

# CS:APP Chapter 4

## Computer Architecture

### Logic Design

### 逻辑设计



任课教师:

宿红毅    张艳    黎有琦    李秀星

原作者:

Randal E. Bryant and David R. O'Hallaron

Carnegie  
Mellon  
University

# 逻辑设计概述

# Overview of Logic Design



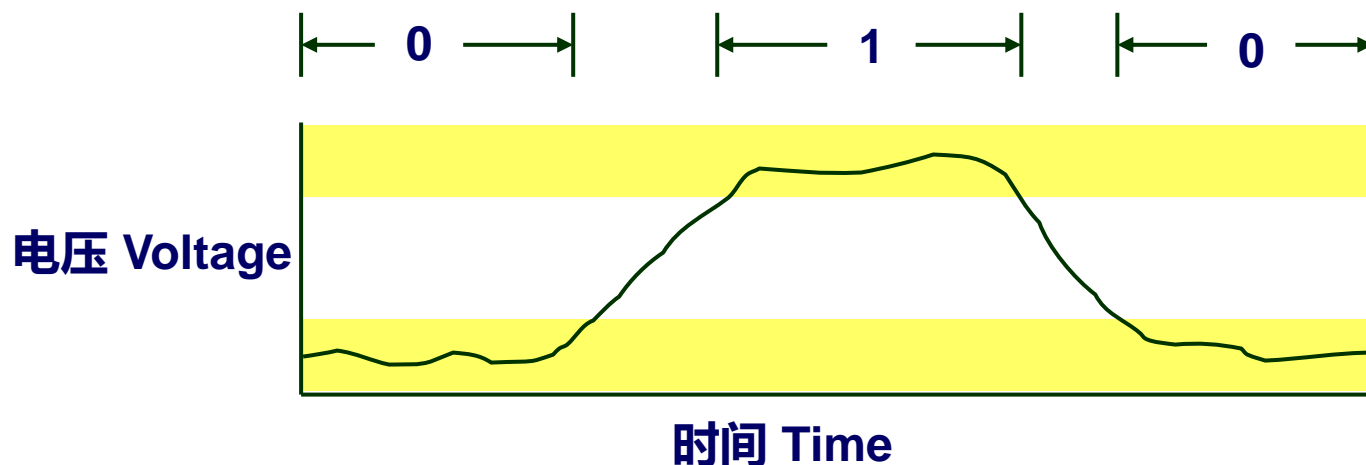
## 基础硬件需求 Fundamental Hardware Requirements

- 通信 Communication
  - 如何将值从一个地方传递到另一个地方 How to get values from one place to another
- 计算 Computation
- 存储 Storage

## 比特位是我们的朋友 Bits are Our Friends

- 一切事情都可以用值0和1进行表达 Everything expressed in terms of values 0 and 1
- 通信 Communication
  - 在电缆上传递低或高电平 Low or high voltage on wire
- 计算 Computation
  - 计算布尔函数 Compute Boolean functions
- 存储 Storage
  - 存储信息比特位 Store bits of information

# 数字信号 Digital Signals



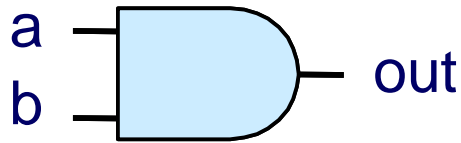
- 使用电压阈值抽取连续信号的离散值 Use voltage thresholds to extract discrete values from continuous signal
- 最简单的版本：1位信号 Simplest version: 1-bit signal
  - 要么在高电平范围（1）要么在低电平范围（0） Either high range (1) or low range (0)
  - 之间的电平值作为警戒范围 With guard range between them
- 不会受噪声或低质量电路元素较强影响 Not strongly affected by noise or low quality circuit elements
  - 可以使电路简单、小型和快速 Can make circuits simple, small, and fast

