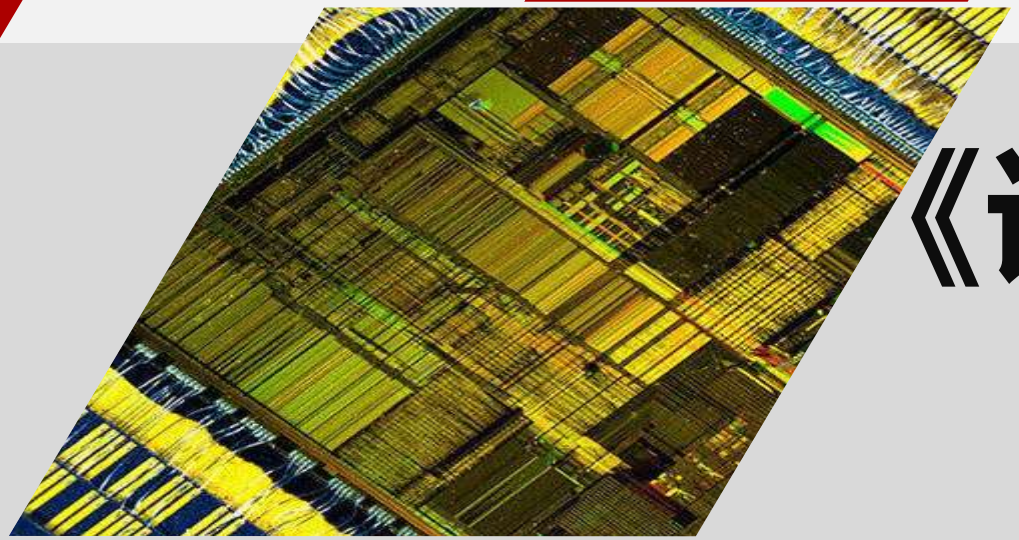
2023年春季学期



《计算机系统》

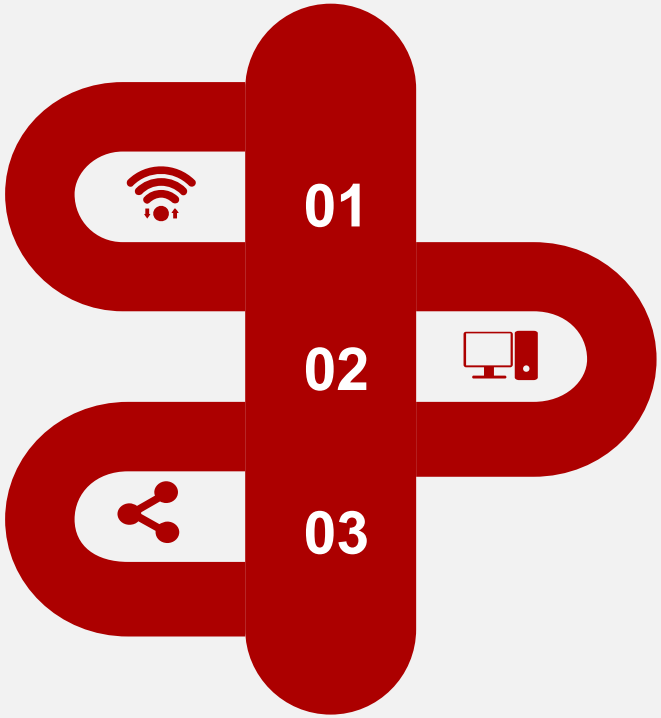
RTL语言

《计算机系统》课程教学组

内容提要

微操作与
寄存器传送语言

一个简单实例



用RTL描述数字系统

微操作（micro operation--μop）

大部分时序数字系统的基础，是更简单的行为。

- 数据从一个寄存器、存储器单元或者I/O设备到另一个的传送
- 修改存储的值
- 执行算术或逻辑功能

重要性：确定时序数字系统

- 确定正确的微操作传送
- 确定保证微操作按正确的顺序执行

硬件描述语言（hardware description language，**HDL**）

电路分析和设计（circuit analysis and design，**CAD**）软件

微操作和寄存器传送语言

设计时序数字系统：

- **首先用微操作表述系统的行为**
- **设计硬件来匹配这些表述**

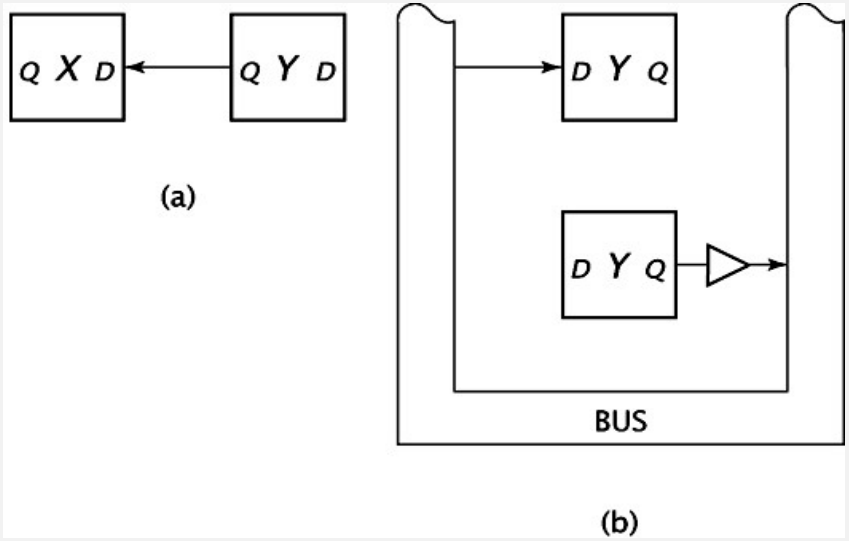
微操作和寄存器传送语言

一、微操作的格式

- 考虑有两个1位寄存器X和Y的一个数字系统。
- 拷贝寄存器Y的内容到寄存器X中的微操作：**X←Y**（有时也可以表示为**Y→X**）

两种实现：

- 由直接连接实现（a）
- 通过总线连接实现（b）



二、传送发生的条件

- 假定传送应发生在输入控制α为高电平时，则传送过程可以写为

