第6章 数字系统

6.1 数字系统的基本概念

6.2 数据通路

6.3 由顶向下的设计方法

6.4 小型控制器的设计

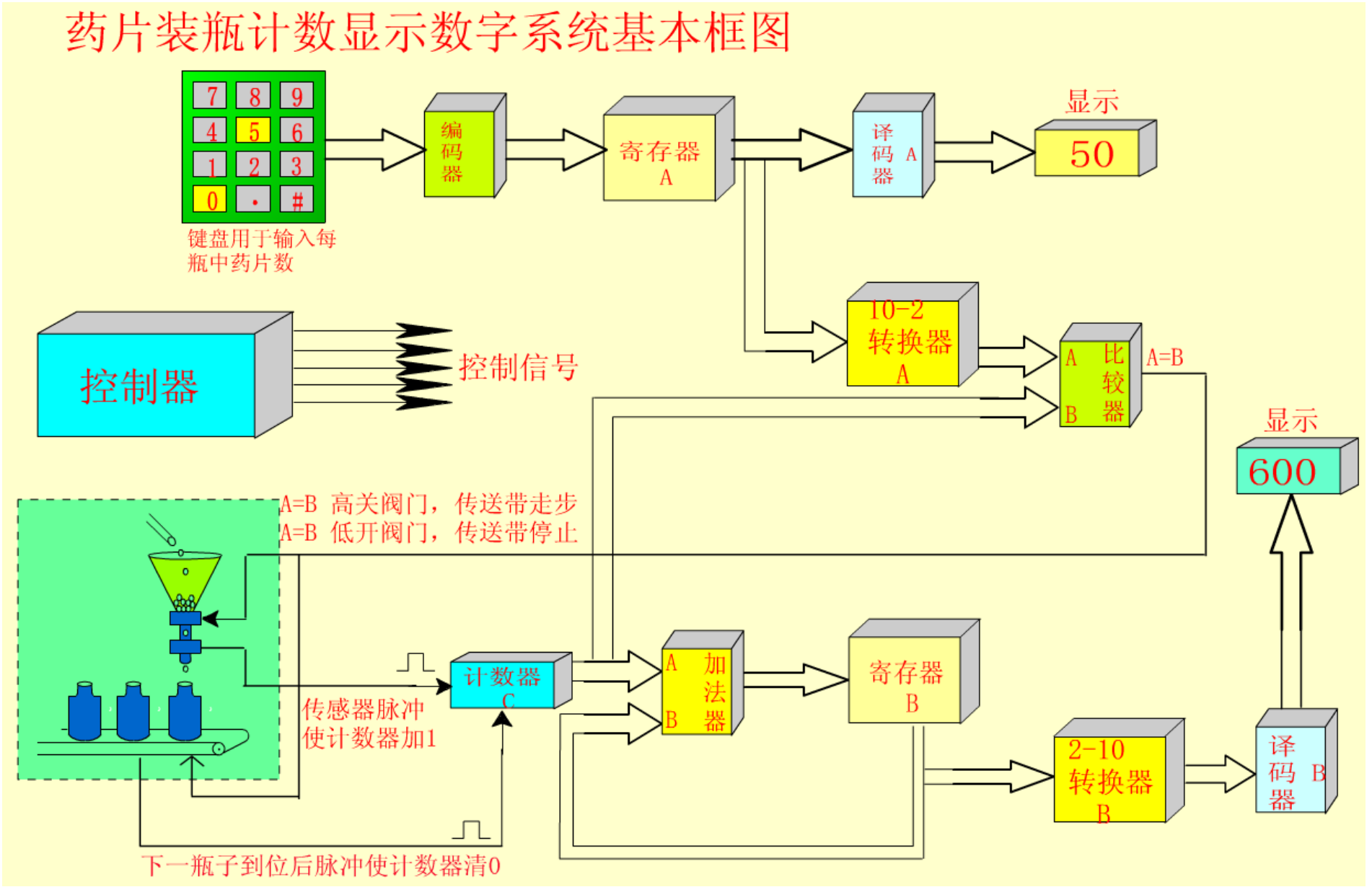
6.5 数字系统设计实例

返回目录

6.1 数字系统的基本概念

6.1.1 一个数字系统实例
6.1.2 数字系统的基本模型
6.1.3 数字系统与逻辑功能部件的区别

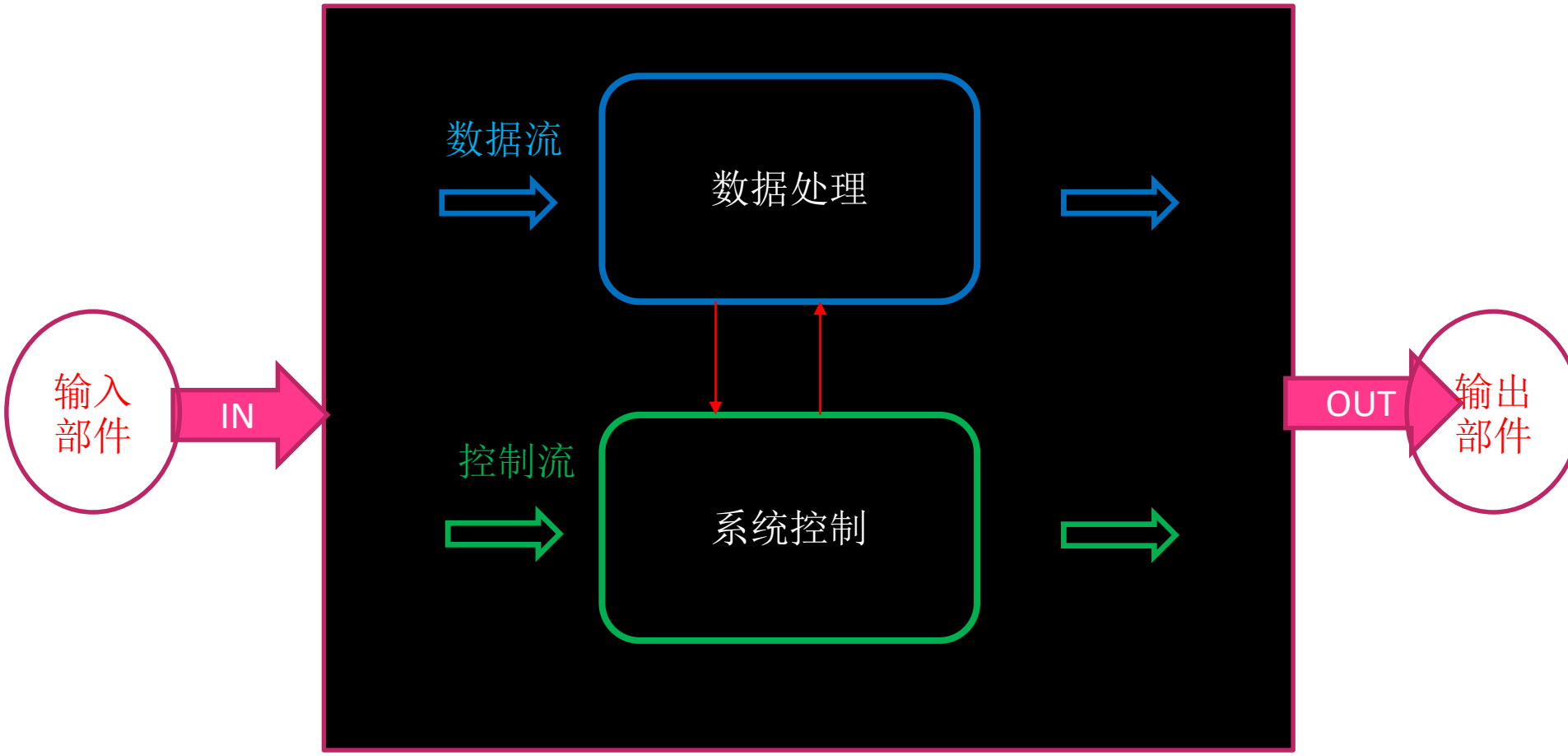
6.1.1 一个数字系统实例



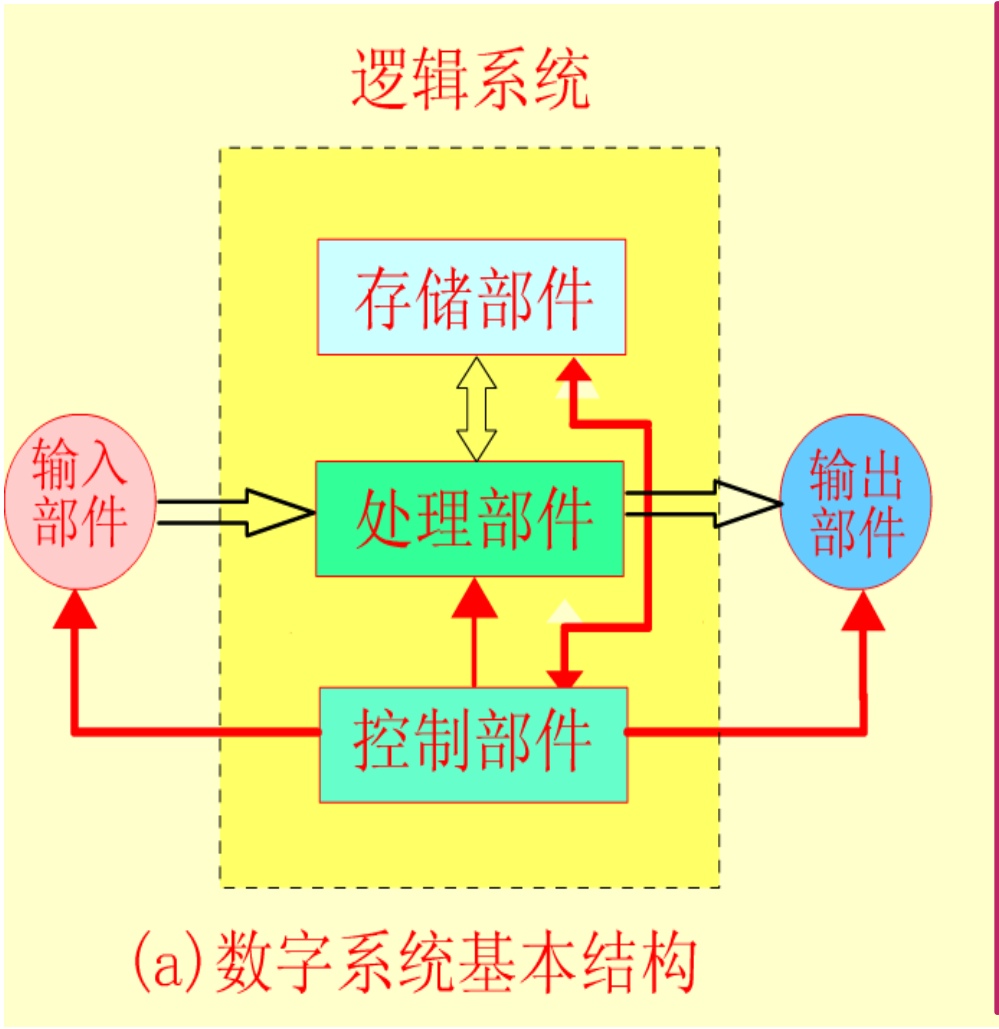
6.1.2 数字系统的基本模型

稳定、精准、可靠、模块化

6.1.2 数字系统的基本模型

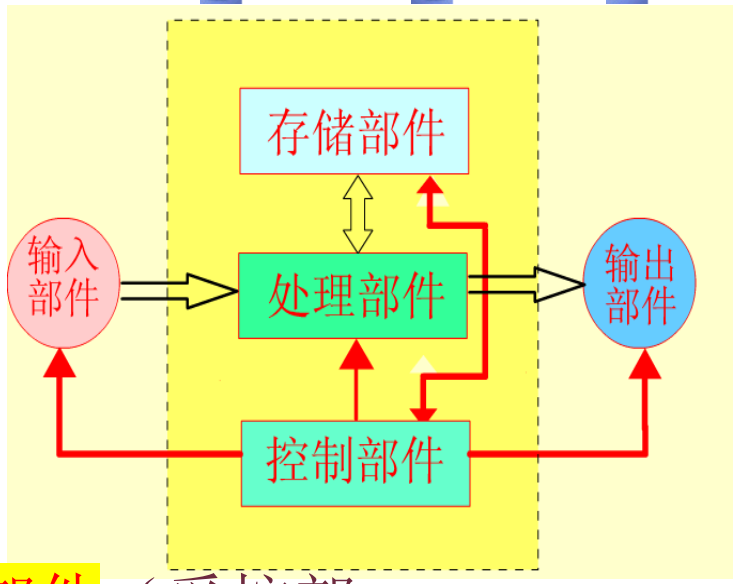


6.1.2 数字系统的基本模型



(a) 数字系统基本结构

数字系统基本模型



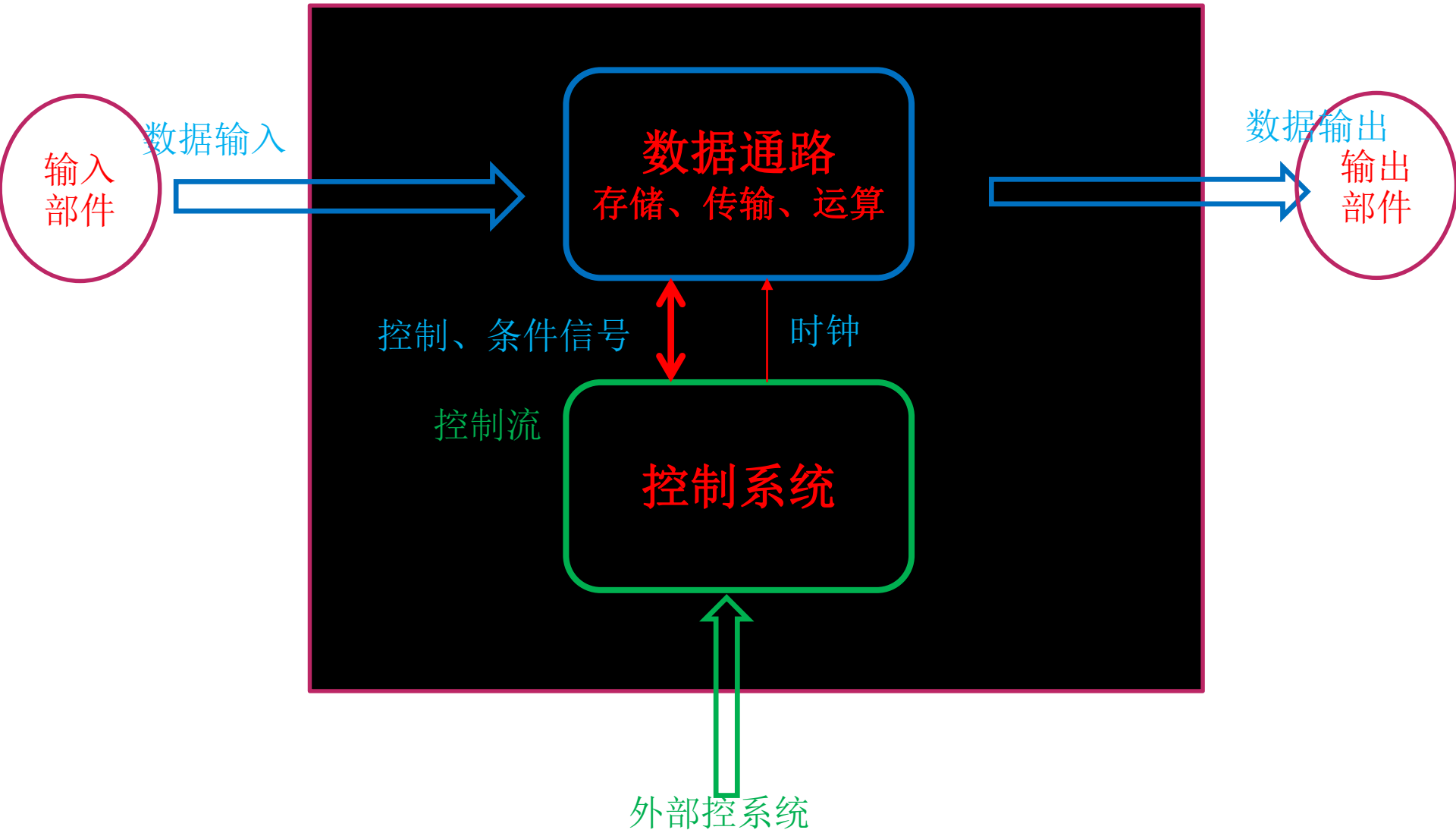
输入部件：收信息，外部。

输出部件：处理结果，外部。

存储部件和**处理部件**统称为执行部件（受控部件）。**执行部件**接受控制信号，进行数据运算、数据存储、输出。处理中产生状态信息反馈到控制单元。

控制部件称为控制单元。控制单元根据外部输入和反馈状态信息，决定下步操作。向执行部件发送控制信号，决定何时、何地、进行何种数据运算。

组成



6.1.3 数字系统与逻辑功能部件的区别

一、有无控制部件

数字系统通常由若干个逻辑功能部件组成，由一个控制器统一指挥

部件（子系统）一般就是一个逻辑功能部件，诸如加法器、乘法器、译码器寄存器堆、存储器等，它们都是典型的逻辑功能部件，可称为逻辑子系统。逻辑功能部件一般不带有控制器

二. 设计方法

数字系统级的设计和**逻辑部件级**的设计是沿不同途径进行的。

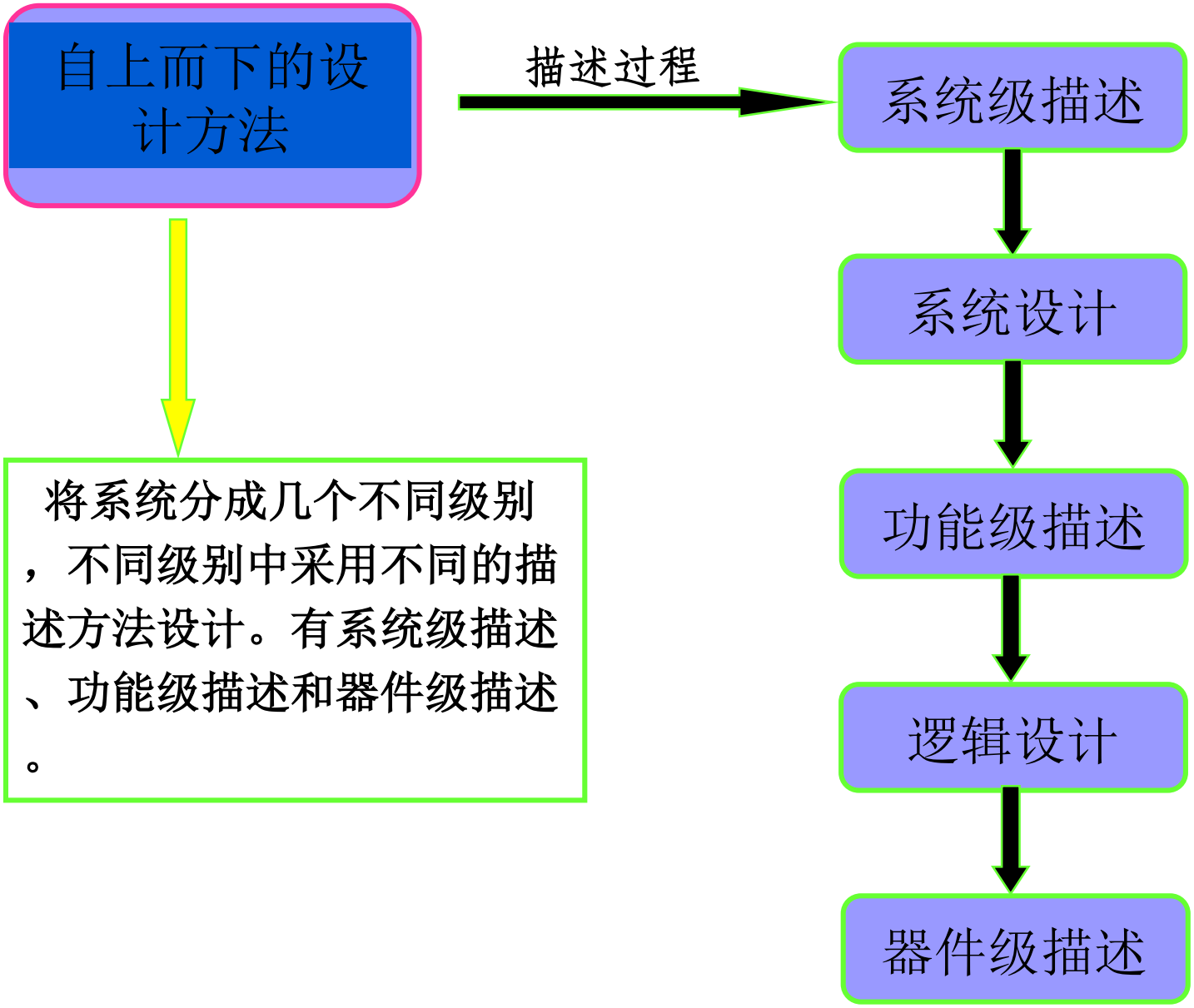
逻辑部件的设计采用自下而上的设计方法:

先按任务要求建立真值表或状态表，给出逻辑功能描述，然后进行逻辑化简或状态化简，最后一举完成逻辑电路的设计。

数字系统的设计方法，是一个自上而下的设计方法，又称为由顶向下的设计过程。

系统级设计→功能级设计--> 器件级设计

自上而下的设计方法


自上而下的设计方法
描述过程
系统级描述
系统设计
功能级描述
逻辑设计
器件级描述
将系统分成几个不同级别，不同级别中采用不同的描述方法设计。有系统级描述、功能级描述和器件级描述。

自上而下设计步骤

（1）逻辑抽象,分析,　将系统划分为若干子系统.

（2）系统方框图设计（系统初步设计）。确定实现系统功能算法。

（3）设计控制器，以控制各子系统工作。

(4)　设计子系统(逻辑电路图设计）。

数字系统设计方法概述

- 基于通用逻辑器件：硬件
- 以微控制器为核心：软件
- 专用集成电路(ASIC)设计：硬件
- 以PLD为基础：硬件软件化
- 以片上系统(Soc)为基础：软硬件协同

6.2 数据通路

1、运算、存储基本子系统

2、传输：总线

3、 数据通路实例

1、运算、存储等基本子系统

基本子系统是数字系统最基本的逻辑功能部件。包括算术逻辑运算单元ALU、寄存器堆、 RAM。

- 算术逻辑运算单元ALU
- 加减法器
- 寄存器堆
- 存储器 RAM

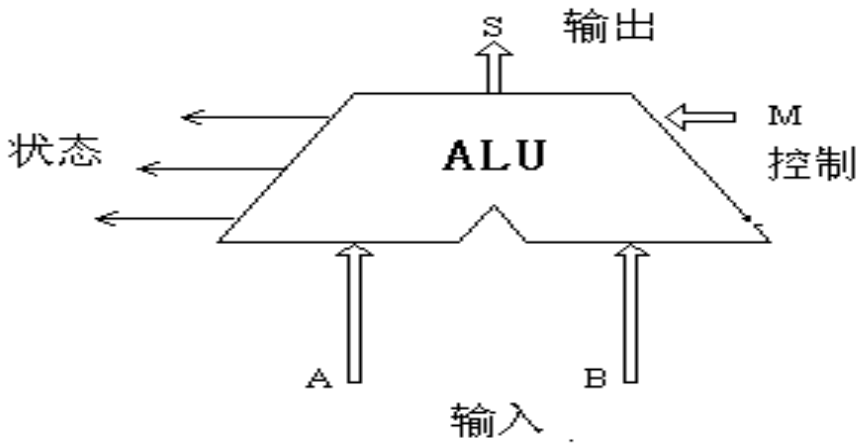
算术逻辑运算单元ALU

ALU的功能

- 算术、逻辑运算，对数据进行加工处理的功能部件。
- 没有ALU，就不成为复杂的数字系统。

ALU的结构

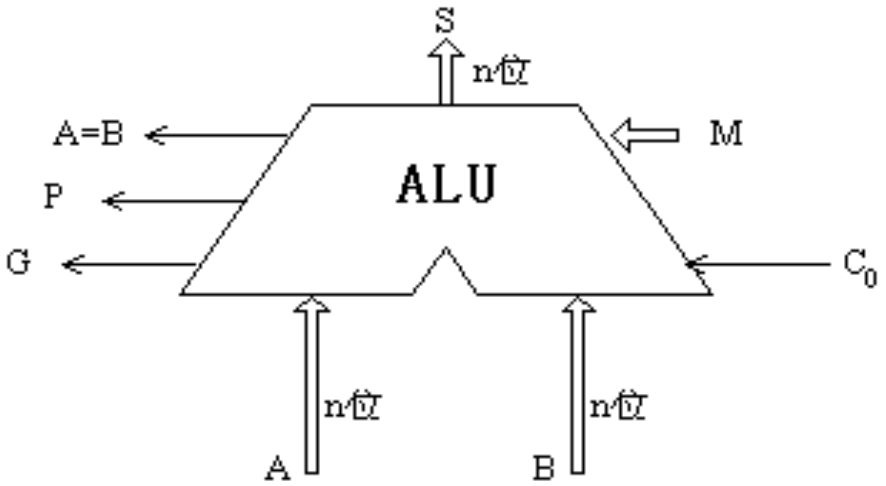
- 数据输入
- 数据输出
- 反馈信息
- 控制信号



算术逻辑运算单元ALU

ALU为多功能运算部件，实现多种算术和逻辑运算。

ALU一般结构



M为控制器参数。根据M的位数，可设计多种运算功能。

算术逻辑运算单元

芯片AM2901中，M有3位，ALU有3种算术运算，5种逻辑运算。

M_2	M_1	M_0	功能操作
0	0	0	A 加 B
0	0	1	B 减 A
0	1	0	A 减 B
0	1	1	A+B
1	0	0	A · B
1	0	1	$\overline{A}$ · B
1	1	0	A ⊕ B
1	1	1	$\overline{A}$ ⊕ $\overline{B}$

•控制参数M为n，实现2^n种算术或逻辑运算。

加（减）法器

A、B：参加运算的两个数（n位）；

S：求和结果。

ASC 为控制信号；

ASC=0，加法运算。

ASC=1，减法运算。

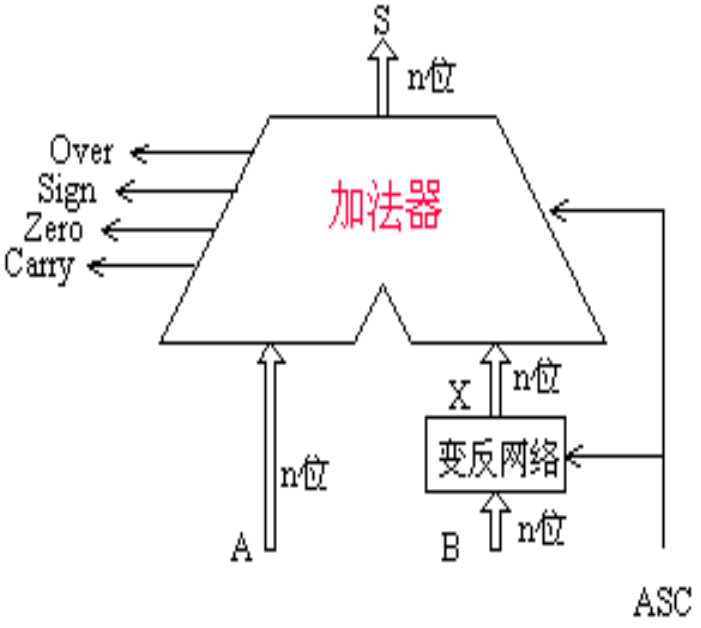
变反网络将减法转化为加法。

Over--- 溢出标志

Sign---运算结果符号（正或负

Carry--- 最高位的进位输出信

Zero---求和结果S=0的标志信号



运算产生的特殊状态信号用标志触发器保存。为控制器使用。

寄存器堆

ALU无记忆功能。

运算数和结果均需要寄存器（堆）保存。寄存器堆是系统中不可缺少的子系统。

分类

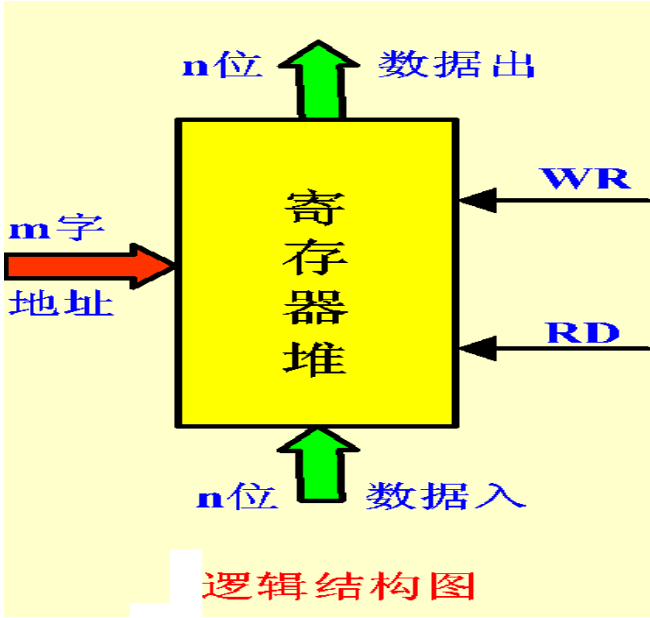
1）通用寄存器 暂存参与ALU运算的数据或结果。数目一般是4、8、16、32个或更多。

