SoC复习

- 考试题型
- SOC重点

考试题型

- 5道大题
- 选择题
- VHDL程序分析
- SOC结构分析设计
- 硬件描述语言编程2道

SoC重点

- VHDL语法结构
  - 描述方式、数据对象
  - 并行与串行语句
  - 进程、过程

1.3.1 VHDL数据对象

1、Constant（常量）在程序中不可以被赋值

2、Variable（变量）在程序中可以被赋值(用“: =”)，赋值后立即变化为新值。

3、Signal（信号）在程序中可以被赋值(用“<=”)，但不立即更新，当进程挂起后，才开始更新。

4、File（文件）在程序中实现对文件的写入和读出操作。

1.4 VHDL基本逻辑语句

- Architecture 中的语句及子模块之间是并行处理的

➢子模块block中的语句是并行处理的

➢子模块process中的语句是顺序处理的

➢子模块subprogram中的function和procedure是顺序处理的

4）顺序执行语句 sequential statement

- Wait语句
- If 语句
- case语句
- for loop语句
- while 语句

5）并行处理语句
concurrent statement

1、信号赋值操作

2、带条件的信号赋值语句

3、带选择的信号赋值语句

1.5 VHDL描述方式

- 行为描述
- 数据流描述
- 结构描述

元件调用

1. 元件说明：出现在architecture和begin之间

```
component <实体名>
    [generic(<类属表>) ; ]
    port(<端口名>);
end component;
```

2. 元件例化：出现在结构体的语句部分

```
<标号名>：<元件名>
[ generic map(<类属关联表>)]
  port map(<端口关联表>);
```

元件调用

（2）端口映射语句(port map)

①位置映射方法

and2端口定义： port(a，b:in bit;c:out bit);

元件例化语句： u1:and2 port map(ina，inb，s1);

即在u1中，ina对应a，inb对应 b，s1对应c。

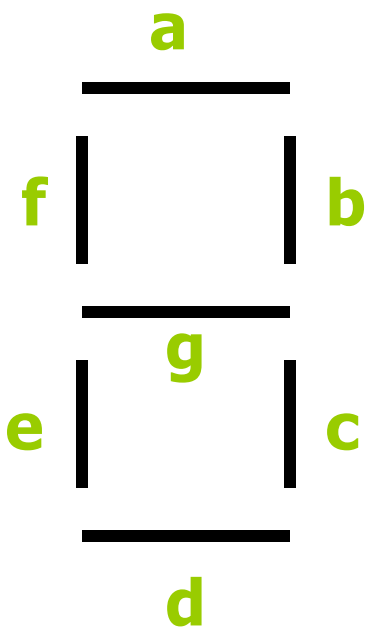
②名称映射方法

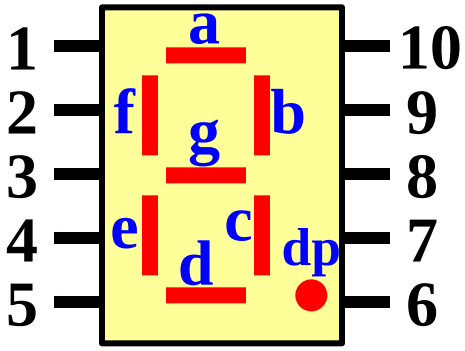
u1:and2 port map(a＝＞ina， b＝＞inb，c＝＞s1);

或

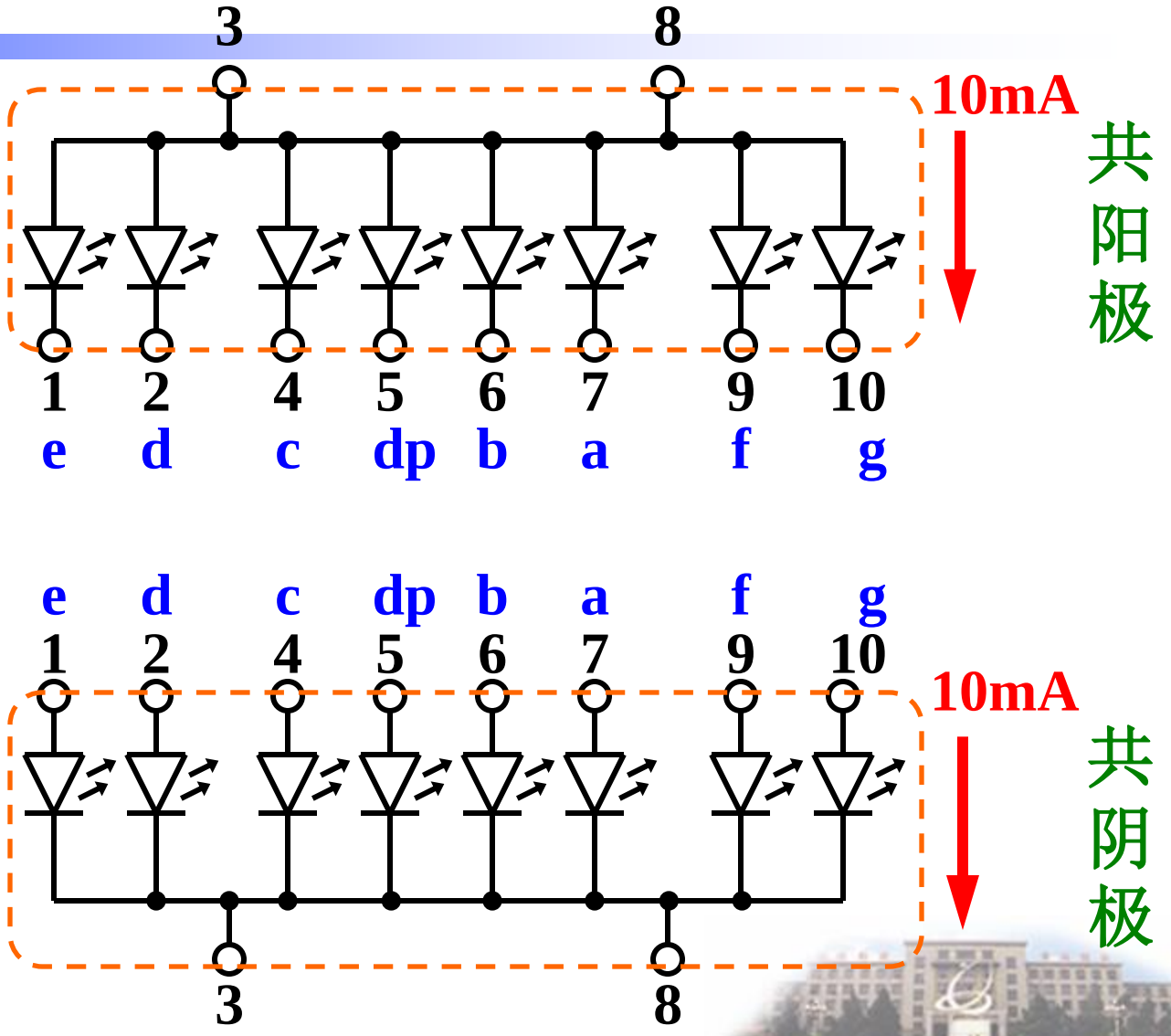
u1:and2 port map(b＝＞inb， a＝＞ina， c＝＞s1);

```
with input select
    drive<=B“1111110” when 0,
           B“0110000” when 1,
           B“1101101” when 2,
           B“1111001” when 3,
           B“0110011” when 4,
           B“1011011” when 5,
           B“1011111” when 6,
           B“1110000” when 7,
           B“1111111” when 8,
           B“1111011” when 9,
           B“0000000” when others;
end architecture art;
```