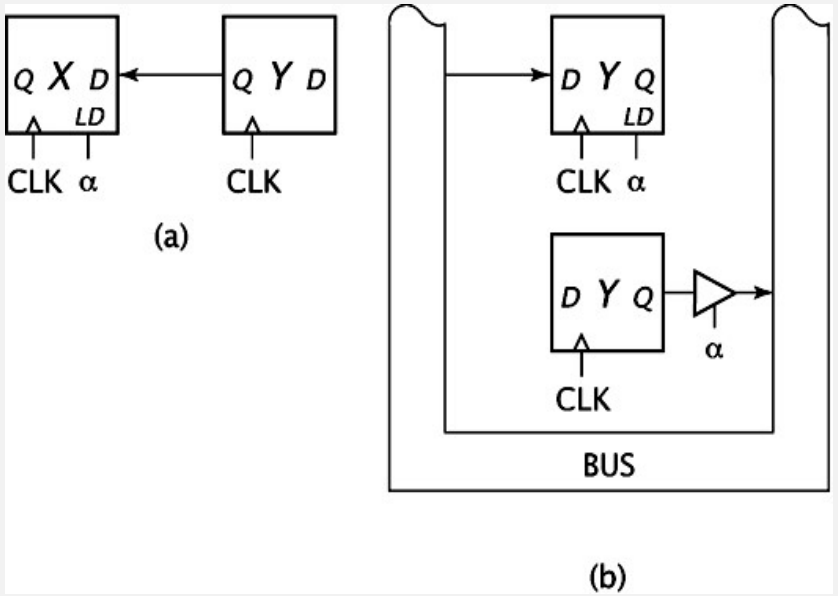
  IF α THEN X←Y

- 表示微操作和它们发生的条件：

  条件：微操作

- 当所有冒号左边的条件满足时，执行微操作（可以是多个）规定的数据传送。
- 上面的传送可以写为

  α：X←Y



具有控制信号的数据传送 α：X←Y 的实现

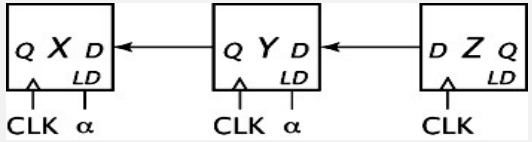
(a)直接通路 (b) 总线

微操作和寄存器传送语言

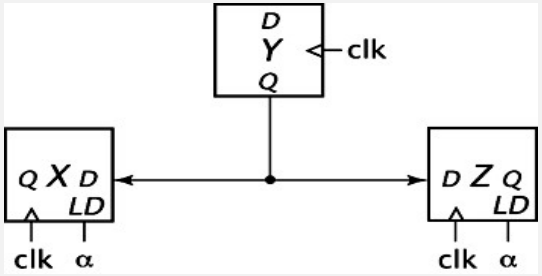
三、提高系统性能的一种方法：同时执行两个或多个微操作

1. 如果某系统在 α=1 时执行 X←Y 和 Y←Z 的传送，则这种情况可以表示成：

α：X←Y，Y←Z 或者 α：Y←Z，X←Y



2. 同时拷贝相同的数据到多个目的地：**α： X←Y，Z←Y （注意硬件上“同时”的概念！）**



3. 数字系统不能同时往同一寄存器中写入两个不同的值。

例如： **α：X←Y，X←Z** 无效！

四、传送常量的有效条件和微操作

α: X←0

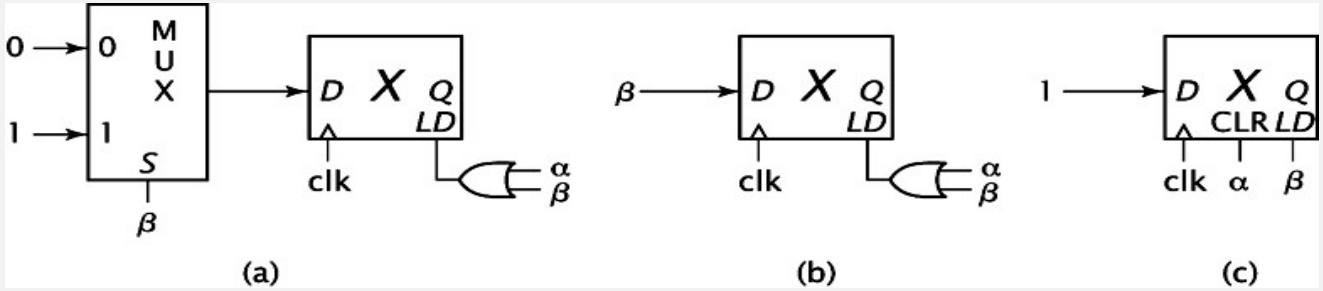
β: X←1

实现这两个传送的三种不同方法：

a. 通过两个传送通路装载数据来设置X的值（多路选择器）

b. 装载数据与 a 完全相同，但它的数据直接由信号β产生（用β作为数据输入）

c. 为简化硬件可使用寄存器的清除输入功能（用CLR信号）



微操作和寄存器传送语言

当α和β同时为1时，怎么办?