2）专用寄存器 保存运算中需要或产生的某些特殊信号。如ALU的状态标志信号寄存器、地址寄存器、指令寄存器、程序计数器等。

专用寄存器根据不同系统而不同。

通用寄存器

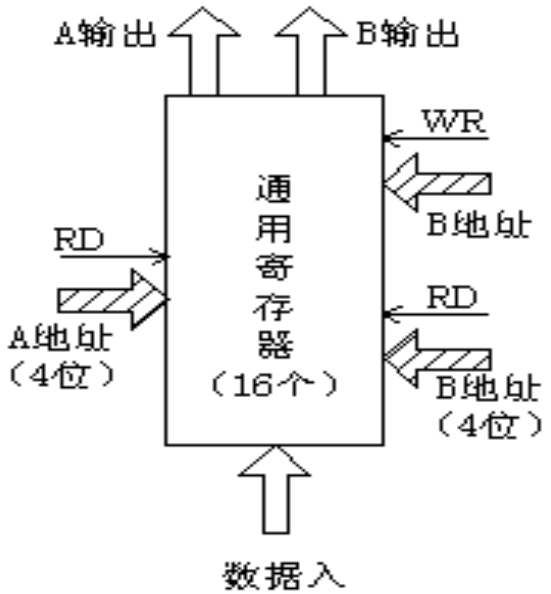
通用寄存器结构（单端口输出）



逻辑结构图

通用寄存器结构（双端口输出）

- 组成：寄存器堆，地址寄存器多路开关 和数据选择器。
- 功能：A、B读数和B写数。

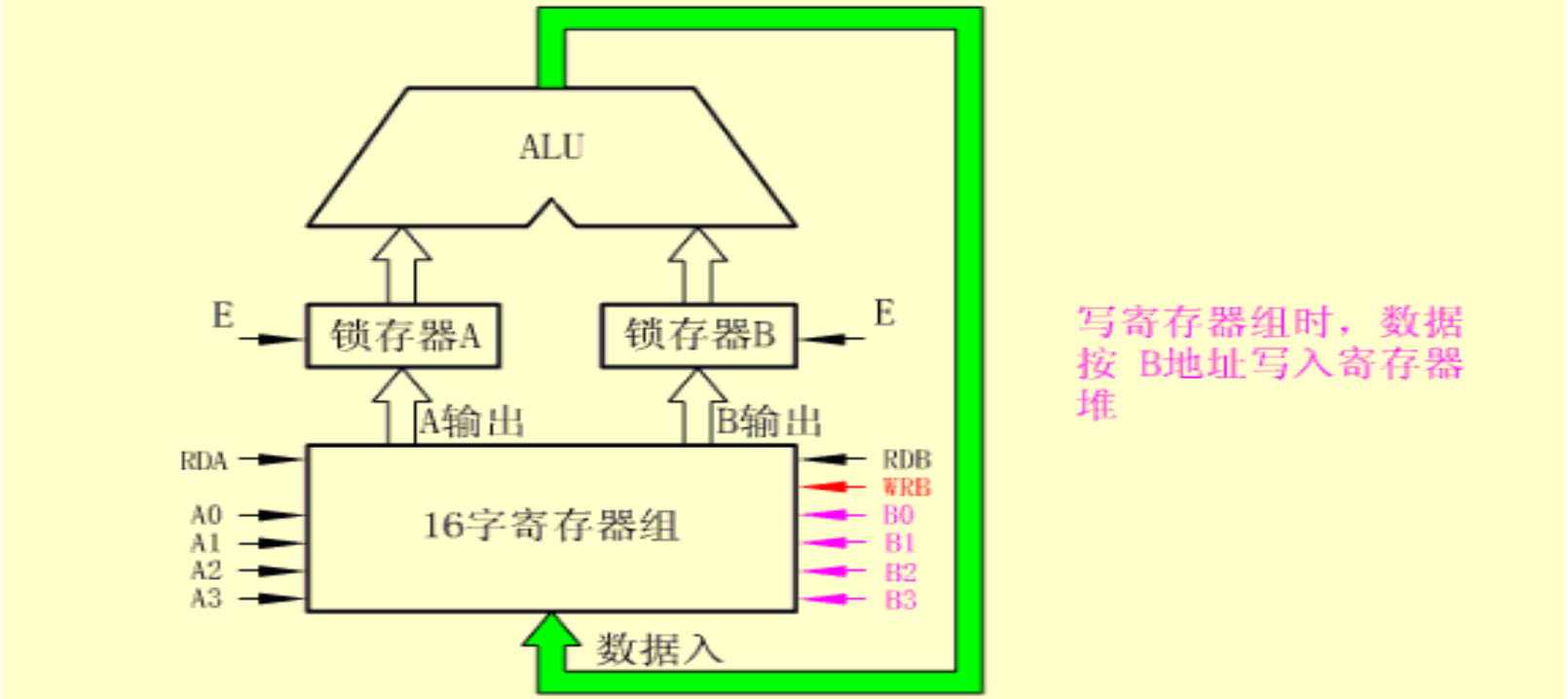


双端口输出寄存器堆

与ALU构成运算器。ALU和寄存器堆间锁存器作为缓冲。

寄存器读出：控制信号RDA(B)有效，A和B地址指定的两个寄存器数据分送到端口A和端口B。

寄存器写入：WRB有效时，待存入数据按B地址指定的寄存器写入（寄存器堆）。



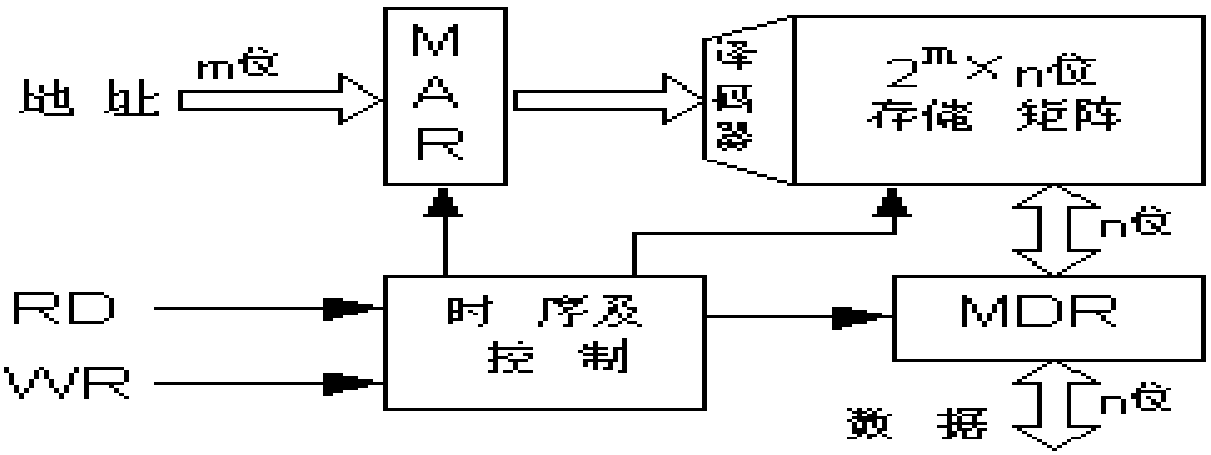
存储器RAM

组成：地址寄存器MAR、数据缓冲器MDR、存储矩阵、RW电路。

读操作：读数控制信号RD有效，存储单元内容读到MDR，再送到数据总线。

写操作：写数控制信号WR有效，写入数据由数据总线送到MDR寄存器，再写入存储矩阵。

读写操作分时进行。位数据线具有双向传送功能。

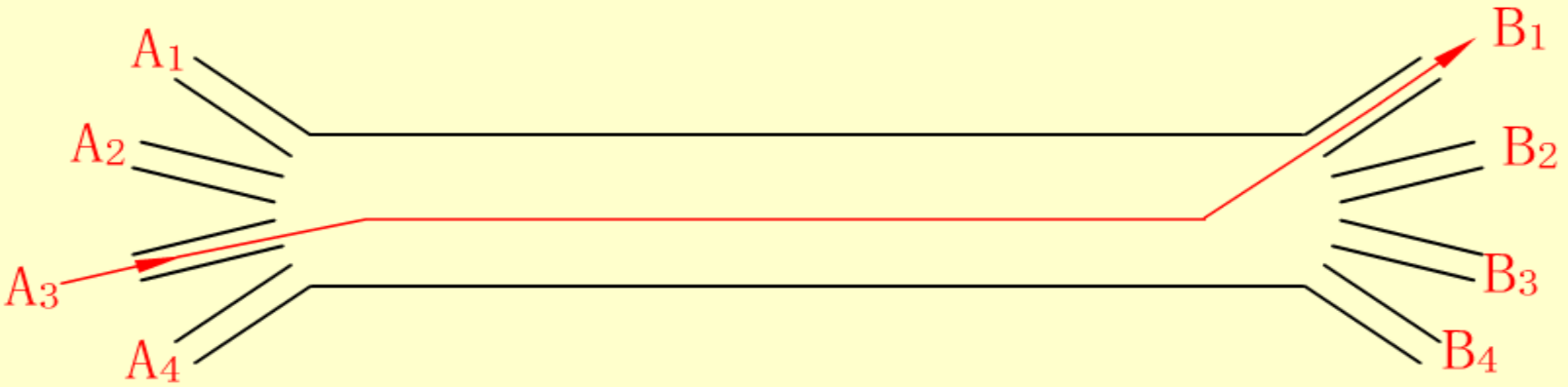


2、总线结构

1. 总线的概念

总线:多个信息源分时传送数据流到多个目的地的传输通路。

总线原理示意图

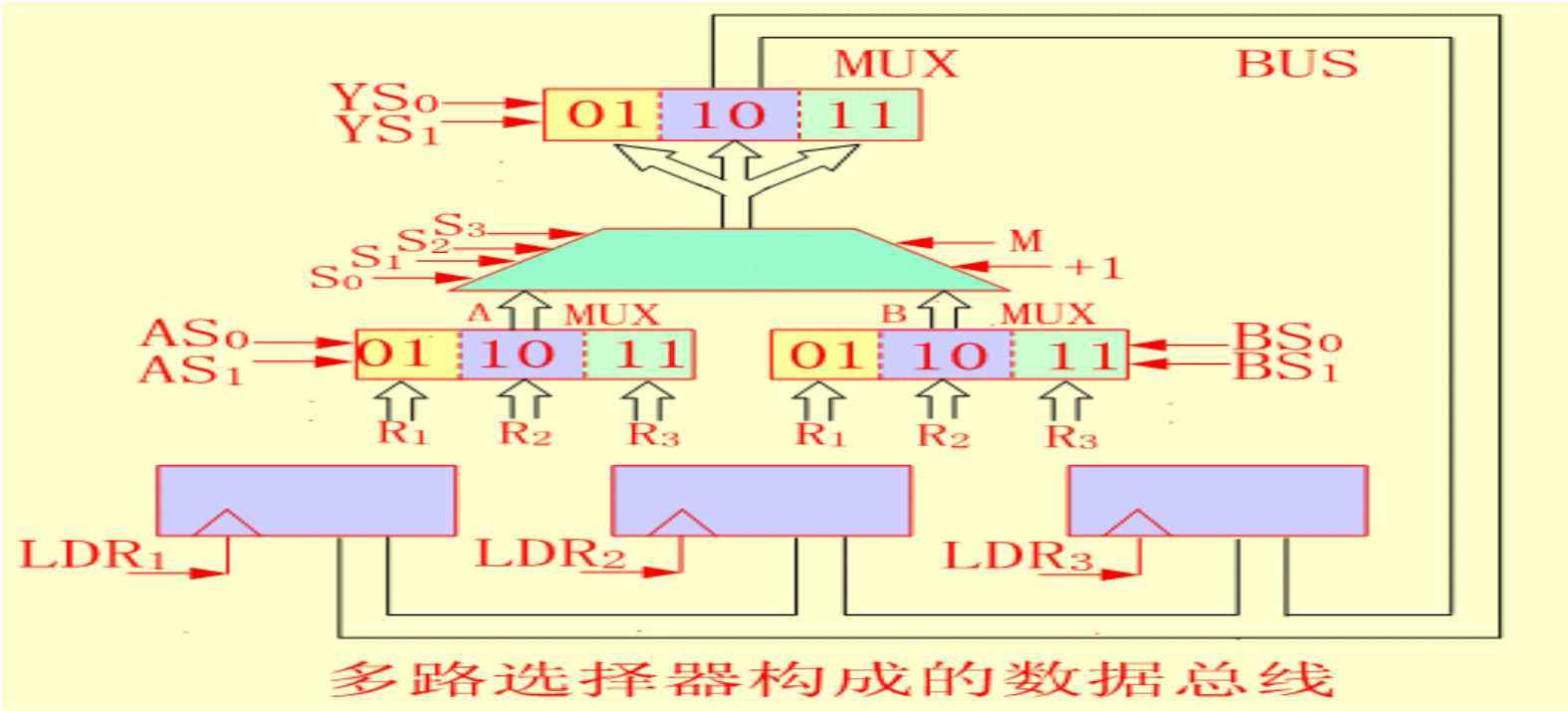


数字系统中多采用双向总线。

双向总线：信息的源端和目的端是相对的，即可以实现信息的双向传送。

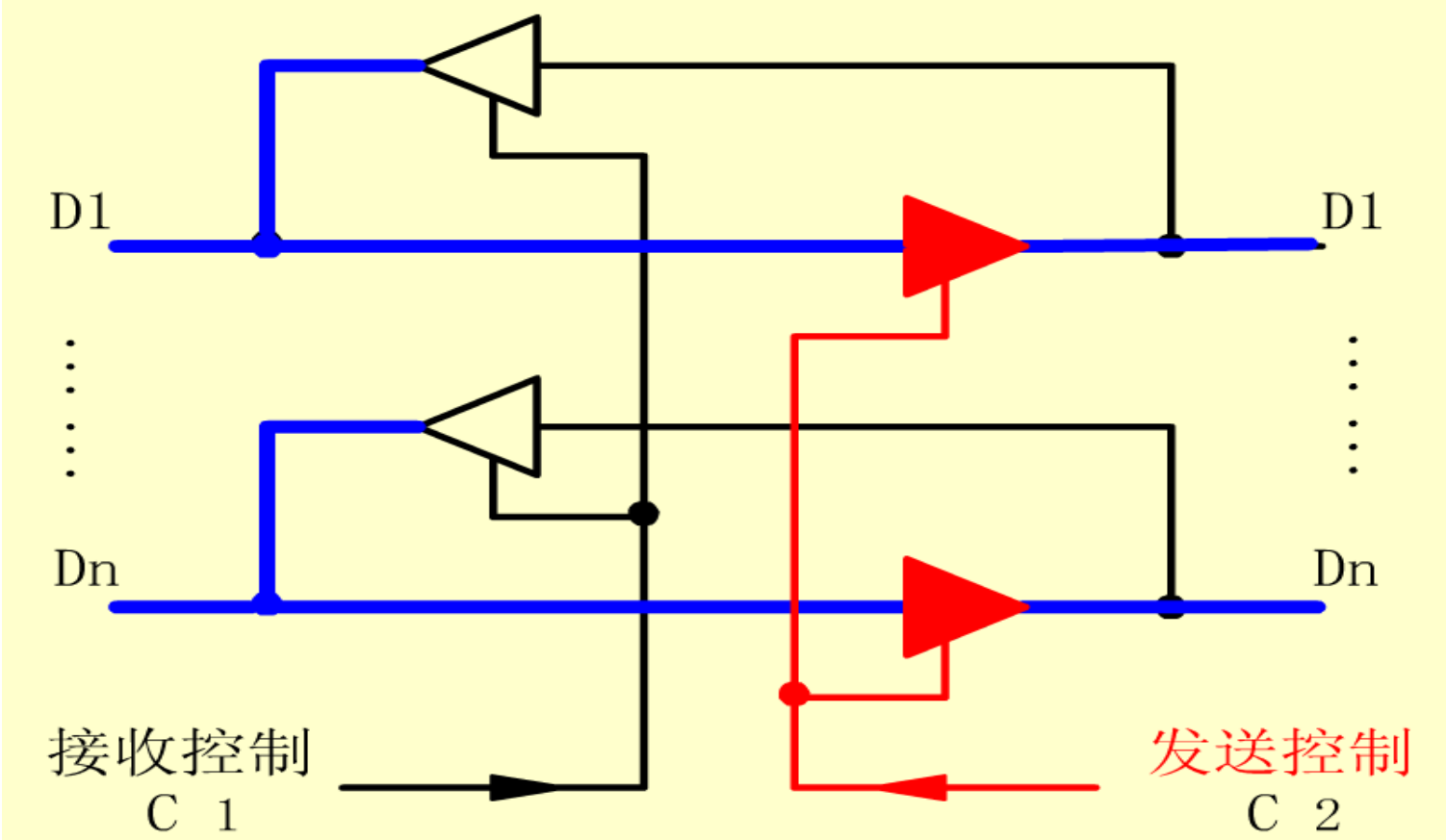
逻辑实现

- 多路选择器方式（适合单向总线）

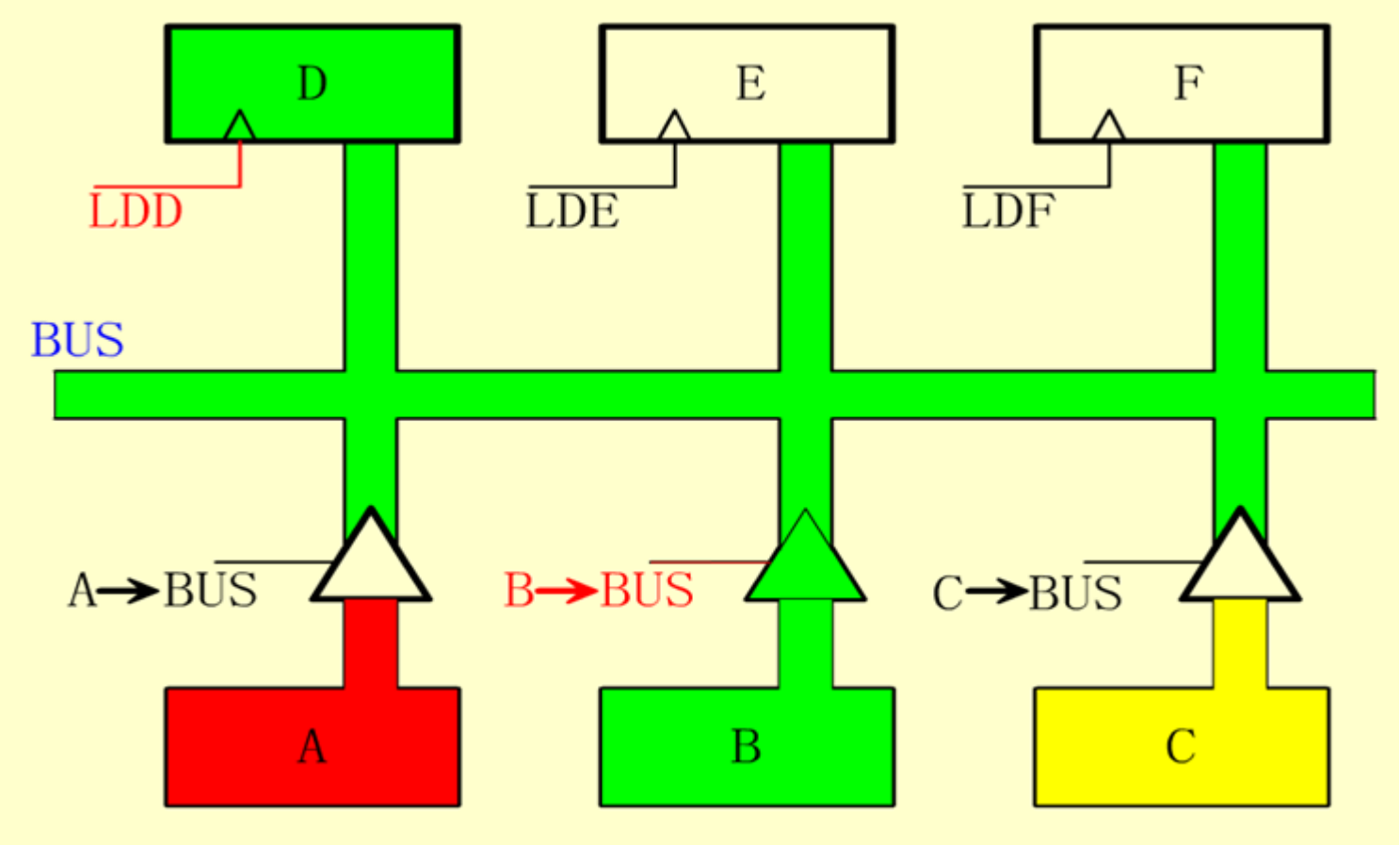


- 三态门方式(适合单向、双向总线)

- 三态门



双向数据总线



三态门构成的数据总线

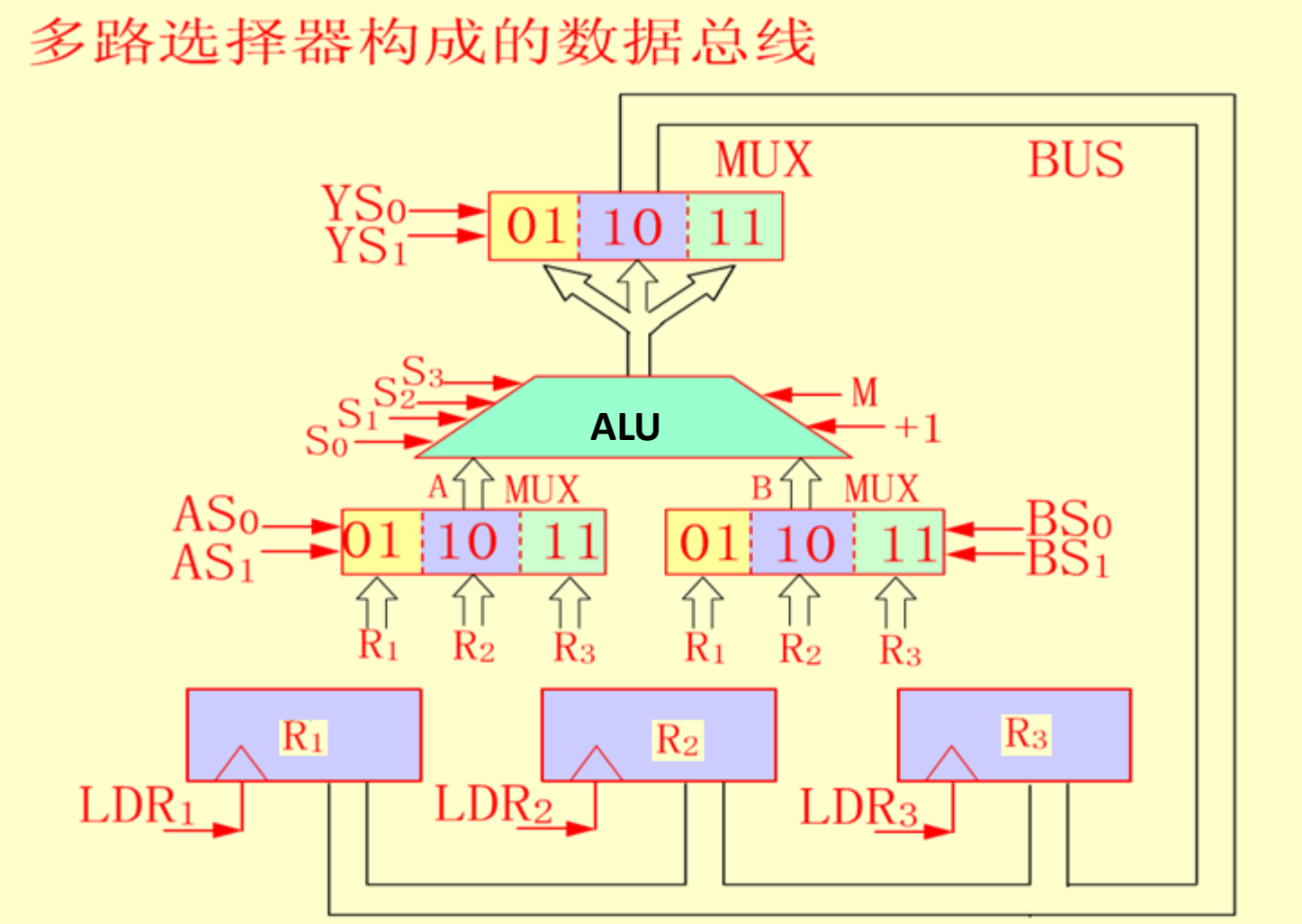
3、数据通路实例

数字系统中，各个子系统通过数据总线联结形成的数据传送路径称为数据通路。

1. 数据通路设计非常重要，不仅影响控制器设计，也影响数字系统性能指标（速度和成本）

   处理速度快的数字系统，要求独立传送信息的数据通也多。但数据通路增加将使控制器复杂。
2. 在满足速度指标前提下，数据通路应尽可能简单。
3. 小型系统中多采用单一总线结构。较大系统可采用双总线或三总线结构。

6.2.2 数据通路实例


多路选择器构成的数据总线
MUX
BUS
YS0
YS1
01
10
11
S3
S2
S1
S0
ALU
M
+1
A
MUX
B
MUX
AS0
AS1
BS0
BS1
R1
R2
R3
LDR1
LDR2
LDR3

单总线中的数据通路及数据流分析

单总线结构的数据通路实例，子系统如下：

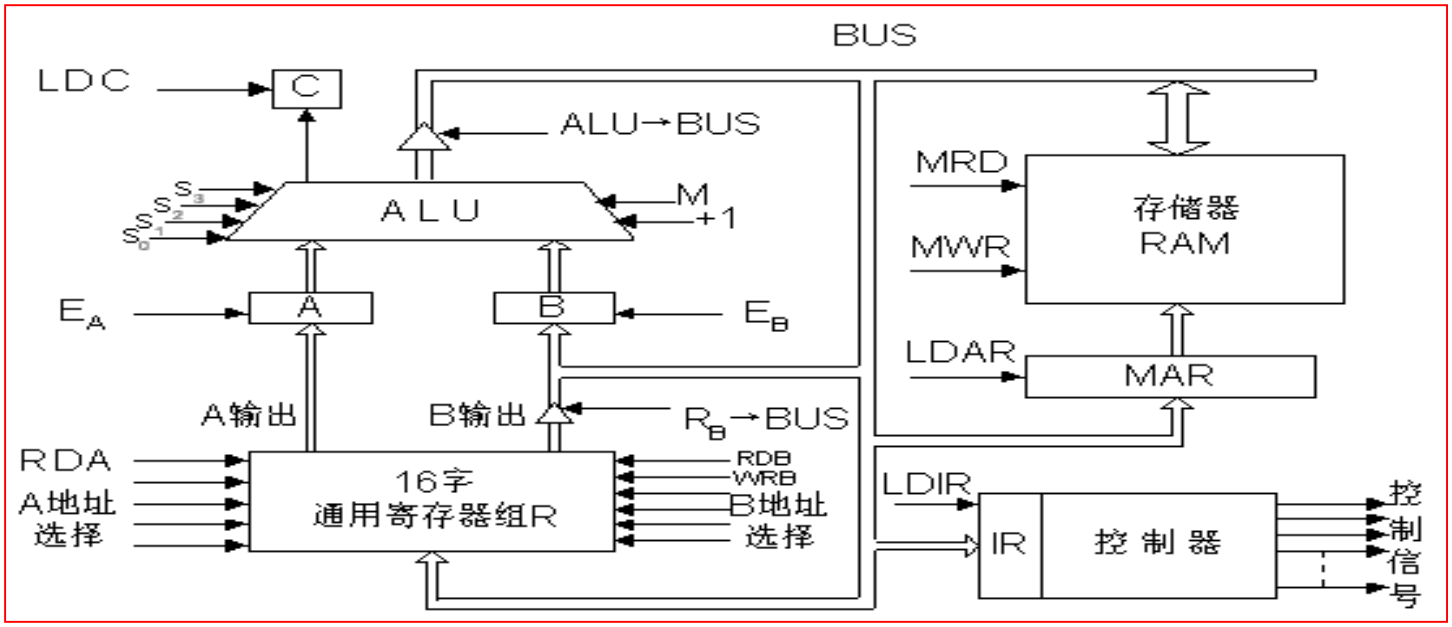
通用寄存器组R（双端口）：存BUS上来的数据。暂存器A和B。

ALU：S3、S2、S1、S0、M五个控制端，选择运算类型。 寄存器C保存ALU运算产生的进位信号。

RAM：受读/写控制信号控制。 MAR是地址寄存器。

BUS：单数据总线，通过三态门与有关子系统连接。

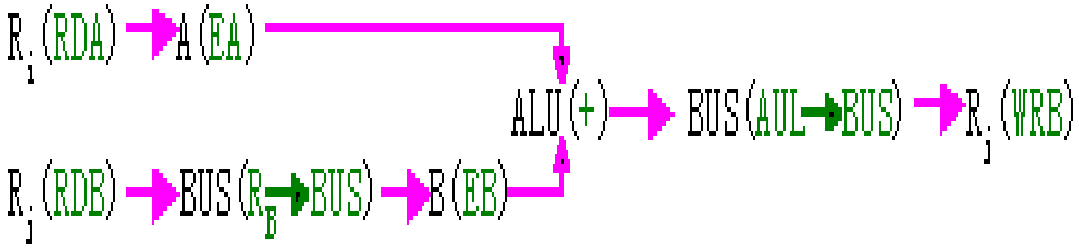
控制器：产生数据通路中控制信号(单线），控制各子系统。

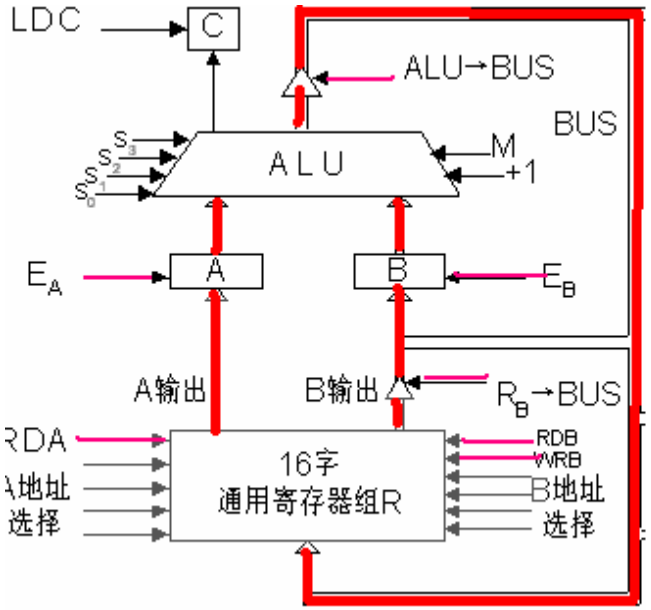


【例1】说明Ri+Rj--->Rj操作的数据流。

读出Ri和Rj的数据并送至ALU进行相加，结果送回Rj。

数据流为：





→表示数据流方向，括号内绿色字母表示控制信号。

RDA：以A地址读通用寄存器Ri。

RDB：　以B地址读通用寄存器Rj。

WRB：以B地址写通用寄存Rj。

【例2】说明Rj--->RAM操作的数据流。

•寄存器Rj中的数据送往RAM中保存。

数据流：Rj(RDB)→BUS(RB→BUS)→RAM(MWR)

