数逻

好乱的笔记，不建议看...还缺斤少两的

0.1 assessment

- Theory: 70%
  – Quizzes 40% * 70% = 28%
  – Projects: 20% *70 = 14%*
  – *The final Examination: 40%*70% = 28%

- Necessary condition: the score ≥50
  Experiments: 30%

- Quizzes
  – Not regular and without notification (once every 2-3 weeks)
  – Questions are from textbook and home assignments

- **Project:**
  – The source code, source project and technical report should be submitted
  – The technical report: including the analysis and design process, the debugging process, and the simulation sequential diagram
  – Deadline: One week after the final examination

- English / Chinese

Requirements of Experiments

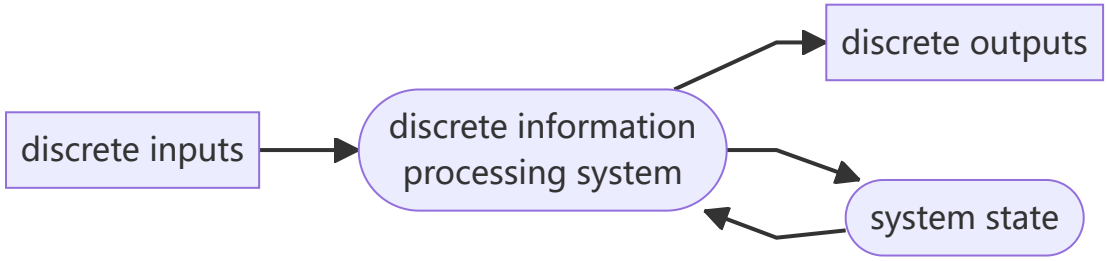
- Study Verilog HDL language by yourself

- In advance of each experiment related to Verilog HDL, every one should input and debug the Verilog source code and perform the behavior simulation

---

1 digital systems and information

1.1 digital system and computers



types of digital systems:

- Combinational Logic System
  – No state present
  – Output = Function(Input)

- Sequential System
  – State present
  – State updated at discrete times
  => **Synchronous** Sequential System
  (同步的，正确度，离散的|固定时间变化的)
  – State updated at any time
  =>**Asynchronous** Sequential System
  (异步的，效率，即时变化的)
  – State = Function (State, Input)
  – Output = Function (State)
  or Function (State, Input)

1.2 information representation

signal example – physical quantity

1.3 number system(binary, octal, hexadecimal)

positional number system(binary for example):

radix(base): r = 2
digitals: 0, 1
radix point
weight: $W_i = 2^i$

1.4 coding – decimal code

- By the binary number system, binary numbers can represent decimal numbers. However,
- Computers input and output data used by most people need to be in the decimal system. Working in the manner of the decimal system is highly important.

1.4.1 decimal codes – binary coded decimal(BCD)

Decimal	8,4,2,1	Excess 3	8,4,-2,-1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

redundancy 冗余码
（in the form of extra bits, can be incorporated into
binary code words to detect and correct errors.）（lg: Parity)

ASCII (American Standard Code for Information Interchange):

$0 \sim 9 : 30_{16} — 39_{16}$

Upper case $A \sim Z : 41_{16} – 5A_{16}$

Lower case $a \sim z : 61_{16} – 7A_{16}$

---

2 combinational logic circuits

实实在在的电路——做不做得出来、成本面积速度

logical operations: AND OR NOT

truth tables

logic gates: relays(继电器)—vacuum tubes(真空管)—transistors(晶体管)

gate delay: the output change does not occur instantaneously

NAND: logically complete

AOI: AND-OR-INVERT

boolean equations(布尔代数式): lg A((BC') + D)

(Truth tables are unique; expressions and logic diagrams are not. This gives flexibility in implementing functions.)

order: ( ) > NOT > AND >OR $\mathbf{X + YZ = (X + Y)(X + Z)}$

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	1-4 Existence of 0 and 1
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	5-6 Idempotence
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	7-8 Existence of complement
9.	$\overline{\overline{X}} = X$			9 Involution
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutative
12.	$(X + Y) + Z = X + (Y + Z)$	13.	$(XY)Z = X(YZ)$	Associative
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributive
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's

- $x \cdot y + \bar{x} \cdot y = y$ $(x + y)(\bar{x} + y) = y$ **Minimization**
- $x + x \cdot y = x$ $x \cdot (x + y) = x$ **Absorption**
- $x + \bar{x} \cdot y = x + y$ $x \cdot (\bar{x} + y) = x \cdot y$ **Simplification**
- $x \cdot y + \bar{x} \cdot z + y \cdot z = x \cdot y + \bar{x} \cdot z$ **Consensus**
  $(x + y) \cdot (\bar{x} + z) \cdot (y + z) = (x + y) \cdot (\bar{x} + z)$
- $\overline{x + y} = \bar{x} \cdot \bar{y}$ $\overline{x \cdot y} = \bar{x} + \bar{y}$ **DeMorgan's Laws**

The **dual(对偶)** of an algebraic expression is obtained by interchanging + and · , and replacing 0s by 1s and 1 by 0.
self-dual(自对偶) H = AB + BC + AC

complementing functions(反函数)

- difference: 对偶-变量不取反

2.0.2 [canonical forms]: SOM/POM

sum of minterms(SOM):
minterms: XYZ(m_7), XYZ'(m_6), X'Y'Z'(m_0)…下标使组合为1

$f = m_3 + m_2 + m_0 = \sum_m(0, 2, 3)$

product of maxterms(POM):
maxterms: X+Y+Z(M_0), X+Y+Z'(M_1), X'+Y'+Z'(M_7)…下标使组合为0

$f = M_2 \cdot M_3 = \Pi_M(2, 3)$

$M_i = m_i'$ $m_i = M_i'$
We can implement any function by **"ORing" the minterms** corresponding to **"1"** entries in the function table. These are called the minterms of the minterms of the function.(**"ANDing",maxterms,"0"**)

$F(x, y, z) = \sum_m(1, 3, 5, 7)$:

$F'(x, y, z) = \sum_m(0, 2, 4, 6) = \Pi_M(1, 3, 5, 7)$

2.0.3 [standard forms]: SOP/POS

standard sum of products(SOP): F = Y' + X'YZ' + XY
standard product of sums(POS): F = X(Y' + Z)(X + Y + Z')

