



数字系统课程设计

东南大学信息科学与工程学院

2023年

课程任务

- ◆对数字系统设计方法有明确的认识
- ◆熟练掌握VHDL/Verilog的基本语法和使用
- ◆熟练使用ISE/vivado软件
- ◆在vivado软件环境下使用VHDL/verilog设计一个数字系统并通过下载到目标实验板并验证其功能



几个专用名词的介绍

◆ EDA (Electronic Design Automation)

电子设计自动化.指利用计算机及相关软件完成电子系统的设计.具体的讲,是以计算机为工具,代替设计人员完成电子系统的逻辑综合、布局布线和设计仿真等工作.设计人员只需要完成对系统功能的描述,就可以由计算机软件进行处理,得到设计结果,极大地提高了设计效率.当今的EDA技术更多的是指芯片内的电子系统设计自动化,即片上系统(SOC, System On Chip)设计。



几个专用名词的介绍

◆ PLD (Programmable Logic Device)

可编程逻辑器件,是一种半导体集成器件的半成品.在PLD的芯片中按一定方式(阵列形式或单元阵列形式)制作了大量的门、触发器等基本逻辑器件,如对这些基本器件适当的连接(此连接的过程为编程或配置),就可完成某个电路或系统的功能。分为SPLD、CPLD和FPGA的合称,现指CPLD或FPGA.



几个专用名词的介绍

- ◆CPLD (Complex Programmable Logic Device)复杂可编程逻辑器件.CPLD多基于乘积项(Product-Term)结构.采用E²PROM或Flash工艺,断电后信息不丢失.多用于1万门以下的小规模设计,适合做复杂的组合逻辑.



几个专用名词的介绍

◆FPGA (Field Programmable Gate Array)

现场可编程门阵列.FPGA多基于查找表(Look-Up Table)结构,采用SRAM工艺,密度高,触发器多,多用于10,000门以上的大规模设计,适合做复杂的时序逻辑,如数字信号处理和各种算法.FGPA能完成任何数字器件的功能,上至高性能CPU,下至简单的74电路,都可以用FPGA来实现.FPGA已经成为高性能数字系统的首选方案.



几个专用名词的介绍

◆ HDL (Hardware Description Language)

硬件描述语言.是电子系统硬件行为描述、结构描述、数据流描述的语言.是EDA的重要组成部分.流行的HDL包括VHDL和Verilog-HDL,都是IEEE的HDL标准



可编程逻辑器件的发展

◆大致的演变过程：

- 20世纪70年代，熔丝编程的**PROM**和**PLA**器件是最早的可编程逻辑器件。
- 20世纪70年代末，对**PLA**进行了改进，推出了**PAL**器件。
- 20世纪80年代初，**Lattice**公司发明电可擦写的、比**PAL**使用更灵活的**GAL**器件。
- 20世纪80年代中期，**Xilinx**公司提出现场可编程概念，同时生产出**FPGA**器件。
- 20世纪80年代末，**Lattice**公司又提出在系统可编程技术，并且推出了一系列具备在系统可编程能力的**CPLD**器件。
- 进入20世纪90年代后，可编程逻辑集成电路技术进入飞速发展时期。出现了内嵌复杂功能模块的**SOPC**。



可编程逻辑器件的分类

◆ 按集成程度来分为两大类器件：

■ 芯片集成度较低的（SPLD）：

- ◆ PROM（可编程只读存储器）、PLA（可编程逻辑阵列）、PAL（可编程阵列逻辑）、GAL（通用阵列逻辑），可用的逻辑门数大约在500门以下

■ 芯片集成度较高的（CPLD）：

- ◆ CPLD、FPGA

◆ 按结构上来分类：

- 乘积项结构器件。其基本结构为“与-或阵列”，大部分的SPLD和CPLD
- 查找表结构器件。由简单的查找表组成可编程门，再构成阵列形式，大部分的FPGA

◆ 按编程工艺划分：

- 熔丝型器件，早期的PROM器件
- 反熔丝型器件，某些FPGA器件

以上两类都只能编程一次，合称为一次性可编程（One Time Programming）器件

- EPROM型，紫外线擦除电可编程逻辑器件，其用较高的编程电压进行编程，当需要再次编程时，用紫外线进行擦除
- EEPROM型，电可擦除编程器件。现有的大部分CPLD及GAL器件
- SRAM型，即SRAM查找表结构的器件。大部分的FPGA器件
- FLASH型，解决了反熔丝结构的可编程逻辑器件只能一次性编程的不足之处，可实现多次编程，同时做到掉电后不需要重新配置



EDA技术发展进程

- ◆ 1、CAD阶段（20世纪70年代—80年代中期）
 - 这一阶段分别研制了一些单独的软件工具，主要有PCB布线设计、电路模拟、逻辑模拟及版图的绘制等。
- ◆ 2、CAE阶段（20世纪80年代中期—90年代初期）
 - 这一阶段在集成电路与电子系统设计方法学以及设计工具集成化方面取得了许多成果。
- ◆ 3、EDA阶段（20世纪90年代以来）
 - 主要出现了高级语言描述、系统仿真和综合技术为特征的第三代EDA技术。



CPLD简介

◆ CPLD多基于乘积项Product-Term或Look-Up-Table结构。采用EEPROM或FLASH工艺，断电后信息不丢失。多用于1万门以下的小规模设计，适合做复杂的组合逻辑。

◆ 采用CPLD结构的PLD芯片有：

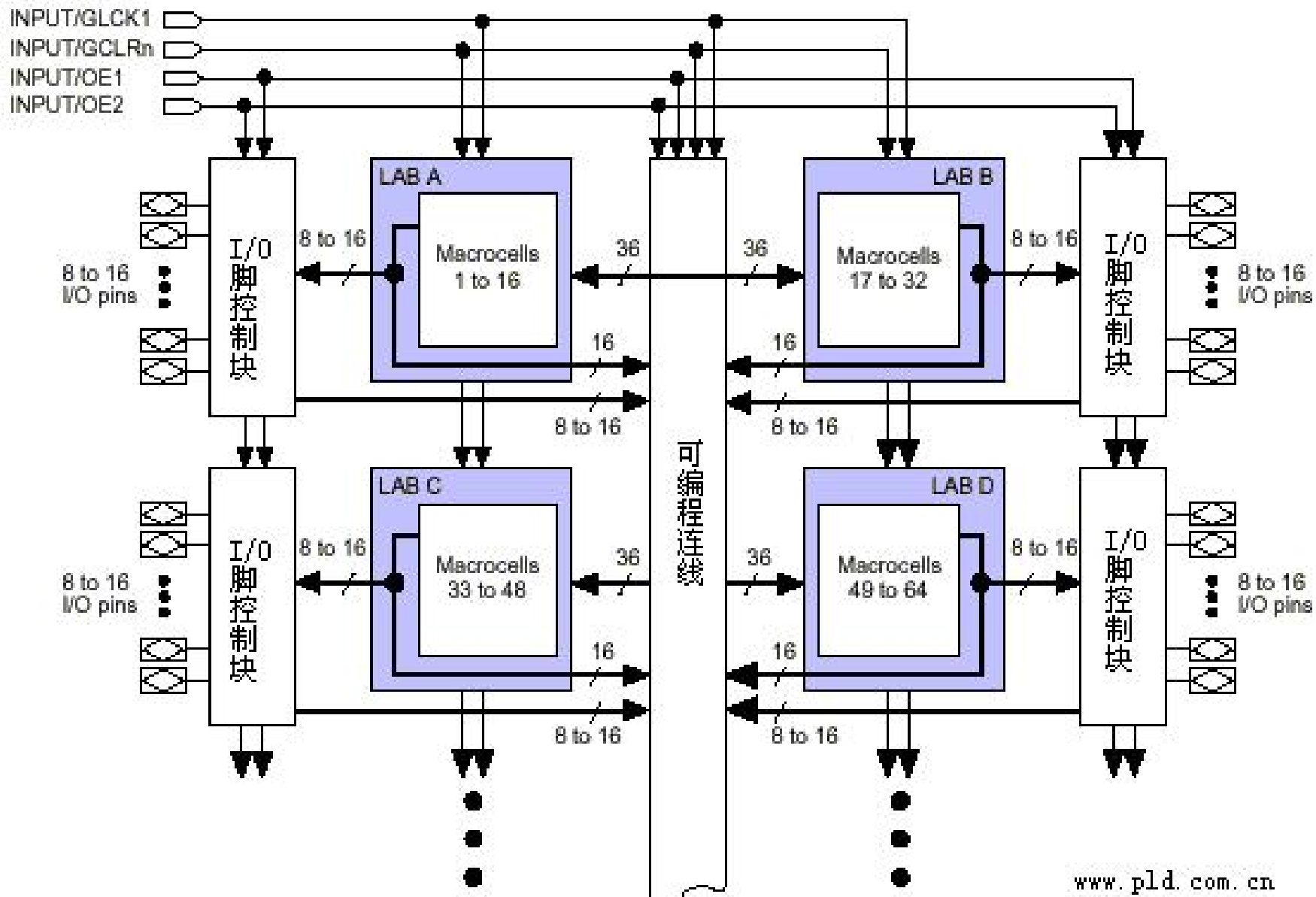
➢ ALTERA MAX CPLD系列简介

	MAX 7000S	MAX 3000A	MAX II	MAX IIZ	MAX V
推出年份	1995	2002	2004	2007	2010
工艺技术	0.5 μm	0.30 μm	0.18 μm	0.18 μm	0.18 μm
关键特性	5.0-V I/O	低成本	I/O数量	零功耗	低成本，低功耗