两种解决方法:

- 产生 α 和 β 的硬件能保证它们决不会被同时置为1
- 修改条件使它们互斥

αβ′ : X←0　　α : X←0　　αβ′ : X←0

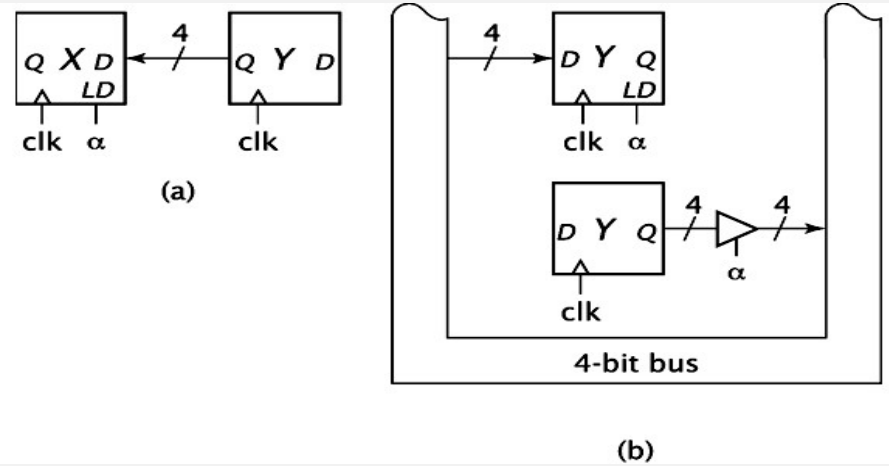
β : X←1　　α'β : X←1　　α'β : X←1

五、寄存器之间的多位数据传送

1. 同样大小寄存器对应位之间传送数据

例如: 当α = 1时，从4位寄存器Y传送数据到4位寄存器X用符号表示

α: X←Y



(a)直接通路 **(b) 总线**

微操作和寄存器传送语言

2.访问一个寄存器的某一位或位组。

表示每一位：如 X_3 或 Y_2

3.表示位组（用一个域表示）

如X_3、X_2和X_1可以写成 X(3-1) 或 X(3:1)

α: X(3-1)←Y(2-0)

β: X_3←X_2

γ: X(3-0)←X(2-0), X_3 --- X(2-0), X_3 等价于 X(2-0,3)

当 γ =1时，X_3←X_2，X_2←X_1，X_1←X_0，X_0←X_3

六、执行数据的算术运算、逻辑运算和移位运算的微操作

1．一些常用的算术运算和逻辑运算的微操作

操　作	示　例
Add	X←X + Y
Subtract	X←X – Y **或** X←X + Y’ + 1
Increment	X←X + 1
Decrement	X←X–1
And	X←X ∧ Y **或** X←XY
OR	X←X ∨ Y
XOR	X←X ⊕ Y
NOT	X←/X **或** X←X’

2．移位微操作

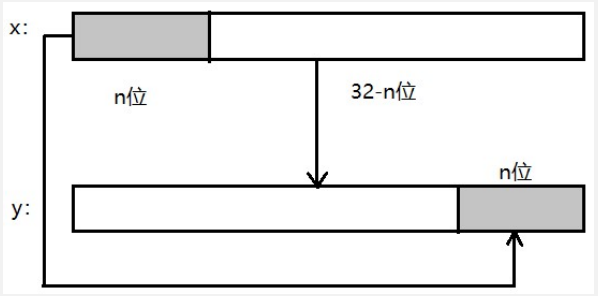
- **线性移位**：每一位的值依次向左（或右）移位。

  最后1位被丢弃，**空位补入0值**。

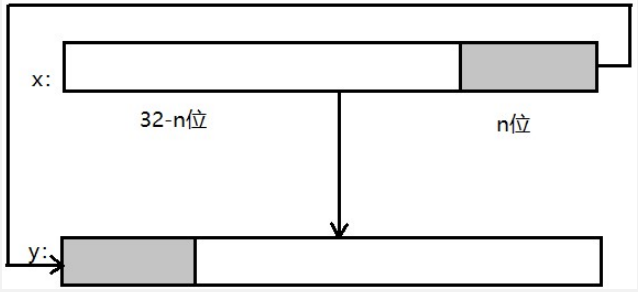
  例如： X = 1011

  左移一位：0110　　右移一位：0101

- **循环移位**：将在线性移位中被丢弃的位循环回来，替代补入的0值。



(a) 循环左移



(b) 循环右移

微操作和寄存器传送语言

- **算术移位**：用于带符号数的移位

  运算规则：符号位在移位操作中保持不变，工作原理与线性移位相似

  例如： X = 1011　　(教材120页，3.5.3关于算术左移有错误！)

  算术左移：1110　　算术右移：1101

- 十进制移位：专门用于BCD表示

  十进制移位与线性移位很相似，但它移动1个数字或4位，而不是移动1位

  例如：X = 1001 0111

  进制左移：0111 0000　进制右移：0000 1001

微操作和寄存器传送语言

◆ 移位操作和它们的表示法

操　作	示　　例
线性左移	shl(X)
线性右移	shr(X)
循环左移	cil(X)
循环右移	cir(X)
算术左移	ashl(X)
算术右移	ashr(X)
十进制左移	dshl(X)
十进制右移	dshr(X)