2.0.4 other gate types:

Buffer (F = X)

the 3-StateBuffer (when EN = 0, whatever IN is, OUT = Hi-Z)

Transmission Gate (can be regarded as a switch)

Hi-Z: 数字电路常见术语，指的是电路的一种输出状态，既不是高电平也不是低电平，如果高阻态再输入下一级电路的话，对下级电路无任何影响，和没接一样，如果用万用表测的话有可能是高电平也有可能是低电平，随它后面接的东西定。

2.0.5 circuit optimization:

cost criteria（成本标准）:

• Literal cost (L)
• Gate input cost (G)
• Gate input cost with NOTs (GN)

2.0.6 boolean function optimization:

Karnaugh Maps (K-map) , dimension of squares

2.0.7 Systematic Simplification:

Prime Implicant (质蕴含项)

Essential Prime Implicant(基本质蕴含项)

---

3 Combinational Logic Design

3.1 implementation technology and logic

combinational circuits(组合电路):

m Boolean inputs, n Boolean outputs, and n switching functions, each mapping the 2^m input combinations to an output such that the current output depends only on the current input values

Technology Parameters(工艺参数):

- Fan-in – the number of inputs available on a gate
- Fan-out – the number of standard loads driven by a gate output
- Logic Levels – the signal value ranges for 1 and 0 on the inputs and 1 and 0 on the outputs (see Figure 1-1)
- Noise Margin – the maximum external noise voltage superimposed on a normal input value that will not cause an undesirable change in the circuit output
- Cost for a gate - a measure of the contribution by the gate to the cost of the integrated circuit
- Propagation Delay(传输延迟) – The time required for a change in the value of a signal to propagate from an input to an output
  - transport delay(输送延迟)
  - inertial delay(惯性延迟):rejection time
- Power Dissipation – the amount of power drawn from the power supply and consumed by the gate

Design procedure:

specification-formulation-optimization-technology mapping-verification

beginning hierarchical(分层的) design

Top-down, Bottom-up

mapping to NAND / NOR gates

3.2 combinational logic:

function block

Rudimentary (基本的) logic functions

Enabling function (使能)

Decoding and decoder (解码器) generate Minterms
lg: two 1-to-2-Line Decoders and four AND gates can generate one 2-to-4-Line Decoder
n-to-2^n-Line Decoder: even 2 x $2^{k/2}$ odd $2^{(k+1)/2} + 2^{(k-1)/2}$

Encoding and encoder (编码器)

Multiplexer (选择器)

> We will refer to the combination of AND gates and OR gates as an m × 2 AND-OR, where m is the number of AND gates and 2 is the number of inputs to the AND gates.

3.3 Arithmetic Functions

3.3.1 Iterative combinational circuits

- Arithmetic functions
  • Operate on binary vectors
  • Use the same subfunction in each bit position
- Can design functional block for subfunction and repeat to obtain functional block for overall function
- *Cell* - subfunction block
- Iterative array - an array of interconnected cells
- An iterative array can be in a single dimension (1D) or multiple dimensions

3.3.2 Binary adders

3.3.2.1 Half-Adder

a half adder adds two bits to produce a two-bit sum (a sum bit S and a carry bit C)

$S = X \oplus Y = XY' + X'Y = (X+Y)(X'+Y')$

$C = XY$

3.3.2.2 Full-Adder

includes a carry-in bit from lower stages

$S = X \oplus Y \oplus Z = XY'Z' + X'YZ' + X'Y'Z + XYZ$

$C = XY + XZ + YZ = XY + (X \oplus Y)Z$

3.3.2.3 Ripple Carry Adder (纹波进位加法器)

a n-bit Ripple Carry Adder made from n 1-bit Full Adders

One problem with the addition of binary numbers is the length of **time** to **propagate** the ripple carry from the least significant bit to the most significant bit.

3.3.2.4 Carry-Look-Ahead Adder (超前进位加法器)

G_i (generate): A_iB_i

P_i (propagate): $A_i \oplus B_i$

$S_i = P_i \oplus C_i$

$C_{i+1} = G_i + P_iC_i$

C_i can be removed from the cells and used to derive a set of carry equations spanning multiple cells

This could be extended to more than four bits, but in practice, due to limited gate fan-in, such extension is not feasible.
Solution:

$G_{0-3} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$

$P_{0-3} = P_3P_2P_1P_0$

$C_4 = G_{0-3} + P_{0-3}C_0$

3.3.3 Binary subtraction

- Subtract the subtrahend N from the minuend M
- If no end borrow occurs, then $M \geq N$, and the result is a non-negative number and correct.
- If an end borrow occurs, then $N > M$ and the difference $M - N + 2^n$ is subtracted from 2^n, and a minus sign is appended to the result.
- The subtraction, $2^n - N$, is taking the 2's complement of N

3.3.3.1 Complements

Diminished Radix Complement of N:

(r - 1)'s complement for radix r, define as $(r^n - 1) - N$

Radix Complement:

r's complement for radix r: $r^n - N$

3.3.3.2 Subtraction with 2's complement

For n-digit, unsigned numbers M and N, find M - N in base 2:

- Add the 2's complement of the subtrahend N to the minuend M:
  $M + (2^n - N) = M - N + 2^n$
- If $M > N$, the sum produces end carry r^n which is discarded; from above, M - N remains.

- If M < N, the sum does not produce an end carry and, from above, is equal to 2^n - (N - M), the 2's complement of (N - M).
- To obtain the result - (N – M) , take the 2's complement of the sum and place a - to its left.