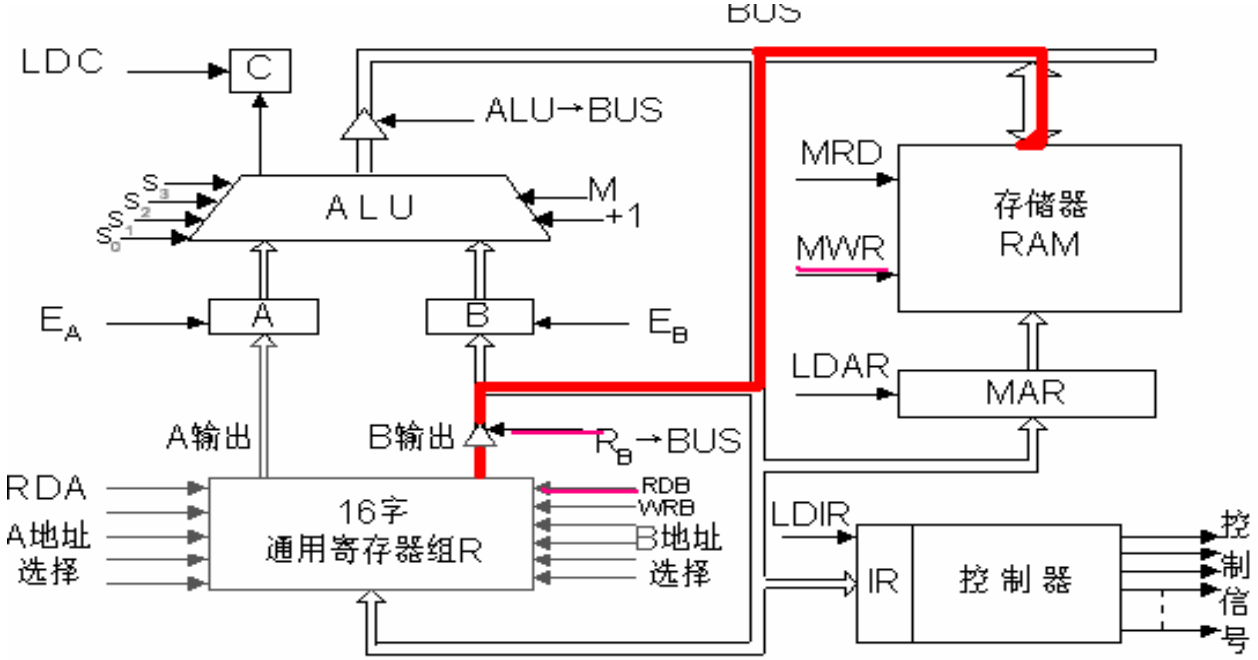
MWR：表示将BUS来的数据写入RAM。

•实现上述数据流，须先向MAR送入地址码，假设由Ri 经 A端口给出，即存在另一数据流：

Ri （RDB） → A （EA）→BUS(ALU→BUS)→MAR(LDAR)；

LDAR：将暂存器B给出的地址码打入到MAR。

•地址和数据信息都占用BUS，需分时进行。



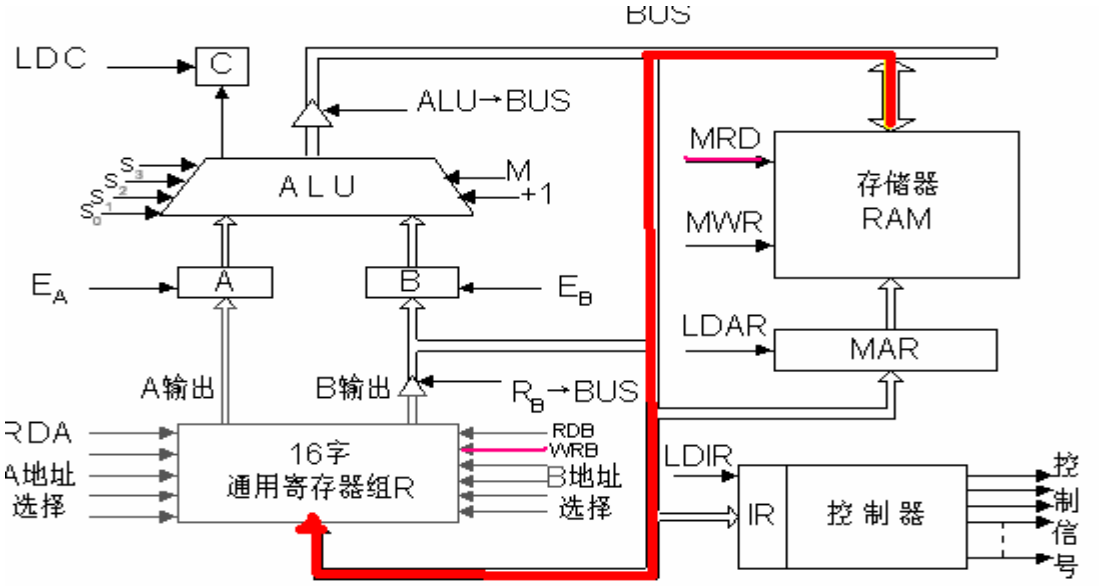
【例3】说明RAM--->Rj操作的数据流。

- 把RAM中的数据，送到通用寄存器Rj。
- 数据流：　　RAM(MRD)→BUS→Rj(WRB)

MRD：RAM的读出命令。

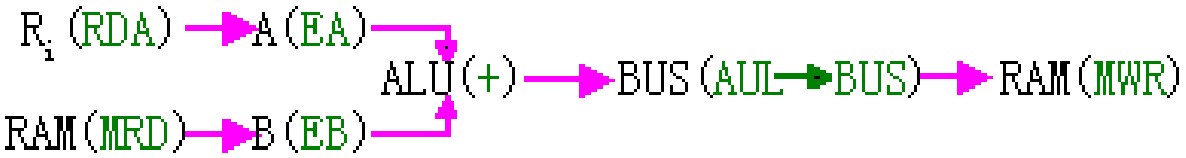
WRB：寄存器Rj的写命令。

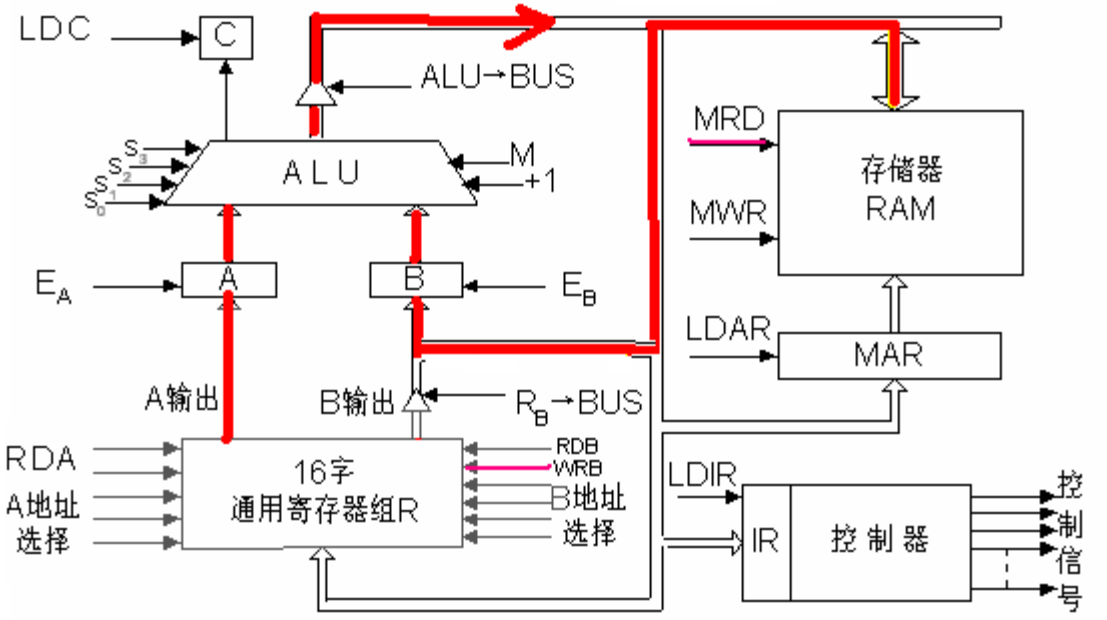
•同样，应先向MAR送入地址。与上述数据流操作分时进行。



【例4】说明R_i+RAM--->RAM操作的数据流。

- 一个运算数据来自R_i，一个来自RAM，在ALU中运行加法运算后又存入到RAM中。
- 数据流为（假设RAM的地址码已经放在MAR中）。





单总线中数据通路及数据流分析

四种数据流处理时间和路径与数据通路有关。数据流通过BUS和数据通路在各子系统间进行流动。

数据通路不同，控制信号(由控制器产生)也不同。

子系统通过三态门与总线连接简单方便。易扩充，BUS上直接增加三态门和子系统即可。

为改善数据通路和系统性能,可采用双端口寄存器输出。

数据通路设计非常重要,不仅影响控制器设计，也影响数字系统性能指标

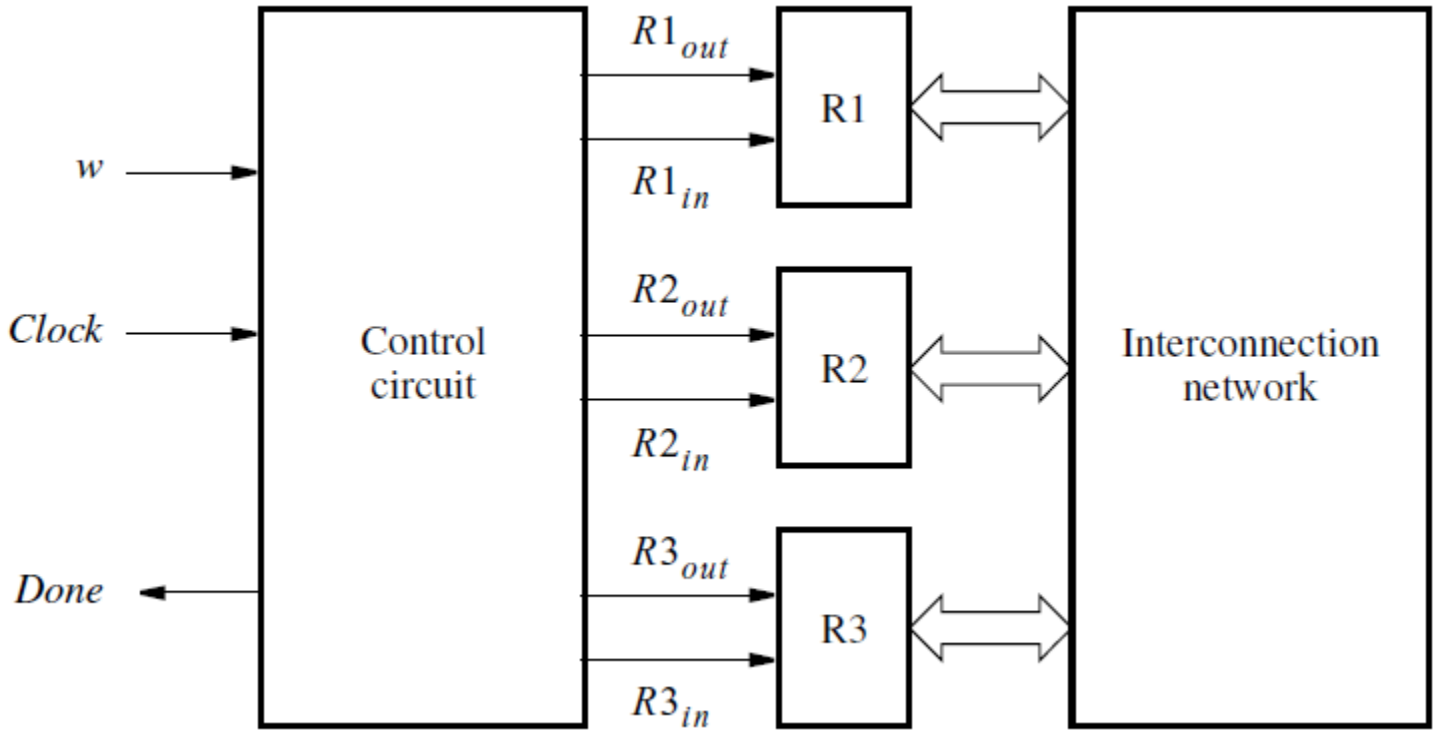
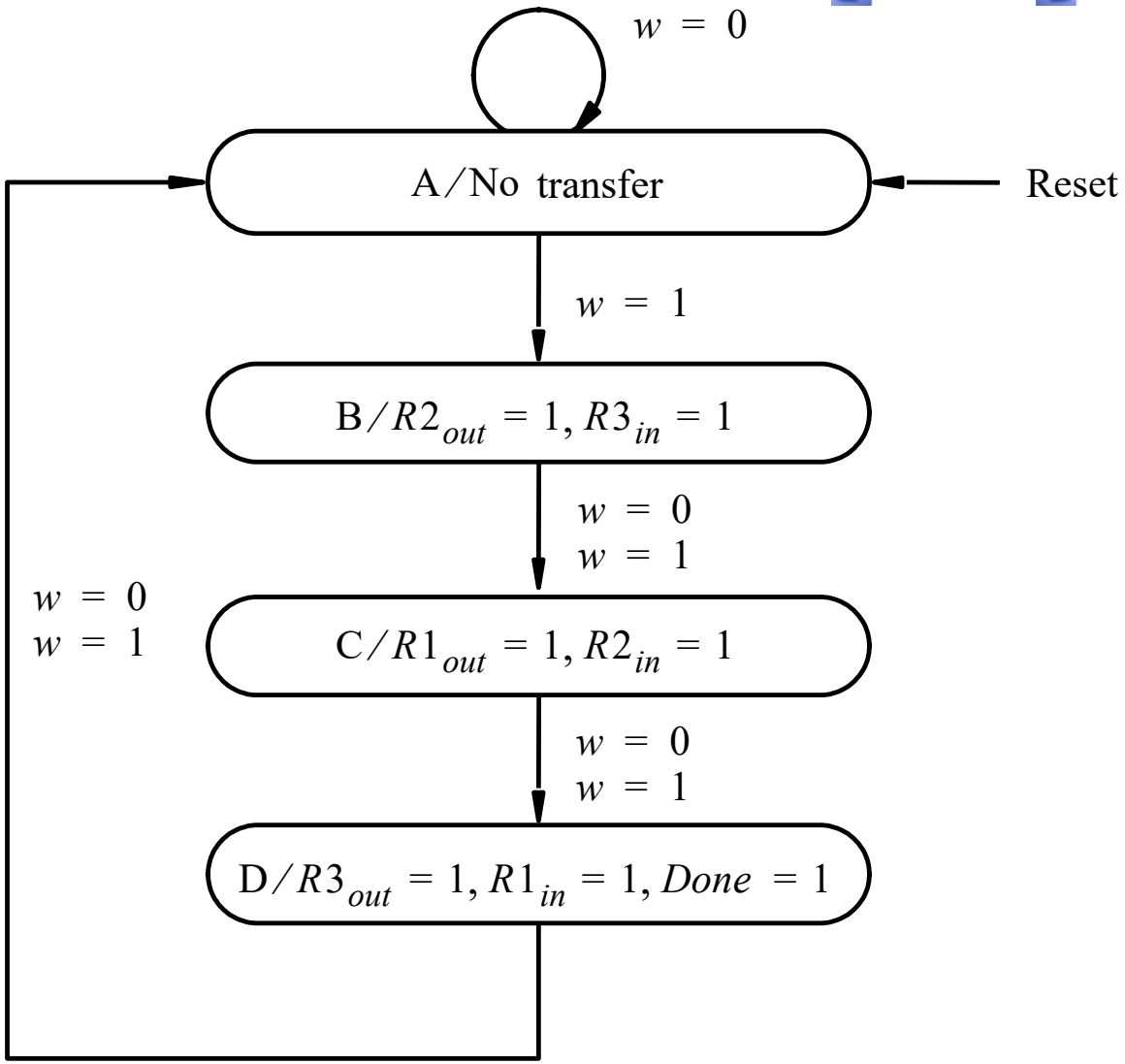
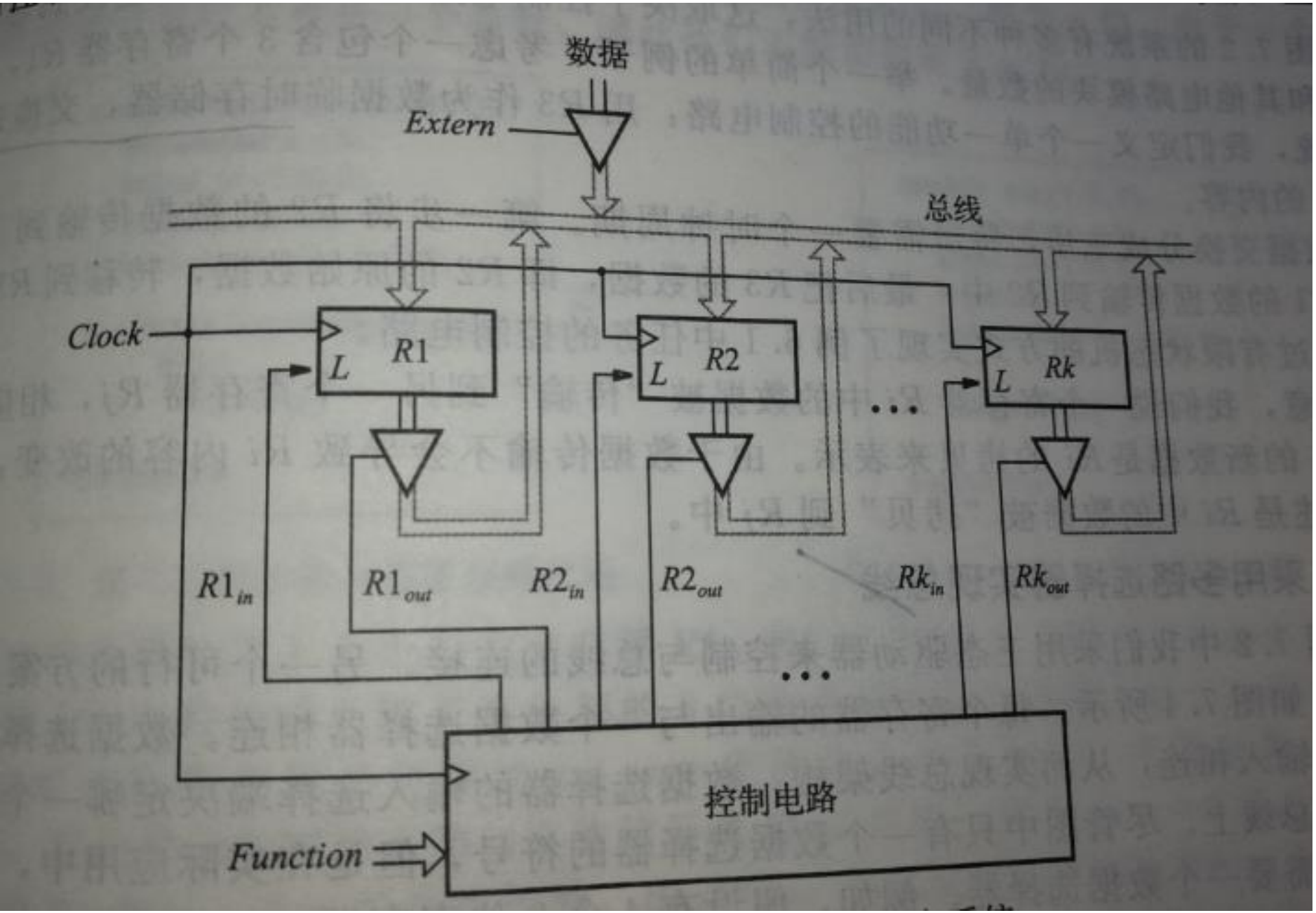
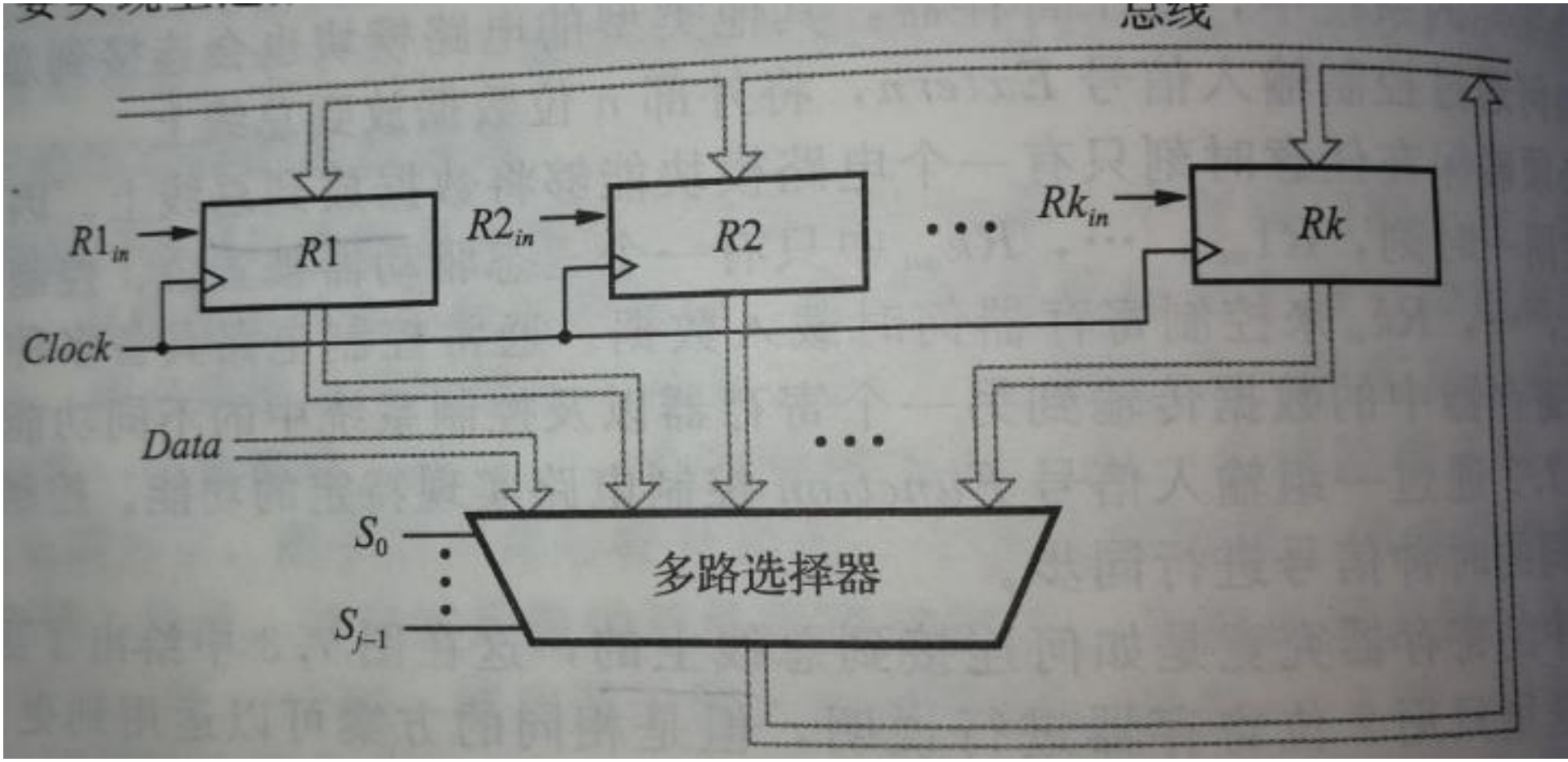


Figure 6.10. System for Example 6.1.



State diagram for Example 6.1.


数据
Extern
总线
Clock
L
R1
L
R2
L
Rk
...
R1in
R1out
R2in
R2out
Rkin
Rkout
...
控制电路
Function


总线
$R1_{in}$
R1
$R2_{in}$
R2
Rk_{in}
Rk
Clock
Data
S_0
S_{j-1}
多路选择器

```
module trin(Y,E,F);
        parameter n=8;
        input [n-1:0] Y;
        input E;
        output wire [n-1:0]F;

        assign F=E?Y:'bz;
endmodule
```

```
module regn(R,L , Clock,Q);
        parametern=8;
        input [n-1:0]R;
        input L,Clock;
        output reg [n-1:0] Q;

        always @(posedge Clock)
                if(L)
                    Q<=R;
endmodule
```

```
module swap (Resetn, Clock, w, Data, Extern, RinExt1, RinExt2, RinExt3, BusWires, Done);
   parameter n = 8;
   input Resetn, Clock, w, Extern, RinExt1, RinExt2, RinExt3;
   input [n–1:0] Data;
   output tri [n–1:0] BusWires;
   output Done;
   wire [n–1:0] R1, R2, R3;
   wire R1in, R1out, R2in, R2out, R3in, R3out;
   reg [2:1] y, Y;
   parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;

   // Define the next state combinational circuit for FSM
   always @(w, y)
      case (y)
         A: if (w)    Y = B;
            else      Y = A;
         B:           Y = C;
         C:           Y = D;
         D:           Y = A;
      endcase

   // Define the sequential block for FSM
   always @(negedge Resetn, posedge Clock)
      if (Resetn == 0)  y <= A;
      else  y <= Y;

   // Define outputs of FSM
   assign R2out = (y == B);
   assign R3in = (y == B);
   assign R1out = (y == C);
   assign R2in = (y == C);
   assign R3out = (y == D);
   assign R1in = (y == D);
   assign Done = (y == D);

   // Instantiate registers
   regn reg_1 (BusWires, RinExt1 | R1in, Clock, R1);
   regn reg_2 (BusWires, RinExt2 | R2in, Clock, R2);
   regn reg_3 (BusWires, RinExt3 | R3in, Clock, R3);
   // Instantiate tri-state drivers
   trin tri_ext (Data, Extern, BusWires);
   trin tri_1 (R1, R1out, BusWires);
   trin tri_2 (R2, R2out, BusWires);
   trin tri_3 (R3, R3out, BusWires);
endmodule
```

```
module swapmux (Resetn, Clock, w, Data, RinExt1, RinExt2, RinExt3, BusWires, Done);
    parameter n = 8;
    input Resetn, Clock, w, RinExt1, RinExt2, RinExt3;
    input [n–1:0] Data;
    output reg [n–1:0] BusWires;
    output Done;
    wire [n–1:0] R1, R2, R3;
    wire R1in, R2in, R3in;
    reg [2:1] y, Y;
    parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;

    // Define the next state combinational circuit for FSM
    always @(w, y)
        case (y)
            A: if (w)    Y = B;
               else      Y = A;
            B:           Y = C;
            C:           Y = D;
            D:           Y = A;
        endcase

    // Define the sequential block for FSM
    always @(negedge Resetn, posedge Clock)
        if (Resetn == 0)  y <= A;
        else  y <= Y;

    // Define control signals
    assign R3in = (y == B);
    assign R2in = (y == C);
    assign R1in = (y == D);
    assign Done = (y == D);

    // Instantiate registers
    regn reg_1 (BusWires, RinExt1 | R1in, Clock, R1);
    regn reg_2 (BusWires, RinExt2 | R2in, Clock, R2);
    regn reg_3 (BusWires, RinExt3 | R3in, Clock, R3);

    // Define the multiplexers
    always @(y, Data, R1, R2, R3)
        if (y == A) BusWires = Data;
        else if (y == B) BusWires = R2;
        else if (y == C) BusWires = R1;
        else BusWires = R3;
endmodule
```