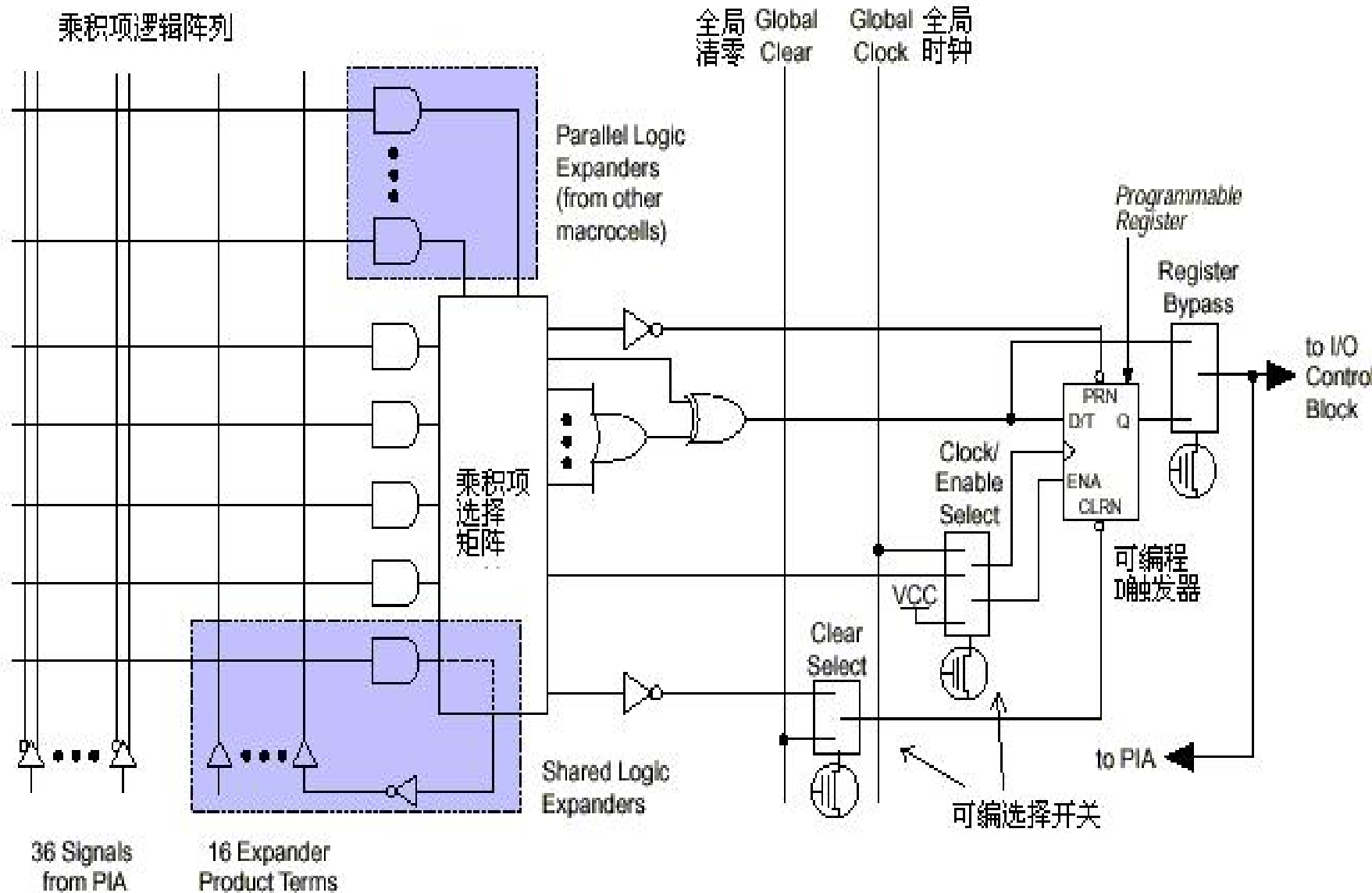
➢ XILINX公司的XC9500系列（FLASH工艺）



基于乘积项的CPLD内部结构（MAX7000系列）

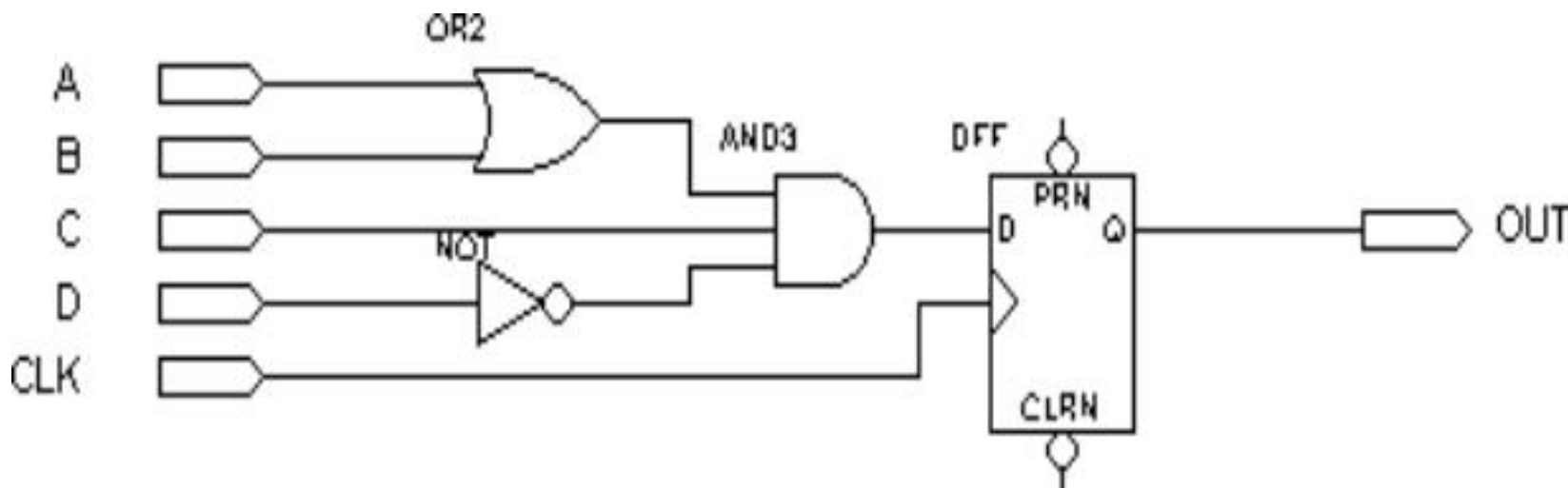


CPLD宏单元结构



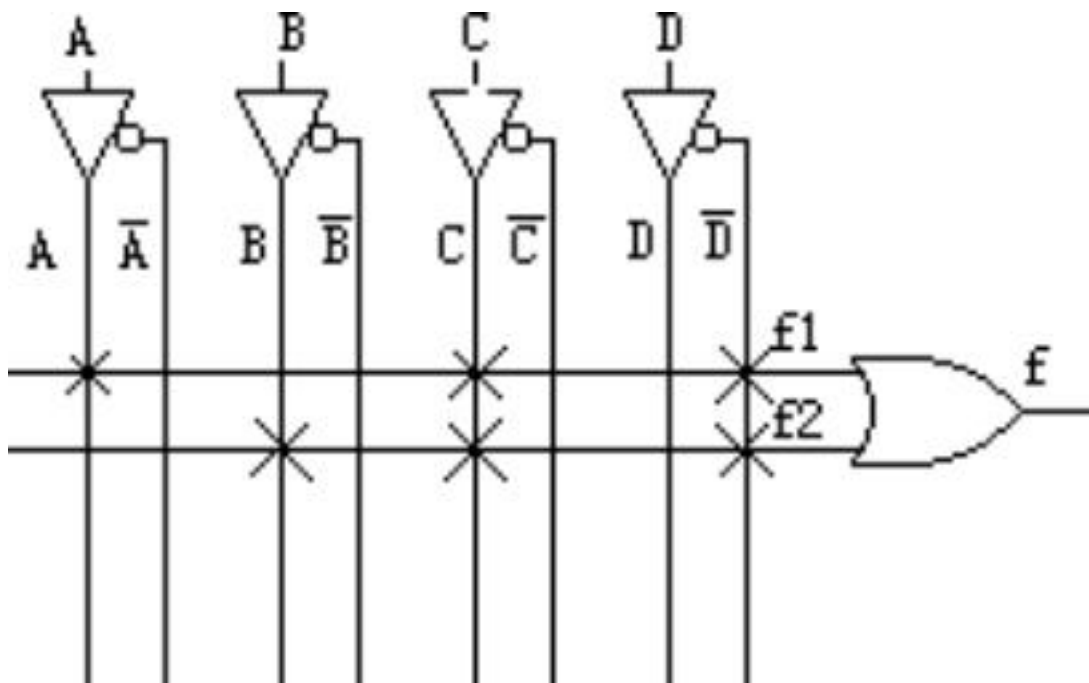
乘积项结构CPLD逻辑实现原理

◆假设组合逻辑的输出(AND3的输出)为f, 则 $f=(A+B)*C*(!D)=A*C*!D + B*C*!D$ (我们以!D表示D的“非”)



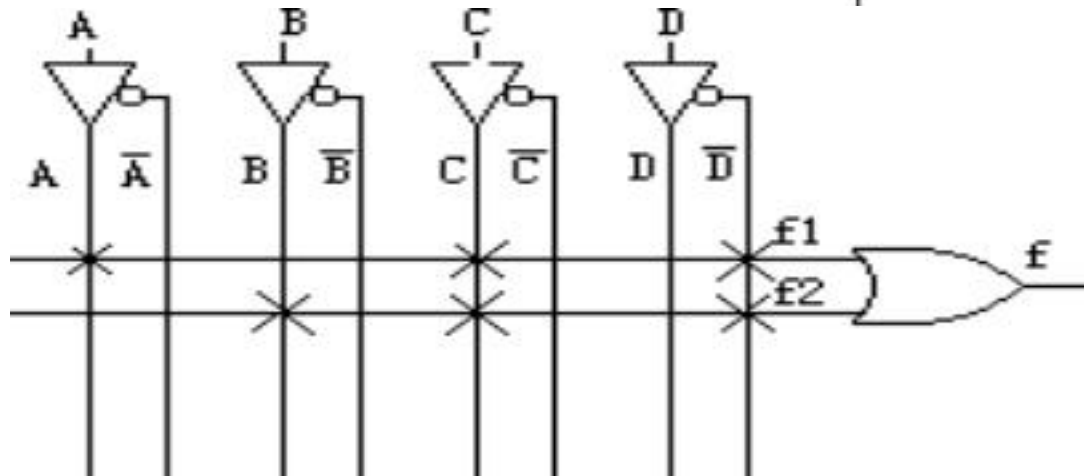
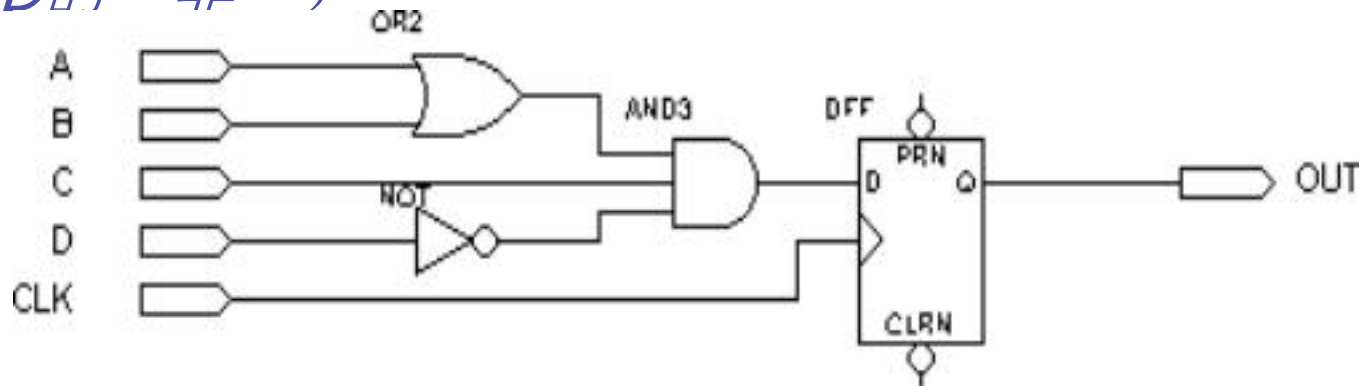
乘积项结构CPLD逻辑实现原理

