

第五章 数字系统设计

Digital System Design

5.1 数字系统概述

5.1.1 信息处理单元的构成

5.1.2 控制单元CU的构成 处理器

5.1.3 数字系统设计的描述工具

5.1.3.1 方框图

5.1.3.2 定时图 (时序图、时间关系图)

5.1.3.3 逻辑流程图

5.1.3.4 ASM图

5.1.3.5 寄存器传送语言

5.2 基本数字系统设计举例

5.3 简易计算机设计

5.3.1 简易计算机结构

5.3.2 举例：设计一台简易计算机 手搓

5.4 A/D转换和D/A转换

第五章 数字系统设计

Digital System Design

组合逻辑电路和时序逻辑电路只能完成某些特定的逻辑功能，属功能部件级。电路分析和设计是建立在真值表、卡诺图、逻辑方程式、状态表和状态图的工具基础上，主要依赖于设计者的熟练技巧和经验，称“**凑试法**”。

若由功能部件级组成一个功能复杂、规模较大的数字系统时，虽然在理论上仍可以把它看成是一个大型时序逻辑电路，仍可以采用凑试法，但实际实现上很难、甚至无法达到完整地描述其逻辑功能。因为这种设计方法：**原始、受限制最多、效率与效果均欠佳、局限性大**。要用方框图、定时图、逻辑流程图、ASM图等系统描述工具。

对数字系统进行分析和设计时，通常把系统从逻辑上划分成**控制单元CU**和**信息处理单元**两大部分。其中：

信息处理单元对信息进行不同的处理和传递，

控制单元保证信息处理单元按规定的微操作序列处理数据。

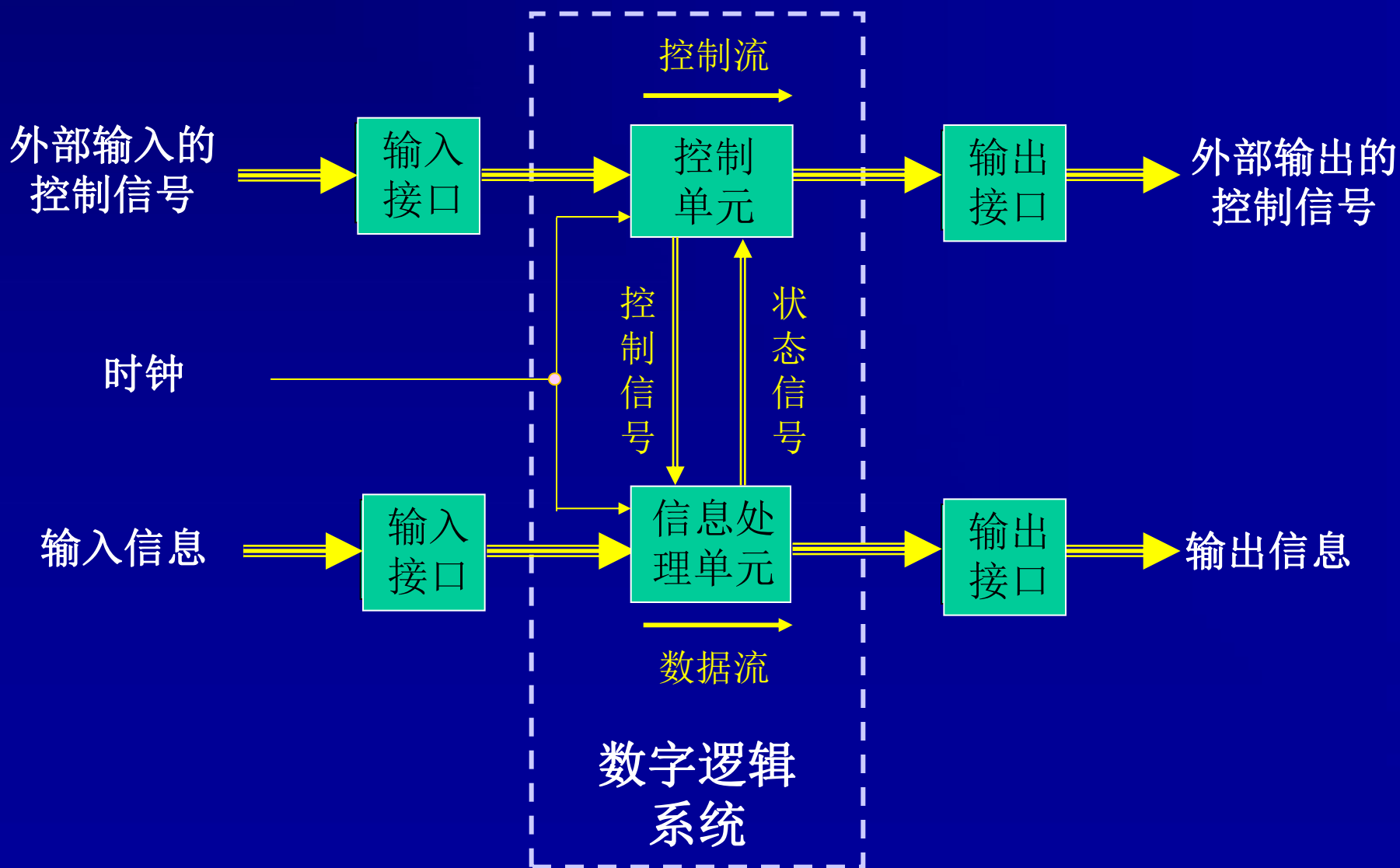
控制单元——**不断生成和发送控制信号序列**，控制信息处理单元不断地执行特定的操作；

——接收来自信息处理单元的**状态信息**，用以选择下一个需执行的操作。

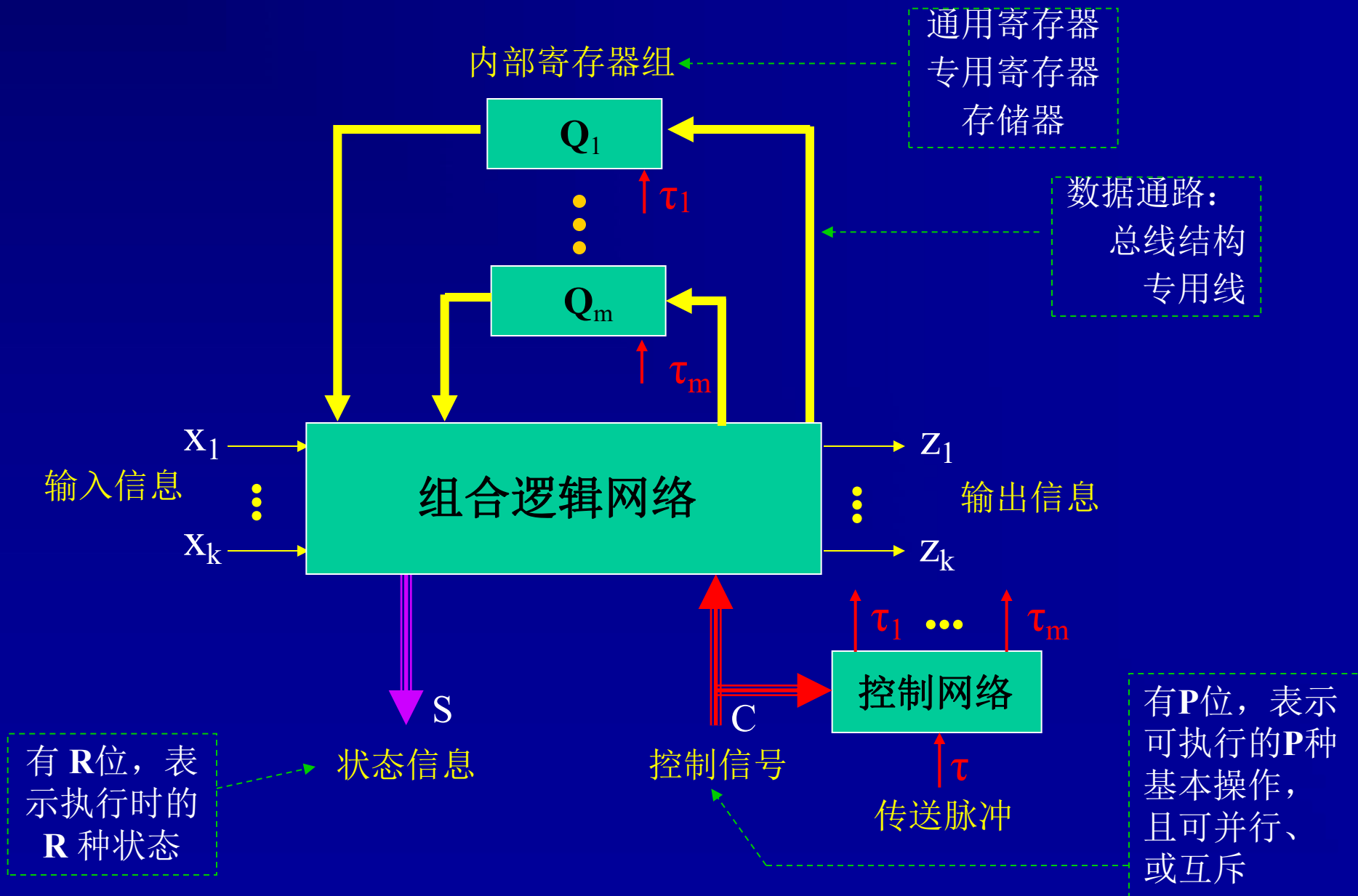
——接收外来的控制信息，用以改变正在执行的操作序列。

控制单元是区别**数字系统**与**功能部件**的**标志**。

5.1 数字系统概述



5.1.1 信息处理单元的构成



状态信息表及操作表举例

S的编码位	该编码位定义的状态标志
S_1	$S_1 := (x > 0)$
S_2	$S_2 := (x = 0)$
S_3	$S_3 := (x < 0)$
\vdots	\vdots
S_r	$S_r := (Q_1 = x) \vee (Q_2 = x)$

(a) 状态信息表

指令

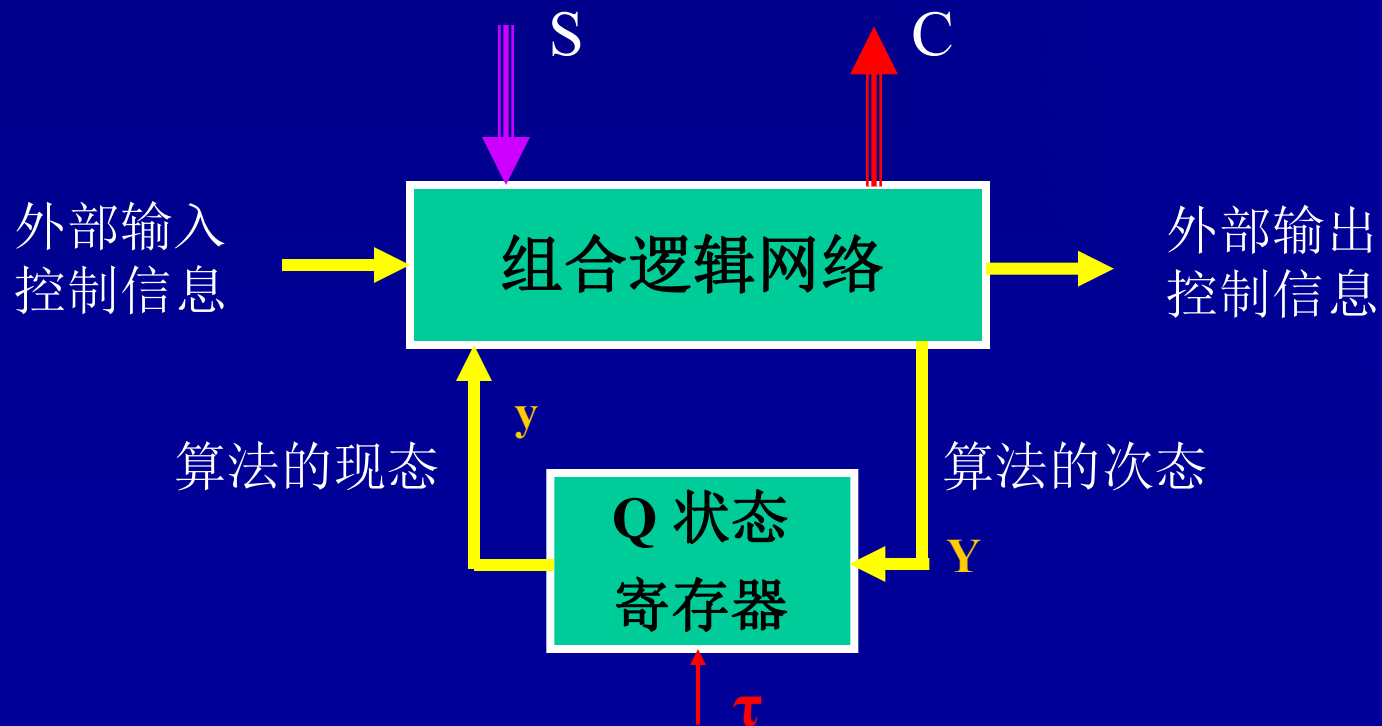
控制信号C	执行的操作
$C_1 := \text{CLR}$	$Q_1 \leftarrow 0$ ^{CLR.} 清0
$C_2 := \text{ADD}$	$Q_1 \leftarrow Q_1 + x$
$C_3 := \text{SUB}$	$Q_1 \leftarrow Q_1 - x$
\vdots	\vdots
$C_p := \text{INC}$	$Q_1 \leftarrow Q_1 + 1$

(b) 操作表

5.1.2 控制单元CU的构成

将数字系统执行的复杂任务转化为一个操作和测试序列，称为“算法”。

用控制单元产生与操作序列相对应的控制信号序列，每一个控制信号控制信息处理单元执行与算法相关的一个操作。所以，控制单元的基本功能具体上是对指令流和数据流实施时间上和空间上的正确的控制。



5.1.2 控制单元CU的构成

控制单元的**核心**是时序电路，本质上是一个**状态寄存器**。状态寄存器主要有两个功能——寄存控制单元的**现态**，生成**次态**。采用触发器作为状态寄存器的元件。

存在着两种不同的控制单元实现方法：

硬件逻辑方法——用**逻辑电路**生成每一个微操作的控制信号；特点：速度快、动一发而动全身。

微程序方法——计算机的每一条指令的功能通过**执行一个微指令序列**(微程序)来实现的。设计好的微程序被固化在只读存储器中，这个存储器称为**控制存储器**。特点：速度低、但设计、修改及扩充容易。

显然，采用不同的实现方法，将影响控制单元的组成和结构。

数字系统逻辑设计的基本步骤

第一步：确定系统的逻辑功能。

第二步：确定系统方案。

第三步：对系统进行逻辑划分。

第四步：设计信息处理单元和控制单元。

5.1.3 数字系统设计的描述工具

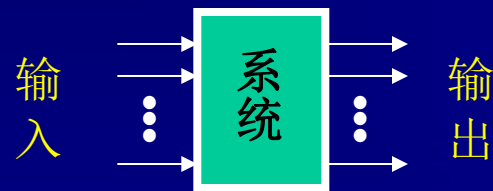
5.1.3.1 方框图

设计系统首先应当**建立模型**，方框图是描述模型最常用、最重要的工具。

- 每一个**方框定义**了一个模块；
- 方框内用文字、表达式、例行符号、图形等**表示**该模块的名称或主要功能；
- 方框之间用**指向线**相连，表示模块之间的数据流或信息流的信息通道及方向。

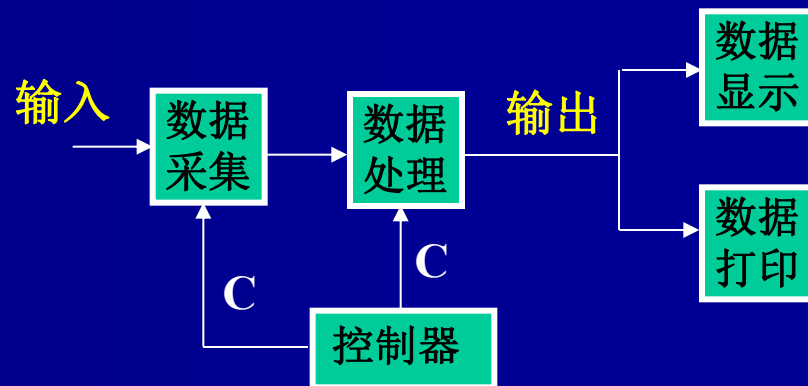
方框图的设计过程：**自顶而下、逐步细化**。

例 一个智能仪表的方框图。



(a)

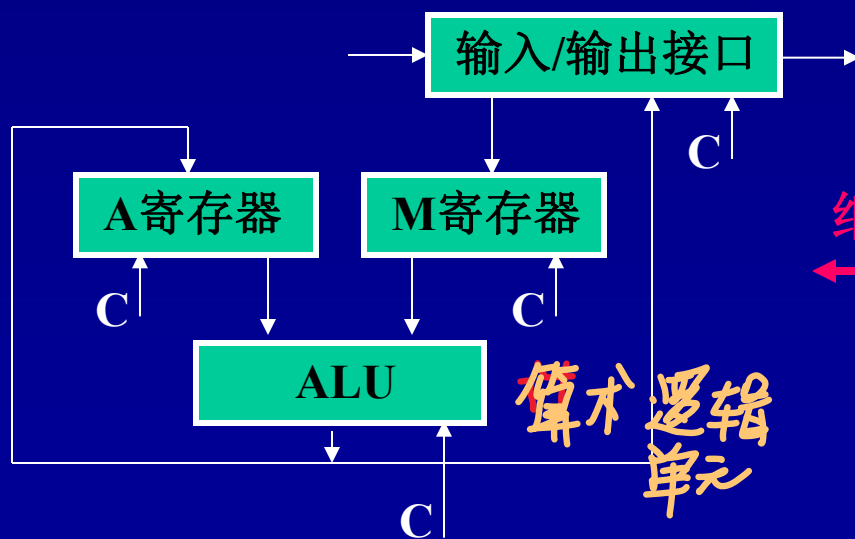
分解



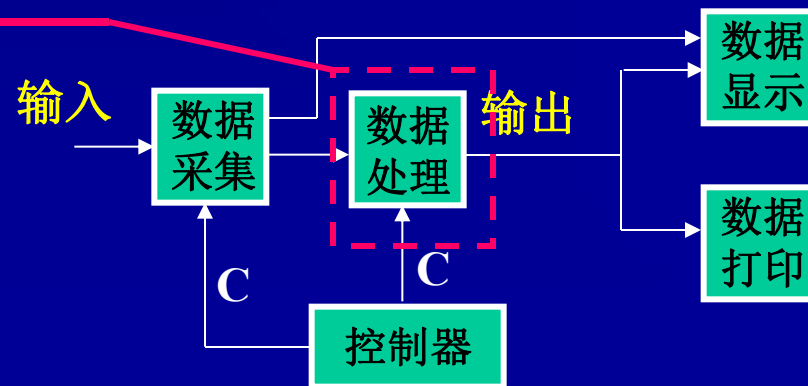
(b)