6.3 由顶向下的设计方法

6.3.1 数字系统的设计任务
6.3.2 算法状态机和算法流程图

6.3.1 数字系统的设计任务

数字系统的设计任务主要包括下列几部分：

（1）对设计任务进行分析，根据课题任务，把所要设计的系统合理地划分成若干子系统，使其分别完成较小的任务。

（2）设计系统控制器，以控制和协调各子系统的工作。

（3）对各子系统功能部件进行逻辑设计。

6.3.1 数字系统的设计任务

设计一个简单的 8 位二进制无符号数并行加法运算器，使之能完成两数相加并存放累加和的要求。

1 逻辑抽象,分析.子系统初步划分：

①一个8位加法器，完成二数相加操作；

②两个8位寄存器(A和D)，存放加数和被加数；

③一个8位寄存器(B)，存放求和结果；

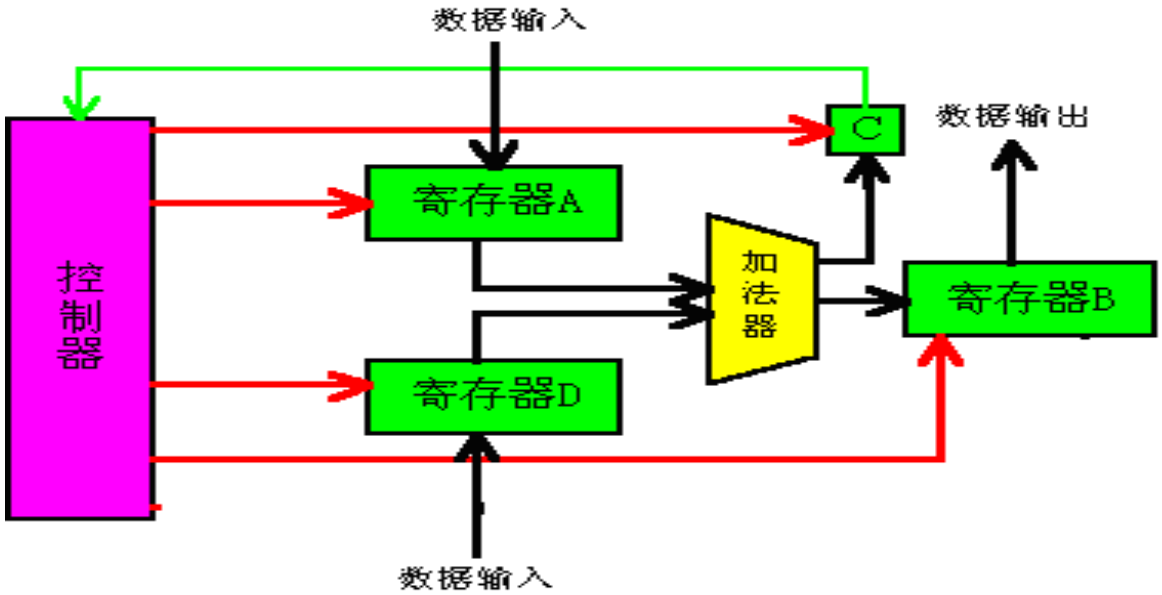
④一个1位的寄存器(C)，存放进位信号；

⑤一个控制器，控制各个子系统；

此外，需设置数据总线，保证数据正确流动。控制器和被控单元间需设置控制线或反馈线等。

基本子系统: ALU、寄存器堆、 RAM、数据通路和控制器。

2 初步设计系统框图

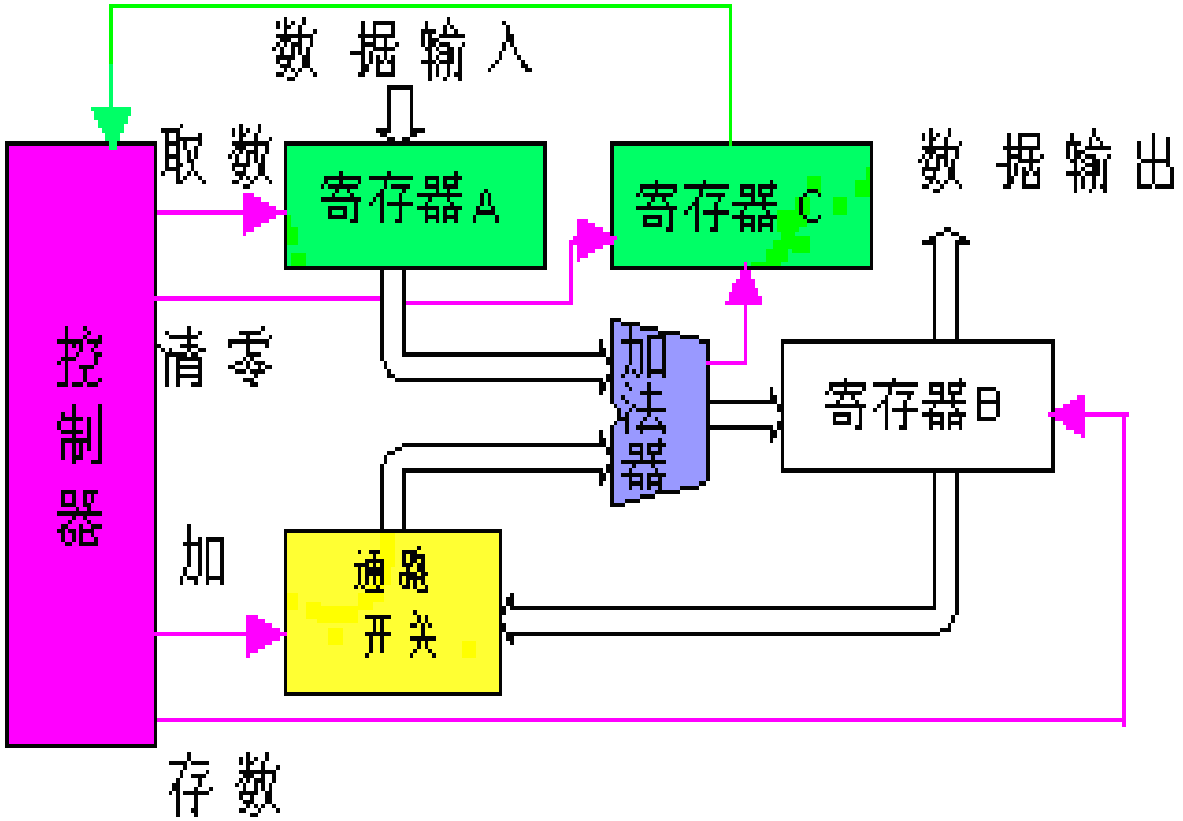


3 系统化简

1.两条独立的数据输入线能否合为一条？

2.寄存器B在开始阶段可做加数寄存器，以后可保存累加结果，寄存器D可否省去？。

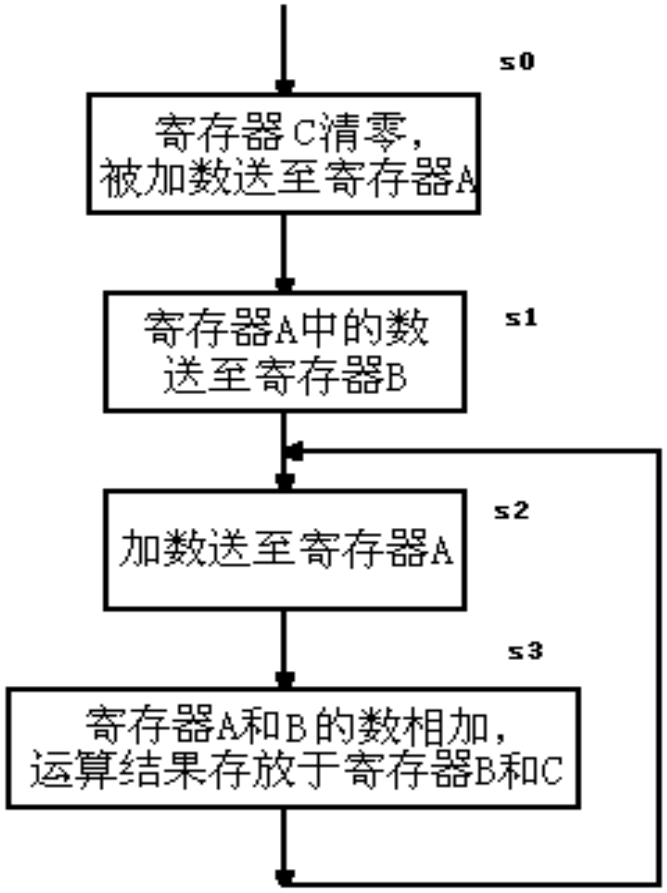
化简后的系统框图

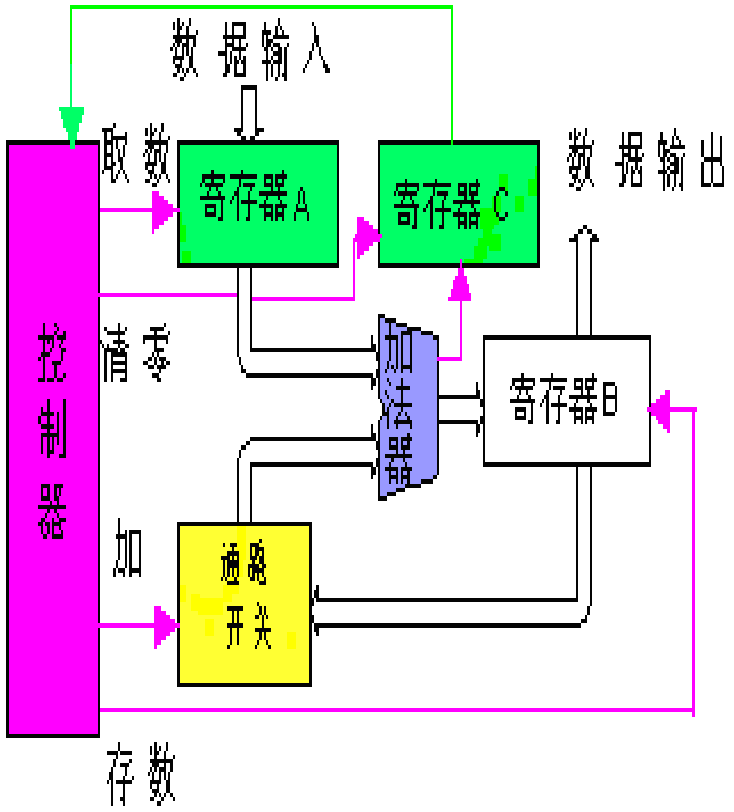


通路开关：可用三态缓冲器代替。

4 根据系统框图，编制控制算法：

B=A+B





控制算法： 控制器对执行部件的控制关系。以某种状态流程图方式来表达。

控制算法可修改为下列四步：

（１）寄存器C清零，取被加数至寄存器A；

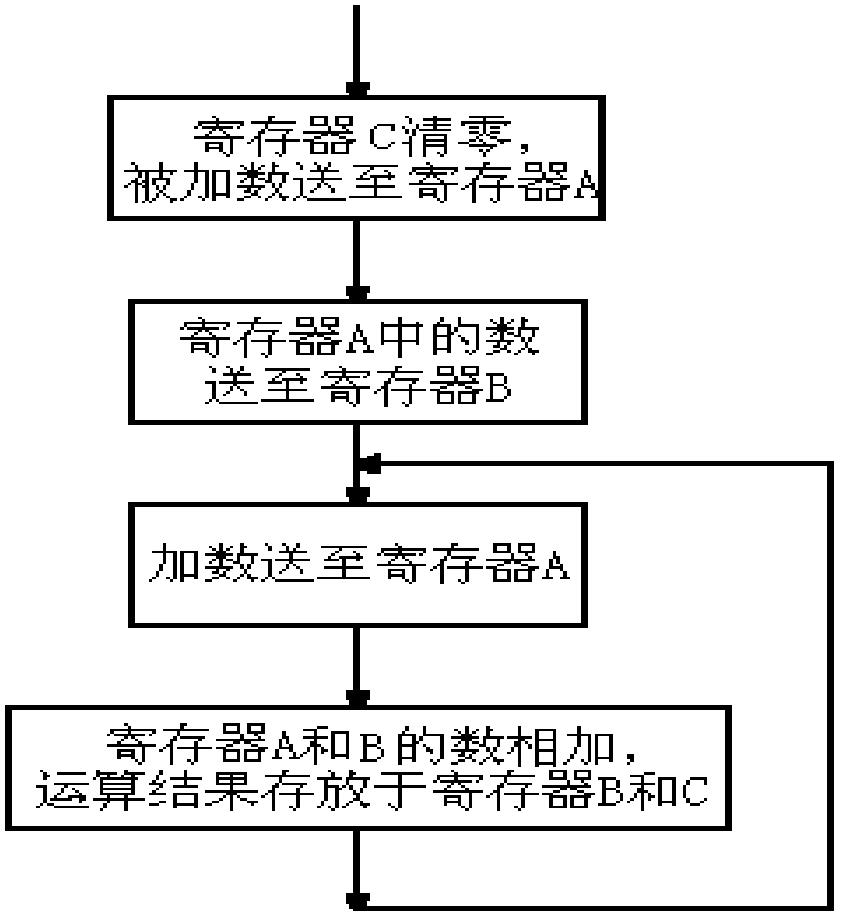
（２）将A中数据送寄存器B；

（３）取加数至寄存器A；

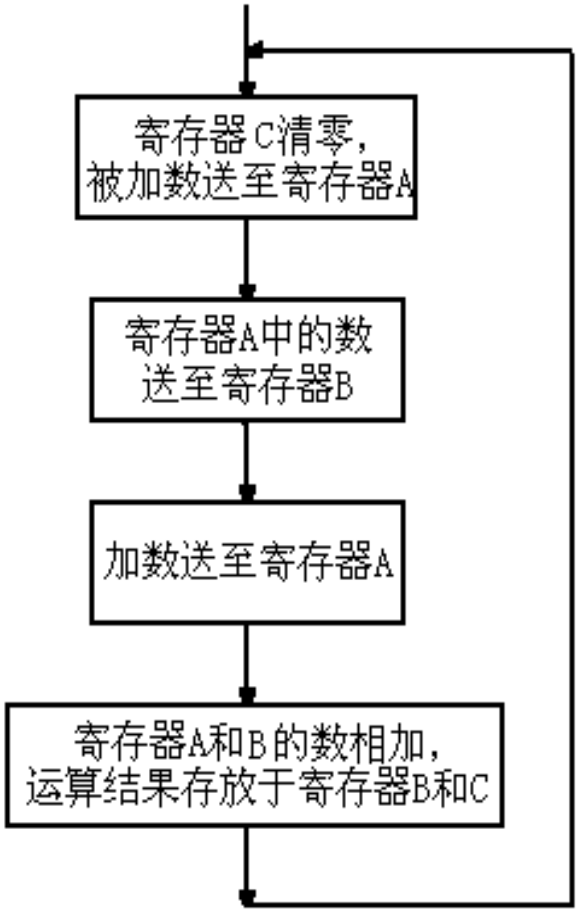
（４）将A与B中的数相加，结果存于B，进位信号送至寄存器C。

控制算法1

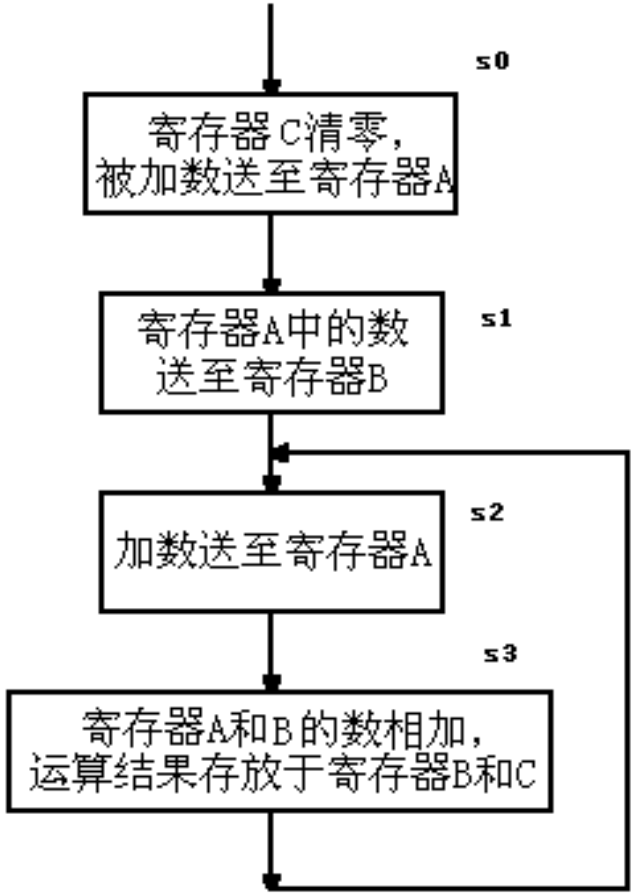
- CLR C LDA
- LDB
- LDA
- ADD LDB



根据系统框图，控制算法2为：



（返回初始状态S0）



（返回到S2，反复累加）

6.3.2 算法状态机和算法流程图

控制算法：控制器对被控对象的控制关系。

把控制算法分离出来就是明确这种控制关系。

算法状态机（简称ASM）本质上是一个有限状态机，主要用于同步系统。

ASM理论可以把非常复杂的控制器的控制过程用框图式的流程图--算法流程图表示出来，所以算法流程图又称为ASM流程图。

算法状态机

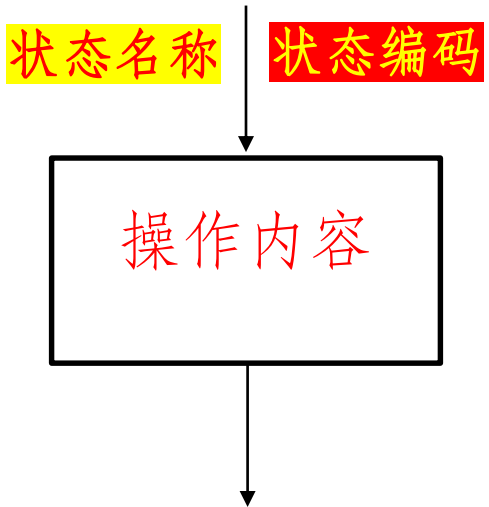
与时序状态机相比，ASM有如下特点：

- ASM以流程图形式描述控制器功能、状态变化及条件、状态转换的时间,指明了寄存器具体操作;
- ASM是时钟驱动的流程图。不是事件驱动的流程图;
- ASM不仅描述控制器控制过程（控制器状态转换、转换条件及控制输出）也指明被控部件的应有操作;
- ASM适用于同步系统。
- 可方便地实现或设计硬件控制器。

ASM流程图的基本图形

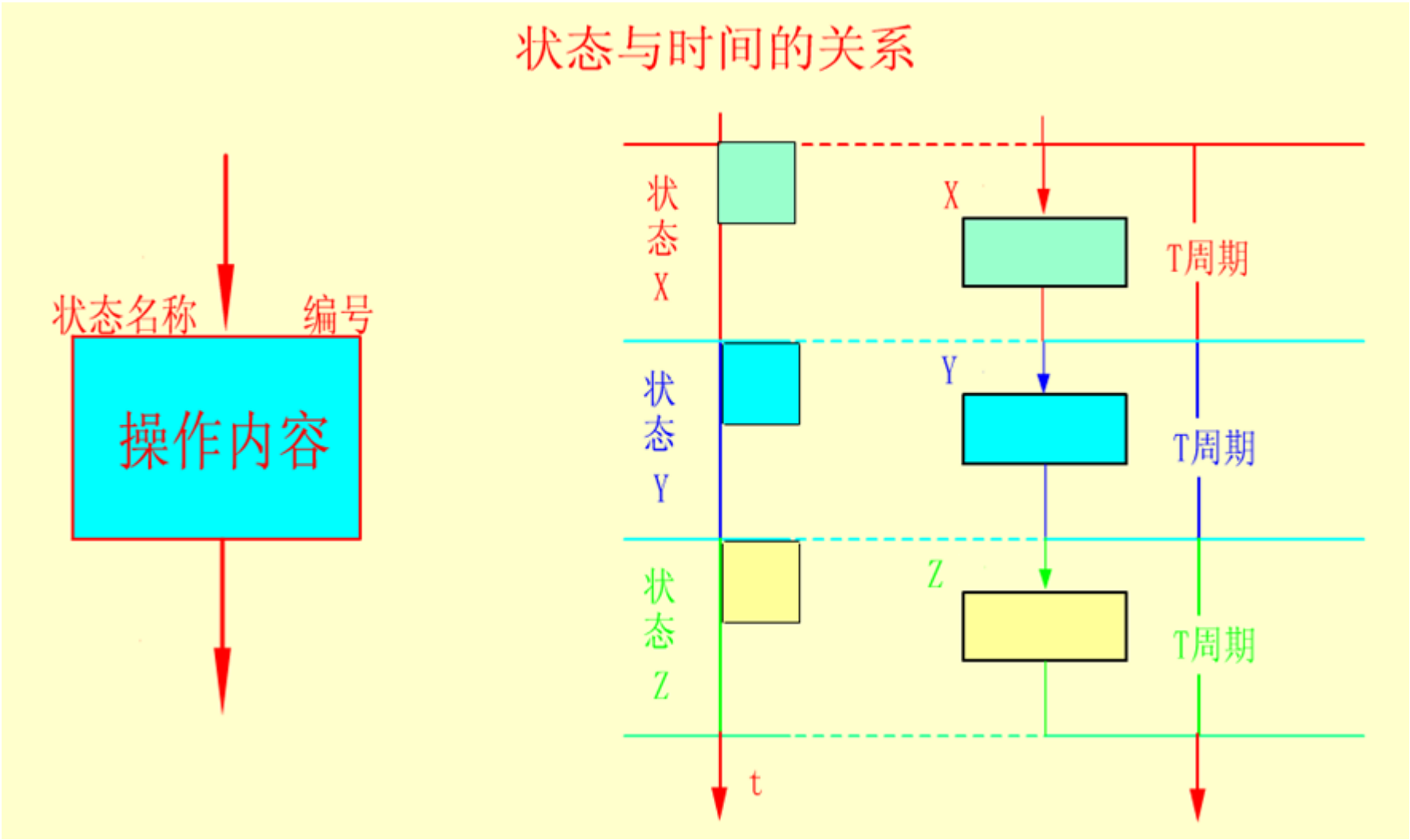
状态框：

- 有进口和出口的矩形框， 代表系统一个状态。状态框内定义此状态实现的寄存器操作和输出。
- 状态经历的时间用状态时间表示，同步系统中状态时间为一个同步时钟周期。
- 状态名写在框外左上方，状态编码写在框外右上方。
- 操作内容写在矩形框内。



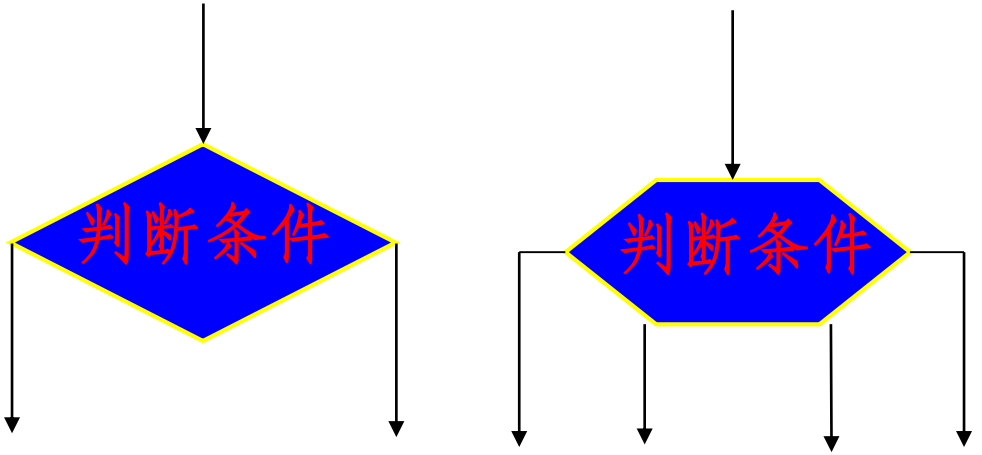
算法流程图由下列几种基本图形组成：

（1）状态框。



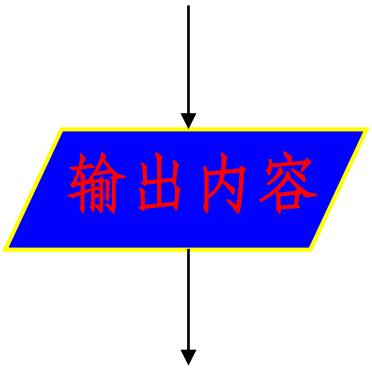
条件判断框：

- 简称分支框，用单入口多出口的菱形或多边形表示。
- 框内写检测条件，判别可以是状态变量或输入变量；个数不限；
- 判别变量的作用可以同等，也可以有优先级。出口处注明各分支所满足的条件。
- 条件判断框不占用时间. 经历时间依附于状态框。



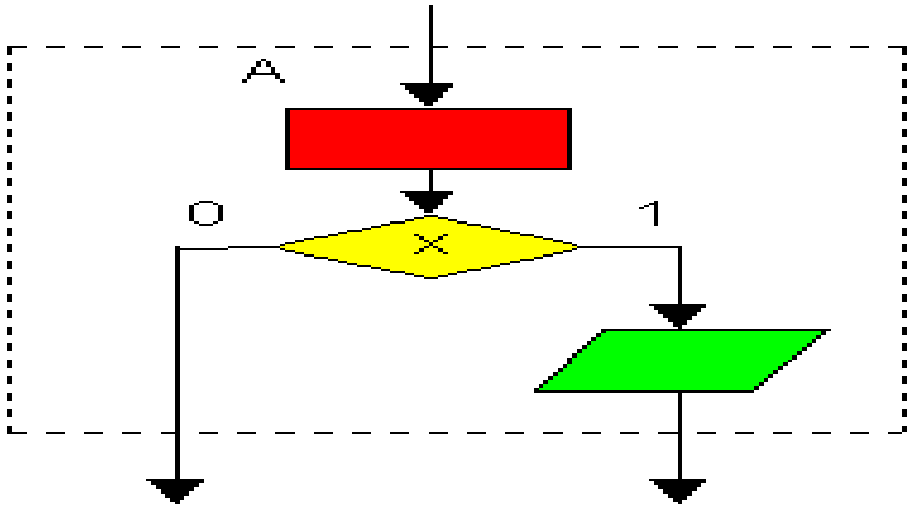
条件输出框：

- 由平行四边形组成。入口来自某一分支，某些条件满足时，给出指定的输出，输出操作内容写在框内。
- *注意*：条件输出框不是控制器的一个状态。不占用时间，它经历的时间依附于状态框。



状态单元：

- 由一个状态框、若干条件判断框或条件输出框组成（上述三种图形的组合）
- 状态单元入口是状态框入口，出口指向状态框（可以有多个）。
- 状态单元经历的时间为状态框时间。
- 状态单元典型结构如下