# 用逻辑门进行计算

## Computing with Logic Gates

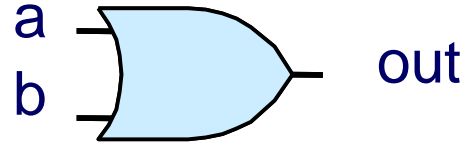


And



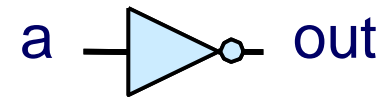
$$\text{out} = a \ \&\& \ b$$

Or



$$\text{out} = a \ || \ b$$

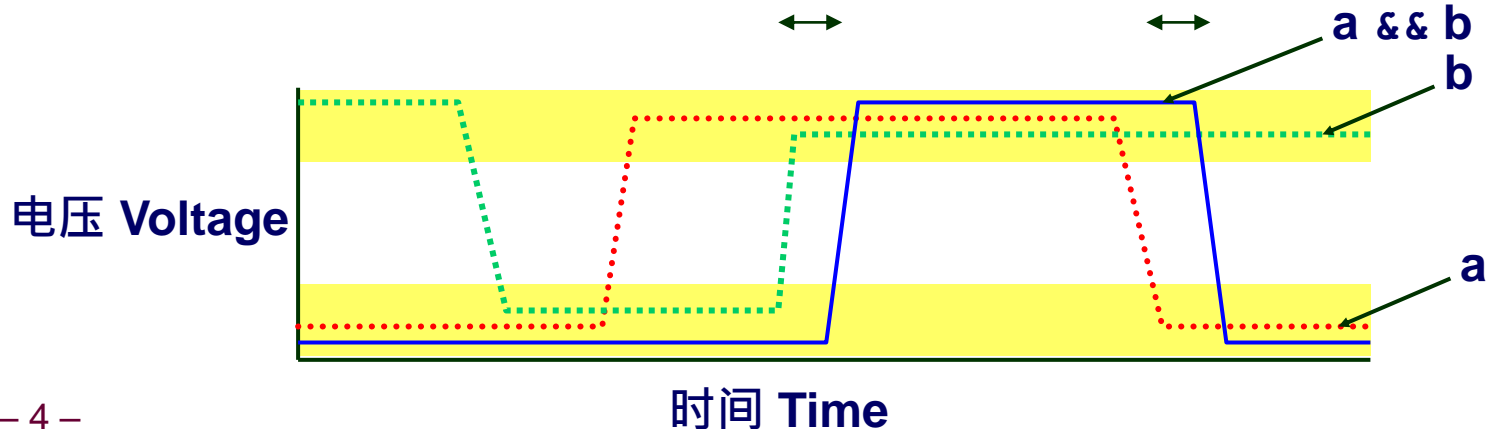
Not



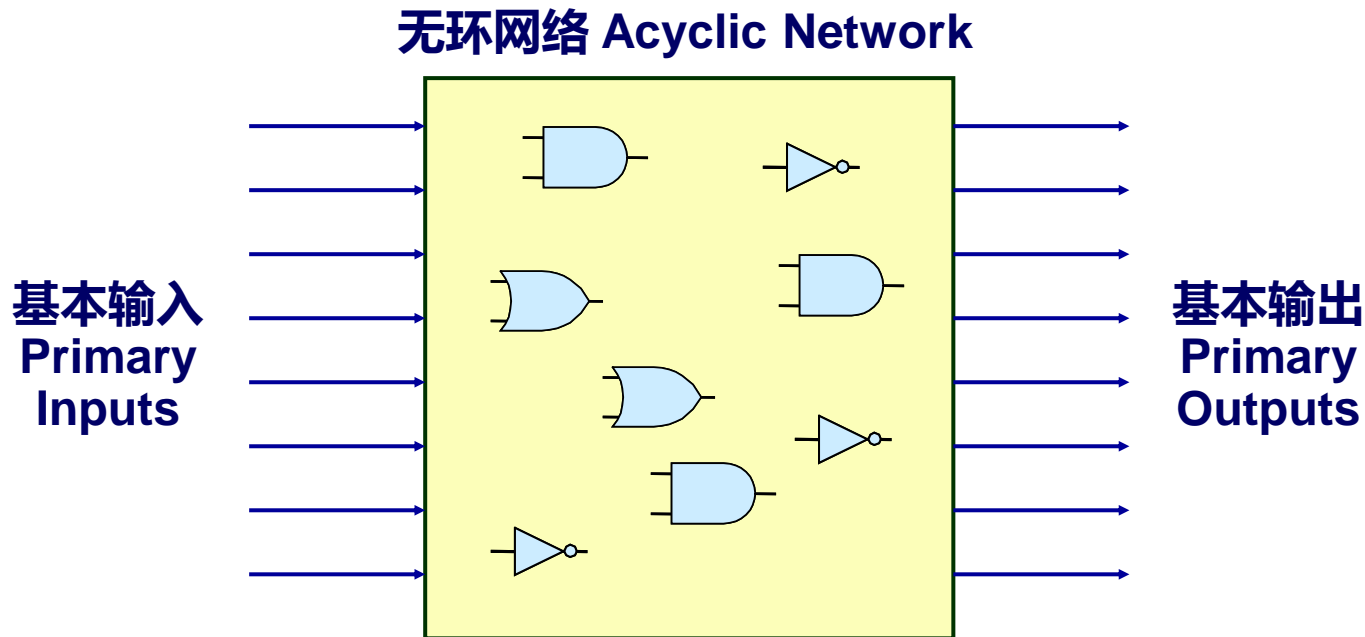
$$\text{out} = !a$$

- 输出是输入的布尔函数 Outputs are Boolean functions of inputs
- 对输入的变化连续进行响应 Respond continuously to changes in inputs
  - 有一点小的时延 With some, small delay

上升时延 Rising Delay    下降时延 Falling Delay



# 组合逻辑电路 Combinational Circuits

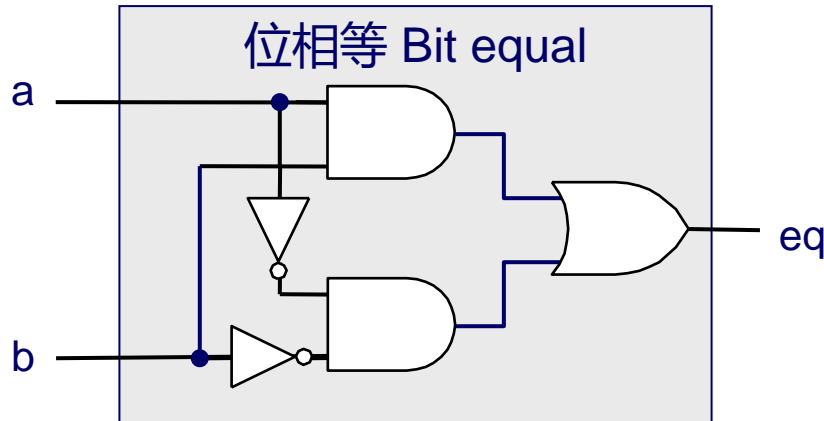


## 逻辑门的无环网络 Acyclic Network of Logic Gates

- 连续响应基本输入的变化 Continuously responds to changes on primary inputs
- 基本输出变成（一些时延后）基本输入的布尔函数 Primary outputs become (after some delay) Boolean functions of primary inputs



# 位相等 Bit Equality



HCL表达式 HCL Expression

```
bool eq = (a&&b) || (!a&&!b)
```

- 如果a和b相等产生1 Generate 1 if a and b are equal

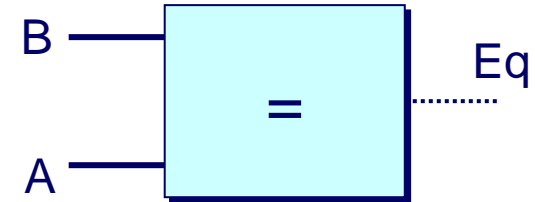
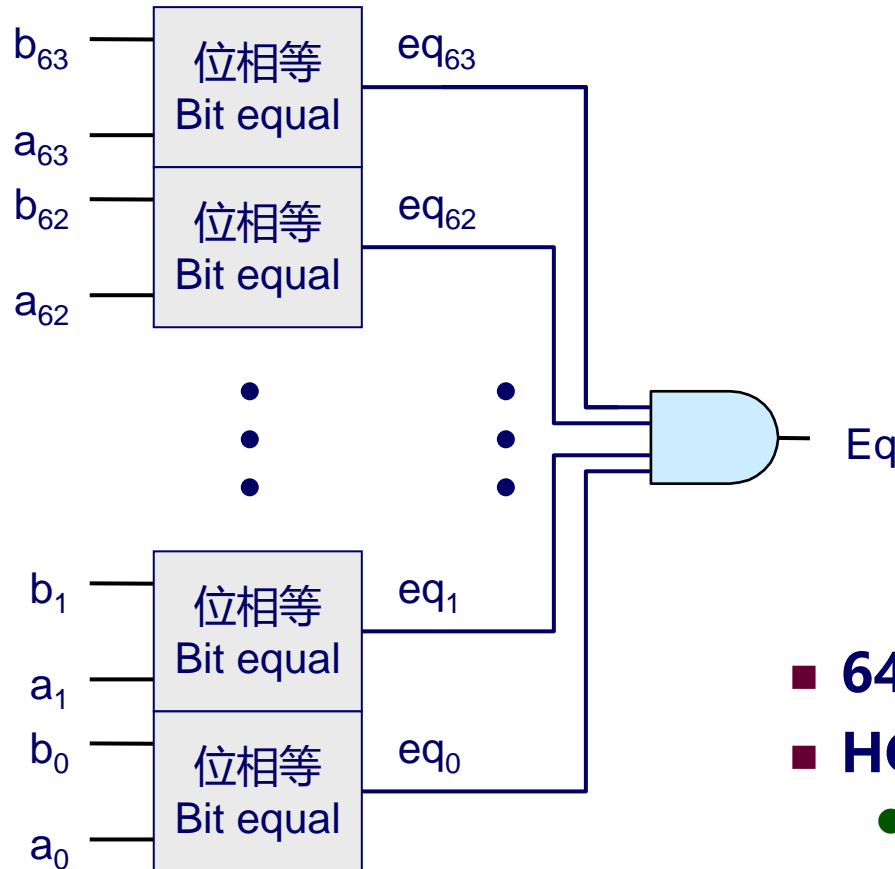
## 硬件控制语言 (HCL) Hardware Control Language (HCL)

- 非常简单的硬件描述语言 Very simple hardware description language
  - 布尔操作与C语言逻辑操作有类似的语法 Boolean operations have syntax similar to C logical operations
- 我们将用它来描述处理器的控制逻辑 We' ll use it to describe control logic for processors



# 字相等 Word Equality

## 字级表示 Word-Level Representation

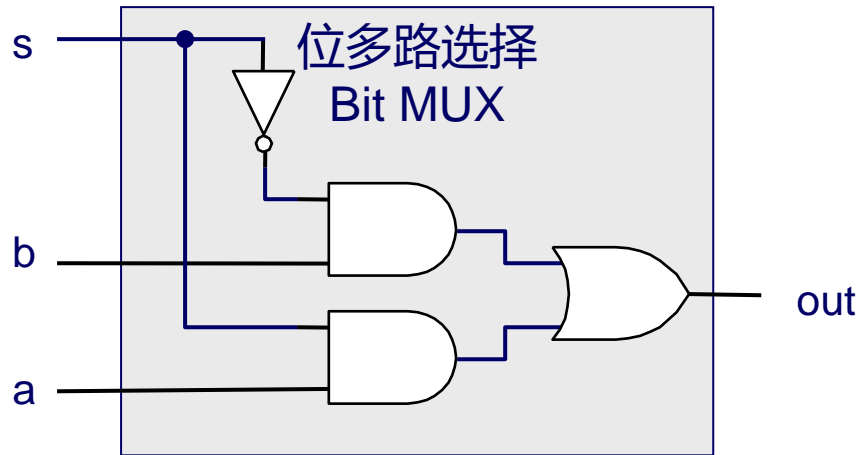


## HCL表示 HCL Representation

`bool Eq = (A == B)`

- 64位字长 64-bit word size
- HCL表示 HCL representation
  - 相等操作 Equality operation
  - 产生布尔值 Generates Boolean value