◆CPLD将以下面的方式来实现组合逻辑f。
（图中每一个叉表示相连及可编程熔丝导通）



CPLD逻辑实现-乘积项

- ◆ 假设组合逻辑的输出(AND3的输出)为 f , 则 $f=(A+B)*C*(!D)=A*C*!D + B*C*!D$ (我们以 $!D$ 表示 D 的“非”)

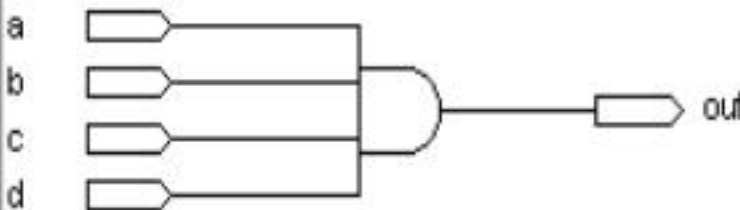
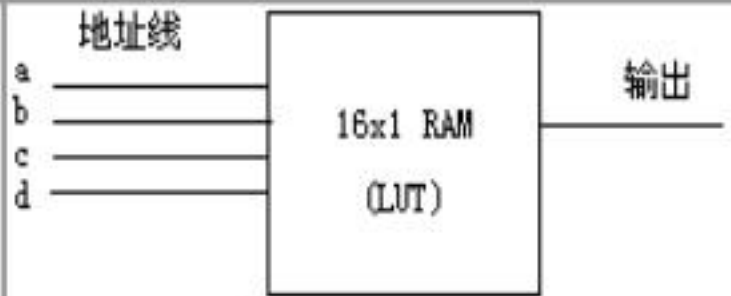


查找表的原理

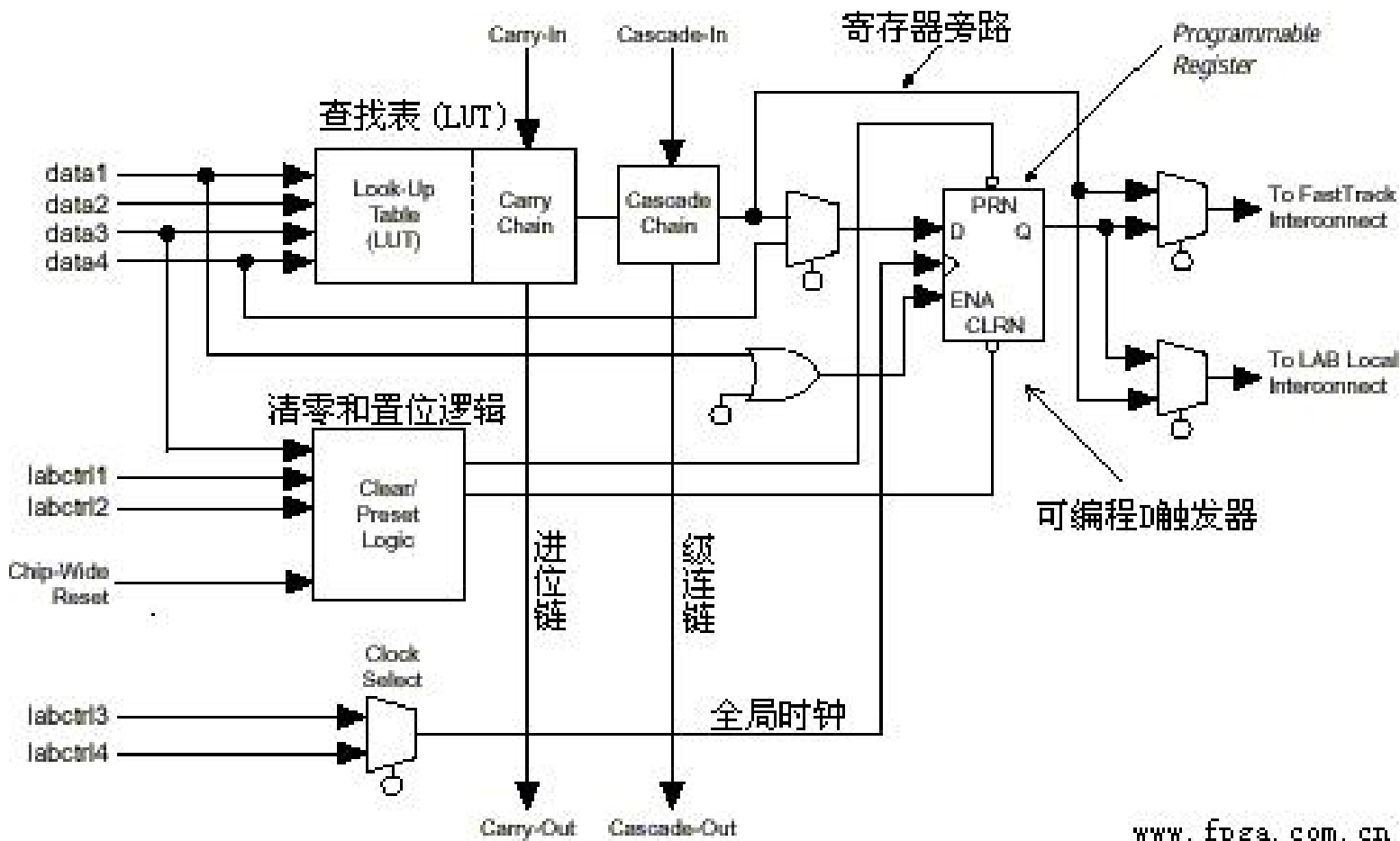
- ◆ 查找表（Look-Up-Table）简称为LUT，LUT本质上是一个RAM。目前FPGA多使用4输入的LUT，所以每一个LUT可以看成为一个具有4位地址线的 16×1 的RAM。当用户通过原理图或HDL语言描述了一个逻辑电路以后，FPGA开发软件会自动计算逻辑电路的所有可能的结果，并把结果事先写入RAM，这样，每输入一个信号进行逻辑运算就等于输入一个地址进行查表，找出地址对应的内容，然后输出即可。



查找表的原理

实际逻辑电路		LUT的实现方式	
			
a, b, c, d 输入	逻辑输出	地址	RAM中存储的内容
0000	0	0000	0
0001	0	0001	0
....	0	...	0
1111	1	1111	1