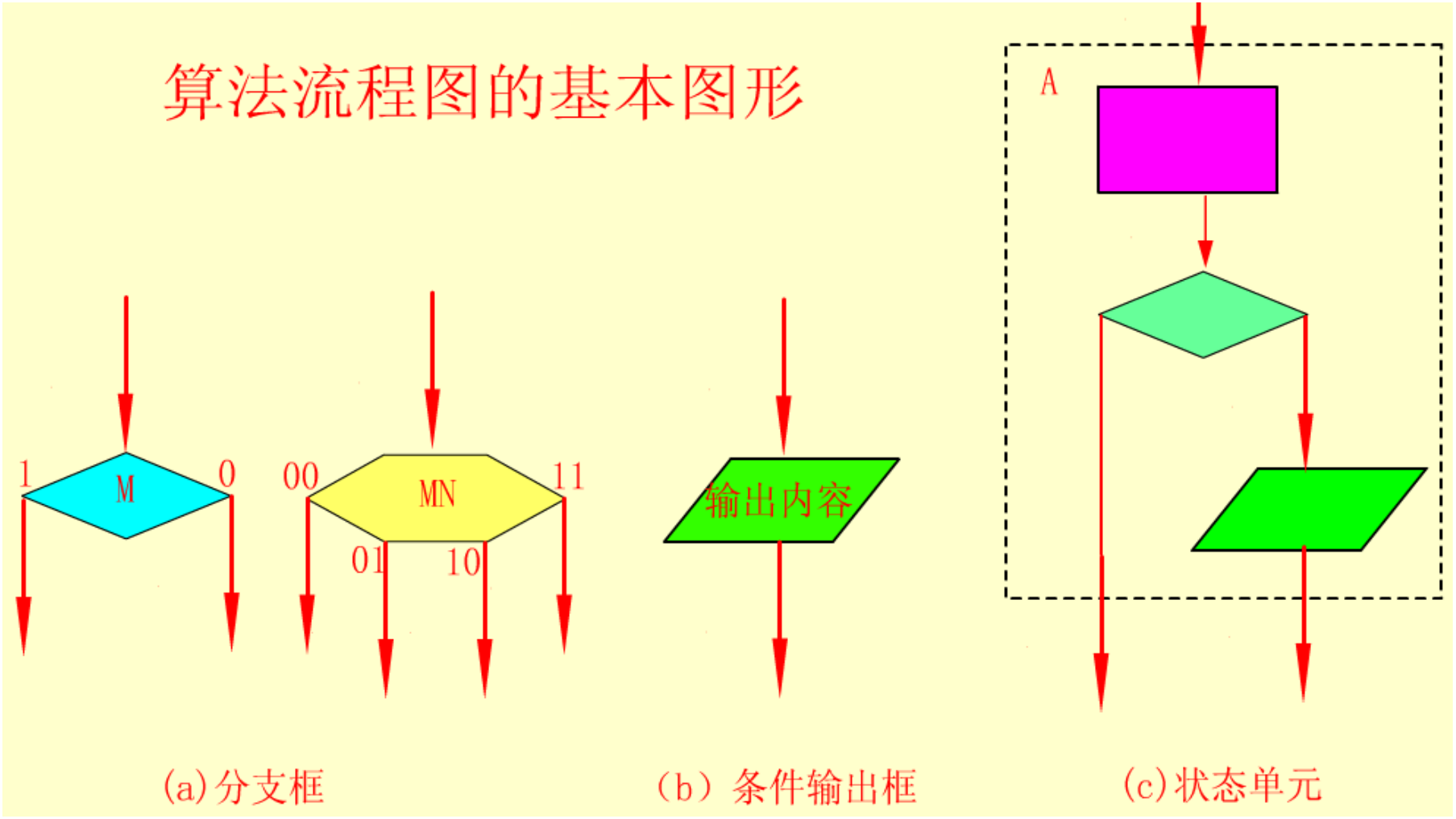
寄存器输出

条件输出

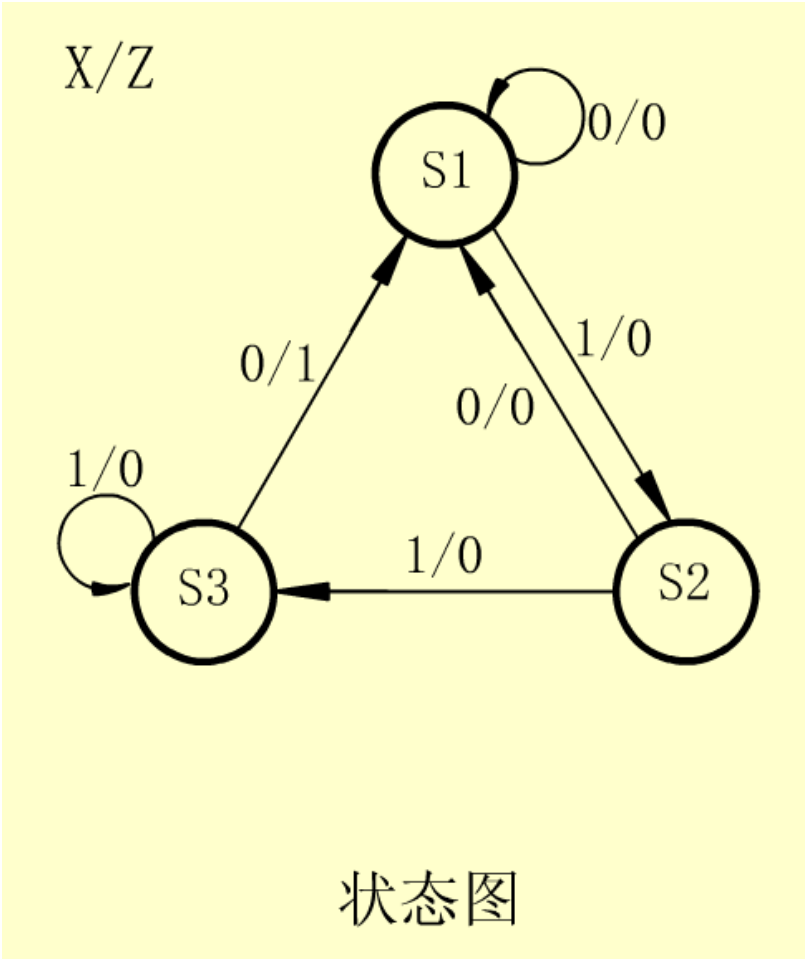
（2）分支框。

（3）条件输出框。

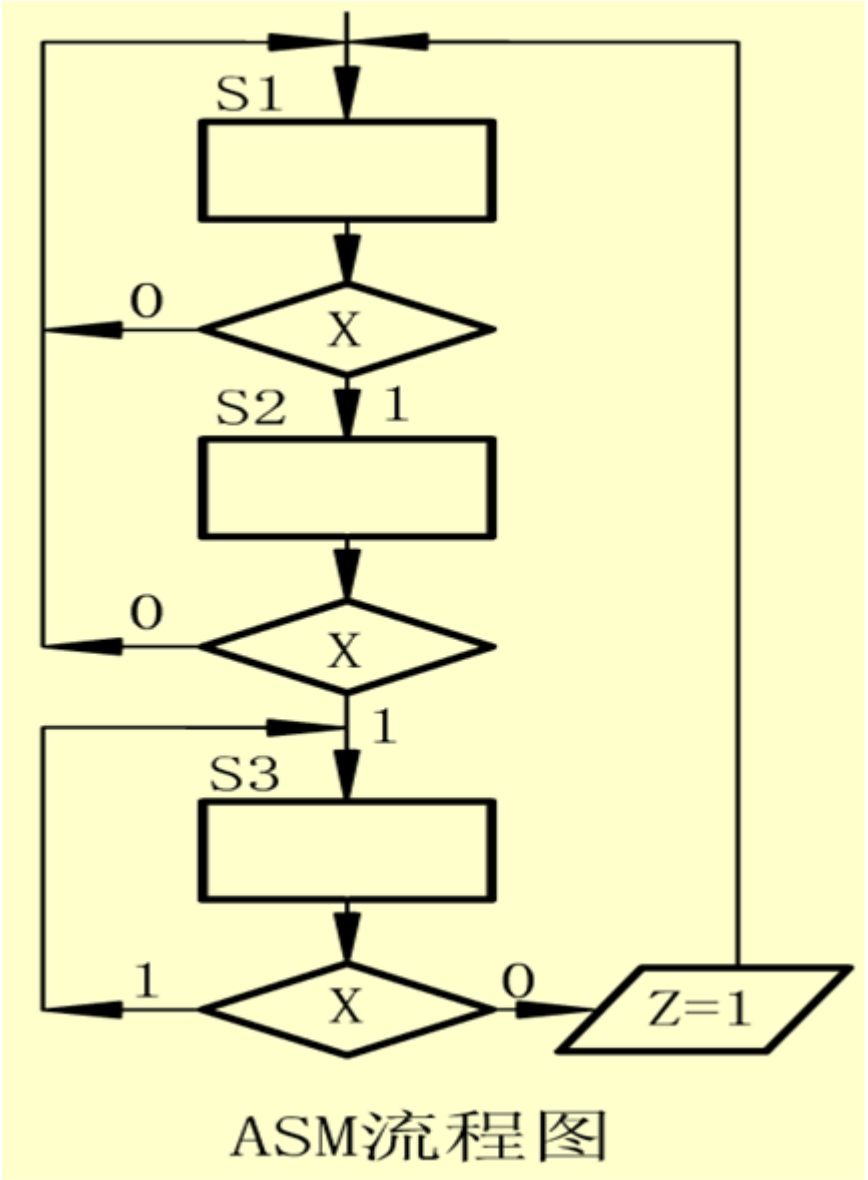
（4）状态单元。



【例】　将下图所示的米里机状态图转换成ASM流程图。

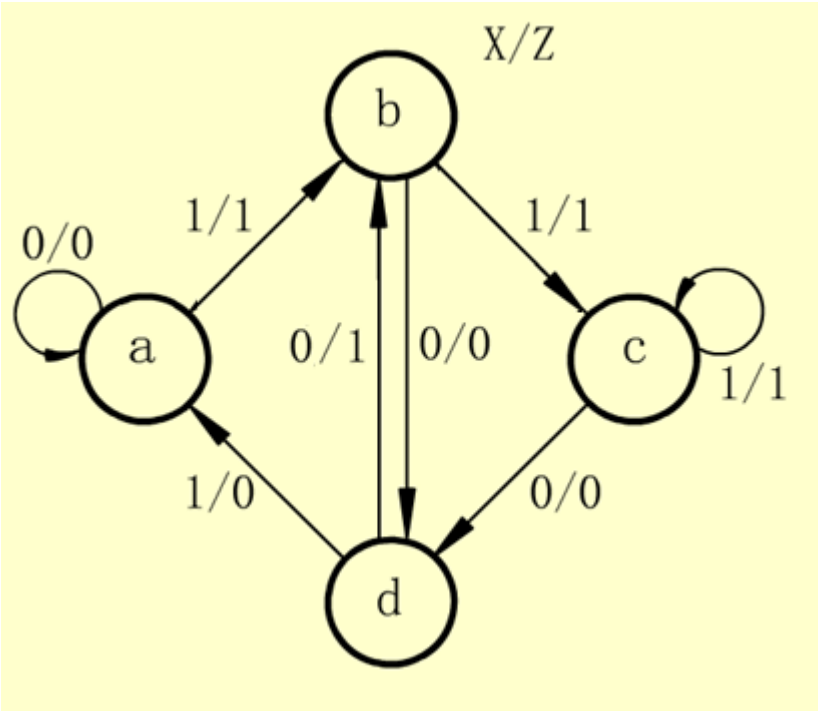


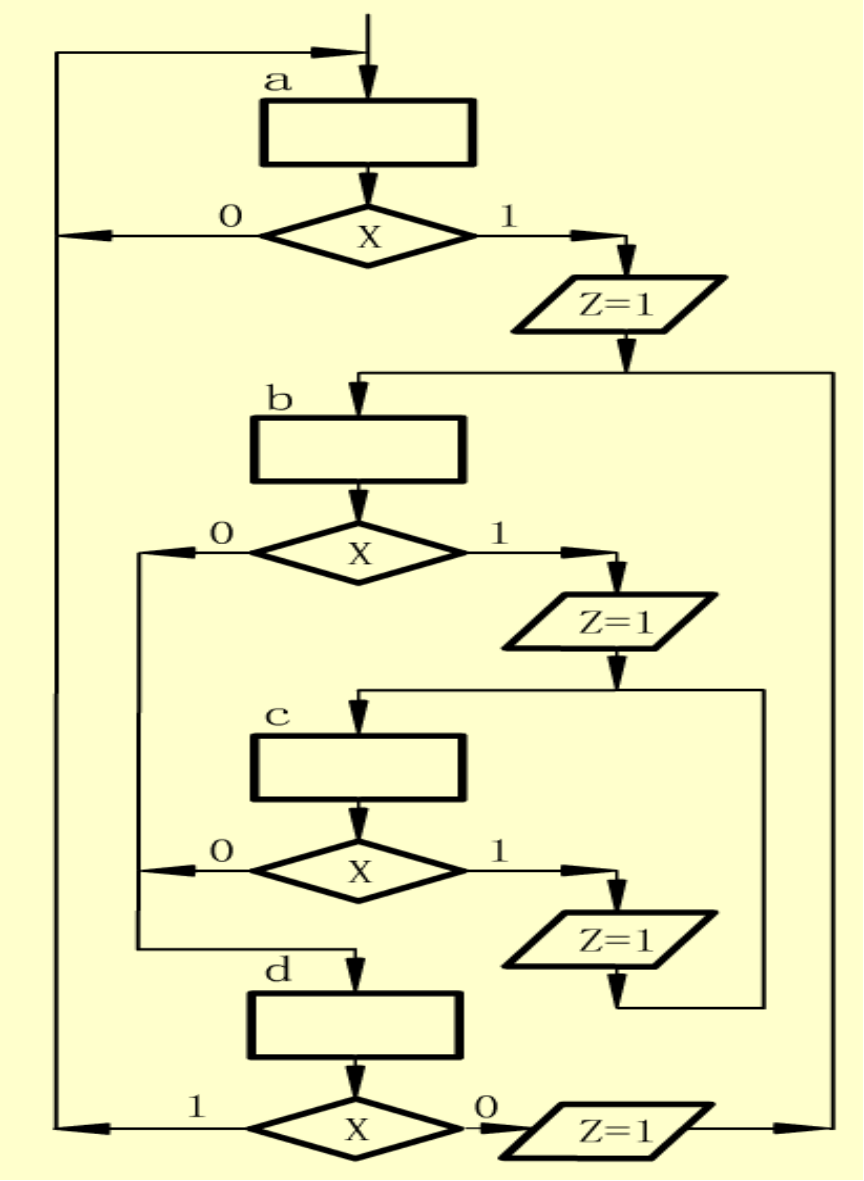
状态图



ASM流程图

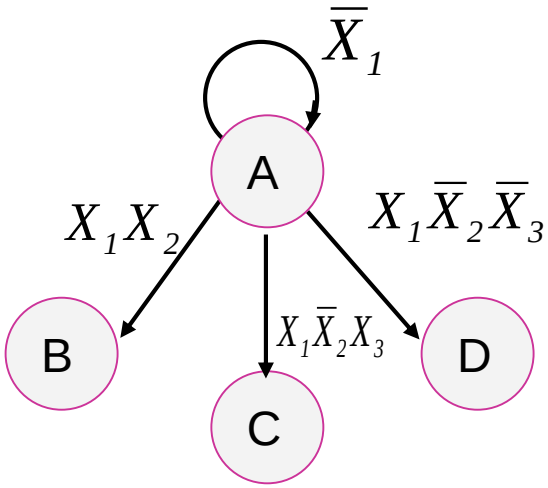
【例】将下图所示的四状态机转换成ASM流程图。

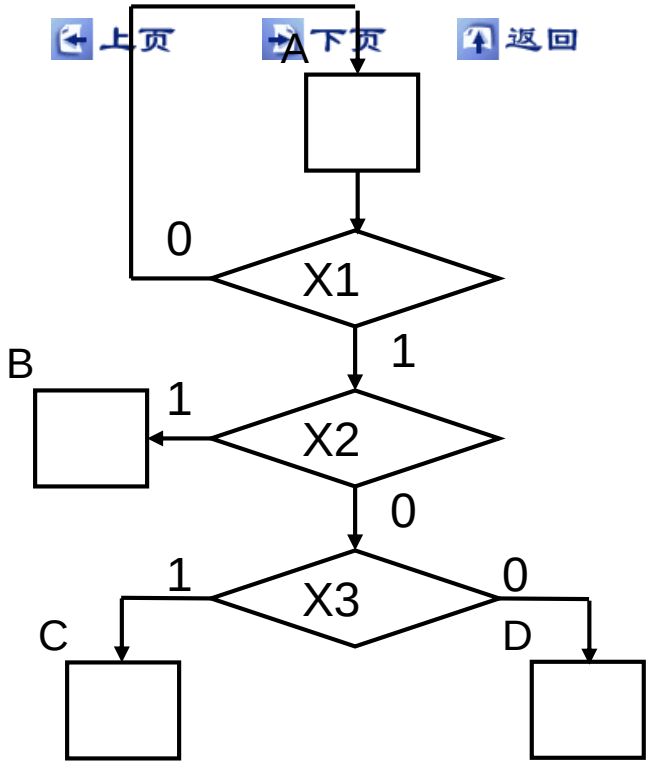




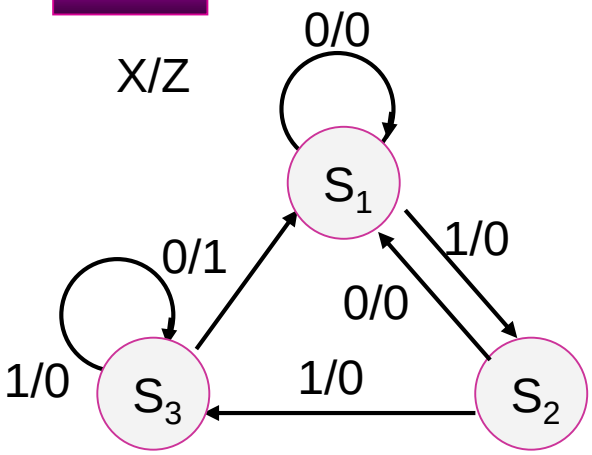
ASM练习

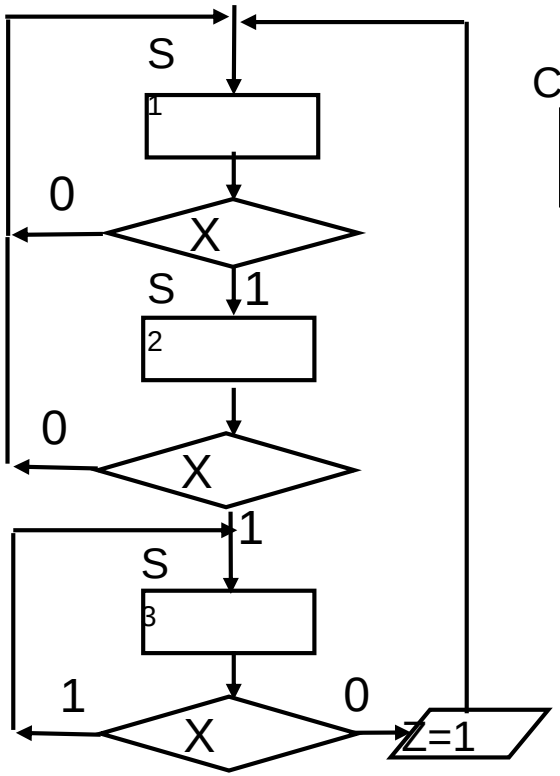
例1

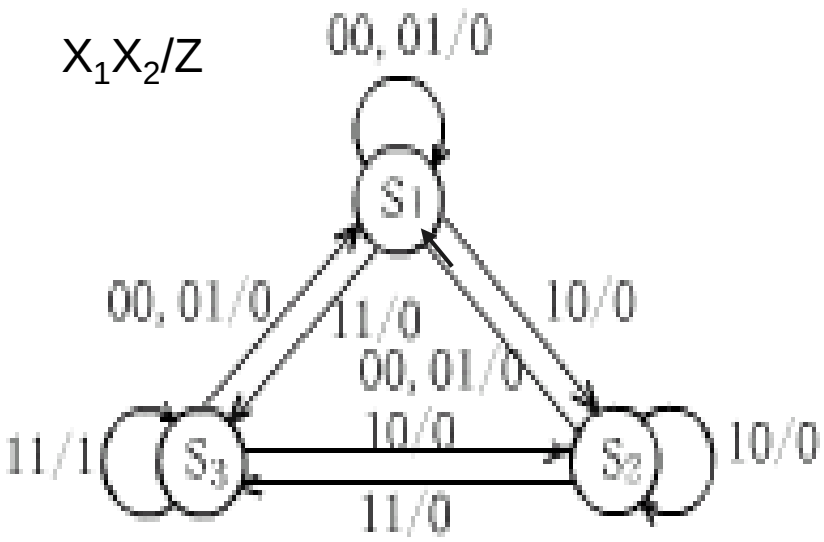




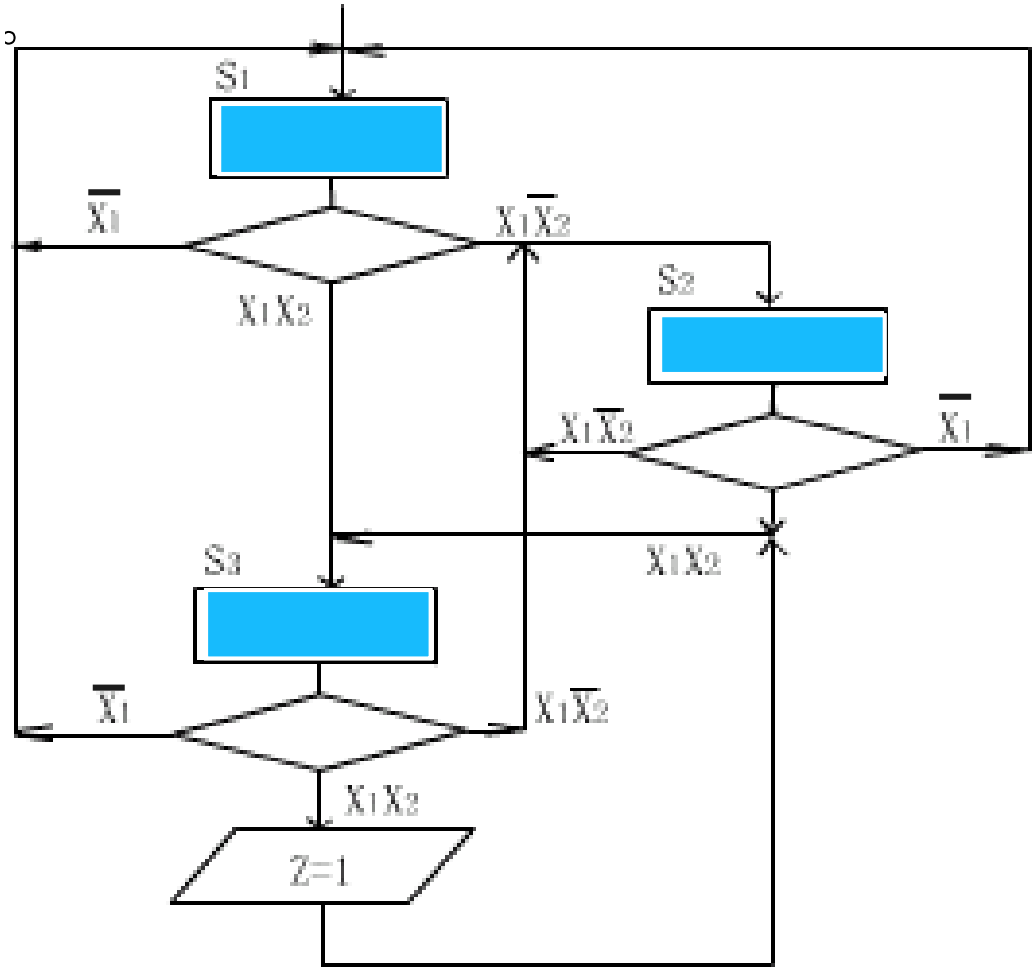
例2







同步时序状态机如图所示，其中X_1、X_2是两个外部输入信号，Z是输出信号。将时序状态机转换为ASM图



6.4 小型控制器的设计

6.4.1 控制器的基本概念
6.4.2 计数器型控制器
6.4.3 多路选择器型控制器
6.4.4 定序型控制器

6.4.1 控制器的基本概念

本质上看，控制器是一种时序逻辑电路。

控制器设计的主要特点：性能至上

不必过分追求状态最简，触发器的数量也不必一味追求最少。

控制器的类型：
小型控制器：其结构形式有计数器型、多路选择器型、定序型等多种，适用于小型数字系统。

微程序控制器：将控制程序固化，通用性强，设计简单，结构规整，适用于大型复杂数字系统。

6.4.2 计数器型控制器

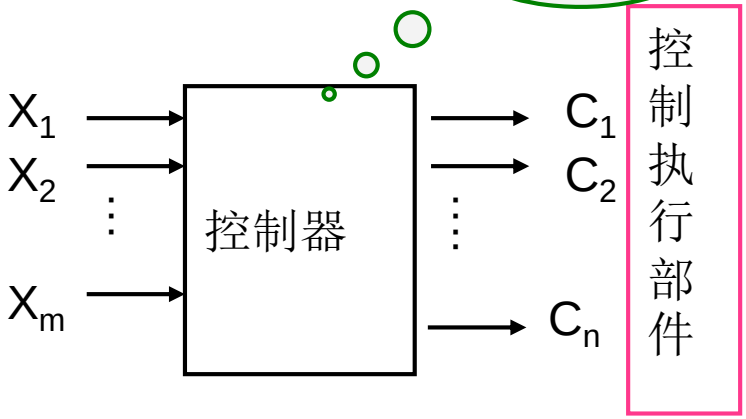
计数器本身有许多不同的状态，因而各种形式的计数器均可以改造为控制器。

当控制状态数较多时，为了节省触发器数目，宜采用编码方式组成状态。

计数器型控制器的一般框图如图6.12所示。

控制器是一种时序逻辑电路

设计依据：输入信号和ASM图



一、计数型控制器

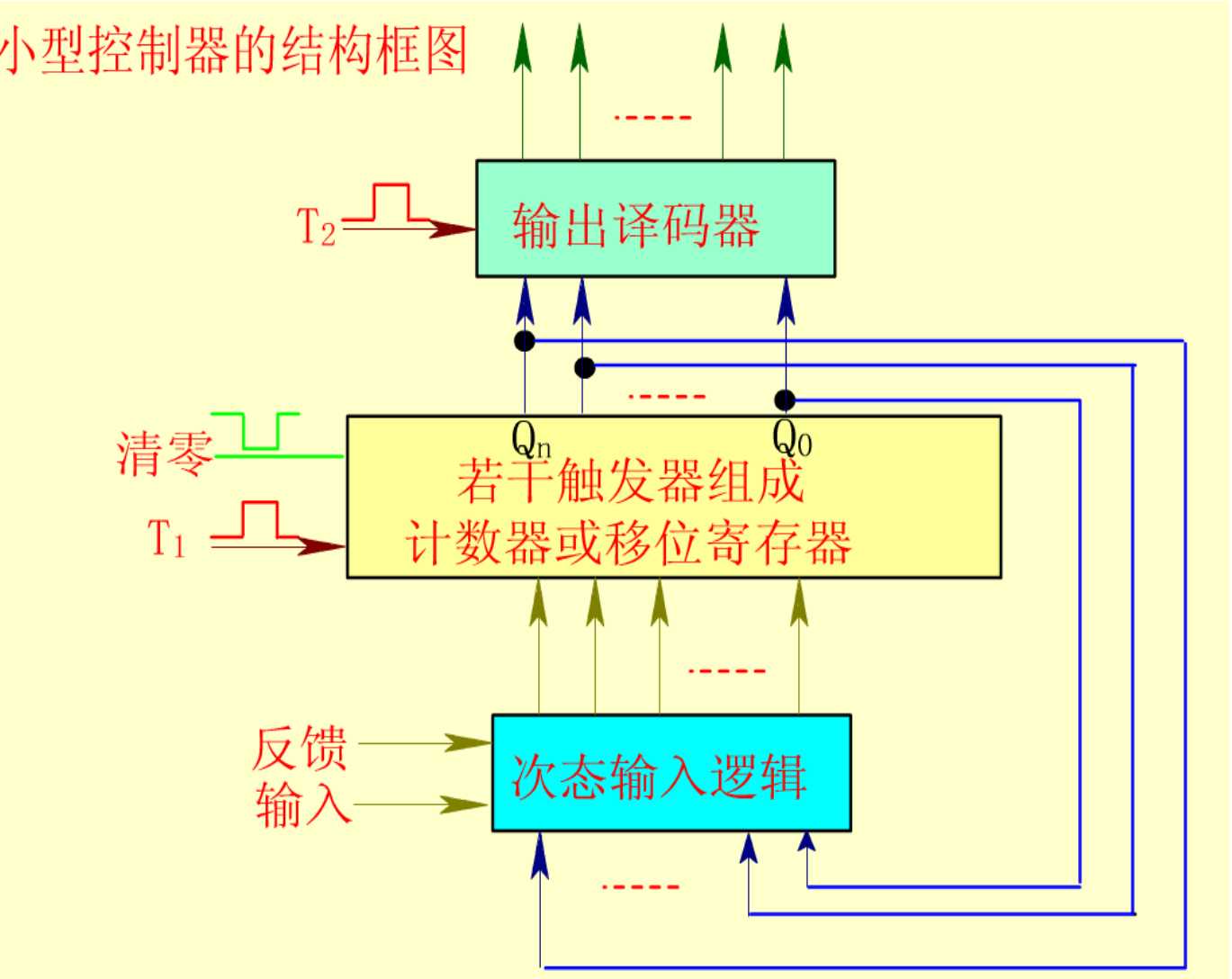
核心 ……计数器

步骤 ……* 根据ASM图确定存在几种状态（n个变量可描述2^n种状态）

7个状态用3个触发器

* 将每个状态给以任意状态编码（标在状态框右上角）
* 根据输入条件及ASM图设计次态控制逻辑
* 计数状态译码后输出控制信号

6.4.2 计数器型控制器

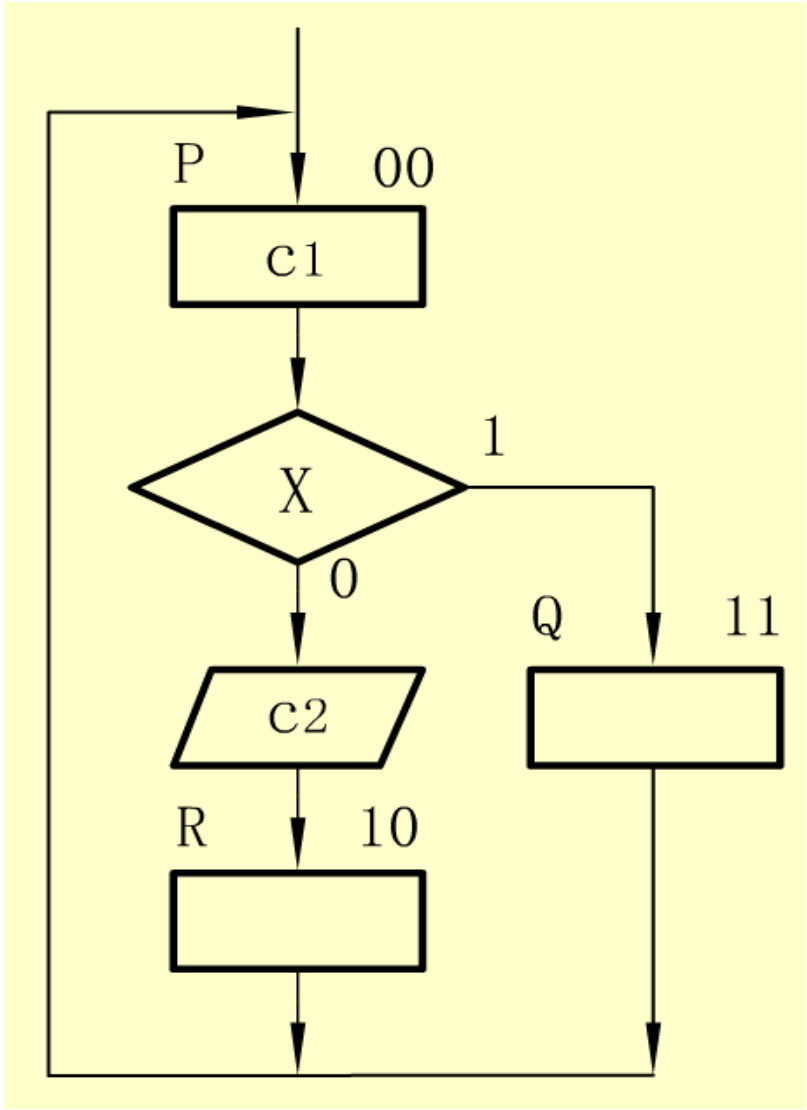


输出译码器　对不同状态下产生的各种控制信号进行译码。

计数器　含有n个触发器。触发器的2^n个状态以二进制编码作为状态变量，并与ASM流程图中每一个已编码的状态框一一对应。

次态控制逻辑　就是实现状态激励函数逻辑表达式。

【例5】 图6.13（a）为某一控制器的算法流程图，请设计一个计数器型控制器。



PS（现态）			NS（次态）	
B	A	X	B(D)	A(D)
0	0	0	1	0
0	0	1	1	1
0	1	×	0	0
1	0	×	0	0
1	1	×	0	0

（3）写出触发器的次态激励函数表达式

根据状态转移表并利用NS=∑PC·C公式，可得：

$B(D)=\overline{B}\,\overline{A}\,\overline{X}+\overline{B}\,\overline{A}X=\overline{B}\,\overline{A}$

