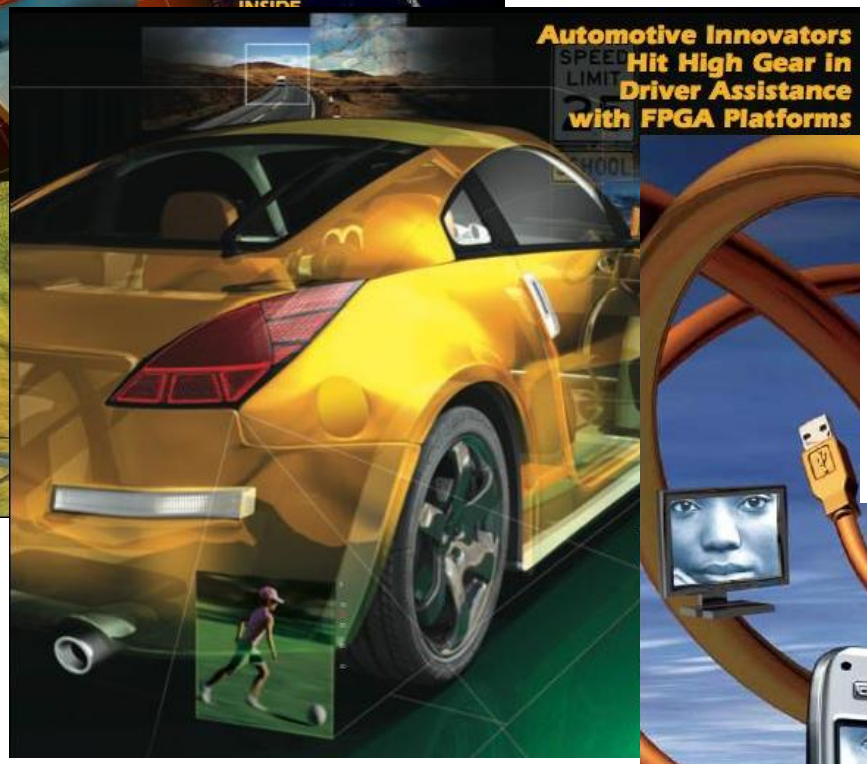
Verilog HDL

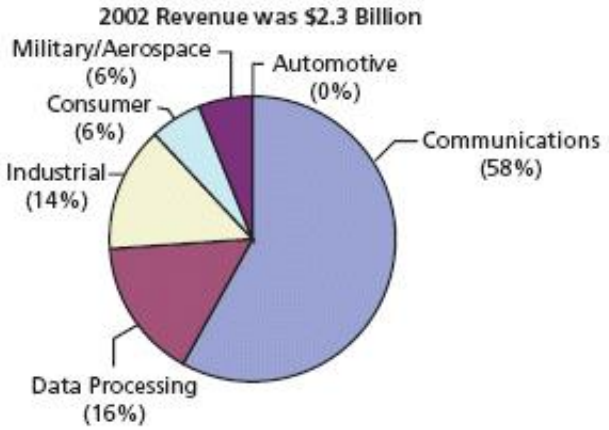
数字系统设计

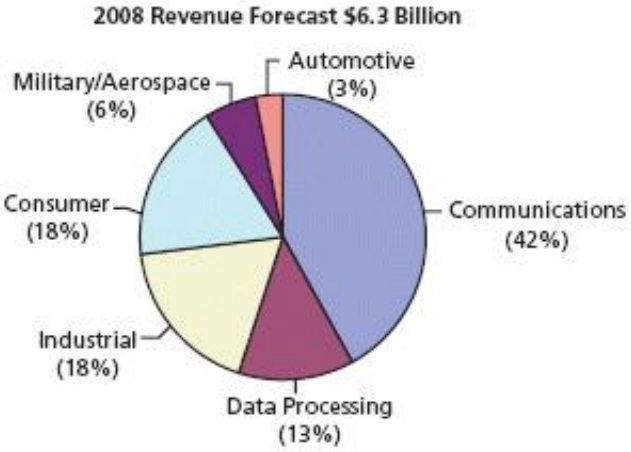
宦　　飞

E-mail: huanfei@sjtu.edu.cn

FPGA目前已广泛应用





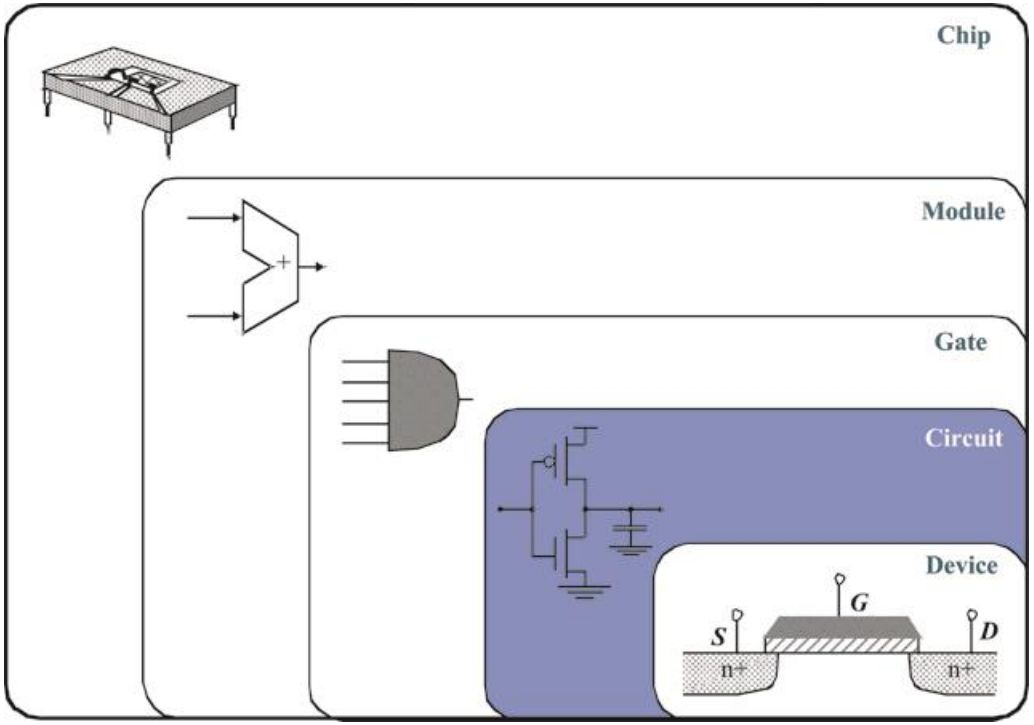


硬件描述语言是基于FPGA 设计数字系统的基础

- 使用HDL设计开发数字系统
  - 设计语言的标准化、开发工具的通用
  - 设计过程几乎与所用的CPLD/FPGA器件的硬件结构没有关系
  - 设计成功的逻辑功能单元有很好的可移植性
- 目前已成为IEEE标准的HDL
  - **Verilog HDL/SystemVerilog、VHDL、SystemC**
- HDL基本功能 ——对硬件逻辑电路的功能进行描述
  - 描述电路基本器件的连接
  - 描述电路的功能
  - 描述电路的时序
  - 描述电路并行处理
  - 在不同抽象层次上描述电路

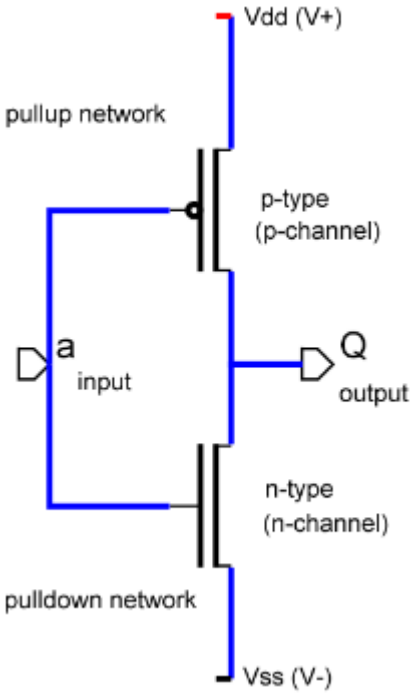
数字系统设计的抽象层次

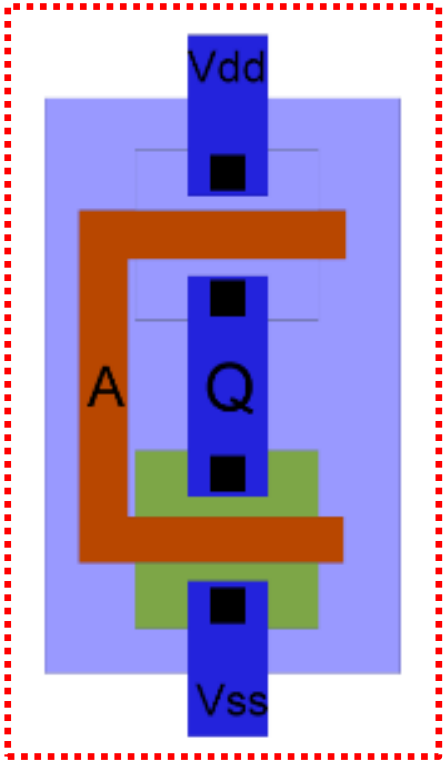
- 按抽象程度——由低到高增加
  - 器件、电路、门、功能模块、芯片和系统
  - 器件层是最低的层次，而系统级是最高的层次
- 可以在任何一个层次描述设计
  - 自上而下进行设计时，设计就逐步接近物理实现
  - 在表示上就更少一些抽象、涉及的细节会逐渐增加
- 抽象层次可以在结构域和行为域中表示
  - 结构——一个部件通过一些基本部件的互连来描述。
  - 行为——一个部件通过定义它的输入/输出响应来描述。