再分解

细化



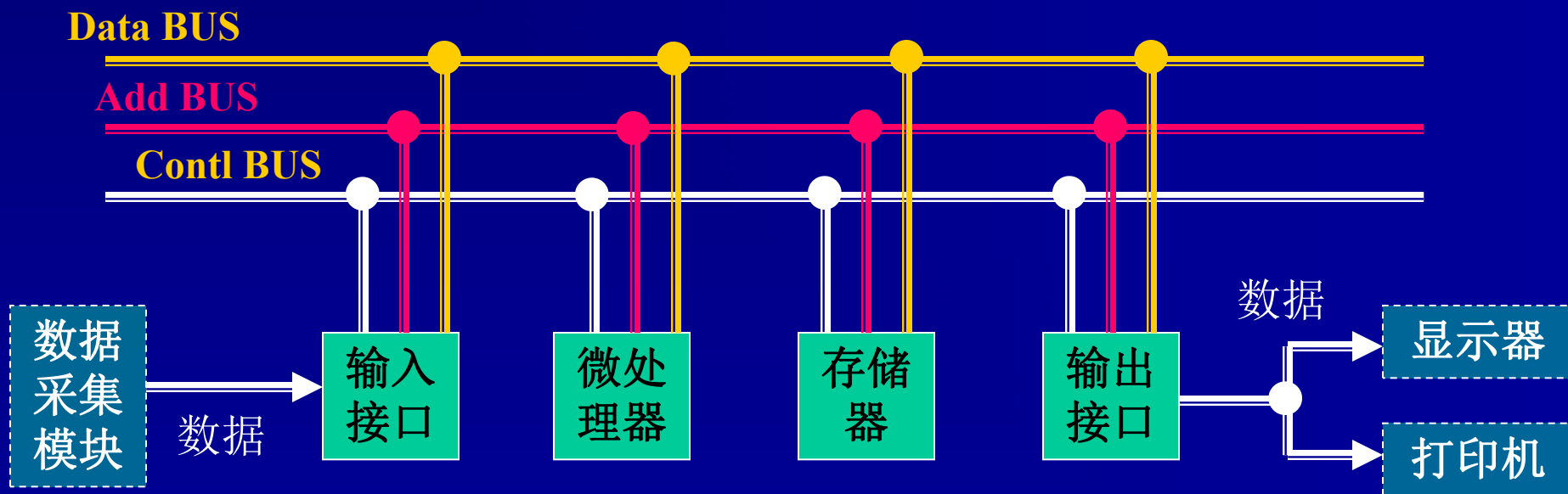
(d)



(c)

上例设计的这种结构框图，其任何一处的功能修改和扩充均涉及全局，而且总控制器的设计将十分复杂。

若采用总线结构，则该智能仪表的方框图如下：



两种结构哪种好？

看一个实例吧

5.1.3 数字系统设计的描述工具

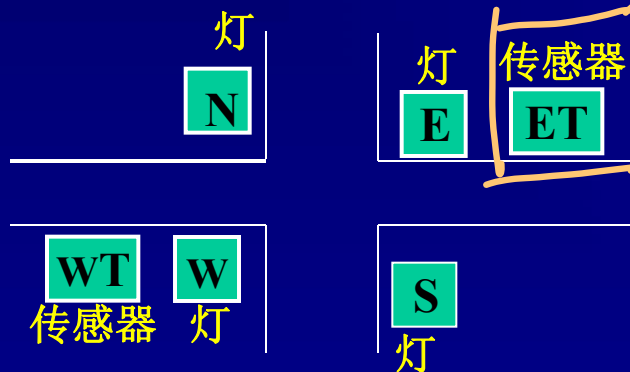
5.1.3.2 定时图 (时序图、时间关系图)

时序图的描述也是一个逐步细化的过程。从描述系统输入、输出之间的定时关系的简单时序图开始，随著系统设计的不断深入，时序图将不断地反映新出现的系统内部信号的时序关系，直到最终一个完整的时序图。

例 交通灯控制系统

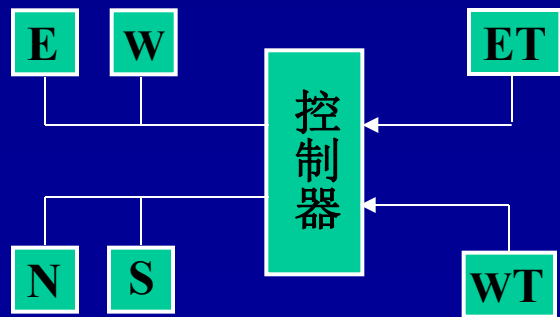
使其为数字系统的标志

关注各信号：电平/脉冲、同步/异步



(a) 示意图

交通工程



(b) 框图

N/S Red

N/S Yellow

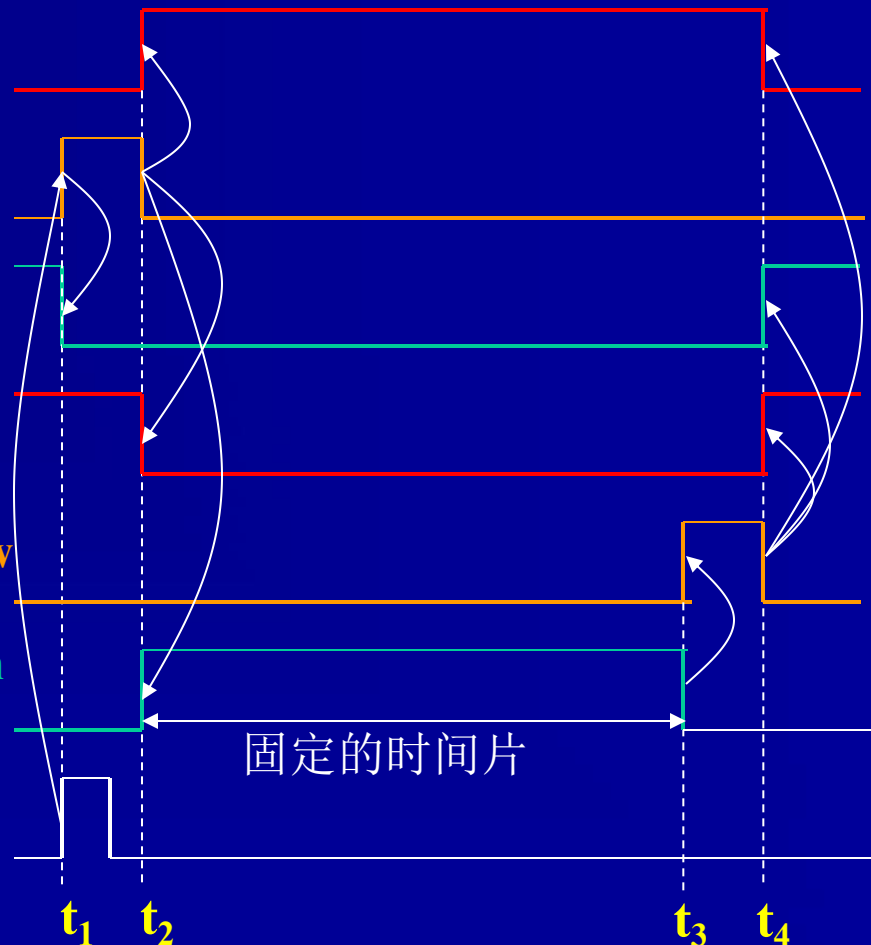
N/S Green

E/W Red

E/W Yellow

E/W Green

ET/WT



(c) 定时图

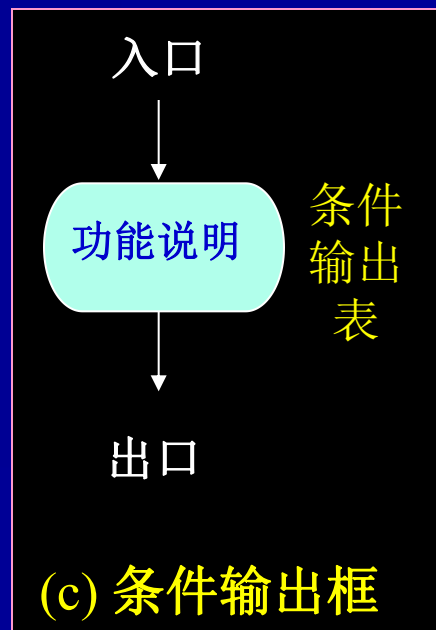
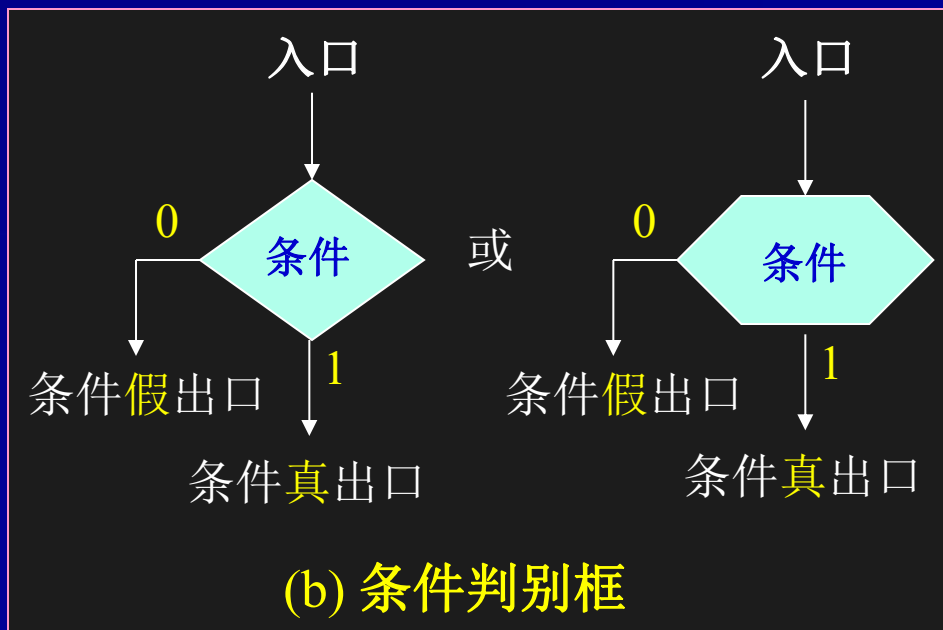
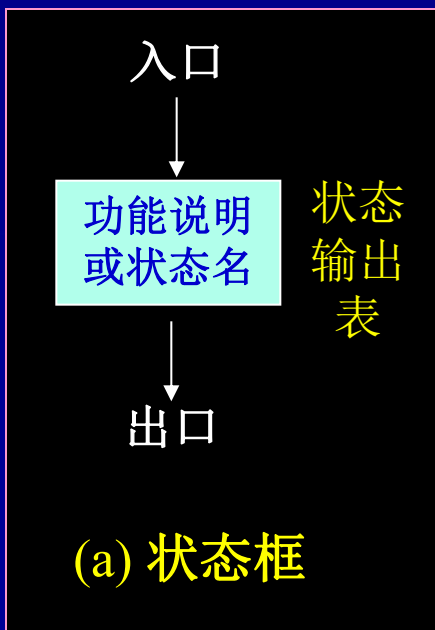
5.1.3.3 逻辑流程图

也称流程图，它用约定的几何图形(矩形、菱形、椭圆形等)、指向线和简练的文字说明，描述系统的基本工作过程。

逻辑流程图的描述对象是**控制单元**，并且以**系统时钟来驱动**整个流程。这一点与由事件驱动的软件流程图不同。

1、基本符号

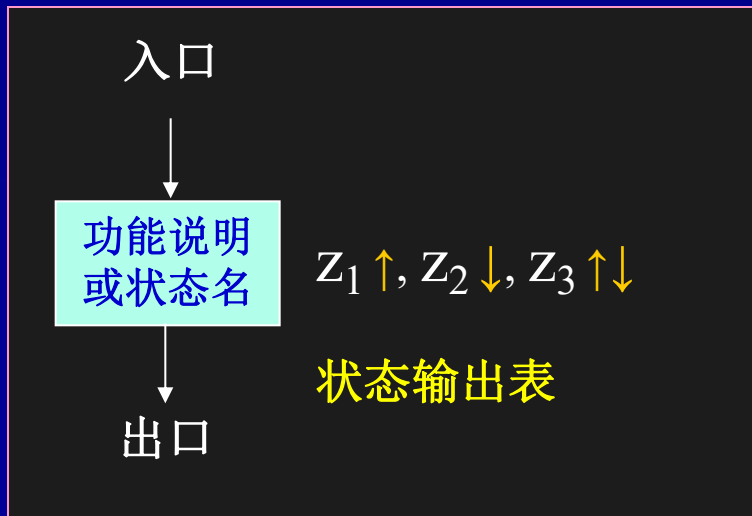
用三种符号：矩形状态框、菱形判别框、椭圆形条件框。



1、基本符号

状态框表示系统必须具备的状态，判别框及条件输出框不表示状态，只表示某状态框在不同的输入条件下的分支出口及条件输出，即用一个状态框及若干个判别框或条件输出框，组成一个状态单元。

如果在某状态下的输出与输入无关，即 **Moore 型** 输出，则该状态输出可标注在状态框旁的状态输出表中，且这个状态单元必定不包括条件输出框。



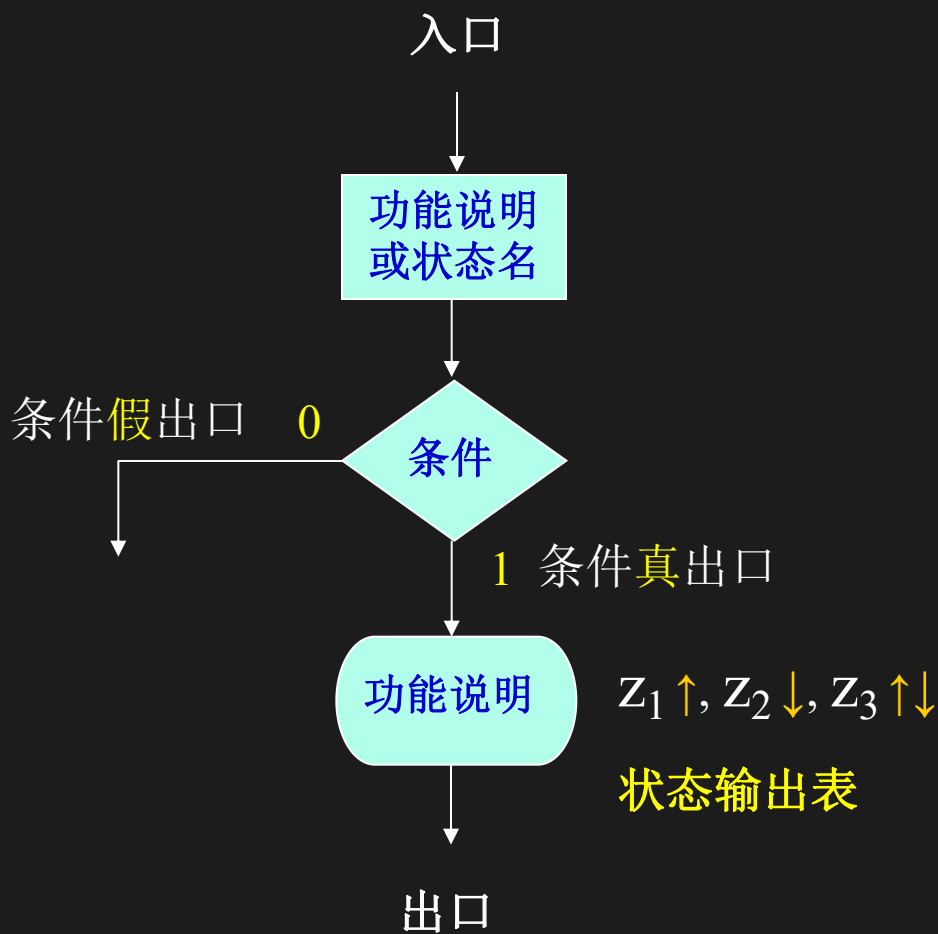
$z_1 \uparrow$ 表示进入状态state，输出 z_1 有效。

$z_2 \downarrow$ 表示进入状态state，输出 z_2 无效。

$z_3 \uparrow \downarrow$ 表示进入状态state，输出 z_3 有效，
并在退出状态state，输出 z_3 无效。

1、基本符号

如果在某状态下的输出与输入有关，即 **Mealy 型** 输出，则该状态输出要标注在菱形判别框下的椭圆形条件框旁的状态输出表中。



$z_1 \uparrow$ 表示进入状态， z_1 有效。
 $z_2 \downarrow$ 表示进入状态， z_2 无效。
 $z_3 \uparrow \downarrow$ 表示进入状态， z_3 有效，
并在退出状态， z_3 无效。

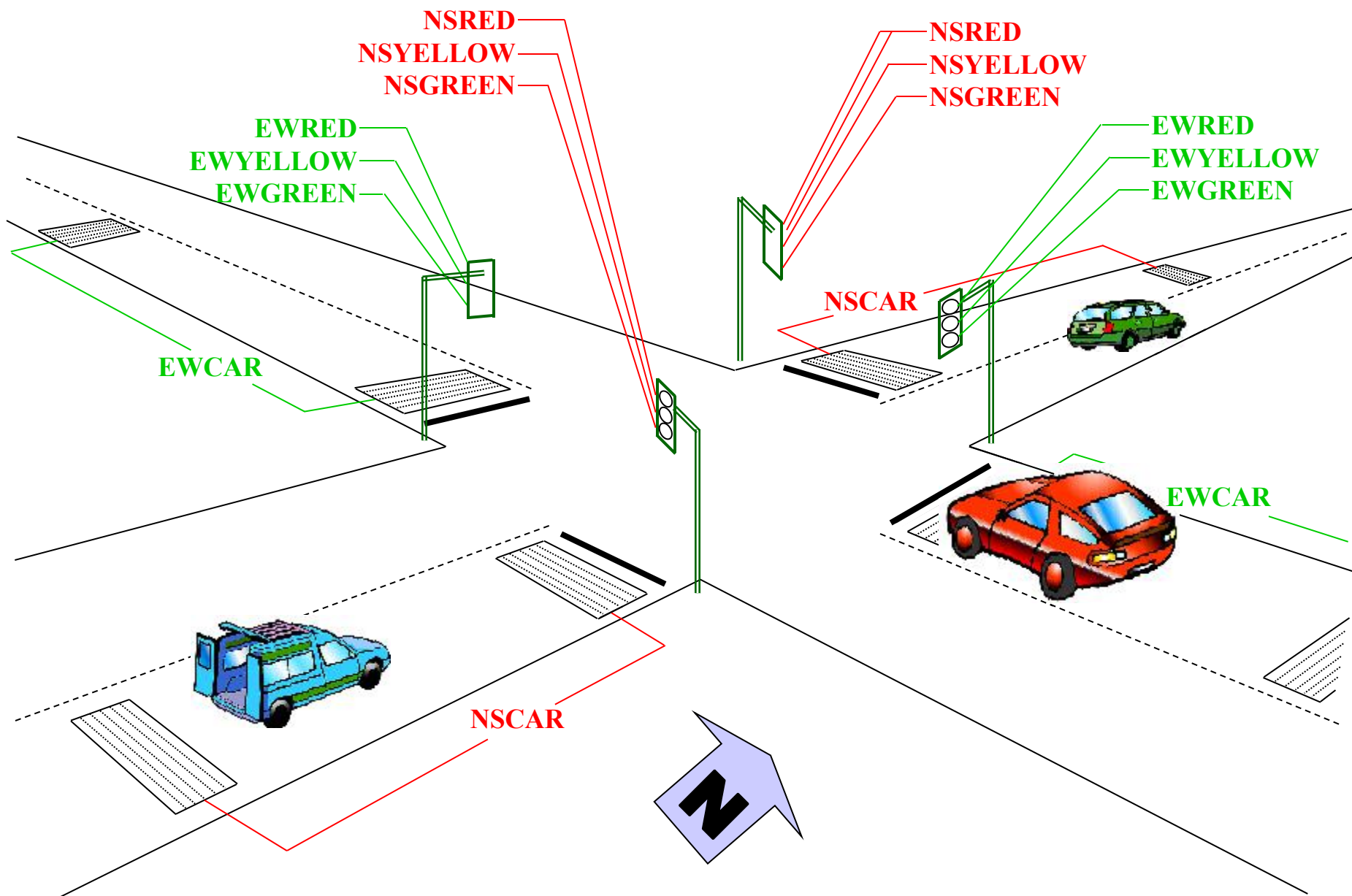
3、逻辑流程图的应用

逻辑流程图的描述过程是一个逐步细化（深化）的过程。它既便于设计者发现和改进信息处理过程中的错误，又是后续电路设计的依据。

例 设计一个十字路口交通灯控制系统。

这个例子来自美国加州美丽之城太阳谷Sunnyvale市，其十字路口的交通灯控制器经过了仔细设计，以使得汽车在十字路口的等待时间最小化。这个经市政部门认可的交通灯管理系统，后应用于芝加哥。