$A(D)=\overline{B}\,\overline{A}\,X$

（4）写出控制信号表达式

从ASM流程图看到：控制命令 C1 发生在状态P；控制命令C2也发生在状态P，但与输入X 有关。

C_1=状态P=$\overline{B}\,\overline{A}$

C_2=状态P和$\overline{X}$的“与”=$\overline{B}\,\overline{A}\,\overline{X}$

由于未指明 C_1、C_2两个控制命令的用途，因此它们是电位控制信号。

（5）画出控制器的具体电路图

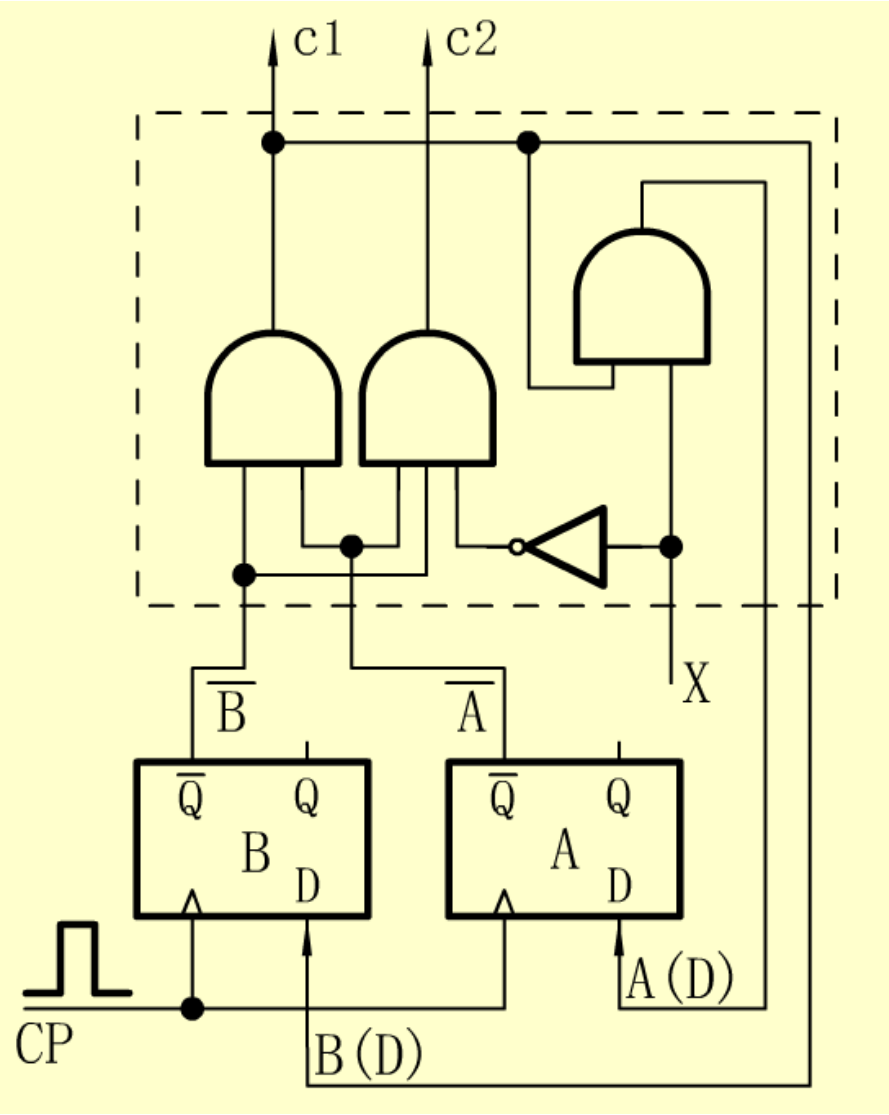
将次态激励函数表达式和控制命令C_1、C_2表达式用于与门、非门实现

$$B(D)=\overline{B}\,\overline{A}$$

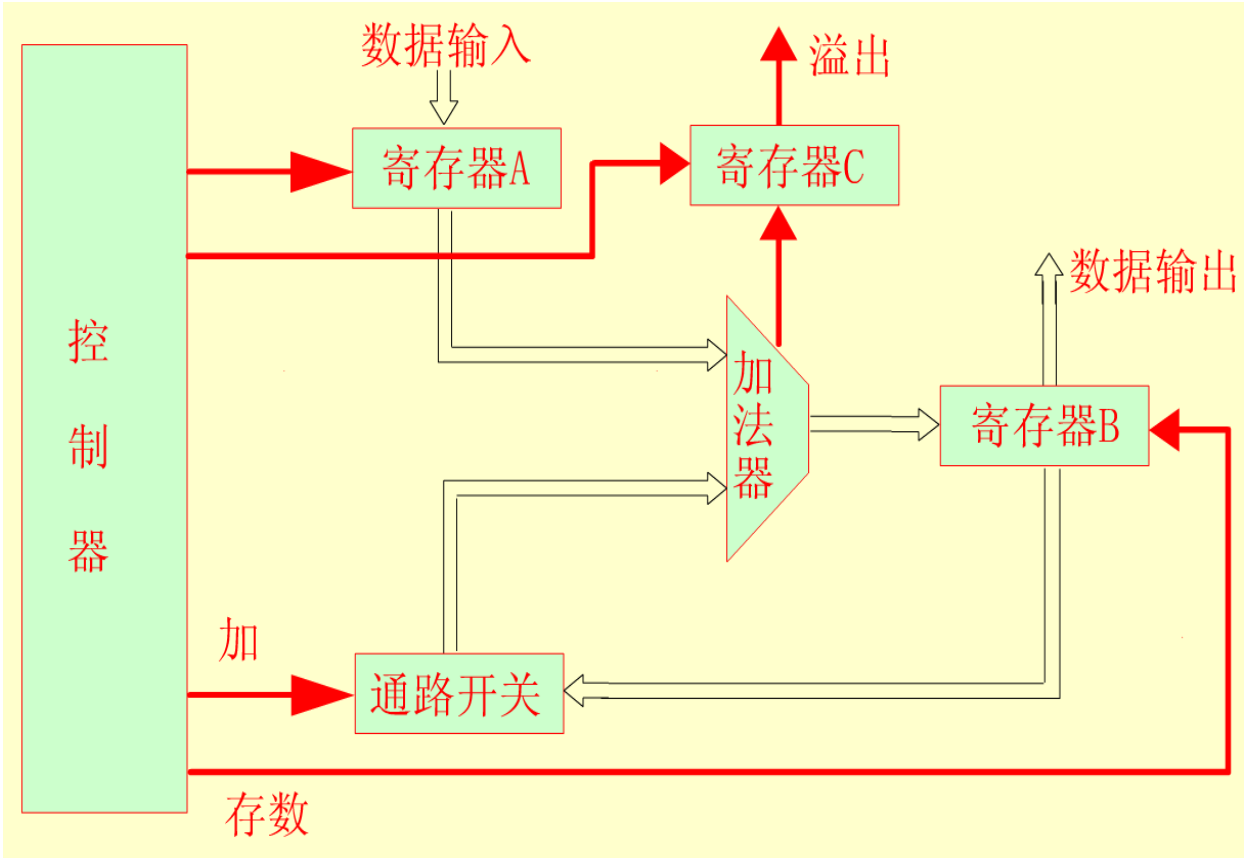
$$A(D)=\overline{B}\,\overline{A}\,X$$

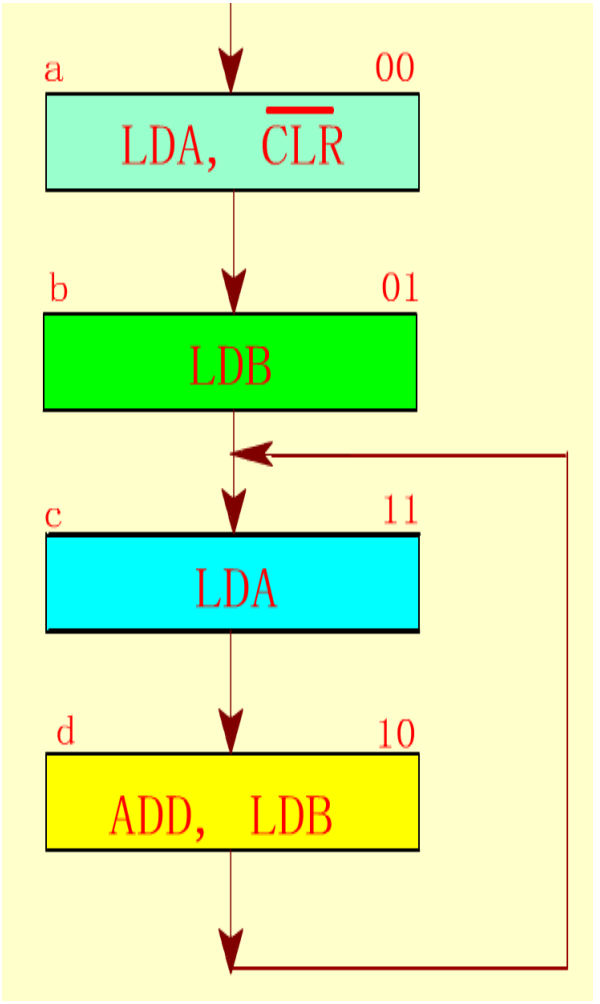
$$C_1=\overline{B}\,\overline{A}$$

$$C_2=\overline{B}\,\overline{A}\,\overline{X}$$



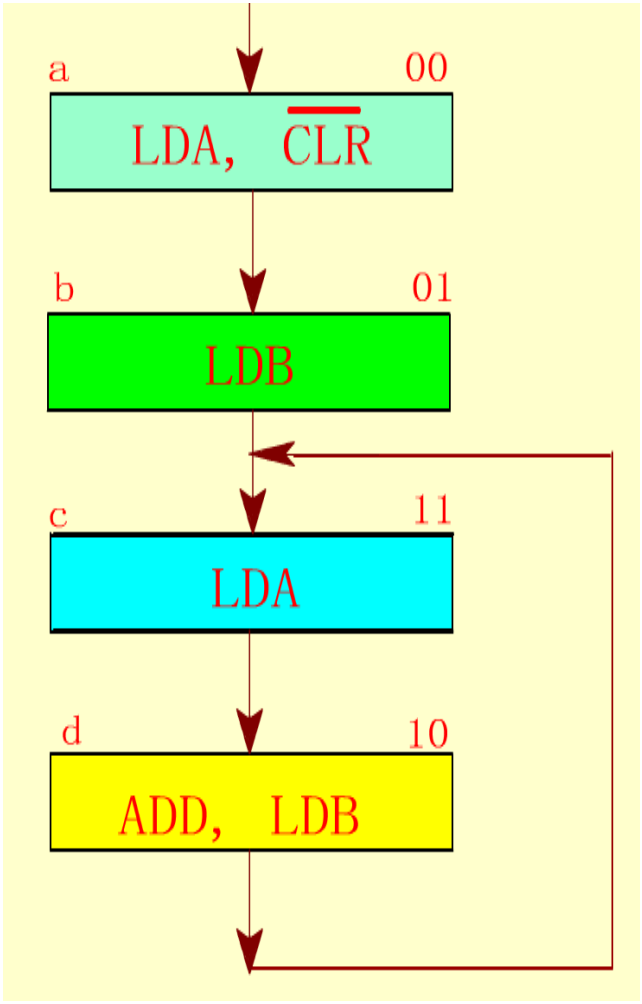
【例6】　请设计图6.7中所示加法累加运算器的控制器，要求采用计数器型控制器。





（3）列出状态转移表

假设采用D 触发器，根据ASM流程图，得到状态转移表如下：



PS(现态)			NS(次态)		
B	A	状态名	状态名	B(D)	A(D)
0	0	a	b	0	1
0	1	b	c	1	1
1	1	c	d	1	0
1	0	d	c	1	1

（4）写出次态激励函数表达式

利用 NS=∑PS·C 公式(此处 C=1,无条件转移),可得:

$$B(D) = \bar{B}A + B\bar{A} + B\bar{A} = B + A = \overline{\bar{B}\bar{A}}$$

$$A(D) = \bar{B}\bar{A} + \bar{B}A + BA = \bar{B} + \bar{A} = \overline{BA}$$

(5)写出控制命令的逻辑表达式

根据ASM流程图，LDA信号在a状态和c状态出现，LDC信号在d状态出现，LDB信号在b状态和d状态出现，都是打入控制信号，需要用T_2 节拍时间进行限制。其他控制信号为电位信号：

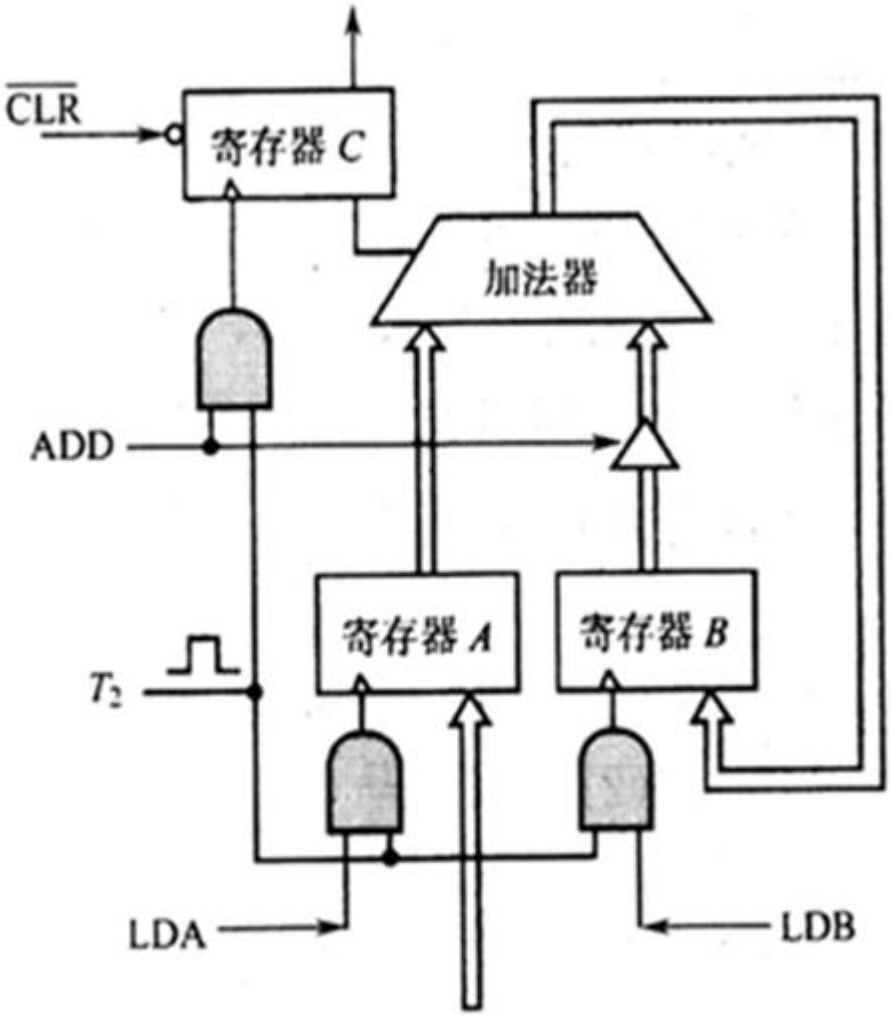
$LDA=(\bar{B}\ \bar{A} + B\ A)T_2$

$LDB=(B\bar{A}+\bar{B}A)T_2=(B\oplus A)T_2$

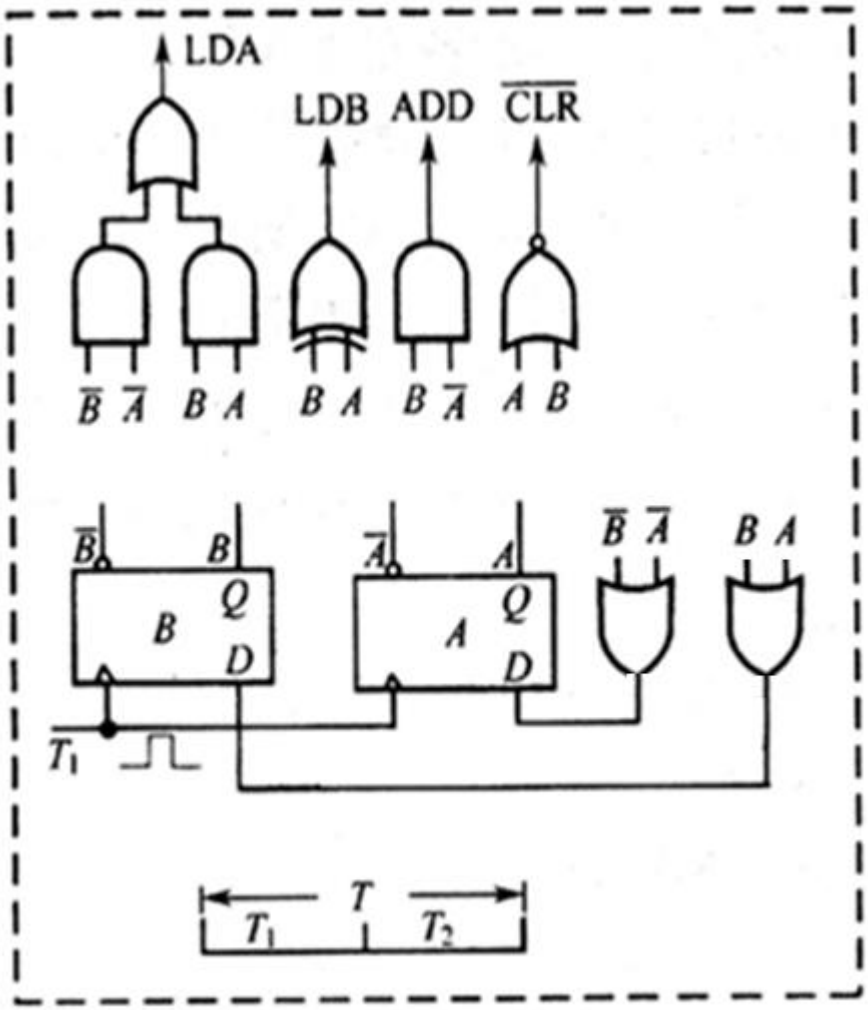
$\overline{CLR}=\bar{B}\ \bar{A}$

$ADD=B\bar{A}$

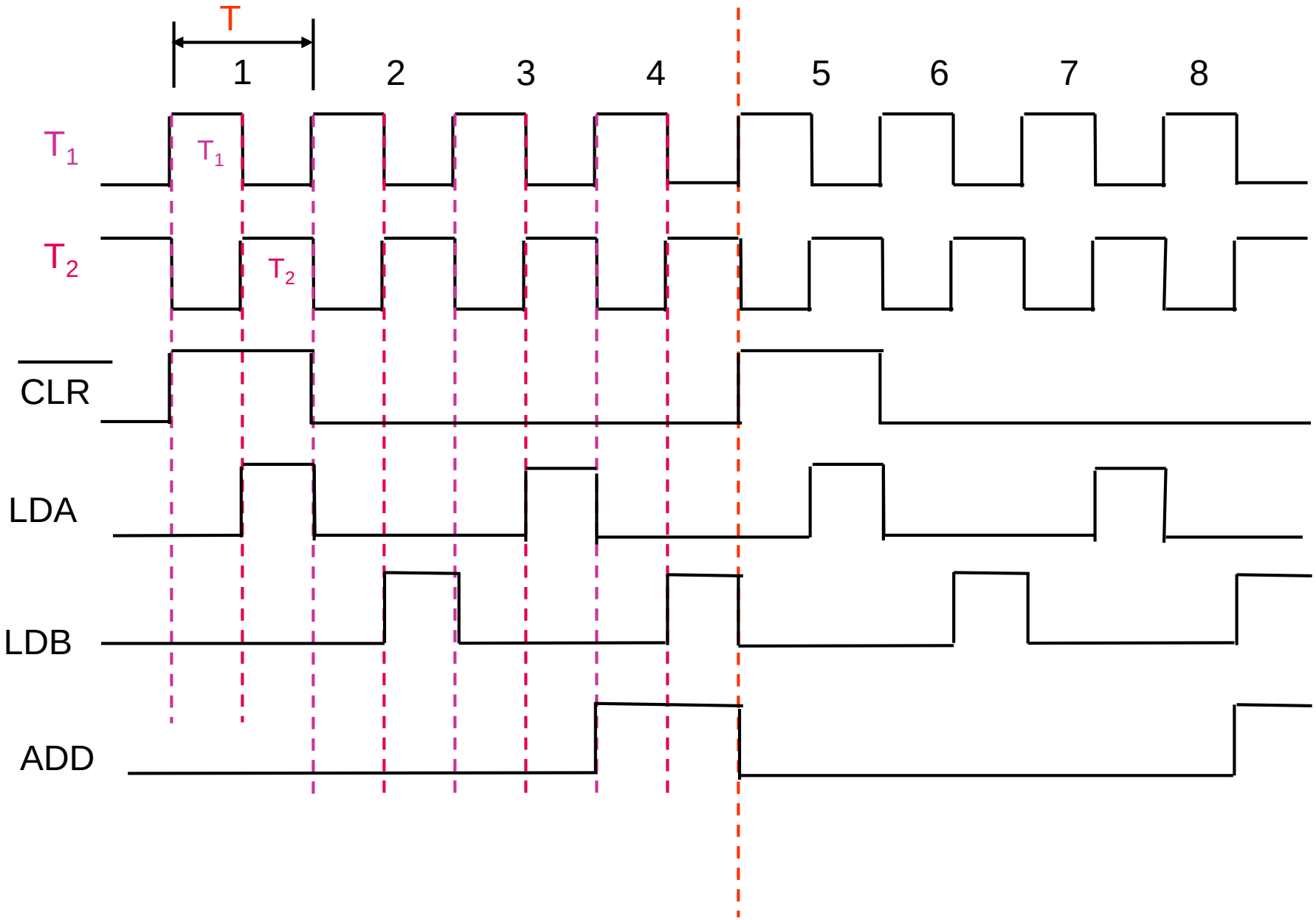
(6) 画出控制器具体电路图



数据人

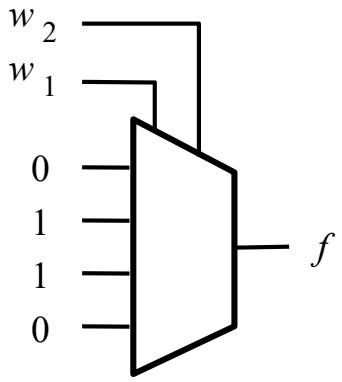


控制器


T
1
2
3
4
5
6
7
8
T1
T1
T2
T2
CLR
LDA
LDB
ADD

用4选1选择器实现异或

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

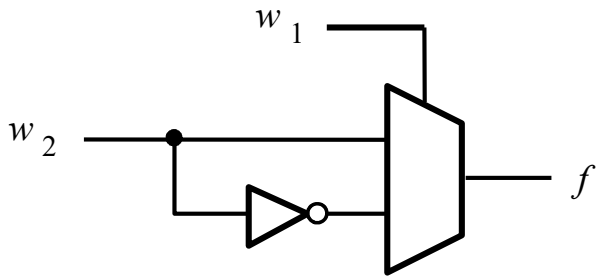


(a) Implementation using a 4-to-1 multiplexer

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

w_1	f
0	w_2
1	$\overline{w}_2$

(b) Modified truth table



(c) Circuit

6.4.3 多路选择器型控制器

特点：

采用MUX作为控制器状态触发器的次态激励函数的生成电路，不仅能使控制器的设计过程标准化，而且能使整个控制器电路清晰明确。

方法：

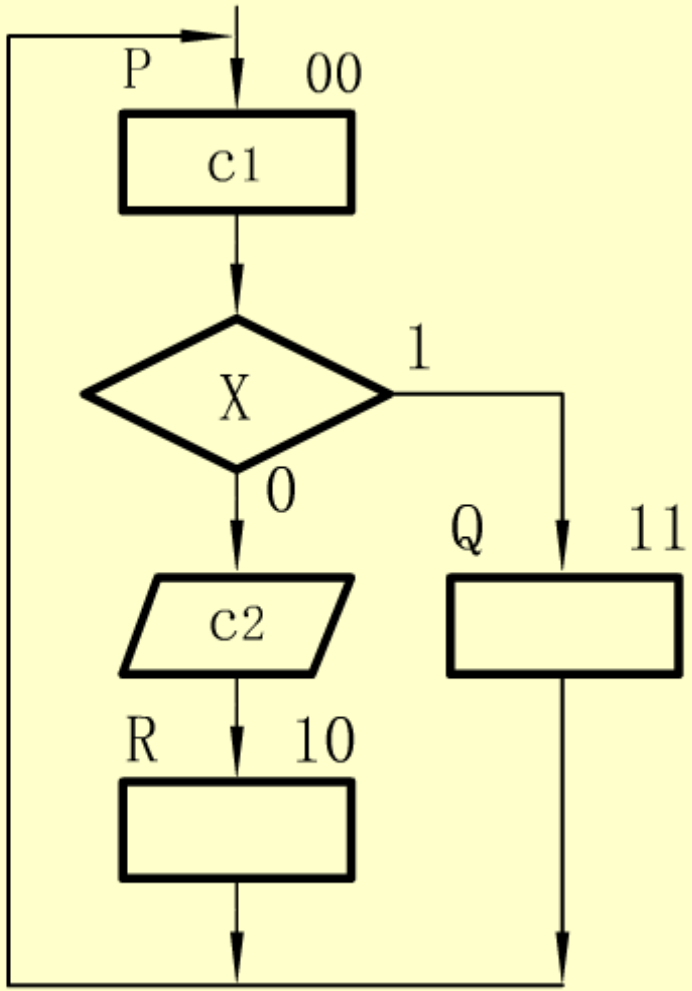
所有MUX输出端的组合就是控制器次态（NS）的编码。而MUX的输入端则是对应所有现态（PS）的状态转移条件。

步骤

* 根据ASM图给多路选择器的输入端提供适当的输入值
* 给ASM图中每个二进制编码状态赋予十进制数码
* 建立状态转换表，求多路选择器数据输入端的函数值

【例 7】设计一个多路选择器MUX型控制器，实现图6.13（a）的控制算法。

解：（1） 由ASM流程图，作出状态转移数据表



现 态		次 态			转换条件
编码	状态名	状态名	B	A	
0 (00)	P	R	1	0	$\overline{X}$
		Q	1	1	X
2 (10)	R	P	0	0	0
3 (11)	Q	P	0	0	0
1 (01)	-	P	0	0	0

(2)选择触发器和MUX

设电路选用两个D触发器FA和FB，对应的四选一MUX命名为MUXA和MUXB。MUXA的输出连接FA的D端，MUXB的输出连接FB的D端。触发器的输出QA、QB作为MUX的地址控制端。

现态		次态			转换条件
编码	状态名	状态名	B	A	
0 (00)	P	R	1	0	$\overline{X}$
		Q	1	1	X
2 (10)	R	P	0	0	0
3 (11)	Q	P	0	0	0
1 (01)	-	P	0	0	0

(3)写出MUXA和MUXB的输出端表达式:

根据状态转移表，将次态变量中真值为1的各项按转换条件写出，而真值为0的各项则输入值为0。

MUXA(0)=C=X　MUXB(0)=C=$\overline{X}$+X=1

MUXA(2)=C=0　MUXB(2)=C=0

MUXA(3)=C=0　MUXB(3)=C=0

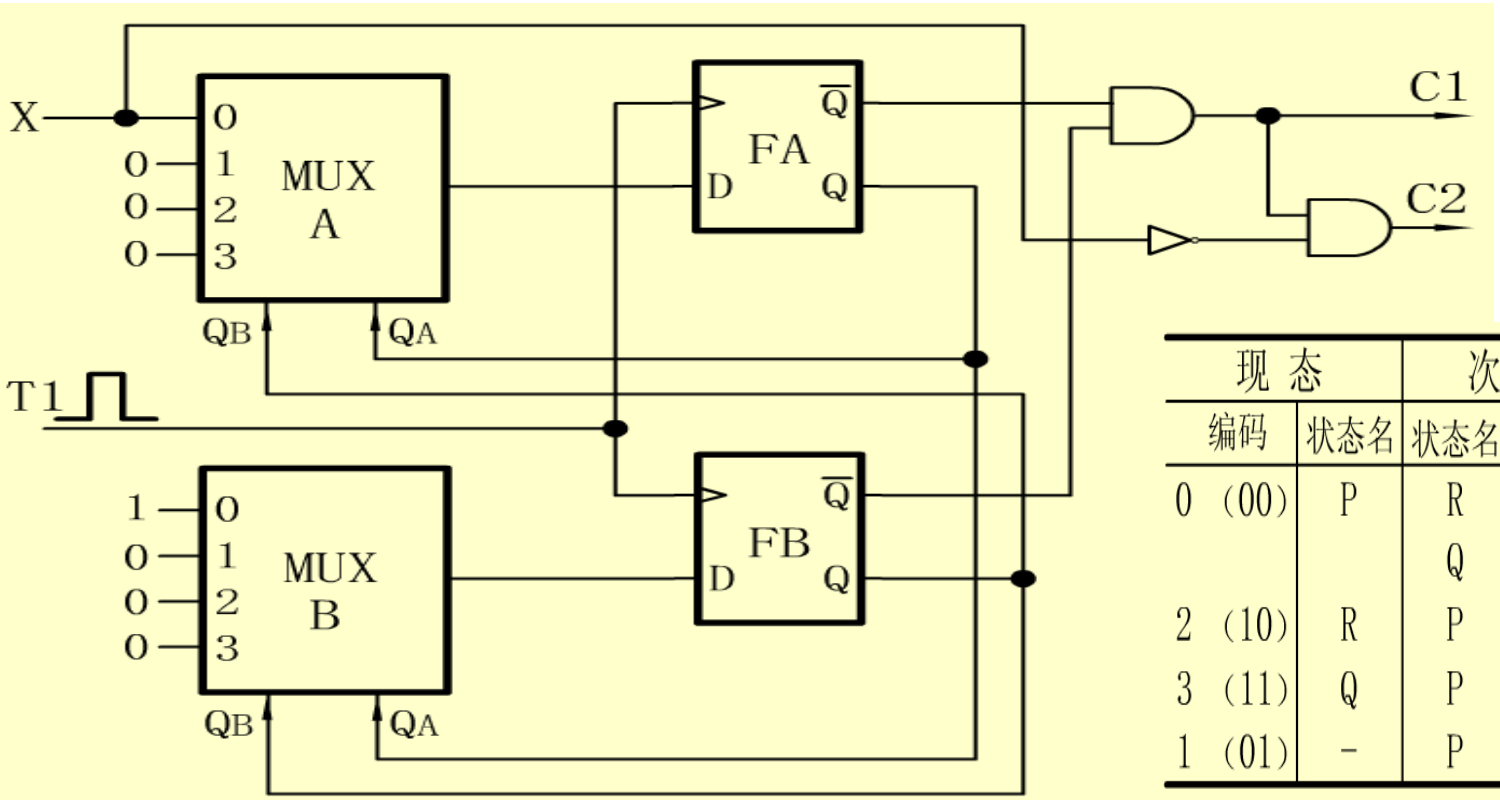
MUXA(1)=C=0　MUXB(1)=C=0

（4）写出控制命令的逻辑表达式

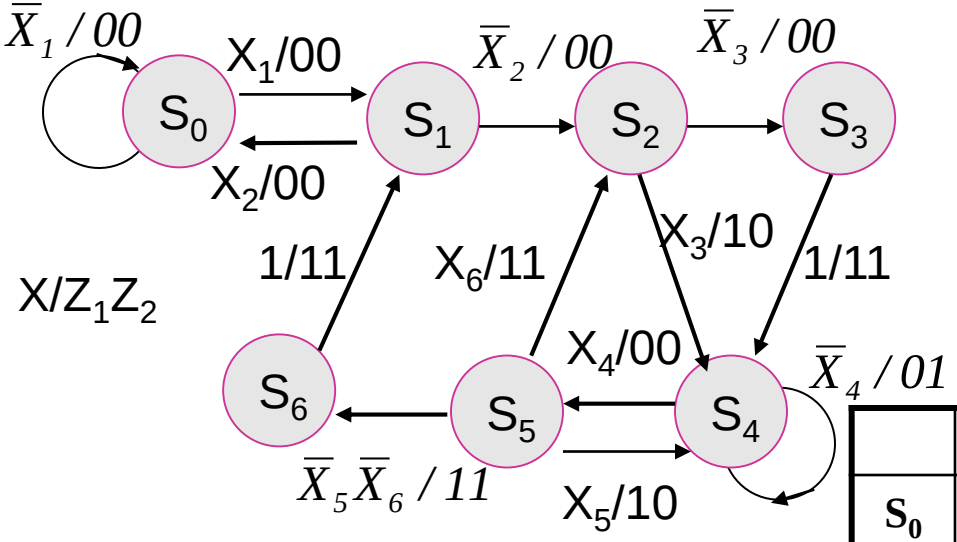
根据ASM流程图，控制命令C_1发生在状态P；命令C_2也发生在状态P，且与输入$\overline{X}$相与。

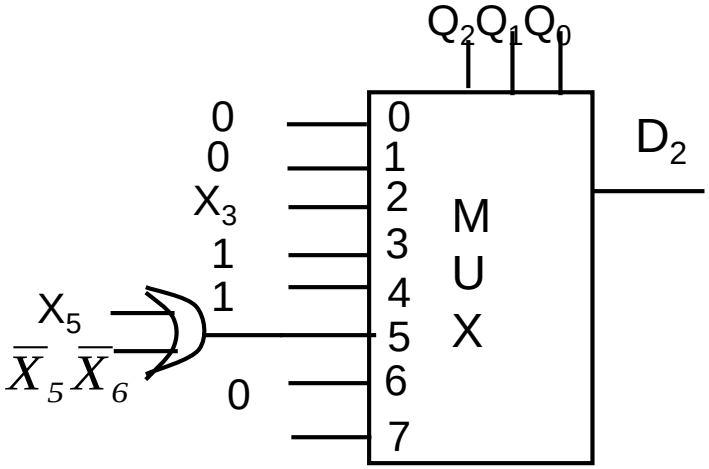
$$C_1=\overline{B}\ \overline{A} \qquad C_2=\overline{B}\ \overline{A}\ \overline{X}$$

（5）画出控制器电路图

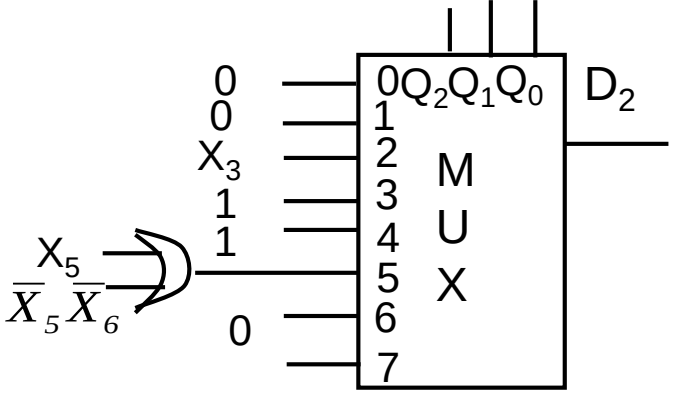


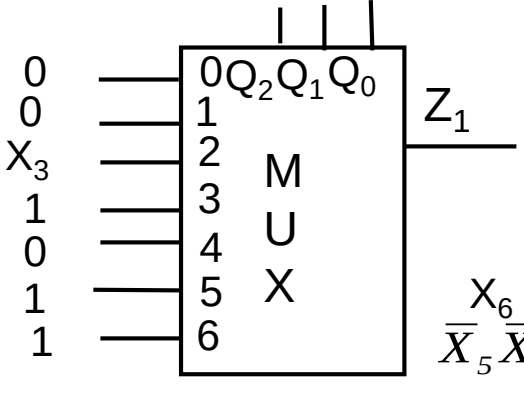
现 态		次 态		转换条件
编码	状态名	状态名	B A	
0 (00)	P	R	1 0	$\overline{X}$
		Q	1 1	X
2 (10)	R	P	0 0	0
3 (11)	Q	P	0 0	0
1 (01)	-	P	0 0	0