例如：

X←shl(X) 和 shl(X) 是等价的

Y←shl(X) 两个寄存器均需指定

微操作和寄存器传送语言

七、寄存器与存储器之间的数据传送

例如： **M[55]←AC和AC←M[55]**

寄存器AC与存储器中55号单元之间的数据传送

更好的方法：把地址存入寄存器中，然后由寄存器提供

存储器的访问地址（地址寄存器，标示为AR-Address Register）。

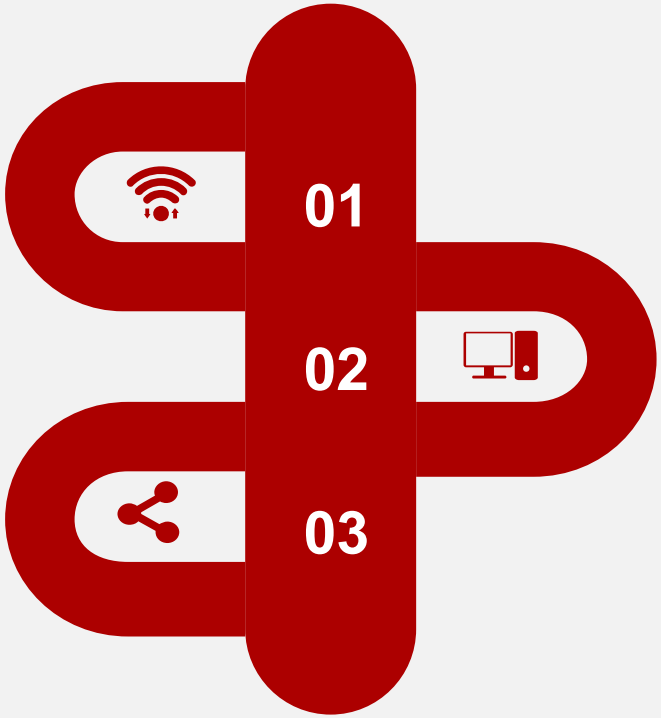
AR←55

M[AR]←AC 或者 **AC←M[AR]（M←AC和AC←M)**

内容提要

微操作与
寄存器传送语言

一个简单实例



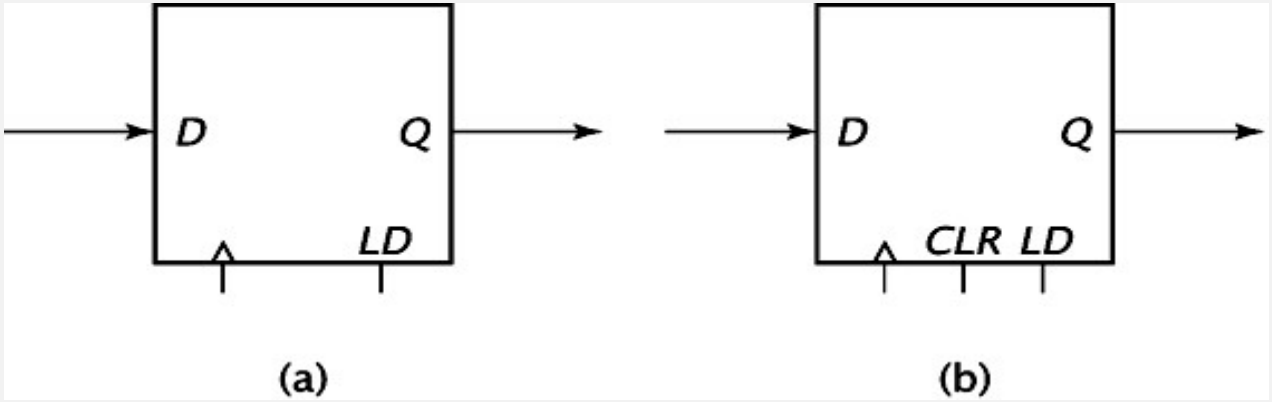
用RTL描述数字系统

用RTL表示数字系统

数字元件表示

一、第一个例子 – D触发器

1. 用RTL描述：　　**LD：Q←D**



D触发器：　　(a)无清0输入端　　(b)有清0输入端

2. 有同步清零输入端的D触发器

LD： Q←D

CLR： Q←0

当D、LD和CLR都等于1时，系统会报错。

解决方法：改变条件使得两者互斥，即当互斥时才执行操作。

CLR’LD： Q←D　　**LD： Q←D**

CLR： Q←0　　**LD’CLR： Q←0**

用RTL表示数字系统

二、第二个例子（一个没有CLR输入端的JK触发器）

用RTL描述：　**J'K：　Q←0**

JK'：　Q←1

JK　：　Q←Q'