# 位级多路选择器 Bit-Level Multiplexor



HCL表达式 HCL Expression

```
bool out = (s&&a) || (!s&&b)
```

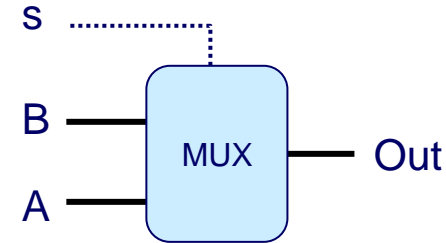
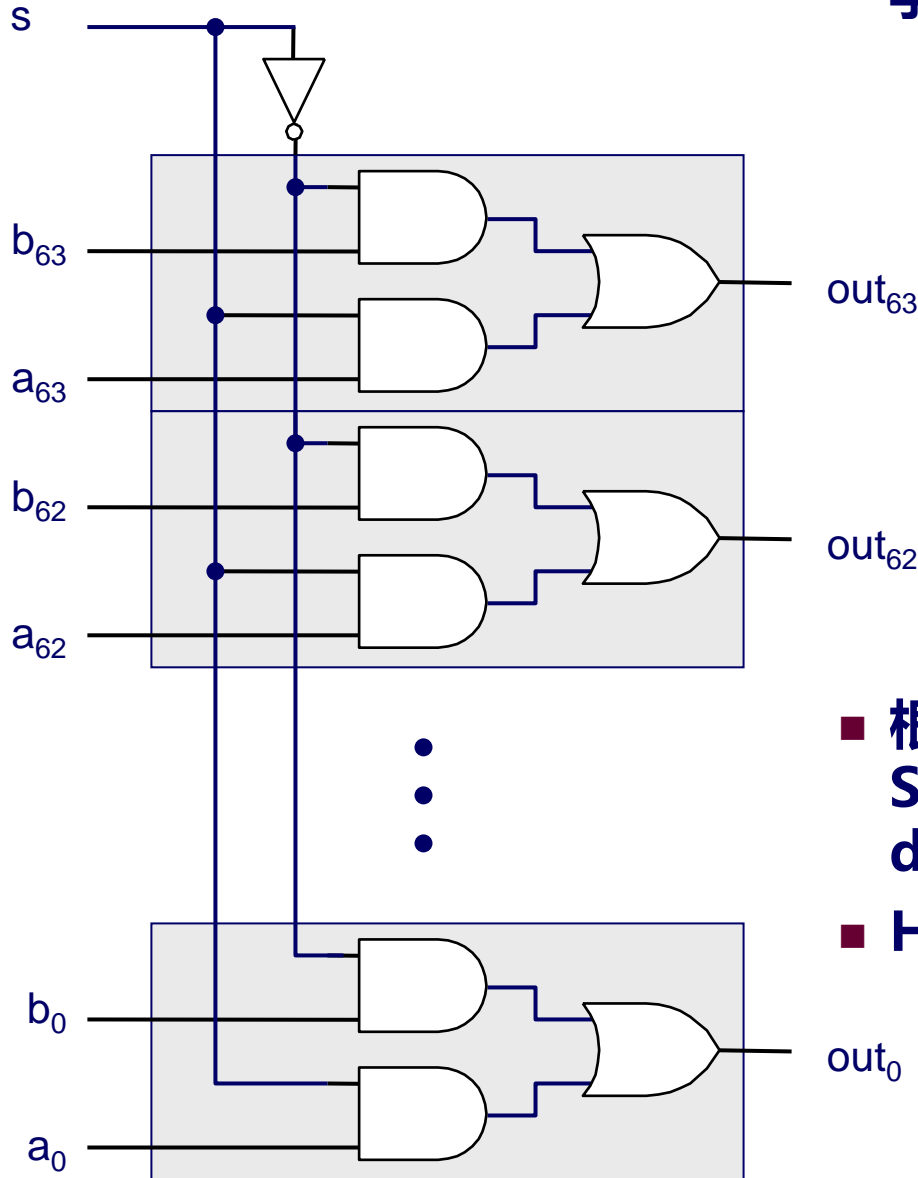
- 控制信号s Control signal s
- 数据信号a和b Data signals a and b
- 当s为1时输出a, s为0时输出为b Output a when s=1, b when s=0



# 字级多路选择器 Word Multiplexor



字级表示 Word-Level Representation



HCL表示 HCL Representation

```
int Out = [
    s : A;
    1 : B;
];
```

- 根据控制信号s选择输入字A还是B  
Select input word A or B  
depending on control signal s

- HCL表达式 HCL representation

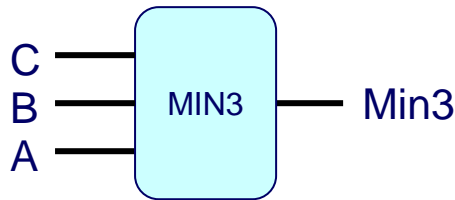
- Case表达式 Case expression
- 一系列测试：值对 Series of test : value pairs
- 第一个成功的测试作为输出值 Output value for first successful test

# HCL字级示例

## HCL Word-Level Examples



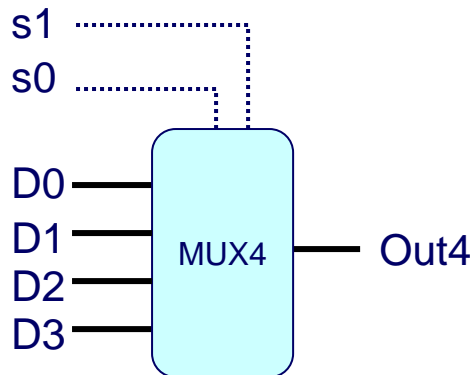
### 3个字中最小值 Minimum of 3 Words



```
int Min3 = [  
    A < B && A < C : A;  
    B < A && B < C : B;  
    1                : C;  
];
```

- 发现三个输入字中最小的  
Find minimum of three input words
- HCL case表达式 HCL case expression
- 最后的case确保匹配 Final case guarantees match

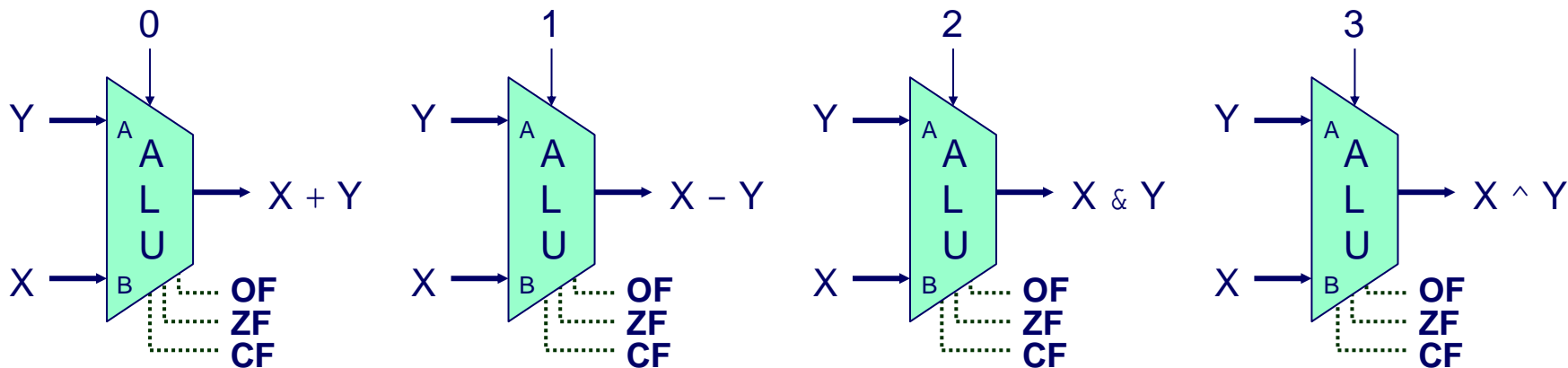
### 4路选择器 4-Way Multiplexor



```
int Out4 = [  
    !s1 && !s0 : D0;  
    !s1        : D1;  
    !s0        : D2;  
    1          : D3;  
];
```