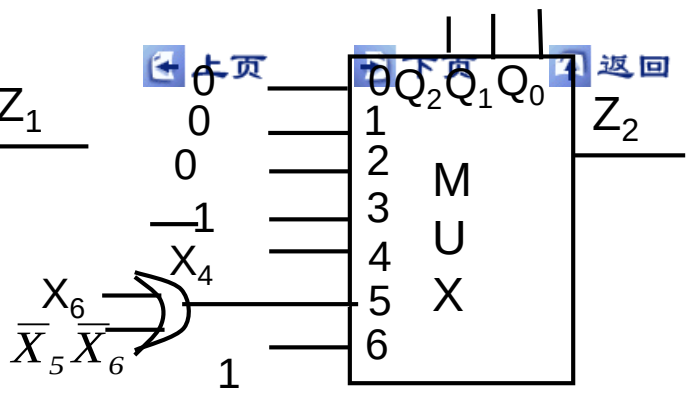


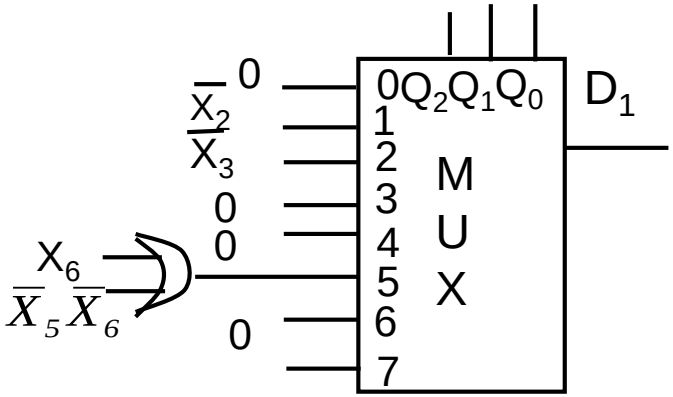


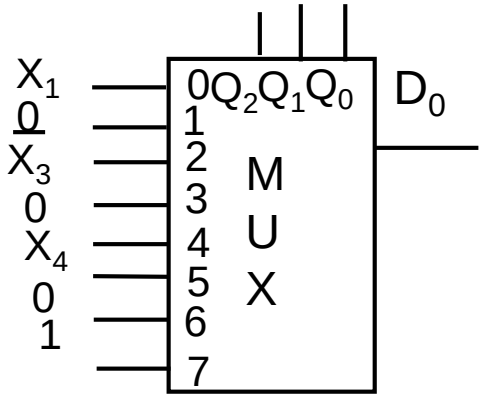
	Q_2	Q_1	Q_0	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	Z_1	Z_2	条件
S_0	0	0	0	0	0	0	0	0	$\overline{X}_1$
				0	0	1	0	0	X_1
S_1	0	0	1	0	0	0	0	0	X_2
				0	1	0	0	0	$\overline{X}_2$
S_2	0	1	0	0	1	1	0	0	$\overline{X}_3$
				1	0	0	1	0	X_3
S_3	0	1	1	1	0	0	1	1	1
S_4	1	0	0	1	0	0	0	1	$\overline{X}_4$
				1	0	1	0	0	X_4
S_5	1	0	1	1	0	0	1	0	X_5
				0	1	0	1	1	X_6
				1	1	0	1	1	$\overline{X}_5\overline{X}_6$
S_6	1	1	0	0	0	1	1	1	1



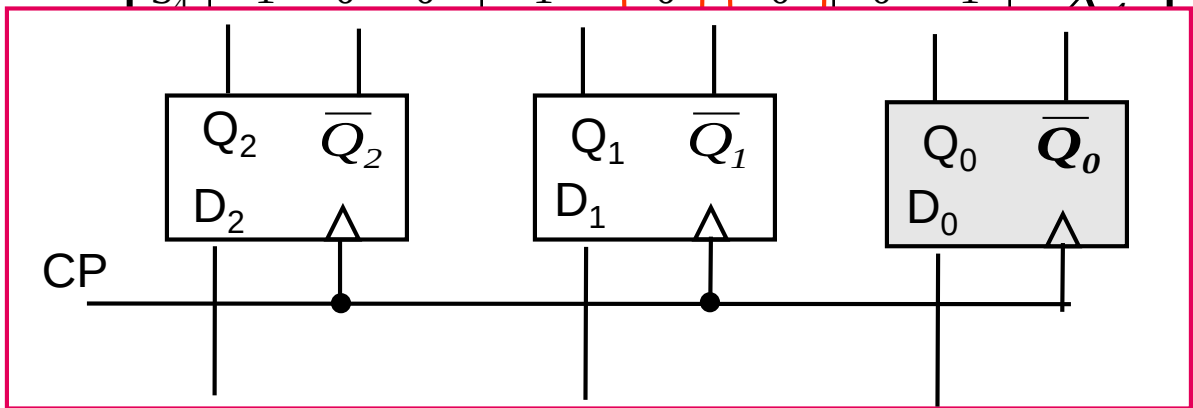


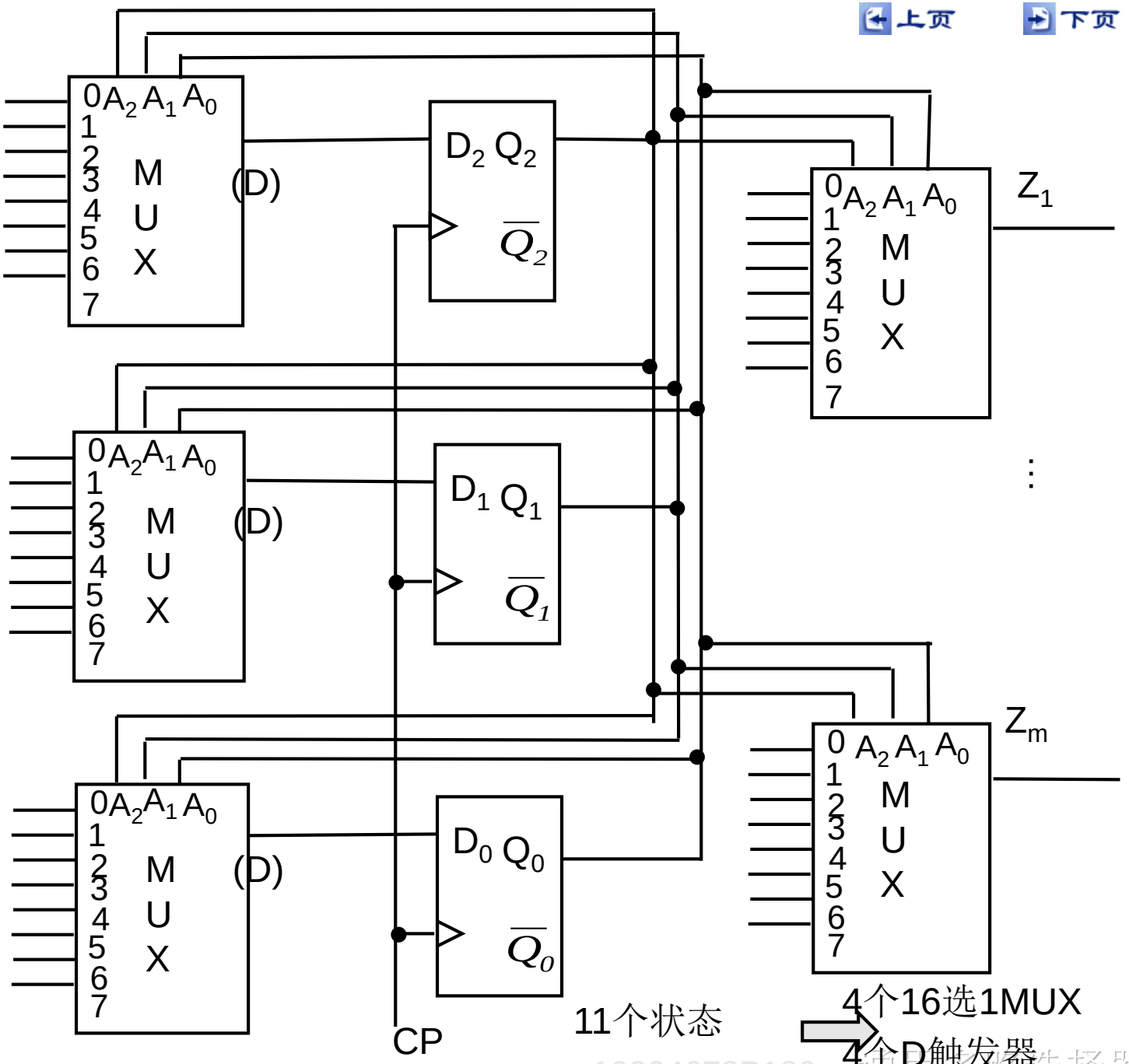


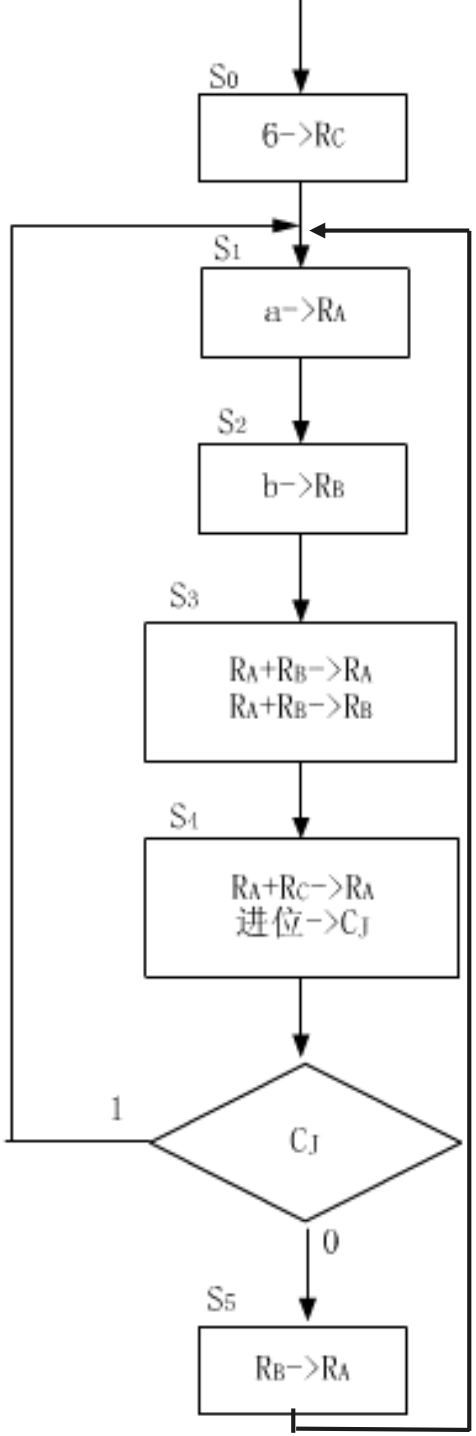




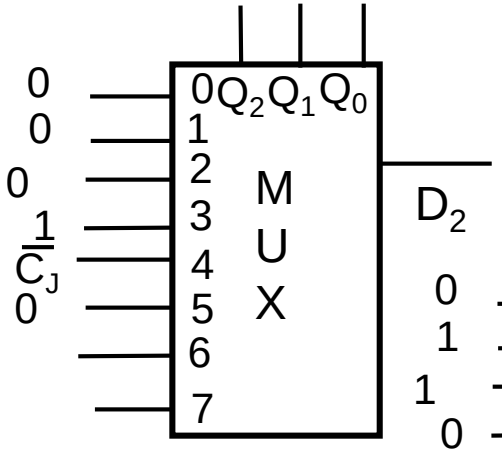
	Q_2	Q_1	Q_0	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	Z_1	Z_2	条件
S_0	0	0	0	0	0	0	0	0	$\overline{X}_1$
				0	0	1	0	0	X_1
S_1	0	0	1	0	0	0	0	0	X_2
				0	1	0	0	0	$\overline{X}_2$
S_2	0	1	0	0	1	1	0	0	$\overline{X}_3$
				1	0	0	1	0	X_3
S_3	0	1	1	1	0	0	1	1	1
S_4	1	0	0	1	0	0	0	1	$\overline{X}_4$

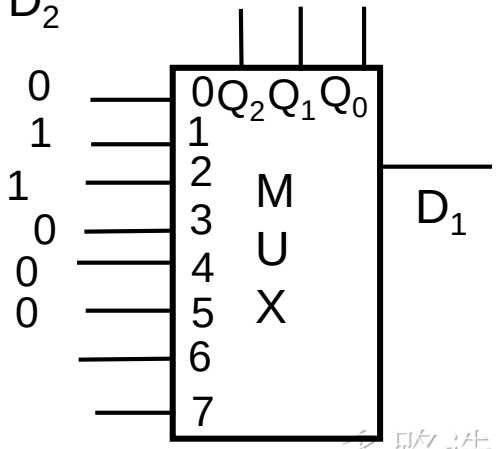


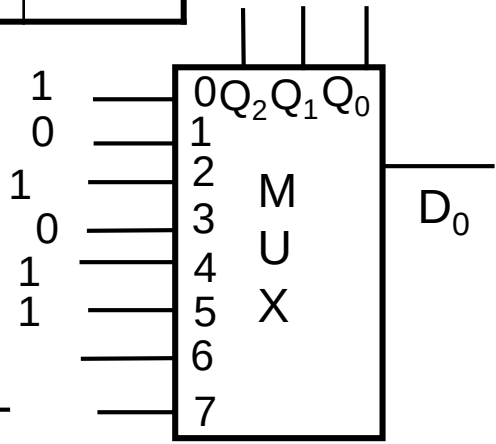


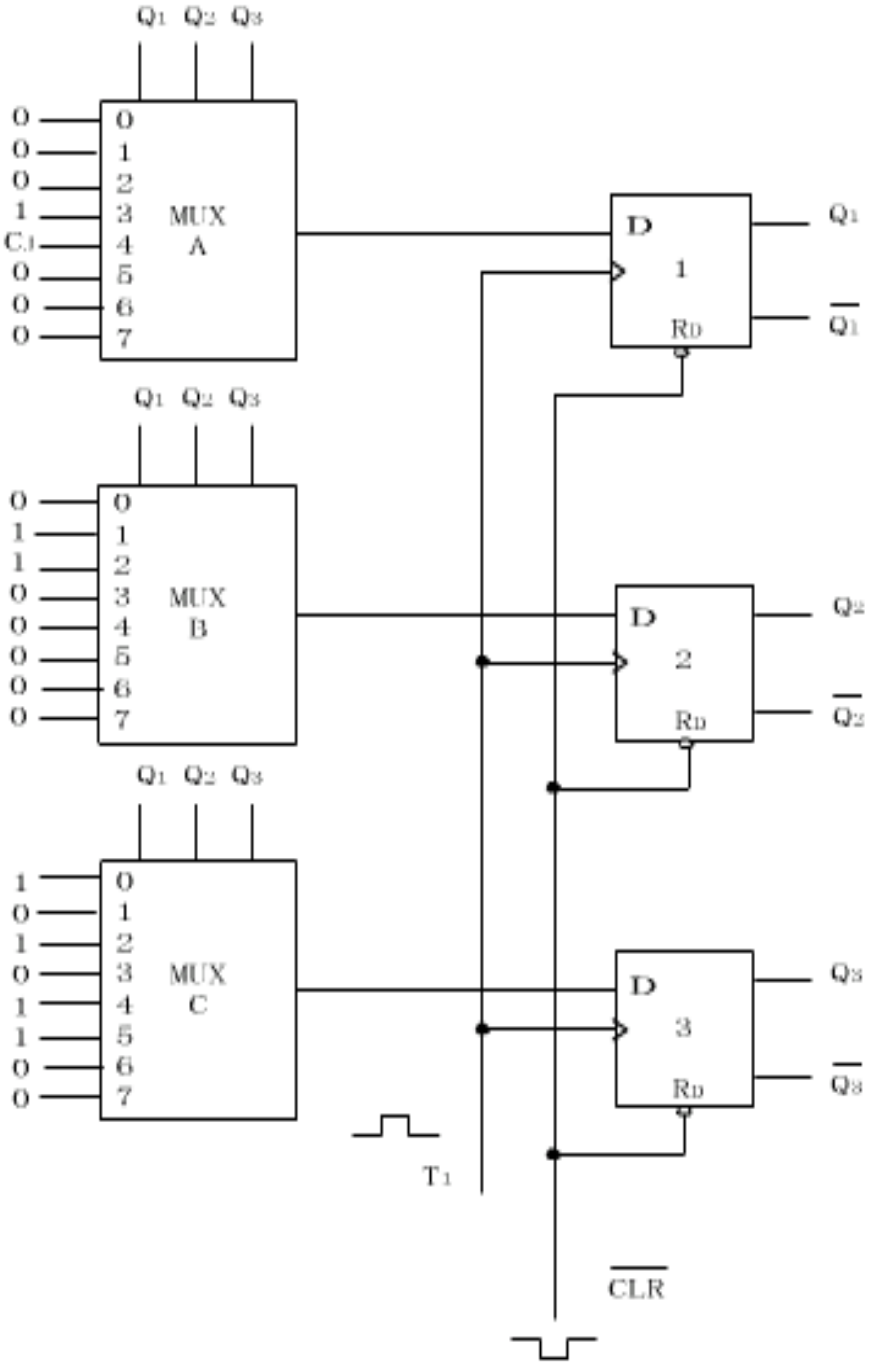


1	Q_2	Q_1	Q_0	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	条件
S_0	0	0	0	S_1	0	0	1	
S_1	0	0	1	S_2	0	1	0	
S_2	0	1	0	S_3	0	1	1	
S_3	0	1	1	S_4	1	0	0	
S_4	1	0	0	S_5	1	0	1	$\overline{C_J}$
				S_1	0	0	1	C_J
S_5	1	0	1	S_1	0	0	1	









$$6 \rightarrow R_C = \overline{Q}_3\overline{Q}_2\overline{Q}_1$$

$$a \rightarrow R_A = \overline{Q}_3\overline{Q}_2Q_1$$

$$b \rightarrow R_B = \overline{Q}_3Q_2\overline{Q}_1$$

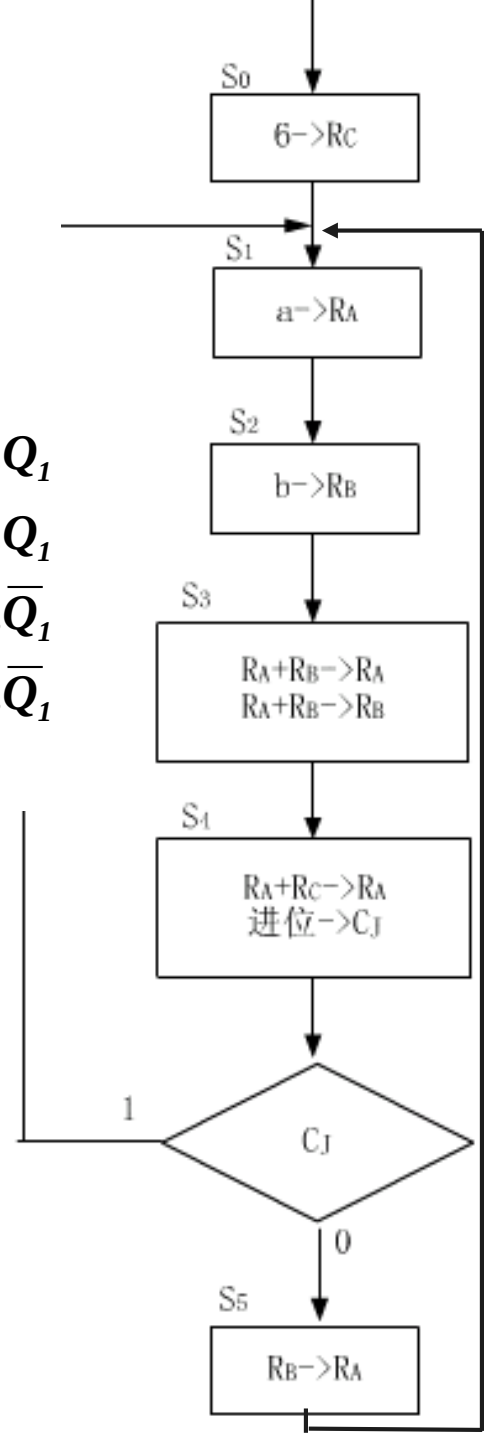
$$R_A + R_B \rightarrow R_A = \overline{Q}_3Q_2Q_1$$

$$R_A + R_B \rightarrow R_B = \overline{Q}_3Q_2Q_1$$

$$R_A + R_C \rightarrow R_A = Q_3\overline{Q}_2\overline{Q}_1$$

进位→C_J $= Q_3\overline{Q}_2\overline{Q}_1$

$$R_B \rightarrow R_A = Q_3\overline{Q}_2Q_1$$



6.4.4 定序型控制器

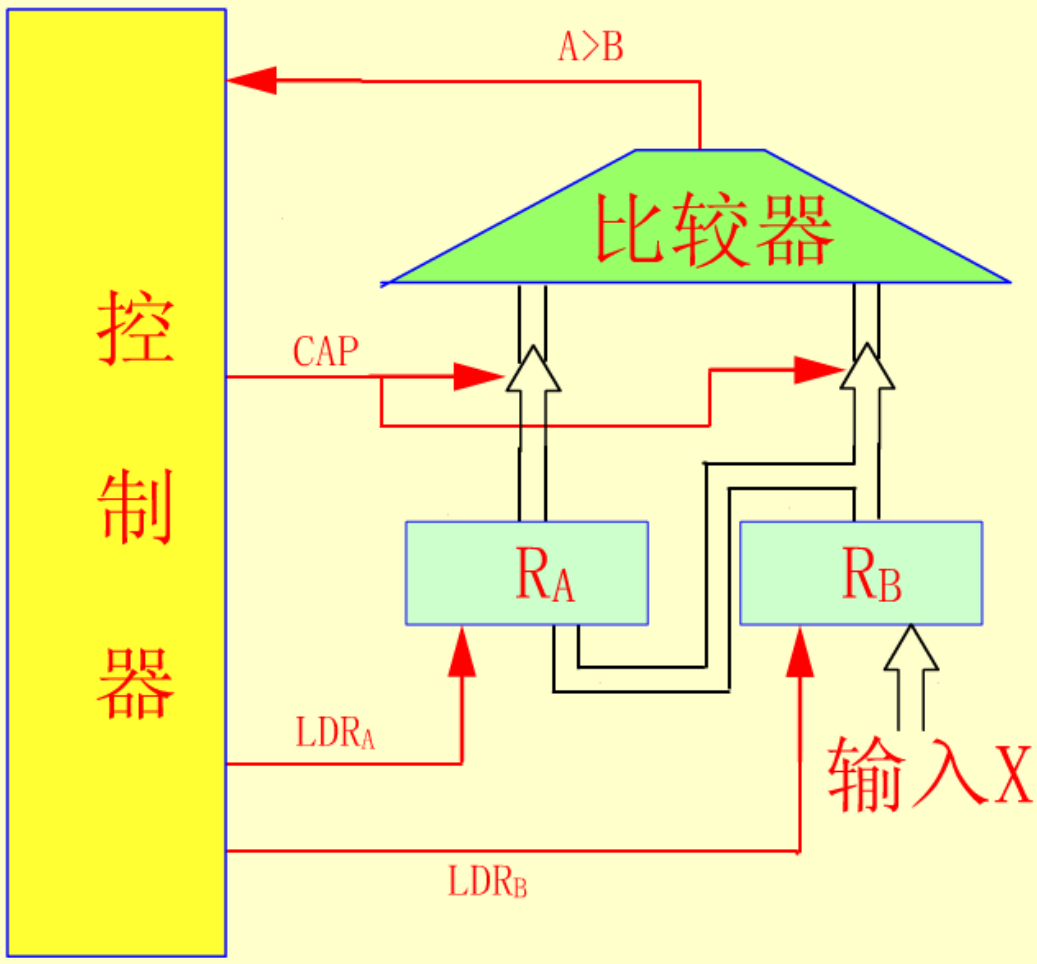
定序型控制器的设计思想：

定序型控制器的基本思想是一对一法，即ASM流程图中有多少个状态，则使用多少个触发器，并依赖一组最新的代码实现状态转换。

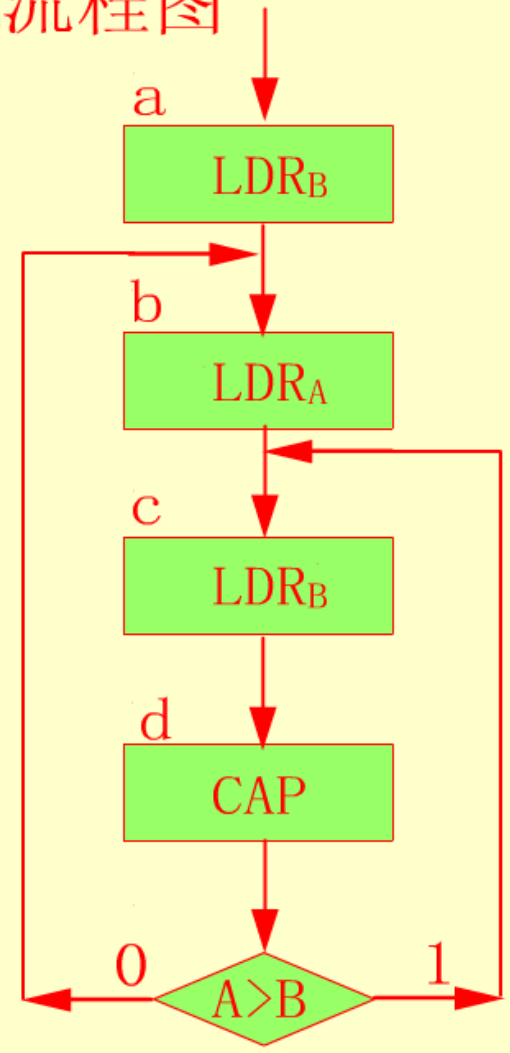
【例9】有一个数字比较系统，它能对两个二进制数进行比较。操作过程如下：先把两个数存入寄存器R_A和R_B，然后进行比较，最后将大数移入寄存器R_A中。系统的方框图与ASM流程图如图6.18所示，请设计定序型控制器。

6.4.4 定序型控制器

数字比较系统框图及ASM流程图



(a) 框图



(b) ASM流程图

解：

（1）分析已知条件

该系统由控制器和执行部件两大部分组成，其中X为并行输入数据。A>B是比较结果指示信号（送往控制器）。控制器发出的三个控制命令是：

LDR_A--寄存器R_A打入控制信号

LDR_B--寄存器R_B打入控制信号

CAP--A数和B数送入比较器的使能控制信号

（2）用一对一法对ASM流程图6.19（a）的状态进行编码

(3)触发器命名为Q_a、Q_b、Q_c、Q_d，列出状态转移表。

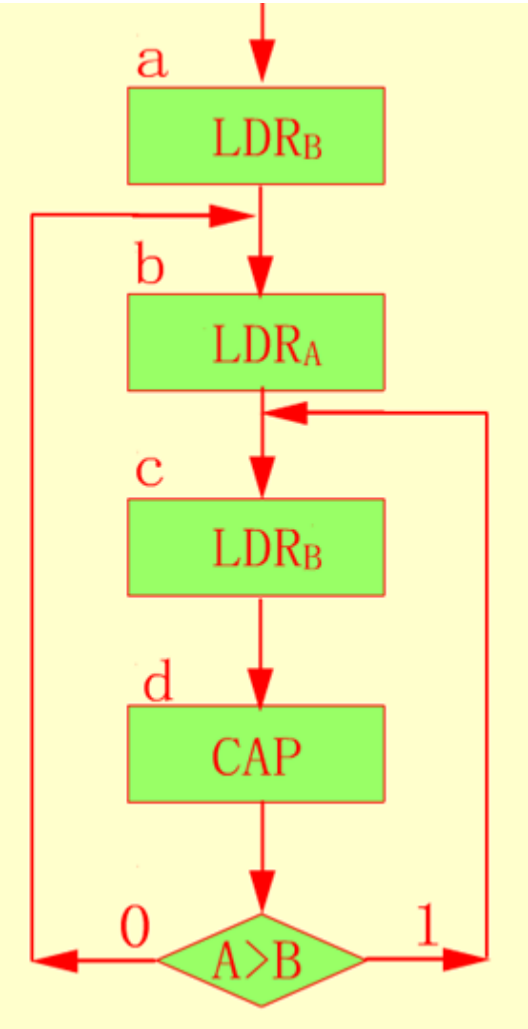


表6.5 状态转移真值表

PS（现态）				NS（次态）				转移条件 C
Q_a	Q_b	Q_c	Q_d	$Q_a(D)$	$Q_b(D)$	$Q_c(D)$	$Q_d(D)$	
1	0	0	0	0	1	0	0	四个触发器初始化清 0
0	1	0	0	0	0	1	0	
0	0	1	0	0	0	0	1	
0	0	0	1	0	0	1	0	A> B
				0	1	0	0	A<=B