该系统使用了一个 1Hz 的时钟和三个计数器，以及 4组传感装置，下面是示意图。

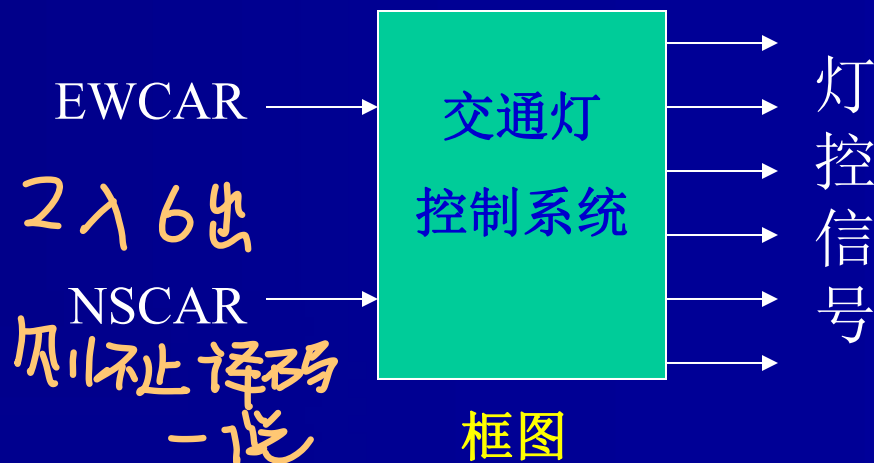


加洲太阳谷的一个十字路口的交通灯传感器和信号

例 设计一个十字路口交通灯控制系统。

设：东西道(EW)为主道，南北道(NS)为副道。

- 若 EW及NS均有车，
则 EW每次通行 60秒(绿灯)， NS每次通行40秒(绿灯)，
EW、NS轮流放行；
- 若 仅有一个通道有车，
则禁止无车通道(红灯)；
- 若 两通道均无车，
则NS禁止， EW 放行；
- 若 通道转换时，
两通道均需停车3秒(黄灯)。



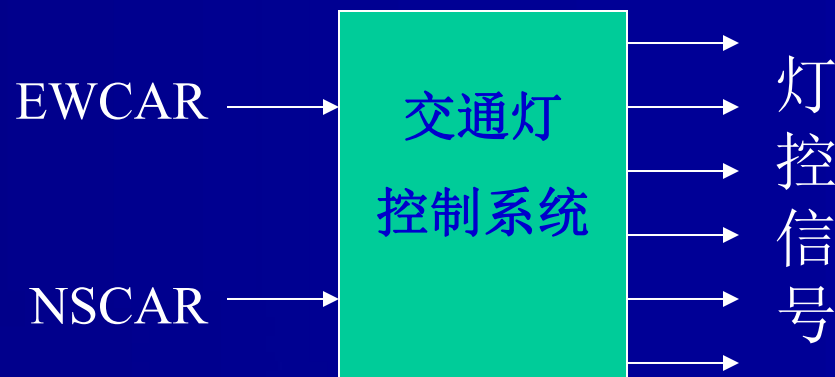
变量定义

监测器输出			输出灯光信号
NSCAR EWCAR (由 ET、ST WT、NT 生成)			NSRed NSGreen NSYellow EWRed EWGreen EWYellow

例 设计一个十字路口交通灯控制系统。

设：东西道(EW)为主道，南北道(NS)为副道。

- 若 EW及NS均有车，
则 EW每次通行 60秒(绿灯)， NS每次通行40秒(绿灯)，
EW、NS轮流放行；
- 若 仅有一个通道有车，
则禁止无车通道(红灯)；
- 若 两通道均无车，
则NS禁止， EW 放行；
- 若 通道转换时，
两通道均需停车3秒(黄灯)。



框图

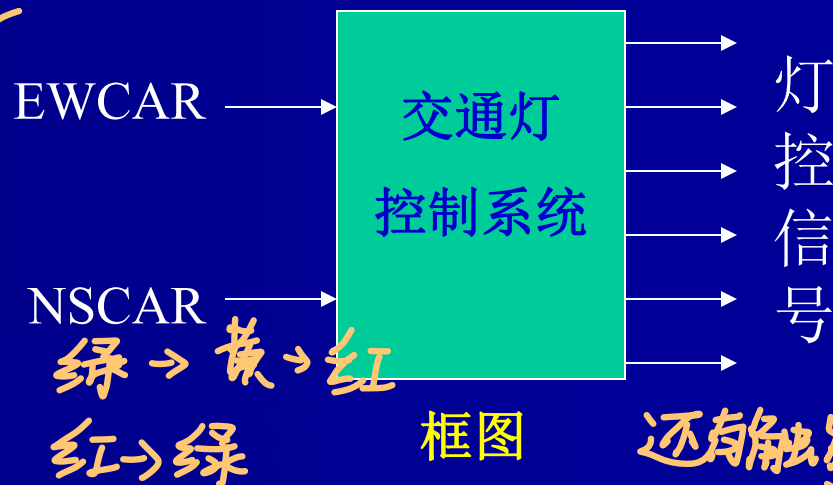
变量定义

监测器输出	定时器输出		输出灯光信号
NSCAR	TM60		NSRed
EWCAR	TM40		NSGreen
(由	TM3		NSYellow
ET、ST	(计数器的		EWRed
WT、NT	进位输出)		EWGreen
生成)			EWYellow

例 设计一个十字路口交通灯控制系统。

设：东西道(EW)为主道，南北道(NS)为副道。

- 若 EW及NS均有车，
则 EW每次通行 60秒(绿灯)， NS每次通行40秒(绿灯)，
EW、NS轮流放行；
- 若 仅有一个通道有车，
则禁止无车通道(红灯)；
- 若 两通道均无车，
则NS禁止， EW 放行；
- 若 通道转换时，
两通道均需停车3秒(黄灯)。

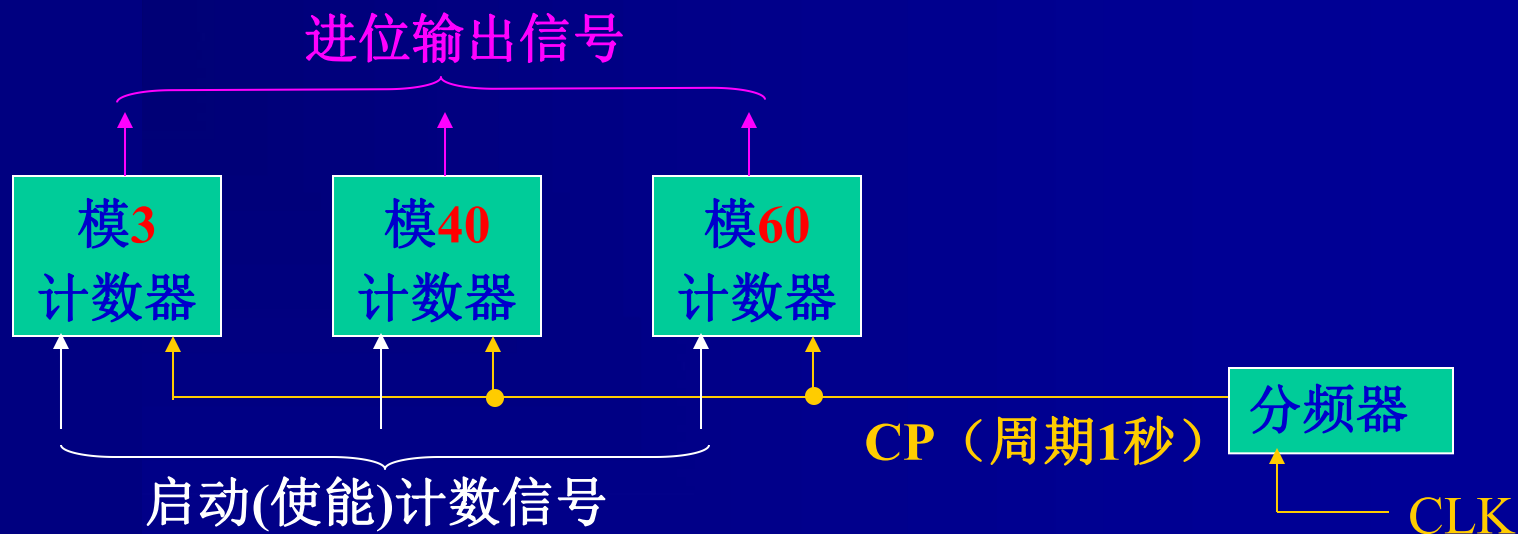


变量定义

监测器输出	定时器输出	定时器使能输入	输出灯光信号
NSCAR	TM60	ENTM60	NSRed
EWCAR	TM40	ENTM40	NSGreen
(由	TM3	ENTM3	NSYellow
ET、ST	(计数器的	使能3个计数	EWRed
WT、NT	进位输出)		EWGreen
生成)	计数满输出		EWYellow

时序电路
要个

系统配有三个3秒、40秒和60秒的定时器，如下：



定时器在系统中起到时间节拍指挥的作用，各部件均按照统一的时间节拍协调地工作。

a. 粗框图



- 若 **EW** 及 **NS** 均有车，则 **EW** 每次通行 **60秒(绿灯)**，**NS** 每次通行 **40秒(绿灯)**，**EW**、**NS** 轮流放行；


- 若 仅有一个通道有车，则禁止无车通道(**红灯**)；

- 若 两通道均无车，则 **NS** 禁止，**EW** 放行；

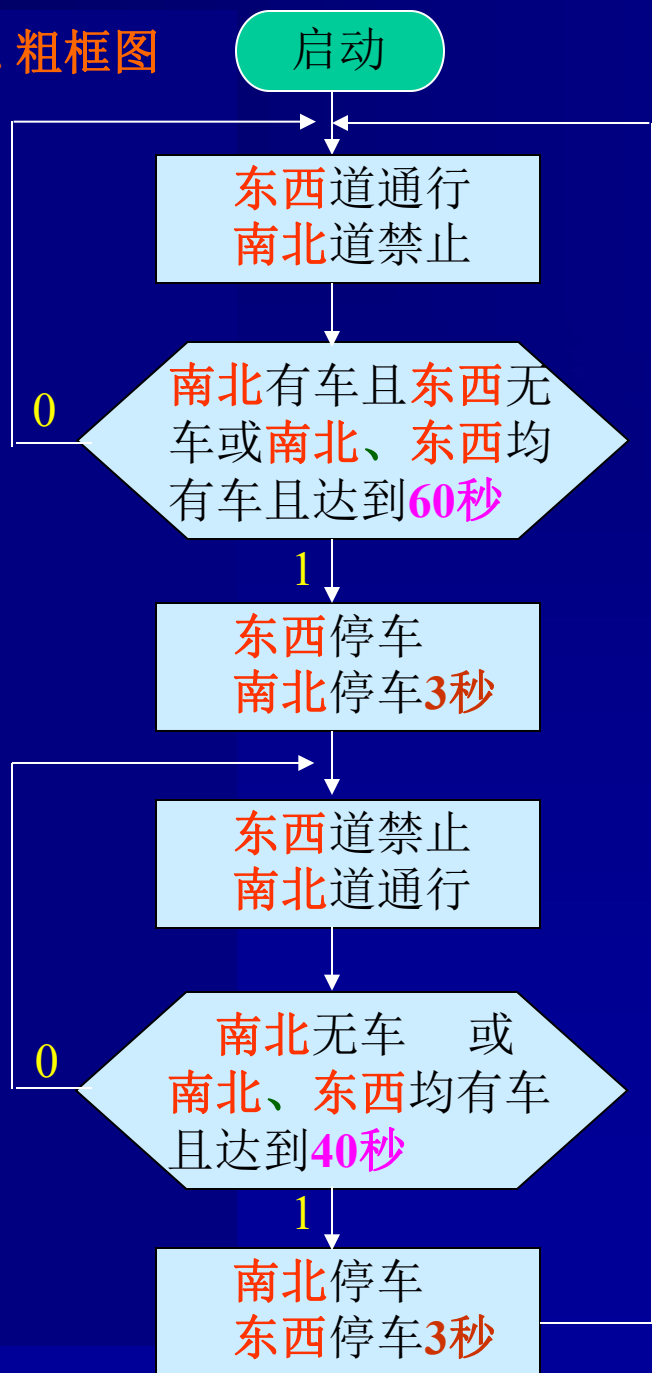
- 若 通道转换时，两通道均需停车 **3秒(黄灯)**。

未来会再见

Mealy or Moore?

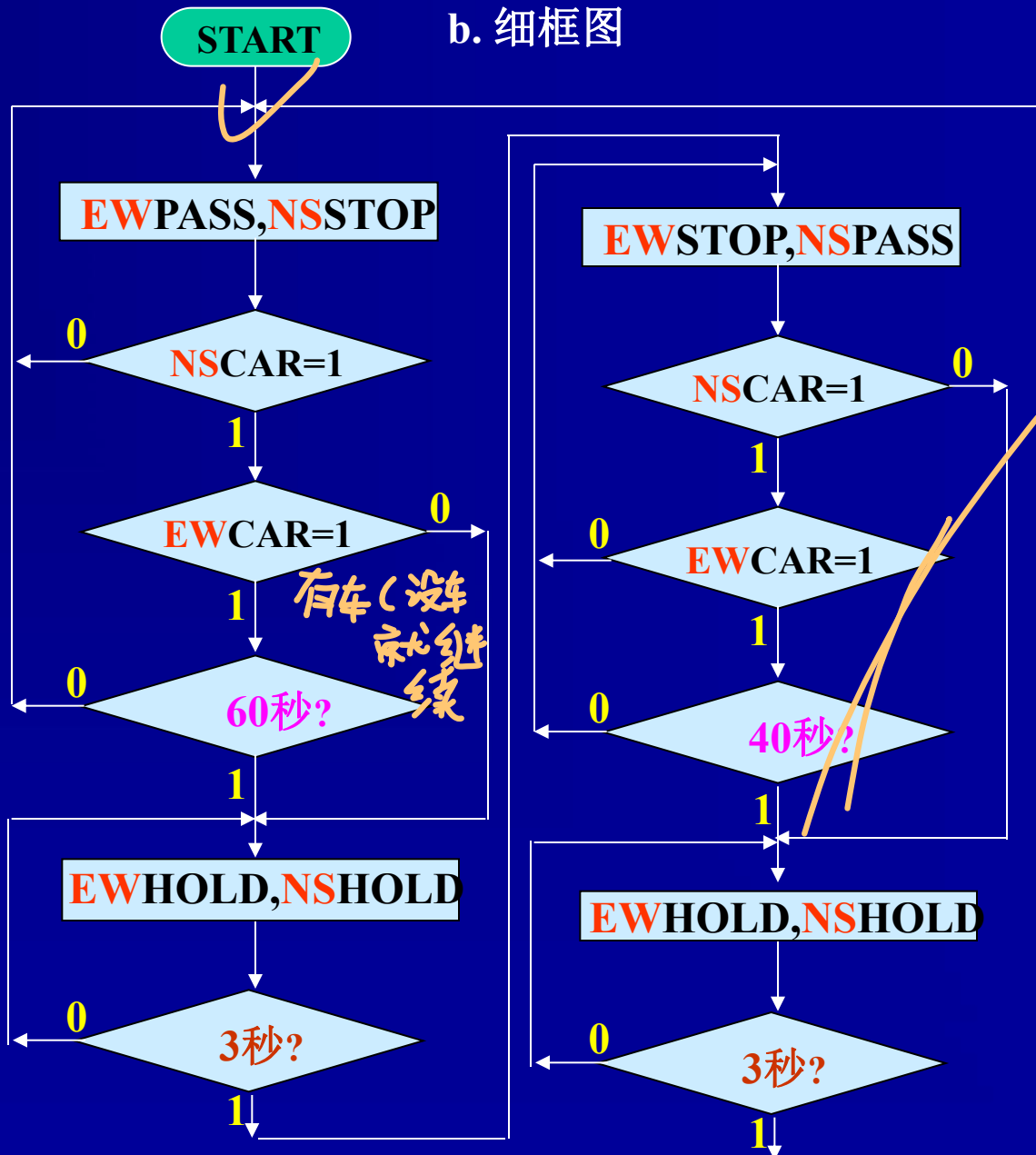


a. 粗框图



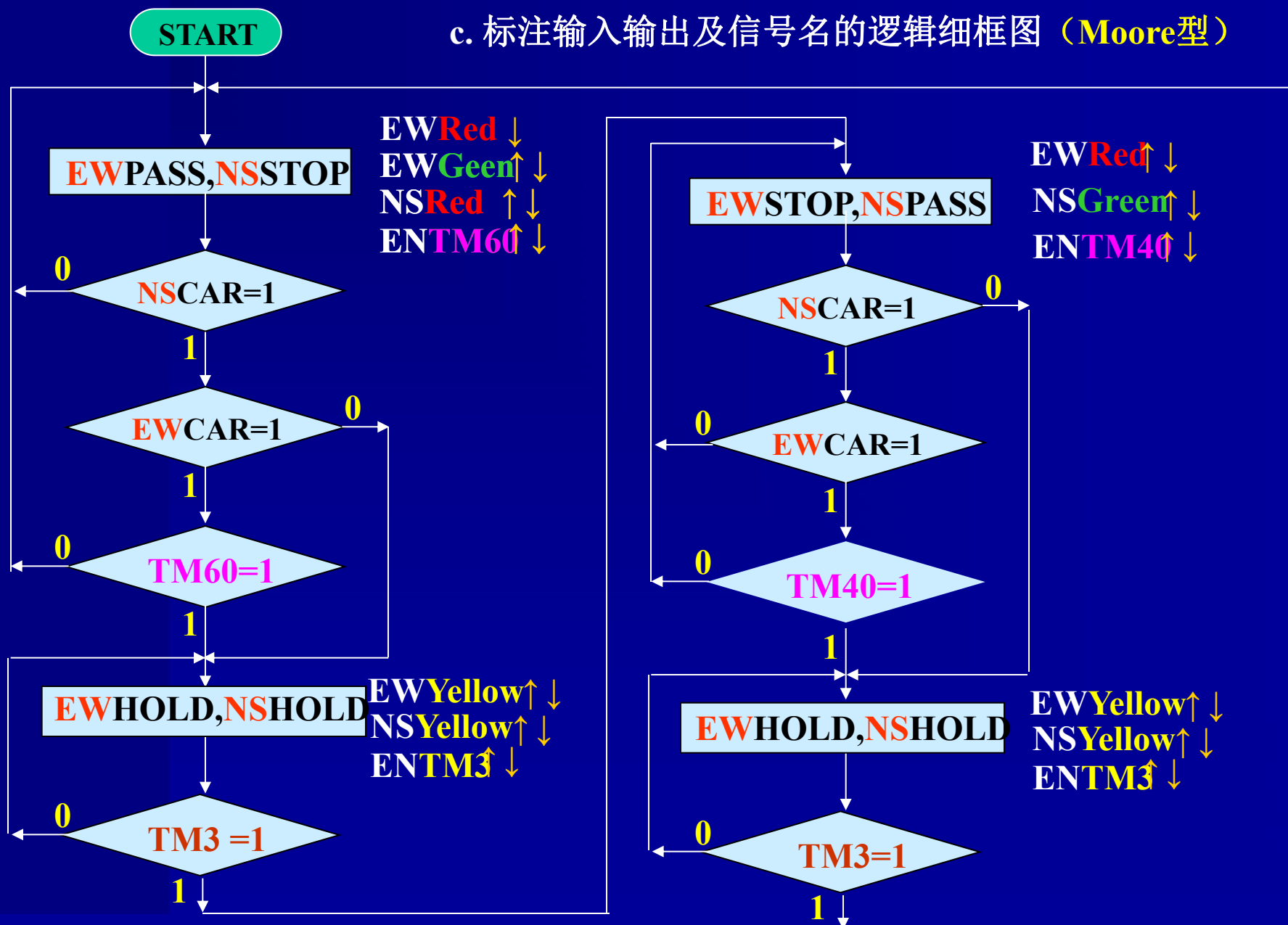
十字路口交通灯控制器逻辑框图

b. 细框图



十字路口交通灯控制器逻辑流程图

c. 标注输入输出及信号名的逻辑细框图 (Moore型)