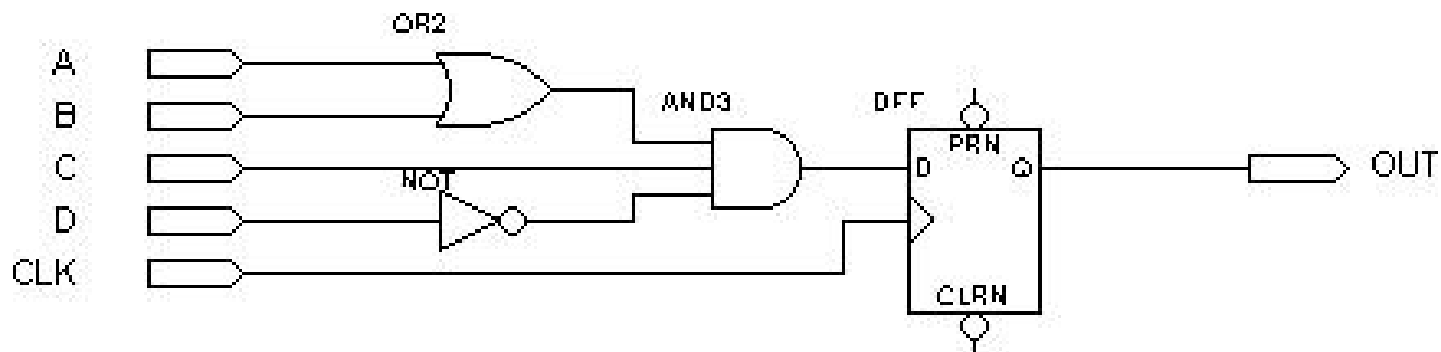
逻辑单元 (LE) 内部结构



www.fpga.com.cn

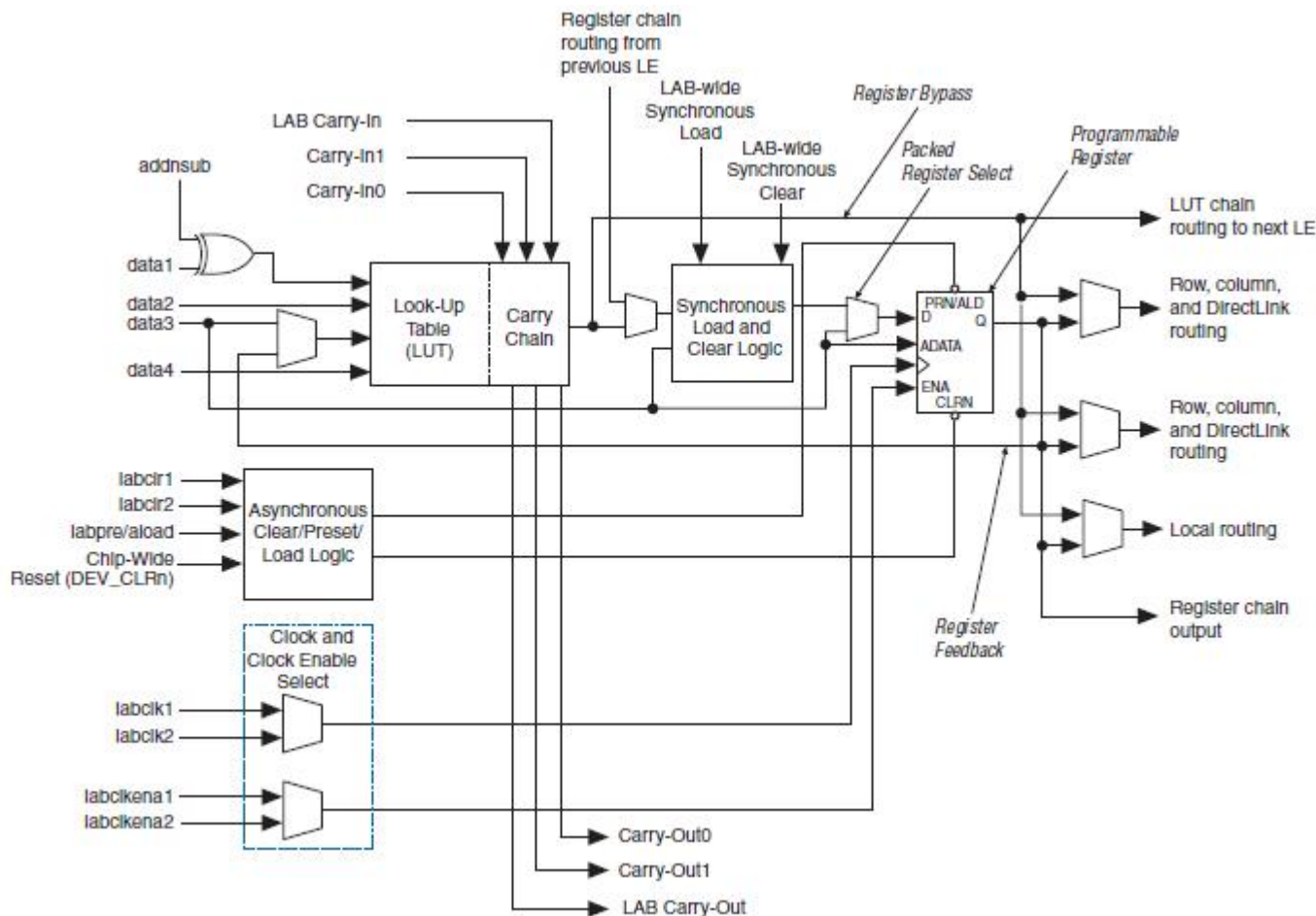


LUT结构的FPGA逻辑实现原理

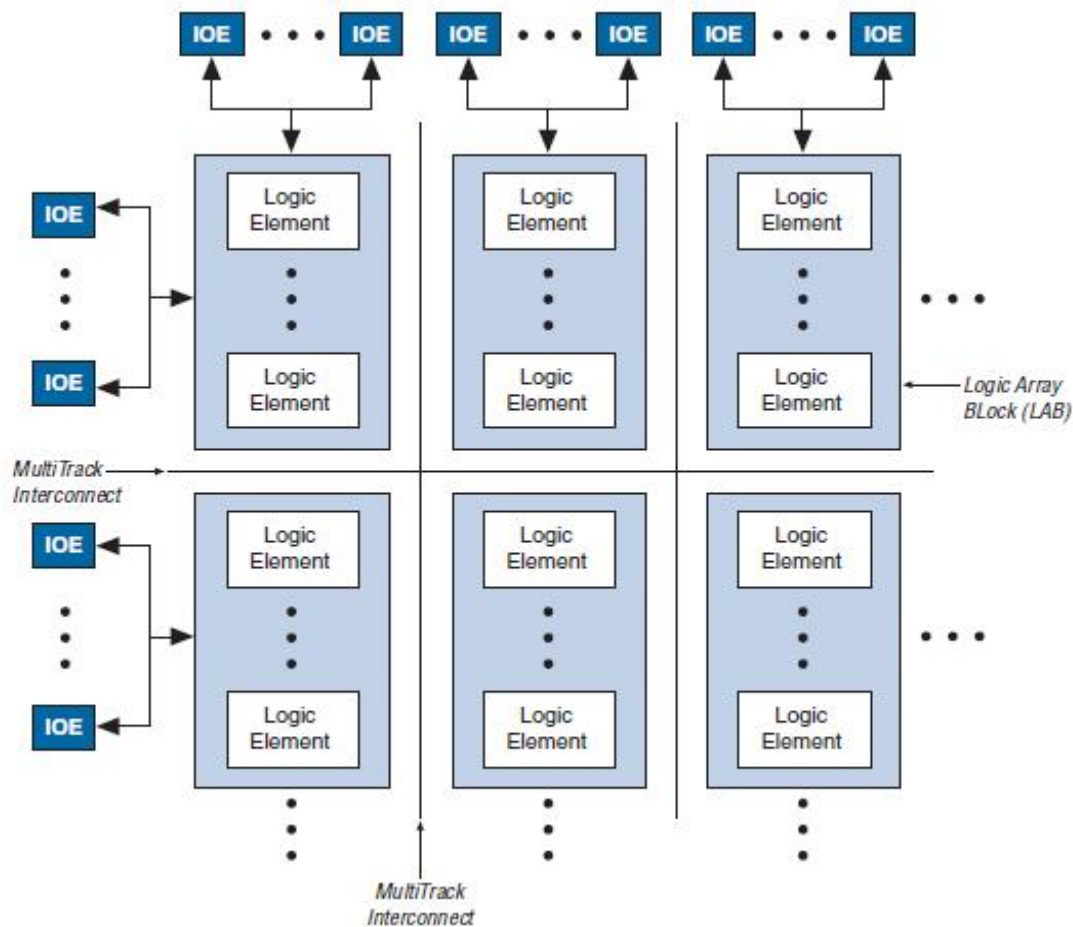


A, B, C, D 输入	逻辑输出	A, B, C, D 输入	逻辑输出
0000	0	1000	0
0001	0	1001	0
0010	0	1010	0
0011	0	1011	0
0100	0	1100	0
0101	1	1101	0
0110	1	1110	0
0111	1	1111	0

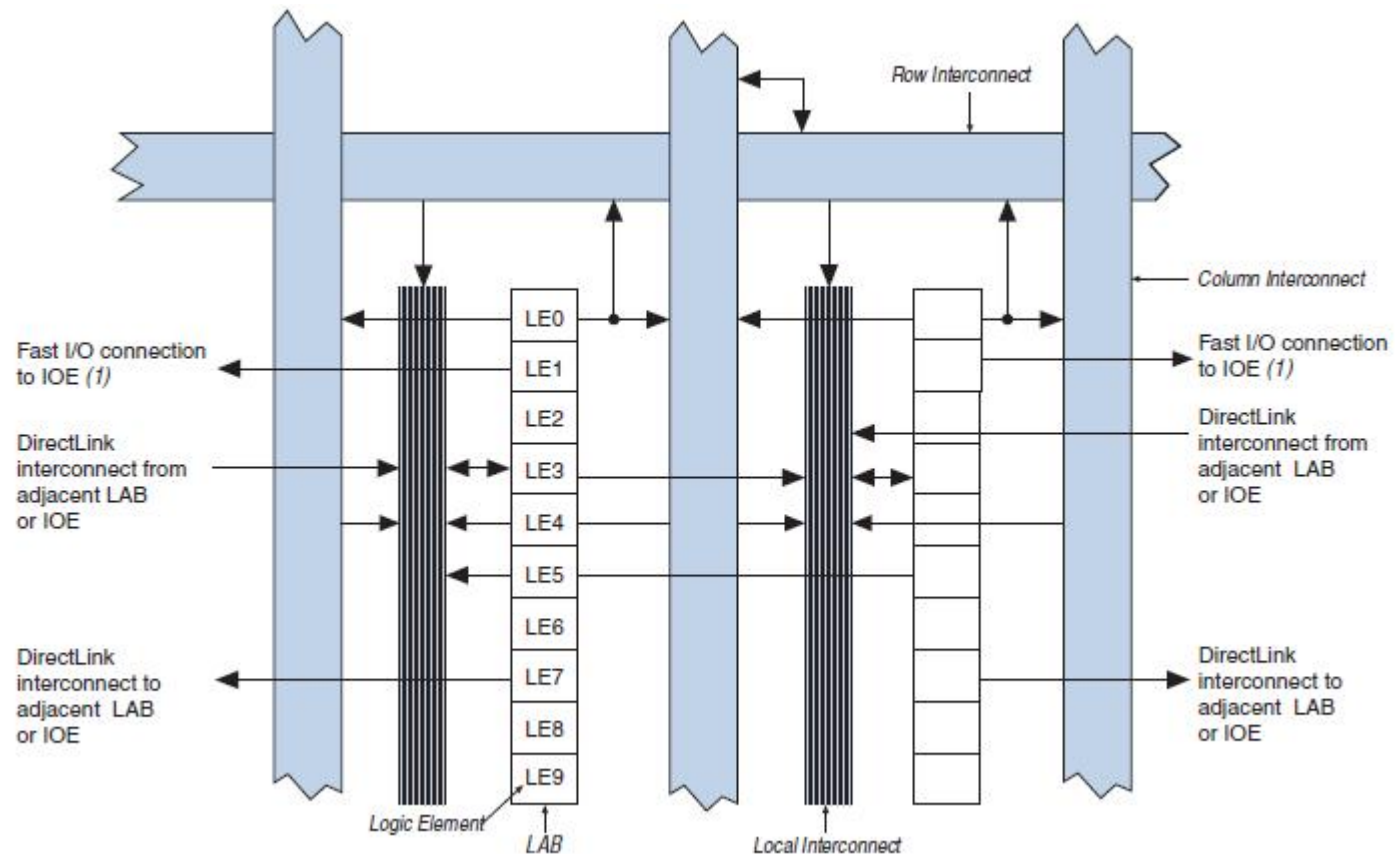
CPLD简介—MAX II 系列LE结构



CPLD简介—MAX II 系列CPLD内部结构



CPLD简介—MAX II 系列LAB结构



FPGA简介

- ◆ FPGA多基于查找表(Look-Up Table)结构，采用SRAM工艺，密度高，触发器多，多用于10,000门以上的大规模设计，适合做复杂的时序逻辑，如数字信号处理和各种算法。FPGA能完成任何数字器件的功能,上至高性能CPU，下至简单的74电路，都可以用FPGA来实现。FPGA已经成为高性能数字系统的首选方案，是电子设计领域最具活力和发展前途的技术，其影响不亚于70年代单片机的发明和使用。



