搭建你的数字积木
—数字电路与逻辑设计
（Verilog HDL&Vivado版）
— 参考课件PPT

东南大学 & Xilinx大学计划部

Xilinx 大学合作部 & 东南大学

序言：教材安排及学习建议

技术定位：数字电路是电子信息行业领域核心技术

- **数字电路**是几乎当前所有先进信息技术的核心，已成为社会科技进步的关键核心。

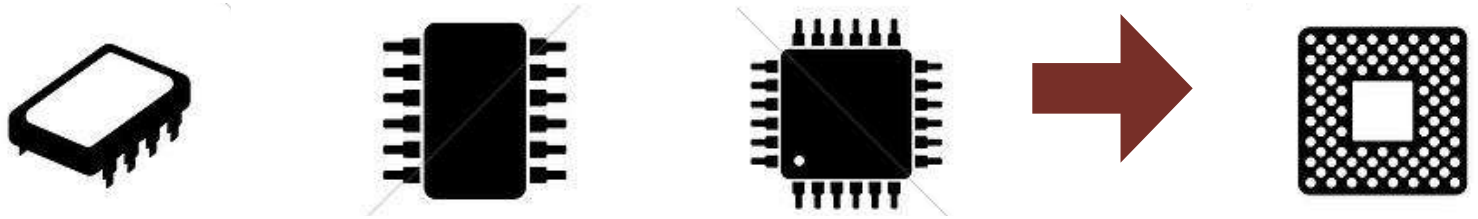
课程定位：数字电路是电子信息专业核心基础课程

- 电子信息类专业**核心基础课程之一**
- 电路类课程逻辑系统设计方向首门课程
- 作用一：培养**逻辑单元/模块基本设计能力**
- 作用二：为**后续课程**学习建立基础
  - *电子系统设计实训*
  - *微机原理与接口、嵌入式系统*
  - *数字信号处理*
  - *数字集成电路设计*

新时代“数字电路”的特点

- 数字电路芯片的**集成度越来越高**

小规模集成电路（SSI） 中规模集成电路（MSI） 大规模集成电路（LSI） 超大规模集成电路（VLSI）



- 数字电路设计普遍采用**编程的方法实现**
  - 主要由一些标准的集成电路块单元连接而成。对于非标准的特殊电路还可以使用可编程序逻辑阵列电路，通过编程的方法实现任意的逻辑功能。

数字电路课程教学目标变迁

- **数字电路基本概念不变**
  - 处理对象不变：**数字量**
  - 主要功能不变：**逻辑运算和逻辑处理**
  - 基本构成不变：**组合逻辑电路、时序逻辑电路**

- **数字电路设计方法变化**
  - 从基于门电路和中小规模集成电路的设计方法变迁到面向**超大规模集成电路**的设计方法
  - 电路设计不再以门电路或中小规模电路连接形式为终点，**硬件电路编程方式**成为主要设计手段，今后继续向通用高级语言编程方式发展

基于PLD的数字电路教学变化

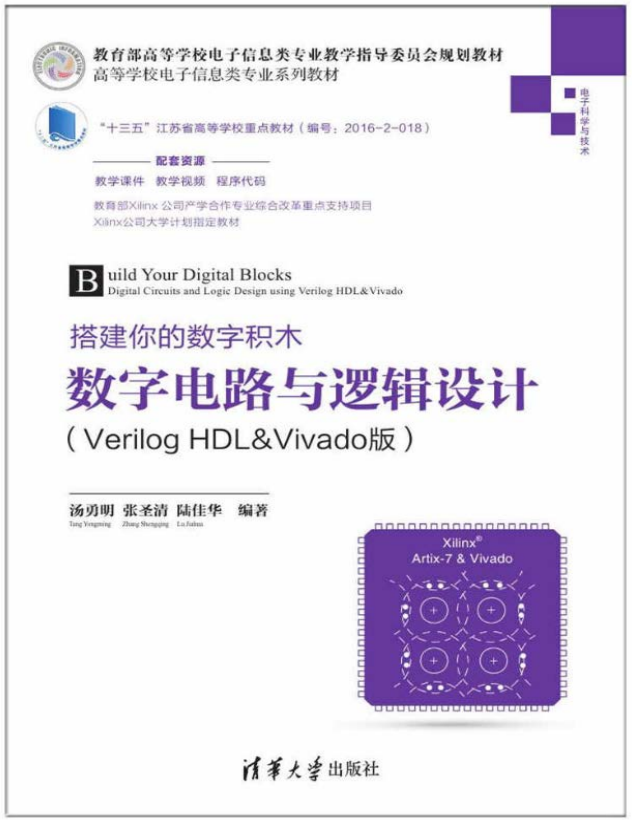
硬件编程设计方法的引入，使得课程实践性增强、教学效率提升、课时需求减少！

本教材编排基本思路

- 以**HDL编程设计为主线**串起数字电路教学内容
  - Vivado综合设计环境起步
  - 布尔代数和数字电路概念均基于HDL设计实现
- 融入**搭积木的理念**（模块化、标准化设计）

第一部分： 逻辑设计基础	第二部分： 常用逻辑设计模块	第三部分： 逻辑系统设计案例
…	IP设计流程	FIR滤波器
…	串口控制器	数字图像处理
…	VGA接口控制器	学生创新作品
HDL编程风格	…	…

- 融入**行业设计案例**和**学生创新作品**提升示范空间



教材可适应的课程

数字电路类课程

- 以第一部分为教学主体，建立逻辑设计基本概念和技巧
- 第二、三部分作为提高部分，根据实际选择

电子系统设计实训类课程

- 快速回顾数字电路内容，跳过第一部分
- 以第二部分为教学主体，讲授设计技巧和实例
- 以第三部分引导同学们开展综合设计

Chap.1. 逻辑设计概述及Vivado基础

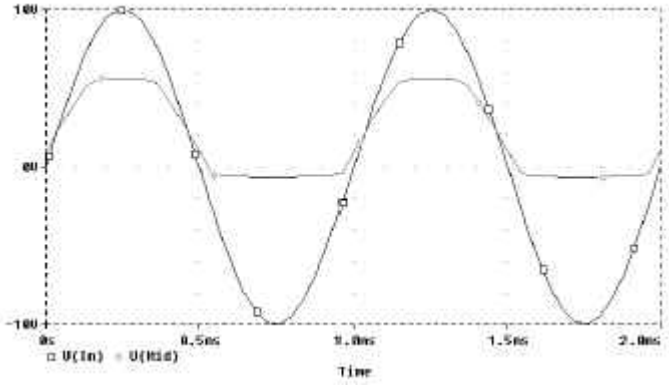
本章学习导言

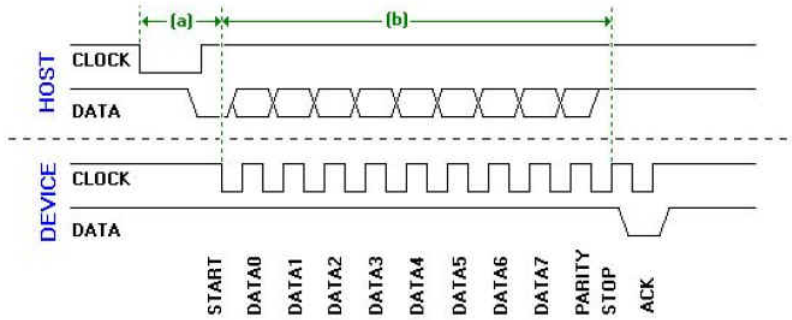
- 以Xilinx为代表的半导体公司推动可编程逻辑器件（PLD）技术进入市场，经过多年的技术发展，已经成为逻辑系统设计和研发行业的主流技术，也带来了与传统基于中小规模逻辑芯片进行逻辑系统设计的巨大变革。
- 本章的教学目的是初步介绍数字电路/逻辑系统设计、可编程逻辑芯片技术和硬件描述语言Verilog HDL等内容，并快速进入最新的可编程逻辑器件综合开发环境Vivado的使用介绍，以帮助学习者及早熟悉工具后对后续基本逻辑单元、模块、系统进行编程练习。

主要内容

- 数字量与数字电路
- 硬件编程语言（HDL）概述
- 可编程逻辑器件基础
- Vivado开发环境及流程演示

模拟信号与数字信号

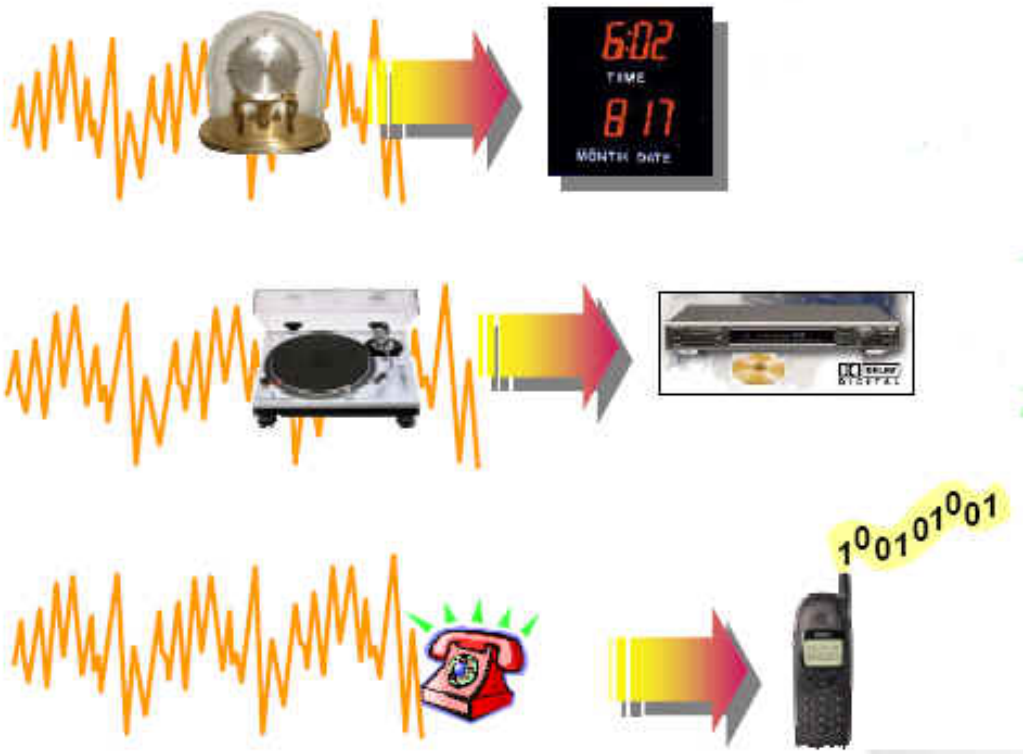




- 连续变化的变量为**模拟量**，离散取值的变量为**数字量**；
- 自然界绝大多数物理量都是模拟量，如温度、湿度、语音等；
- 数字量可以被用来近似表示出模拟量；
- 数字系统用高低电平构成二进制数系统处理数字信号。

数字电路系统的优点

- 保真度高（结果再现性强）；
  - –易存储、易传输、高精度；
  - –抗噪声、受环境影响小；
- 易于设计，EDA支持强；
  - –灵活性和功能性；
  - –可编程性；
- 处理速度快；
- 经济性。