3.3.4 Binary adder-subtractors

signed integers representations: signed-magnitude, signed-complement

overflow detection: $V = C_n \oplus C_{n-1}$

3.3.5 Binary multiplication

3.3.6 Other arithmetic functions

convenient to design the functional blocks by *contraction* (收缩) — removal

of redunancy from circuit to which input fixing has been applied

functions:

- incrementing
- decrementing
- multiplication by constant
- division by constant
- zero fill and extension

---

4 Sequential Circuits

4.1 Storage Elements and Sequential Circuit Analysis

4.1.1 A Sequential circuit contains:

- Storage elements: Latches (锁存器) or Flip-Flops (触发器)
- Combinational Logic

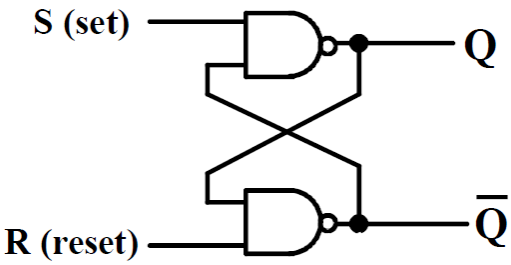
4.1.2 Types of sequential circuits:

- depends on the times at which: storage elements observe their inputs and change their state
- **Synchronous** (同步的) : Behavior defined from knowledge of its signals at discrete instances of time; Storage elements observe inputs and can change state only in relation to a timing signal (clock pulses from a clock)
- Asynchronous (异步的) : Behavior defined from knowledge of inputs an any instant of time and the order in continuous time in which inputs change; If clock just regarded as another input, all circuits are asynchronous!; Nevertheless, the synchronous abstraction makes complex designs tractable!

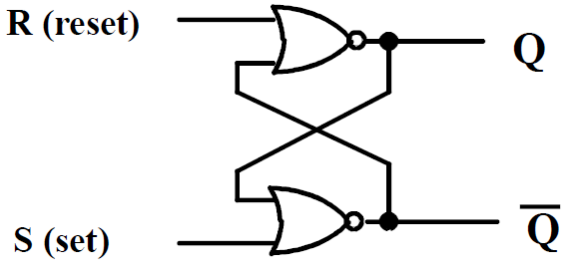
Discrete event simulation (离散事件模拟/仿真)

4.1.3 Latches (锁存器)

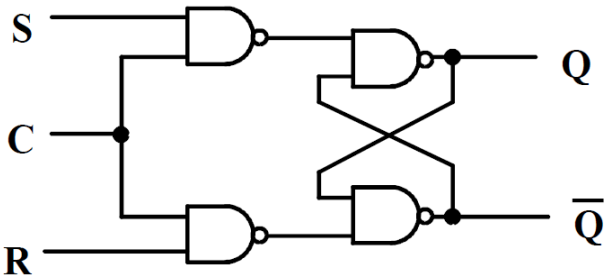
Basic $\overline{S} - \overline{R}$ *Latch* : two NAND gates



Basic $S - R$ *Latch*: two NOR gates

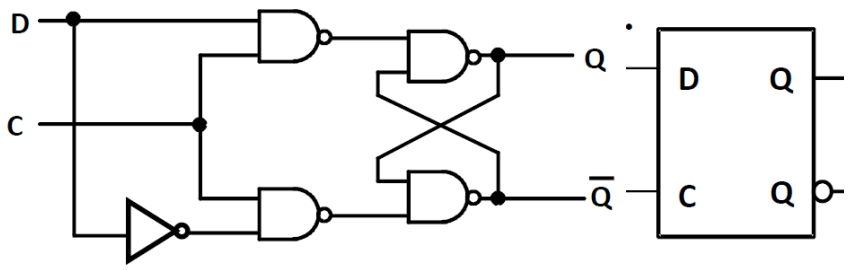


Clock $S - R$ *Latch*: adding two NAND to the *Basic* $\overline{S} - \overline{R}$ *Latch*



C	S	R	Q(t+1)
0	X	X	No Change
1	0	0	No Change
1	0	1	0: Clear Q
1	1	0	1: Set Q
1	1	1	Indeterminate

D Latch : adding an inverter to the S-R Latch (No 'indeterminate' states)



D锁存器的缺点：存在空翻现象——如果D锁存器直接用在时序电路中作为状态存储元件，当使能控制信号有效时，会导致该元件内部的状态值随时多次改变，而不是保持所需的原始状态值

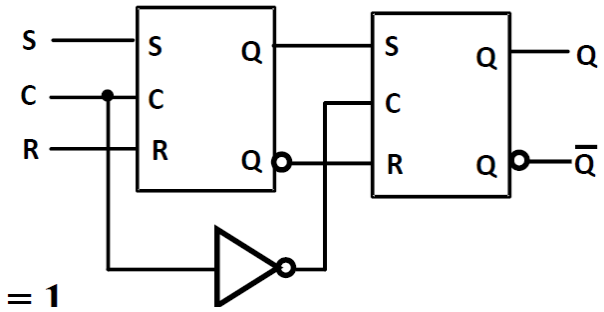
解决办法：用两个锁存器，主锁存器在脉冲控制下**接收**输入数据，从锁存器在脉冲结束后**改变并保持**状态。

4.1.4 Flipflop (触发器)

$S-R\ Master-Slave\ Flip-Flop$:

在有脉冲（高电平）时，修改第一个锁存器的值，保持第二个锁存器的值；在没有脉冲（低电平）时候保持第一个锁存器的值，修改第二个锁存器的值，更新触发器的状态。

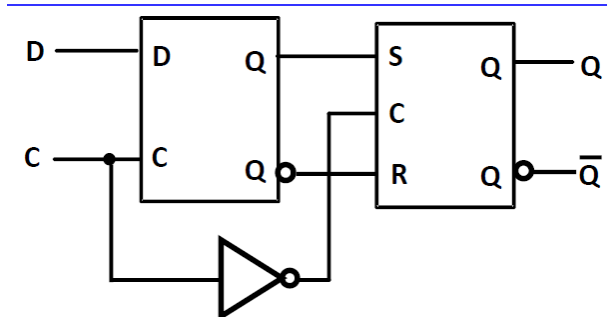
The input is observed by the first latch with C = 1
The output is changed by the second latch with C = 0
(Problem: delay, S/R are permitted to change while C = 1)