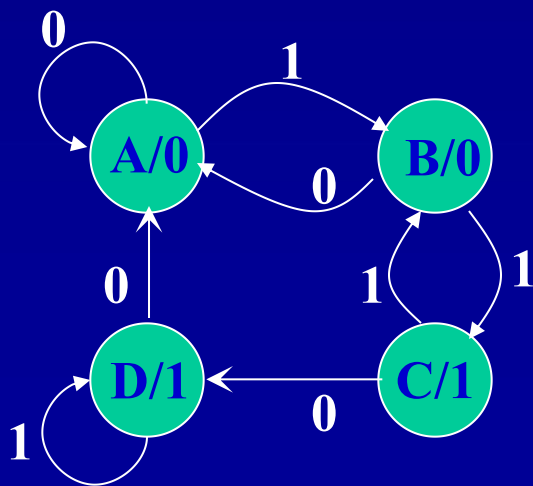


3、从状态图得到逻辑流程图

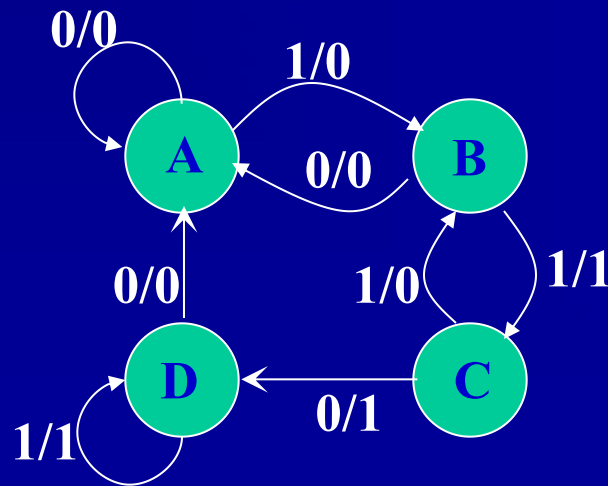
逻辑流程图上的一个状态框及若干个判别框或条件输出框所组成一个状态单元**对应了状态图**上的一个状态和它的输入输出。

如果某状态的输出与输入有关(**Mealy型**)，则逻辑流程图中对应的状态单元必定包括**有条件输出框**；

如果某状态的输出与输入无关(**Moore型**)，则逻辑流程图中对应的状态单元必定**没有条件输出框**。



Moore 型状态图



Mealy 型状态图

5.1.3.4 ASM图

逻辑流程图是数字系统中使用得最广泛的一种非形式化的描述工具，但它的规范性不够。经过不断改进，将流程图改造成描述数字系统硬件的形式化工具

——**算法状态机图ASM**

(*Algorithmic State Machine Chart*)。

在时序电路的ASM图中，每一个状态由一个ASM块来表示。

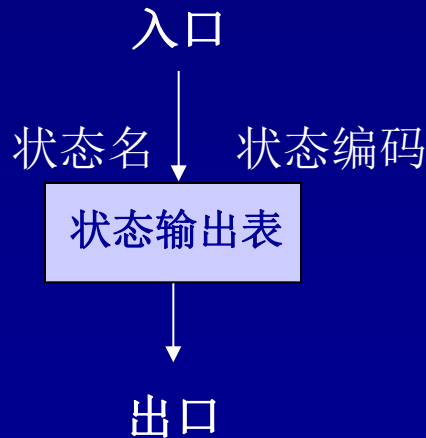
人·流程图&关系

1、基本符号

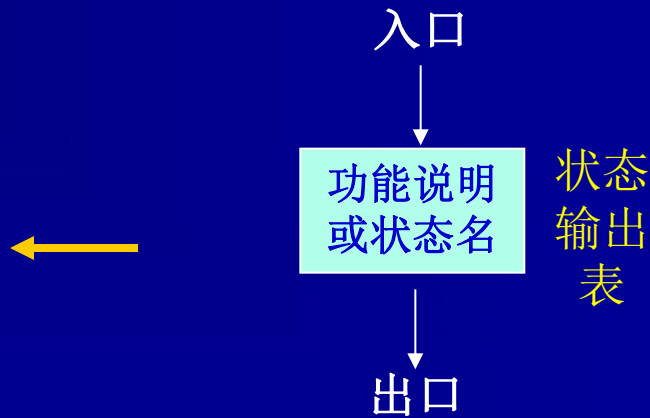
一个ASM块至多包含三种不同类型的符号：

矩形状态框、**菱形判别框**和**条件输出框**。

注意ASM图的标注（左边）与流程图的（右边）不同。



(a) **ASM**的状态框



(a) 流程图的状态框

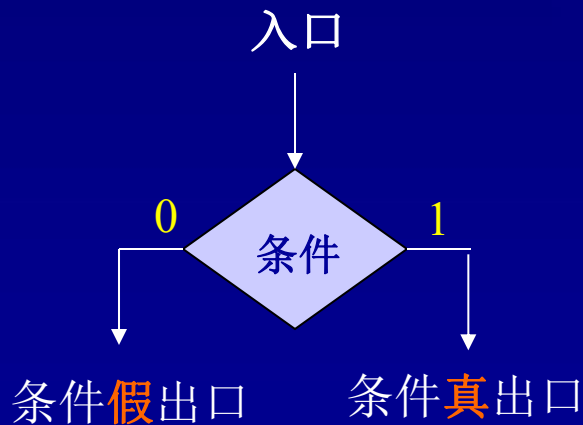
其中：状态编码为该状态下的触发器值。

1、基本符号

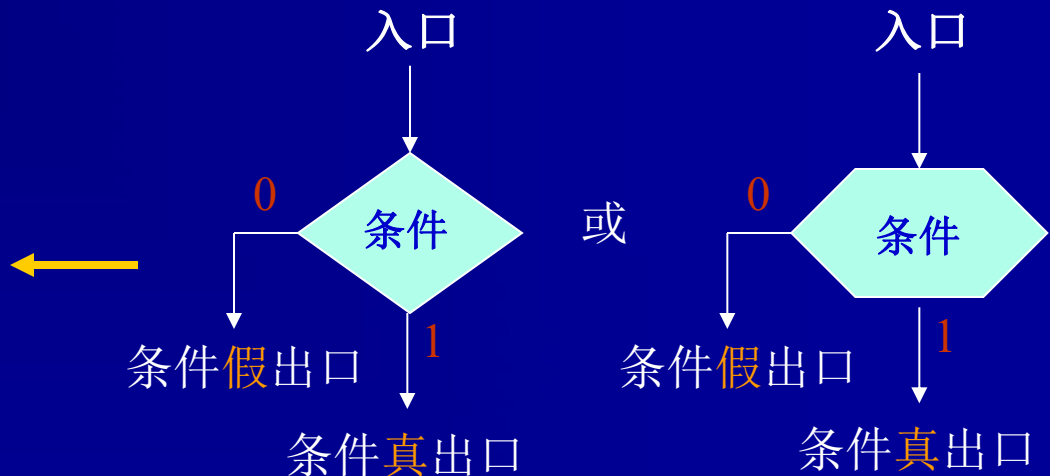
一个ASM块至多包含三种不同类型的符号：

矩形状态框、**菱形判别框**和**条件输出框**。

注意ASM图的标注与流程图的不同。



(b) ASM的判别框



(b) 流程图的条件判别框

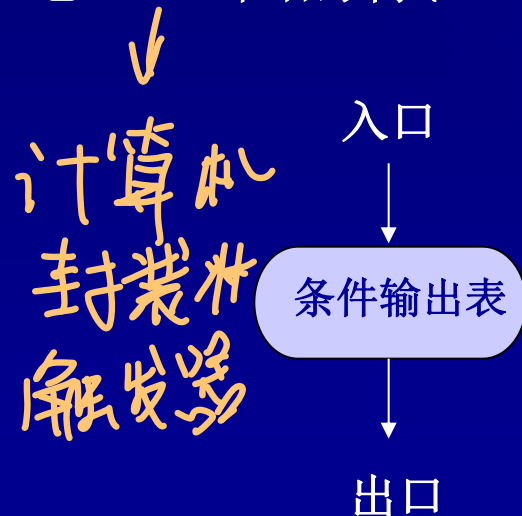
其中：框内的输入条件为**布尔表达式**。

1、基本符号

一个ASM块至多包含三种不同类型的符号：

矩形状态框、菱形判别框和条件输出框。

注意ASM图的标注与流程图的不同。



(c) 条件输出框



(c) 条件输出框

条件输出框的输入总是来自判别框，由这个判别框给出了输出所需要的条件，即在某一状态下，某个输出变量是输入变量的函数，就在条件输出框中填入**条件满足时产生的输出**。

2、由逻辑流程图转换成ASM图

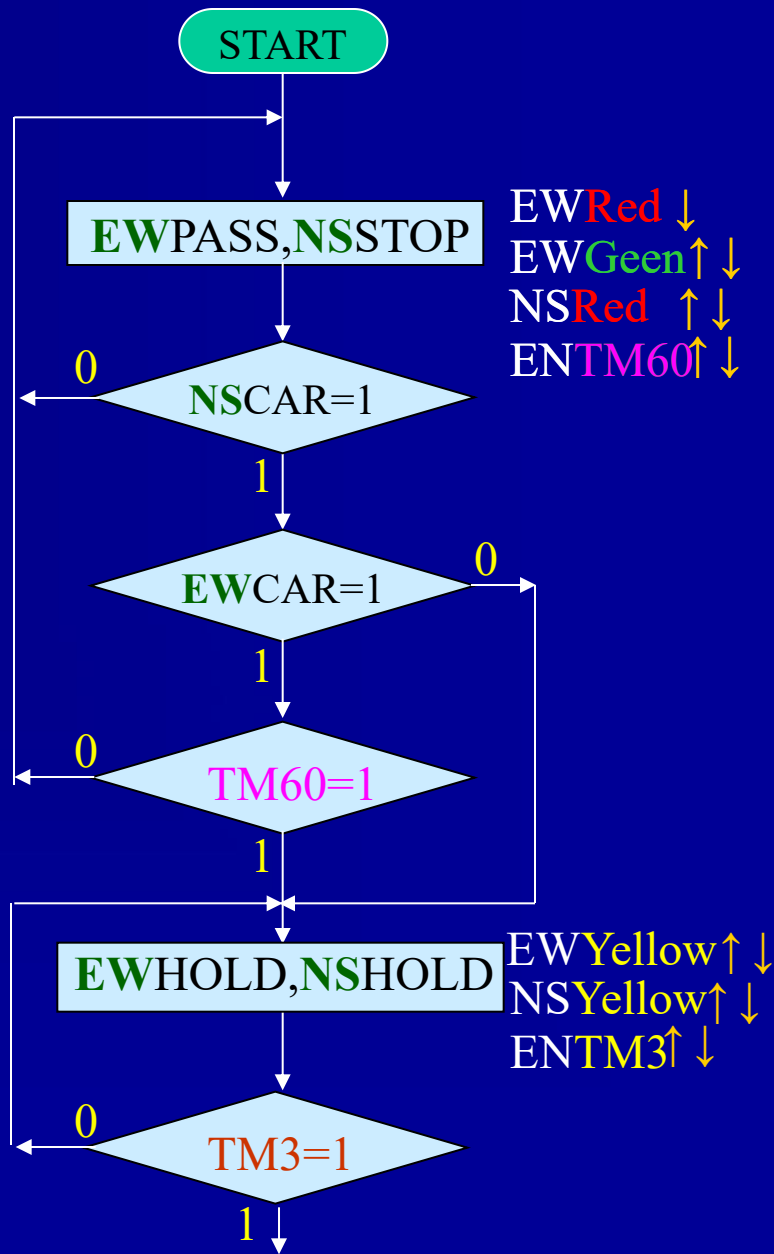
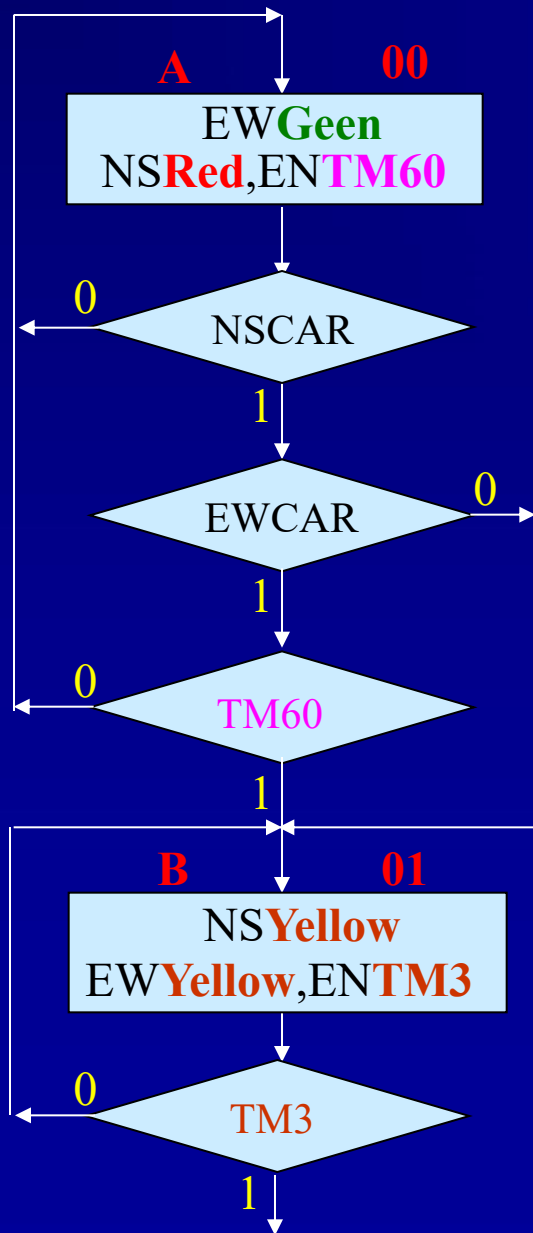
ASM图可以直接送入计算机辅助逻辑设计系统，由该系统自动完成控制单元的设计。

逻辑流程图可以很容易地转换成ASM图。

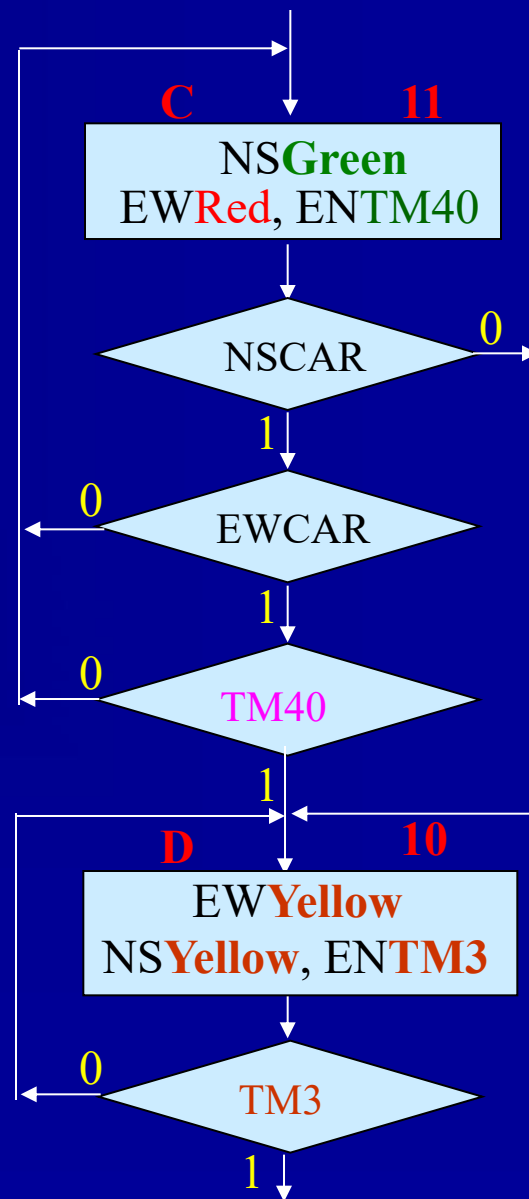
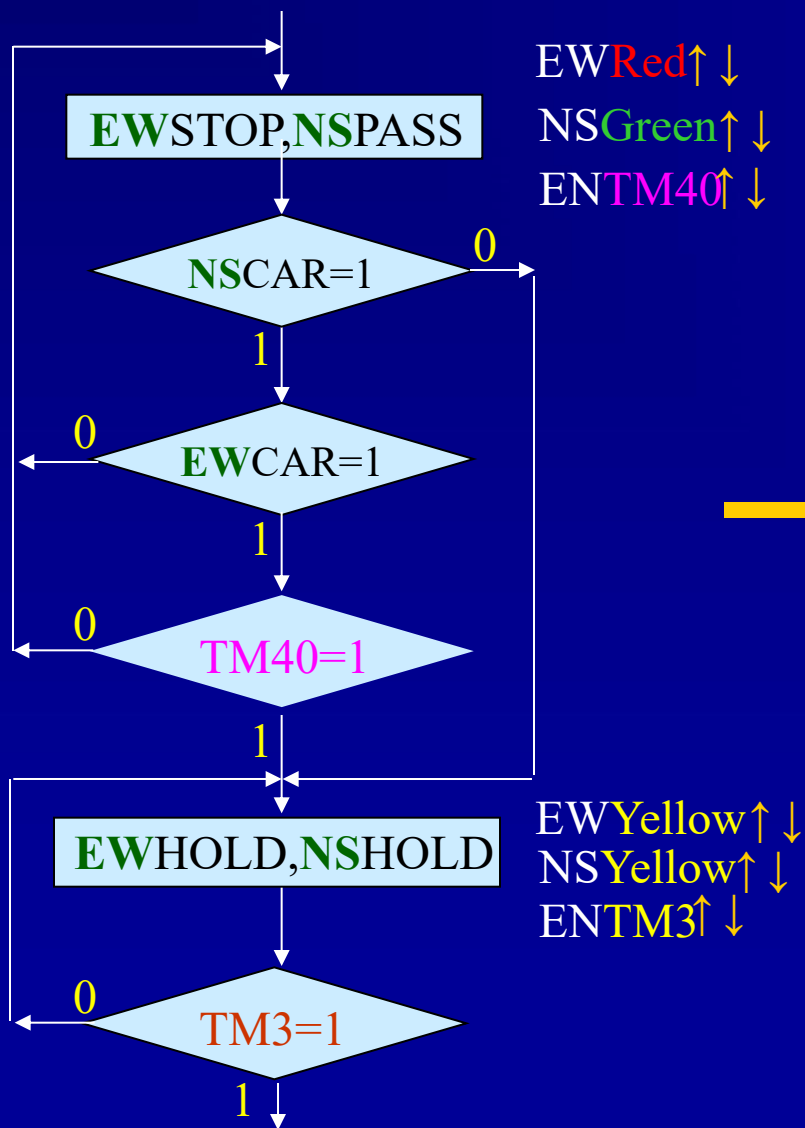
在ASM图中，所有的当前状态下的**Moore型**有效输出都应列在状态框内；