三、第三个例子（一个n位的移位寄存器）

- Q_{n-1}是最高位，Q_0是最低位。
- 当SHL信号为高时，它将其中的数据左移一位。
- 输入S_{in}移进最低位。

移位寄存器：　**SHL：　Q←Q(n-2:0)，S_{in}**

用RTL表示数字系统

简单系统的表示与实现

一个有4个1位触发器的系统，用RTL代码表示传送

（假设条件 j，o，h 和 n 是互斥的）

j：M←A

o：A←Y

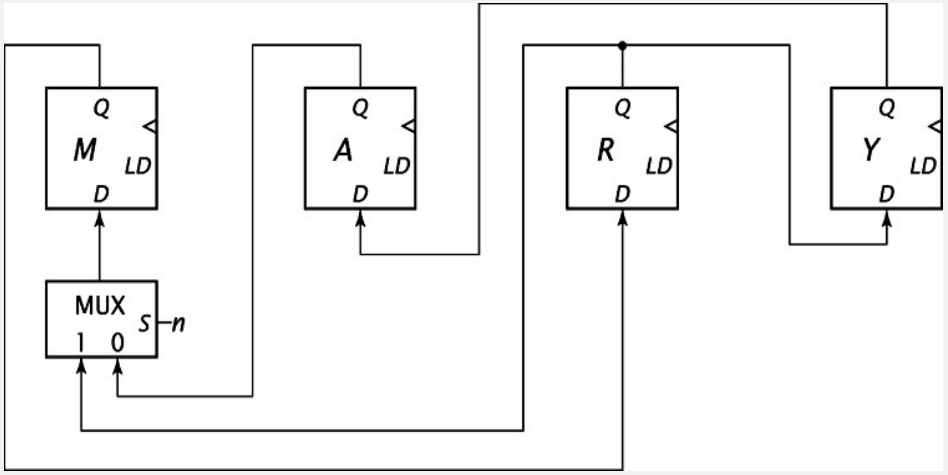
h：R←M

n：Y←R, M←R

用RTL表示数字系统

几种不同的方法实现

1．用直接连接实现系统的数据通路



j：M←A

o：A←Y

h：R←M

n：Y←R, M←R

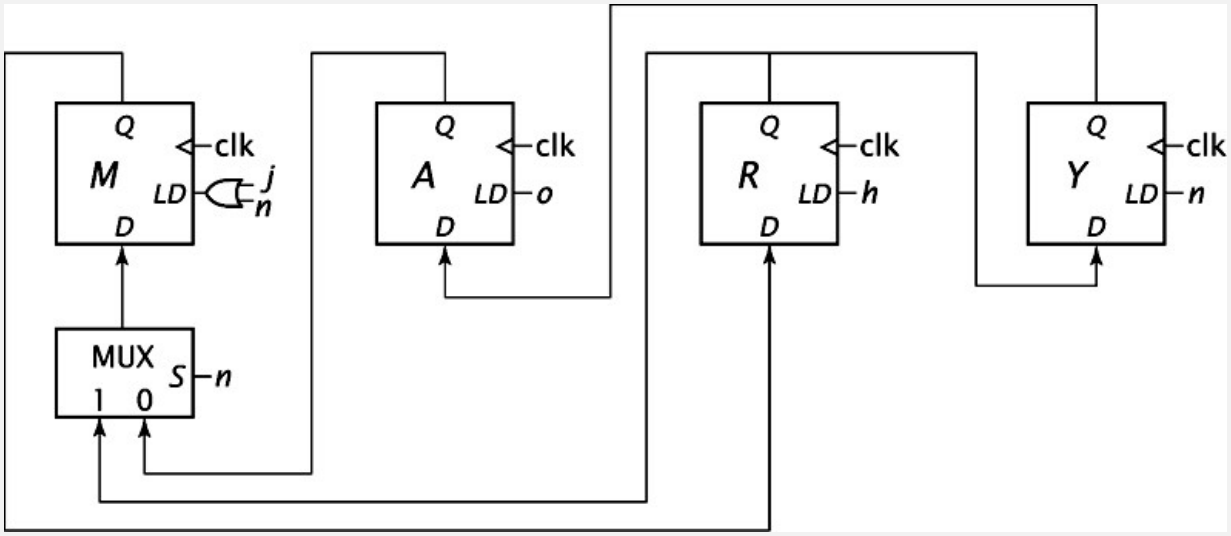
当 j=1（M←A）或 n=1(M←R) 时，触发器M装载数据。

在满足单个条件 o、h 和 n 时，触发器A、R和Y装载数据。

用RTL表示数字系统

用直接连接实现该RTL代码的系统的**完整设计**

在合适的时间激励触发器的**LD**信号来装载数据，从而完成传送。



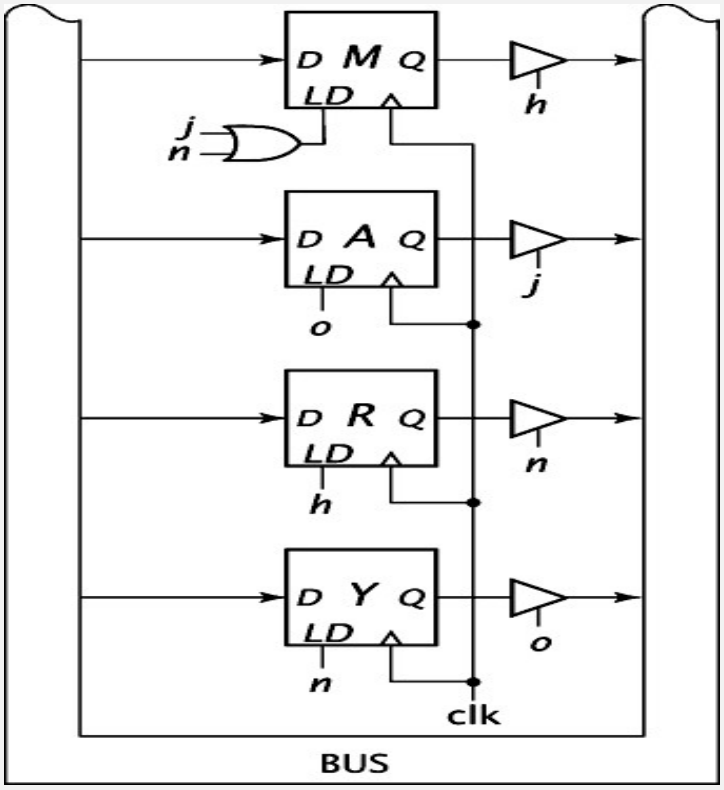
j : M←A