数字系统与产品实例

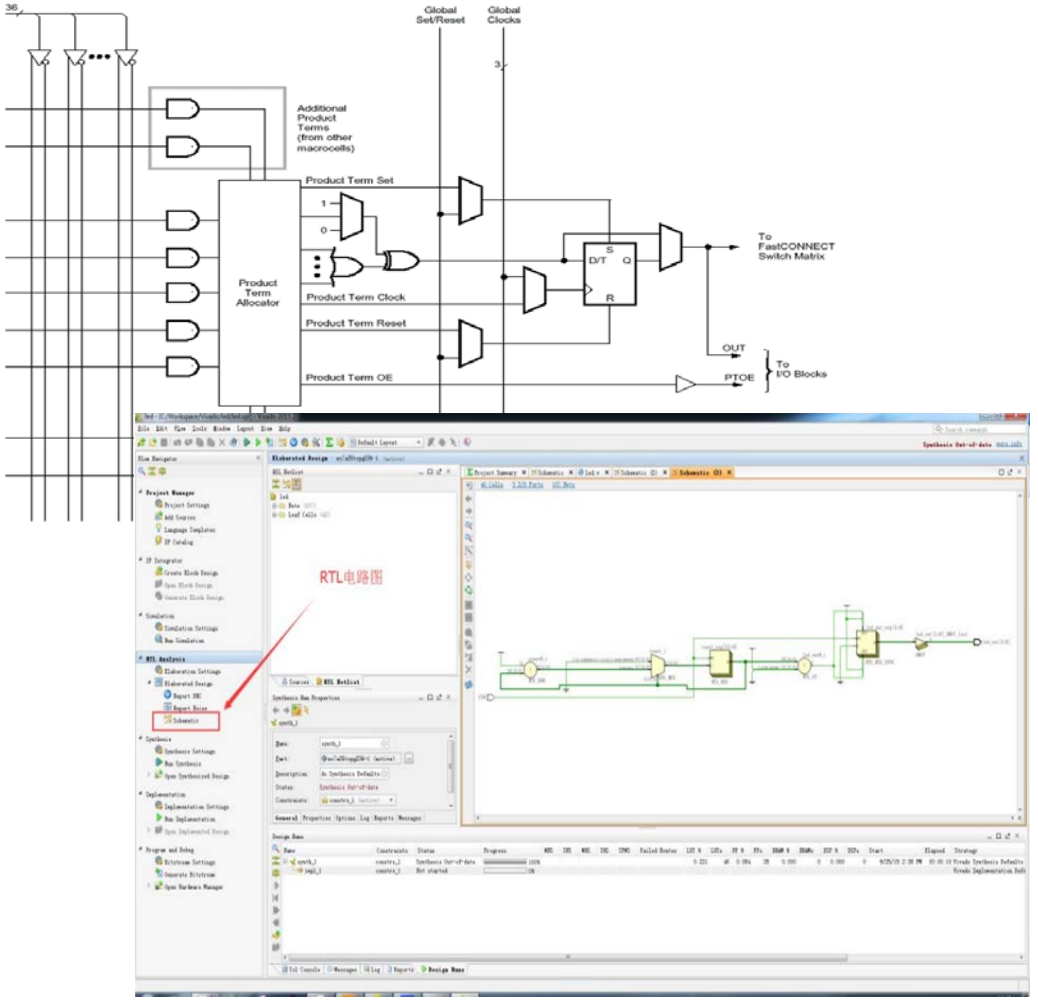
数字电路及其设计技术的发展

数字系统的发展

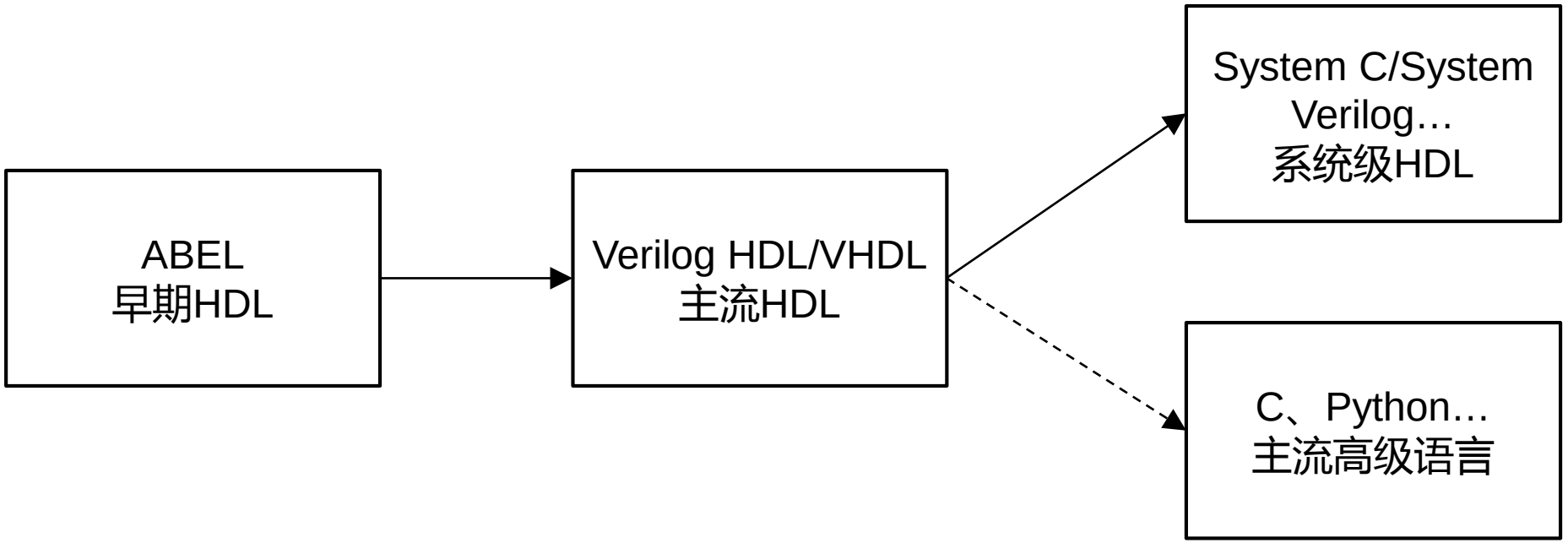
- 门电路和触发器；
- 集成电路；
- 可编程逻辑器件和超大规模专用集成电路；

数字系统设计技术的发展

- 公式法/卡诺图化简
- 计算机辅助设计
  - 硬件描述语言（HDL）
  - 软件综合与仿真



硬件描述语言（*Hardware Description Language*, HDL）



ABEL语言程序段示例

ABEL 源文件构成

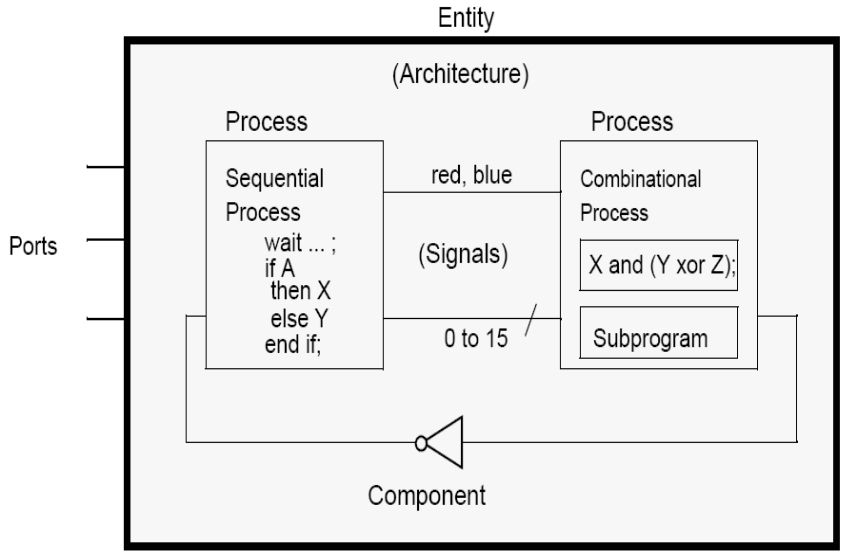
模块开始（module 语句）

标志（flags 语句）	说明段
标题（title 语句）	
器件定义（device 语句）	定义段
管脚、节点定义（pin，node 语句）	
属性定义（istype 语句）	
常量定义（constant 语句）	
宏定义 （macro 语句）	
逻辑方程式（equations 语句）	描述段
真值表（truth_table 语句）	
状态图（state_diagram 语句）	
熔丝段定义（fuses 语句）	熔丝段
测试向量（test_vectors 语句）	测试段

模块结束（end 语句）

```
module m6809a        （模块语句）
title '6809 memory decode        （标题语句）
Jean Designer        Data I/O Corp Redmond WA      24 Feb 1984'

        U09a      device    'P14L4';    （器件定义）
        A15,A14,A13,A12,A11,A10 pin 1,2,3,4,5,6;        （管脚定义）
        ROM1,IO,ROM2,DRAM     pin 14,15,16,17;

        H,L,X     = 1,0, .X.;                                    （常量定义）
        Address = [A15,A14,A13,A12, A11,A10,X,X, X,X,X,X, X,X,X,X];

equations            （方程）
        !DRAM     = (Address <= ^hDFFF);

        !IO       = (Address >= ^hE000) & (Address <= ^hE7FF);

        !ROM2     = (Address >= ^hF000) & (Address <= ^hF7FF);

        !ROM1     = (Address >= ^hF800);

test_vectors (Address -> [ROM1,ROM2,IO,DRAM])        （测试向量）
              ^h0000   -> [   H,   H,   H,    L ];
              ^h4000   -> [   H,   H,   H,    L ];
              ^h8000   -> [   H,   H,   H,    L ];
              ^hC000   -> [   H,   H,   H,    L ];
              ^hE000   -> [   H,   H,   L,    H ];
              ^hE800   -> [   H,   H,   H,    H ];
              ^hF000   -> [   H,   L,   H,    H ];
              ^hF800   -> [   L,   H,   H,    H ];
end m6809a
```

VHDL语言程序段示例




```
entity COUNTER3 is
port ( CLK :  in bit;
       RESET: in bit;
       COUNT: out integer range 0 to 7);
end COUNTER3;

architecture MY_ARCH of COUNTER3 is
signal COUNT_tmp : integer range 0 to 7;
begin
  process
  begin
     wait until (CLK'event and CLK = '1');
                      -- wait for the clock
     if RESET = '1' or COUNT_tmp = 7 then
                      -- Ck. for RESET or max. count
          COUNT_tmp <= 0;
     else COUNT_tmp <= COUNT_tmp + 1;
                      -- Keep counting
     end if;

  end process;
  COUNT <= COUNT_tmp;
end MY_ARCH;
```

3Bit计数器

Verilog HDL语言程序段范例

```
module decoder(in,out);
input [2:0] in;
output[7:0] out;
reg   [7:0] out;

always @(in)
begin
 case(in)
 3'b000: out <= 8'b10000000;
 3'b001: out <= 8'b01000000;
 3'b010: out <= 8'b00100000;
 3'b011: out <= 8'b00010000;
 3'b100: out <= 8'b00001000;
 3'b101: out <= 8'b00000100;
 3'b110: out <= 8'b00000010;
 3'b111: out <= 8'b00000001;
 default: out<= 8'b00000000;
 endcase
end
endmodule
```

```
module counter(clk,rst,set,add,num,cnt);

input clk,rst,set,add;
input [2:0] num;
output[2:0] cnt;
reg[2:0]  cnt;

always @(posedge clk or negedge rst)
begin
  if(!rst)    cnt <= 3'b000;
  else if(set)  cnt <= num;
  else
  begin
    if(add) cnt <= cnt+1'b1;
    else  cnt <= cnt-1'b1;
  end
end

endmodule
```