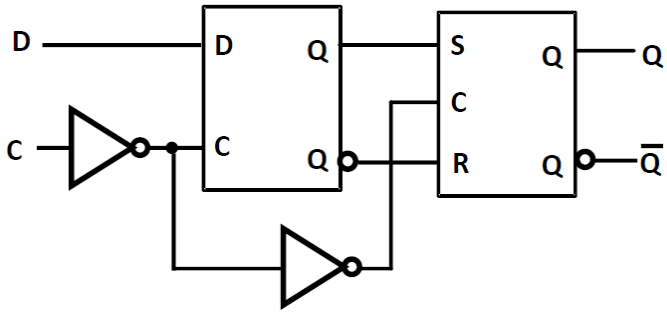
$Edge-Triggered\ D\ Flip-Flop$:

仅在时钟的边缘触发，即在特定时刻仅接受一个输入。
（正边沿）关注上升沿前后，上升沿前一刻，主锁存器可写，从锁存器只读；上升沿后一刻，主锁存器只读，从锁存器只写，且写入的是主锁存器存储的值——换句话来说，存的是上升沿前一刻写入主锁存器的内容。

The delay of the S-R master-slave flip-flop can be avoided since the 1s-catching behavior is not present with D replacing S and R inputs.
The change of the D flip-flop output is associated with the negative edge at the end of the pulse.
(also called a *negative-edge triggered flip-flop*)

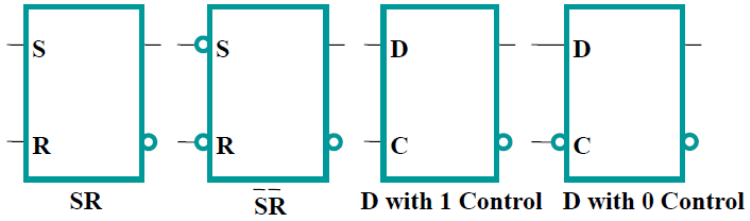


standard flip-flop (Positive-Edge Triggered D Flip-Flop) * not actual used one



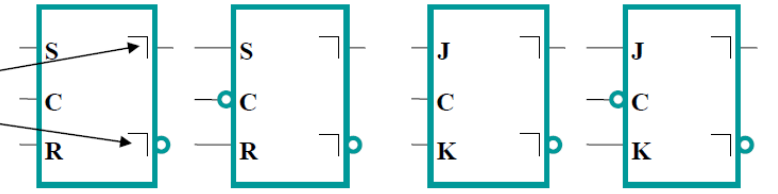
STANDARD SYMBOLS:

Standard Symbols for Storage Elements



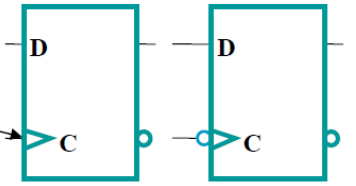
(a) Latches

- **Master-Slave: Postponed output indicators**



(b) Master-Slave Flip-Flops

- **Edge-Triggered: Dynamic indicator**



(c) Edge-Triggered Flip-Flops

4.1.5 Direct inputs:

At power up or at reset, all or part of a sequential circuit usually is initialized to a known state before it begins operation.

This initialization is often done outside of the clocked behavior of the circuit.

direct R and S inputs that control the state of the latches within the flip-flops are used for this initialization.
(S=1,Q=1;R=1,Q=0)

4.1.6 Sequential Circuit Analysis

General Model:

output = Boolean function of state (and input).

state = Boolean function of **last** state and input.

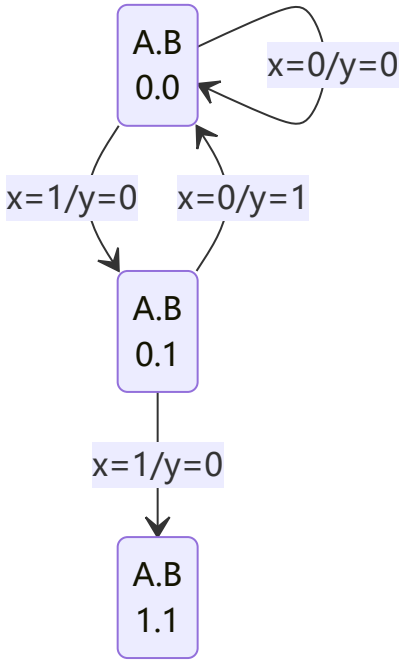
State Table:

present state, input, next state, output

(当前状态的分析和定义有很多不同情况)

State Diagrams:

- A *circle* with the *state name* in it for each state
- A *directed arc* from the Present State to the Next State for each *state transition*
- A *label* on each directed arc with the *Input values* which causes the state transition, and a label on each circle with the *output value* produced, or on each directed arc with the output value produced.

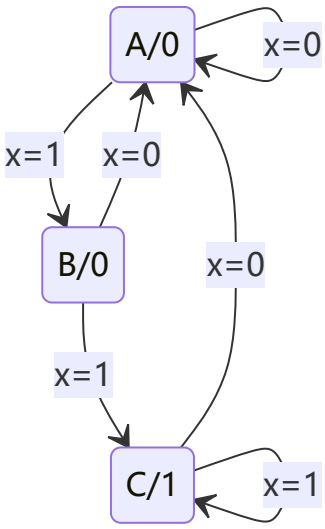


Equivalent State Definitions:

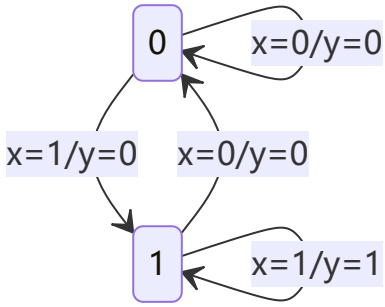
Two states are equivalent if their response for each possible **input** sequence is an identical output sequence.
Alternatively, two states are equivalent if their **outputs** produced for each input
symbol is identical and their **next states** for each input symbol are the same or equivalent.