o : A←Y

h : R←M

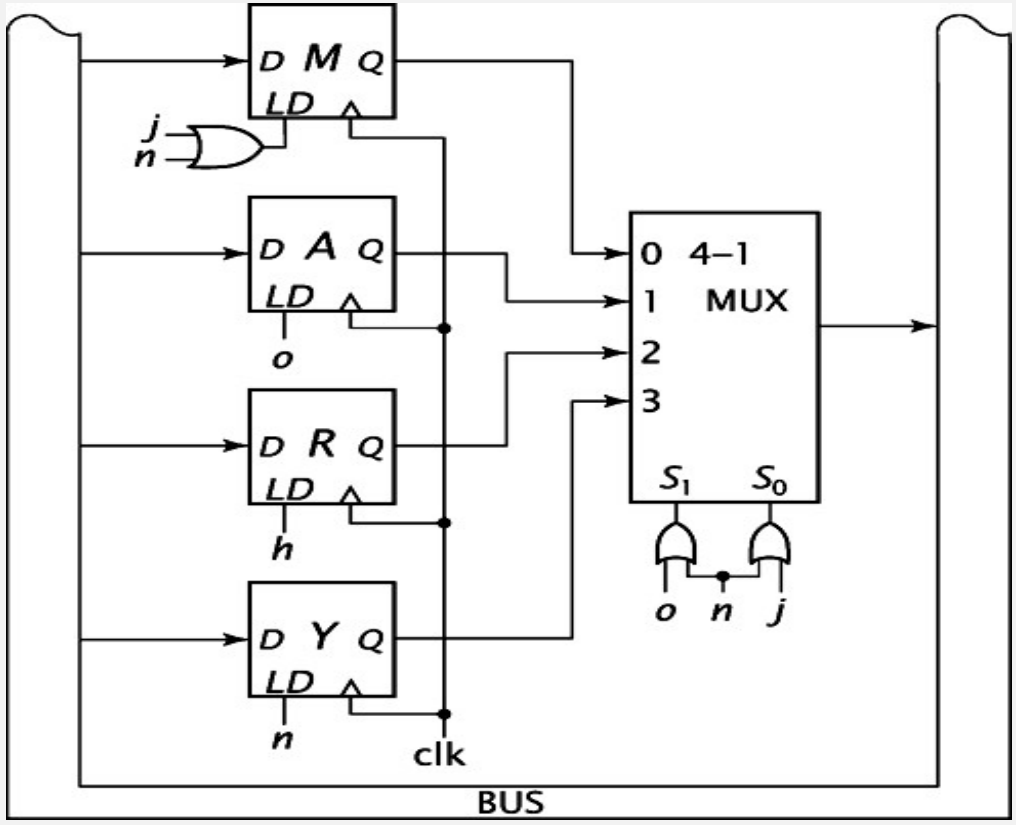
n : Y←R, M←R

2．用总线和三态门实现



用RTL表示数字系统

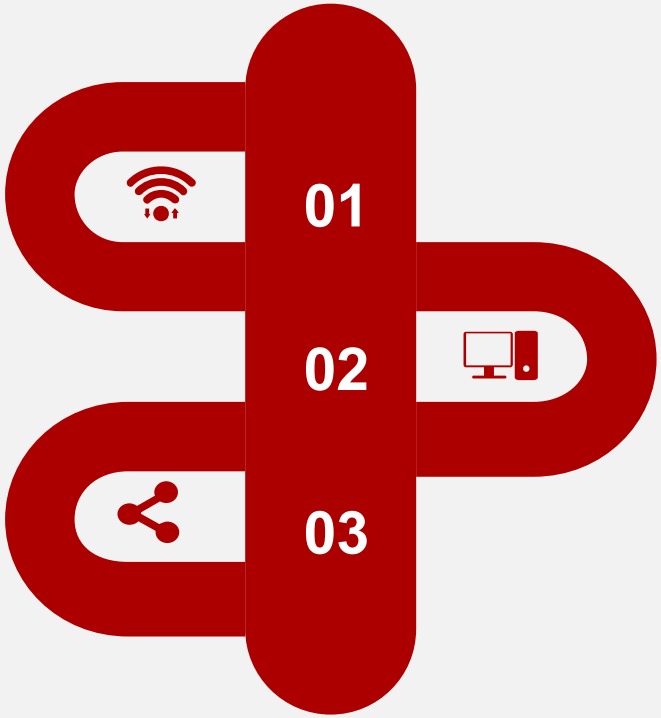
3. 用总线和多路选择器实现



内容提要

微操作与
寄存器传送语言

一个简单实例



用RTL描述数字系统

一个简单实例

更复杂数字系统和RTL——模6计数器

目的：设计一个模6计数器

步骤：

- **用RTL表示计数器的功能**
- **用数字逻辑实现RTL的代码**

模6计数器：

000→001→010→011→100→101→000→...

(0→1→2→3→4→5→0...)

一个简单实例

假设：

输入端U：控制计数

当U = 1时，计数器在时钟的上升沿增加它的值。

当U = 0时，不管时钟的值如何它都保持当前值不变。

输出$V_2V_1V_0$：计数器的值

进位输出：C

状态：$S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5 \rightarrow S_0 \rightarrow \ldots$ （两个另外状态S_6和S_7）完备二级制位模式