Verilog HDL与其他HDL的差异

与ABEL等早期语言相比

- 上述语言多应用逻辑等式来描述逻辑功能，侧重于结构描述方法
- Verilog HDL适合算法级、寄存器传输级(RTL)、门级、版图级等各类设计描述应用

与VHDL等同期语言相比

- 两者都具备良好的行为描述能力
- 一般认为，Verilog HDL在描述硬件单元的结构时更简单易读，相比较而言，VHDL的描述长度通常是Verilog HDL的两倍

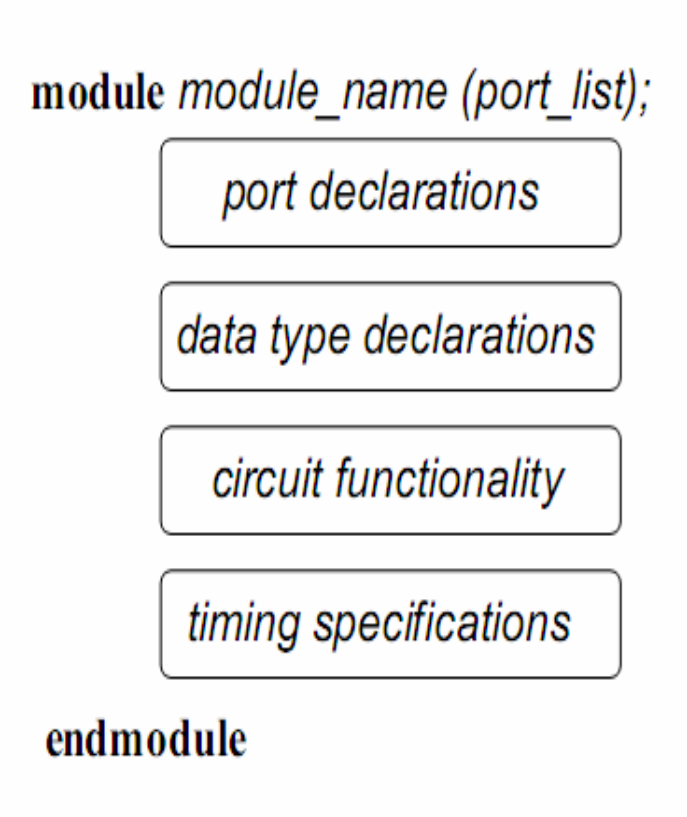
Verilog HDL发展史

- Verilog HDL语言最初于1983年由Gateway Design Automation (GDA) 公司为其模拟器产品开发的硬件建模语言
- Cadence在1989年收购GDA后，Verilog HDL语言于1990年正式对外发布
- Open Verilog International (OVI)成立，以促进Verilog语言规范的发展
- 1993年，OVI推出了2.0版本
- Verilog 语言于1995年成为IEEE标准，称为IEEE Std 1364-1995，2001年又更新了一版标准

Verilog HDL与PLD设计

- PLD设计是Verilog HDL的一大应用
- PLD设计仅支持Verilog HDL的一个子集
- 本课程后续只学习PLD设计支持的Verilog HDL

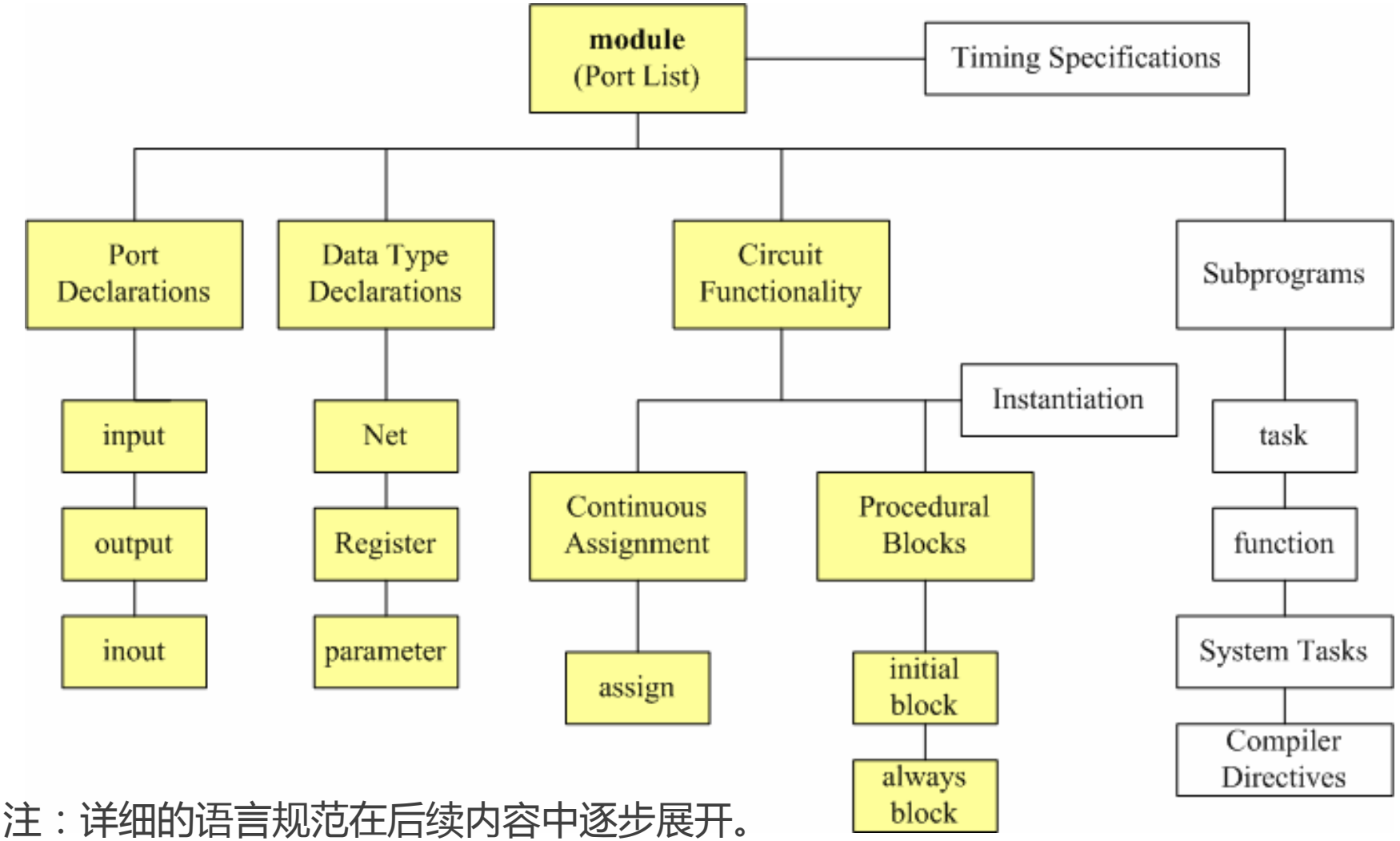
Verilog HDL基本要素



注意点：

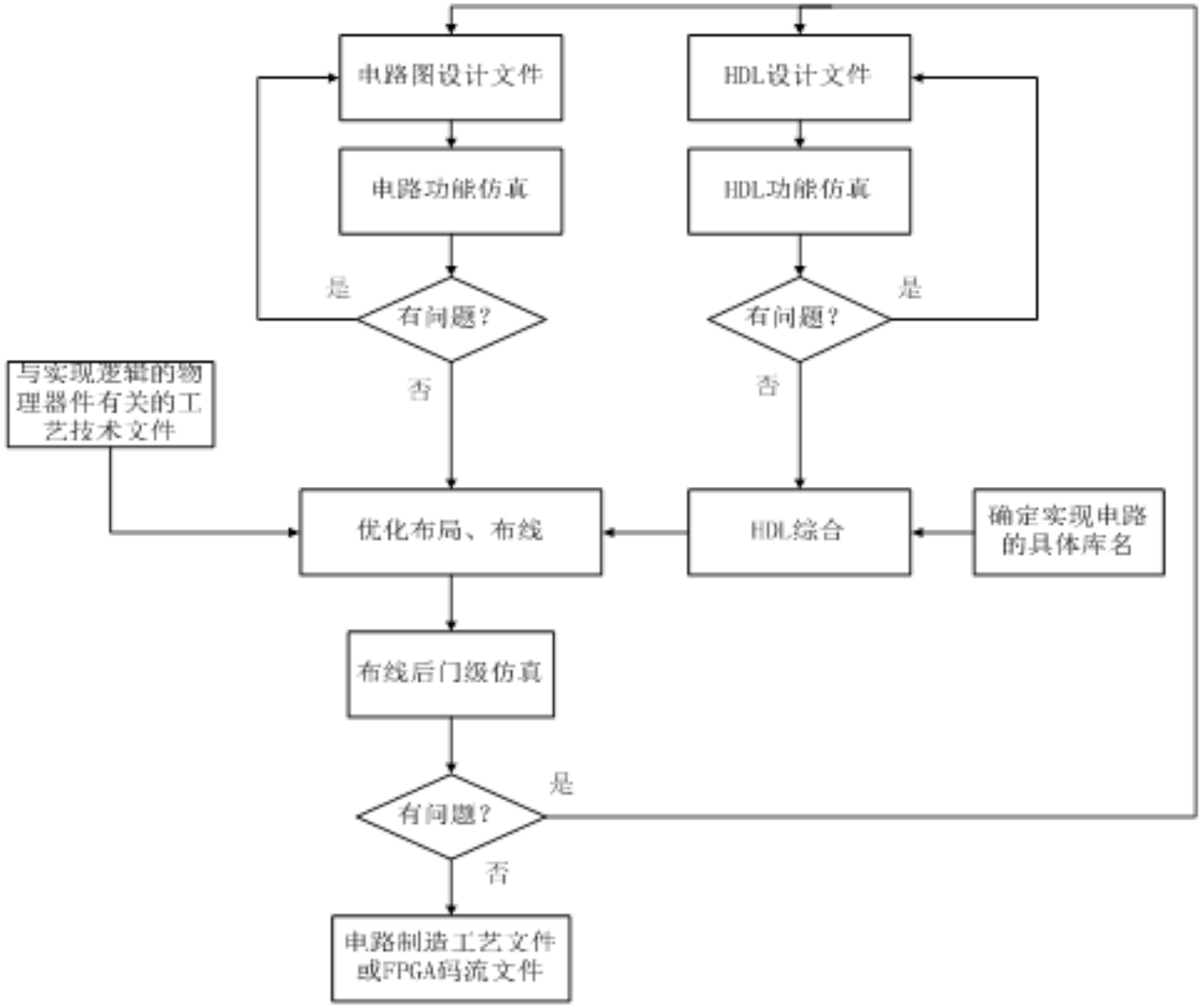
- 大小写敏感
- 所有关键词须小写
- 空格用于增加可读性
- 分号是语句终结符
- 单行注释：//
- 多行注释：/* */
- 时间规范用于仿真