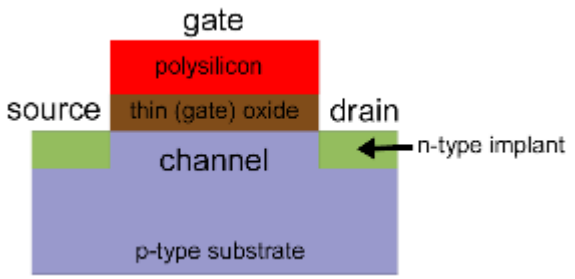


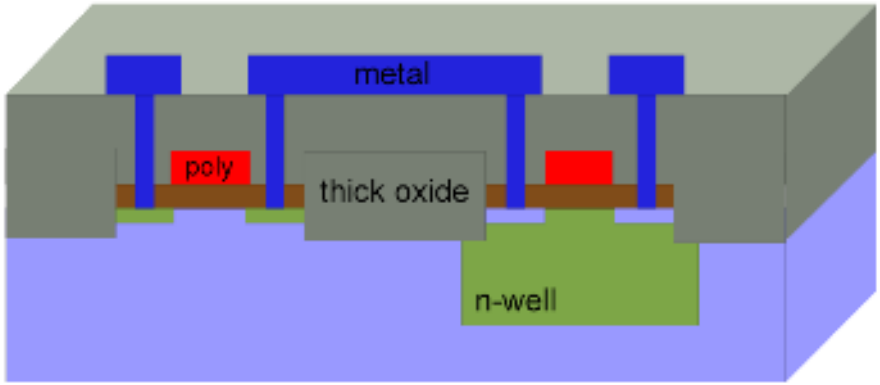
设计的抽象层次——器件级

- 器件级——最低层
  - 硅片级
  - 设计的基本单元是代表扩散区、多晶硅和硅片表面金属层的几何形状
  - 使用这些图案和其互连关系为器件制造过程建立模型
  - 行为描述 —— 使用电子材料中电子和空穴迁移的物理公式
- 例、反相器





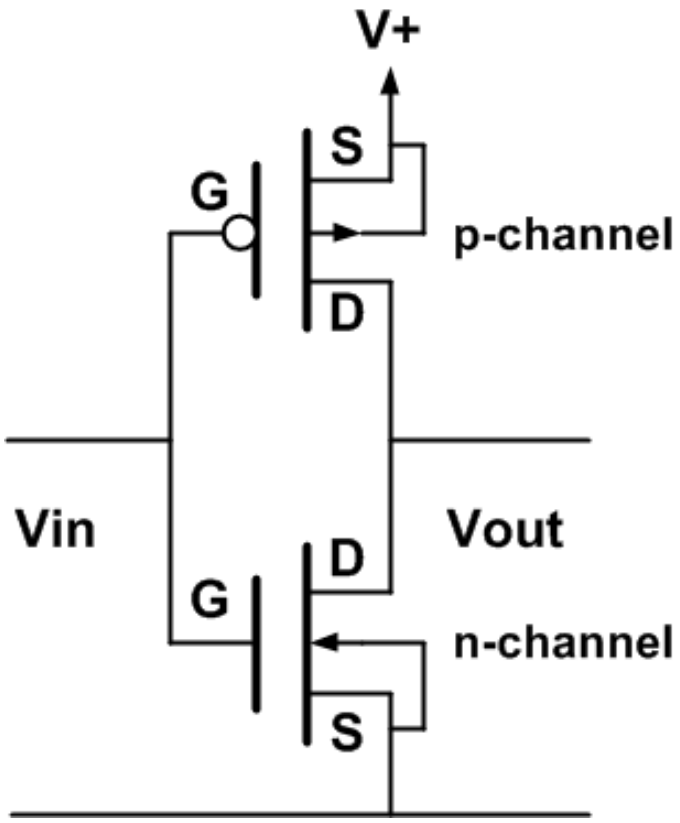




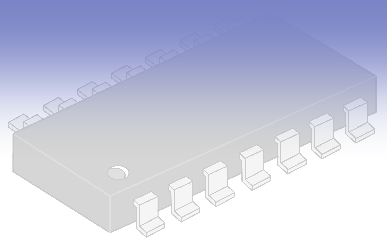
设计的抽象层次——电路级

电路级

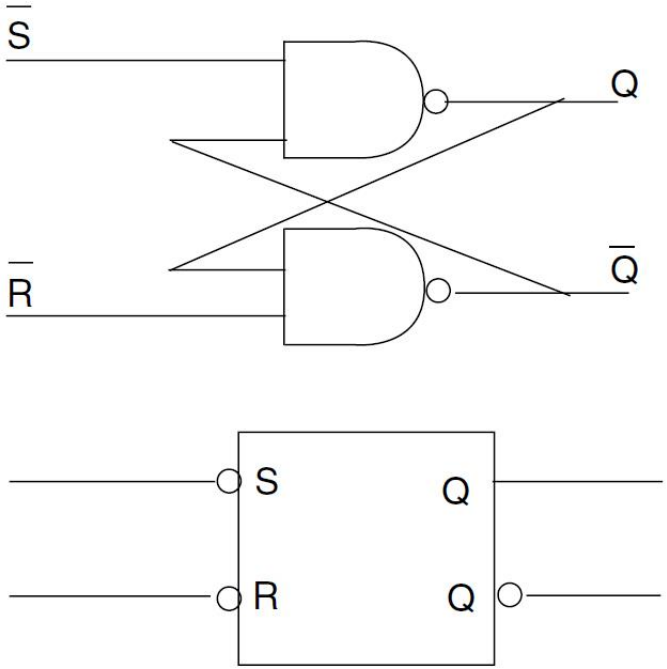
- 设计表示是传统无源和有源电子电路元件的互连
- 电路元件包括 —— 电阻、电容、二极管和MOS晶体管等
- 结构描绘——利用元件的互连表示
  - 以电压和电流的形式模拟电路的行为
- 行为描述——通过**微分方程**表示



设计的抽象层次——门级

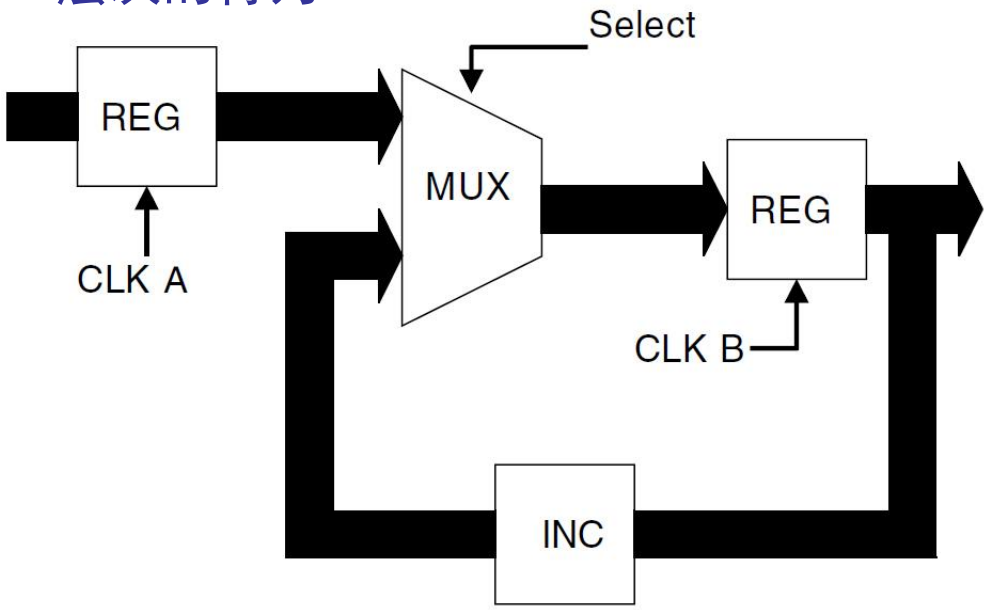


- 门级是传统数字器件的主要设计层次
- 基本部件包括
  - 与、或、异或、反相操作门和各种不同类型的触发器
  - 这些单元的互连构成组合和时序逻辑电路
  - 行为描述——布尔方程式

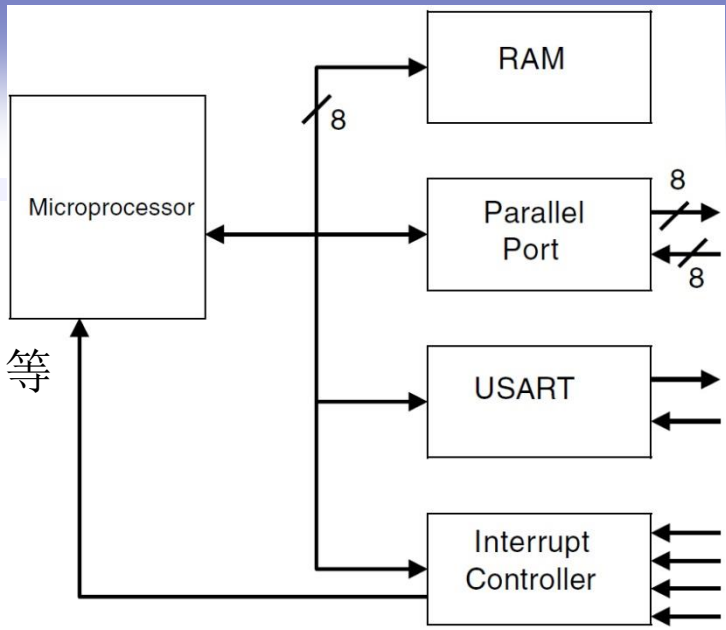


设计的抽象层次——寄存器传输级

- 寄存器传输级（RTL， Register transfer level）
- 基本部件包括
  - 寄存器、计数器、多路复用器和ALU
  - 基本部件也称为功能块
    - 对应着 VLSI 设计的宏
  - 寄存器级也称为功能级或宏级
  - 寄存器级的基本部件
    - 可以用互连的门来表示 —— 一般并不采用这种方法
- 采用真值表和状态表描述这一层次的行为