a
f
g
b
e
c
d
dp
1
2
3
4
5
10
9
8
7
6

数码管


3
8
10mA
共阳极
1 2 4 5 6 7 9 10
e d c dp b a f g
e d c dp b a f g
1 2 4 5 6 7 9 10
10mA
共阴极
3
8

VHDL语言程序分析

- LIBRARY IEEE;
- USE IEEE.STD_LOGIC_1164.ALL;
- 1 ENTITY MUX21 IS
- PORT(SEL:IN STD_LOGIC;
- A,B:IN STD_LOGIC;
- Q: OUT STD_LOGIC);
- END MUX21;
- 2 ARCHITECTURE BHV OF MUX21 IS
- BEGIN
- Q<=A WHEN SEL=’1’ ELSE B;
- END BHV;

```
ENTITY CLKDIV8_1TO2 IS
        PORT(CLK:IN STD_LOGIC;
                CLKOUT:OUT STD_LOGIC );
END CLKDIV8_1TO2;
ARCHITECTURE TWO OF CLKDIV8_1TO2 IS
SIGNAL CNT:STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL CK:STD_LOGIC;
BEGIN
PROCESS(CLK)
BEGIN
IF RISING_EDGE(    3    ) THEN
  IF CNT="11" THEN
CNT<="00";
CK<=     4    ;
  ELSE CNT<=     5    ;
  END IF;
END IF;
CLKOUT<=CK;
END PROCESS;
END;
```

CLK

NOT CK

CNT+1

SoC重点

- SoC结构
  - SoC基本概念、基本结构
  - SoC研究内容、关键技术、设计验证、软硬件综合
- 四则运算设计
  - 加法器
  - 乘法器
  - 除法器

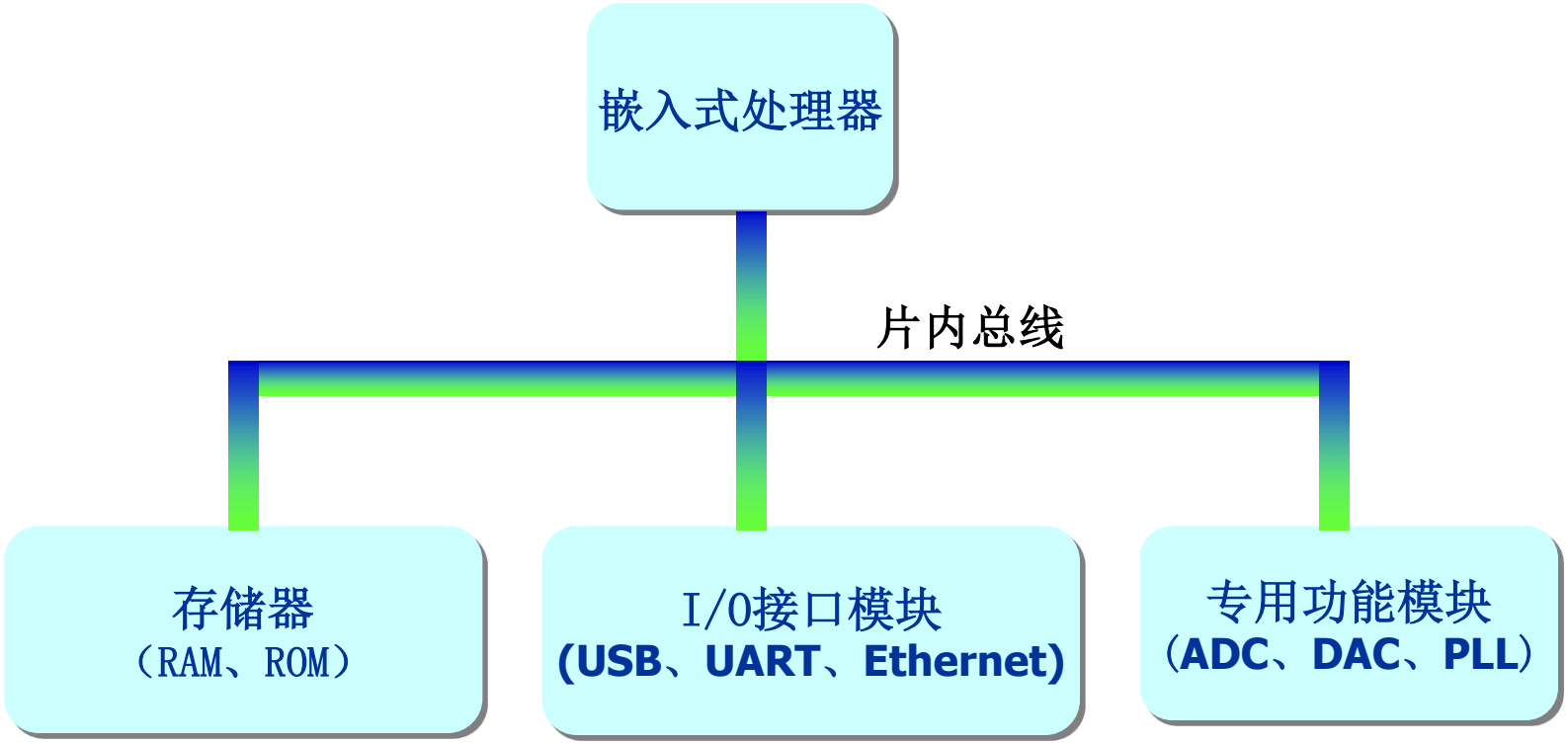
1.1 什么是SoC？

- 片上系统（System on Chip,SoC），是指在单一芯片上集成了数字电路、模拟电路、信号采集和转换电路、存储器、MPU、MCU、DSP、MPEG等，实现了一个系统的功能。