Verilog HDL基本要素图



注：详细的语言规范在后续内容中逐步展开。

逻辑电路与系统设计流程



可编程逻辑器件概述

- **PLD：Programmable Logic Device 可编程逻辑器件**
  - 可编程：执行功能“可且必须”由用户定制
  - 逻　辑：属于逻辑电路，不同于模拟电路
  - 器　件：以单独芯片形式封装和形成产品
- **PLD可实现通用逻辑功能设计的原理**
  - “与或非门+寄存器”的基本逻辑单元组合
  - PLD可提供丰富的基本逻辑单元和连线资源

PLD vs. ASIC

ASIC

优点

- 性能优越
- 可靠性高
- 大批量下单位成本低

缺点

- 开发周期长
  - 设计工具及设计流程复杂
  - 快速进入市场的压力大
- 功能固定，不利修改和扩展
- NRE费用高
- 设计工具及开发人员昂贵

PLD

优点

- 开发初期无投入资金壁垒
- 设计工具使用方便，设计简单快速
- 产品原型机开发时间短
- 管脚定义灵活，便于PCB布板
- 随时进行硬件功能设计修改

缺点

- 单位成本高
- 可靠性低/功耗大

PLD vs. MCU/CPU/DSP

- PLD (CPLD/FPGA)开发模式的特点
  - –以硬件描述语言（HDL）或电路图作为主要开发手段
  - –程序中不同的逻辑功能被设计成不同的硬件模块并存
  - –高速、设计灵活性好
  - –从功能角度可完全取代CPU式开发模式

- CPU (MCU/CPU/DSP)开发模式的特点
  - –以C语言等为主的高级语言作为主要开发手段
  - –微程序模式，程序被分解后依次在同一硬件系统中执行
  - –成本低、功耗小

PLD的发展史

- PLD雏形：PROM/EPROM
- 第一代PLD（二十世纪70年代）
  - **PLA / PAL**
- 第二代PLD（二十世纪80年代）
  - **GAL**
- 第三代PLD（二十世纪90年代起）
  - **CPLD**
  - **FPGA**
  - **其它衍生PLD产品：结构化ASIC、ASSP等**

PLD的发展步伐

- 从一次烧写发展到可反复烧写
- 不断增加可编程逻辑资源的集成度
- 单元逻辑资源的成本和价格不断降低
- 由传统可编程逻辑资源集成扩展更多IP
- HDL的不断成熟（标准化）和进化
- 设计和仿真EDA技术越来越成熟
- 建立向ASIC转化的方案

CPLD vs. FPGA

- 名词解释
  - **CPLD：Complex Programmable Logic Device**
  - **FPGA：Field Programmable Gate Array**

- PLD技术的持续发展使两者的对比有所变化
  - **传统CPLD/FPGA的技术对比**
  - **发展中的新CPLD/FPGA概念**

传统CPLD与FPGA技术对比

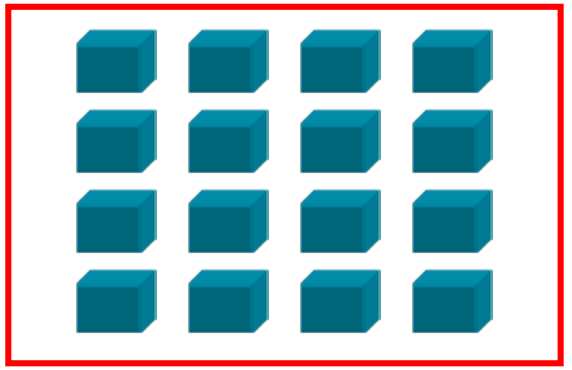
1. 结构对比

CPLD

PAL-like

Macro-cell based

更多组合逻辑资源



FPGA

GAL-like

Logic Element based

更多寄存器和存储器资源

传统CPLD与FPGA技术对比（续）

2. 资源容量对比

CPLD	FPGA
中低规模集成度	中高规模集成度
0.5K-10K logic gates	1K-1000M system gates

3. 性能对比

Predictable timing	Application dependence
芯片引脚间延迟时间可预测	引脚延迟时间受逻辑设计及底层布局影响

传统CPLD与FPGA技术对比（续）

4. 程序易失性对比

CPLD	FPGA
EEPROM or Flash ROM based	SRAM based
程序掉电非易失	程序掉电易失

5. 下载模式对比

单一ISP模式（JTAG）	支持多种配置模式
	程序固化需额外芯片支持

典型CPLD内部构架

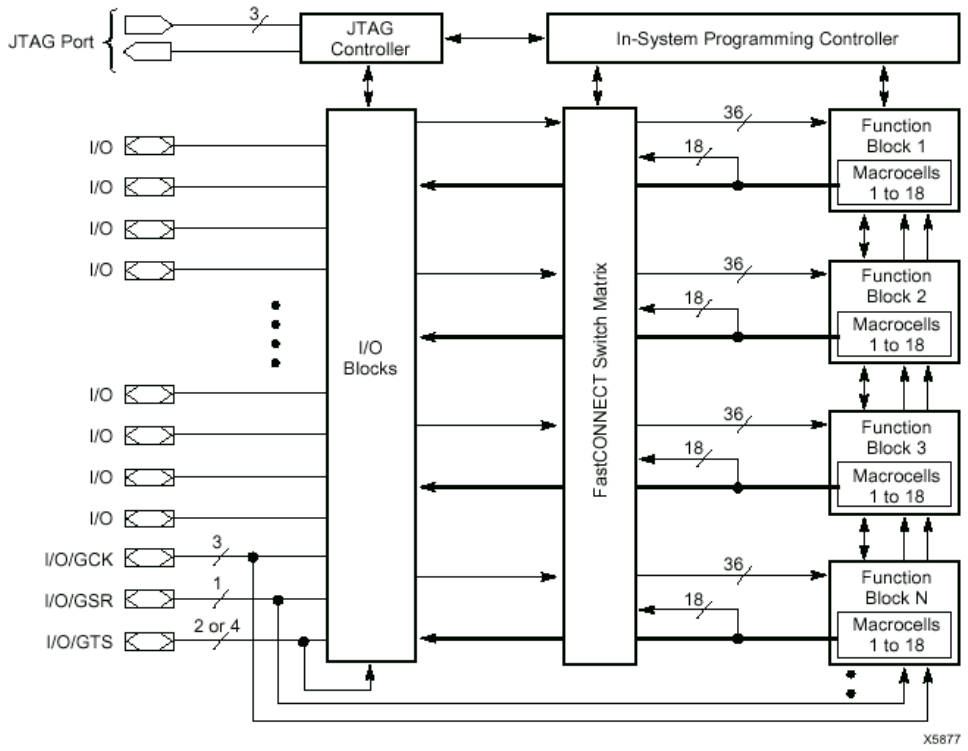


Figure 1: XC9500 Architecture

Note: Function Block outputs (indicated by the bold line) drive the I/O Blocks directly.