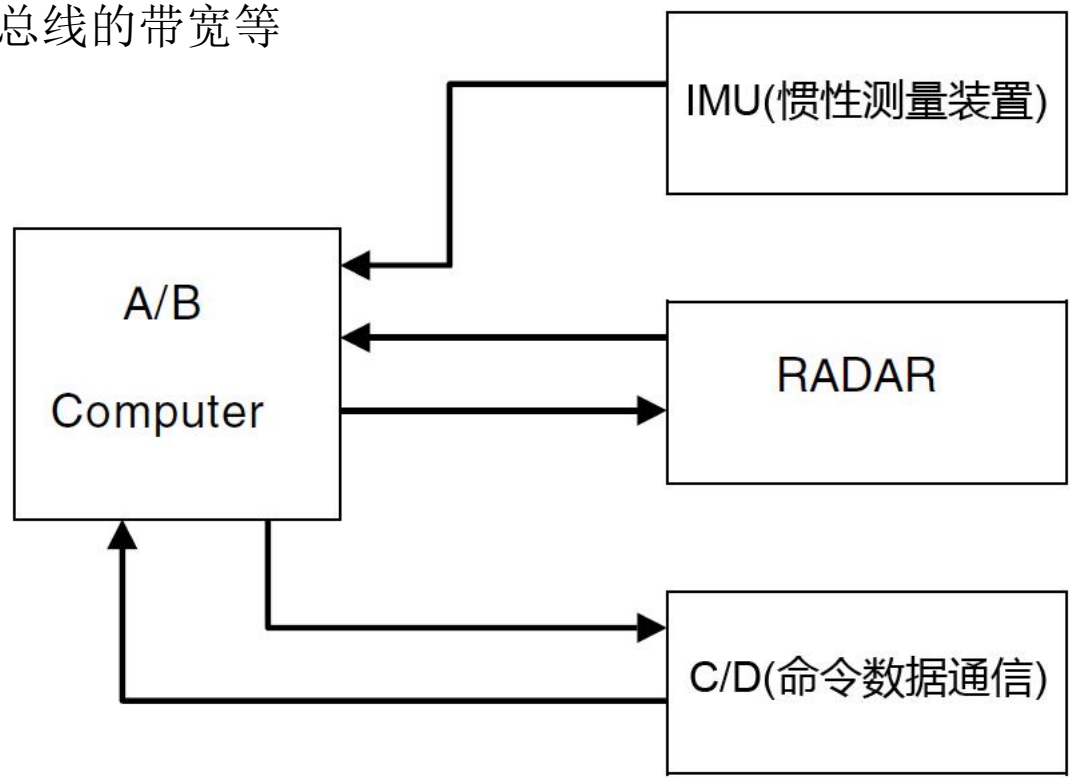
设计的抽象层次——芯片级



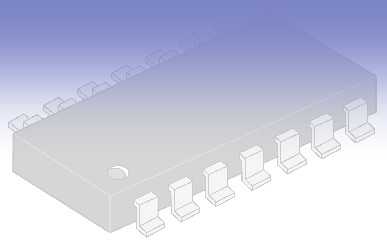
- 芯片级基本部件
  - 微处理器、存储器、串口、并口和中断控制器，等
  - 一般芯片的**边界**一般就是模型的边界
  - 也有例外，如
    - 对一个由多个芯片构成的子系统建立模型
    - 对一个芯片中的各个组成部分分别建模
- 芯片级建模描述内容 —— 基本部件的 I/O 响应，芯片实现的算法
  - 关键在于对从输入到输出的数据通道（Datapath）进行建模
    - **datapath —— a collection of functional units**
- 这一级的行为域描述
  - **基本部件不是用更基本的部件构造和连接描述，而是用行为来描述**
  - 例如、对一个串行 I/O 端口(UART)建模
    - 一般不是用寄存器、计数器等更简单的模块连接而成
    - 而是将UART本身作为一个基本实体建模——逻辑功能
- 复杂电路的芯片级模型被当作知识产权（ Intellectual Property，IP）
  - 常常由一个公司出售给另一个公司 —— 使用**IP**设计复杂的系统

设计的抽象层次——系统级

- 系统级的基本部件
  - 计算机、总线接口部件、磁盘部件、雷达部件，等
- 行为描述内容
  - 用性能规格说明表示，如
    - 处理器的MIPS速率、总线的带宽等



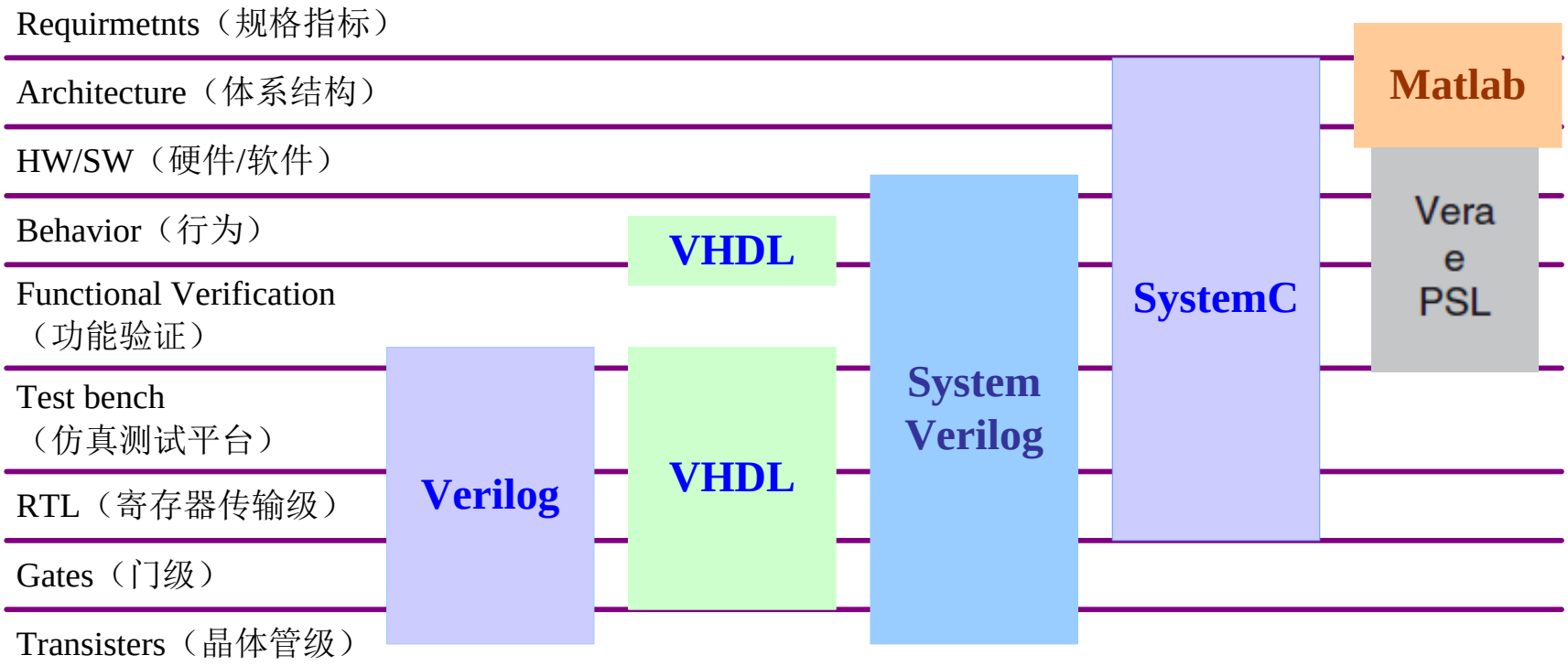
不同抽象层次的设计描述方法总结



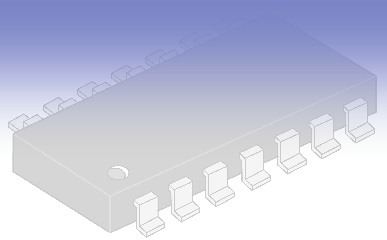
- 系统级
  - 电子设备 —— PC，示波器，MP3，电视机
  - 描述方法 —— 性能规格说明
- 芯片级
  - 基本部件 —— CPU、DSP、MCU、RAM
  - 描述方法 —— 行为（算法、数据流）描述
- 寄存器（功能模块）级（RTL，Register transfer level）
  - 基本部件 —— 寄存器、ALU、计数器、多路复用器、ROM
  - 描述方法 —— 行为（数据流）描述
- 门
  - 基本部件 —— AND、OR、NOT、XOR、各种触发器
  - 描述方法 —— 布尔方程（结构描述）
- 电路
  - 基本部件 —— 晶体管、电阻、电容、电感
  - 描述方法 —— 微分方程（PSpice电路网表）
- 器件
  - 硅结构版图

目前最主要的几种设计语言的比较

- Verilog HDL/SystemVerilog、VHDL、SystemC
  - IEEE Std
- **Matlab**
- **C\C++ —— 基于HLS（高层次综合）设计技术**
  - Vera, e, PSL(Property Specification Language) ——专用于验证的语言

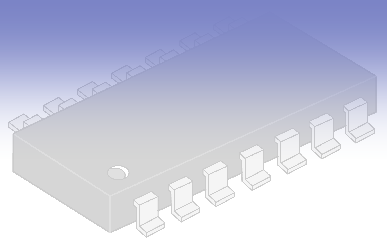


使用 Verilog 进行数字系统的结构描述



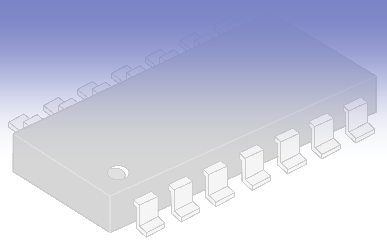
- 结构描述 —— 通过一些基本部件的互连来描述
- 使用基本元件的原语表示组合逻辑
  - 原语（primitive）—— 预先定义的常用功能模型
- 建立MOS器件的模型——开关级建模
  - 基本MOS开关原语：NMOS，PMOS，CMOS
  - 双向传输开关原语，等
- 能够建立精确的信号模型
  - 延时
  - 强度

使用 Verilog 进行数字系统的行为描述



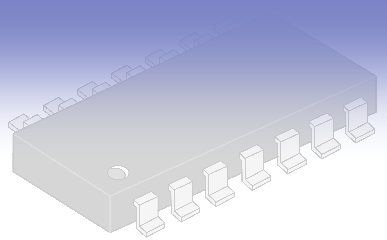
- 行为描述 ——通过定义它的输入/输出响应来描述
- 描述电路的并发执行
  - 使用延时、事件精确地控制过程执行
  - 通过命名事件控制其它过程中的激活/停止行为
- 程序结构
  - 函数（function）
  - 任务（task）
- 过程控制结构
  - 描述顺序执行的过程（procedure）
  - 条件转移
  - 循环
  - 生成快
- 用于建立各种表达式运算符算术
  - 逻辑
  - 位

Verilog 和 VHDL 的比较



- 两种主要的HDL 工业标准（IEEE Std.）
  - VHDL：1987年
    - VHSIC Hardware Description Language
    - VHSIC —— Very High Speed Integrated Circuits
  - Verilog HDL：1995年
- 支持电路的结构、RTL级和行为的描述
- 能够仿真和验证电路
- 支持电路设计的综合（Synthesis）
- 两种语言均已支持数字/模拟电路的描述
  - 可用于混合信号电路的设计

Verilog 和 VHDL 的比较



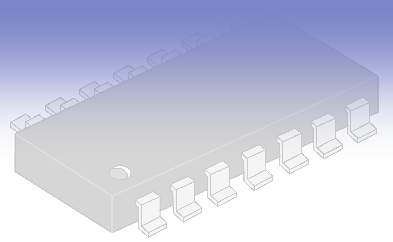
Verilog 的特点

- 编程灵活
  - 类C语言的编程风格
- 结构描述的性能优越
  - 易于电路结构描述
  - 能够描述晶体管开关特性
- 仿真性能非常好
  - Verilog HDL 包含有PLI（C语言编程语言接口）
- 最新标准：
  - Verilog 2005
  - （IEEE Standard 1364-2005，7 April 2006 ）
  - 2001标准的局部修订