所有的当前状态下的满足输入条件的**Mealy型**有效输出都应列在条件输出框内。

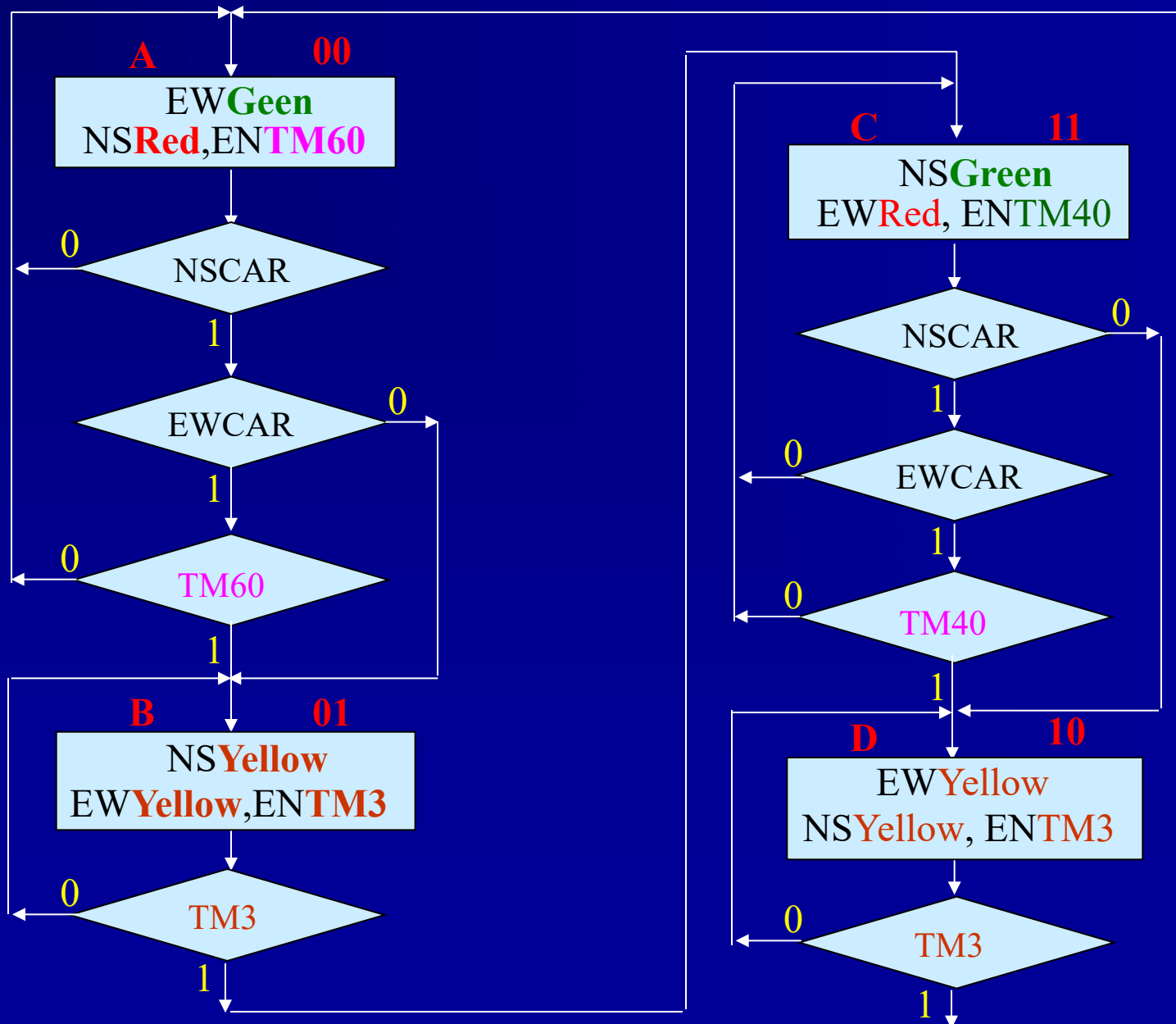
十字路口交通灯控制器ASM图



十字路口交通灯控制器ASM图



十字路口交通灯控制器ASM图



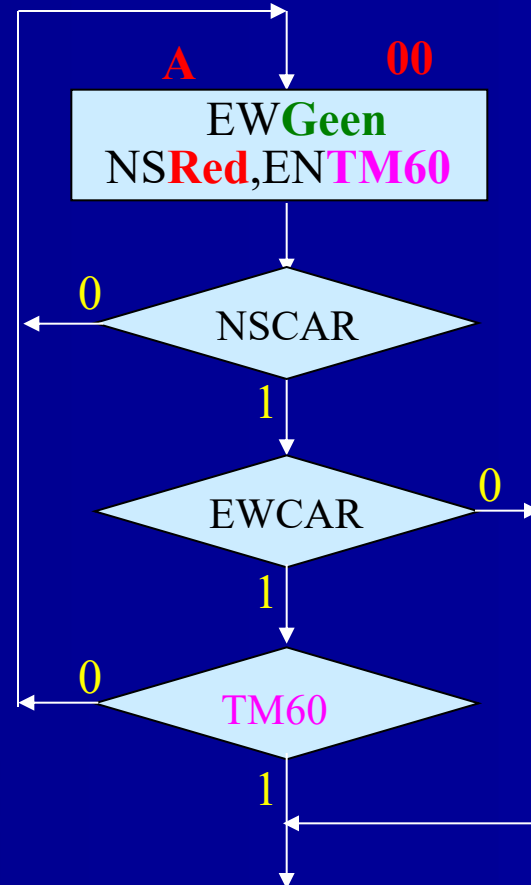
3、应用ASM图进行数字系统中控制器的设计

主要步骤：

- (1) 按设计要求写出问题说明。
- (2) 定义输入、输出信号并用助记符表示。
- (3) 将问题说明转换成详细**逻辑流程图**。
- (4) 将详细逻辑流程图转换成**ASM图**。
- (5) 从ASM图得到**状态转换表**。
- (6) 由状态转换表得到**次态方程式**，从ASM图列出**输出函数表达式**。
- (7) 按照次态方程式及输出函数表达式，画出控制器**逻辑电路图**。

4、状态转换表达式表

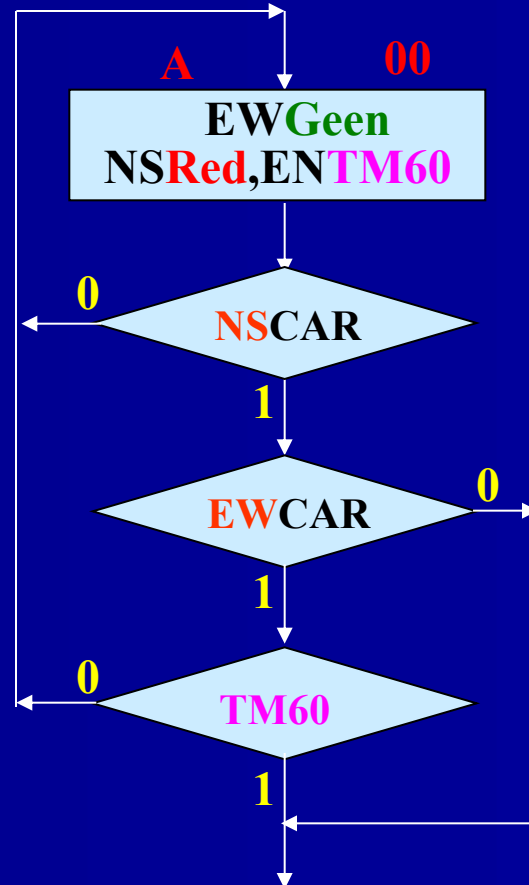
按照ASM图中的每一个ASM块中判别框和条件输出框内标注的有效输入，列出状态转换表达式表，如下：



现态	输入条件			状态转换表达式	次态
S(A)	NSCAR	EWCAR	TM60		S ⁿ⁺¹
$\overline{Q_1}\overline{Q_0}$					
$\overline{Q_1}Q_0$					
$Q_1\overline{Q_0}$					
Q_1Q_0					

4、状态转换表达式表

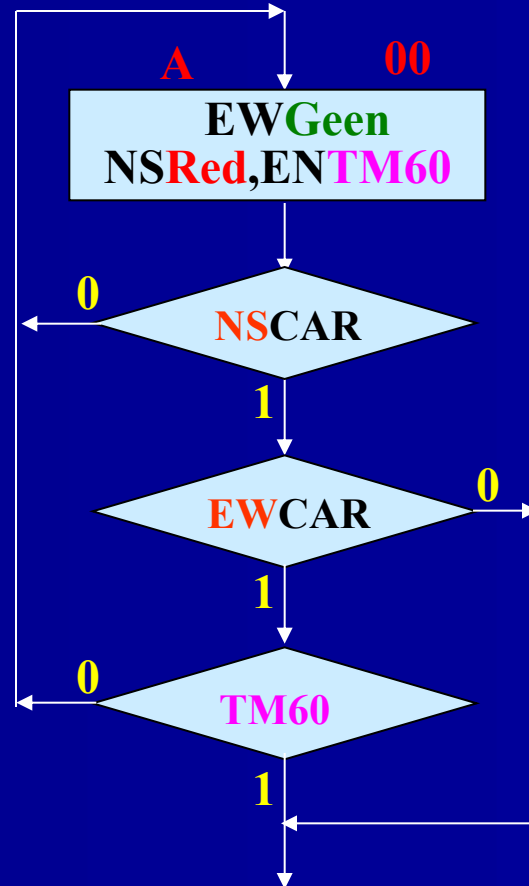
按照ASM图中的每一个ASM块中判别框和条件输出框内标注的有效输入，列出状态转换表达式表，如下：



现态	输入条件			状态转换表达式	次态
S(A)	NSCAR	EWCAR	TM60		S ⁿ⁺¹
$\overline{Q_1}\overline{Q_0}$	0				A
$\overline{Q_1}Q_0$					
$Q_1\overline{Q_0}$					
Q_1Q_0					

4、状态转换表达式表

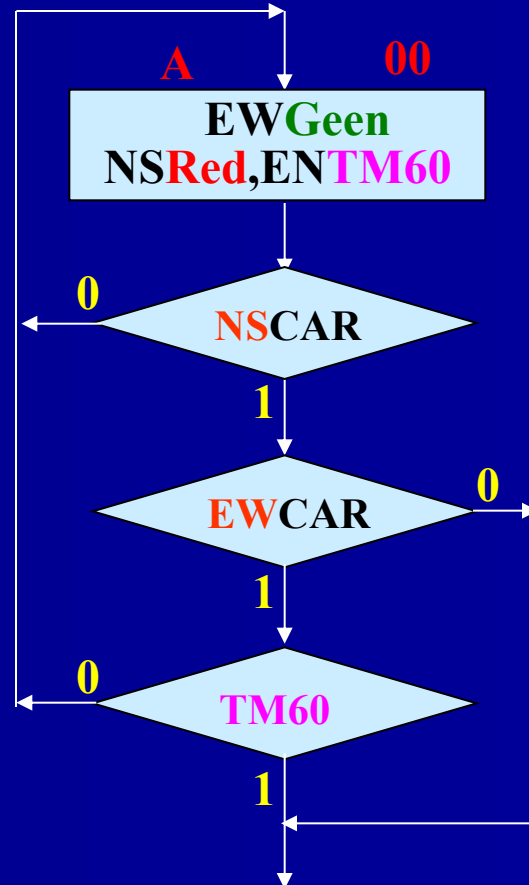
按照ASM图中的每一个ASM块中判别框和条件输出框内标注的有效输入，列出状态转换表达式表，如下：



现态	输入条件			状态转换表达式	次态
S(A)	NSCAR	EWCAR	TM60		S ⁿ⁺¹
$\overline{Q_1}\overline{Q_0}$	0	—	—		A
$\overline{Q_1}Q_0$					
$Q_1\overline{Q_0}$					
Q_1Q_0					

4、状态转换表达式表

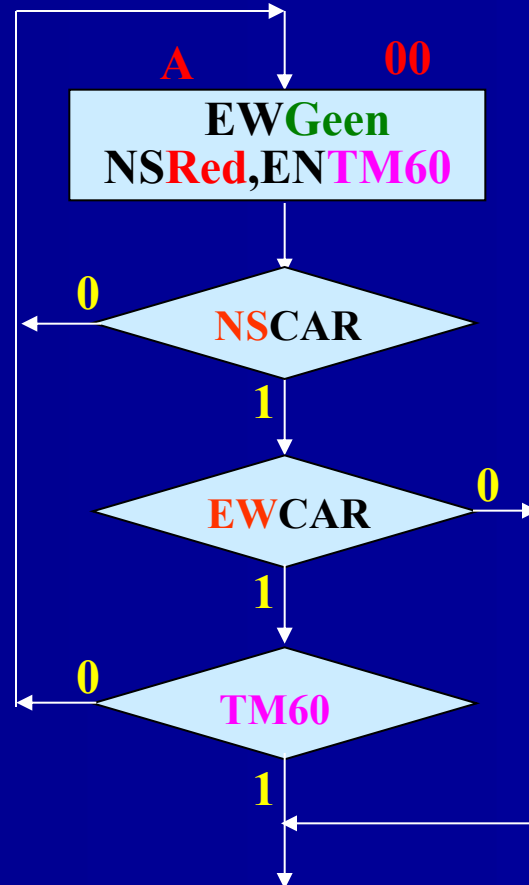
按照ASM图中的每一个ASM块中判别框和条件输出框内标注的有效输入，列出状态转换表达式表，如下：



现态	输入条件			状态转换表达式	次态
S(A)	NSCAR	EWCAR	TM60		S ⁿ⁺¹
$\overline{Q_1}\overline{Q_0}$	0	—	—	$\overline{\text{NSCAR}}$	A
$\overline{Q_1}Q_0$					
$Q_1\overline{Q_0}$					
Q_1Q_0					

4、状态转换表达式表

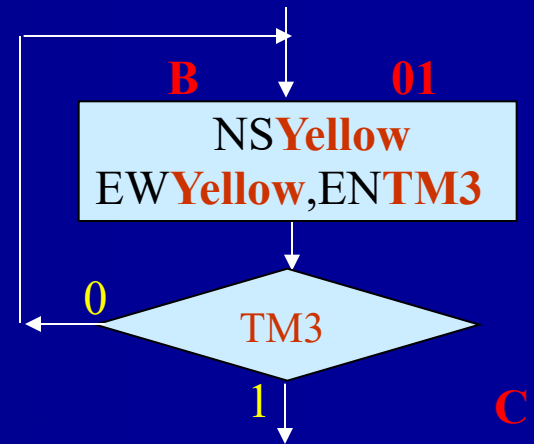
按照ASM图中的每一个ASM块中判别框和条件输出框内标注的有效输入，列出状态转换表达式表，如下：



现态	输入条件			状态转换表达式	次态
S(A)	NSCAR	EWCAR	TM60		S ⁿ⁺¹
$\overline{Q_1}\overline{Q_0}$	0	—	—	$\overline{\text{NSCAR}}$	A
$\overline{Q_1}\overline{Q_0}$	1	0	—	$\text{NSCAR} \cdot \overline{\text{EWCAR}}$	B
$\overline{Q_1}\overline{Q_0}$	1	1	0	$\text{NSCAR} \cdot \text{EWCAR} \cdot \overline{\text{TM60}}$	A
$\overline{Q_1}\overline{Q_0}$	1	1	1	$\text{NSCAR} \cdot \text{EWCAR} \cdot \text{TM60}$	B