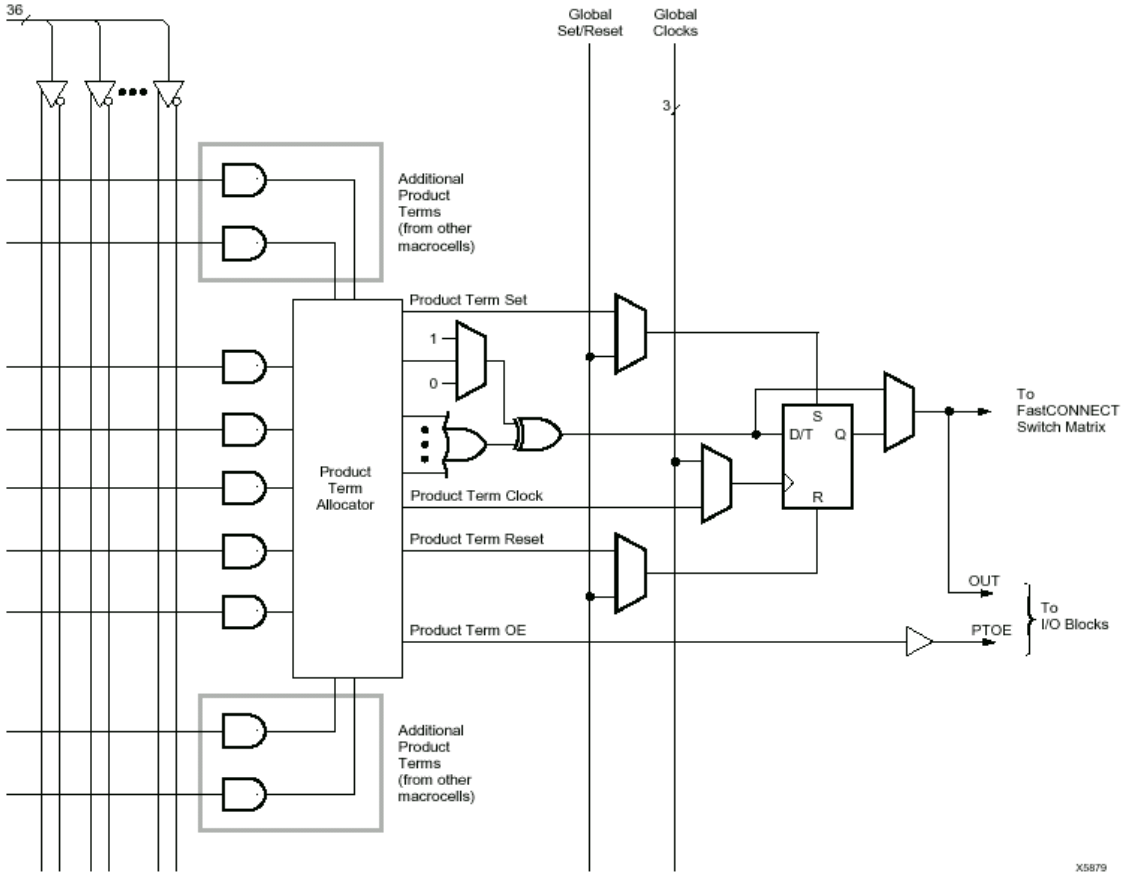


Figure 3: XC9500 Marcocell Within Function Block

典型FPGA内部构架

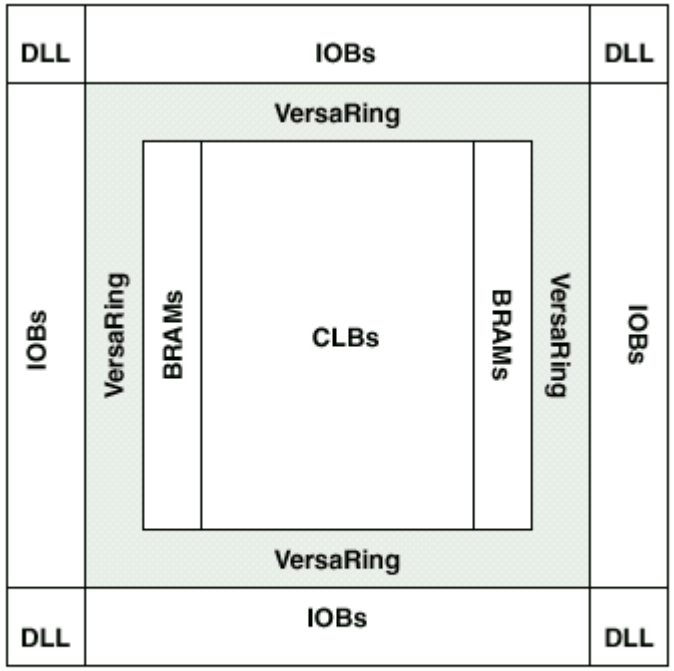


Figure 1: **Virtex Architecture Overview**

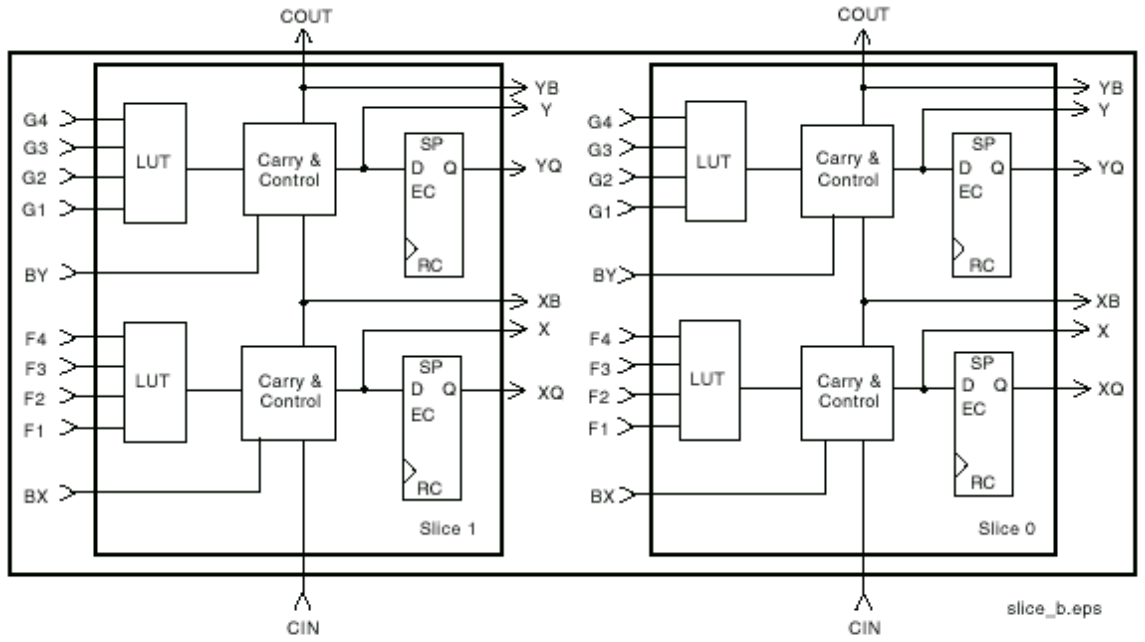


Figure 4: **2-Slice Virtex CLB**

发展中的新CPLD/FPGA概念

- 新CPLD/FPGA产品
  - 新CPLD产品基于LE和非单一总线构架
  - 新FPGA产品具有掉电非易失特性
- 目前区分CPLD/FPGA的主要标准
  - CPLD主要是逻辑资源较少、配置方式单一、具备掉电非易失特性的PLD产品，通常内部仅含传统可编程逻辑资源及少量存储器
  - FPGA在集成大量可编程逻辑资源的基础上通常包含大量PLL、RAM、多功能I/O等资源

Xilinx公司主流7系列FPGA产品

Table 1: **7 Series Families Comparison**

Max. Capability	Spartan-7	Artix-7	Kintex-7	Virtex-7
Logic Cells	102K	215K	478K	1,955K
Block RAM(1)	4.2 Mb	13 Mb	34 Mb	68 Mb
DSP Slices	160	740	1,920	3,600
DSP Performance(2)	176 GMAC/s	929 GMAC/s	2,845 GMAC/s	5,335 GMAC/s
MicroBlaze CPU(3)	260 DMIPs	303 DMIPs	438 DMIPs	441 DMIPs
Transceivers	–	16	32	96
Transceiver Speed	–	6.6 Gb/s	12.5 Gb/s	28.05 Gb/s
Serial Bandwidth	–	211 Gb/s	800 Gb/s	2,784 Gb/s
PCIe Interface	–	x4 Gen2	x8 Gen2	x8 Gen3
Memory Interface	800 Mb/s	1,066 Mb/s	1,866 Mb/s	1,866 Mb/s
I/O Pins	400	500	500	1,200
I/O Voltage	1.2V–3.3V	1.2V–3.3V	1.2V–3.3V	1.2V–3.3V
Package Options	Low-Cost, Wire-Bond	Low-Cost, Wire-Bond, Bare-Die Flip-Chip	Bare-Die Flip-Chip and High-Performance Flip-Chip	Highest Performance Flip-Chip

FPGA芯片选型

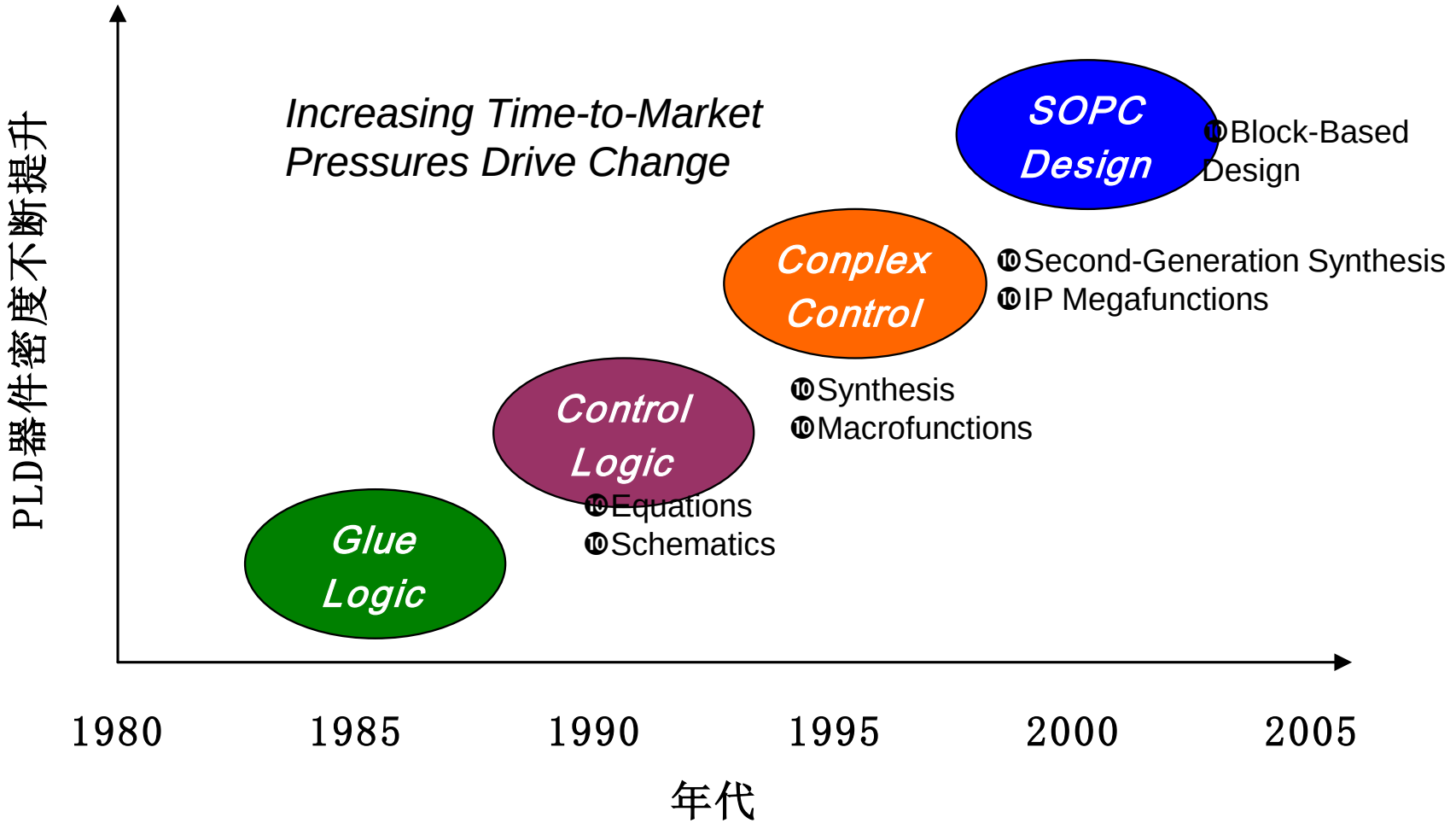
选型考虑因素

- 管脚数量（IO）
- 逻辑资源
- 片内存储器
- DSP资源
- 功耗
- 封装

Table 5: **Artix-7 FPGA Device-Package Combinations and Maximum I/Os**

Package[1]	CPG236		CPG238		CSG324		CSG325		FTG256		SBG484		FGG484[2]		FBG484[2]		FGG676[3]		FBG676[3]		FFG1156	
Size (mm)	10 x 10		10 x 10		15 x 15		15 x 15		17 x 17		19 x 19		23 x 23		23 x 23		27 x 27		27 x 27		35 x 35	
Ball Pitch (mm)	0.5		0.5		0.8		0.8		1.0		0.8		1.0		1.0		1.0		1.0		1.0	
Device	GTP[4]	I/O HR[5]	GTP[4]	I/O HR[5]	GTP[4]	I/O HR[5]	GTP[4]	I/O HR[5]	GTP[4]	I/O HR[5]	GTP	I/O HR[5]	GTP[4]	I/O HR[5]	GTP	I/O HR[5]	GTP[4]	I/O HR[5]	GTP	I/O HR[5]	GTP	I/O HR[5]
XC7A12T			2	112			2	150														
XC7A15T	2	106			0	210	4	150	0	170			4	250								
XC7A25T			2	112			4	150														
XC7A35T	2	106			0	210	4	150	0	170			4	250								
XC7A50T	2	106			0	210	4	150	0	170			4	250								
XC7A75T					0	210			0	170			4	285			8	300				
XC7A100T					0	210			0	170			4	285			8	300				
XC7A200T											4	285			4	285			8	400	16	500

