--定义标准逻辑位矢量数据

BEGIN

s0s1<=s1&s0; --s1相并s0,即s1与s0并置操作

PROCESS(s0s1,a,b,c,d)

BEGIN

IF s0s1 = "00" THEN y <= a;

ELSIF s0s1 = "01" THEN y <= b;

ELSIF s0s1 = "10" THEN y <= c;

ELSE y <= d;

END IF;

END PROCESS;

END ARCHITECTURE if\_mux41;

--解2：用CASE语句实现4选1多路选择器

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY mux41 IS

PORT (a,b,c,d: IN STD\_LOGIC;

s0: IN STD\_LOGIC;

s1: IN STD\_LOGIC;

y: OUT STD\_LOGIC);

END ENTITY mux41;

ARCHITECTURE case\_mux41 OF mux41 IS

SIGNAL s0s1 : STD\_LOGIC\_VECTOR(1 DOWNTO 0);--定义标准逻辑位矢量数据类型

BEGIN

s0s1<=s1&s0; --s1相并s0,即s1与s0并置操作

PROCESS(s0s1,a,b,c,d)

BEGIN

CASE s0s1 IS --类似于真值表的case语句

WHEN "00" => y <= a;

WHEN "01" => y <= b;

WHEN "10" => y <= c;

WHEN "11" => y <= d;

WHEN OTHERS =>NULL ;

END CASE;

END PROCESS;

END ARCHITECTURE case\_mux41;

3.编写一个8线－3线优先编码器的VHDL程序

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY coder IS

PORT(a:IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

y:OUT STD\_LOGIC\_VECTOR(2 DOWNTO 0));

END coder;

ARCHITECTURE one OF coder IS

BEGIN

PROCESS(a)

BEGIN

IF a(7)='0' THEN y<="111"; ELSIF (a(6)='0') THEN y<="110"; ELSIF (a(5)='0') THEN y<="101"; ELSIF (a(4)='0') THEN y<="100"; ELSIF (a(3)='0') THEN y<="011"; ELSIF (a(2)='0') THEN y<="010"; ELSIF (a(1)='0') THEN y<="001"; ELSE y<="000";

END IF;

END PROCESS;END one;

4.用CASE语句和IF语句编写3线-8线译码器VHDL程序.(课本习题3.3)

(1)用CASE语句

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

entity decoder38 is

port(a,b,c,g1,g2a,g2b: in std\_logic; y: out std\_logic\_vector(7 downto 0));

end decoder38;

architecture behave38 OF decoder38 is

signal indata: std\_logic\_vector(2 downto 0);

begin

indata<=c&b&a;

process(indata,g1,g2a,g2b) begin

if(g1='1' and g2a='0' and g2b='0') then case indata is

when "000"=>y<="11111110";

when "001"=>y<="11111101";

when "010"=>y<="11111011";

when "011"=>y<="11110111";

when "100"=>y<="11101111";

when "101"=>y<="11011111";

when "110"=>y<="10111111";

when "111"=>y<="01111111";

when others=>y<="XXXXXXXX";

end case;

else

y<="11111111";

end if;

end process;

end behave38;

（2用IF语句

library IEEE;

use IEEE.std\_logic\_1164.all;

entity decoder38 is

port ( a,b,c,g1,g2a,g2b: in std\_logic; y: out STD\_LOGIC\_VECTOR (7 downto 0));

end decoder38;

architecture decoder\_if of decoder38 is

signal indata: std\_logic\_vector(2 downto 0);

begin

indata<=c&b&a;

process(indata,g1,g2a,g2b) begin

if (g1='1' and g2a='0' and g2b='0') then if indata="000" then y<=(0=>'0',others=>'1');

elsif indata="001" then y<=(1=>'0',others=>'1');

elsif indata="010" then y<=(2=>'0',others=>'1');

elsif indata="011" then y<=(3=>'0',others=>'1');

elsif indata="100" then y<=(4=>'0',others=>'1');

elsif indata="101" then y<=(5=>'0',others=>'1');

elsif indata="110" then y<=(6=>'0',others=>'1');

elsif indata="111" then y<=(7=>'0',others=>'1');

else y<=(others=>'X');

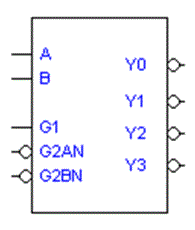
end if;

end if;

end process;