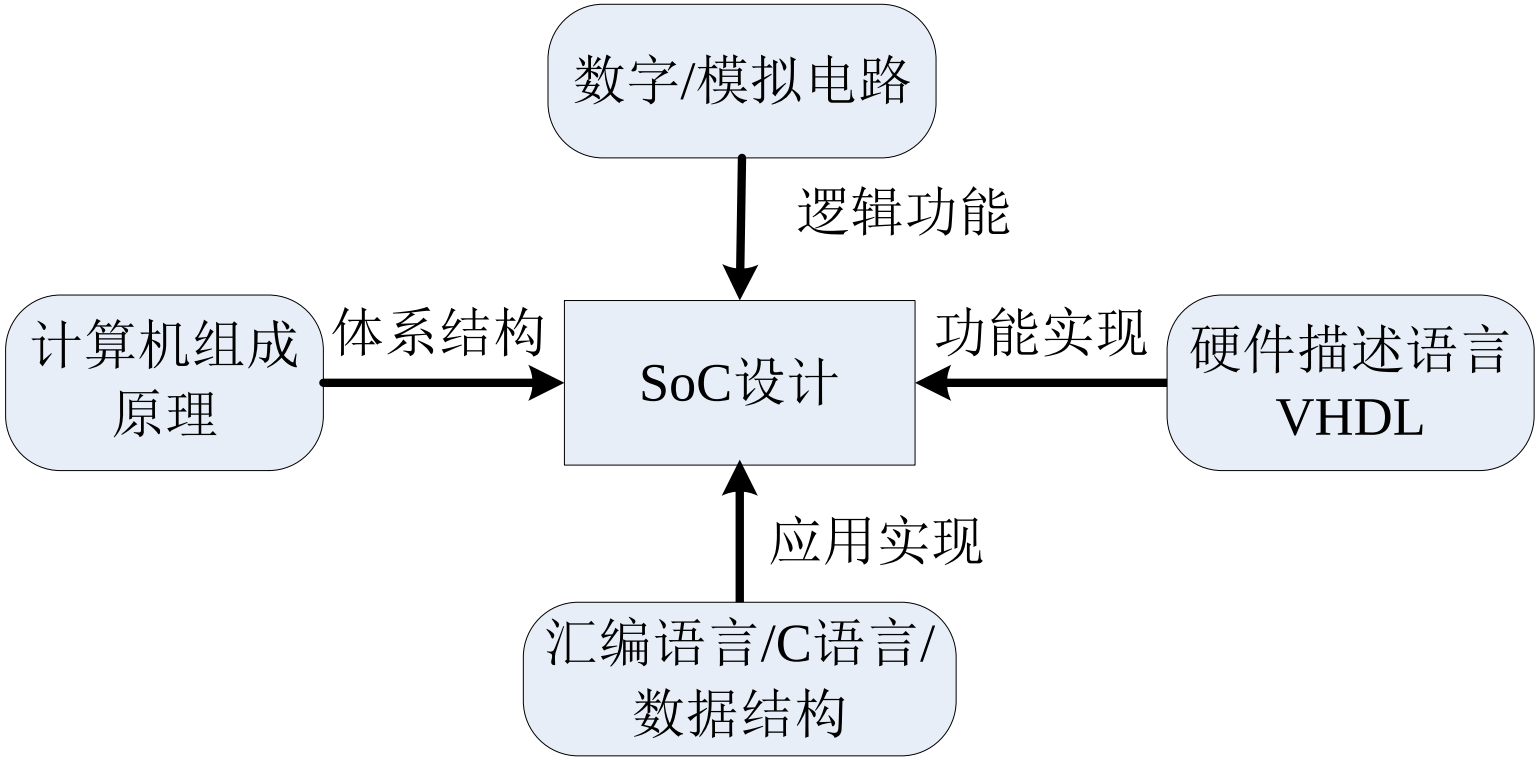
1.1.3 SoC基本构成

- 嵌入式处理器核（如MPU、MCU或DSP）
- 存储器（如SRAM、SDRAM、Flash ROM）
- 专用功能模块（如ADC、DAC、PLL、2D/3D图形运算单元）
- I/O接口模块（如USB、UART、Ethernet等）等多种功能模块
- 片内总线（AMBA、Wishbone、Avalon等）