VHDL 的特点

- 编程规范、严谨、易学
  - 类似 Ada 编程风格
- 在设计中，一般不容易发生的歧义
- 行为描述的能力更强
  - 利于算法实现
  - 如：浮点数算术运算
- 最新标准
  - VHDL 2019
  - 在 VHDL 2008 的基础上修改

Verilog 和 VHDL 的比较



全加器——Verilog

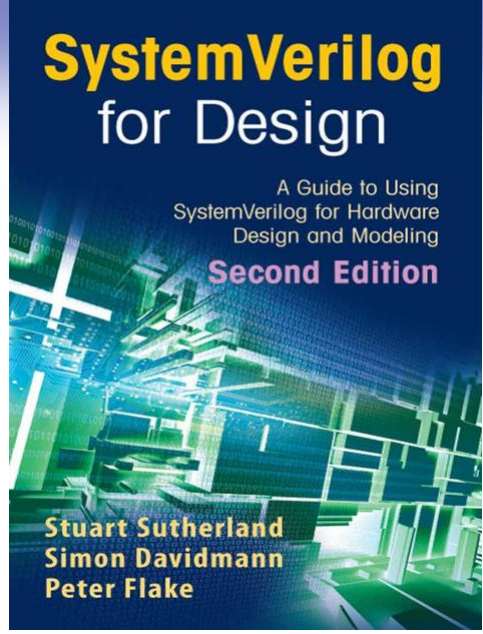
```
module adder #(parameter N=8)
    ( input aclr, clk,
      input [N-1:0] a, b,
      output [N: 0] q );

    reg [N:0] q_s;
    assign q = q_s;

    always @(posedge clk or posedge aclr )
    begin
        if ( aclr == 1'b1 )
            q_s <= 0;
        else
            q_s <= a + b;
    end
endmodule
```

全加器——VHDL

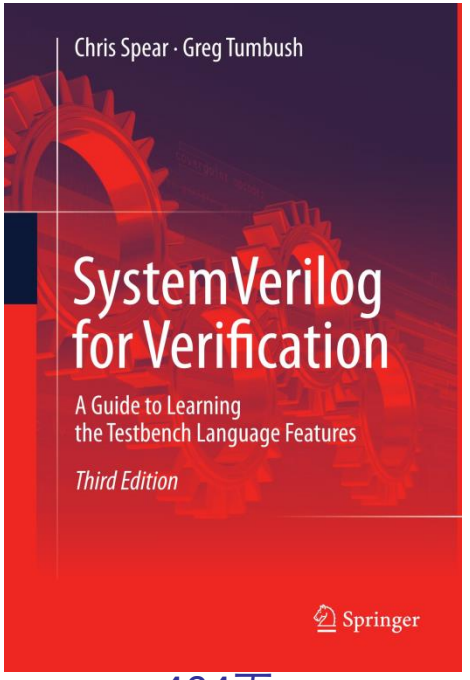
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity signed_adder is
    port  ( aclr : in   std_logic;
            clk  : in   std_logic;
            a    : in   std_logic_vector;
            b    : in   std_logic_vector;
            q    : out  std_logic_vector  );
end signed_adder;
architecture signed_adder_arch of signed_adder is
    -- extra bit wide
    signal q_s : signed(a'high+1 downto 0);

begin -- architecture
    q <= std_logic_vector(q_s);

    adding_proc:  process (aclr, clk)
    begin
        if (aclr = '1') then
            q_s <= (others => '0');
        elsif rising_edge(clk) then
            q_s <= ('0'&signed(a)) + ('0'&signed(b));
        end if; -- clk'd
    end process;
end signed adder arch;
```

硬件描述语言标准的最新进展

- Verilog ——IEEE Std. 1364-2005
- SystemVerilog —— IEEE Std. 1800-2017
  - Verilog- 2005 Std. 扩展 —— Verilog 超集
  - HDVL —— Hardware Design and Verification Language（硬件描述和验证语言）
  - ≈(Verilog + VHDL + **C/C++** + e + Vera)
- VHDL —— IEEE Std. 1076-2019
- SystemC —— IEEE Std. 1666-2011
  - 最初于 2005年 12月14日成为 IEEE Std. 1666
  - 标准 C++ 扩展
- 验证语言
  - Vera —— Synopsys
  - **e**、PSL（Property Specification Language）—— IEEE Std.



418页



464页

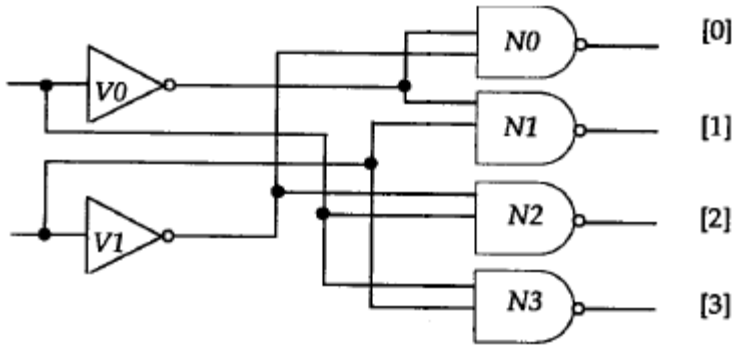
HDL 设计举例 —— 2 到 4 译码器

- Verilog

```
2   module decoder2x4( output reg [3:0] y, input [1:0] sel );
3
4   always @(*)
5   begin
6       case ( sel )
7           0:    y = 1;
8           1:    y = 2;
9           2:    y = 4;
10          3:    y = 8;
11          default:  y = 4'bx;
12      endcase
13  end
14  endmodule
```

- SystemVerilog

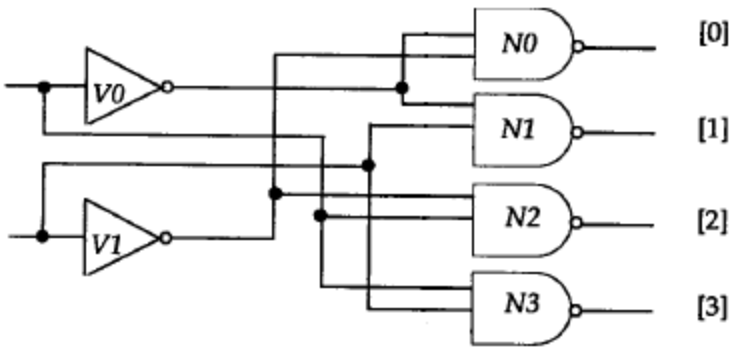
```
2   module decoder2x4( output logic [3:0] y, input logic [1:0] sel );
3
4   always_comb
5   begin
6       case ( sel )
7           0:    y = 1;
8           1:    y = 2;
9           2:    y = 4;
10          3:    y = 8;
11          default:  y = 'x;
12      endcase
13  end
14  endmodule
15
```



HDL 设计举例—— VHDL

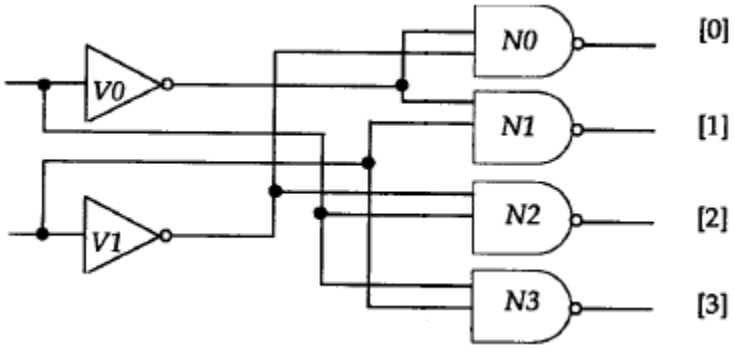
- 2 到 4 译码器

```
 2    library ieee;
 3    use ieee.std_logic_1164.all;
 4
 5    entity decoder2x4 is
 6        port (   y: out std_logic_vector ( 3 downto 0 );
 7                 sel: in std_logic_vector ( 1 downto 0)
 8                );
 9    end entity decoder2x4;
10
11    architecture dataflow of decoder2x4 is
12    begin
13        y <= "0001" when sel = "00" else
14             "0010" when sel = "01" else
15             "0100" when sel = "10" else
16             "1000" when sel = "11" else
17             ( others => 'X' );
18    end architecture dataflow;
19
```



HDL 设计举例—— SystemC

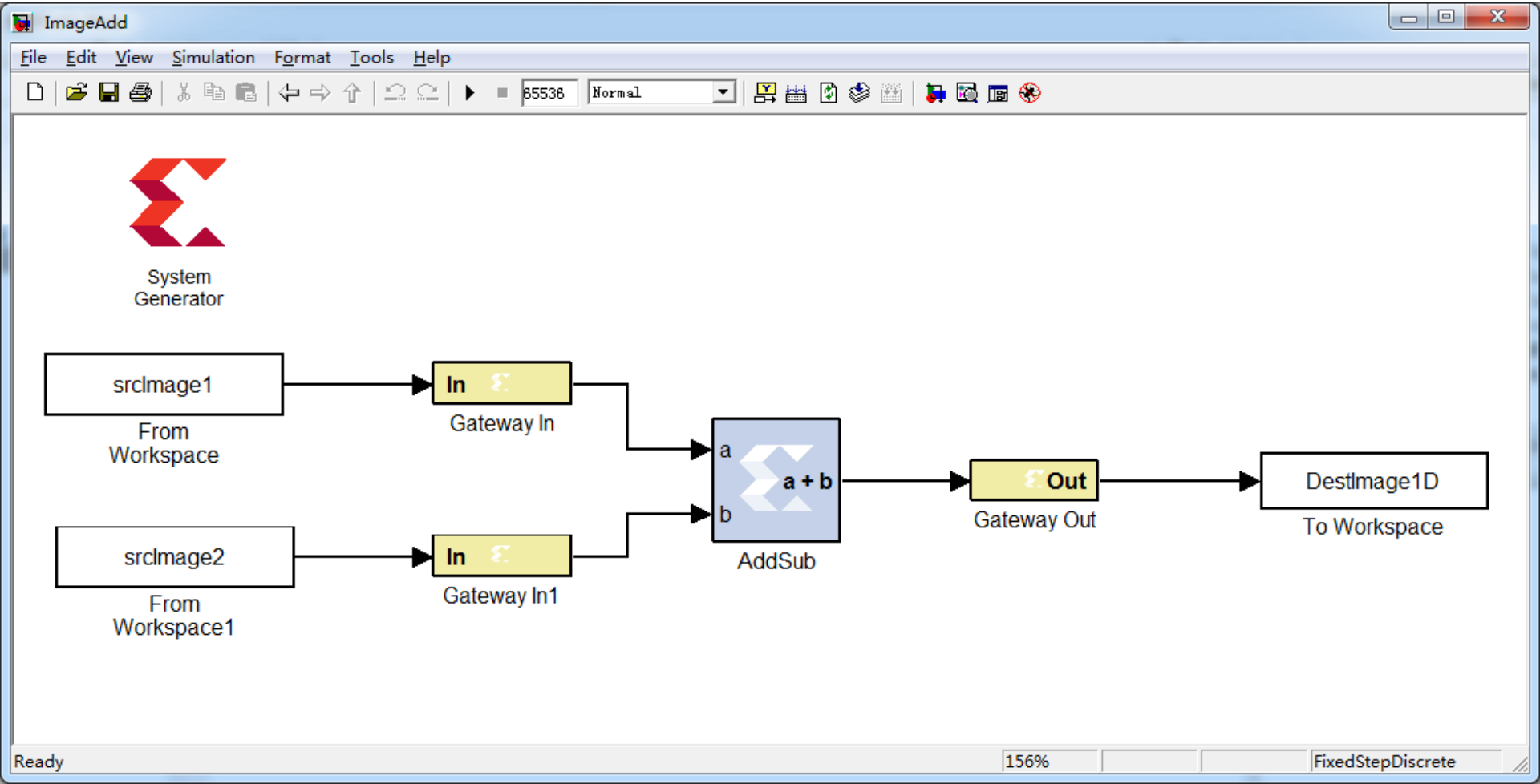
- SystemC —— 标准的 C++