- 根据两个控制位选择4个输入之一 Select one of 4 inputs based on two control bits
- HCL case表达式 HCL case expression
- 假设顺序匹配简化测试  
Simplify tests by assuming sequential matching

# 算术逻辑单元 Arithmetic Logic Unit

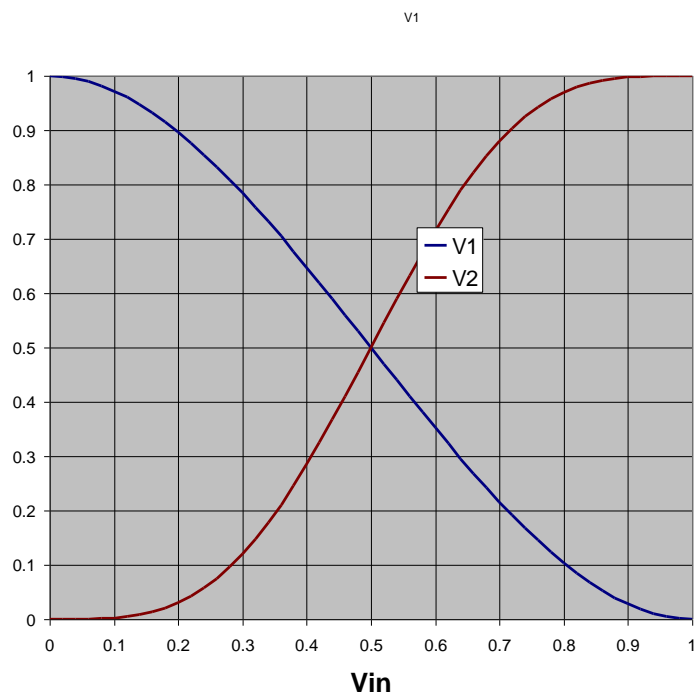
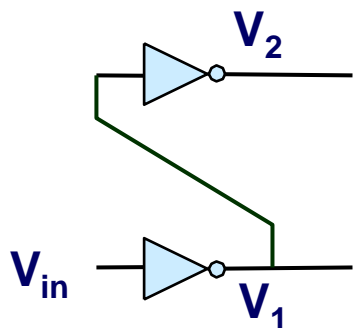
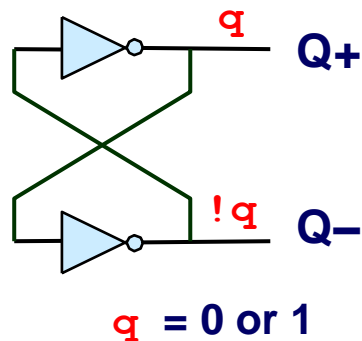


- **组合逻辑 Combinational logic**
  - 连续响应输入 Continuously responding to inputs
- **控制信号选择计算的功能 Control signal selects function computed**
  - 对应于Y86-64中的4种算术/逻辑运算 Corresponding to 4 arithmetic/logical operations in Y86-64
- **也计算条件码的值 Also computes values for condition codes**

# 存储1位 Storing 1 Bit



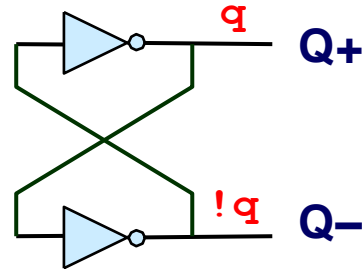
## 双稳态元件 Bistable Element



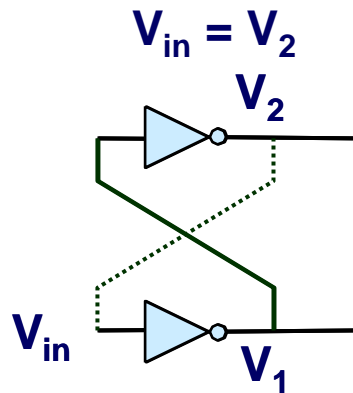
# 存储1位 (续) Storing 1 Bit (cont.)