SoC基本结构


嵌入式处理器
片内总线
存储器
（RAM、ROM）
I/O接口模块
(USB、UART、Ethernet)
专用功能模块
（ADC、DAC、PLL）

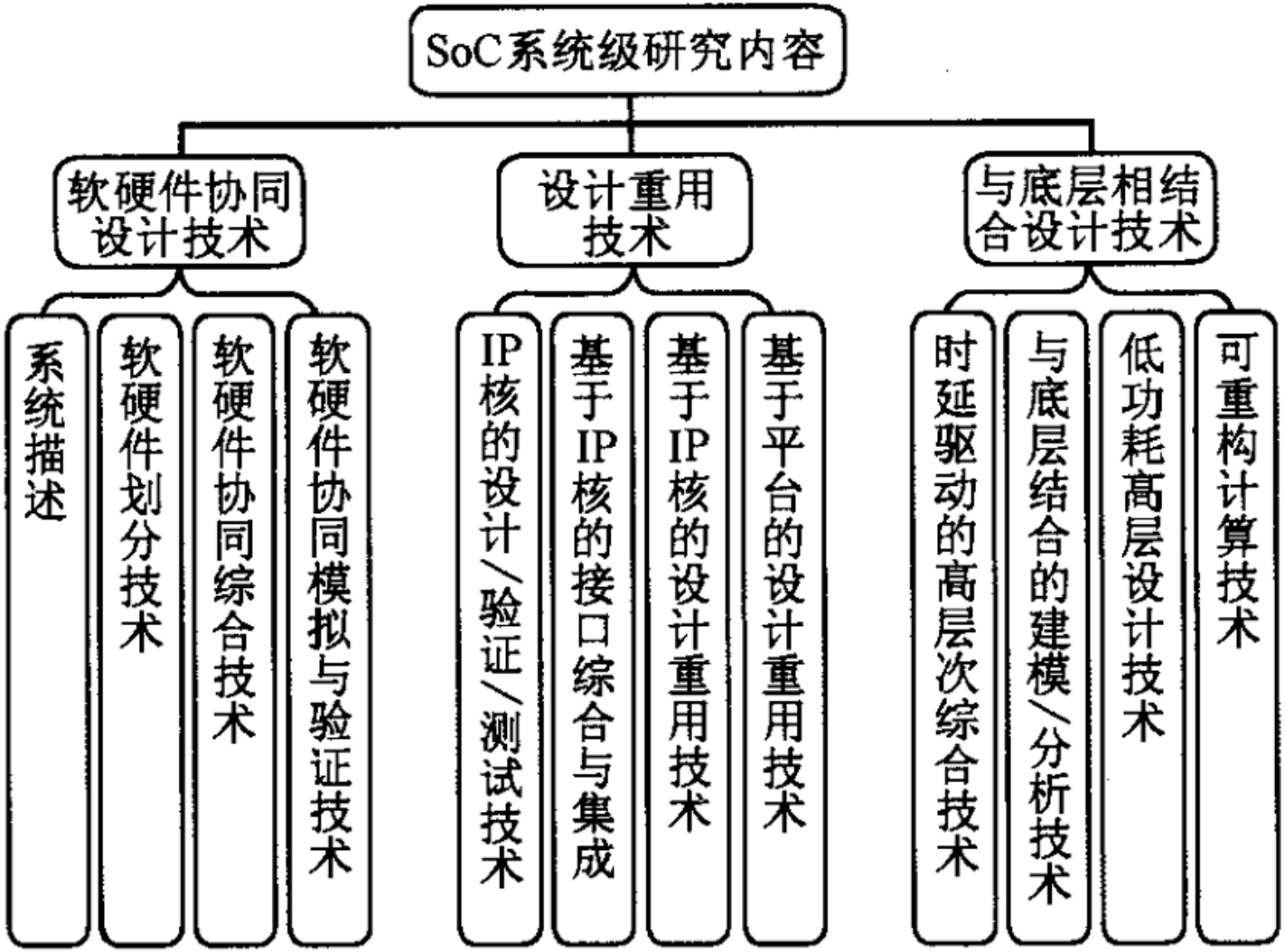
SoC与计算机相关基础课程



SoC类型

- 计算控制型
- 通信网络型
- 信号处理型

1.1.7 SoC系统级研究内容



1.2 SoC设计关键技术

- 设计重用技术
- 低功耗设计技术
- 软硬件协同设计
- 总线架构
- 可测试性设计
- 设计验证
- 物理综合

1.2.1 设计重用技术

◆IP核是指经过反复验证过的、具有特定功能的，可重复利用的逻辑块或数据块，用于专用集成电路（ASIC）或者可编辑逻辑器件（FPGA）。

IP芯核的分类

- 软核
- 固核
- 硬核

类型	定义	使用灵活性	时序性能 可预测性
Soft Cores (软核)	行为级或 RTL 级 HDL 源代码	灵活度高，可修改设计代码，与实现工艺无关。	IP 很难保护，时序性能无法保证，由使用者确定。
Firm cores (固核)	完成软核所有的设计外，还完成了门电路级综合和时序仿真等设计环节，一般以门电路级网表形式提交用户使用。	部分功能可修改，采用指定的实现技术，与实现工艺相关 。	关键路径时序可控制
Hard cores (硬核)	基于某种半导体工艺的物理设计，已有固定的拓扑布局和具体工艺，并已经过工艺验证，具有可保证的性能。提供电路物理结构掩模版图和全套工艺文件。	不能修改设计，必须采用指定实现技术。	包含工艺相关的布局和时序信息， IP 很容易保护，多数的处理器和存储器。

1.2.3 软硬件协同设计关键技术

- 系统建模
- 软硬件划分技术
- 软硬件协同综合
- 软硬件协同仿真与验证

1.3 SoC系统级设计方法

◆自顶向下

美国加州大学Irvine分校嵌入式系统研究小组的基于 SpecC的逐层细化求精设计方法。

◆自底向上

法国TIMA实验室系统级综合小组的基于组件的多处理器核SoC设计方法。

◆上下结合(中间相遇)

美国加州大学Berkeley分校CAD研究小组的基于平台的设计方法。

1.4典型SoC片上总线

- AMBA
- Core Connect
- Avalon
- Wishbone
- OPC

3.1.2 串行进位

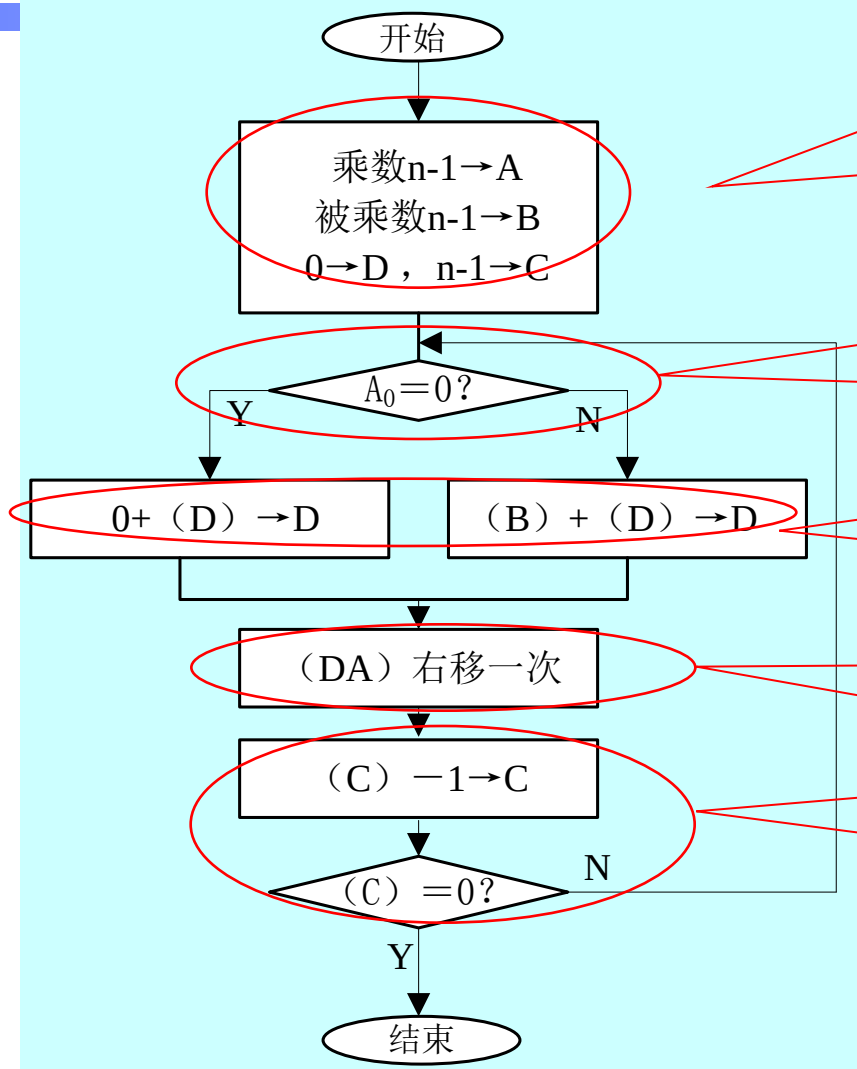
实现8位全加器时，只要在顶层模块进行相应位的映射即可实现。

```
u0:FAdder1 port map(a(0),b(0),cin,sum(0),c(1));
u1:FAdder1 port map(a(1),b(1),c(1),sum(1),c(2));
u2:FAdder1 port map(a(2),b(2),c(2),sum(2),c(3));
u3:FAdder1 port map(a(3),b(3),c(3),sum(3),c(4));
u4:FAdder1 port map(a(4),b(4),c(4),sum(4),c(5));
u5:FAdder1 port map(a(5),b(5),c(5),sum(5),c(6));
u6:FAdder1 port map(a(6),b(6),c(6),sum(6),c(7));
u7:FAdder1 port map(a(7),b(7),c(7),sum(7),cout);
```