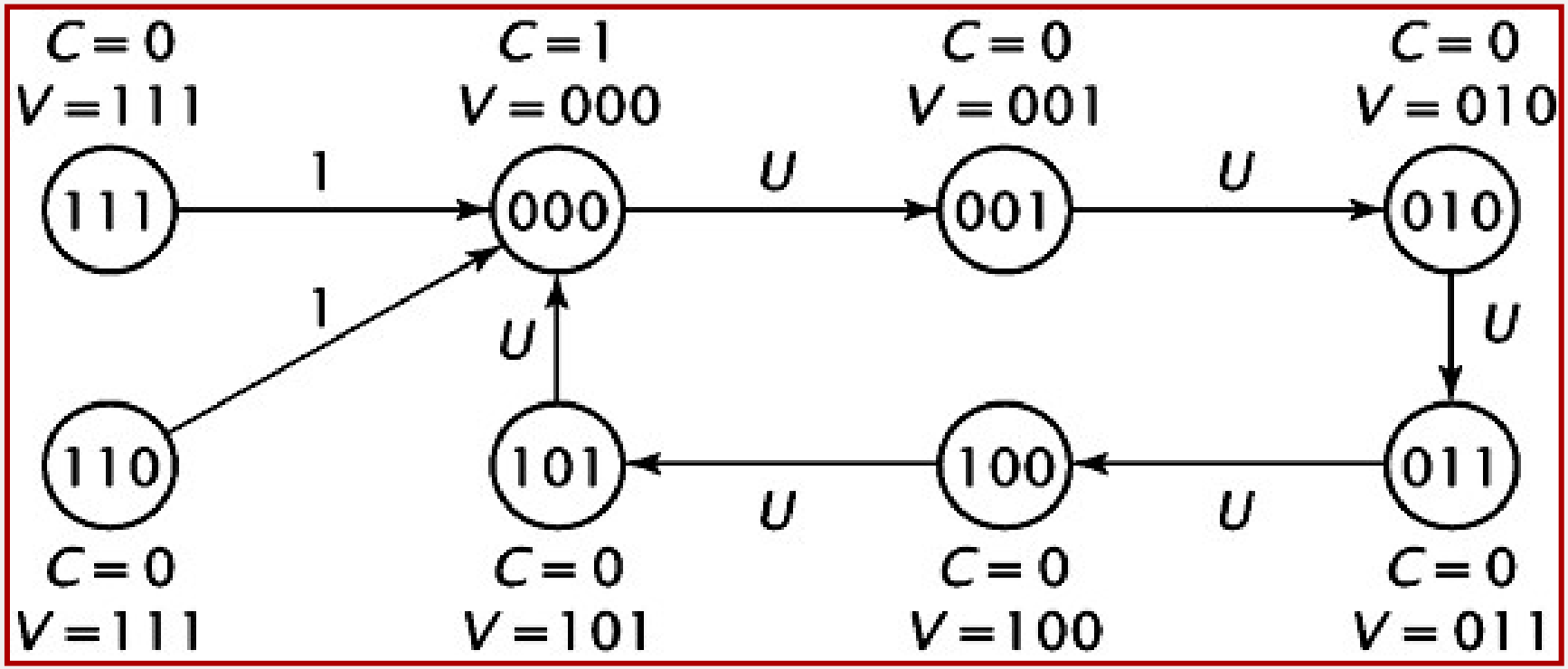
一个简单实例

模6计数器

状态表和状态图

当前状态	U	下一状态	C	$V_2V_1V_0$
S_0	0	S_0	1	000
S_0	1	S_1	0	001
S_1	0	S_1	0	001
S_1	1	S_2	0	010
S_2	0	S_2	0	010
S_2	1	S_3	0	011
S_3	0	S_3	0	011
S_3	1	S_4	0	100
S_4	0	S_4	0	100
S_4	1	S_5	0	101
S_5	0	S_5	0	101
S_5	1	S_0	1	000
S_6	X	S_0	1	111
S_7	X	S_0	1	111

一个简单实例


C = 0
V = 111
C = 1
V = 000
C = 0
V = 001
C = 0
V = 010
111
000
001
010
1
U
U
1
U
U
110
101
100
011
U
U
C = 0
V = 111
C = 0
V = 101
C = 0
V = 100
C = 0
V = 011

一个简单实例

一、用RTL表示模6计数器

$(S_0+S_1+S_2+S_3+S_4)\ U\ :\ V\leftarrow V+1,\ C\leftarrow 0$

$S_5U:\ V\leftarrow 0,\ C\leftarrow 1$

$S_6+S_7:\ V\leftarrow 0,\ C\leftarrow 1$

在条件 $(S_0+S_1+S_2+S_3+S_4+S_5)\ U'$ 下，计数器保持当前值与C值不变。

可以用两条RTL语句表示：（S_5U和S_6+S_7触发相同的微操作）

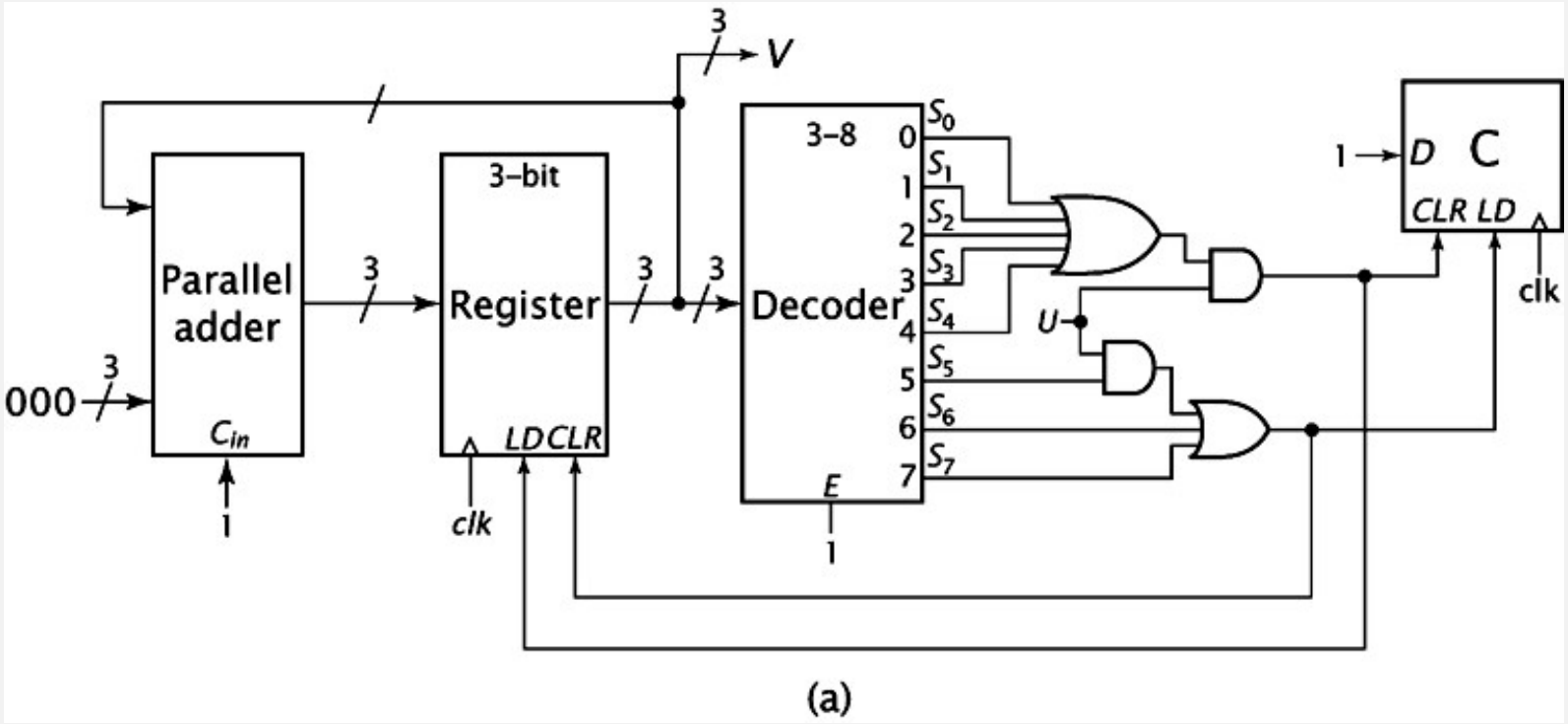
$(S_0+S_1+S_2+S_3+S_4)\ U:\ V\leftarrow V+1,\ C\leftarrow 0$