Finite State Machines (FSMs) : Sequential Circuits or Sequential Machines

Moore Model Diagrams: maps states to outputs



Mealy Model Diagrams: maps inputs and state to outputs



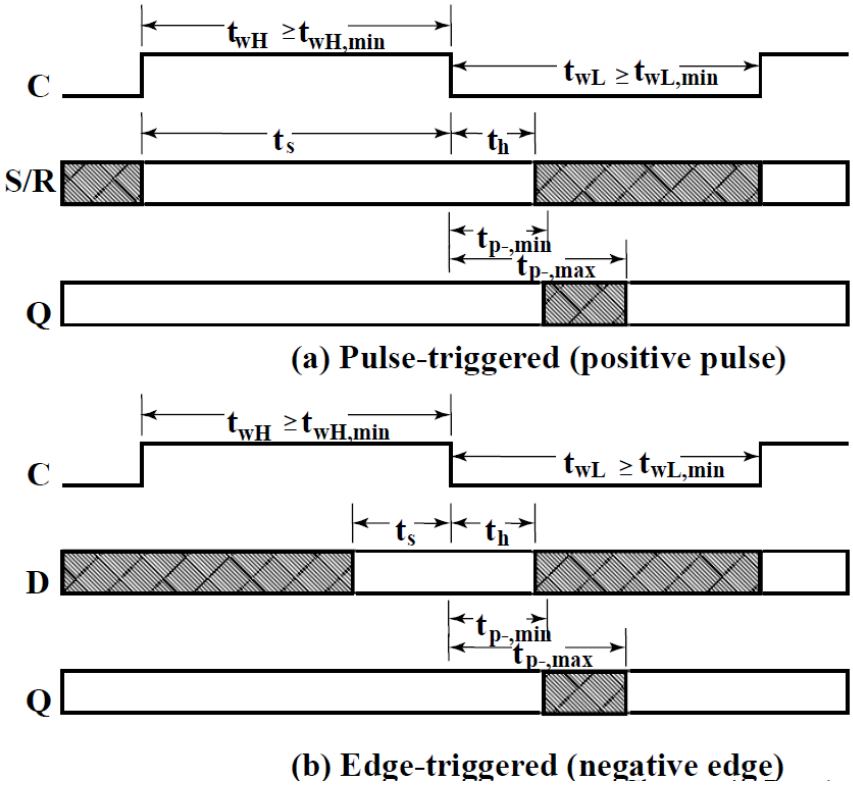
Mixed Moore and Mealy Outputs

4.1.7 Flip-Flop Parameters (参数)

t	mean
t_s	setup time
t_h	hold time
t_w	clock pulse width
t_{px}	propagation delay
t_{pd}	max(t_{PHL} , t_{PLH})

Flip-Flop Timing Parameters

- t_s - setup time
- t_h - hold time
- t_w - clock pulse width
- t_{px} - propagation delay
  - t_{PHL} - High-to-Low
  - t_{PLH} - Low-to-High
  - t_{pd} - max (t_{PHL}, t_{PLH})



4.1.8 Circuit and System Level Timing

$t_p = t_{stack} + (t_{pd,FF} + t_{pd,COMB} + t_s)$, t_{stack} is greater than or equal to zero

4.2 Sequential Circuit Design

4.2.1 The Design Procedure

- Specification
- Formulation - Obtain a state diagram or state table
- State Assignment - Assign binary codes to the states
- Flip-Flop Input Equation Determination - Select flip-flop types and derive flip-flop equation from the next state entries in the table
- Output Equation Determination - Derive output equations from output entries in the table
- Optimization - Optimize the equations
- Technology Mapping - Find circuit from equations and map to flip-flop and gate technology
- Verification - Verify correctness of final design

4.2.2 Sequence Recognizer Procedure

- Begin in an **initial state** in which NONE of the initial portion of the sequence has occurred (typically “reset” state).
- Add a state that recognizes that the first symbol has occurred.
- Add states that recognize each successive symbol occurring.
- The final state represents the **input sequence** (possibly less the final input value) occurrence.
- Add state transition arcs which specify what happens when a symbol **not** in the proper sequence has occurred.
- Add other arcs on non-sequence inputs which transition to states that represent the input subsequence that has occurred.

4.2.3 Formulation: Find State Table

Present State	Next State x=0 x=1	Output x=0 x=1
A	A B	0 0
B	…	…
C	…	…

Mealy Model

Present State	Next State x=0 x=1	Output
A	A B	0
B	…	…
C	…	…

Moore Model

Mealy Model侧重记录输入前的状态；Moore Model侧重输入后的状态

`Moore is More`

4.2.4 状态表的简化

等效状态：设状态S1和S2是完全确定状态表中的两个状态,如果对于所有可能的输入序列，分别从状态S1和状态S2出发，所得到的输出响应序列完全相同，则状态S1和S2是等效的，记作(S1, S2), 或者说，状态S1和S2是等效对。
等效状态可以合并。这里“所有可能的输入序列”是指长度和结构是任意的，它包含无穷多位，且有无穷多种组合。

输出相同 & (次态相同 | 次态交错 | 次态循环)

观察法、隐含表法（处于循环链中是等效的:互相依赖的是等效的）

4.2.5 状态分配

任务：决定编码的长度，寻找一种最佳的或接近最佳的状态分配方案

原则：

- **在相同输入条件下具有相同次态的现态，应尽可能分配相邻的二进制代码**
- 在相邻输入条件，同一现态的次态应尽可能分配相邻的二进制代码

- 输出完全相同的现态应尽可能分配相邻的二进制代码
- 最小化状态表中出现次数最多的状态或初始状态应分配逻辑0

4.2.6 Other Flip-Flop Types

J-K Flip-flop:
J = K = 1 is allowed, and the flip-flop changes to the opposite state
has some “1’s catching” behavior as S-R flip-flop
if the master changes to the wrong state, it will be passed to the slave

T Flip-flop:
T = 0, no change to state; T = 1, changes to opposite state
J-K flip-flop with J = K = T
“1’s catching”
cannot be initialized to a known state using the T input (Reset essential)

4.2.7 Basic Flip-flop Descriptors

Characteristic Table:

D	Q(t+1)	Operation
0	0	Reset
1	1	Set

Characteristic Equation:

$Q(t+1) = D$
$Q(t+1) = T \otimes Q$
$Q(t+1) = S + R' Q,\ S \cdot R = 0$
$Q(t+1) = J Q' + K' Q$ **(AB+A’ C+BC = AB + A’ C)**

Excitation Table:

Q(t+1)	D	Operation
0	0	Reset
1	1	Set

4.3 State Machine Design

---

5 Digital Hardware Implementation

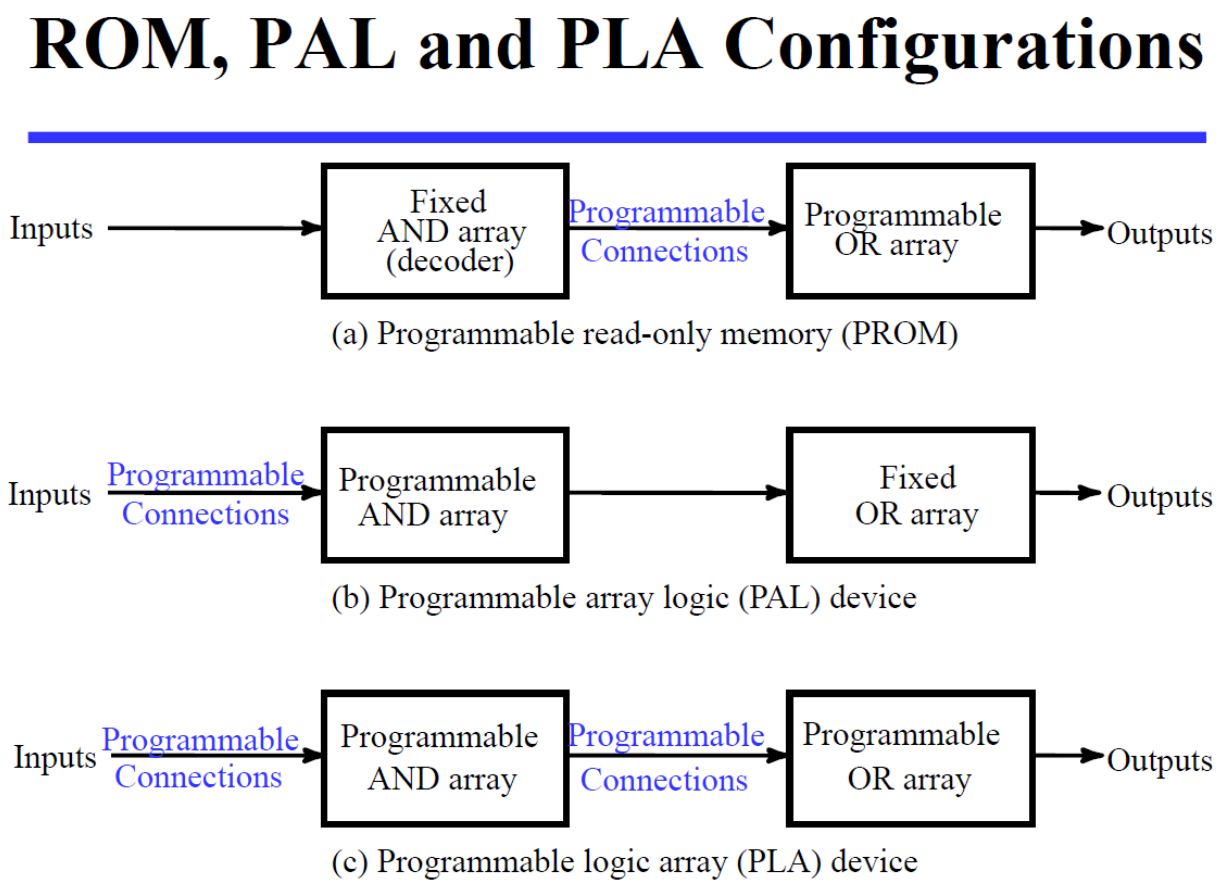
5.1 The Design Space

5.2 Programmable Implementation Technologies

▶ Why Programmable Logic
▶ Programmable Logic-More Advantages
▶ Programming Technologies

5.2.1 Programmable Configuration

- **Read Only Memory (ROM):** a fixed array of AND gates and a programmable array of OR gates
- **Programmable Array Logic (PAL)**: a programmable array of AND gates feeding a fixed array of OR gates
- **Programmable Logic Array (PLA)**: a programmable array of AND gates feeding a programmable array of OR gates
- **Complex Programmable Logic Device (CPLD) / Field-Programmable Gate Array (FPGA)**: complex enough to be called “architectures”



---

6 Registers and Register Transfers

6.1 Registers, Microoperations and Implemenetations

Register: a collection of binary storage elements

Register Design Models:

- Add predefined combinational circuits to registers
- Design individual cells using the state diagram / state table model and combine them into a register (Output is usually the state variable)

Register Storage and Load Enable: Load = 1, load the values on the data inputs; Load = 0, store the values in the register.

- **Registers with Clock Gating**: the $\overline{Load}$ signal enables the clock signal to pass through if 0 and prevents the clock signal from passing through if 1. $Gated\ Clock\ = Clock + \overline{Load}$.
  Problem: Clock Skew of gated clocks with respect to clock or each other

- **Registers with Load-Controlled Feedback**: Run the clock continuously, and selectively use a load control to change the register contents.
  (Example) 2-bit register: Load = 0, loads register contents; = 1, input values.

Register Transfer Operations:
the movement and processing of data stored in registers
three basic components: set of registers, operations, control of operations
Elementary Operations (Microoperations): load, count, shift, add, bitwise"OR", etc

Register Notation:

- Letters and numbers – denotes a register (ex. R2, PC, IR)
- Parentheses ( ) – denotes a range of register bits (ex. R1(1), PC(7:0), PC(L))
- Arrow (←) – denotes data transfer (ex. R1 ←R2, PC(L) ← R0)
- Comma – separates parallel operations
- Brackets [ ] – Specifies a memory address (ex. R0 ← M[AR], R3 ← M[PC])

Conditional Transfer: If (K1 = 1) then (R2←R1) is shortened to **K1: (R2←R1)**

Control Expressions:
appears to the left of the operation and is separated from it by a colon
specify the logical condition for the operation to occur
(Example) $\overline{X}\ K1 : R1 \leftarrow R1 + R2$

Microoperations:

- Logical Groupings:
  - Transfer - move data from one register to another
  - Arithmetic - perform arithmetic on data in register (+ - * /)
  - Logic - manipulate data or use bitwise logical operations ($\vee\ \wedge\ \otimes\ \bar{}$)
  - Shift - shift data in registers