常用FPGA产品

■ ALTERA公司



Stratix IV FPGA

性能最好的设计，
逻辑密度和存储器密度最高的设计，
以及ASIC原型开发。

Arria II FPGA

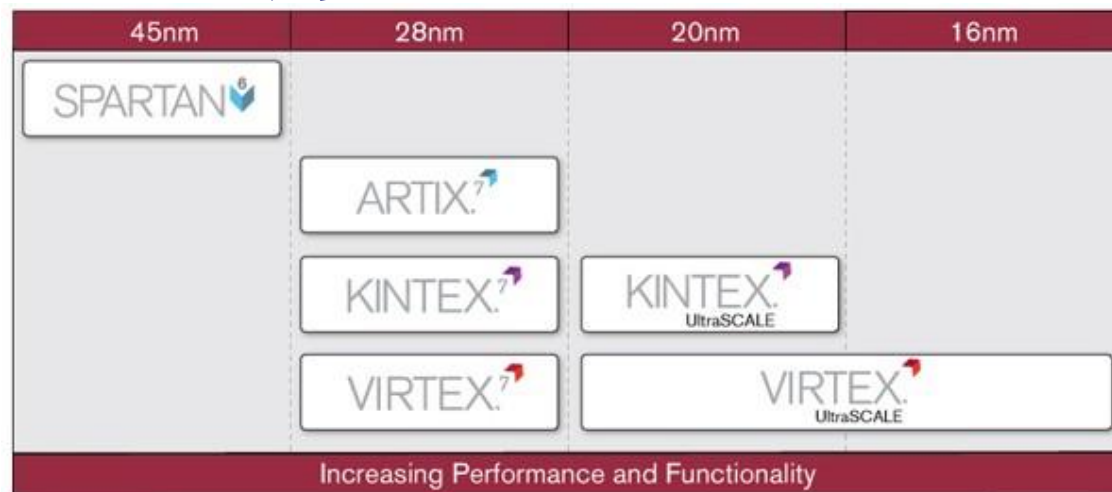
需要数字信号处理 (DSP) 等高性能计算的
低成本应用

Cyclone III FPGA

成本最低、功耗最低的大批量应用



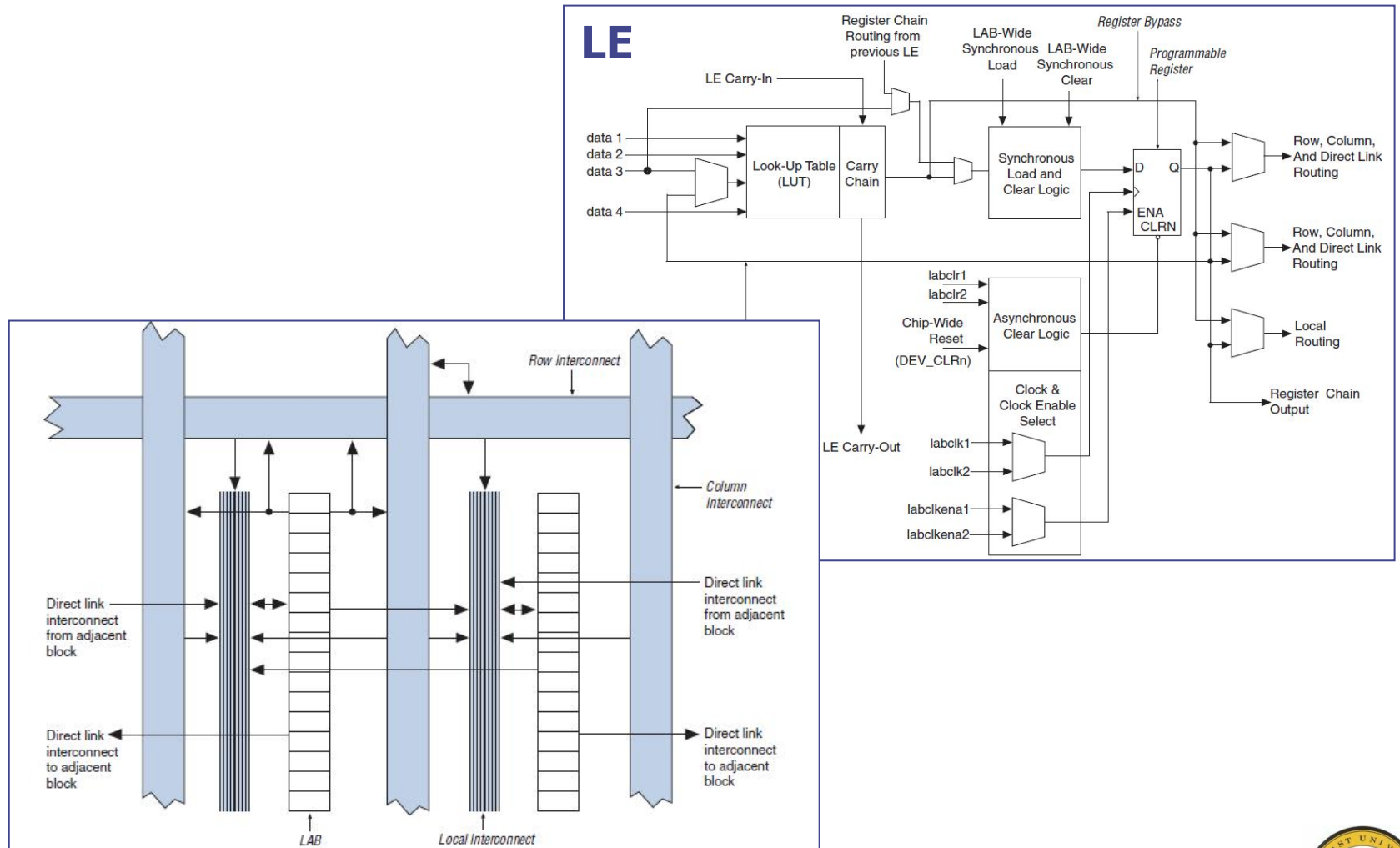
■ XILINX公司



■ LATTICE公司

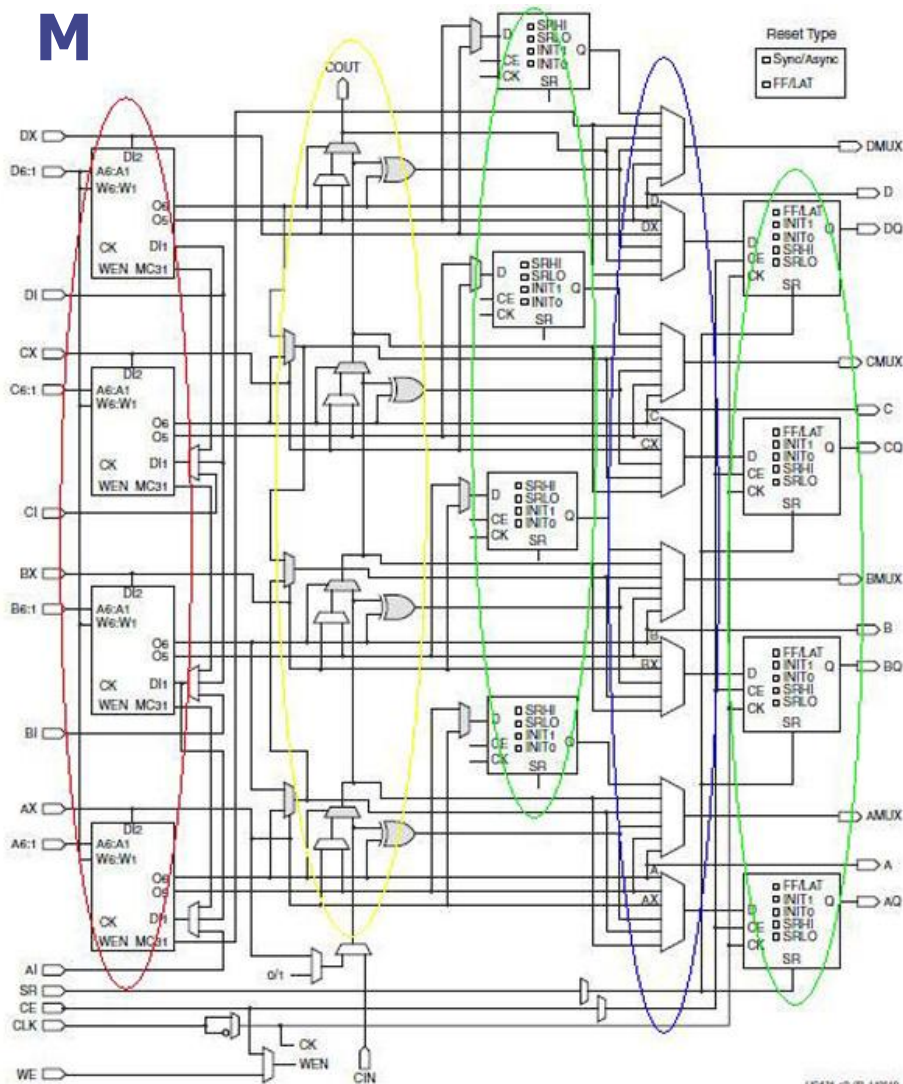


FPGA简介—CycloneIII系列内部结构



FPGA简介—Kintex-7系列内部结构

SLICE M



- CLB（可编程逻辑块）构成了Kintex7主要逻辑单元，其中包含2个Slice，并且Slice分为2种：SLICEL和SLICEM。
- SLICEL为普通的Slice逻辑单元，而SLICEM在基本逻辑功能的基础上可以扩展为分布式RAM或者移位寄存器。
- 在所有Slice资源中，有2/3是SLICEL，因此一个CLB可以有2个SLICEL或者1个SLICEL、1个SLICEM组成。



FPGA与CPLD的区别

◆ 简单地说，FPGA就是将CPLD的电路规模，功能，性能等方面强化之后的产物。

■ **PLD : Programmable Logic Device**（可编程逻辑器件）

- 可反复编程的逻辑器件
- 用户可自行设计与实现
- 可即时进行设计与产品规格上的变更
- 可以以标准零件的形式购买



FPGA与CPLD的区别

	CPLD	FPGA
组合逻辑的实现方法	乘积项 (product-term) , 查找表 (LUT, Look up table)	查找表 (LUT, look up table)
编程元素	非易失性 (Flash, EEPROM)	易失性 (SRAM)
特点	<ul style="list-style-type: none">•非易失性：即使切断电源，电路上的数据也不会丢失•立即上电：上电后立即开始运作•可在单芯片上运作	<ul style="list-style-type: none">•内建高性能硬宏功能<ul style="list-style-type: none">○ PLL○ 存储器模块○ DSP 模块•用最先进的技术实现高集成度，高性能•需要外部配置ROM
应用范围	偏向于简单的控制通道应用以及胶合逻辑	偏向于较复杂且高速的控制通道应用以及数据处理
集成度	小~中规模	中~大规模



FPGA的应用

- ◆ 如下图所示，FPGA被广泛地使用在通讯基站、大型路由器等高端网络设备，以及显示器(电视)、投影仪等日常家用电器里。



FPGA的优势

- ◆ **FPGA**最大的优势特点就是能够缩短开发所需时间。换句话说，通过使用**FPGA**，设计人员可以有效地利用每一分钟进行开发。例如，在开发过程中使用**FPGA**与否，可以导致开发时间上 $1/2 \sim 1/3$ 的差别。这使得**FPGA**成为实现“少量多品种”以及“产品周期短”市场不可缺少的器件之一。



FPGA的优势

工程师



- 迅速应用最新的协议与规格
- 可以在产品开发的任何阶段修改设计
- 开发人员可以调用丰富的IP，集中精力在开发创新技术上
- 应用众多可靠的功能，从而缩短设计时间
- 降低功耗以及空间占用量
- 通过使用各种自动化工具，使时序分析等复杂的设计验证更准确，更容易

开发部门负责人



- 缩短开发周期
- 消除了器件停产所带来的风险
- 通过丰富的IP与自动化工具，可以将开发资源集中在不同的产品线上
- 迅速应用最新的协议与规格
- 更有效率的工程师培训
- 可以重新使用设计资源，降低开发成本并且提高设计质量

管理层



- 降低开发成本
 - ✓ 不需要NRE（Non-recurring Expense：非经常性费用，开发初期所需费用）
 - ✓ 避免因重新制作所造成的NRE负担
 - ✓ 开发周期的短缩，从而降低劳动力成本
- 降低风险
 - ✓ 不存在产品停产所带来的风险
 - ✓ 众多可靠的功能
- 迅速使产品投入市场
- 针对竞争产品实施差别化战略