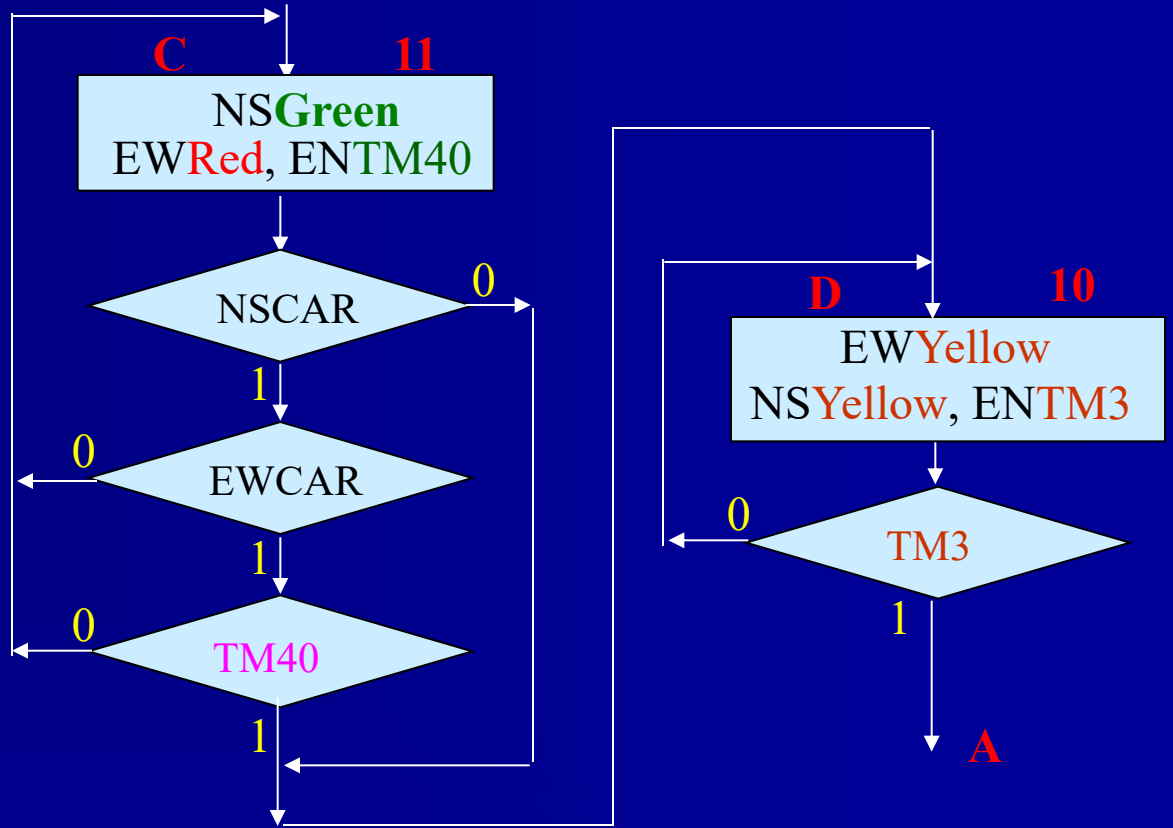
4、状态转换表达式表

按照ASM图中的每一个ASM块中判别框和条件输出框内标注的有效输入，列出状态转换表达式表，如下：



现态	输 入 条 件			状态转换表达式	次态
S(B)			TM 3		S^{n+1}
$\overline{Q_1}Q_0$			0	$\overline{TM\ 3}$	B
$\overline{Q_1}Q_0$			1	TM 3	C

4、状态转换表达式表



现态	输 入 条 件				状态转换表达式	次态
S	NSCAR	EWCAR	TM40	TM3		S ⁿ⁺¹
$Q_1\overline{Q_0}$	0	—	—	—	$\overline{\text{NSCAR}}$	D
$Q_1\overline{Q_0}$	1	0	—	—	$\text{NSCAR} \cdot \overline{\text{EWCAR}}$	C
$Q_1\overline{Q_0}$	1	1	0	—	$\text{NSCAR} \cdot \text{EWCAR} \cdot \overline{\text{TM60}}$	C
$Q_1\overline{Q_0}$	1	1	1	—	$\text{NSCAR} \cdot \text{EWCAR} \cdot \text{TM60}$	D
Q_1Q_0	—	—	—	0	$\overline{\text{TM3}}$	D
Q_1Q_0	—	—	—	1	TM3	A

5、状态转换表

由状态转换表达式表合并成状态转换表，如下：

现态	输 入 条 件			状态转换表达式	次态
S(A)	NSCAR	EWCAR	TM60		S^{n+1}
$\overline{Q_1}\overline{Q_0}$	0	—	—	$\overline{\text{NSCAR}}$	A
$\overline{Q_1}Q_0$	1	0	—	$\text{NSCAR} \cdot \overline{\text{EWCAR}}$	B
$Q_1\overline{Q_0}$	1	1	0	$\text{NSCAR} \cdot \text{EWCAR} \cdot \overline{\text{TM60}}$	A
Q_1Q_0	1	1	1	$\text{NSCAR} \cdot \text{EWCAR} \cdot \text{TM60}$	B



S	Q_1	Q_0	转换条件表达式	S^{n+1}	Q_1^{n+1}	Q_0^{n+1}
A	0	0	$\overline{\text{NSCAR}}$	A	0	0
A	0	0	$\text{NSCAR} \cdot \overline{\text{EWCAR}}$	B	0	1
A	0	0	$\text{NSCAR} \cdot \text{EWCAR} \cdot \overline{\text{TM60}}$	A	0	0
A	0	0	$\text{NSCAR} \cdot \text{EWCAR} \cdot \text{TM60}$	B	0	1

十字路口交通灯控制器状态转换表

S	Q ₁	Q ₀	转换条件表达式	S ⁿ⁺¹	Q ₁ ⁿ⁺¹	Q ₀ ⁿ⁺¹
A	0	0	$\overline{\text{NSCAR}}$	A	0	0
A	0	0	$\text{NSCAR} \cdot \overline{\text{EWCAR}}$	B	0	1
A	0	0	$\text{NSCAR} \cdot \text{EWCAR} \cdot \overline{\text{TM60}}$	A	0	0
A	0	0	$\text{NSCAR} \cdot \text{EWCAR} \cdot \text{TM60}$	B	0	1
B	0	1	$\overline{\text{TM3}}$	B	0	1
B	0	1	TM3	C	1	1
C	1	1	NSCAR	D	1	0
C	1	1	$\text{NSCAR} \cdot \overline{\text{EWCAR}}$	C	1	1
C	1	1	$\text{NSCAR} \cdot \text{EWCAR} \cdot \overline{\text{TM40}}$	C	1	1
C	1	1	$\text{NSCAR} \cdot \text{EWCAR} \cdot \text{TM40}$	D	1	0
D	1	0	$\overline{\text{TM3}}$	D	1	0
D	1	0	TM3	A	0	0

十字路口交通灯控制器次态方程式

由于次态为 $S^{n+1} = S \cdot (\text{转换条件表达式})$ ，由前表可得到次态方程式并化简为：

$$\begin{aligned} Q_1^{n+1} &= \overline{Q_1} \cdot Q_0 \cdot \text{TM3} + Q_1 \cdot Q_0 \cdot \overline{\text{NSCAR}} + Q_1 \cdot Q_0 \cdot \text{NSCAR} \cdot \overline{\text{EWCAR}} \\ &\quad + Q_1 \cdot Q_0 \cdot \text{NSCAR} \cdot \text{EWCAR} \cdot \overline{\text{TM40}} \\ &\quad + Q_1 \cdot Q_0 \cdot \text{NSCAR} \cdot \text{EWCAR} \cdot \text{TM40} + Q_1 \cdot \overline{Q_0} \cdot \overline{\text{TM3}} \\ &= \overline{Q_1} \cdot Q_0 \cdot \text{TM3} + Q_1 \cdot Q_0 \cdot \overline{\text{NSCAR}} + Q_1 \cdot Q_0 \cdot \text{NSCAR} \cdot \overline{\text{EWCAR}} \\ &\quad + Q_1 \cdot Q_0 \cdot \text{NSCAR} \cdot \text{EWCAR} + Q_1 \cdot \overline{Q_0} \cdot \overline{\text{TM3}} \\ &= \overline{Q_1} \cdot Q_0 \cdot \text{TM3} + Q_1 \cdot Q_0 \cdot \overline{\text{NSCAR}} + Q_1 \cdot Q_0 \cdot \text{NSCAR} + Q_1 \cdot \overline{Q_0} \cdot \overline{\text{TM3}} \\ &= \overline{Q_1} \cdot Q_0 \cdot \text{TM3} + Q_1 \cdot Q_0 + Q_1 \cdot \overline{Q_0} \cdot \overline{\text{TM3}} \\ &= \overline{Q_1} \cdot Q_0 \cdot \text{TM3} + Q_1 \cdot Q_0 \cdot \text{TM3} + Q_1 \cdot Q_0 \cdot \overline{\text{TM3}} + Q_1 \cdot \overline{Q_0} \cdot \overline{\text{TM3}} \\ &= Q_0 \cdot \text{TM3} + Q_1 \cdot \overline{\text{TM3}} \end{aligned}$$

此方程也称为次态 Q^{n+1} 的转移方程，包含有转换条件表达式。

十字路口交通灯控制器次态方程式

由于次态为 $S^{n+1} = S \cdot (\text{转换条件表达式})$ ，由前表可得到次态方程式：

$$\begin{aligned} Q_1^{n+1} &= \overline{Q_1} \cdot Q_0 \cdot TM3 + Q_1 \cdot Q_0 \cdot \overline{NSCAR} \\ &\quad + Q_1 \cdot Q_0 \cdot NSCAR \cdot \overline{EWCAR} \\ &\quad + Q_1 \cdot Q_0 \cdot NSCAR \cdot EWCAR \cdot \overline{TM40} \\ &\quad + Q_1 \cdot Q_0 \cdot NSCAR \cdot EWCAR \cdot TM40 + Q_1 \cdot \overline{Q_0} \cdot \overline{TM3} \\ &= Q_0 \cdot TM3 + Q_1 \cdot \overline{TM3} \end{aligned}$$

$$\begin{aligned} Q_0^{n+1} &= \overline{Q_1} \cdot \overline{Q_0} \cdot \overline{NSCAR} \cdot \overline{EWCAR} + \overline{Q_1} \cdot \overline{Q_0} \cdot NSCAR \\ &\quad \cdot \overline{EWCAR} \cdot TM60 + \overline{Q_1} \cdot Q_0 \cdot \overline{TM3} + \overline{Q_1} \cdot Q_0 \cdot TM3 \\ &\quad + Q_1 \cdot Q_0 \cdot \overline{NSCAR} \cdot \overline{EWCAR} \\ &\quad + Q_1 \cdot Q_0 \cdot NSCAR \cdot \overline{EWCAR} \cdot \overline{TM40} \end{aligned}$$

由ASM图可直接写出输出函数表达式

$$EW_{\text{Green}} = \overline{Q_1} \cdot \overline{Q_0}$$

$$NS_{\text{Red}} = \overline{Q_1} \cdot \overline{Q_0}$$

$$NS_{\text{Yellow}} = \overline{Q_1} \cdot Q_0$$

$$EW_{\text{Yellow}} = \overline{Q_1} \cdot Q_0$$

$$NS_{\text{Green}} =$$

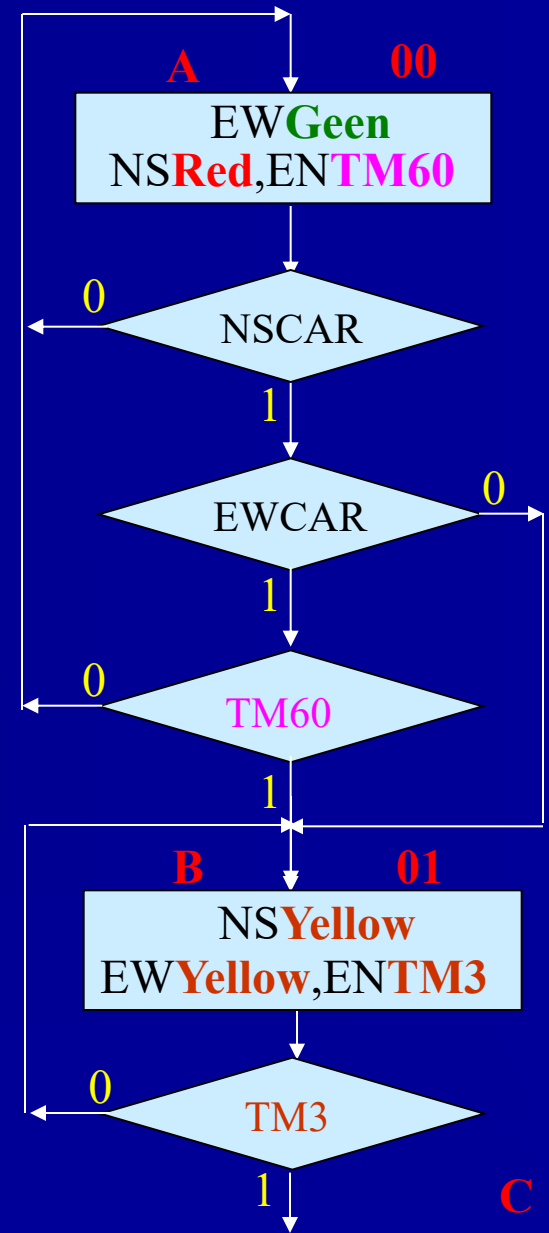
$$EW_{\text{Red}} =$$

$$ENTM60 = \overline{Q_1} \cdot \overline{Q_0}$$

$$ENTM40 =$$

$$ENTM3 = \overline{Q_1} \cdot Q_0$$

Moore型电路



由ASM图可直接写出输出函数表达式

$$\text{EWGreen} = \overline{Q_1} \cdot \overline{Q_0}$$

$$\text{NSRed} = \overline{Q_1} \cdot \overline{Q_0}$$

$$\text{NSYellow} = \overline{Q_1} \cdot Q_0 + \overline{Q_1} \cdot Q_0 = Q_1 \oplus Q_0$$

$$\text{EWYellow} = \overline{Q_1} \cdot Q_0 + \overline{Q_1} \cdot Q_0 = Q_1 \oplus Q_0$$

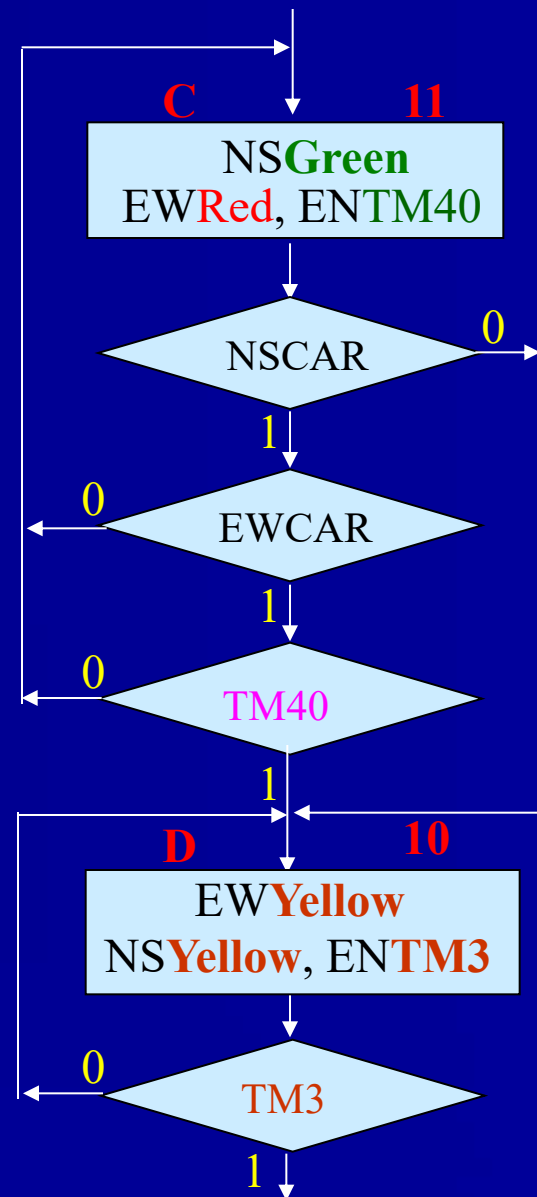
$$\text{NSGreen} = Q_1 \cdot Q_0$$

$$\text{EWRed} = Q_1 \cdot Q_0$$

$$\text{ENTM60} = \overline{Q_1} \cdot \overline{Q_0}$$

$$\text{ENTM40} = Q_1 \cdot Q_0$$

$$\text{ENTM3} = \overline{Q_1} \cdot Q_0 + Q_1 \cdot \overline{Q_0} = Q_1 \oplus Q_0$$

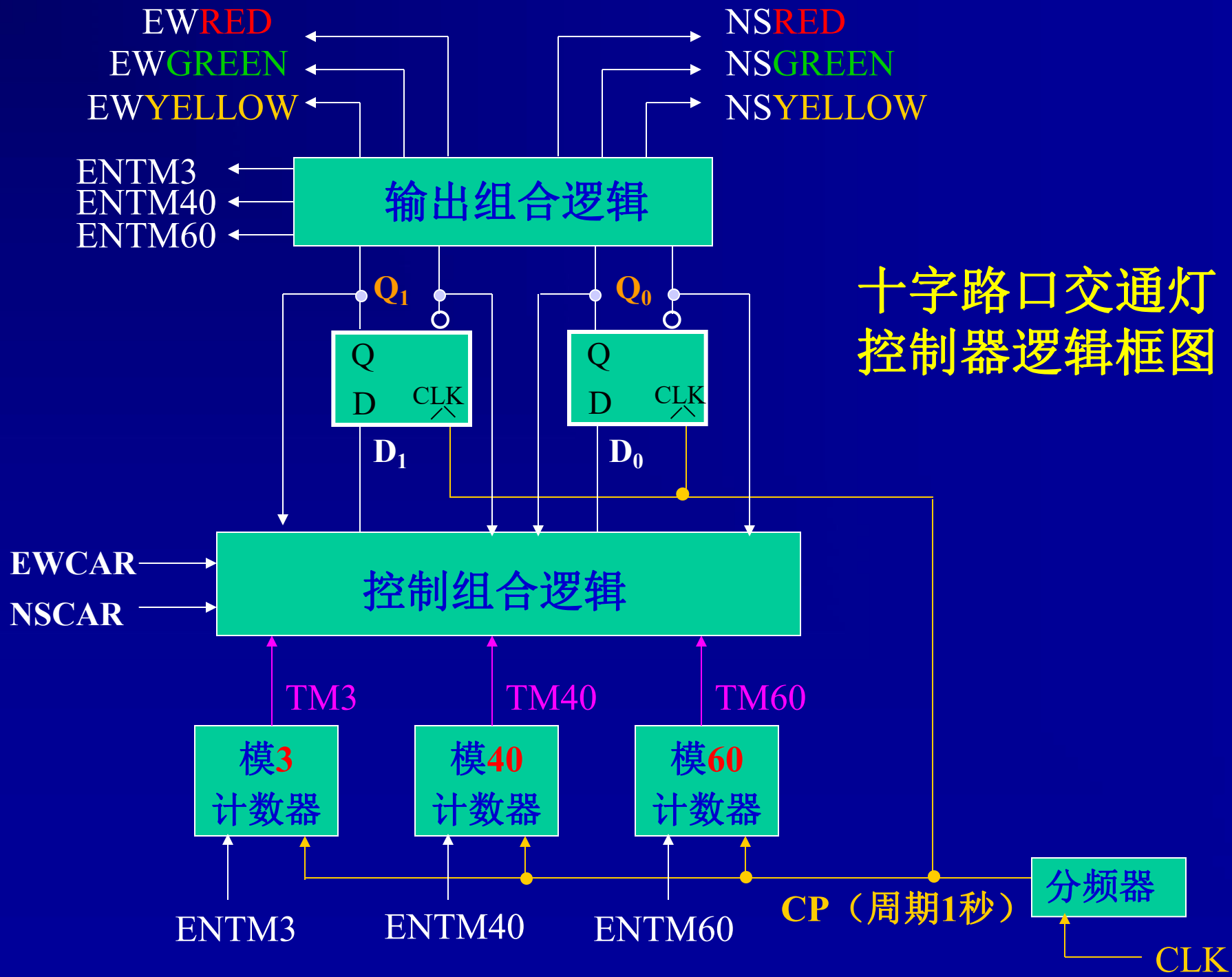


可选用**D**触发器或**JK**触发器及门电路构成控制系统。

选用**D触发器**构成的控制系统，次态 $Q^{n+1}=D$
则可直接写出控制函数：

$$\begin{aligned} D_1 &= \overline{Q_1} \cdot Q_0 \cdot TM3 + Q_1 \cdot Q_0 \cdot \overline{NSCAR} \\ &\quad + Q_1 \cdot Q_0 \cdot NSCAR \cdot \overline{EWCAR} \\ &\quad + Q_1 \cdot Q_0 \cdot NSCAR \cdot EWCAR \cdot \overline{TM40} \\ &\quad + Q_1 \cdot Q_0 \cdot NSCAR \cdot EWCAR \cdot TM40 + Q_1 \cdot \overline{Q_0} \cdot \overline{TM3} \\ &= Q_0 \cdot TM3 + Q_1 \cdot \overline{TM3} \end{aligned}$$

$$\begin{aligned} D_0 &= \overline{Q_1} \cdot \overline{Q_0} \cdot \overline{NSCAR} \cdot \overline{EWCAR} + \overline{Q_1} \cdot \overline{Q_0} \cdot NSCAR \\ &\quad \cdot EWCAR \cdot TM60 + \overline{Q_1} \cdot Q_0 \cdot \overline{TM3} + \overline{Q_1} \cdot Q_0 \cdot TM3 \\ &\quad + Q_1 \cdot Q_0 \cdot \overline{NSCAR} \cdot \overline{EWCAR} \\ &\quad + Q_1 \cdot Q_0 \cdot NSCAR \cdot EWCAR \cdot \overline{TM40} \end{aligned}$$