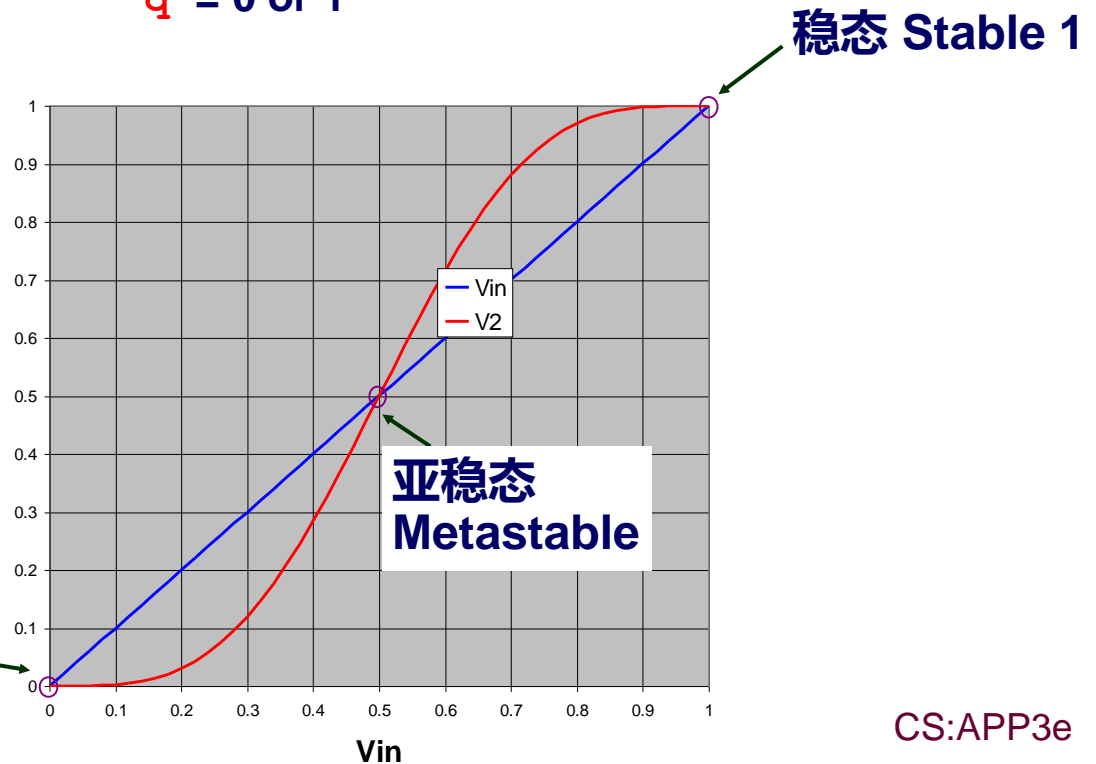
## 双稳态元件 Bistable Element



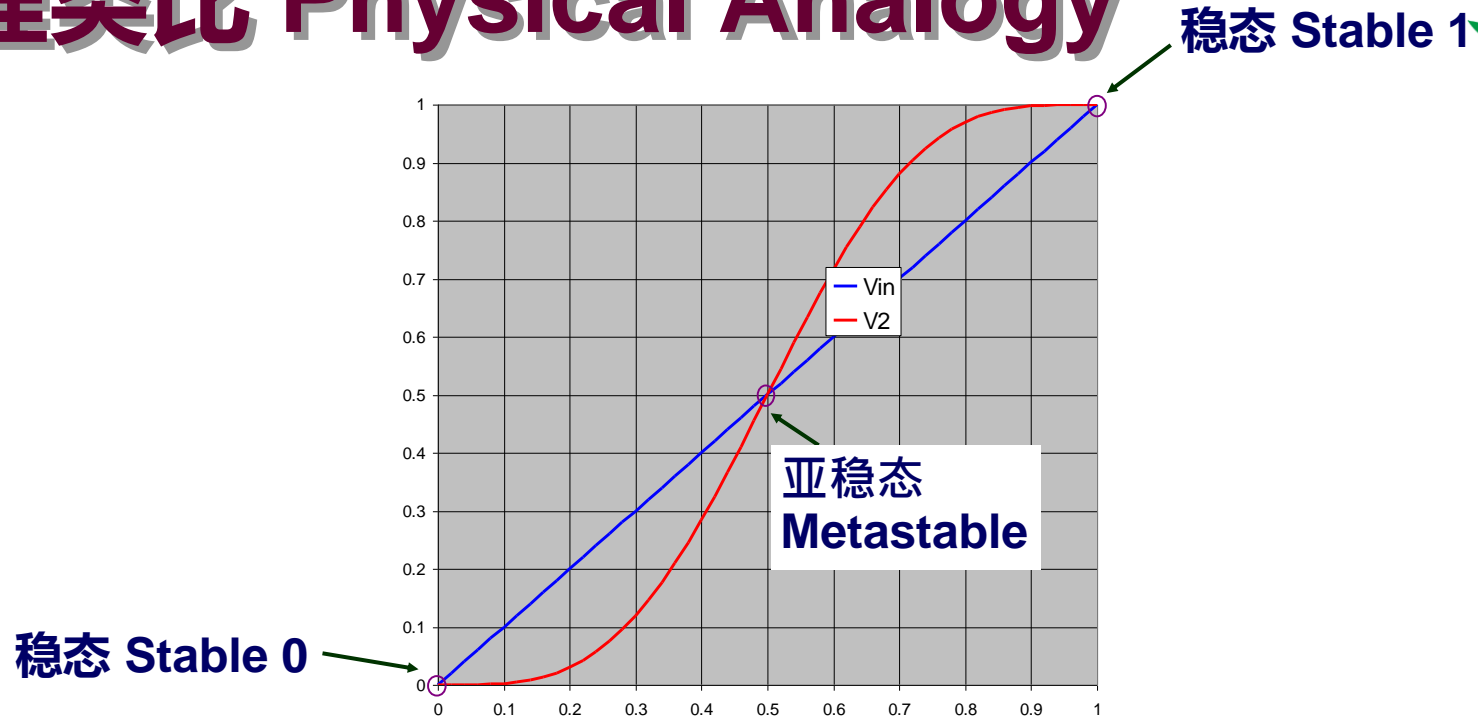
$q = 0 \text{ or } 1$



稳态 Stable 0



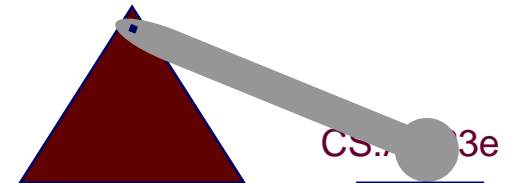
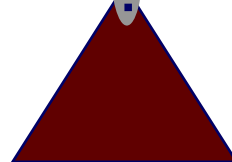
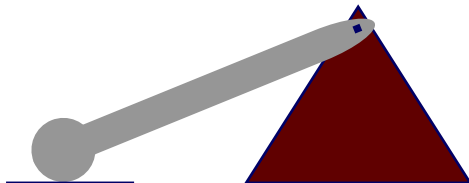
# 物理类比 Physical Analogy