```
1   #include "systemc.h"
2
3   SC_MODULE ( decoder2x4 )
4   {
5       sc_out<sc_uint<4> > y;
6       sc_in <sc_uint<2> > sel;
7
8       void proc_decoder2x4();
9
10      SC_CTOR(decoder2x4)
11      {
12          SC_METHOD( proc_decoder2x4);
13          sensitive << sel;
14      }
15  };
16
17  void decoder2x4::proc_decoder2x4()
18  {
19      switch( sel.read() )
        {
          case 0:  y = 0x1;
                   break;
          case 1:  y = 0x2;
                   break;
          case 2:  y = 0x4;
                   break;
          case 3:  y = 0x8;
                   break;
          default: y.write( 'X');
29      }
30  }
31
```

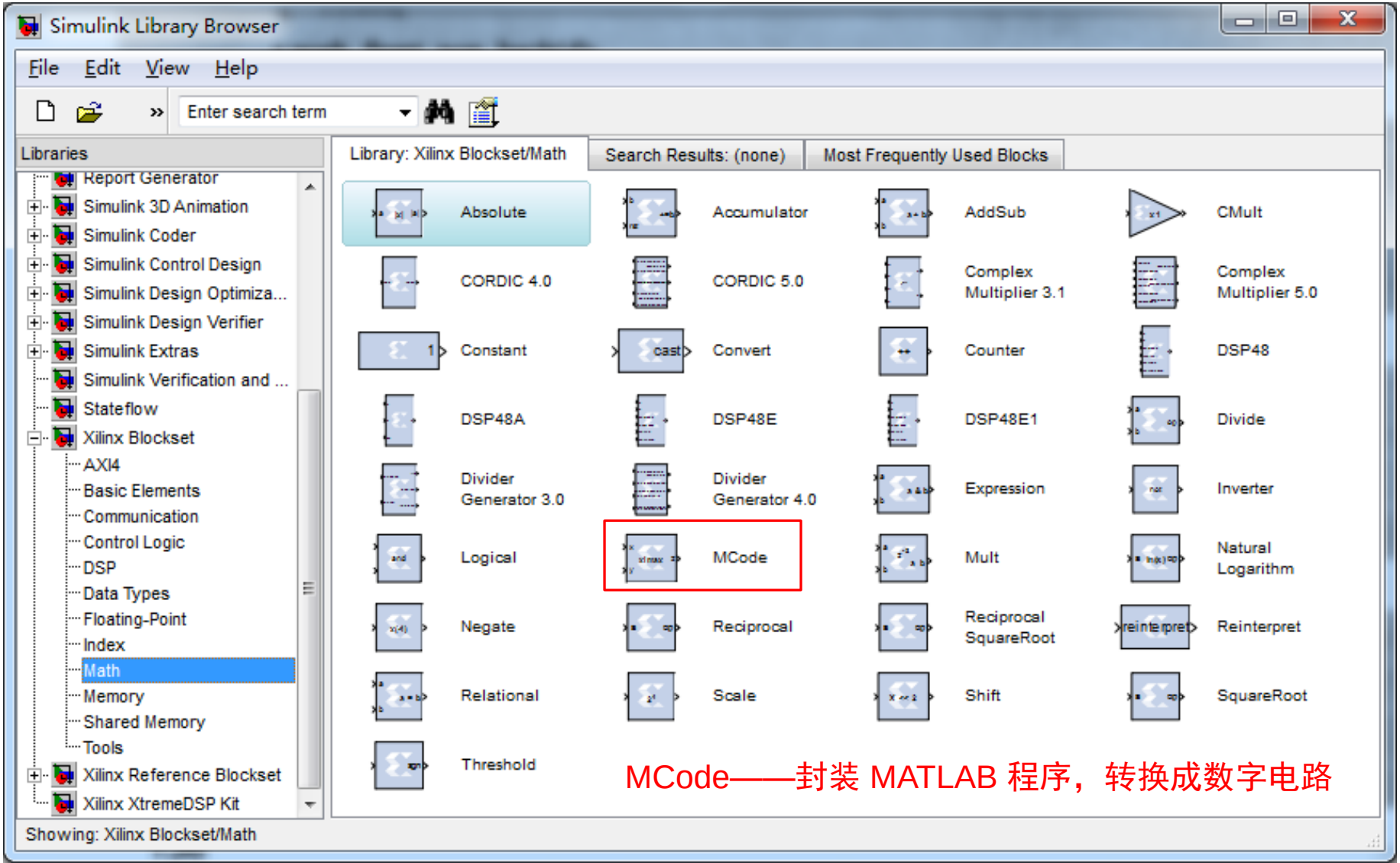
设计方法举例 —— Matlab/Simulink

- Simulink —— 基于 MATLAB 的可视化系统仿真工具
  - 动态系统建模、仿真、综合分析工具
- Xilinx System Generator 工具—— Simulink 扩展工具箱
  - 基于FPGA 实现图像相加： Dest = I1 + I2



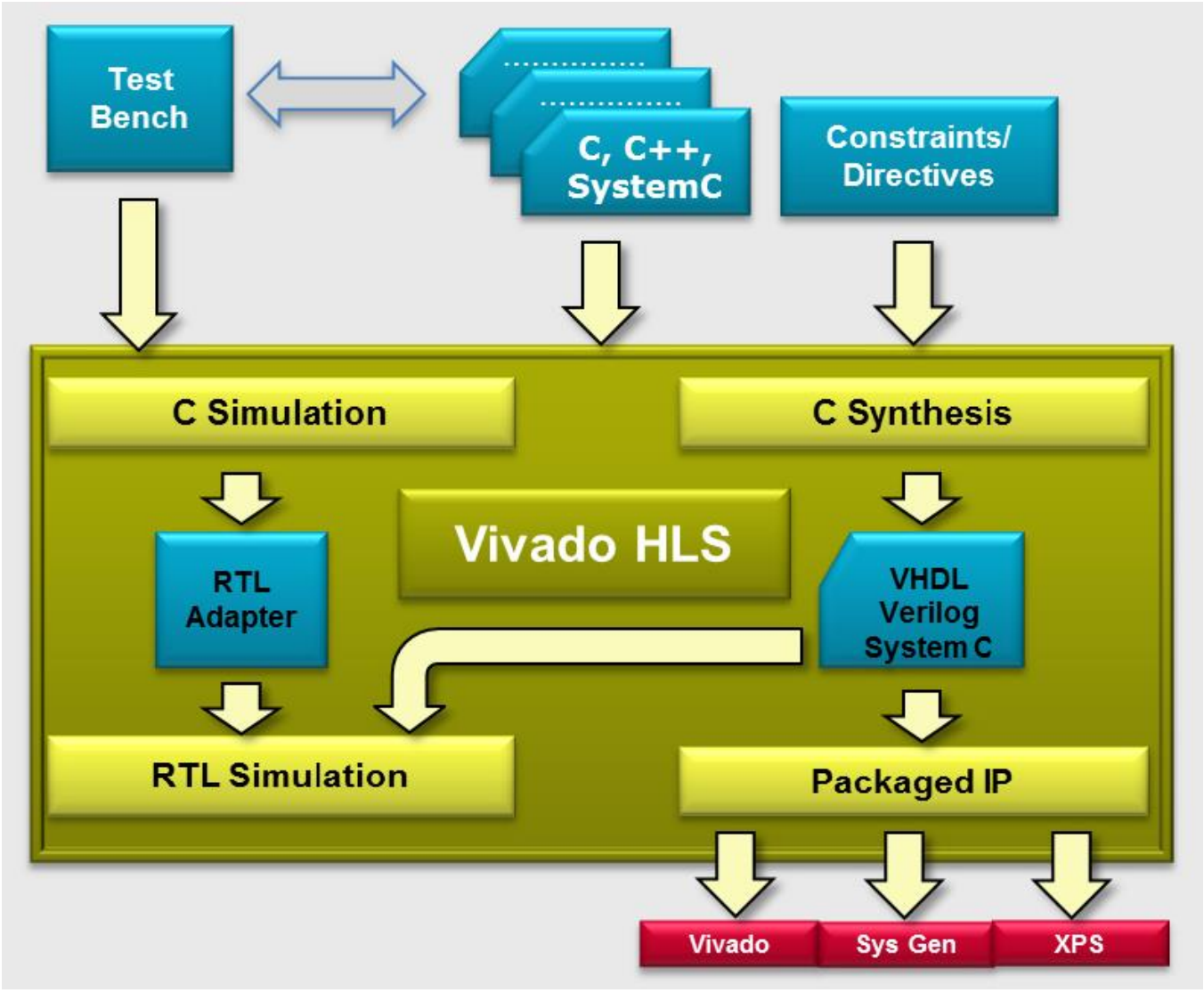
Xilinx System Generator

- 对建模工具 Simulink 进行扩展，提供基于 Simulink 的硬件 DSP 建模和设计环境

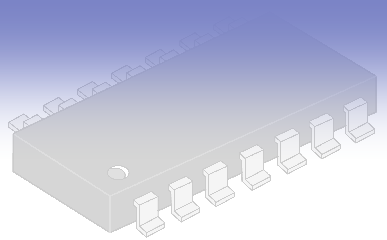


Xilinx Vivado HLS （High-Level Synthesis）

- HLS —— High-Level Synthesis
  - 高层次综合

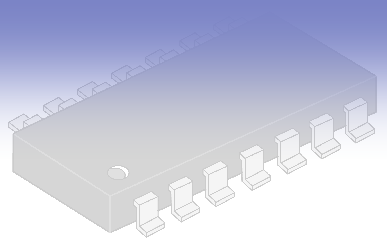


现代数字电路设计方法



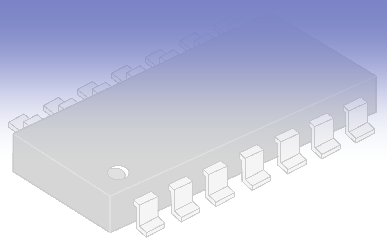
- 采用 HDL 进行建模、仿真验证、设计和综合
- 利用 HDL 对数字电路分层次进行设计
  - 从上层（抽象）到下层（具体）的一系列分层模块描述
- 使用EDA软件逐层进行仿真、验证
- 使用自动综合工具产生物理电路
  - 将需要产生物理电路的模块转换成门级电路网表（netlist）
- 使用自动布线工具进行布线
  - 在FPGA 或 CPLD上将网表转换成具体电路的布线
  - 利用标准单元库实现ASIC具体的电路布线
- 在制成物理器件之前，使用门级模型代替具体元件进行验证
  - 验证复杂数字系统物理结构的正确性

基于 HDL 的设计



- 可以对设计结果进行早期仿真
  - 在设计过程中，每一层设计完成后，都进行仿真
  - 可以在系统设计的早期发现问题
  - 与后期仿真相比，大大缩短设计周期，降低设计成本
- 使用HDL设计电路降低了硬件电路设计难度
  - 系统硬件设计容易实现
  - 设计方法灵活、技术支持广泛
  - 设计编程与工艺无关
  - 设计文件易于共享和重用

Verilog 建模、仿真与综合