$S_5U+S_6+S_7:\ V\leftarrow 0,\ C\leftarrow 1$

一个简单实例

二、模6计数器RTL代码的两种实现

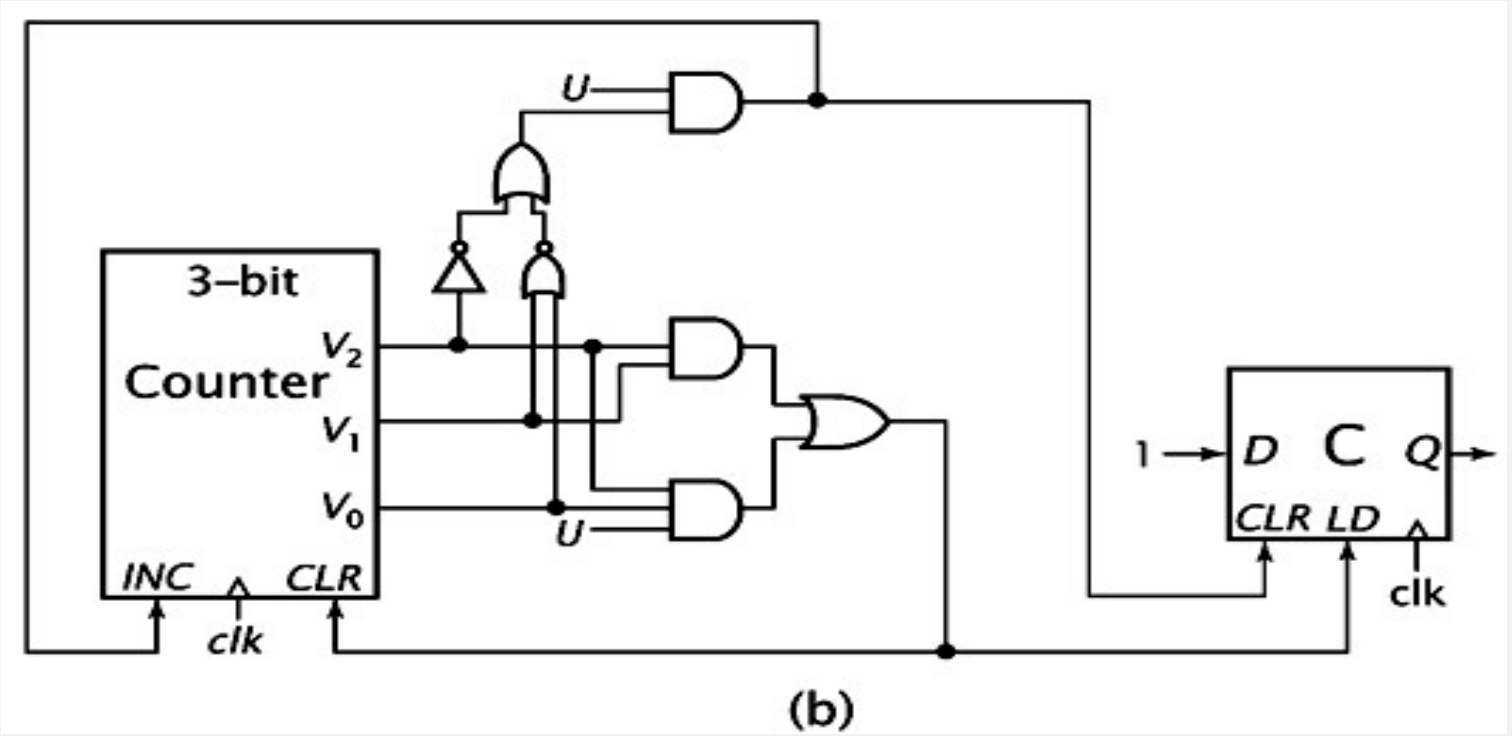
1. 用一个寄存器



(a)

一个简单实例

2. 用一个计数器（简单）



2023年春季学期

下一节：CPU

《计算机系统》课程教学组