PLD设计方法的演变

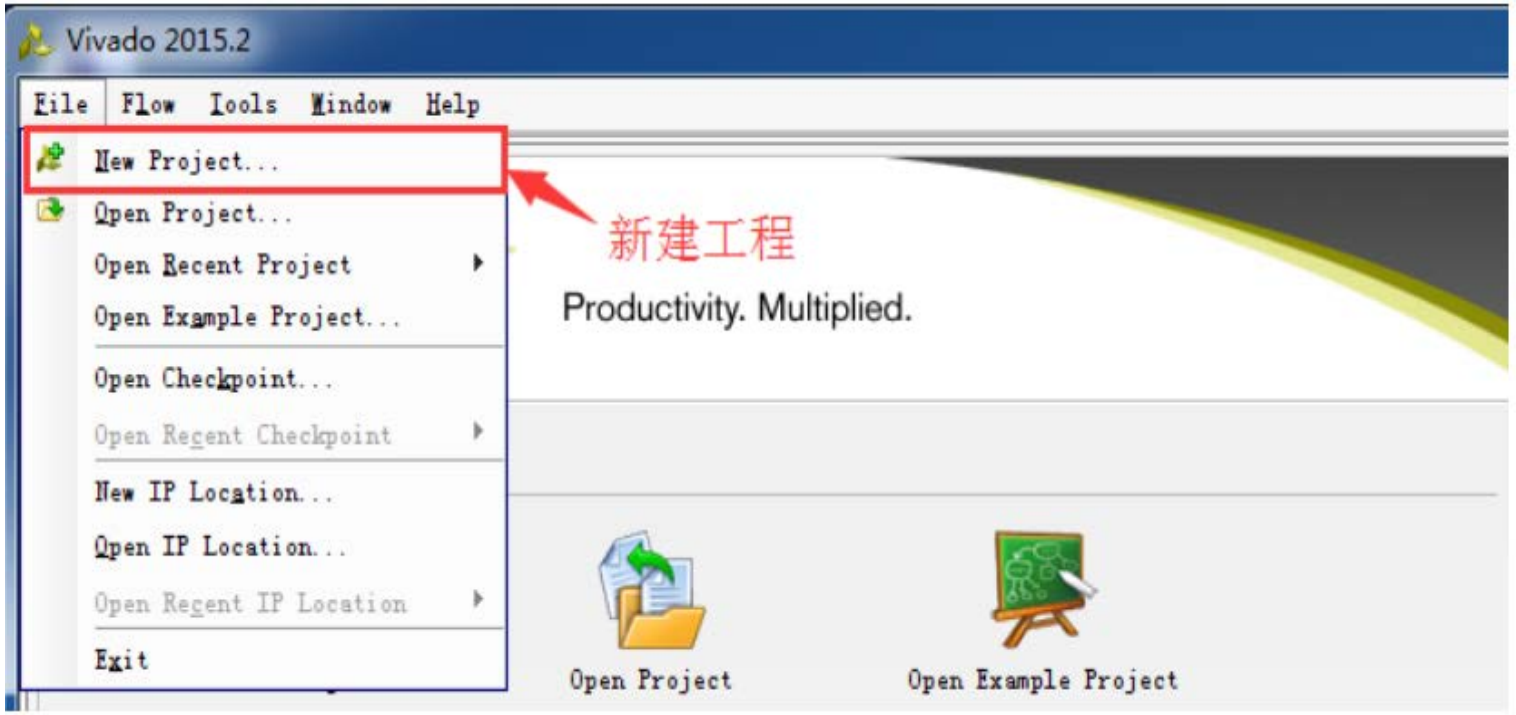

PLD器件密度不断提升
Increasing Time-to-Market
Pressures Drive Change
SOPC Design
Block-Based Design
Conplex Control
Second-Generation Synthesis
IP Megafunctions
Control Logic
Synthesis
Macrofunctions
Equations
Schematics
Glue Logic
1980
1985
1990
1995
2000
2005
年代

Vivado综合开发环境功能介绍

- Vivado是Xilinx公司基于其FPGA产品进行开发设计的综合开发环境，集成文本编辑器、逻辑函数库、编译/布线/仿真工具、下载器等功能；
- 相比较原来的ISE综合开发环境，主要特点在于：
  - 支持Xilinx公司新一代（7系列、28nm以下工艺器件）可编程逻辑器件；
  - 整合高层次综合（HLS）、高级语言编译等功能；
  - 支持IP库自定义封装；
  - 增量式工程变更管理，编译速度提升。

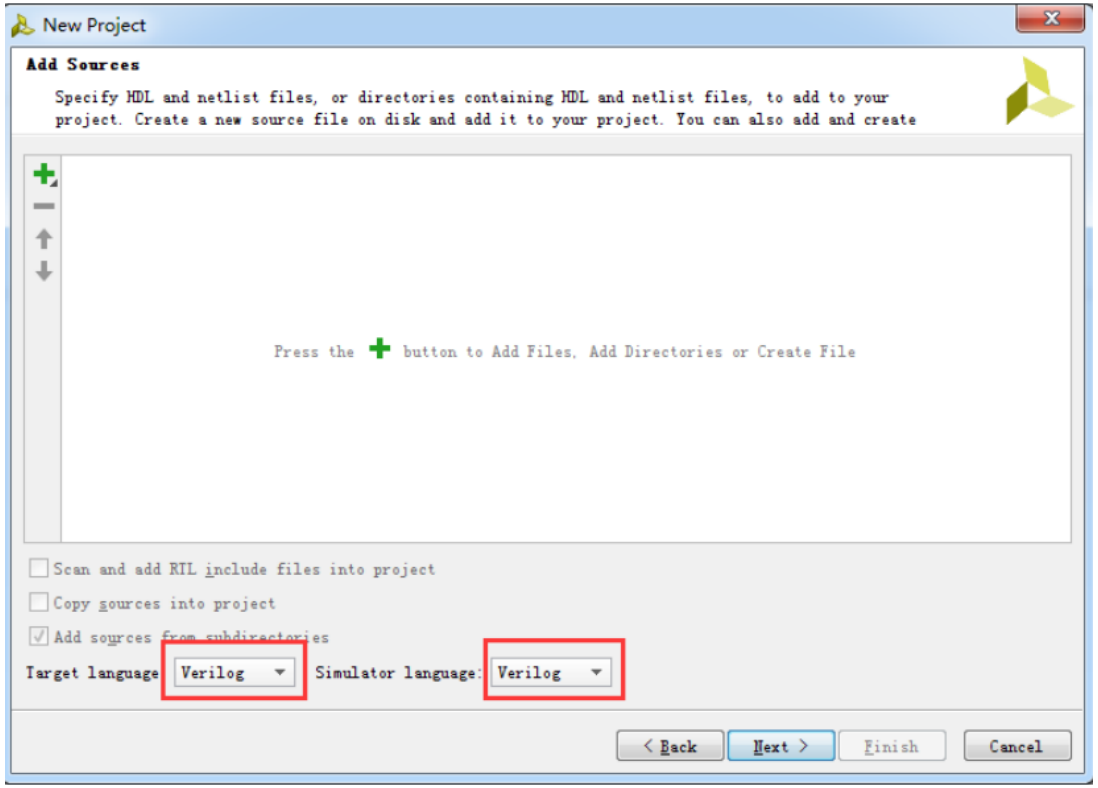
Vivado开发环境及流程演示

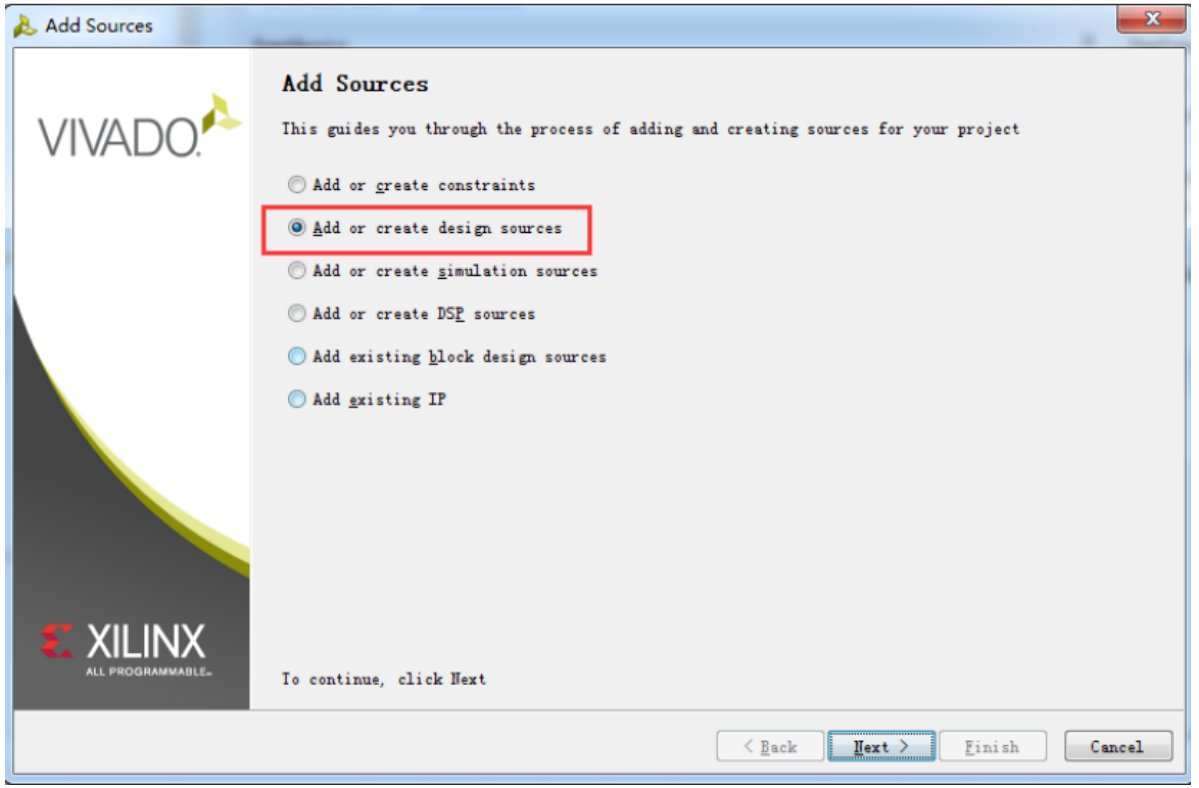
一、启动Vivado，新建工程



Vivado开发环境及流程演示

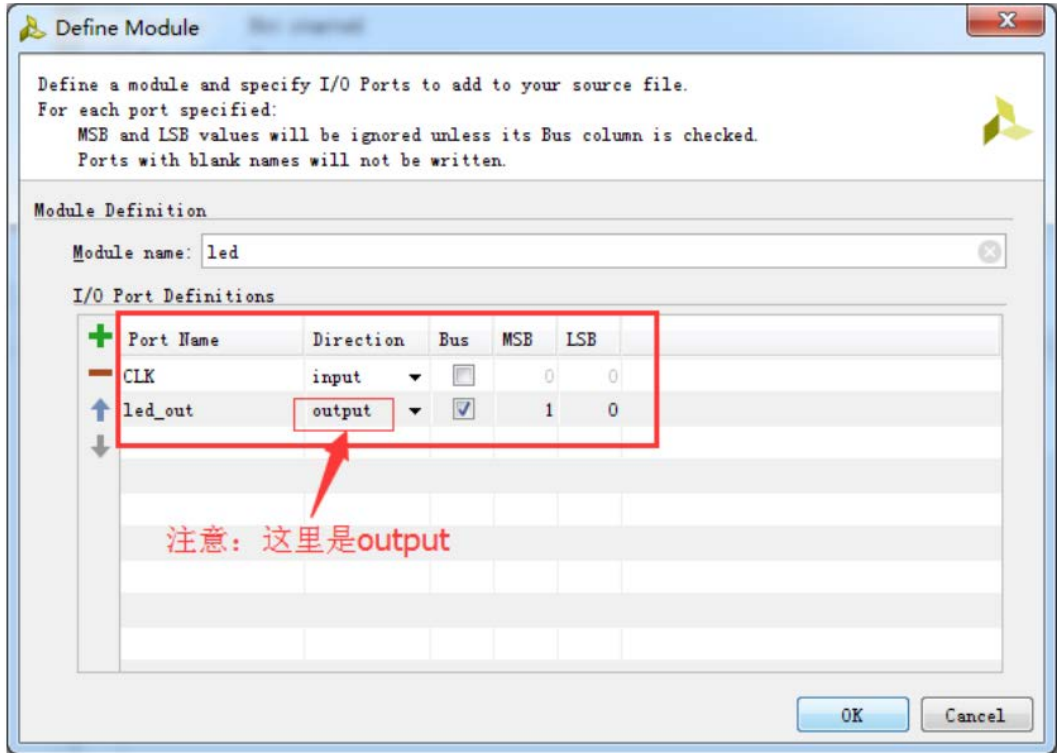
二、利用向导，新建项目，语言选择为Verilog，器件选择与平台相关的，新建Verilog HDL 文件