  NOTE:
  "+" means "OR" in K1 + K2; and means "plus" In R1 ← R1 + R3

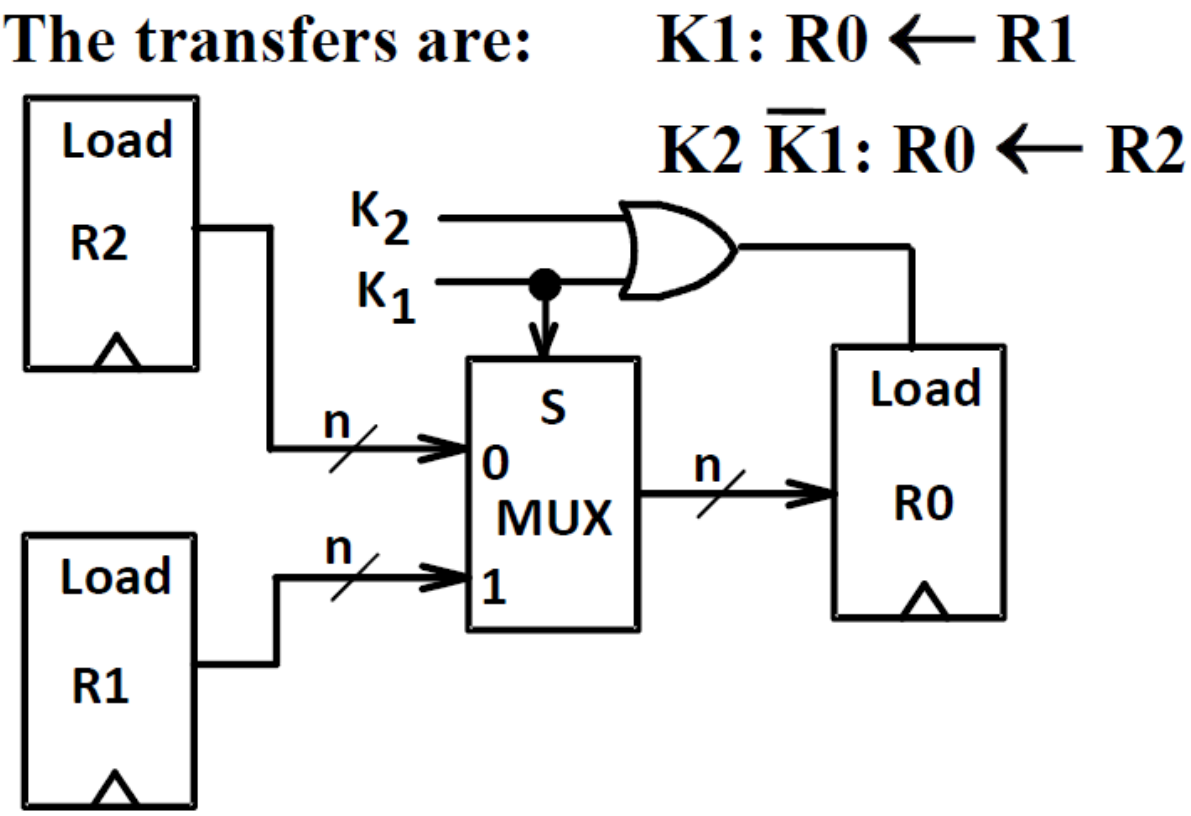
Symbolic Designation	Description
$R0 \leftarrow R1 + R2$	Addition
$R0 \leftarrow \overline{R1}$	One's Complement
$R0 \leftarrow \overline{R1} + 1$	Two's Complement
$R0 \leftarrow R2 + \overline{R1} + 1$	R2 minus R1
$R0 \leftarrow R1 + 1$	Increment (count up)
$R0 \leftarrow R1 - 1$	Decrement (count down)

Symbolic Designation	Description
$R0 \leftarrow \overline{R1}$	Bitwise NOT
$R0 \leftarrow R1 \vee R2$	Bitwise OR (set bits)
$R0 \leftarrow R1 \wedge R2$	Bitwise AND (clears bits)
$R0 \leftarrow R1 \otimes R2$	Bitwise EXOR (complements bits)

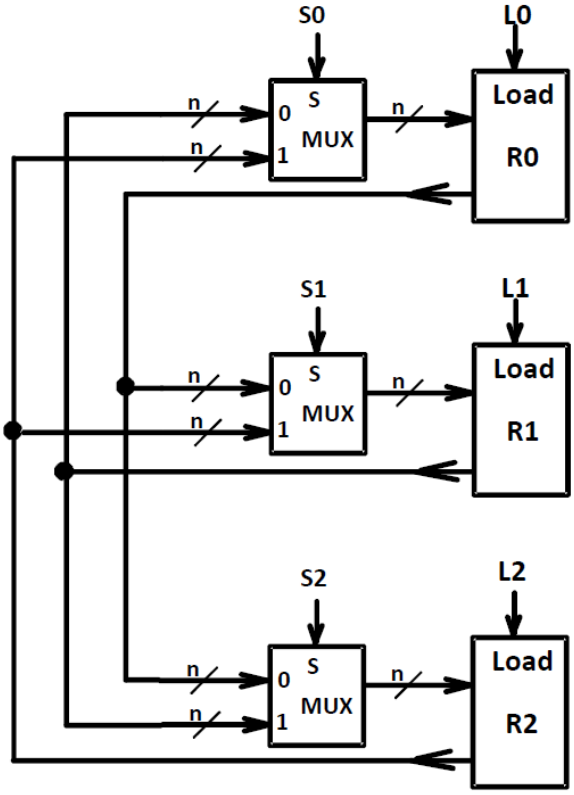
Symbolic Designation	Description	(R2 = 11001001) R1
$R1 \leftarrow sl\ R2$	Shift Left	10010010
$R1 \leftarrow sr\ R2$	Shift Right	01100100

Register Transfer Structures:

- **Multiplexer-Based Transfers** - Multiple inputs are selected by a multiplexer dedicated to the register