内容提要

- ◆数字系统设计方法
- ◆VHDL/verilog语法与使用
- ◆vivado使用介绍
- ◆一个实例



数字系统设计方法

- ◆几个专用名词的介绍
- ◆数字系统设计的概念
- ◆数字系统设计流程
- ◆数字系统设计方法论



数字系统设计的概念

- ◆传统设计——中规模集成电路（**MSI**）和小规模集成电路（**SSI**）的适当组合.主要设计小型数字系统
- ◆现代设计——利用**EDA**软件,使用硬件描述语言在**PLD**器件上设计数字系统.适合大中型系统



数字系统设计的概念

◆传统设计——

1. 缺少灵活性
2. 所需芯片种类众多, 数量众多
3. 测试、修改困难, 设计效率低
4. 设计资源难以重用, 资源难以共享
5. 设计成本高, 设计周期长



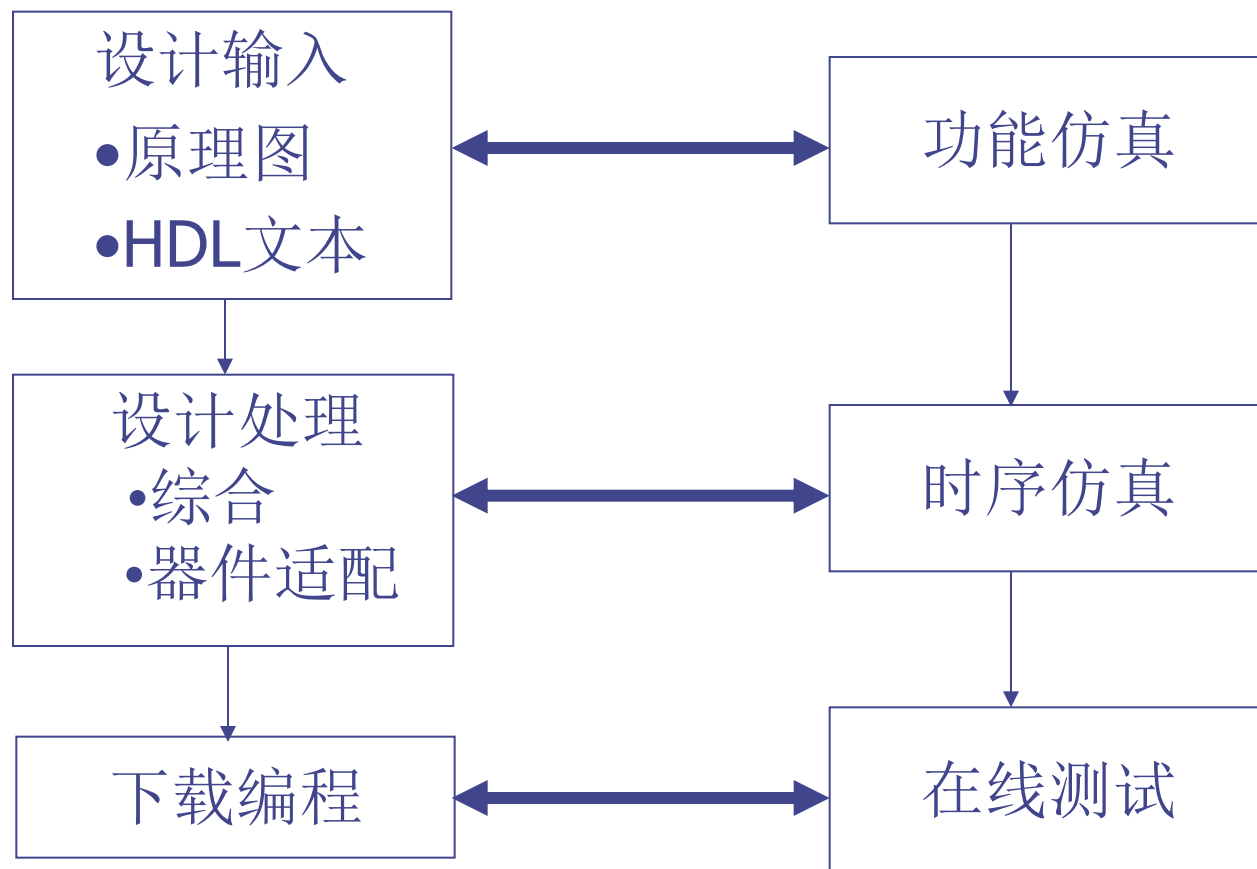
数字系统设计的概念

◆现代设计——

1. 可以从行为级开始设计
2. 设计的自由度和灵活性大, 高效率
3. 芯片单片即可完成, 可重复利用
4. 缩小体积, 降低功耗, 降低成本
5. 测试修改简单, 设计资源重用性强
6. 方便资源共享, 方便程序移植



数字系统设计流程



数字系统设计流程

◆设计输入

1.原理图输入方式

图形化的输入方式,利用元件符号和连线来描述设计.适合描述连接关系和接口关系,简单直观,适合描述层次结构和模块化结构.适合简单逻辑.

2.HDL输入方式

用文本方式来描述设计,逻辑描述能力强,但描述接口和连接关系不如图形方式直观,适合于描述和仿真复杂的逻辑设计.



数字系统设计流程

◆设计处理

1、综合

综合工具将**HDL**行为级描述或原理图描述转化为结构化的门级电路或电路描述网表文件。就是将软件描述与给定的硬件结构用某种网表文件的方式对应起来，成为互相对应的映射关系。

2、器件适配

适配器将综合器产生的网表文件配置与指定的目标器件中，产生最终的下载文件。适配所选定的目标器件(FPGA/CPLD芯片)必须属于原综合器指定的目标器件系列。



数字系统设计流程

◆功能仿真时序仿真

1、功能仿真（前仿真）

直接对**HDL**、原理图描述的逻辑功能进行模拟测试,以了解实现的功能是否满足设计要求，不经过综合和器件适配，编译后即可进行仿真。

2、时序仿真（后仿真）

综合和器件适配后的仿真，包含了具体器件的硬件特性参数，仿真精度高，接近于实际器件运行状况。



数字系统设计流程

◆编程下载

把适配后生成的下载或配置文件,通过编程器或编程电缆向**FPGA**或**CPLD**进行下载,以便进行硬件调试和验证;

- 通常,将对CPLD的下载称为编程(Program),对FPGA中的SRAM进行直接下载的方式称为配置(Configure),但对于OTP FPGA的下载和对FPGA的专用配置ROM的下载仍称为编程。



设计方法论(Design Methodology)

- ◆设计说明书(Specification)
- ◆自顶向下(Top-down)
- ◆自底向上(Bottom up)
- ◆基于原理图设计(Schematic based)
- ◆基于HDL的设计(HDL based)
- ◆仿真与验证
(Simulation & verification)



设计说明书(Specification)

- ◆明确设计任务和指标是关键的第一步
- ◆在想要达到和能够达到之间权衡
(Compromise between what is wanted and what can be made)
- ◆详细说明必须被系统级设计人员认可,设计过程中说明书大的改动将导致设计的显著迟滞



设计说明书(Specification)

- ◆设计需求应当在多个层面考虑:系统级 (System Level)、子系统级 (Sub-System Level)、板级 (Board Level) 等等
- ◆设计说明书必须能够被系统仿真验证正确
- ◆设计说明书的制定、仿真、验证占整个 Project 的 $\frac{1}{4}$ — $\frac{1}{3}$ 工作量



自顶向下(Top-down)

- ◆确定系统级算法 (Algorithm)
- ◆确定系统级架构 (Architecture)
- ◆定义功能模块 (Functional modules)
- ◆定义设计层次 (Design hierarchy)
- ◆确定子模块 (Sub-blocks)
- ◆定义子模块单元(Units)
- ◆综合,布局布线,适配,仿真,验证



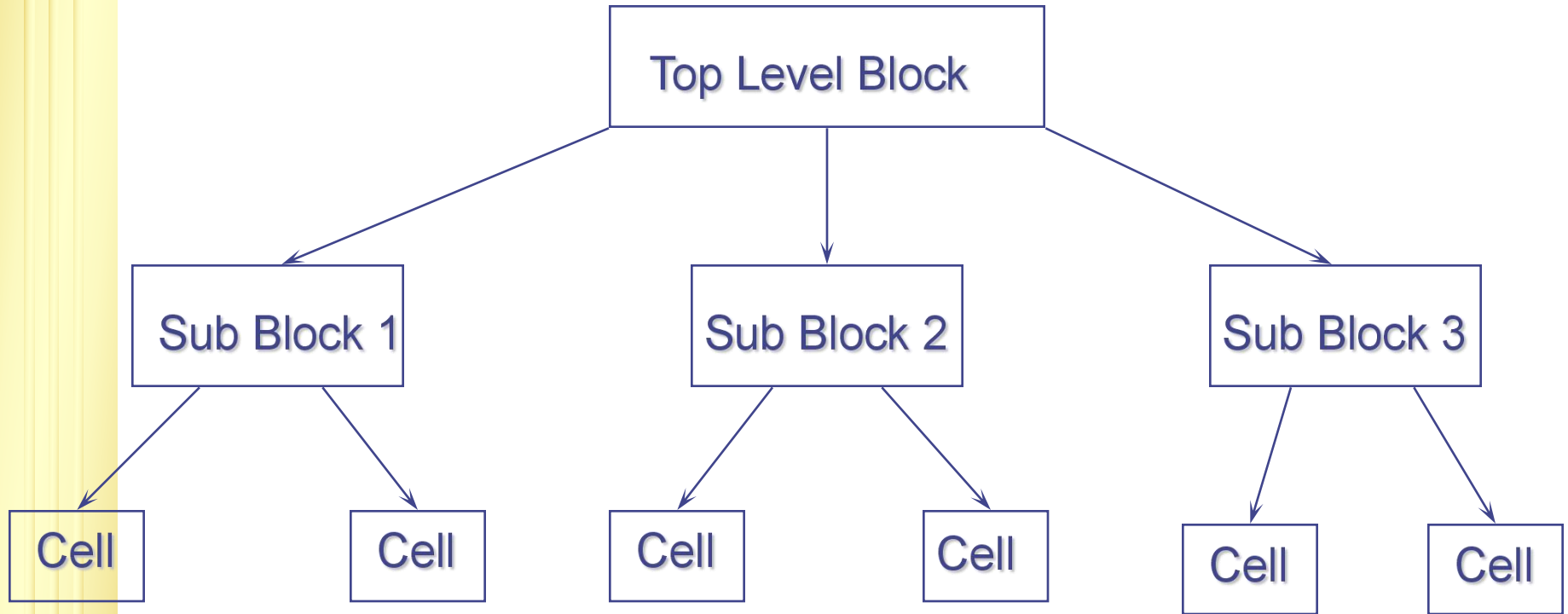
自顶向下(Top-down)

◆FOR SIMPLE:

- 1.定义顶层模块并确定子模块
(Define top-level block and identify the sub-blocks)
- 2.细化子模块直到底层逻辑单元
(Divide sub-block until we come to logic cells)