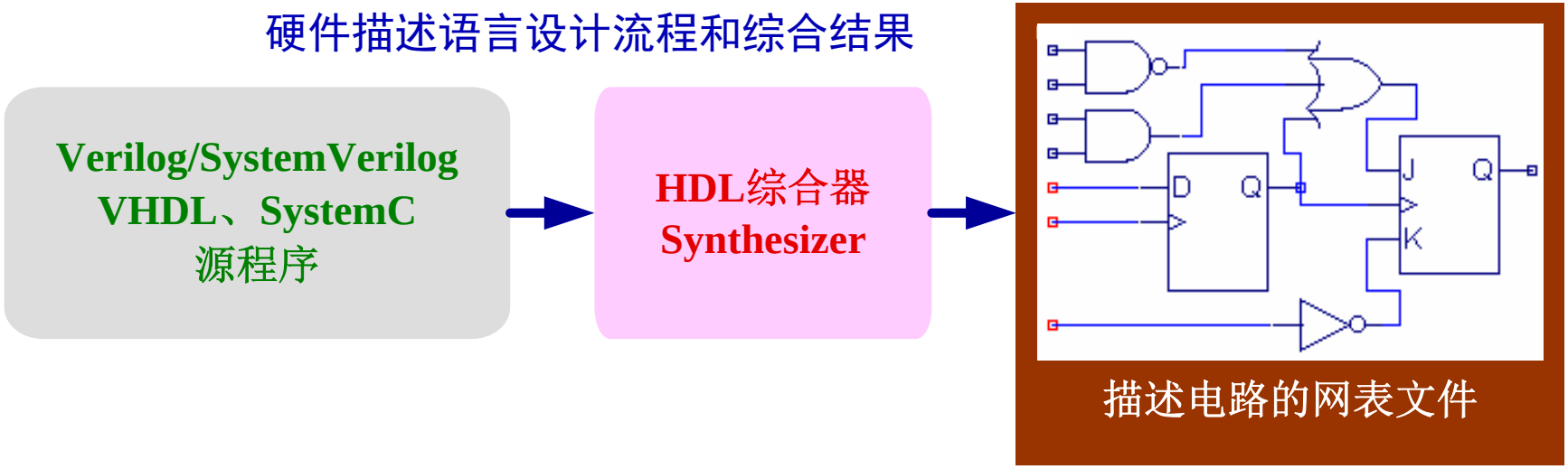
- 数字逻辑电路建模、仿真、综合
  - 建模和仿真—— 可以使用 Verilog 全部功能
  - 可综合设计 —— 只能使用 Verilog 的一个子集（可综合子集）
- 综合 —— 将电路描述由高层到低层的转换
  - 将使用 Verilog 可综合子集设计的电路映射成实际物理电路

综合（Synthesis）

- 综合 —— 将电路描述由高层到低层的转换
  - 将使用 Verilog 可综合子集设计的电路映射成实际物理电路
  - 行为/算法综合、RTL综合
- 编译器和综合工具的功能比较

程序语言设计流程和编译结果

硬件描述语言设计流程和综合结果

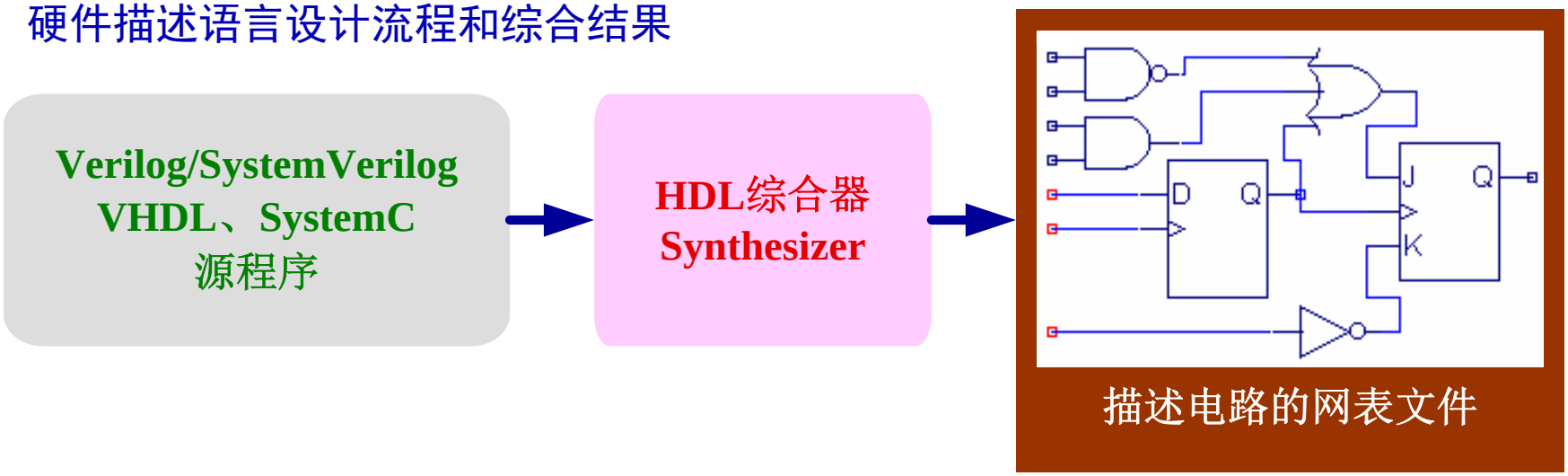


仿真与综合（Synthesis）

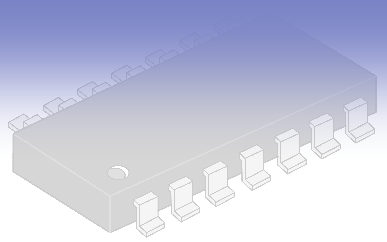
- 仿真工具和综合工具的功能比较

硬件描述语言仿真验证和编译结果

硬件描述语言设计流程和综合结果



HDL数字系统设计过程



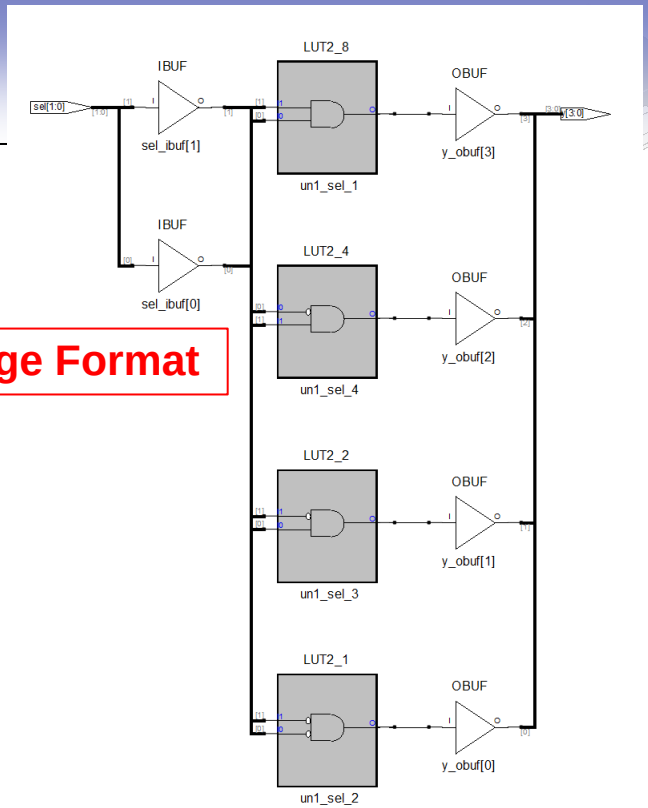
- 利用HDL的设计电路的三个层次
- 第一层：行为描述（算法描述）
  - 对数字系统的数学模型描述
  - 抽象程度高、难以直接映射到具体的逻辑单元
- 第二层：寄存器传输描述
  - RTL（register-transfer level）方式描述
  - 基于时钟的数据流，任何钟控元件在时钟沿处的行为都要精确描述
  - 可以进行综合
- 第三层：逻辑综合 —— 利用综合工具
  - 将RTL方式描述的程序转换成用基本逻辑单元（门级）互连表示的文件（网表）
  - 利用门级网络表
    - ASIC设计——由自动布线程序将网络表转换成相应的制造工艺，借助版图工具等
    - PLD设计 —— 将网络表转换成FPGA的编程码流，生成实现物理电路的芯片

Verilog 设计模块与网表

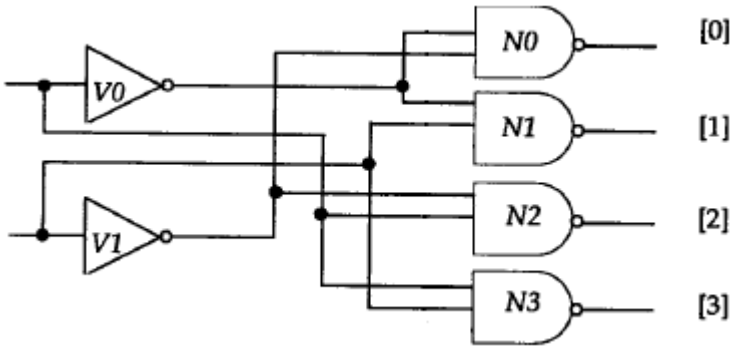
网表文件：decoder2x4.edf

edif: Electronic Design Interchange Format

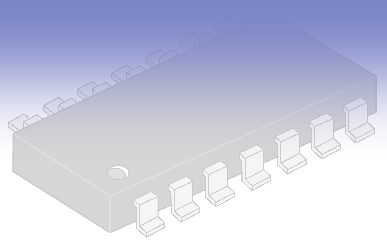
```
(edif decoder2x4
 (edifVersion 2 0 0)
 (edifLevel 0)
 (keywordMap (keywordLevel 0))
 (status
  (written
   (timeStamp 2016 9 18 18 25 55)
   (author "Synopsys, Inc.")
   (program "Synplify Premier" (version "K-2015.09, mapper maprc, Build 2950R"))
  )
 )
 (library VIRTEX
  (edifLevel 0)
  (technology (numberDefinition ))
  (cell IBUF (cellType GENERIC)
   (view PRIM (viewType NETLIST)
    (interface
     (port O (direction OUTPUT))
     (port I (direction INPUT))
    )
   )
  )
  (cell OBUF (cellType GENERIC)
   (view PRIM (viewType NETLIST)
    (interface
     (port O (direction OUTPUT))
     (port I (direction INPUT))
    )
   )
  )
```