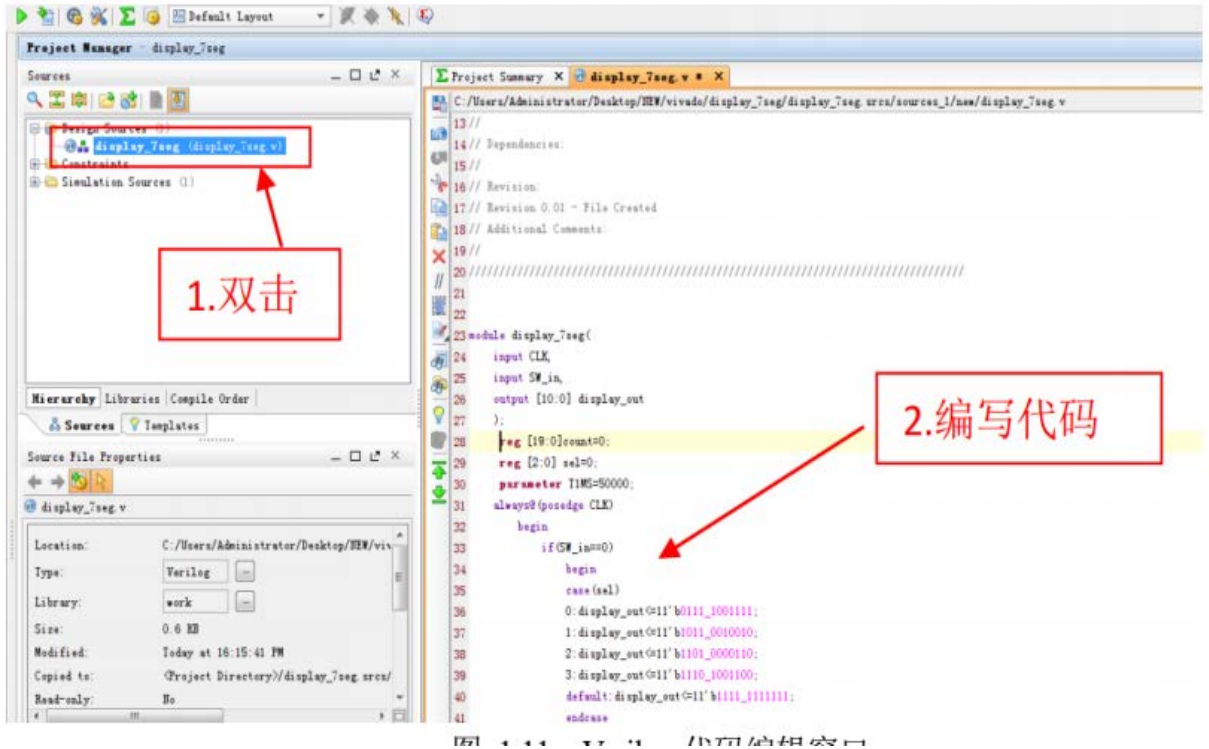




Vivado开发环境及流程演示

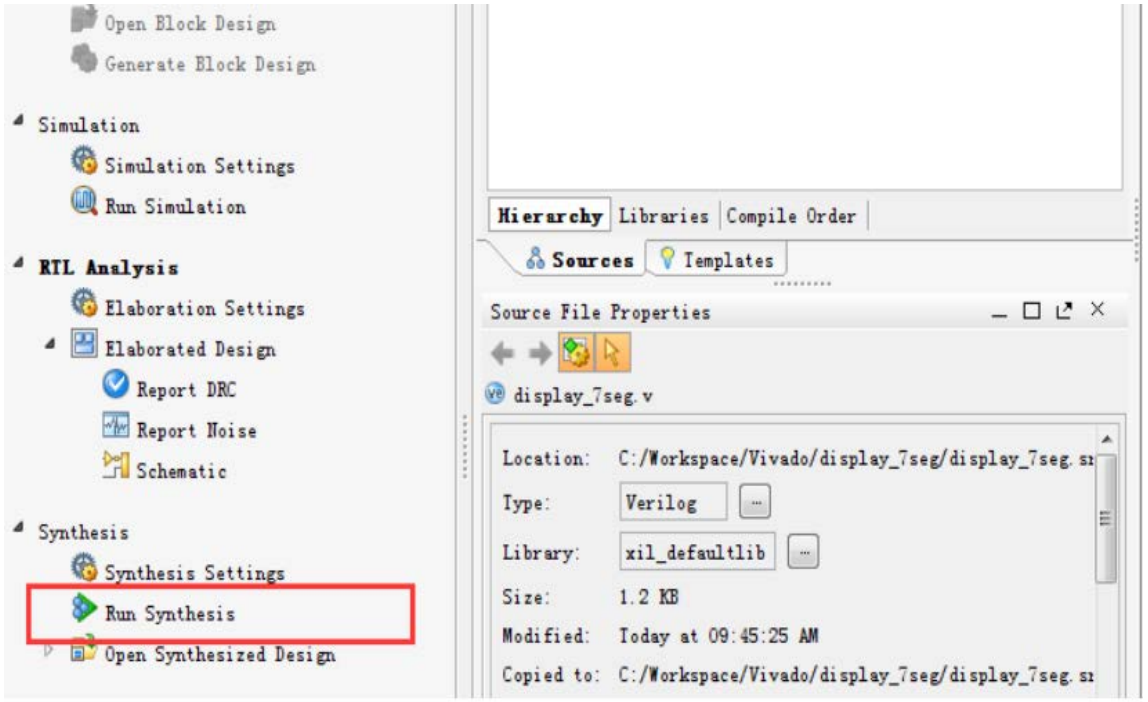
三、填写模块名称和端口，文件窗口区编写代码

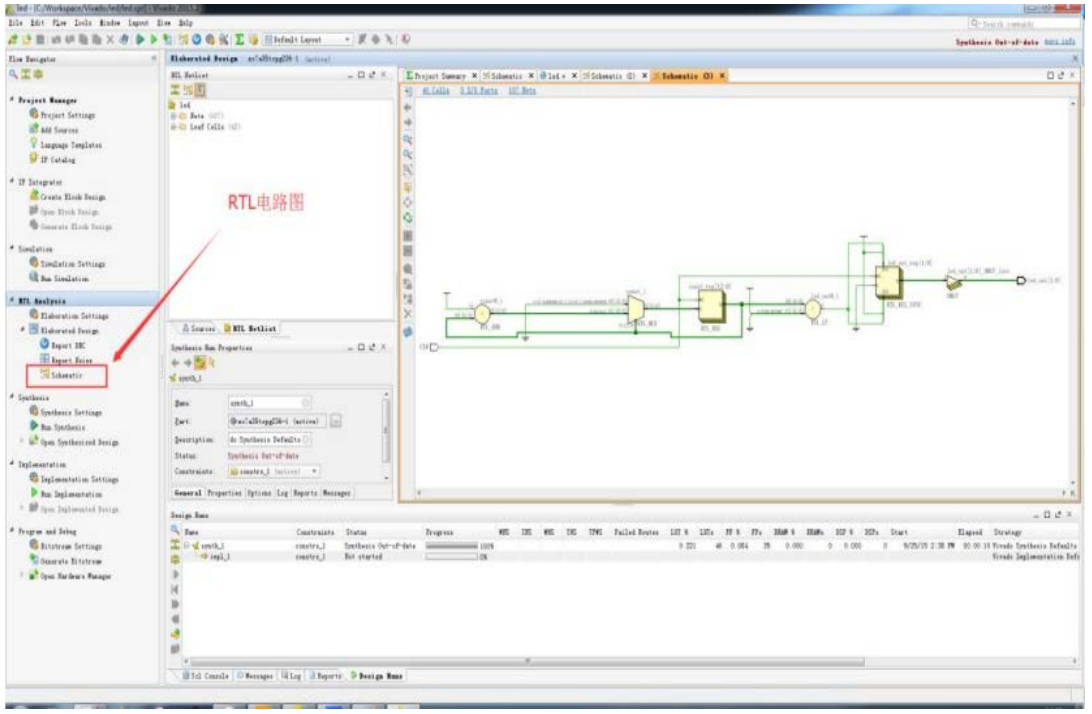




Vivado开发环境及流程演示

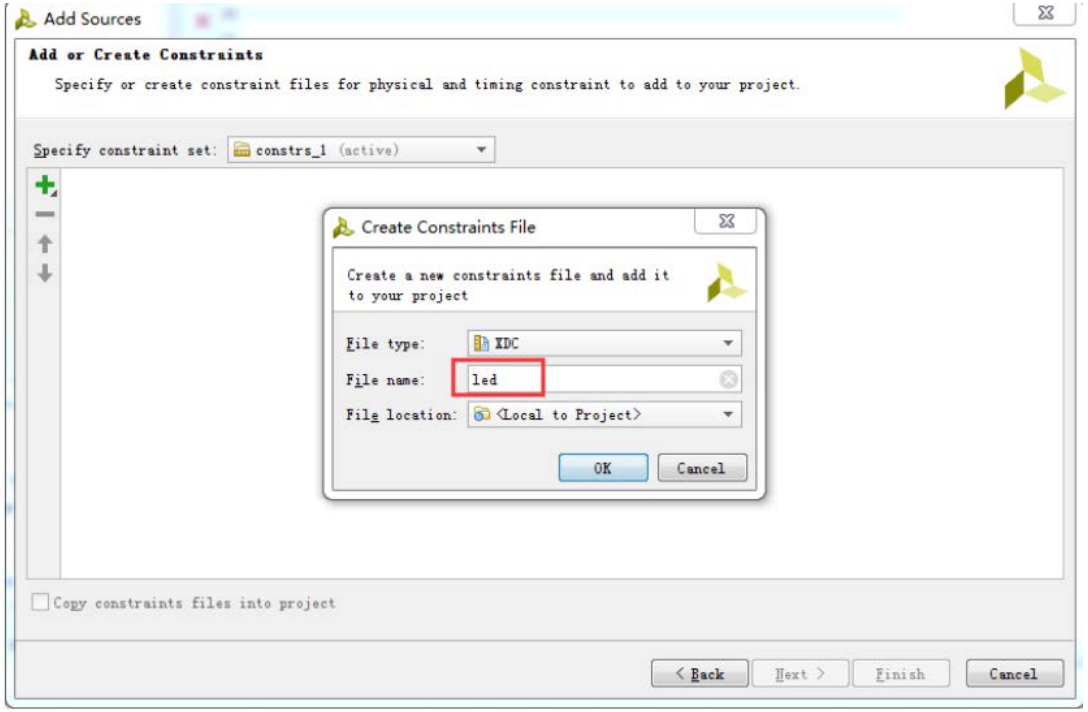
四、Vivado程序编译，查看RTL级电路

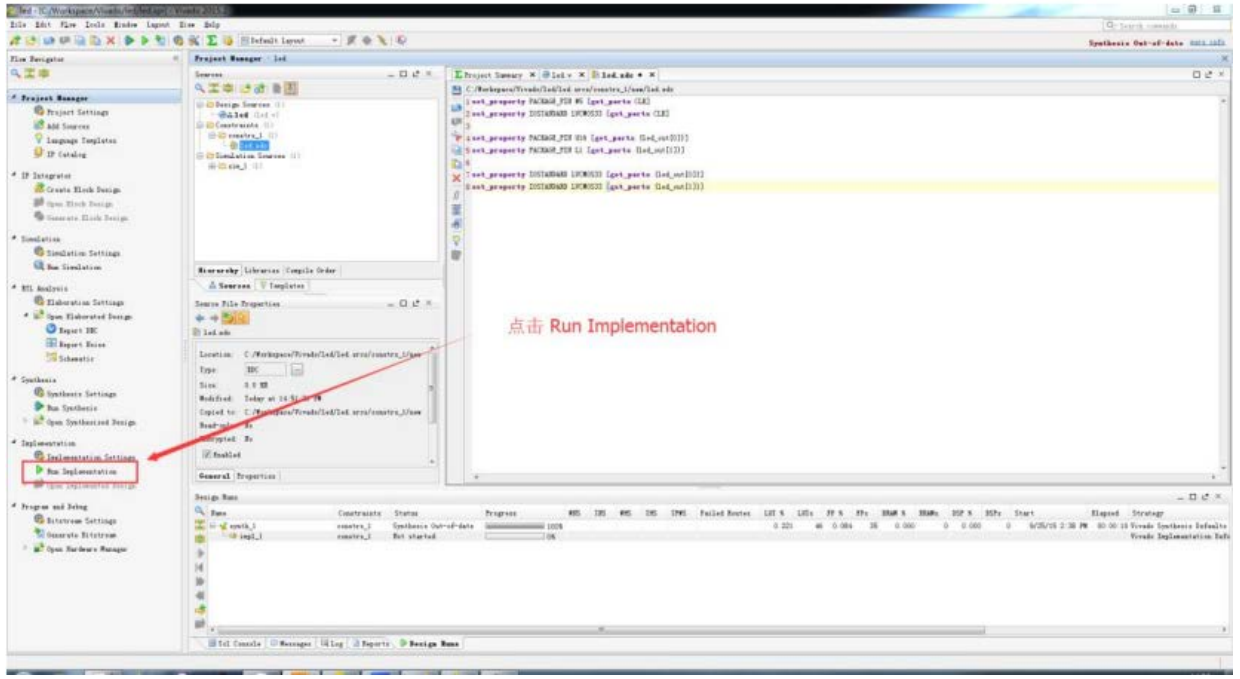