自顶向下(Top-down)



自底向上(Bottom up)

- ◆用给定的工艺建立门单元(Gates)
- ◆用门单元建立基本单元(Basic Units)
- ◆建立通用模块(General Modules)
- ◆组合这些模块(Assembled Modules)
- ◆门级仿真(Gate Level Simulation)



自底向上(Bottom up)

◆FOR SIMPLE:

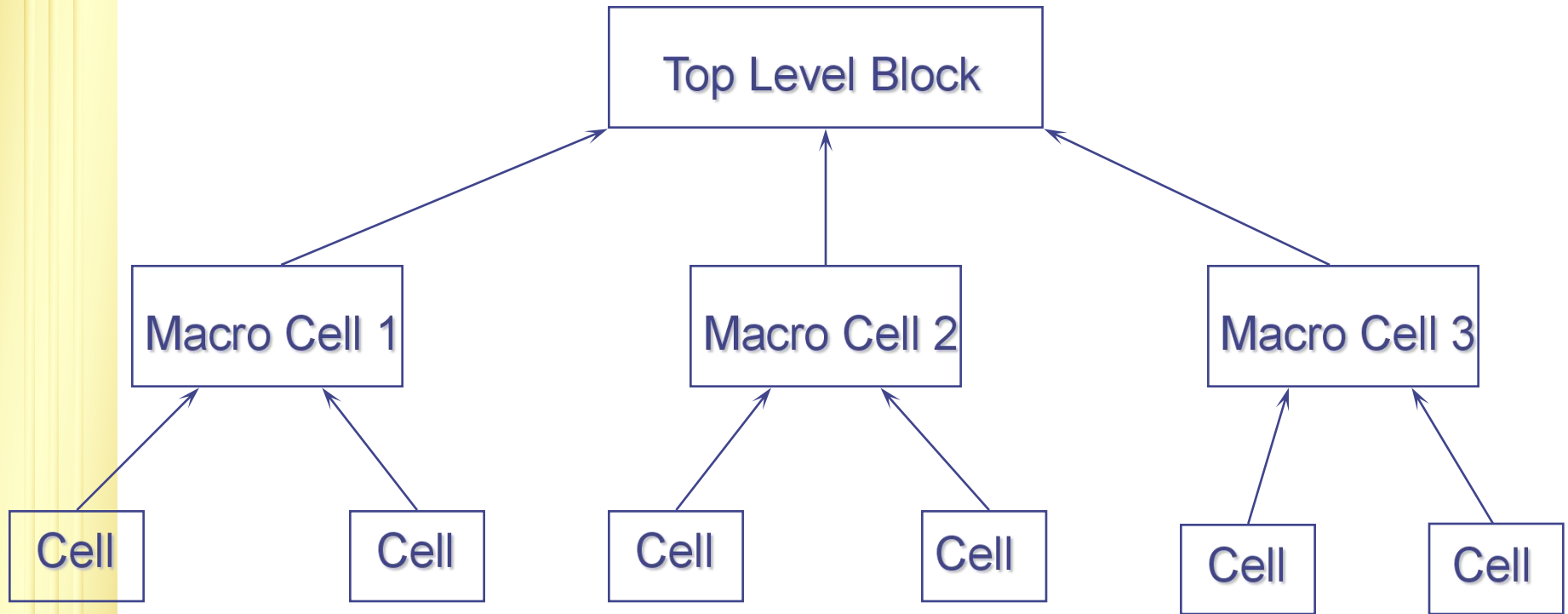
1.确定那些我们可以使用的单元
(Identify building block that are available for us)

2.用这些模块建立较大的单元
(Build bigger cells using these block)

3.继续直到顶层
(Continue until we build the top level)



自底向上(Bottom up)



比较

◆传统设计—自底向上

必须首先关注并致力于解决系统底层硬件的可获得性,以及它们的功能特性的细节问题,在逐级设计和测试过程中,必须顾及目标器件的技术细节.设计的任意时刻,底层目标器件的更换、缺货、成本限制、性能参数限制等等不可预料的因素都可能使整个设计工作前功尽弃.

因此,某些情况下,自底向上的设计方法是低效的,低可靠的,费时费力的,成本高昂的设计方法.



比较

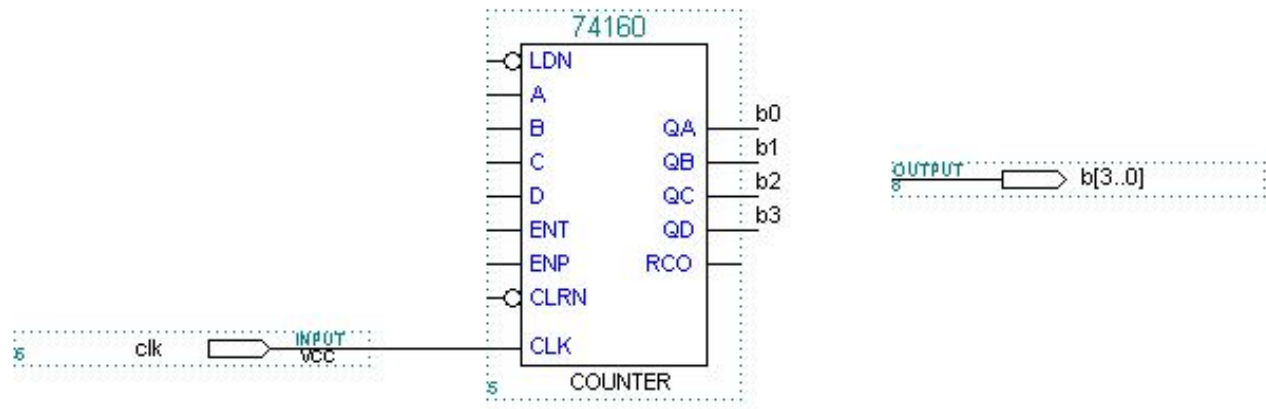
◆现代设计—自顶向下

自顶向下的设计方法就是在设计流程中各环节逐步求精的过程.从自然语言的说明到算法的HDL描述,系统的分解,RTL模型的建立,门级模型的产生,到最后可以物理布线实现的底层电路,就是从高抽象级别到低抽象级别的设计周期.基于强大的EDA工具,自顶向下的设计方法已经成为大规模电路的设计首选,是ASIC和FPGA开发的主要设计手段.



基于原理图设计(Schematic based)

◆ 以图形方式定义模块/器件的互连和功能



基于HDL的设计(HDL based)