ASM图的特点

ASM图为时序电路系统提供了形式化描述方法。

在ASM图中，每个状态框仅有一个出口（分支由判断框提供），保证了无二义性，且自动满足**闭合性和完整性**。

由于每个ASM块只能描述系统的一个状态，对于计算机和类似的复杂数字系统来说，仅仅采用ASM图作为硬件的描述工具是不够的，因此，需要一种功能更强的形式化工具来描述和定义数字系统中的操作和实现这些操作的硬件结构。

寄存器传送语言（RTL）就是这种形式语言之一。

5.1.3.5 寄存器传送语言

寄存器传送语言(**RTL**)是一种用以描述数字系统各种设备以及它们之间相互连接和相互关系的**形式语言**。这是一种**硬件描述语言**，可以直接对操作过程和系统进行描述，并可由此得到系统的硬件结构。

用寄存器传送语言描述数字系统时，寄存器的涵义是广泛的，它包括了所有形式的寄存器：

如 **移位寄存器**
 计数器
 存储器 等。

计数器可看作具有使存储着的信息进行加、减和移位功能的寄存器；

存储器可以看作是存储信息的寄存器集合。

这样，任何形式的**时序电路**都被看作寄存器。

目前，存在着各式各样的寄存器传送语言，尚无统一的规范 and 标准，此节只**介绍一种最简便的寄存器传送语言**。

最简便的寄存器传送语言

这种寄存器传送语言的语句通常包括

控制功能和**微操作表**

其中：**控制功能**确定了控制条件和微操作的时序，

微操作表确定了对存储在寄存器中的信息所进行的基本操作。

控制功能可以由定时信号组成，也可以根据以前的操作结果决定。当控制信号为某一个二进制状态时，就启动某个操作，而为另一个状态时，则禁止这种操作。

数字系统中最常见的微操作有：

- **寄存器传送微操作**
- **算术微操作**
- **逻辑微操作**
- **移位微操作**

对存储在寄存器中的信息所进行的微操作与数据的类型有关。计算机常用的信息有**数字数据**、**非数字数据**及其他控制信息。

1、寄存器传送操作

对寄存器所存信息的加工和存储称为寄存器传送操作。

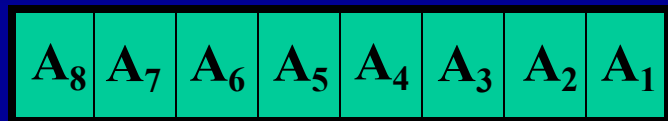
在数字系统中，寄存器一般用大写字母（有时还加上数字）表示。例：如下图，用符号A，B，R1，R2，MAR等表示寄存器。



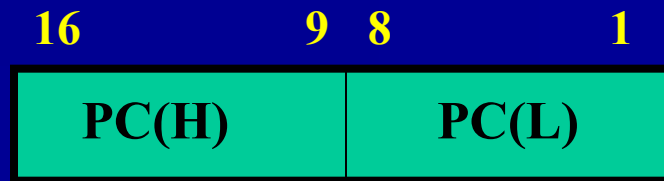
(a) 最常用的矩形框表示



(c) 一个12位寄存器



(b) 划分成若干单元并编号



(d) 划分为高(H)、低(L)两字节的16位寄存器

寄存器框图

例1. 传送语句 $A \leftarrow B$

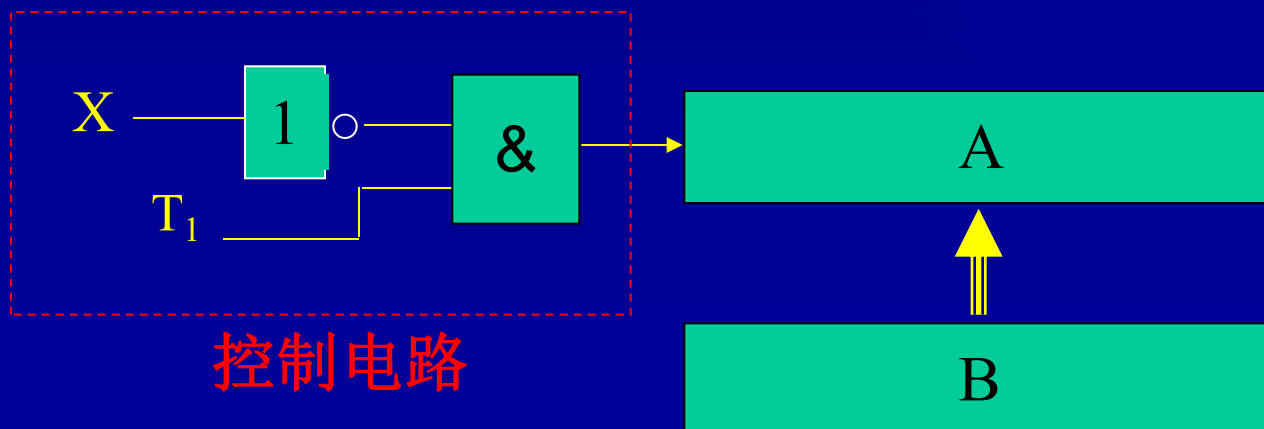
A: 目标寄存器

B: 源寄存器

(信息传送完成后, A、B内容相同)

例2. 条件传送语句 $\overline{X}T_1: A \leftarrow B$

表示只有当传送发生的条件(控制功能)成立时,
即 $\overline{X}T_1=1$ 时,才执行传送操作。如下图:



表示寄存器传送的基本符号

符 号	说 明	示 例
字母(及数字)	表示一个寄存器	A , MBR , R2
下标	表示寄存器中的一位	A ₂ , A ₆
括号()	表示寄存器中的一部分	PC (H), MBR (OP)
箭头←	表示信息的传送	A ← B
冒号 :	用于终止控制功能	\overline{X}T ₀ :
逗号 ,	用于分隔同时执行的多个微操作	A ← B , B ← C
方括号 []	用于指定寄存器字的地址	MBR ← M [MAR]

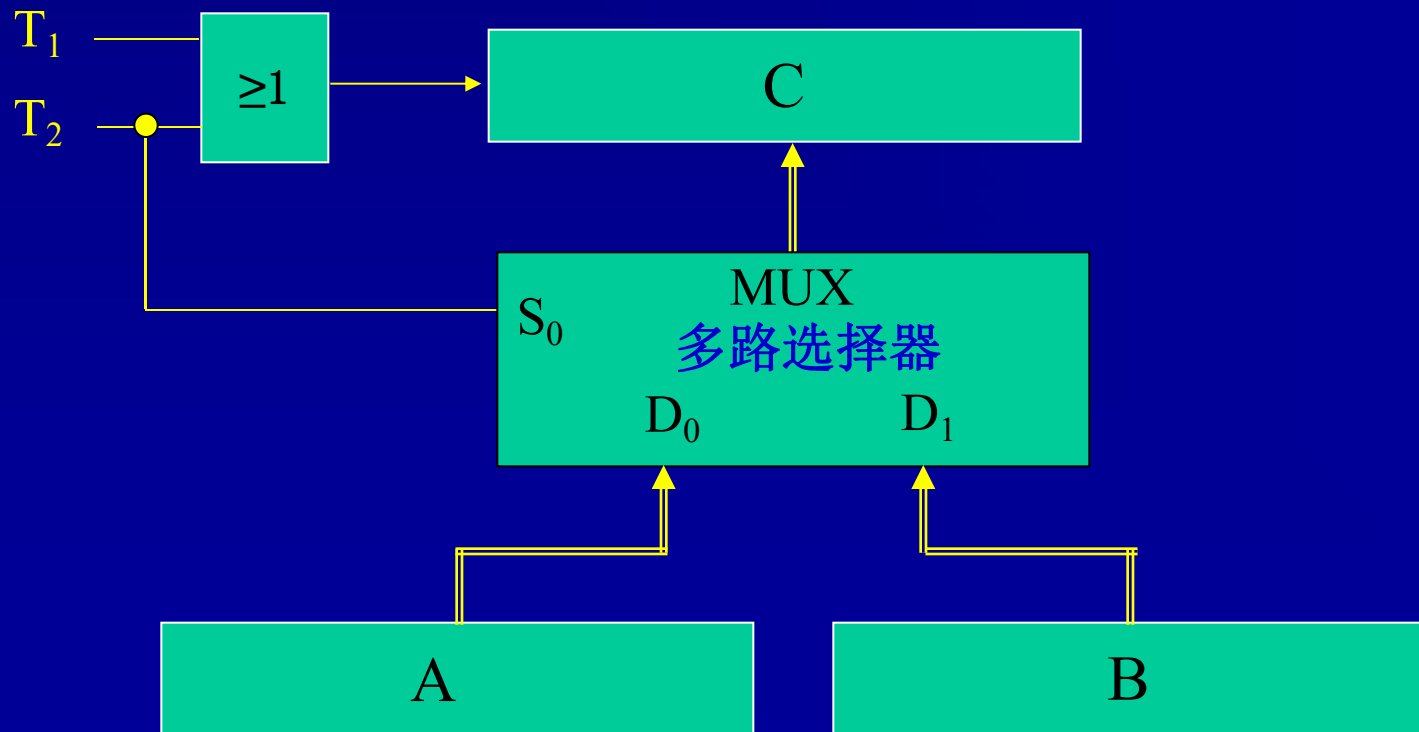
注: **MAR**通常表示为存储器的地址寄存器。

例3. 使同一目标寄存器接收来自两个源寄存器的
信息传送语句

$T_1: C \leftarrow A$

$T_2: C \leftarrow B$

注意：这个语句说明控制函数 T_1 、 T_2 是互斥的。
电路实现如下：



例4. 寄存器与存储器的信息传送语句

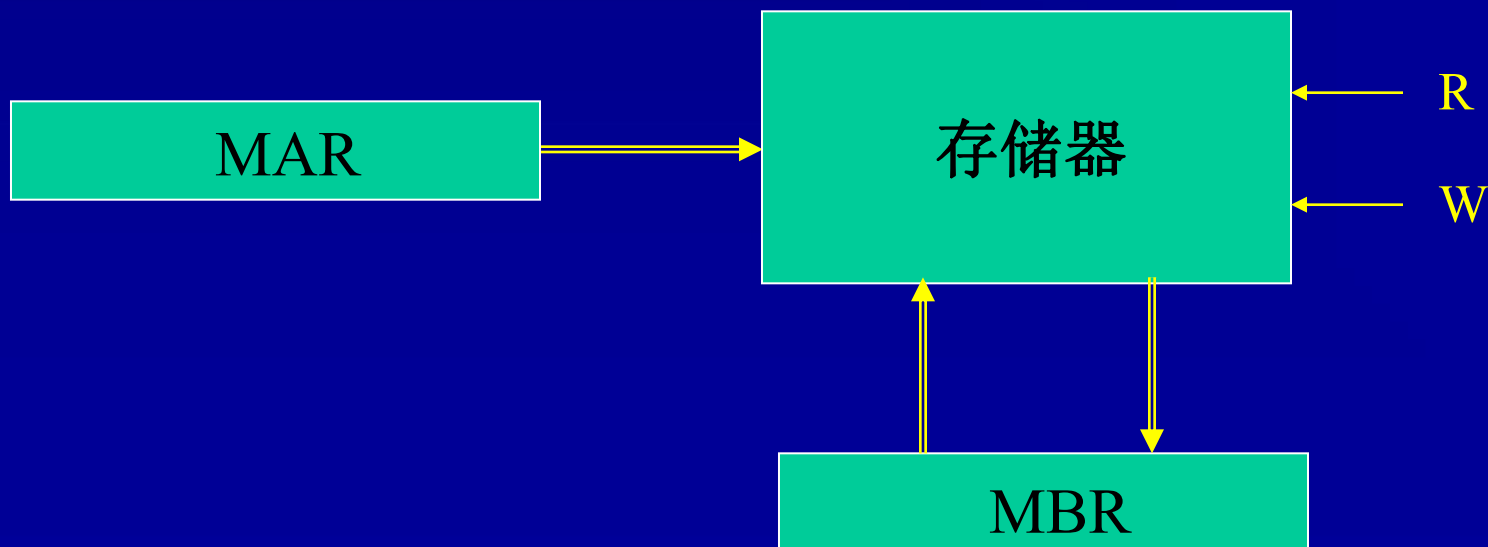
用字母**M**表示存储器，字母**M**后面的方括号则指明给定地址的存储器。

功能：信息可以从存储器读出，
也可以将信息写入存储器。

例如：

W: M[MAR] \leftarrow MBR

R: MBR \leftarrow M[MAR]



2、算术操作

寄存器 之间的算术微操作，如加、减、取反等。

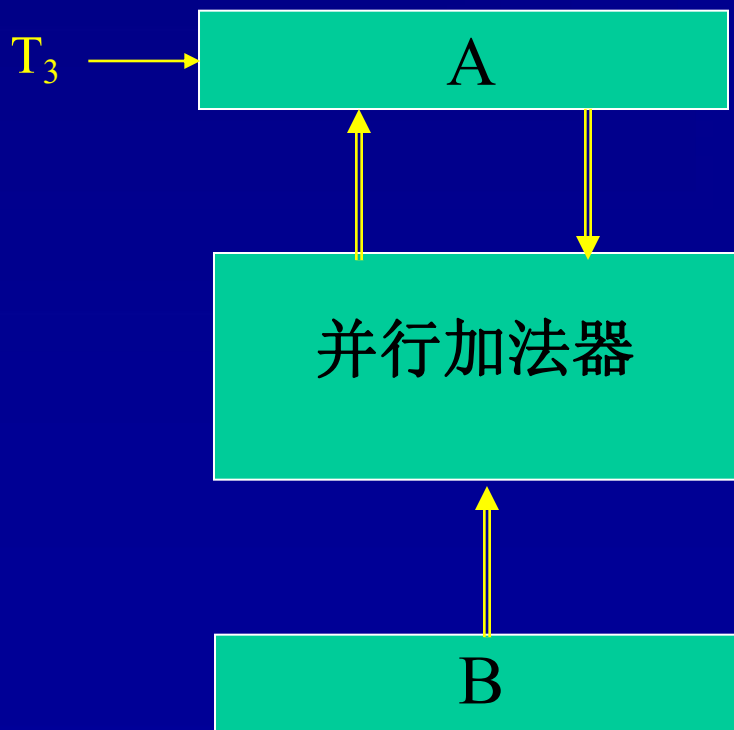
基本算术微操作

符 号	说 明
$F \leftarrow A+B$	寄存器A和B的内容相加结果传送到寄存器F
$F \leftarrow A-B$	寄存器A和B的内容相减结果传送到寄存器F
$B \leftarrow \bar{B}$	对寄存器B的内容取反码，结果留B中
$B \leftarrow \bar{B}+1$	对寄存器B的内容取补码，结果留B中
$F \leftarrow A+\bar{B}+1$	寄存器A和B的内容相减(取补码相加)结果传送到寄存器F
$A \leftarrow A+1$	寄存器A的内容加1，结果留A中
$A \leftarrow A-1$	寄存器A的内容减1，结果留A中

例1. 相加操作语句

$T_3: A \leftarrow A+B$

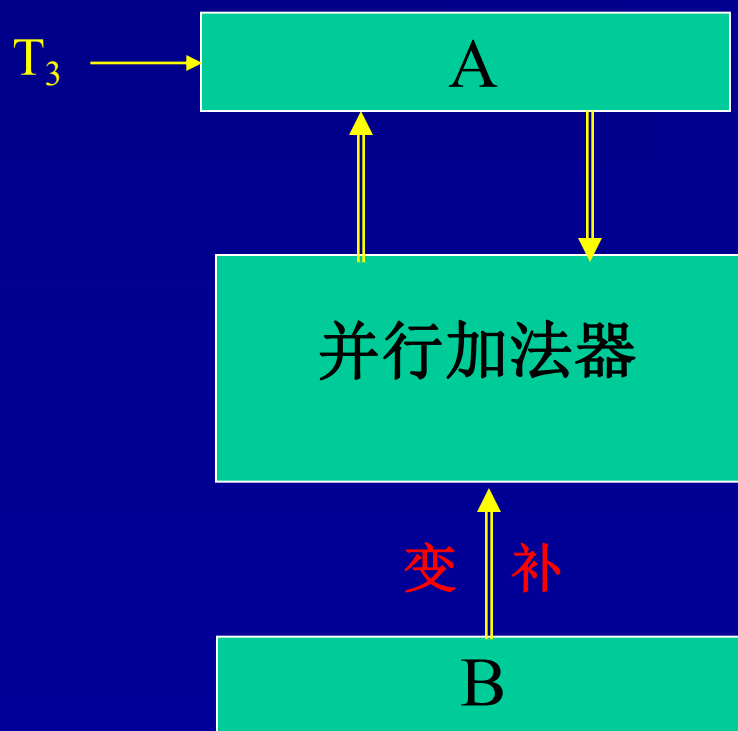
表示**加法操作**，将寄存器A的内容和寄存器B的内容相加，结果送到寄存器A。实现这行语句需要A、B两个寄存器，以及执行加法运算的逻辑部件（如并行加法器），其逻辑框图如下：



例2. 相减操作语句

$$T_3: A \leftarrow A + \bar{B} + 1$$

表示加法操作，将寄存器A的内容和寄存器B的补码相加，这是一个减法微操作，运算结果送到寄存器A。实现这行语句需要A、B两个寄存器，以及执行补码加法运算的逻辑部件，其逻辑框图如下：