```
// File: decoder2x4.v
module decoder2x4(output reg[3:0] y, input [1:0] sel );
    always @(*) begin
        case (sel)
            0:  y = 1;
            1:  y = 2;
            2:  y = 4;
            3:  y = 8;
            default: y = 4'
        endcase
    end
endmodule
```



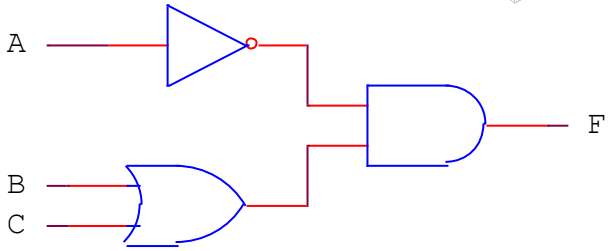
使用 HDL 基于FPGA/CPLD的设计流程

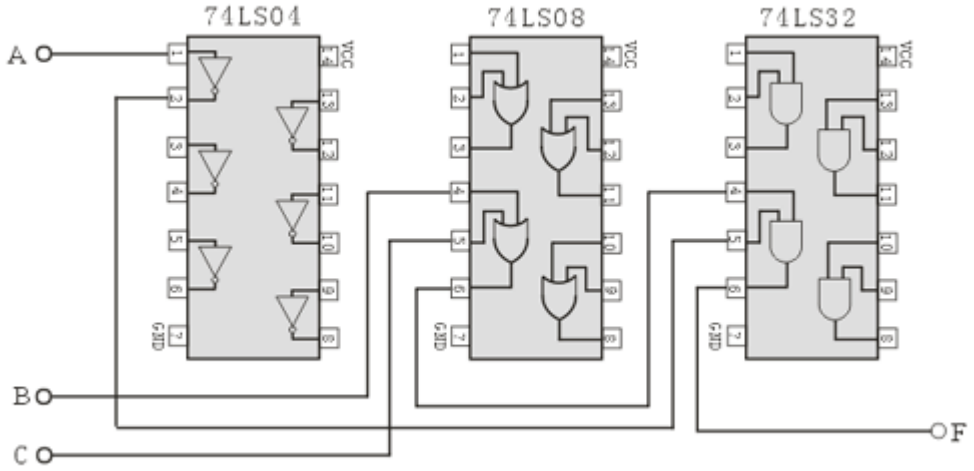


- HDL设计描述
  - 设计的行为或结构描述
- 功能仿真
  - 功能仿真、验证逻辑模型
- 综合 —— 把 HDL 设计翻译成目标电路的工艺
  - 最优化
    - 最小面积——合适的面积要求
    - 最快速度——满足性能要求
  - 综合后仿真—— 门级网表仿真，对综合后的门级网表仿真
- 布局和布线
  - 映射设计到目标可编程器件中的指定位置
  - 使用指定的布线资源
  - 时序仿真——使用布线后生成的时序模型进行仿真
    - 使用时序模型进行仿真可以验证设计是否满足时序要求
- 在系统编程和测试器件
  - 编程文件——布局布线程序生成的配置文件
  - 将配置文件下载到PLD器件中可以实现对器件编程

例、一个简单的逻辑电路

- 逻辑表达式为： $F = \bar{A} \cdot (B + C)$
- 传统的实现方法
  - 使用74LS04、74LS08和74LS32
  - 实现这个简单的逻辑运算，需要有3个芯片
  - 设计完成后无法更改逻辑功能





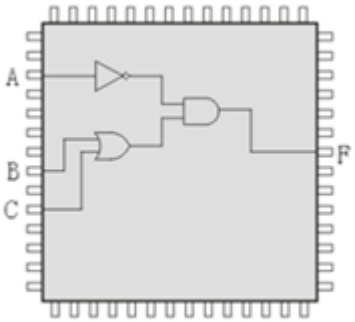
基于 FPGA/CPLD的实现方法

- 用可编程逻辑器件来实现
  - 例如：
    - 用 Xilinx 公司的XC9572，如下图所示
- 设计过程简单
  - 利用HDL，将逻辑表达式按规定的语法进行描述
  - 进行仿真、综合
  - 最后，下载到可编程逻辑器件中
- 优点
  - 只用一片集成电路就可以实现
  - 逻辑电路越复杂，其优越性越显示
  - 容易实现一些用普通的集成电路难以实现的功能

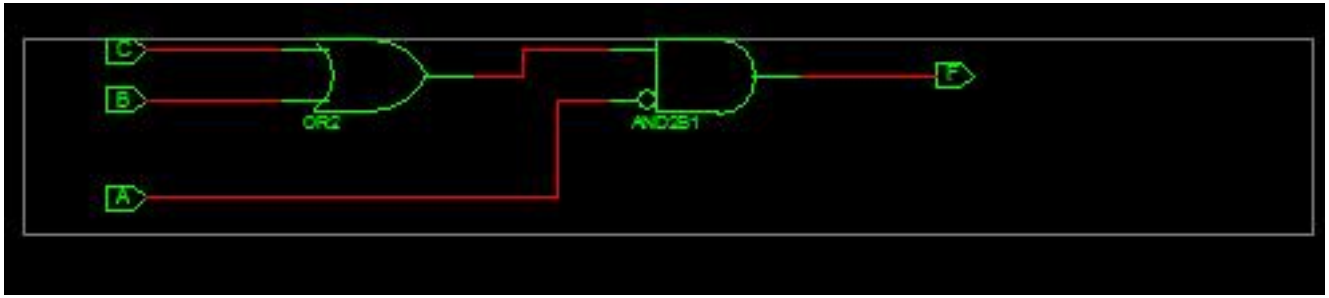
用 Verilog HDL 描述逻辑电路

- 前面的简单逻辑电路用Verilog HDL描述如下：

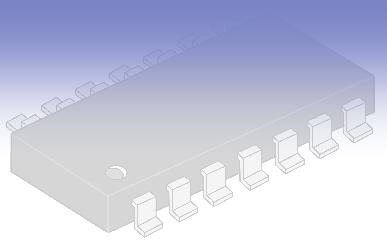
```
module simple( F, A, B, C );
     input A, B, C;
      output F;
      assign F = ~A & (B | C);
endmodule
```

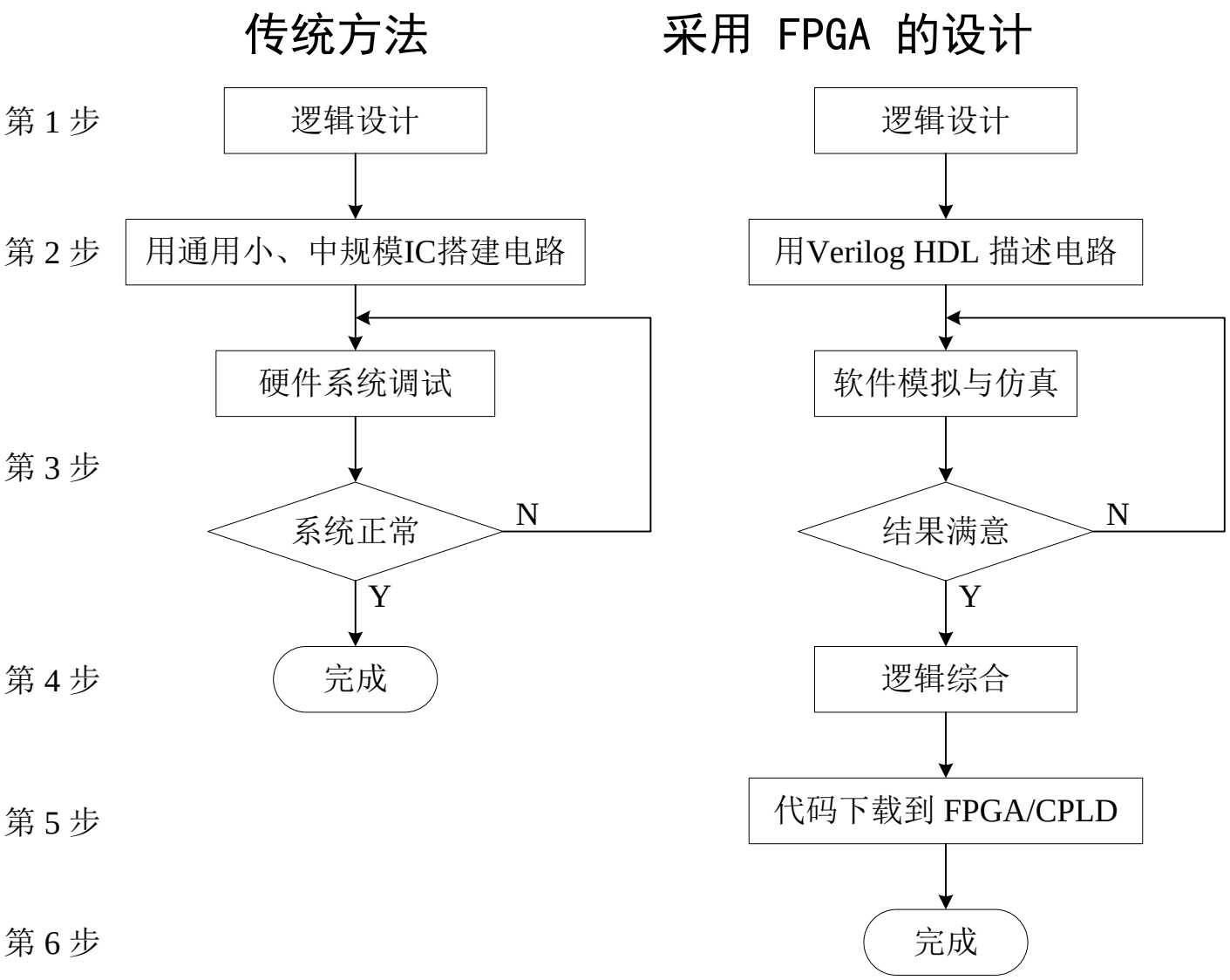


- 仿真确认逻辑关系正确
- 综合、下载到可编程逻辑器件中
- 生成所需逻辑电路
- 一片可编程逻辑器件实现了3片集成电路所构建的电路功能