稳态左 Stable left

亚稳态 Metastable

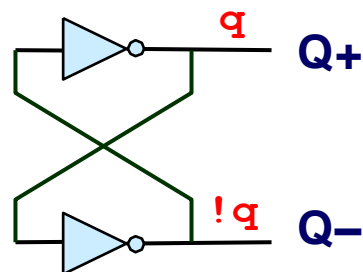
稳态右 Stable right



# 存储和访问1位 Storing and Accessing 1 Bit

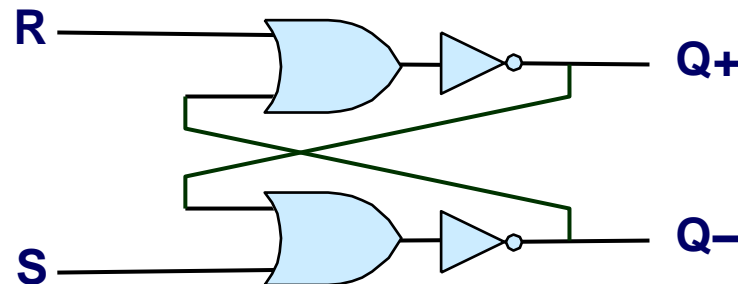


双稳态元件 Bistable Element

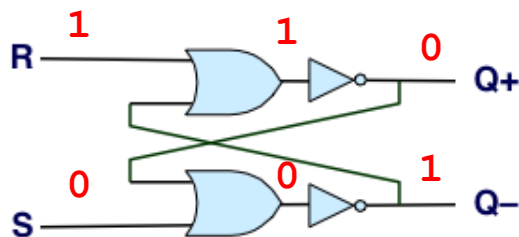


$q = 0 \text{ or } 1$

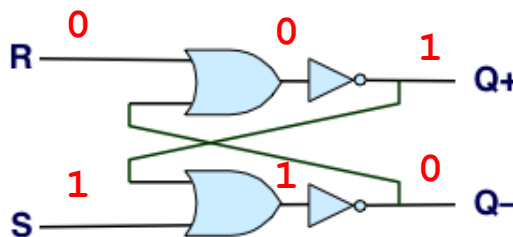
R-S锁存器 R-S Latch



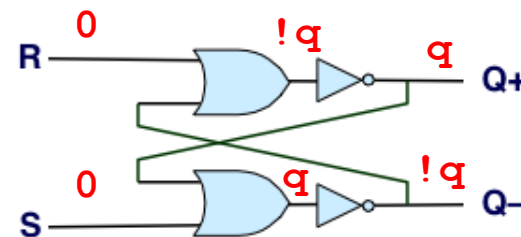
置0 Resetting



置1 Setting



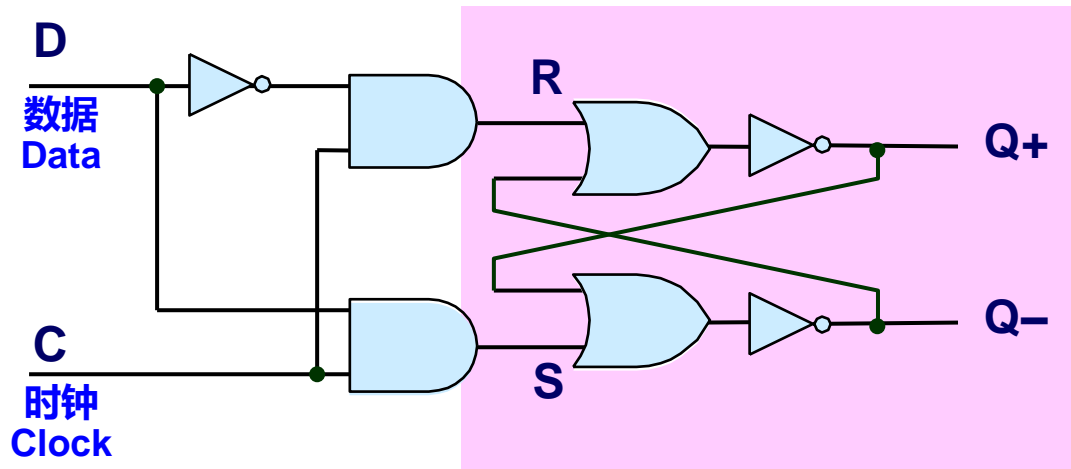
存储 Storing



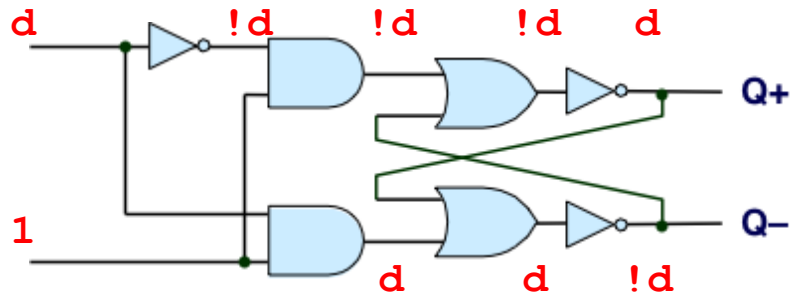


# 1位锁存器 1-Bit Latch

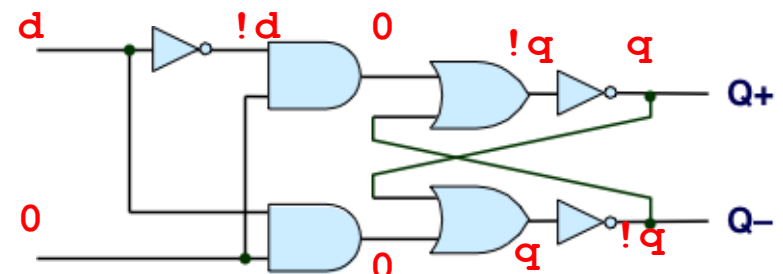
D锁存器 D Latch



锁定 Latching



存储 Storing

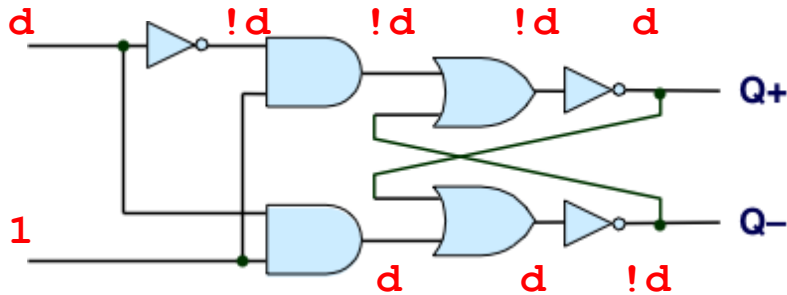




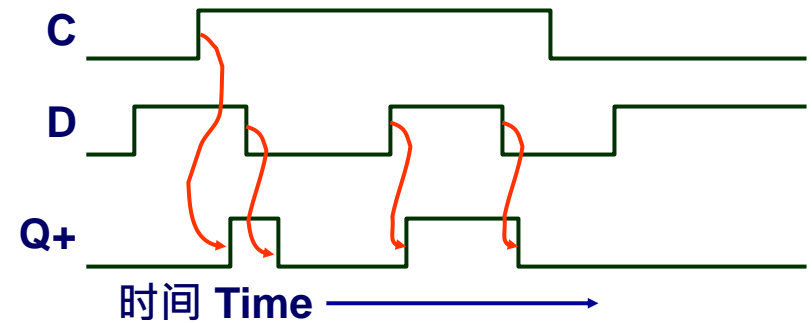
# 透明的1位锁存器 Transparent 1-Bit Latch



锁定 Latching



改变D Changing D

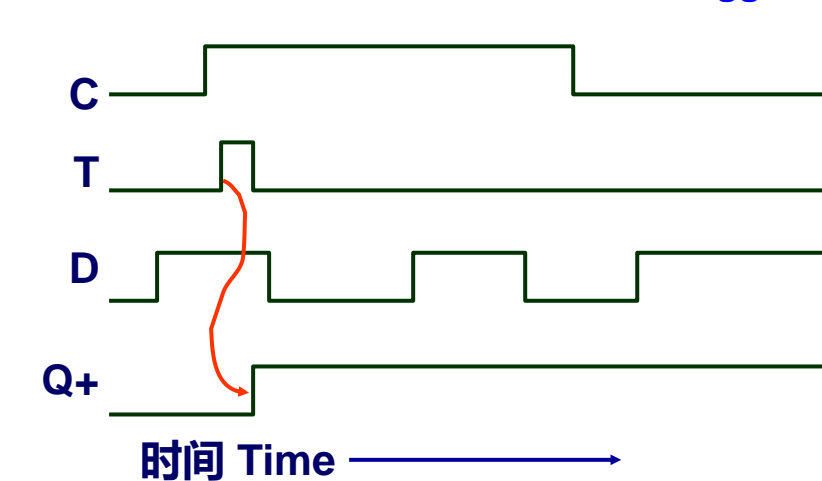
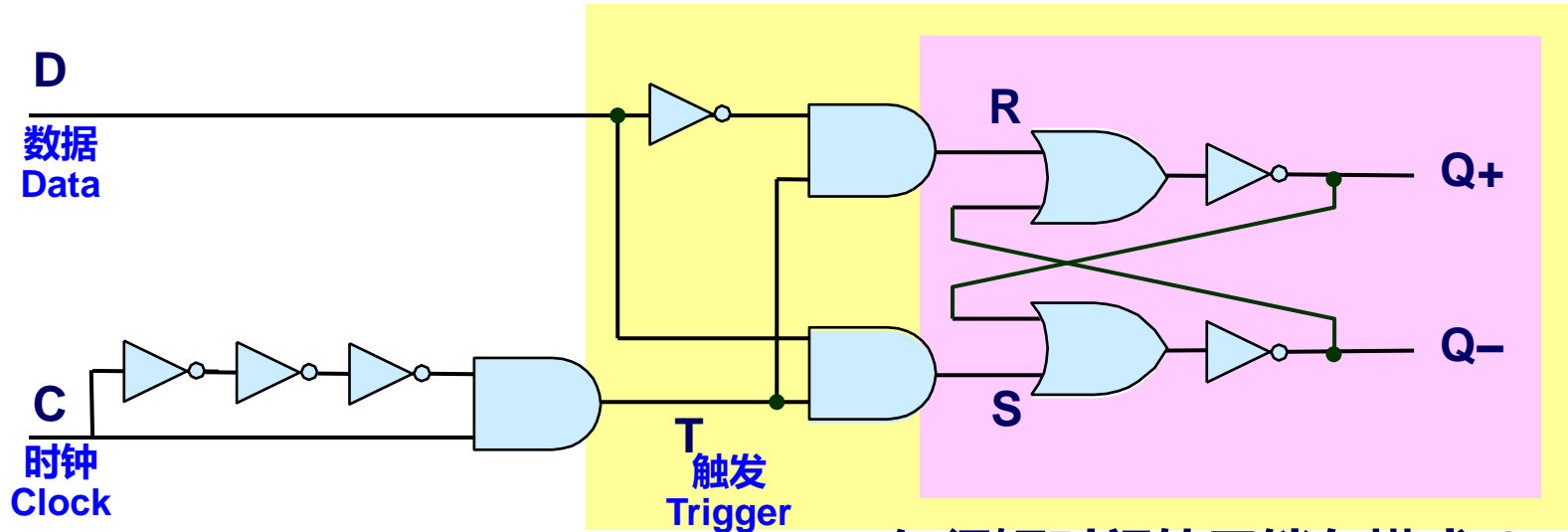