（4）选用D触发器，按NS=∑PS·C公式写出次态激励方程：

$$Q_a(D)=\overline{Q_a+Q_b+Q_c+Q_d}$$

$$Q_b(D)=Q_a+\overline{(A>B)}\cdot Q_d$$

$$Q_c(D)=Q_b+Q_d\cdot(A>B)$$

$$Q_d(D)=Q_c$$

（5）由ASM流程图写出控制命令表达式

$LDR_B=(Q_a+Q_c)\cdot T_2$ ；脉冲控制信号

$LDR_A=Q_b\cdot T_2$ ；脉冲控制信号

$CAP=Q_d$ ；电位控制信号

其中LDR_B和LDR_A是脉冲控制信号，需要用T_2节拍时间相与。

（6）画出控制器的具体电路图

$$Q_a(D) = \overline{Q_a + Q_b + Q_c + Q_d}$$

$$Q_b(D) = Q_a + \overline{(A>B)} \cdot Q_d$$

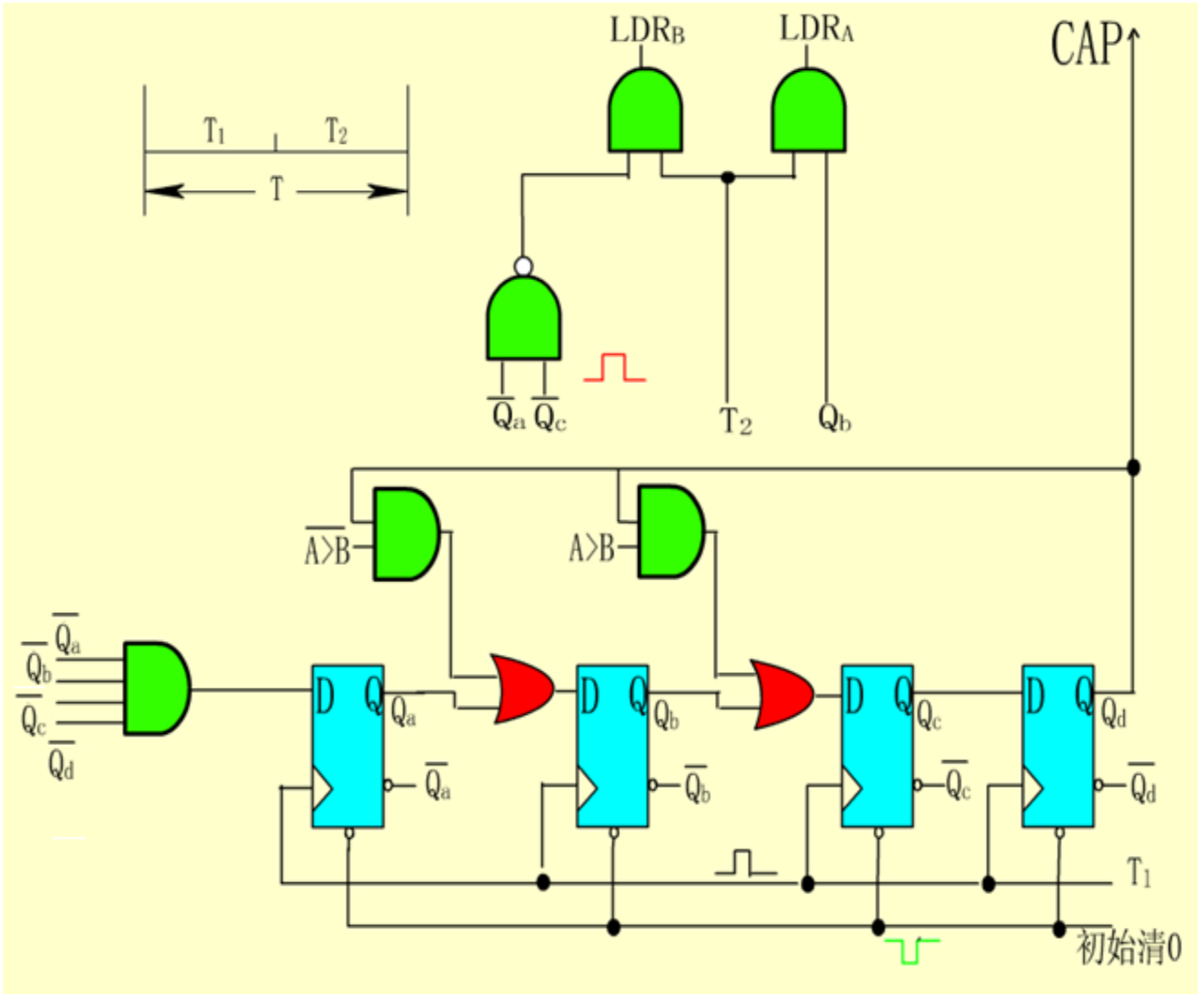
$$Q_c(D) = Q_b + Q_d \cdot (A>B)$$

$$Q_d(D) = Q_c$$

$$LDR_B = (Q_a + Q_c) \cdot T_2$$

$$LDR_A = Q_b \cdot T_2$$

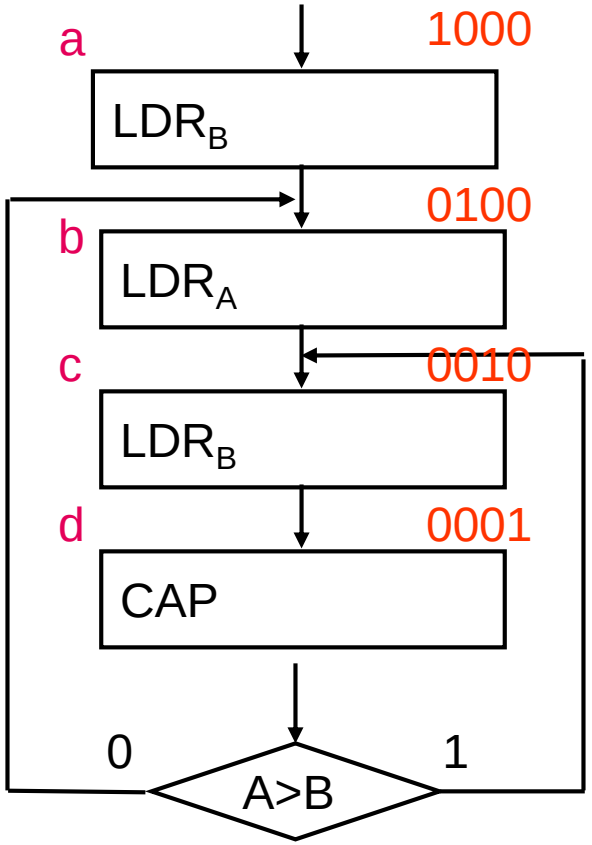
$$CAP = Q_d$$



三、定序型控制器

基本思想 ..... 一个状态对应一个触发器，一对一法

现态				次态				
Q_a	Q_b	Q_c	Q_d	$Q_a(D)$	$Q_b(D)$	$Q_c(D)$	$Q_d(D)$	转移条件
1	0	0	0	0	1	0	0	初始Qa=1
0	1	0	0	0	0	1	0	
0	0	1	0	0	0	0	1	
0	0	0	1	0	0	1	0	A>B
0	0	0	1	0	1	0	0	$\overline{A>B}$



$$Q_a(D)=0$$

$$Q_b(D)=Q_a+Q_d(\overline{A>B})$$

$$Q_c(D)=Q_b+Q_d(A>B)$$

$$Q_d(D)=Q_c$$

$$LDR_B=(Q_a+Q_c)T_2$$

$$LDR_A=Q_bT_2$$

$$CAP=Q_d$$

$$Q_a(D)=0$$

$$Q_b(D)=Q_a+Q_d(\overline{A>B})$$

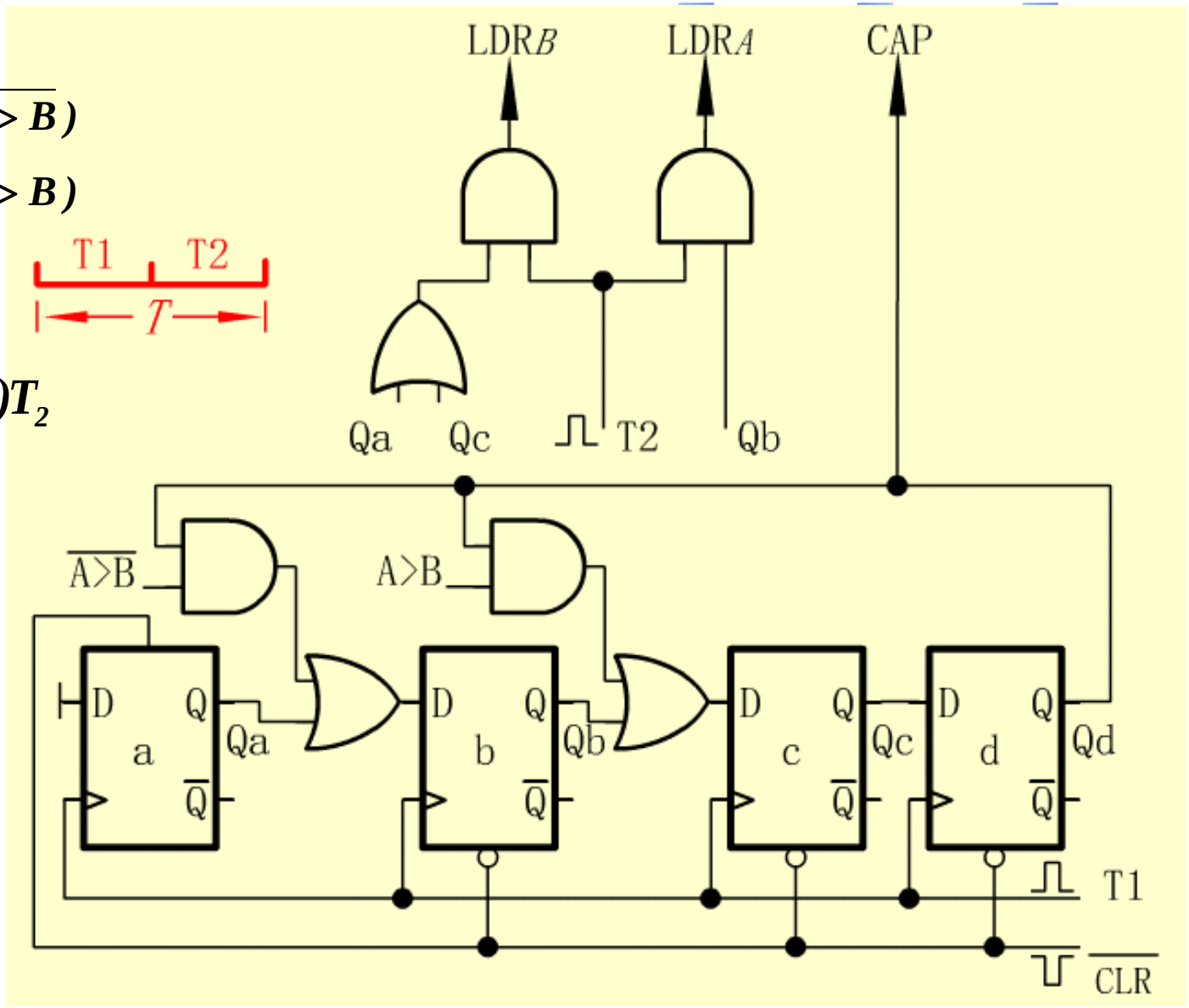
$$Q_c(D)=Q_b+Q_d(A>B)$$

$$Q_d(D)=Q_c$$

$$LDR_B=(Q_a+Q_c)T_2$$

$$LDR_A=Q_bT_2$$

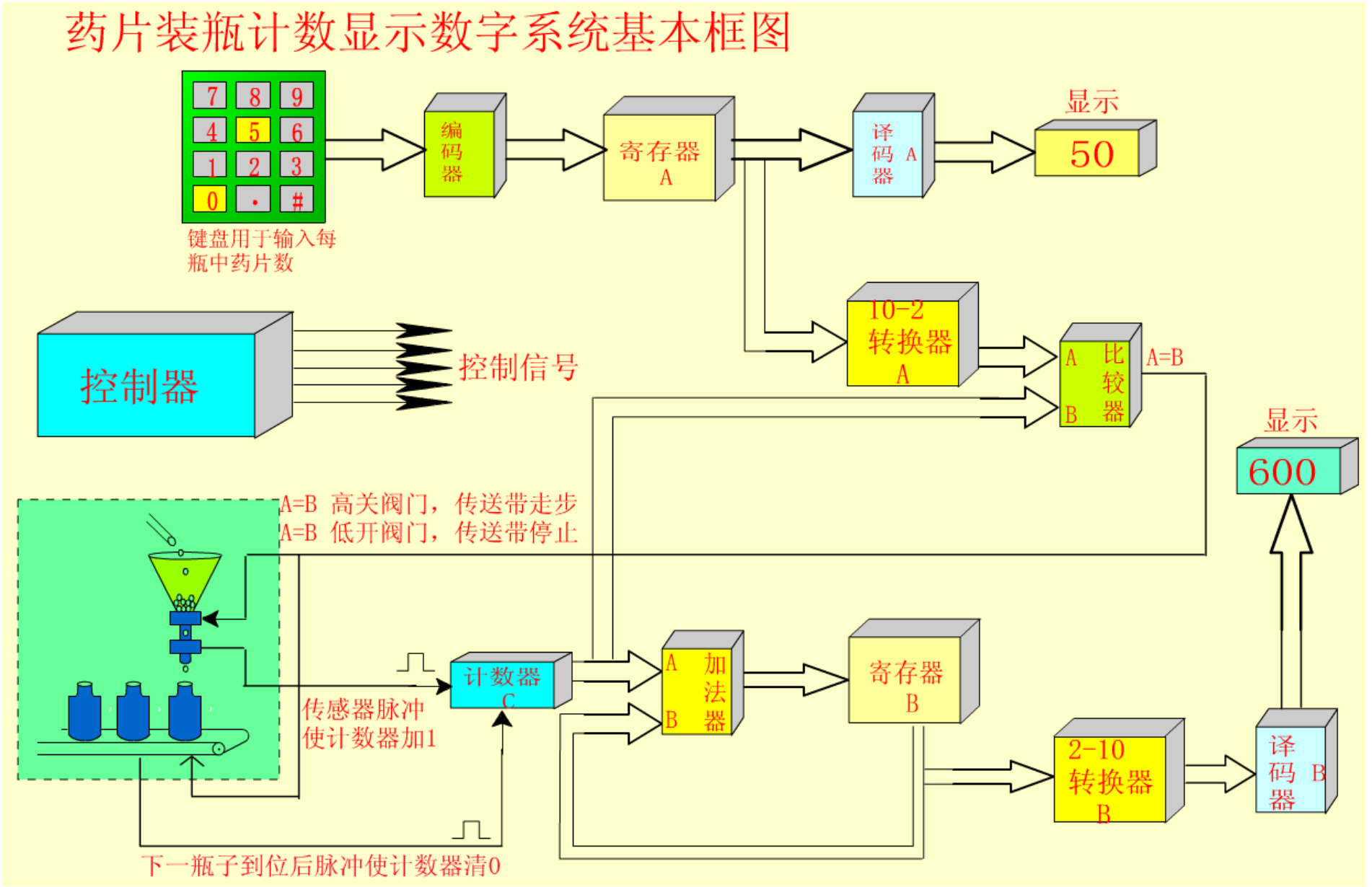
$$CAP=Q_d$$



6.5 数字系统设计实例

6.5.1 由顶向下—子系统的划分
6.5.2 小型控制器的实现方案

6.1.1 一个数字系统实例



导入

本章前面介绍的数字系统设计方法是一种由顶向下的方法，其过程大致分为三步：

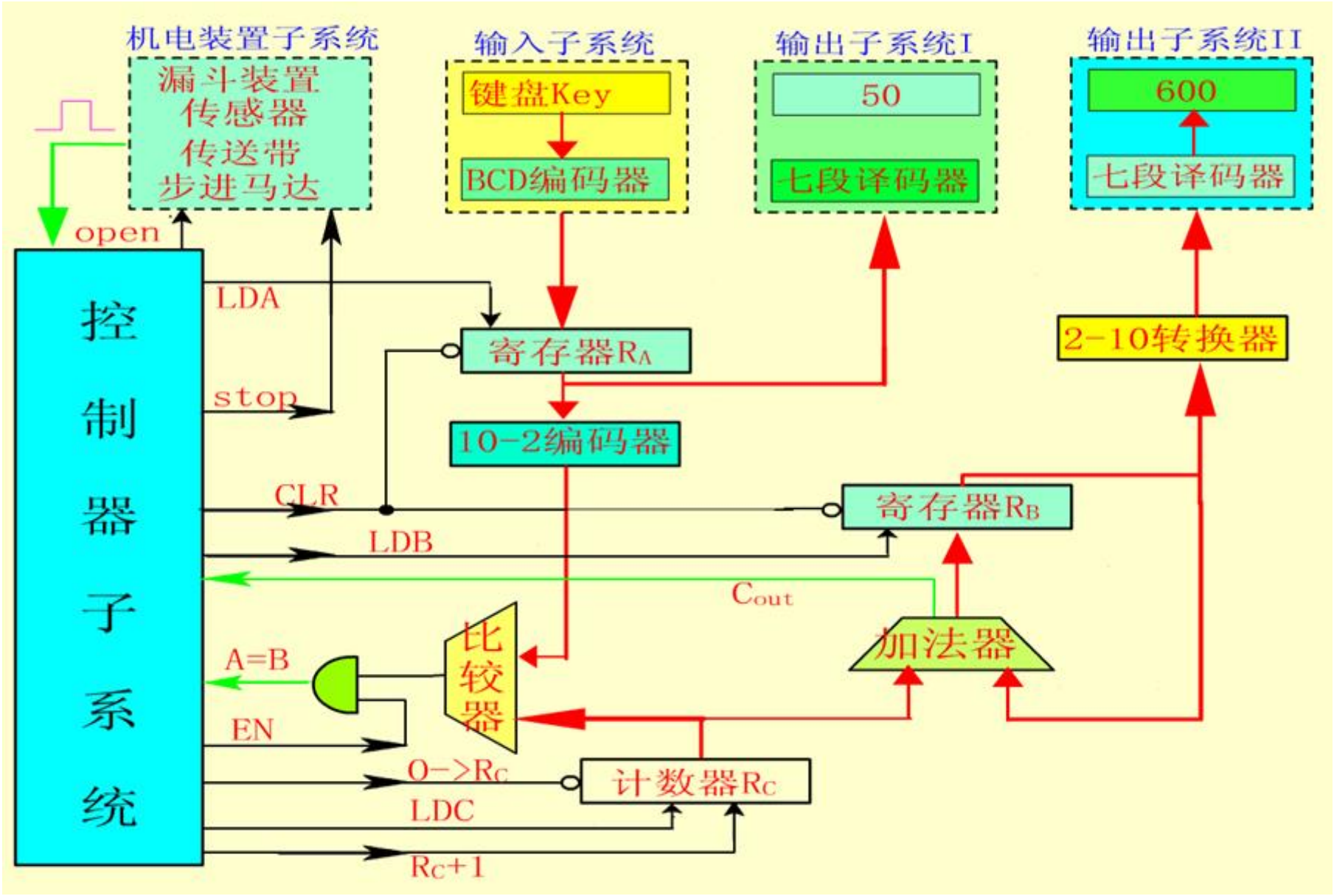
①确定初步方案；

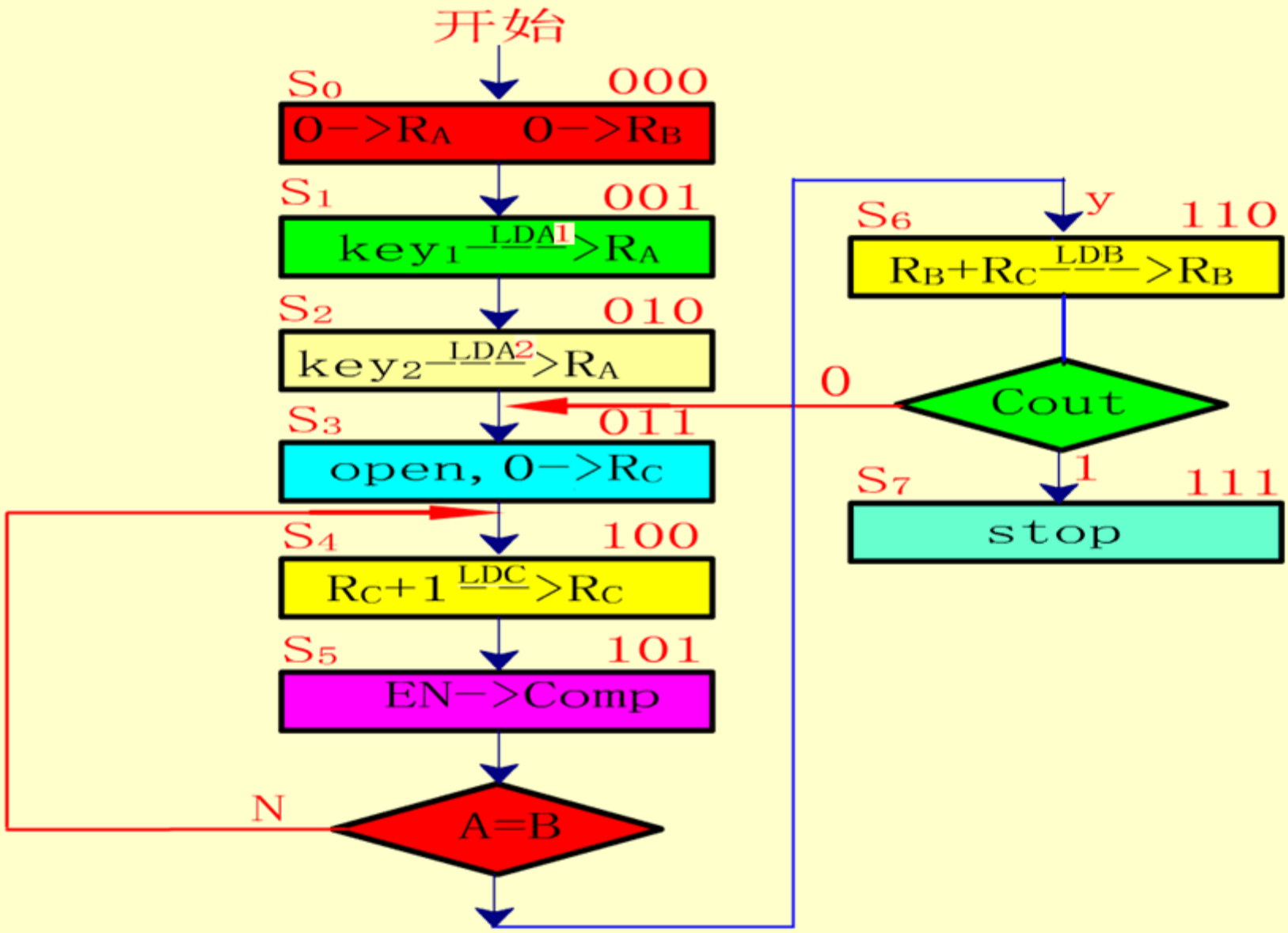
②子系统划分，确定详细方案；

③选用子系统，完成具体设计。

下面通过药片装瓶控制数字系统设计实例，进一步体验数字系统的设计方法和过程，并取得实践经验。

6.5.1 由顶向下—子系统的划分





药片装瓶控制数字系统ASM流程图

表 6.8 状态转移真值表

PS				NS				
状态名	A	B	C	状态名	$A(D)$	$B(D)$	$C(D)$	转移条件
S_0	0	0	0	S_1	0	0	1	
S_1	0	0	1	S_2	0	1	0	
S_2	0	1	0	S_3	0	1	1	
S_3	0	1	1	S_4	1	0	0	
S_4	1	0	0	S_5	1	0	1	
S_5	1	0	1	S_4	1	0	0	$(A=B)=0$
S_5	1	0	1	S_6	1	1	0	$(A=B)=1$
S_6	1	1	0	S_3	0	1	1	$C_{out}=0$
S_6	1	1	0	S_7	1	1	1	$C_{out}=1$

利用NS=ΣPC·C公式写出三个D触发器激励表达式：

$A(D)=\overline{A}BC+A\overline{B}+AB\overline{C}\cdot C_{out}$

$B(D)=\overline{A}\overline{B}C+\overline{A}B\overline{C}+A\overline{B}C\ (A=B)$

$C(D)=\overline{A}\overline{C}+A\overline{B}\overline{C}+AB$

然后写出控制命令表达式如下：

$CLR(0\rightarrow R_A\cdot R_B)=S_0=\ \overline{A}\overline{B}\overline{C}$（电位）

$LDA_1(key_1\rightarrow R_A)=S_1\cdot T_2=\ \overline{A}\overline{B}\overline{C}\cdot T_2$（脉冲）

$LDA_2(key_2\rightarrow R_A)=S_2\cdot T_2=\ \overline{A}B\overline{C}\cdot T_2$（脉冲）

$open=S_3=\ \overline{A}BC$（电位）

$0\rightarrow R_C=S_3=\ \overline{A}BC$（电位）

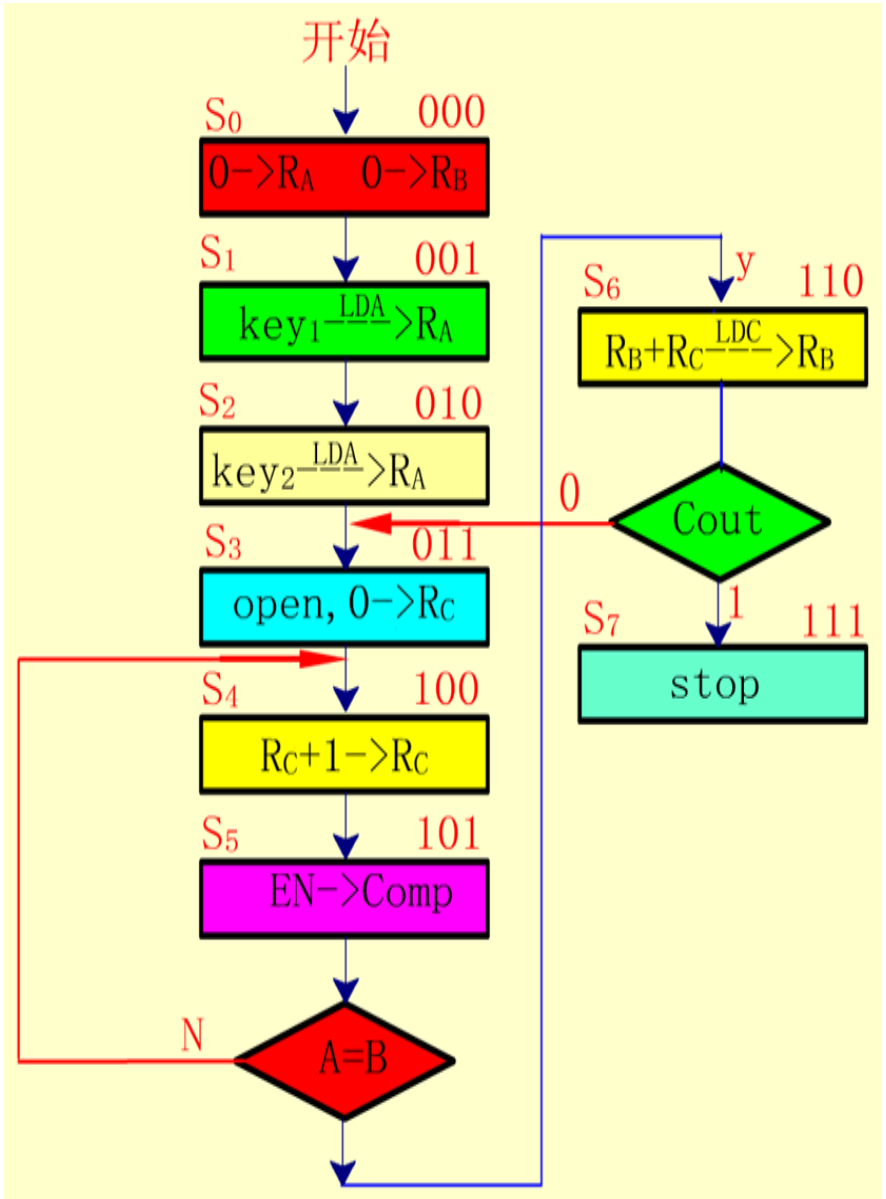
$R_C+1=S_4=\ A\overline{B}\overline{C}$（电位）

$LDC=S_4\cdot T_2=\ A\overline{B}\overline{C}\cdot T_1$（脉冲）

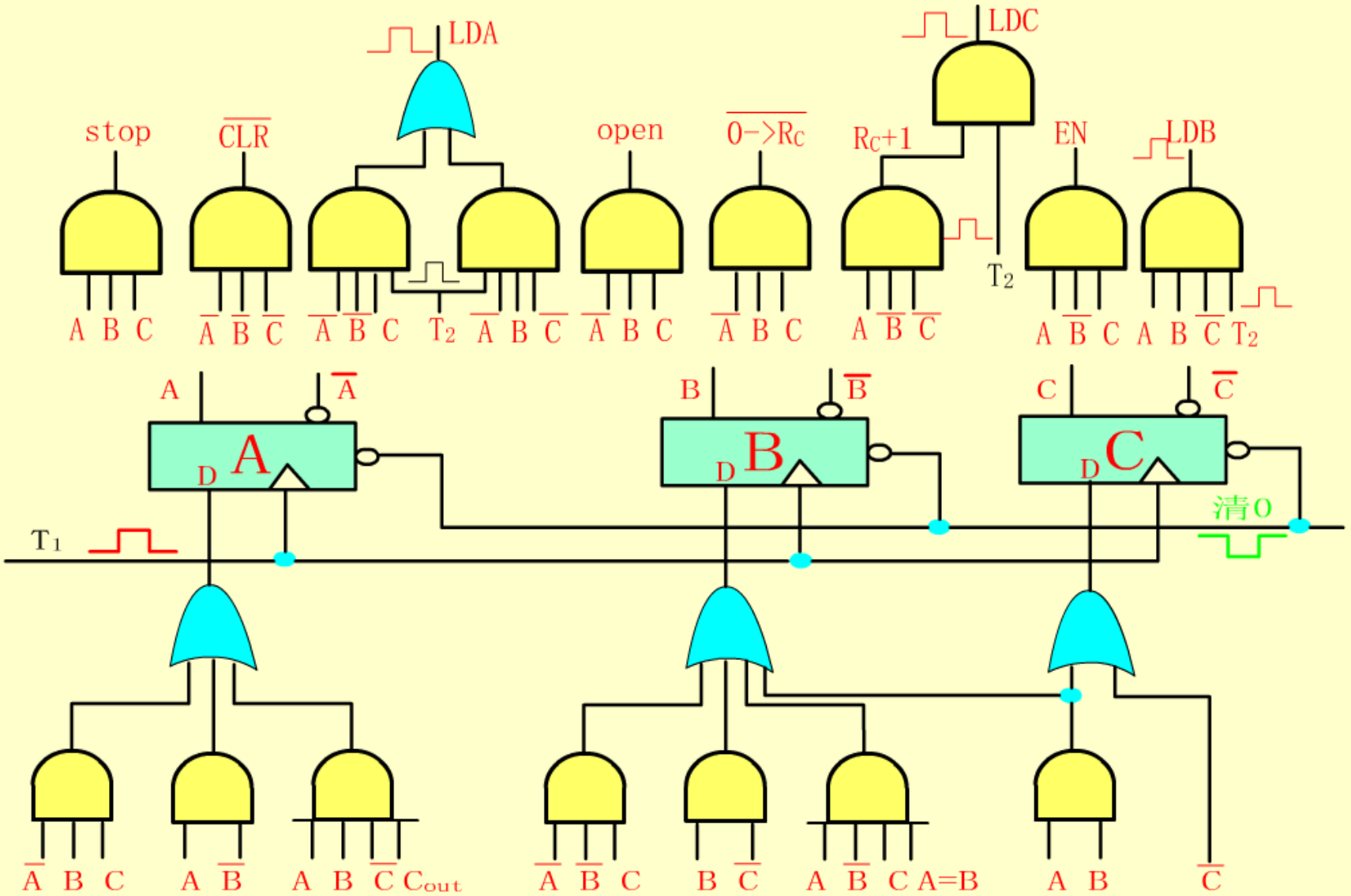
$EN=S_5=\ A\overline{B}C$（电位）

$LDB=S_6\cdot T_2=\ AB\overline{C}\cdot T_2$（脉冲）

$stop=S_7=\ ABC$（电位）



小型控制器逻辑方案图


LDA
LDC
stop
CLR
open
0->RC
RC+1
EN
LDB
T2
A B C
A B C
A B C T2 A B C
A B C
A B C
A B C
A B C A B C T2
A
B
C
D A
D B
D C
T1
清0
A B C
A B
A B C Cout
A B C
B C
A B C A=B
A B
C