- 逻辑门之间的连线是通过编程在芯片内部完成



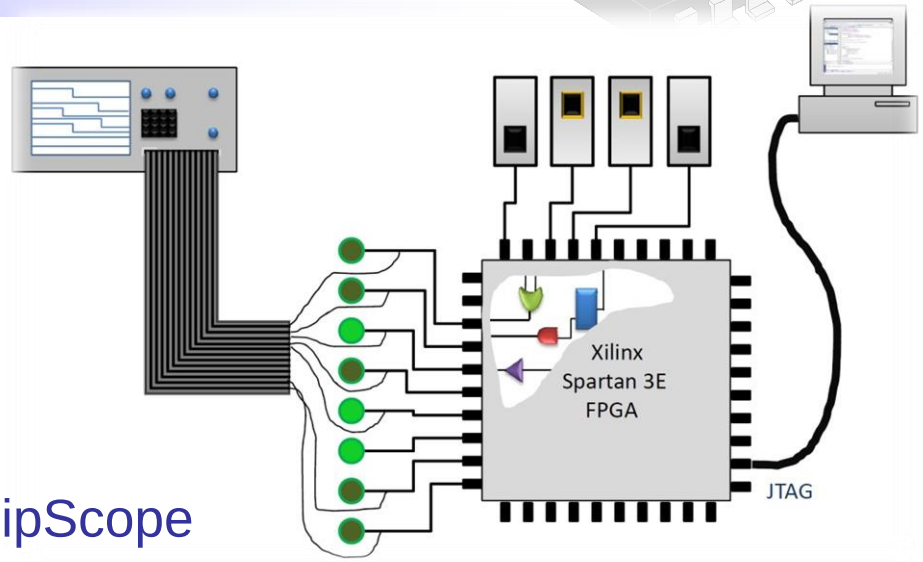
采用 FPGA 的设计方法与传统方法的比较



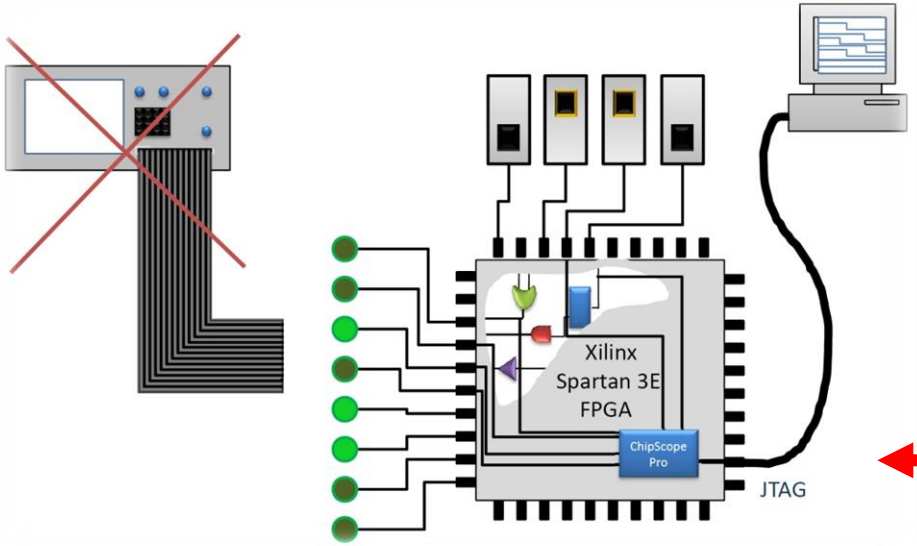


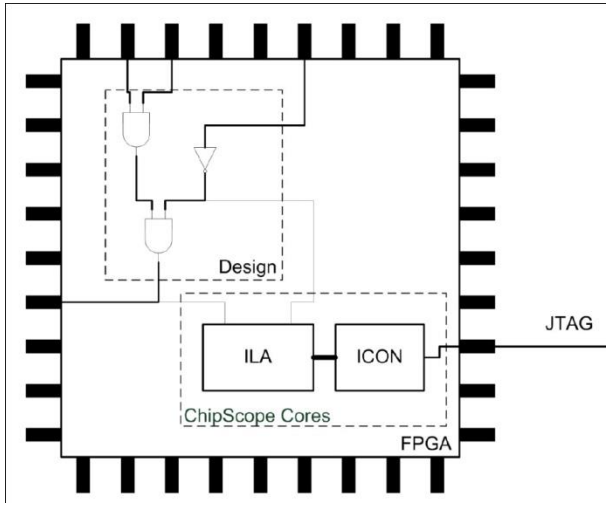
基于 FPGA 的数字设计的测试和调试方法

- 传统测试方法 —— 外接逻辑分析仪



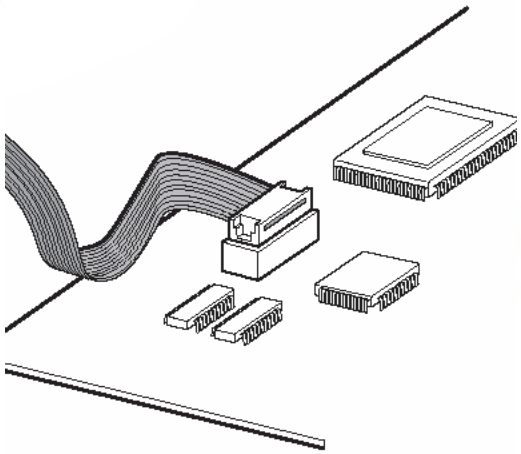
- 在设计中插入逻辑分析仪核 —— ChipScope
  - 不再需要外接逻辑分析仪

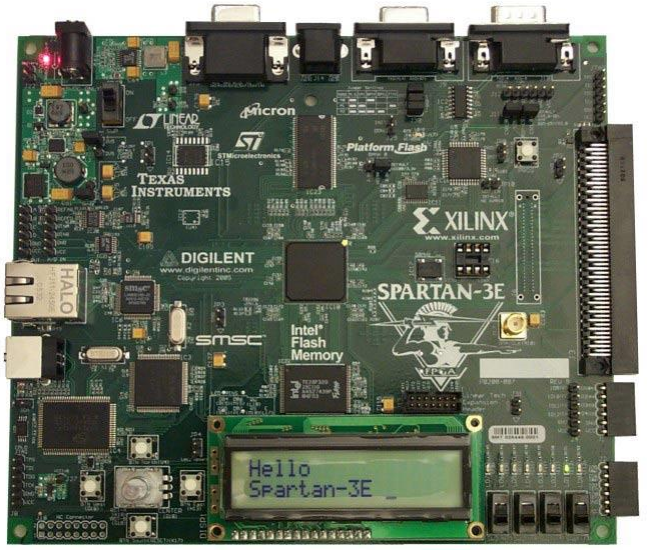




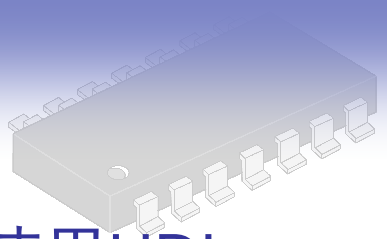
FPGA/CPLD 数字系统开发

- 特点
  - 减轻设计工作量，缩短系统开发时间
  - 有完善先进的开发工具
    - 提供语言、图形等设计方法，十分灵活
    - 通过仿真工具来验证设计的正确性
  - 集成度高，可以替代多至几千块通用IC芯片
  - 极大减小电路的面积，降低功耗，提高可靠性
  - 可以反复地擦除、编程，方便设计的修改和升级



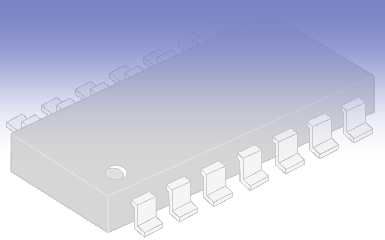


主要的FPGA生产厂家



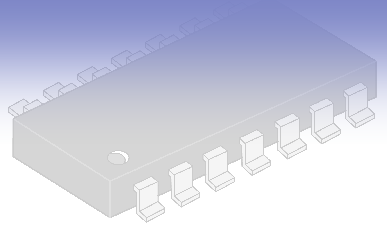
- 各PLD /FPGA芯片制造厂家都支持使用HDL设计
- www.xilinx.com
  - FPGA的发明者，最大的PLD供应商之一
- www.intel.com
  - 2015年6月，Intel公司收购 Altera
  - Altera：最大的PLD供应商之一
- www.latticesemi.com
  - 主要的 PLD 供应商之一
- www.microsemi.com
  - 提供军品及宇航级产品、混合信号FPGA
  - 2010 收购 Actel
- www.cypress.com
  - 提供数字/模拟混合信号处理的 PSoC
    - Programmable System-on-Chip
- www.atmel.com
  - 主要的 PLD 供应商之一

国内的FPGA生产厂家



- 中国航天电子技术研究院第772究所
  - 北京微电子技术研究所
  - 宇航级FPGA
- 京微雅格
  - www.capital-micro.com
- 高云半导体
  - http://www.gowinsemi.com.cn
- 紫光国芯（深圳国微电子）
  - http://www.ssmec.com
- 西安智多晶微电子
  - http://isilicontech.com
- 成都华微电子科技
  - http://www.chinabut.com
- 上海遨格芯微电子
- 复旦微电子/复旦微电子学院

总结



- 现代数字系统设计方法
  - 采用 HDL 描述数字系统
  - 利用 HDL 对数字电路分层次进行设计
  - 使用 EDA 软件逐层进行仿真、验证、综合
- 基于 FPGA/CPLD 数字系统设计
  - 可以重复编程
  - 灵活、快速地完成多种数字设计工作