**（练习：仿照此例自己设计2线-4线的译码器VHDL程序）**

参考VHDL程序：

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY decoder2\_4 IS

PORT(

A, B,G1,G2A,G2B: IN STD\_LOGIC; Y: OUT STD\_LOGIC\_VECTOR(3 DOWNTO 0));END decoder2\_4;

ARCHITECTURE one OF decoder2\_4 IS

SIGNAL indata: STD\_LOGIC\_VECTOR(1 DOWNTO 0);

BEGIN

indata <= B&A; PROCESS (indata, G1, G2A,G2B) BEGIN

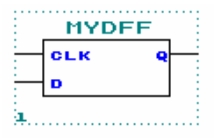
IF (G1='1' AND G2A='0' AND G2B='0') THEN CASE indata IS WHEN "00"=>Y<="1110";

WHEN "01"=>Y<="1101"; WHEN "10"=>Y<="1011"; WHEN "11"=>Y<="0111";

WHEN OTHERS =>Y<="XXXX"; END CASE; ELSE Y<="1111"; END IF; END PROCESS;

END one;

5.试用VHDL描述一个外部特性如图所示的D触发器。



参考程序如下：

LIBRARY IEEE;

USE IEEE.STD\_LOG\_1164.ALL;

ENTITY mydff IS

PORT(CLK:IN STD\_LOG;

D:IN STD\_LOGIC;

Q:OUT STD\_LOGIC);

END; ARCHITECTURE bhv OF mydff IS

BEGIN

PROCESS(CLK)

BEGIN

IF CLK'EVENT AND CLK='1' THEN

Q<=D;

END IF;

END PROCESS;

END;

6.设计一个异步清除十进制加法计数器的VHDL程序

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY cnt10y IS

PORT(clr:IN STD\_LOGIC;

clk:IN STD\_LOGIC;

cnt:BUFFER INTEGER RANGE 9 DOWNTO 0);

END cnt10y;

ARCHITECTURE one OF cnt10y IS

BEGIN

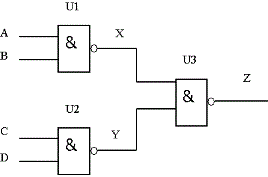
PROCESS(clr,clk)

BEGIN

IF clr='0' THEN cnt<=0; ELSIF clk'EVENT AND clk='1' THEN IF (cnt=9) THEN cnt<=0; ELSE cnt<=cnt+1; END IF; END IF; END PROCESS;END one;

7.用VHDL语言的元件例化语句设计如图所示电路。元件为2输入与非门。

提示：(1)先设计2输入的与非门电路；(2)再用元件例化语句设计4输入的与非门电路。



解答：(1)先设计2输入的与非门电路

LIBRARY ieee;USE ieee.std\_logic\_1164.ALL;ENTITY nand\_2 IS

PORT( A1,B1: IN std\_logic; C1: OUT std\_logic);END nand\_2;

ARCHITECTURE one OF nand\_2 IS

Begin

C1<=not (A1 and B1);

End one;

(2)再用元件例化语句设计4输入的与非门电路

LIBRARY ieee;USE ieee.std\_logic\_1164.ALL;ENTITY nand\_4 IS

PORT( A,B,C,D: IN std\_logic; Z: OUT std\_logic);END nand\_4;

ARCHITECTURE a OF nand\_4 IS COMPONENT nand\_2

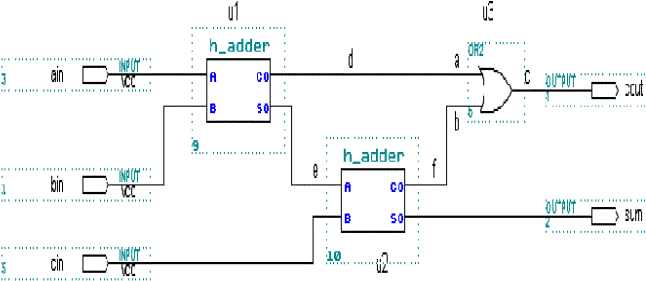
PORT (A1,B1: IN std\_logic; C1: OUT std\_logic); END COMPONENT;

SIGNAL X,Y: std\_logic ;