Vivado开发环境及流程演示

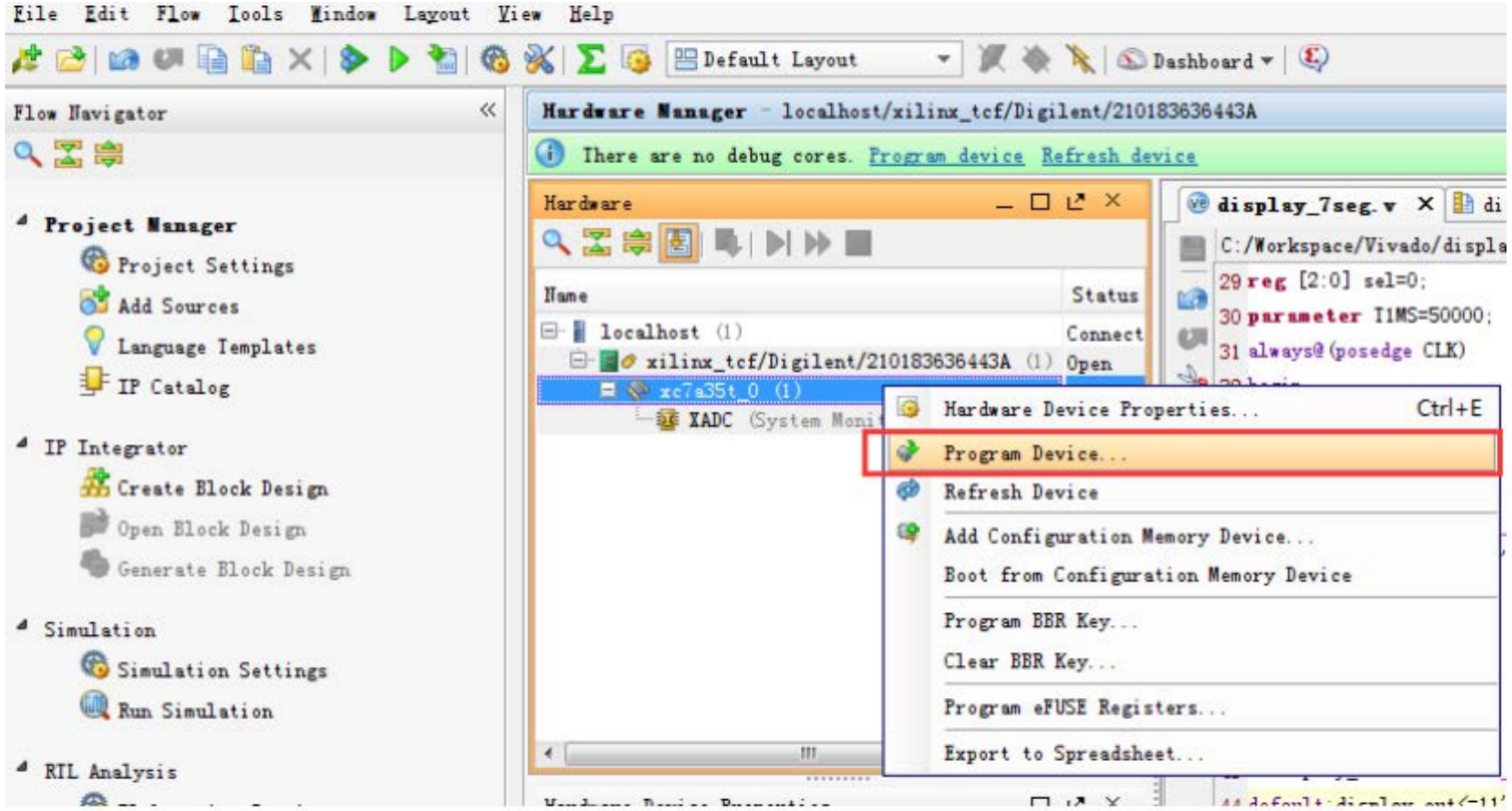
五、分配引脚，运行代码





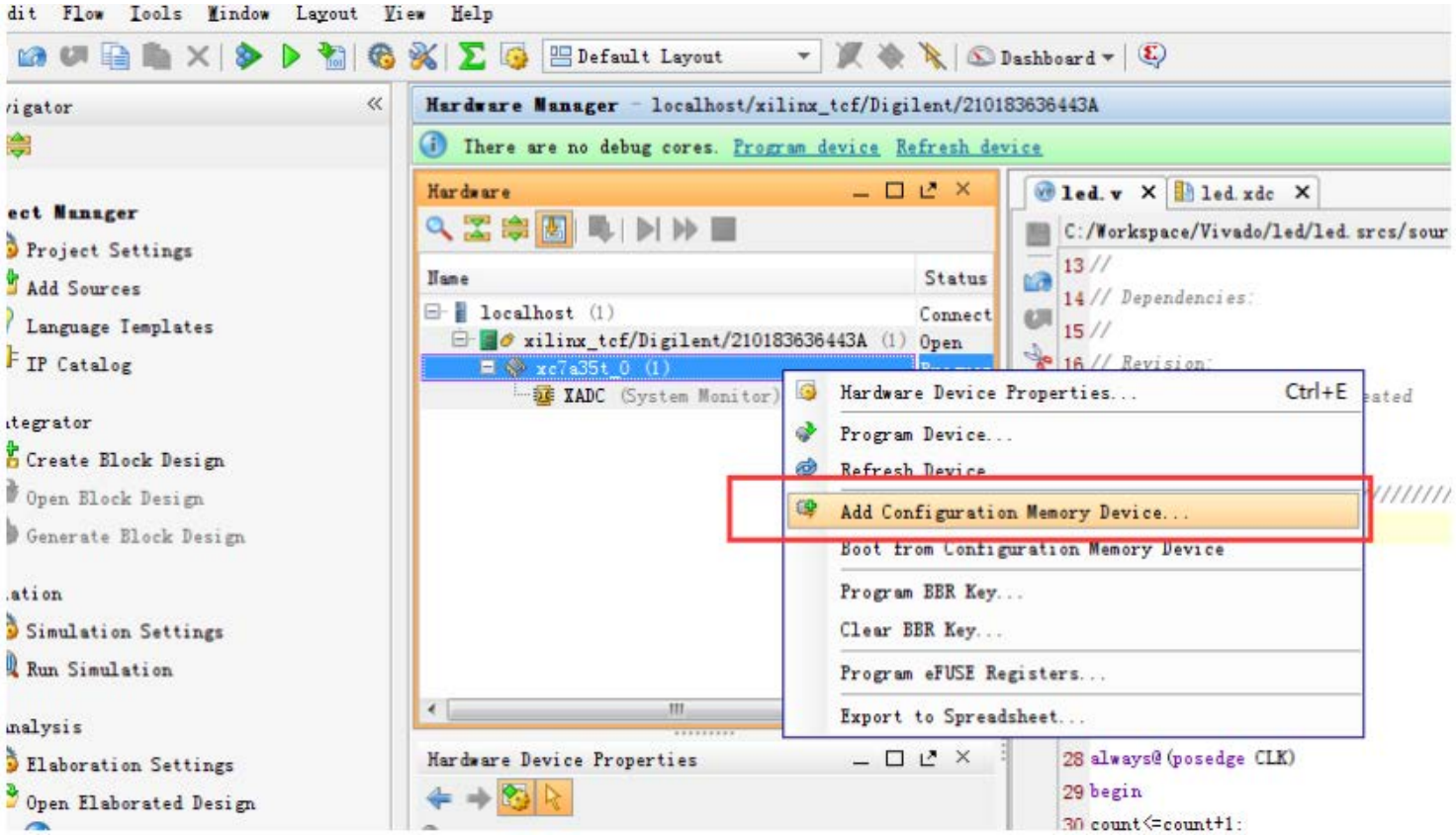
Vivado开发环境及流程演示

六、连接电脑下载演示



Vivado开发环境及流程演示

七、ROM烧写，掉电不丢失



习题或练习

- 自主完成Vivado下工程建立、输入、编译、仿真和下载流程操作。

- **例一：2输入与非门电路**
- **例二：时钟分频器电路**

XILINX

UNIVERSITY PROGRAM

xup@xilinx.com