3.1.3 并行进位

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ParAdder is
    Port ( a : in  STD_LOGIC_VECTOR(3 DOWNTO 0);
           b : in  STD_LOGIC_VECTOR(3 DOWNTO 0);
           cin : in  STD_LOGIC;
           s : out  STD_LOGIC_VECTOR(3 DOWNTO 0);
           cout : out  STD_LOGIC;
           pp : out  STD_LOGIC;
           gg : out  STD_LOGIC);
end ParAdder;
architecture Behavioral of ParAdder is
signal c: std_logic_vector(3 downto 1);
signal p,g:std_logic_vector(3 downto 0);
begin
--绝对进位
   g(0)<=a(0) and b(0);
   g(1)<=a(1) and b(1);
   g(2)<=a(2) and b(2);
   g(3)<=a(3) and b(3);
--进位传递条件
   p(0)<=a(0) xor b(0);
   p(1)<=a(1) xor b(1);
   p(2)<=a(2) xor b(2);
   p(3)<=a(3) xor b(3);
--产生并行进位
   c(1)<=g(0) or (p(0) and cin);
   c(2)<=g(1) or (p(1) and g(0) ) or (p(1)and p(0) and cin);
   c(3)<=g(2) or (p(2) and g(1) ) or (p(2)and p(1) and g(0)) or (p(2)and p(1)and p(0)and cin);
   cout<=g(3)or(p(3) and g(2) )or(p(3)and p(2) and g(1))or(p(3)and p(2)
         and p(1) and g(0)) or(p(3)and p(2)and p(1)and p(0) and cin );
--本组进位传递条件
   pp<=p(0) and p(1) and p(2) and p(3);
   gg<=g(3) or (p(3) and g(2)) or (p(3)and p(2) and g(1)) or (p(3)and p(2) and p(1) and g(0));
--和输出
   s(0)<=p(0) xor cin;
   s(1)<=p(1) xor c(1);
   s(2)<=p(2) xor c(2);
   s(3)<=p(3) xor c(3);
end Behavioral;
```

4.2.1 原码一位乘法原理及实现



绝对值乘法流程图

A,B,C,D都是寄存器

二选一选择器

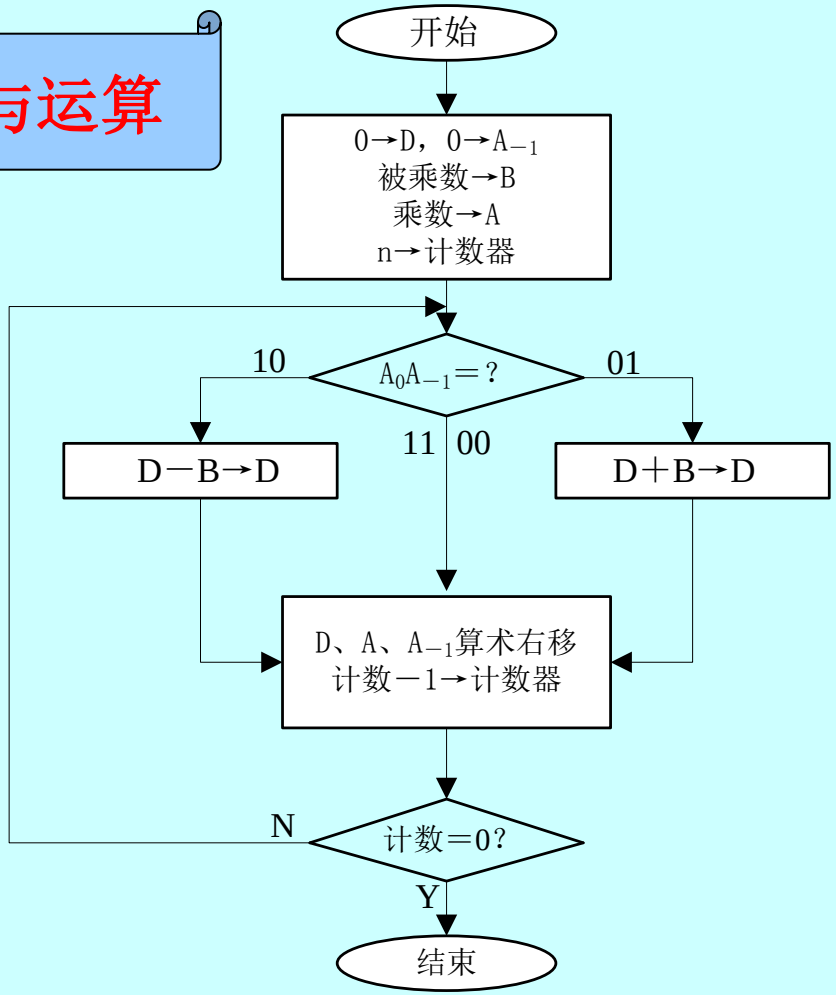
一位全加器

移位寄存器

计数器

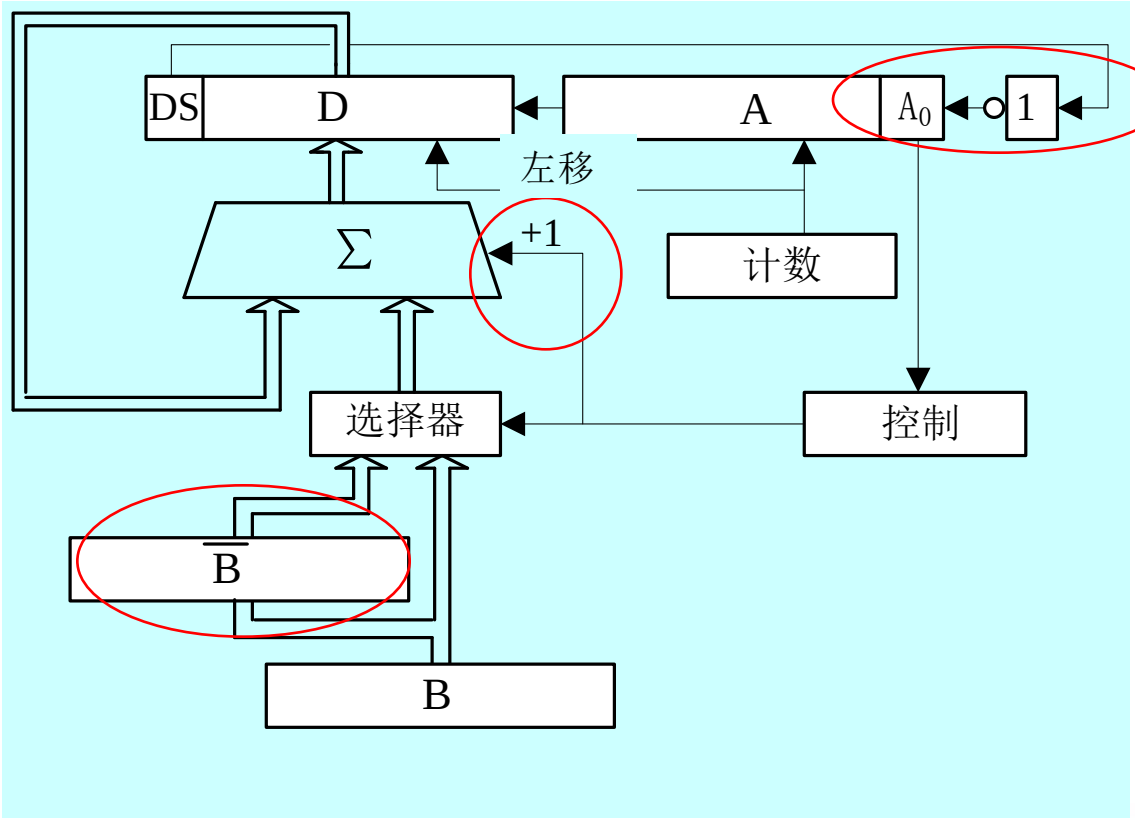
4.2.3 布斯补码一位乘法运算

符号位参与运算

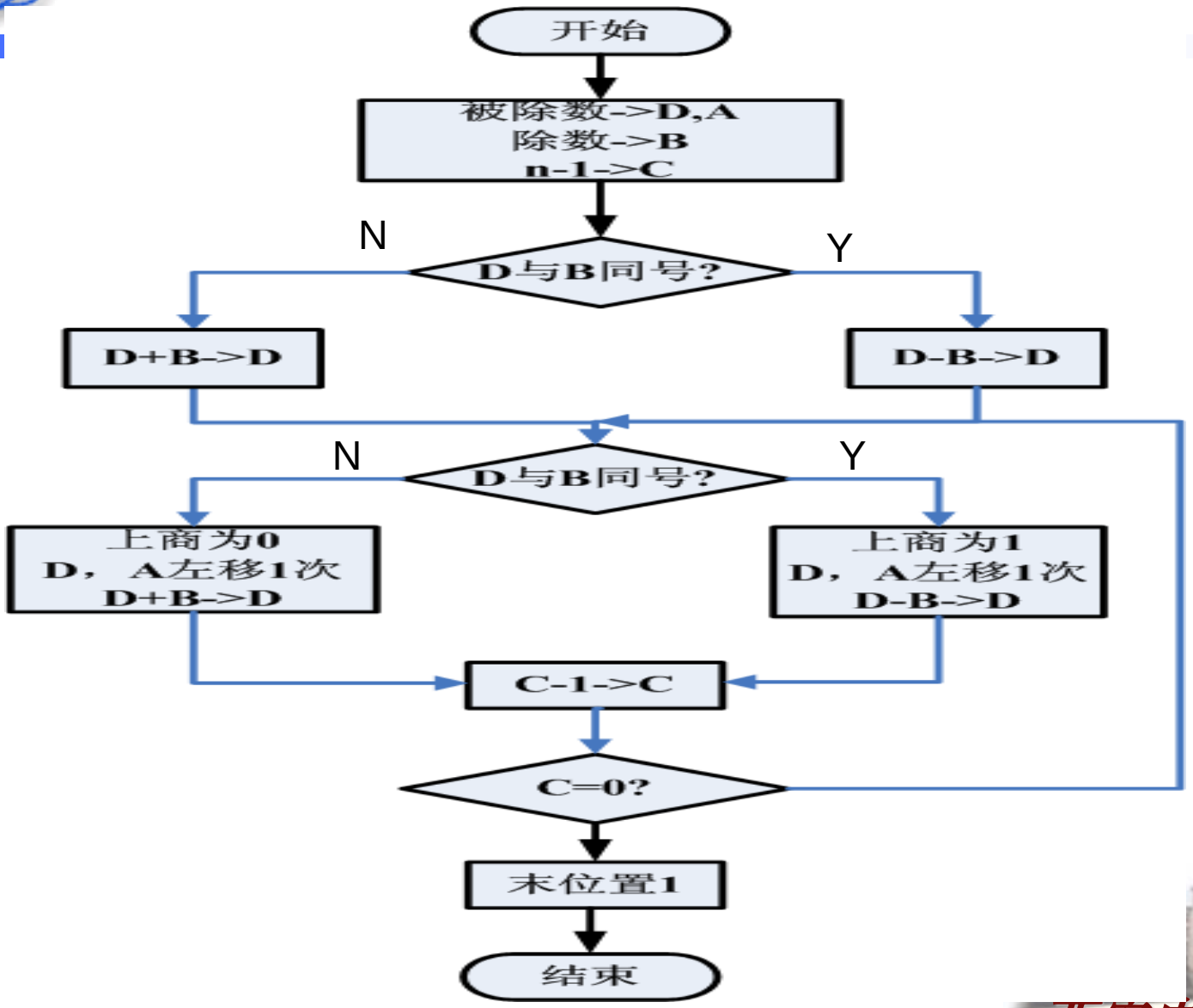


布斯补码一位乘法流程图

4.3.3 原码加减交替除法器



4.3.4 补码除法运算


开始
被除数->D,A
除数->B
n-1->C
N
D与B同号?
Y
D+B->D