- 当在锁定模式时，组合逻辑传播从D到Q+和Q- When in latching mode, combinational propagation from D to Q+ and Q-
- 锁存的值取决于当C下降时D的值 Value latched depends on value of D as C falls



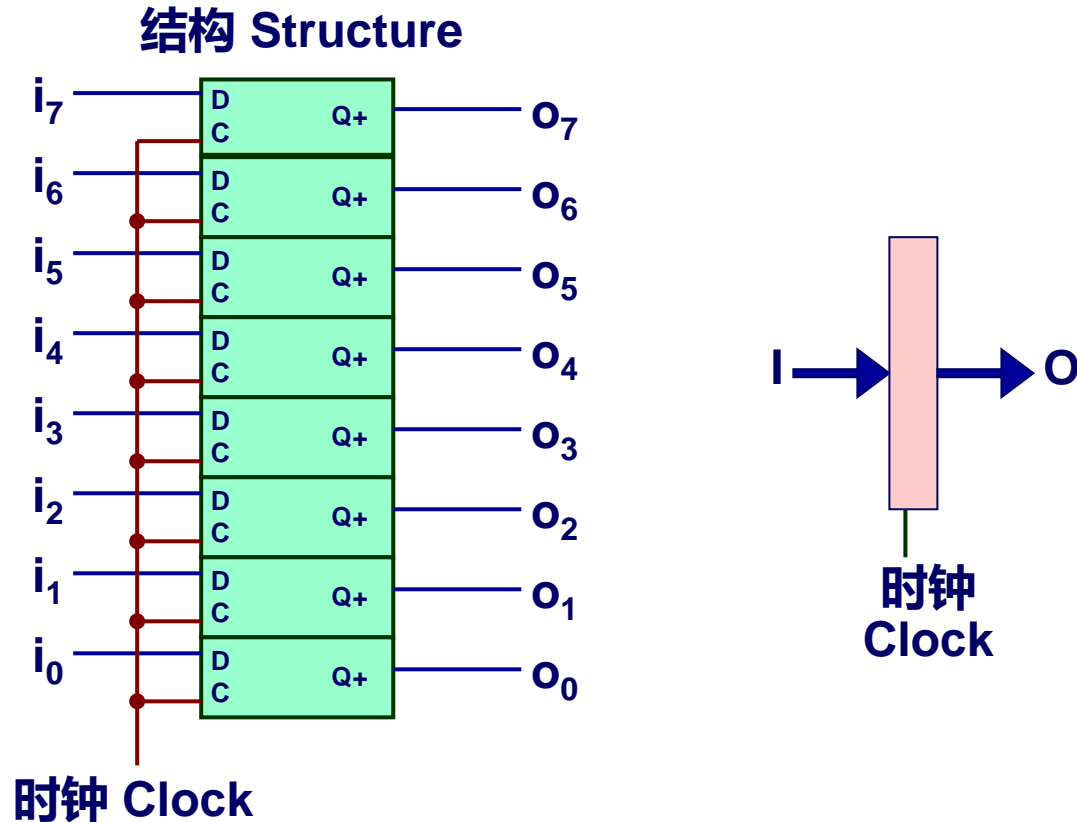
# 边沿触发的锁存器

## Edge-Triggered Latch



- 仅很短时间处于锁存模式 Only in latching mode for brief period
  - 上升时钟沿 Rising clock edge
- 锁存的值取决于当时钟上升时的数据 Value latched depends on data as clock rises
- 输出保持稳态在所有其它时间 Output remains stable at all

# 寄存器 Registers



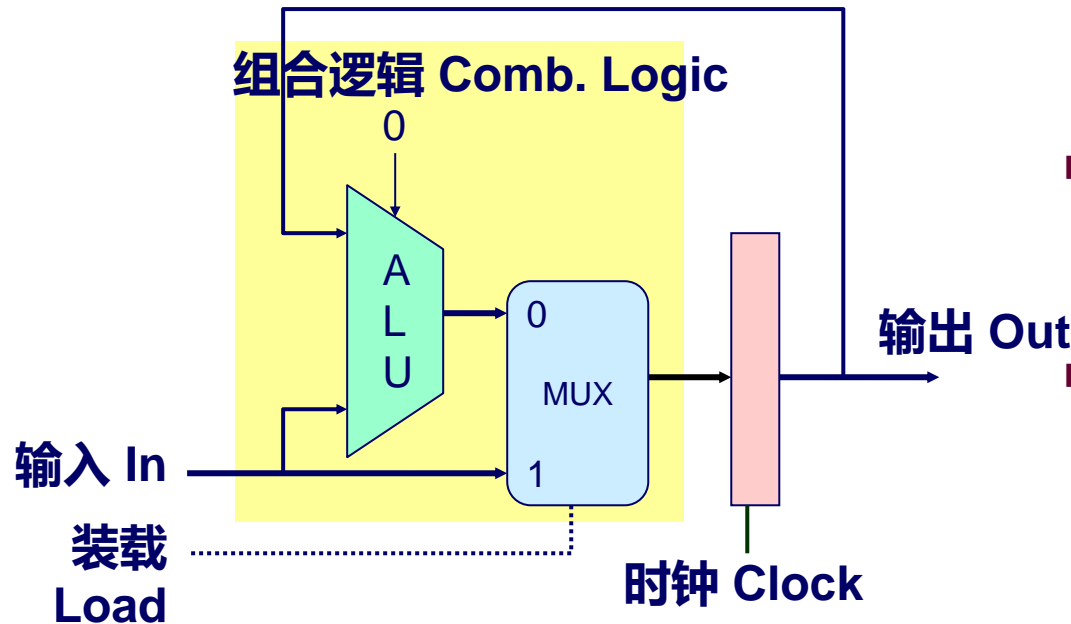
- 存储数据字 Stores word of data
  - 不同于汇编代码中看到的程序寄存器 Different from *program registers* seen in assembly code
- 边沿触发的锁存器集合 Collection of edge-triggered latches
- 在时钟上升沿装载输入 Loads input on rising edge of clock

# 寄存器操作 Register Operation

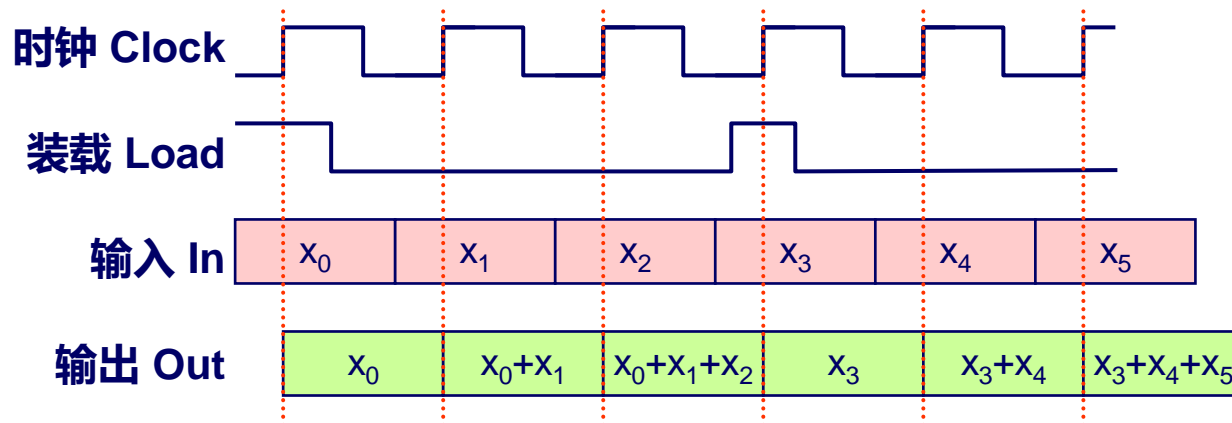


- 存储数据位 Stores data bits
- 大多数时间充当输入和输出之间的障碍 For most of time acts as barrier between input and output
- 当时钟上升时，装载输入 As clock rises, loads input