- ◆ 用HDL定义模块及其行为(Behavior)
- ◆ 用综合工具生成原理图或网表(Net lists)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter is port(clk: in std_logic;
                        count:buffer std_logic_vector(3 downto 0));
end counter;
architecture behave of counter is
Begin
process(clk)
Begin
if(clk'event and clk='1')then
    if count="1001"then count<="0000";else count<=count+1;
    end if;
end if;
end process;
end behave;
```



数字系统设计基础知识

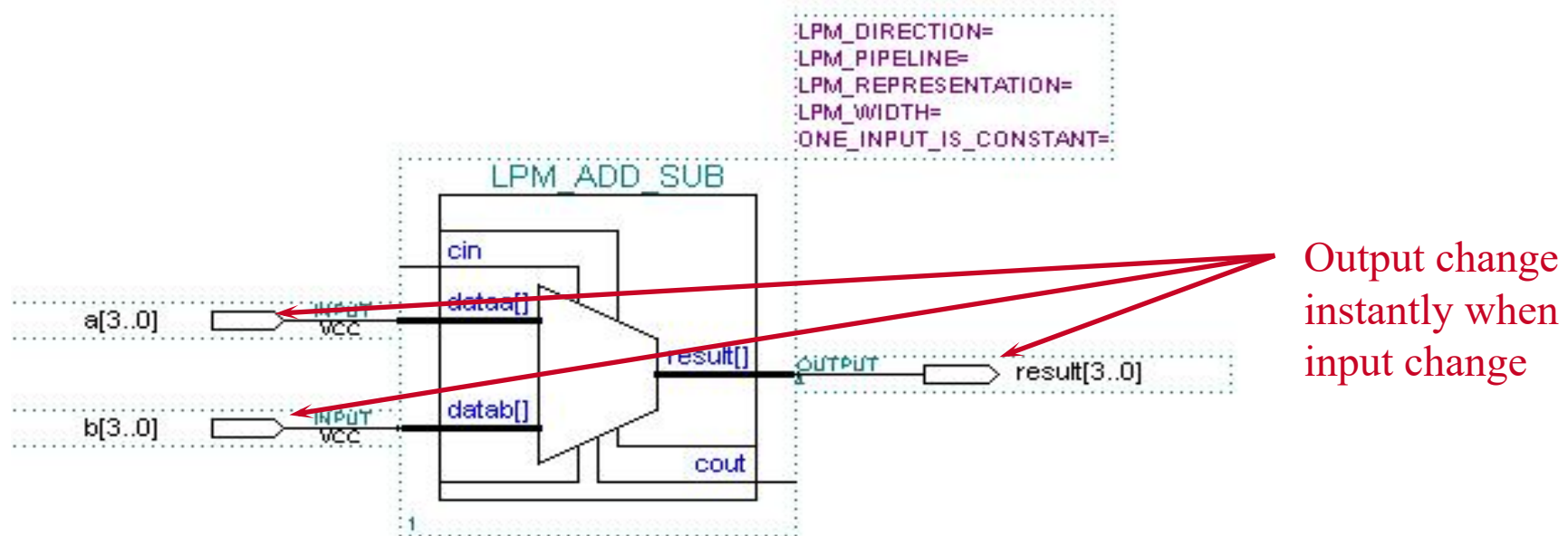
- ◆组合逻辑&时序逻辑
- ◆建立时间&保持时间
- ◆触发器&锁存器
- ◆资源共享
- ◆流水线设计



组合逻辑 (Combinational Logic)

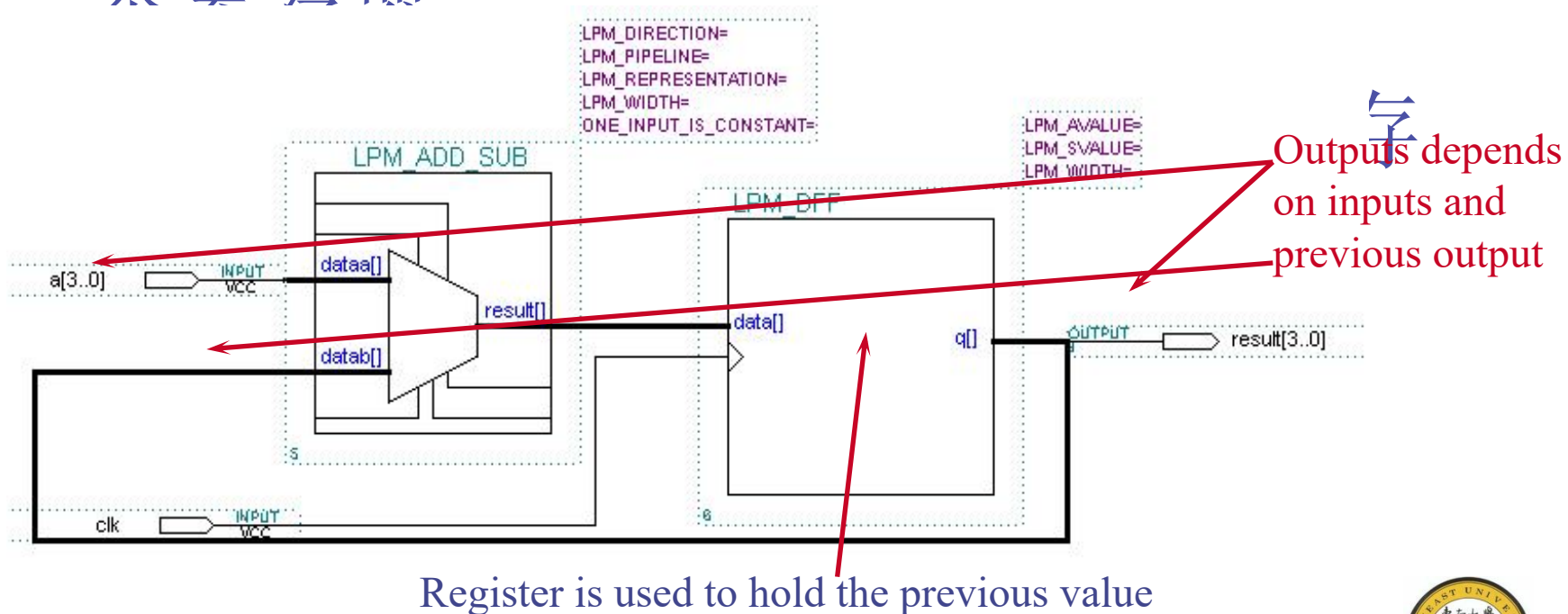
◆ 某一时刻的输出只和此刻的输入有关系

比如：译码器、加法器、数据选择器



时序逻辑 (Sequential Logic)

- ◆ 某一时刻的输出和当时以及以前的时刻都有关系
- ◆ 所有的时序电路都必须包含一个或多个寄存器



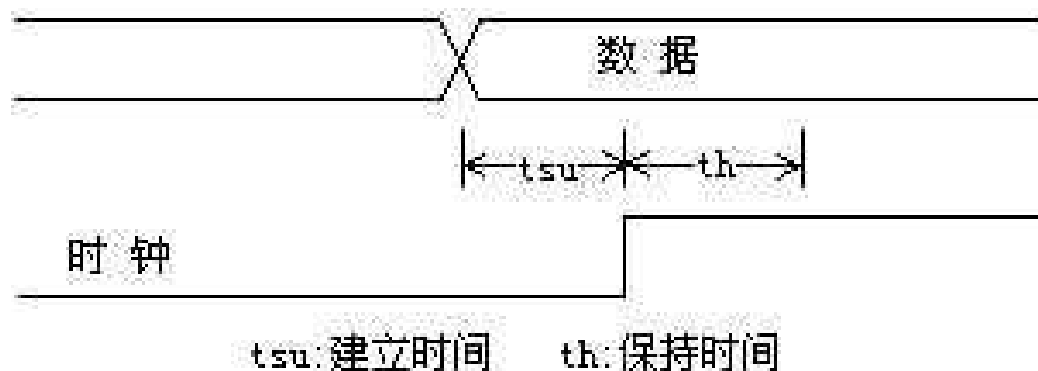
建立时间 & 保持时间

- ◆建立时间（setup time）是指在触发器的时钟信号上升沿到来以前，数据稳定不变的时间，如果建立时间不够，数据将不能在这个时钟上升沿被打入触发器。
- ◆保持时间（hold time）是指在触发器的时钟信号上升沿到来以后，数据稳定不变的时间，如果保持时间不够，数据同样不能被打入触发器。



建立时间 & 保持时间

◆数据稳定传输必须满足建立和保持时间的要求。



触发器 & 锁存器

◆ 触发器是在时钟边沿进行数据的锁存的，而锁存器是用电平使能来锁存数据的。所以触发器的Q输出端在每一个时钟边沿被更新，而锁存器在使能电平有效期间才会被更新。从经验上讲，FPGA设计中应该尽量使用触发器。



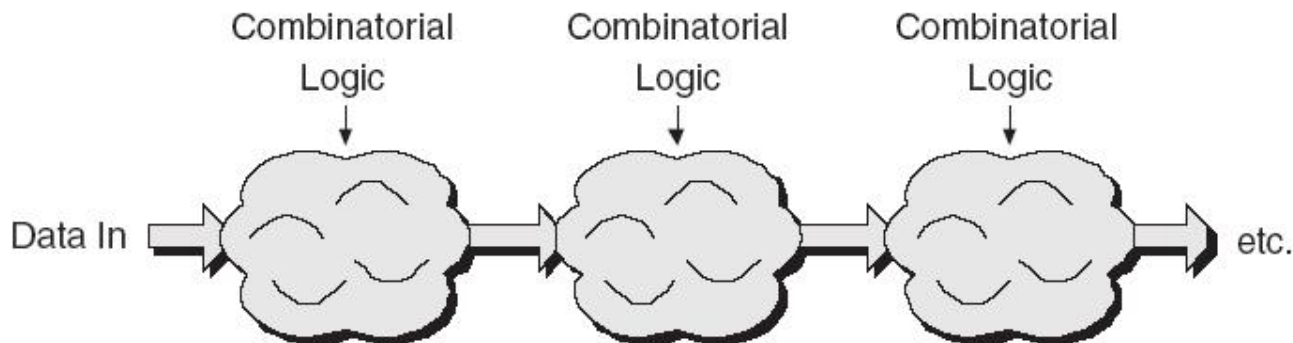
资源共享

◆资源共享是一种优化技术，该技术使得可以使用单个功能模块（比如说一个加法器或一个比较器）来实现一系列操作。比如说，一个乘法器可能先处理两个值A和B，然后同样是这个乘法器，再来处理C和D。资源共享的典型应用是时分复用（Time-Division Multiplexing—TDM）。



流水线设计 (Pipeline)

- ◆ 假设我们要设计的电路或电路的一部分可以由一些组合逻辑块串联实现，如图表示。
- ◆ 假设每个组合逻辑块需要使用 N 纳秒来实现其功能，而我们假设现在有3个这样的组合逻辑块，那么从数据进入第一个块到最后输出，共花费 $3 \times N$ 纳秒的时间。



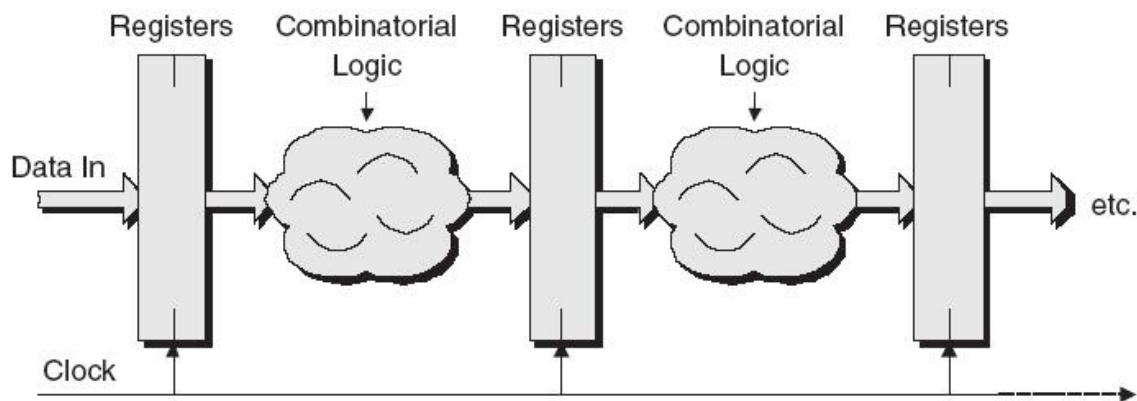
流水线设计 (Pipeline)

◆通常情况下，组合电路中，在我们得到有效的输出数据之前，无法输入新的数据。这样带来的结果是每个组合逻辑块在一次数据处理过程中（ $3 \times N$ 纳秒）只有 N 纳秒在有效工作。那么如何每个块都充分的利用起来呢？答案就是使用流水线操作技术，将逻辑块与块之间用寄存器进行隔离，以提高效率和速度。



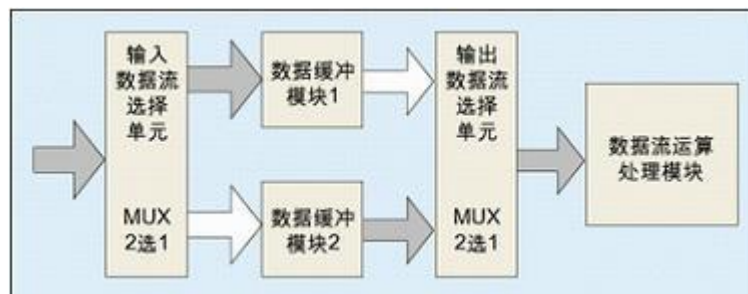
流水线设计 (Pipeline)

- ◆所有寄存器使用共同的时钟信号，在每一个有效的时钟边沿时刻，寄存器的值都要更新一次，更新的值是该寄存器前一级的组合电路的输出。于是这些值被逐级传输直到输出。在这种情况下，一旦“水泵中的水被抽满”，流水线饱和运作，处理一个数据的时间将变为 N 纳秒。



乒乓操作

- 在第一个缓冲周期，将输入的数据流缓存到“数据缓冲模块1”；在第2个缓冲周期，通过“输入数据选择单元”的切换，将输入的数据流缓存到“数据缓冲模块2”，同时将“数据缓冲模块1”缓存的第1个周期数据通过“输出数据选择单元”的选择，送到“数据流运算处理模块”进行运算处理；在第3个缓冲周期通过“输入数据选择单元”的再次切换，将输入的数据流缓存到“数据缓冲模块1”，同时将“数据缓冲模块2”缓存的第2个周期的数据通过“输出数据选择单元”切换，送到“数据流运算处理模块”进行运算处理。如此循环。



乒乓操作

- ◆ 乒乓操作的最大特点是通过“输入数据选择单元”和“输出数据选择单元”按节拍、相互配合的切换，将经过缓冲的数据流没有停顿地送到“数据流运算处理模块”进行运算与处理。把乒乓操作模块当做一个整体，站在这个模块的两端看数据，输入数据流和输出数据流都是连续不断的，没有任何停顿，因此非常适合对数据流进行流水线式处理。所以乒乓操作常常应用于流水线式算法，完成数据的无缝缓冲与处理。



◆ Vivado 2017.4

东大云盘