D-B->D
N
D与B同号?
Y
上商为0
D，A左移1次
D+B->D
上商为1
D，A左移1次
D-B->D
C-1->C
C=0?
末位置1
结束

SoC重点

- 基本逻辑电路设计
  - 组合逻辑电路
  - 时序电路
  - 有限状态机

逻辑电路设计

逻辑电路

- 组合逻辑电路：现时的输出仅取决于现时的输入
- 时序逻辑电路：除与现时输入有关外还与电路的原状态有关

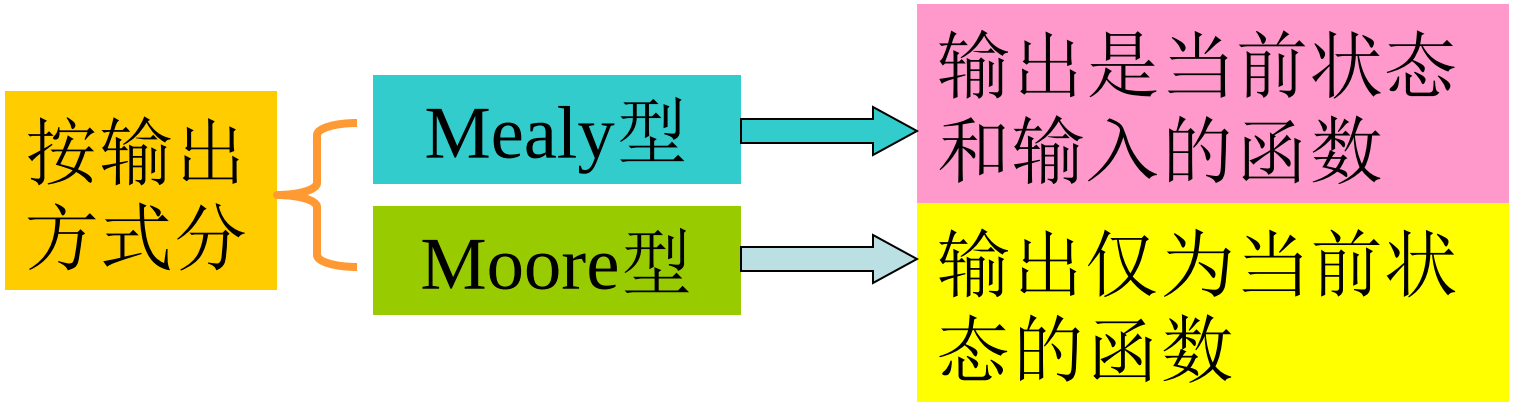
二、 组合逻辑电路设计

- 编码器/译码器设计
- 多路选择器设计
- 数码转化电路设计

三、时序逻辑电路设计方法

- 基于VHDL的时序逻辑电路设计方法
- ① 确定状态数量及状态转换图
- ②按照时钟、输出和驱动状态方程编写VHDL代码
- ③利用EDA工具进行功能仿真验证
- ④FPGA验证

4.2 有限状态机分类



- 课件的例题分析

SoC重点

- 存储器设计
  - ROM、RAM
  - FIFO

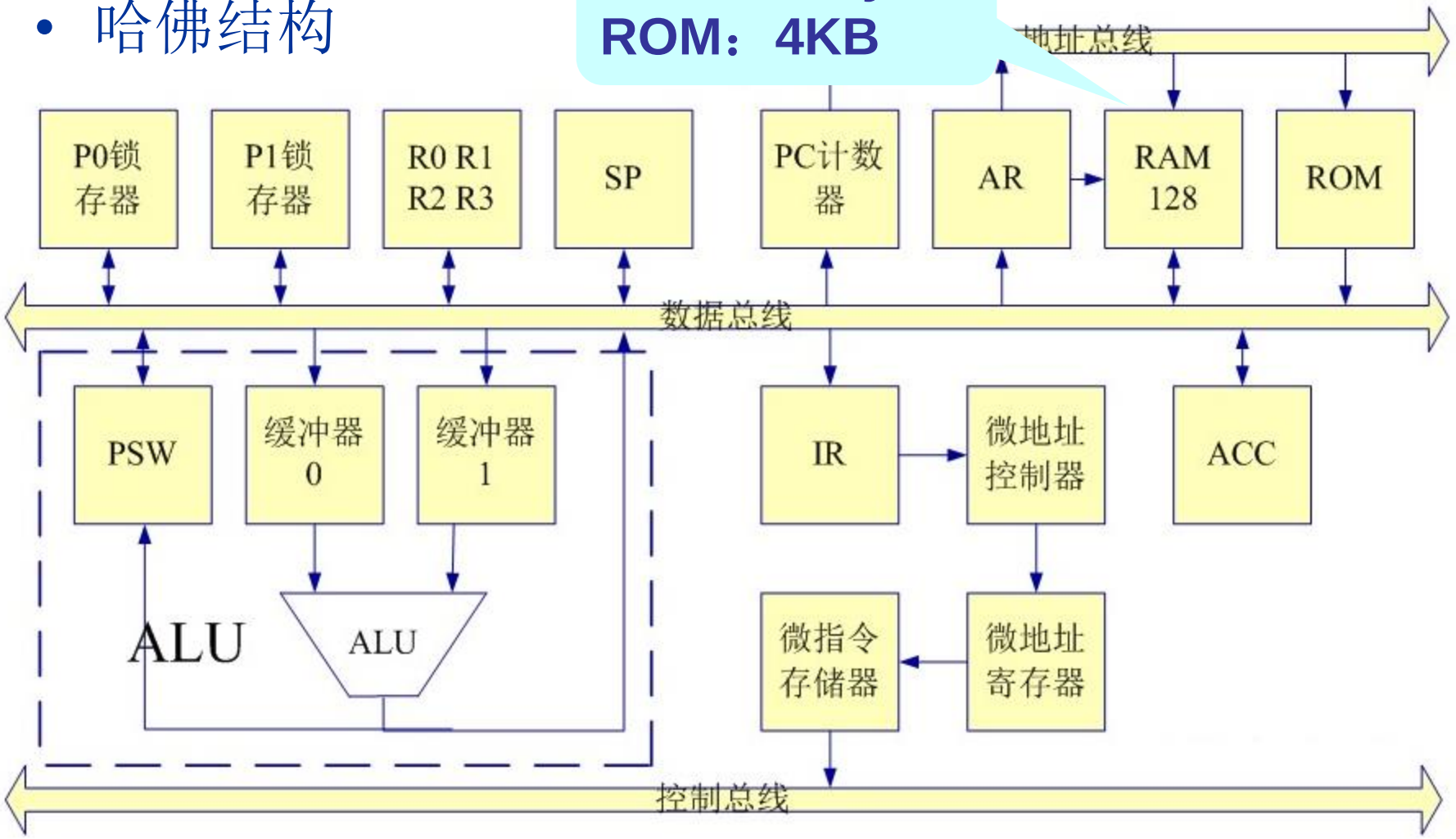
SoC重点

- CPU设计
  - IR、RN、PC
  - ALU、微控制器

8位CPU结构

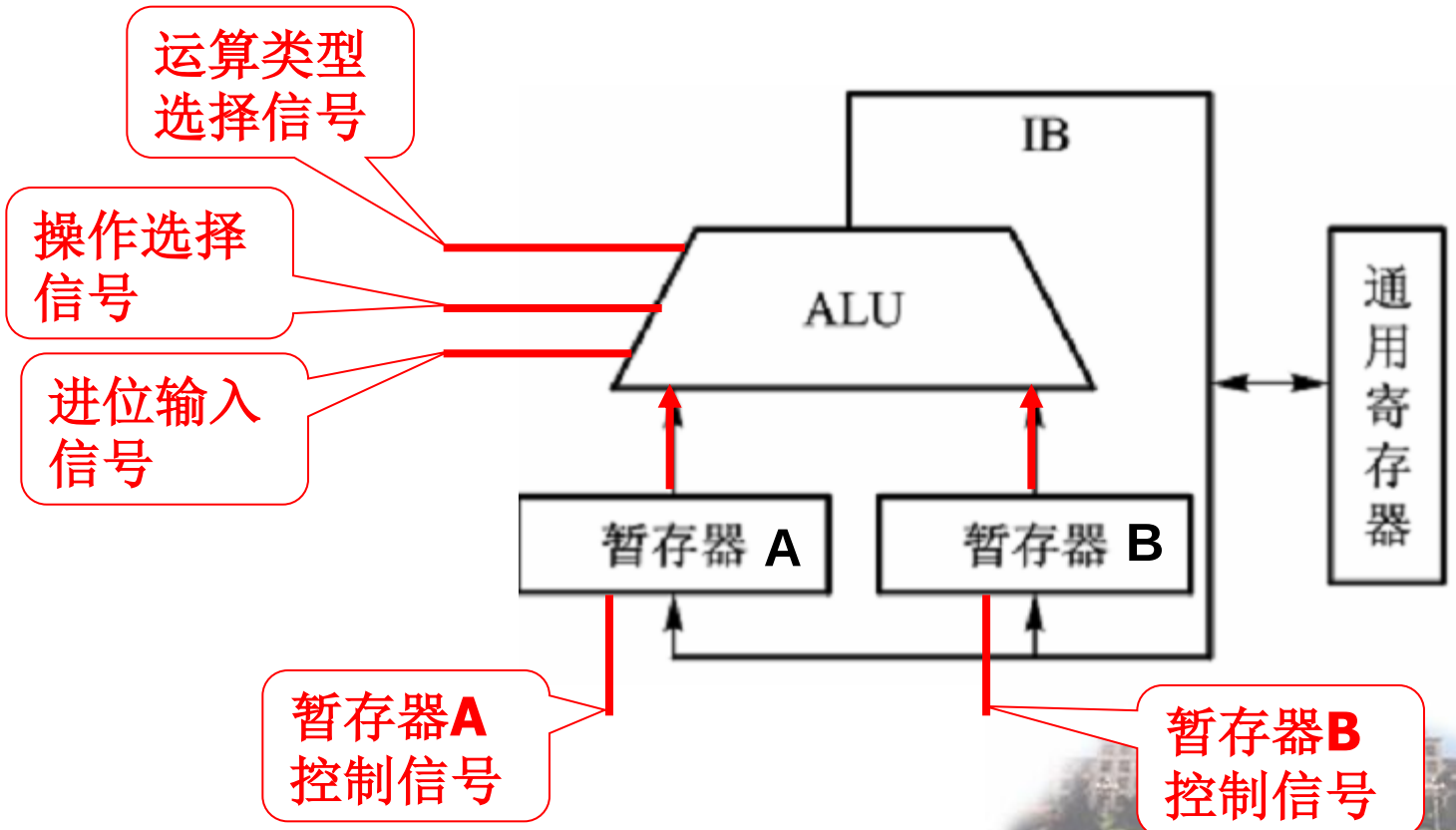
- 哈佛结构

RAM:128Byte
ROM：4KB



ALU结构

- ADD R0，R1；



互斥与相容

- 互斥的微操作：是指不能同时或不能在同一个节拍内并行执行的微操作。
- 相容的微操作：是指能够同时或在同一个节拍内并行执行的微操作。

- 把互斥的微操作组合在同一字段中，采用编码方式存取。
- 把相容的微操作组合在不同字段中，各段单独译码。

循环左移运算实现过程

- RL R0;

  指令编码为011 11000，即78H