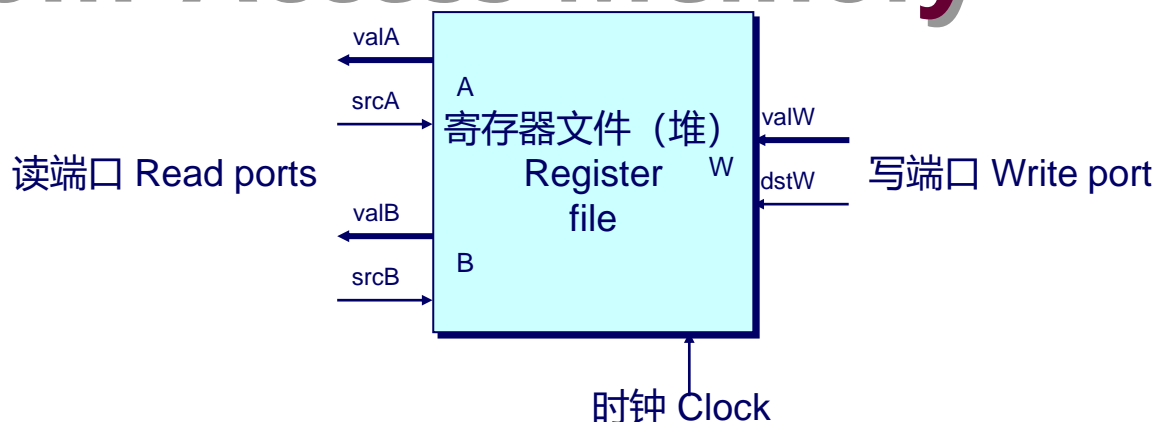
# 状态机示例 State Machine Example



- 累加器电路  
Accumulator circuit
- 每个周期装载或累加 Load or accumulate on each cycle

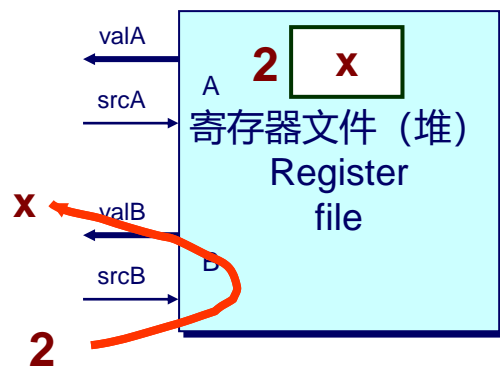


# 随机访问存储器 Random-Access Memory



- **存储多个内存字 Stores multiple words of memory**
  - **地址输入指定读或写哪个字 Address input specifies which word to read or write**
- **寄存器文件 (堆) Register file**
  - **存储程序寄存器的值 Holds values of program registers**
  - **`%rax`, `%rsp`, etc.**
  - **寄存器标识符服务作为地址 Register identifier serves as address**
    - » ID 15 (0xF) 暗含不执行读或写 ID 15 (0xF) implies no read or write performed
- **多个端口 Multiple Ports**
  - **可以一个周期读和/或写多个字 Can read and/or write multiple words in one cycle**

# 寄存器文件 (堆) 时序 Register File Timing

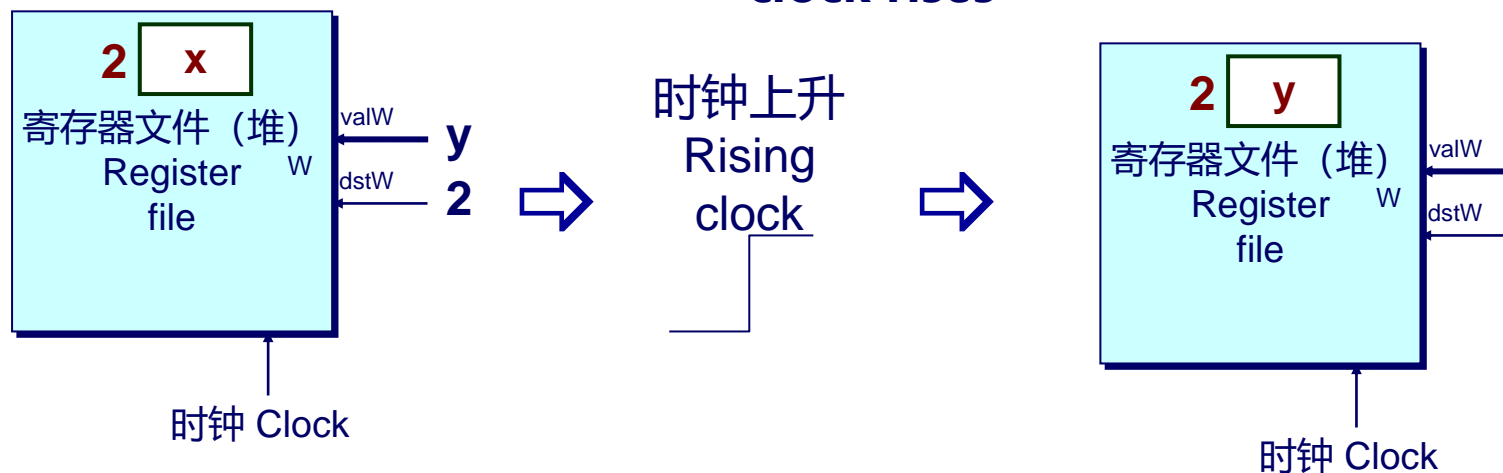


## 读 Reading

- 类似组合逻辑 Like combinational logic
- 根据输入地址产生输出数据 Output data generated based on input address
  - 一些时延后 After some delay

## 写 Writing

- 类似寄存器 Like register
- 仅在时钟上升沿更新 Update only as clock rises



# 硬件控制语言 Hardware Control Language



- 非常简单的硬件描述语言 Very simple hardware description language
- 仅可以表达有限的硬件操作 Can only express limited aspects of hardware operation
  - 我们想要探索和修改的部分 Parts we want to explore and modify

## 数据类型 Data Types

- `bool`: Boolean
  - `a, b, c, ...`
- `int`: words
  - `A, B, C, ...`
  - 不指定字长-字节还是64位字 Does not specify word size---bytes, 64-bit words, ...

## 语句 Statements

- `bool a = bool-expr ;`



# HCL操作 HCL Operations



- 按照返回值类型来分类 Classify by type of value returned

## 布尔表达式 Boolean Expressions

- 逻辑操作 Logic Operations

- `a && b, a || b, !a`

- 字比较 Word Comparisons

- `A == B, A != B, A < B, A <= B, A >= B, A > B`

- 集合成员关系 Set Membership

- `A in { B, C, D }`

» 等同于 Same as `A == B || A == C || A == D`

## 字表达式 Word Expressions

- Case表达式 Case expressions

- `[ a : A; b : B; c : C ]`

- 按顺序评估测试表达式 Evaluate test expressions `a, b, c, ...` in sequence

- 返回第一个成功测试的字表达式 Return word expression `A, B, C, ...` for first successful test

# 小结 Summary



## 计算 Computation

- 由组合逻辑执行 Performed by combinational logic
- 计算布尔函数 Computes Boolean functions
- 连续对输入变化做出反应 Continuously reacts to input changes

## 存储 Storage

- 寄存器 Registers
  - 存储单个字 Hold single words
  - 在时钟上升时装载 Loaded as clock rises
- 随机访问存储器 Random-access memories
  - 存储多个字 Hold multiple words
  - 可能有多个读或写端口 Possible multiple read or write ports
  - 当地址输入变化时读出字 Read word when address input changes
  - 当时钟上升时写入字 Write word as clock rises