BEGIN U1:nand\_2 PORT MAP (A,B,X);

U2:nand\_2 PORT MAP (C,D,Y); U3:nand\_2 PORT MAP (A1 => X, B1 => Y, C1=>Z);END a;

8.用VHDL语言的元件例化语句设计1位全加器，如图所示。



解答： （1）先设计半加器和或门元件

1）半加器程序h\_adder.vhd：

library ieee;

use ieee.std\_logic\_1164.all;

entity h\_adder is

port(a,b:in std\_logic;

co,so:out std\_logic);

end h\_adder;

architecture one of h\_adder is

begin

so<=not(a xor(not b));

co<=a and b;

end one;

2）或门程序or2a.vhd：

library ieee;

use ieee.std\_logic\_1164.all;

entity or2a is port(a,b:in std\_logic;

c:out std\_logic);

end or2a;architecture one of or2a isbegin

c<=a or b;

end one;

（2）再用元件例化语句设计一位全加器程序f\_adder.vhd

library ieee;

use ieee.std\_logic\_1164.all;

entity f\_adder is

port(ain,bin,cin:in std\_logic;

cout,sum:out std\_logic);

end entity f\_adder;

architecture one of f\_adder is

component h\_adder

port(a,b:in std\_logic;

co,so:out std\_logic);

end component;

component or2a port(a,b:in std\_logic;

c:out std\_logic);

end component;

signal d,e,f:std\_logic;

begin

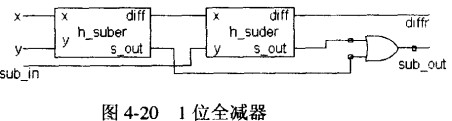
u1 : h\_adder port map(a=>ain,b=>bin,co=>d,so=>e);

u2 : h\_adder port map(a=>e,b=>cin,co=>f,so=>sum);

u3 : or2a port map(d,f,cout);end one;

9. 给出1位全减器的VHDL描述；最终实现8位全减器。要求:

1)首先设计1位半减器,然后用例化语句将它们连接起来,图4-20中h\_suber是半减器,diff是输出差(diff=x-y),s\_out是借位输出(s\_out=1,x<y),sub\_in是借位输入。



--解(1.1)：实现1位半减器h\_suber(diff=x-y；s\_out=1,x<y)

LIBRARY IEEE; --半减器描述(1):布尔方程描述方法

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY h\_suber IS

PORT( x,y: IN STD\_LOGIC;

diff,s\_out: OUT STD\_LOGIC);

END ENTITY h\_suber;

ARCHITECTURE hs1 OF h\_suber IS

BEGIN

Diff <= x XOR (NOT y);

s\_out <= (NOT x) AND y;

END ARCHITECTURE hs1;

--解(1.2)：采用例化实现图4-20的1位全减器

LIBRARY IEEE; --1位二进制全减器顺层设计描述

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY f\_suber IS

PORT(xin,yin,sub\_in: IN STD\_LOGIC;

sub\_out,diff\_out: OUT STD\_LOGIC);

END ENTITY f\_suber;

ARCHITECTURE fs1 OF f\_suber IS

COMPONENT h\_suber --调用半减器声明语句

PORT(x, y: IN STD\_LOGIC;

diff,s\_out: OUT STD\_LOGIC);

END COMPONENT;

SIGNAL a,b,c: STD\_LOGIC; --定义1个信号作为内部的连接线。

BEGIN

u1: h\_suber PORT MAP(x=>xin,y=>yin,diff=>a,s\_out=>b);

u2: h\_suber PORT MAP(x=>a,y=>sub\_in,diff=>diff\_out,s\_out=>c);

sub\_out <= c OR b;

END ARCHITECTURE fs1;

**八、考试题型说明：**

1、填空题（20分）

2、选择题（10分）

3、EDA名词解释（10分）

4、简答题（10分）

5、判断题（5分）

6、VHDL程序分析和处理（10分）

7、VHDL程序填空题（10分）

8、阅读下列VHDL程序，画出结构体的原理图（RTL级）（10分）

9、VHDL应用设计（15分）

**九、复习参考书：**

1、《EDA技术与与应用》（第3版） （江国强主编，电子工业出版社）

2、《EDA技术实用教程》（第3版） （潘松主编 科学出版社）