3、逻辑操作

对寄存器内的每一位信息进行的二进制微操作，因此这每一位信息当作一个二进制变量来处理。

逻辑微操作

符 号	说 明
$F \leftarrow \overline{A}$	寄存器A的所有位按位 求反 ，结果送寄存器F
$F \leftarrow A \vee B$	寄存器A和B的 对应位相“或” ，结果送寄存器F
$F \leftarrow A \wedge B$	寄存器A和B的 对应位相“与” ，结果送寄存器F
$F \leftarrow A \oplus B$	寄存器A和B的 对应位相“异或” ，结果送寄存器F

例. $T_1 + T_2 : A \leftarrow A + B, C \leftarrow D \vee E$

表示：控制功能为两个控制变量 T_1 、 T_2 相“或”。

当 $T_1 + T_2 = 1$ 时，寄存器A和B的内容相加，结果送寄存器A，

而寄存器D和E的对应位相“或”，结果送寄存器C。

符号“+”有两种涵义，当它在微操作中出现时，表示

算术加微操作；

当它在控制功能中出现时，表示

“或”操作。

4、移位操作

在寄存器之间数据的**移位**微操作，也可以表示**算术**、**逻辑**微操作和**控制功能**。

移位微操作

符 号	说 明
$A \leftarrow \text{Shl}A$	寄存器A左移一位
$A \leftarrow \text{Shr}A$	寄存器A右移一位

例1. $W_1: A \leftarrow \text{Shl}A$

$W_2: B \leftarrow \text{Shr}B$

当寄存器**右移**时，**最左边**的触发器从串行输入端接收信息，
当寄存器**左移**时，**最右边**的触发器从串行输入端接收信息。

符号**Shl**和**Shr**并没有对串行输入信息作出说明，必须用另一个微操作来确定。

例2. $W_3: A \leftarrow \text{Shl}A, A_1 \leftarrow A_n$

表示寄存器A**循环左移**，移位时，将**最左边**的触发器 **A_n** 中的信息送入**最右边**的触发器 **A_1** 。

例3. $W_4: B \leftarrow \text{Shr}B, B_n \leftarrow E$

表示寄存器B**循环右移**，移位时，将触发器**E**的一位信息送入**最左边**的触发器 **B_n** 。

例1. $W_1: A \leftarrow \text{Shl}A$

$W_2: B \leftarrow \text{Shr}B$

当寄存器**右移**时，**最左边**的触发器从串行输入端接收信息，
当寄存器**左移**时，**最右边**的触发器从串行输入端接收信息。

符号**Shl**和**Shr**并没有对串行输入信息作出说明，必须用另一个微操作来确定。

5、条件控制语句

P: **if** (条件) **then**(微操作1) **else** (微操作2)

表示：当控制函数 **P=1** 时，

如果 **if** 后面括号内的 条件满足，

则，执行 **then** 后面括号内的微操作1，

否则，执行**else**后面括号内的微操作2。

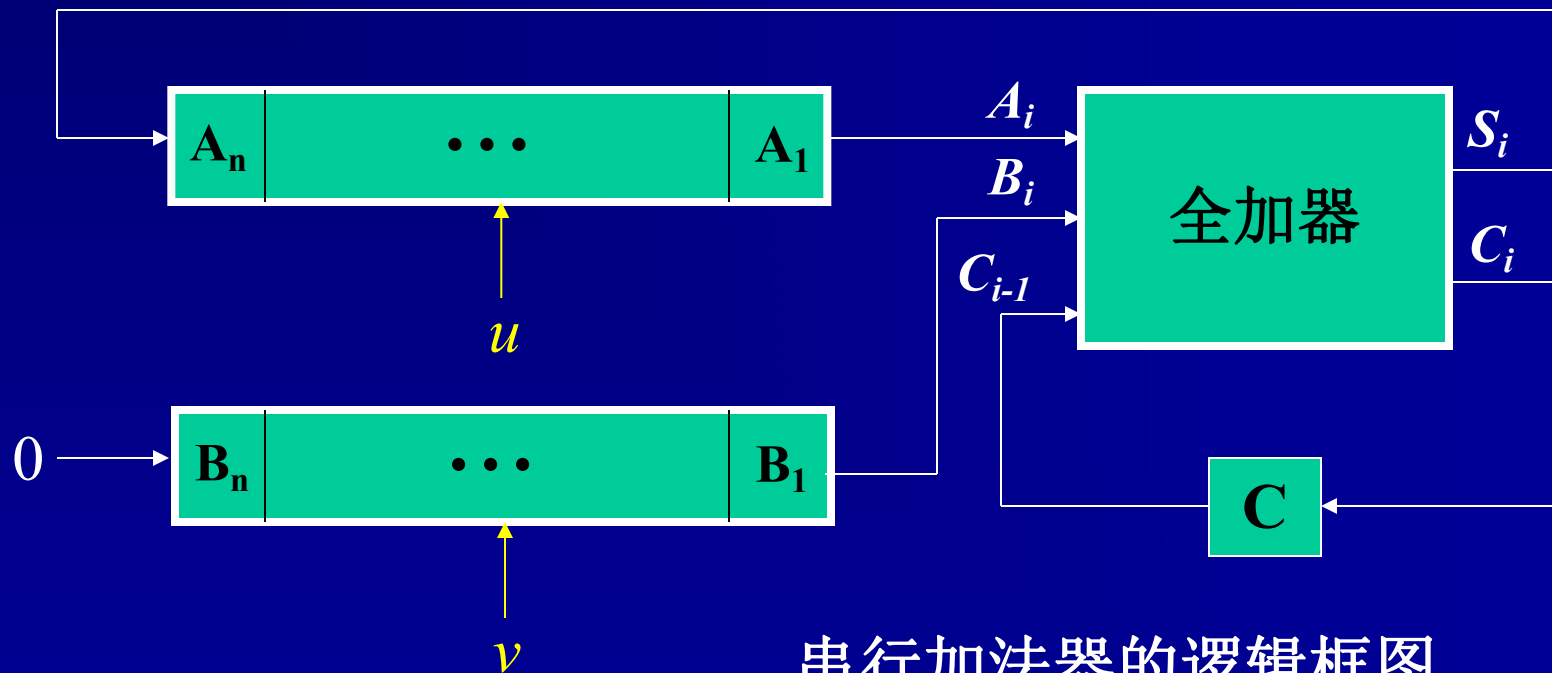
例 **T₂:** **if** (**C=0**) **then** (**F**← **1**) **else** (**F** ← **0**)

它与下面两个语句是等价的：

\overline{C} **T₂:** **F** ← **1**

C **T₂:** **F** ← **0**

例 串行加法器的逻辑框图如下所示，使用寄存器传送语言描述图示的串行加法器的工作。



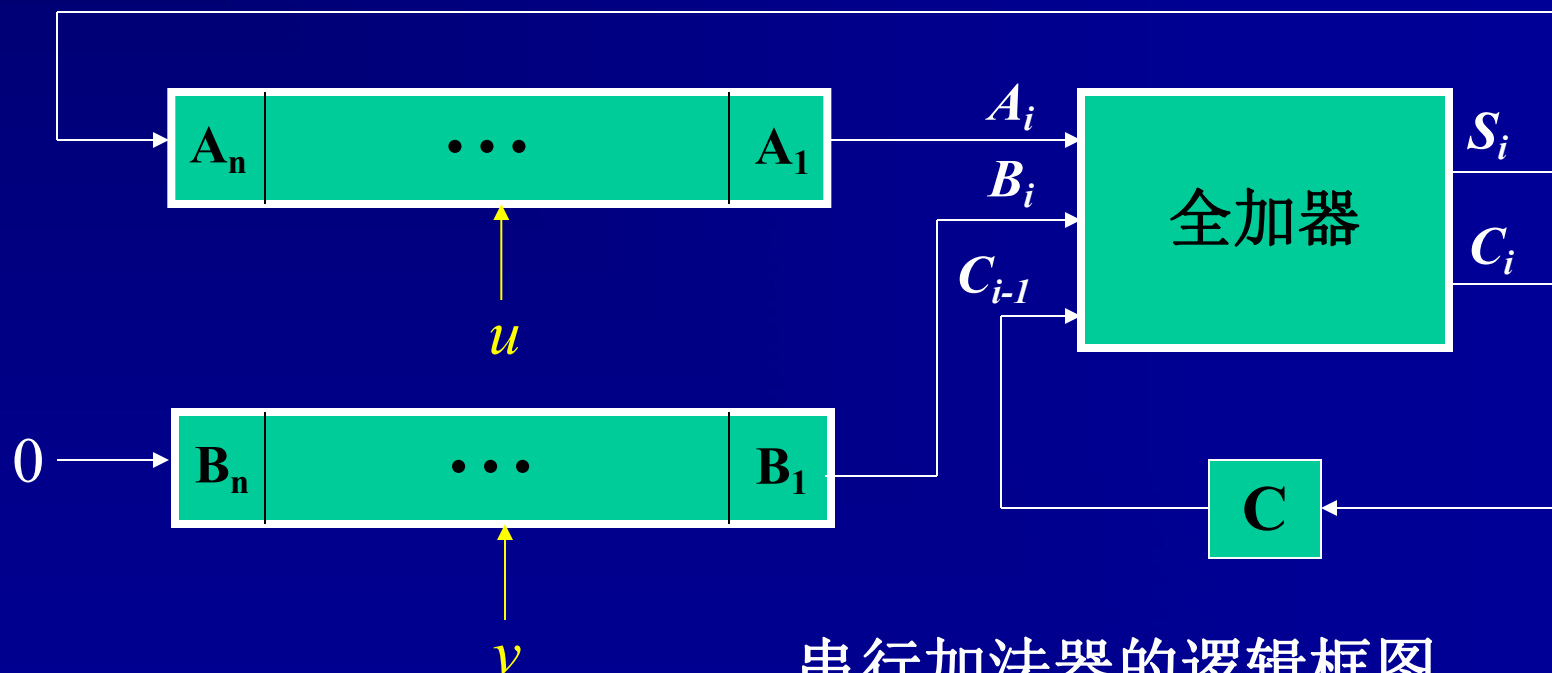
串行加法器的逻辑框图

设：寄存器A中的被加数 $U = u_n u_{n-1} \dots u_2 u_1$

寄存器B中的加数 $V = v_n v_{n-1} \dots v_2 v_1$

工作过程：置进位触发器C的初始状态 C_0 为0；当第一个时钟信号 T_1 作用时，全加器完成最低位 u_1 和 v_1 相加，并将和数 S_1 送入寄存器的最高位A，寄存器A和B同时右移一位；

例 串行加法器的逻辑框图如下所示，使用寄存器传送语言描述图示的串行加法器的工作。

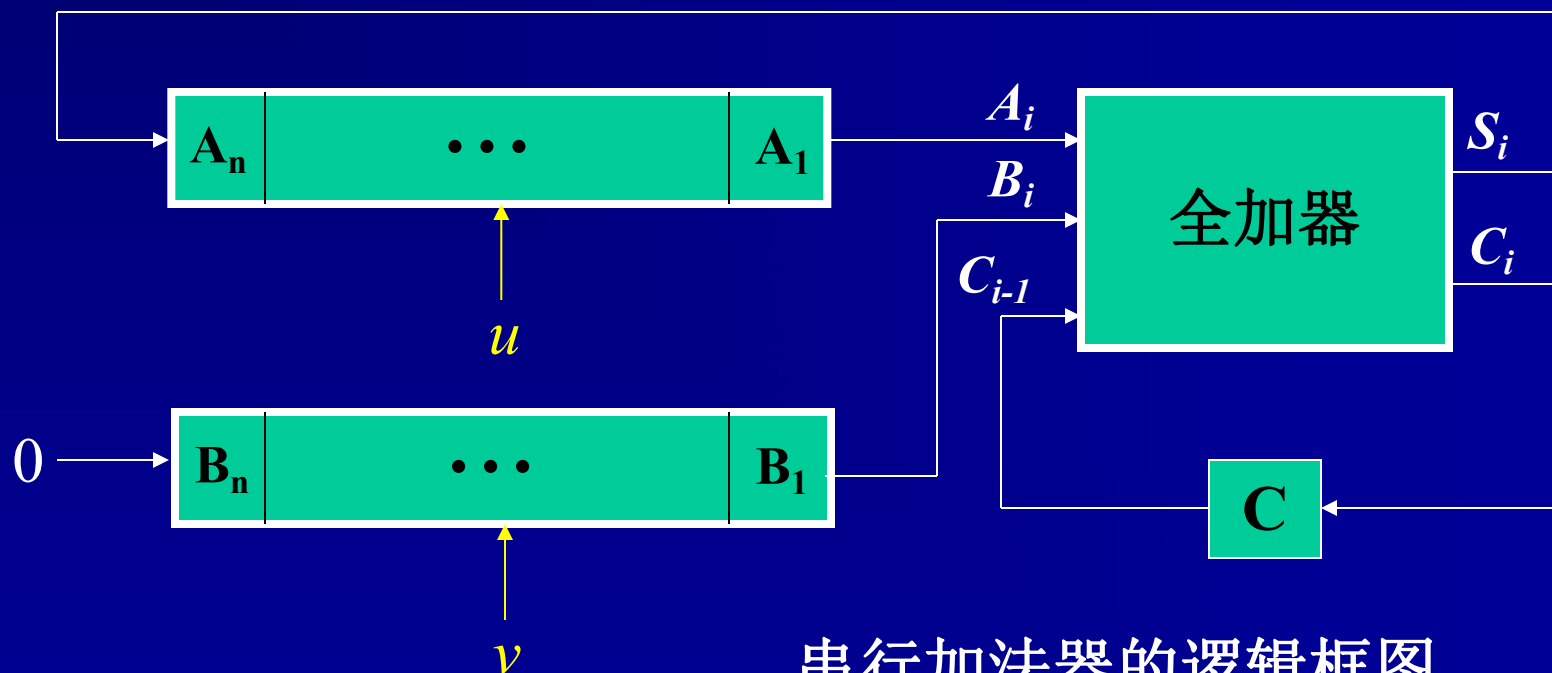


串行加法器的逻辑框图

当寄存器A和B右移一位后，送到全加器的是被加数A和加数B的新的最低位 u_2 和 v_2 ，若此次运算有进位 C_1 发生，则进位触发器C置为1。

当第二个时钟信号 T_2 作用时，全加器完成 u_2 、 v_2 和进位 C_1 相加，和数 S_2 送入寄存器的最高位A，寄存器A和B又同时右移一位，并将新的进位 C_2 送入触发器C。

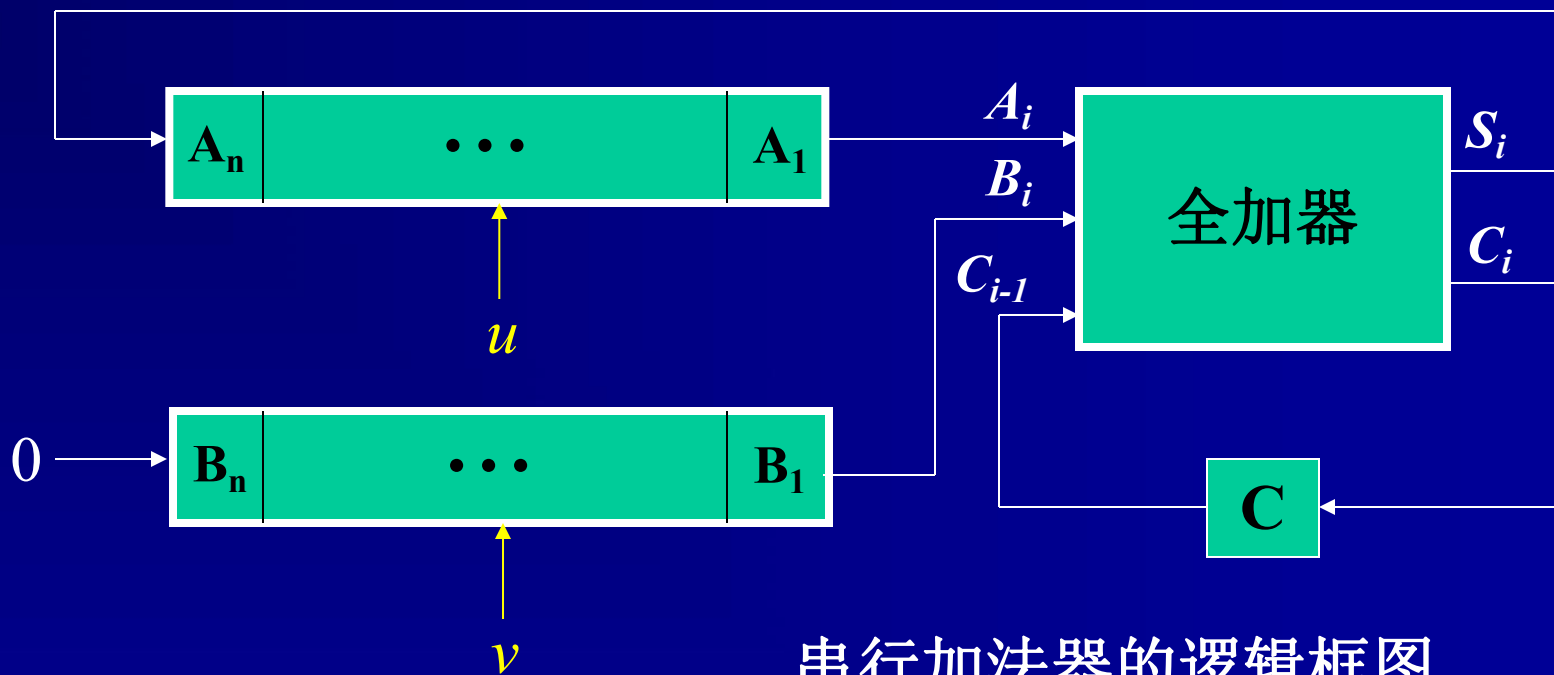
例 串行加法器的逻辑框图如下所示，使用寄存器传送语言描述图示的串行加法器的工作。



串行加法器的逻辑框图

这样，通过寄存器移位、整个串行相加过程，使全加器依次对被加数和加数的每一位 u_i 、 v_i 连同前一位的进位 C_{i-1} 逐位相加。

用寄存器传送语言描述整个串行加法器的工作，微操作序列：



串行加法器的逻辑框图

$$\mathbf{T}_1: \quad A_n \leftarrow A_1 \oplus B_1 \oplus C_0, \quad C \leftarrow (A_1 \wedge B_1) \vee (A_1 \wedge C_0) \vee (B_1 \wedge C_0)$$

$$A \leftarrow \text{Shr}A, \quad B \leftarrow \text{Shr}B, \quad B_n \leftarrow 0$$

$$\mathbf{T}_2: \quad A_n \leftarrow A_1 \oplus B_1 \oplus C_0, \quad C \leftarrow (A_1 \wedge B_1) \vee (A_1 \wedge C_0) \vee (B_1 \wedge C_0)$$

$$A \leftarrow \text{Shr}A, \quad B \leftarrow \text{Shr}B, \quad B_n \leftarrow 0$$

\vdots

\vdots

$$\mathbf{T}_n: \quad A_n \leftarrow A_1 \oplus B_1 \oplus C_0, \quad C \leftarrow (A_1 \wedge B_1) \vee (A_1 \wedge C_0) \vee (B_1 \wedge C_0)$$

$$A \leftarrow \text{Shr}A, \quad B \leftarrow \text{Shr}B, \quad B_n \leftarrow 0$$