- 取指操作

微操作	控制信号	功能说明
PC-->ADDR[11..0]	M_ROM; /ROM_EN	ROM片选信号有效， PC指向程序入口地址
BUS-->IR; PC + 1 -->PC	LDIR1; M_PC	IR使能，微指令通过总线传送到IR，PC+1。

循环左移运算实现过程

- 取操作数

微操作	控制信号	功能说明
R0-->BUS	RDRi， /Ri_EN	寄存器读使能，读信号有效，R0数据送到数据总线
BUS-->A	M_A	暂存器A使能，数据从总线输入到暂存器A

循环左移运算实现过程

- 执行指令

微操作	控制信号	功能说明
A-->ALU，	M=0； Cn=1； S3…S0=0000	暂存器数据送到ALU，直接输出到移位寄存器
ALU-->BUS	F1F0=10； /ALU_EN=0	循环左移运算，ALU输出使能，ALU运算结果输出到总线
BUS-->R0	M_Rn; WRRi	寄存器使能，写信号有效，数据通过总线写入寄存器R0

IR信号说明

- **生成PC的新地址**

  clk_IR 上升沿有效，LD_IR2高电平有效，
  data[3..0]→PC[11..8];

  clk_IR 上升沿有效，LD_IR3高电平有效，
  data[7..0]→PC[7..0]。

- **生成RAM的读写地址**

  clk_IR 上升沿有效，LD_IR3高电平有效
  data[7..0]→PC[7..0];

  nARen低电平有效，PC[6..0]→AR[6..0]。

JTAG接口

JTAG(Joint Test Action Group,联合测试行动小组)

- TCK为测试时钟输入
- TDI为测试数据输入
- TDO为测试数据输出
- TMS为测试模式选择
- /TRST为测试复位，输入引脚，低电平有效。

- 祝大家考出好成绩!