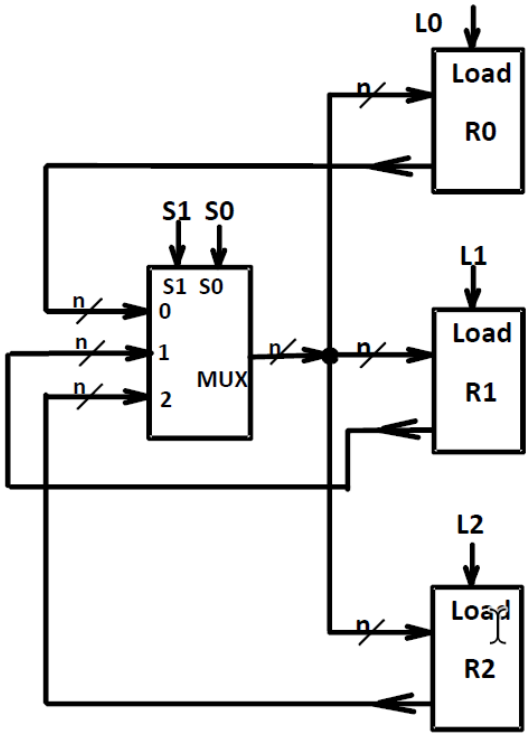
- **Multiplexer connected to each register input produces a very flexible transfer structure =>**
- **Characterize the simultaneous transfers possible with this structure.**
- **18 gate inputs per bit plus 3 shared inverters with total of 3 inputs**



simultaneous: 同步的

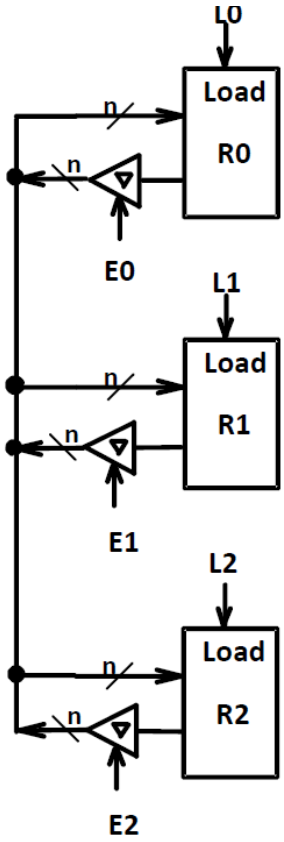
- **Bus-Based Transfers** - Multiple inputs are selected by a shared multiplexer driving a bus that feeds inputs to multiple registers

- **A single bus driven by a multiplexer lowers cost, but limits the available transfers =>**
- **Characterize the simultaneous transfers possible with this structure.**
- **Characterize the cost savings compared to dedicated multiplexers**
- **9 gate inputs per bit + shared decoder with 8 inputs**



- **Three-State Bus** - Multiple inputs are selected by 3-state drivers with outputs connected to a bus that feeds multiple registers

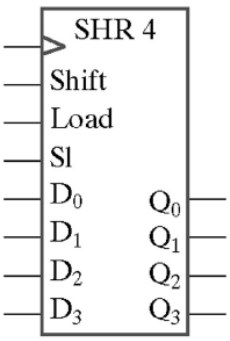
  - **The 3-input MUX can be replaced by a 3-state node (bus) and 3-state buffers.**
  - **Cost is further reduced, but transfers are limited**
  - **Characterize the simultaneous transfers possible with this structure.**
  - **Characterize the cost savings and compare**
  - **3 gate inputs per three state driver = 9 gate inputs**
  - **Other advantages?**

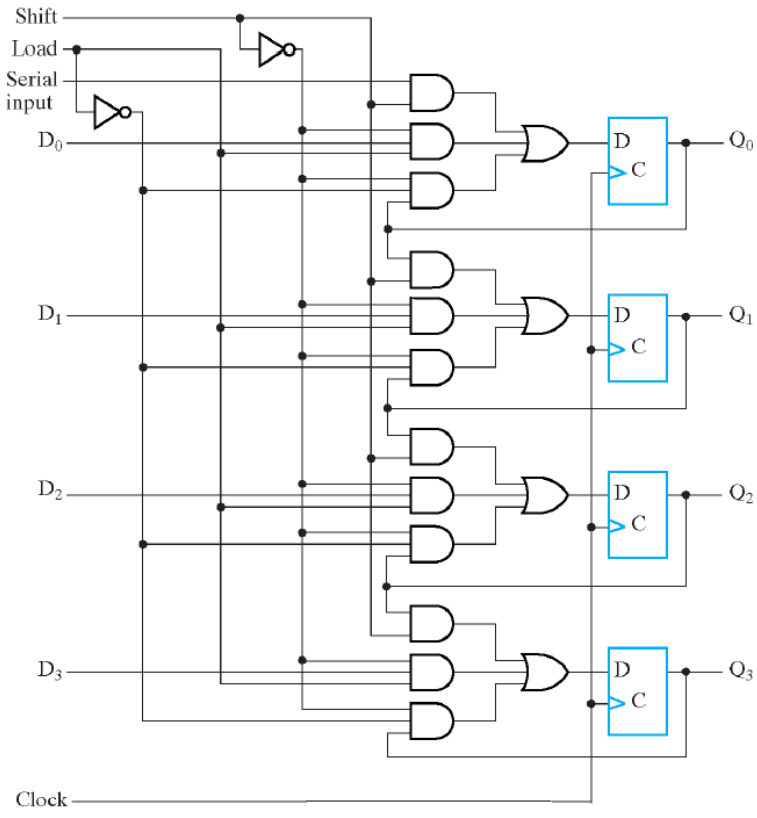


- **Other Transfer Structures** - Use multiple multiplexers, multiple buses, and combinations of all the above.

Shift Registers:
serial input (串行输入), shift right input (右移输入)
serial output (串行输出), parallel output (并行输出)

- **A parallel load shift register with an added "hold" operation that stores data:**





LSB是“最低有效位”（Least Significant Bit）的缩写，是指在二进制数中最右边的一位。在计算机科学中，LSB通常用于表示数字的二进制表示中的最后一位。MSB是“最高有效位”（Most Significant Bit）的缩写。

6.2 Counters, Register Cells, Buses, &Serial Operations

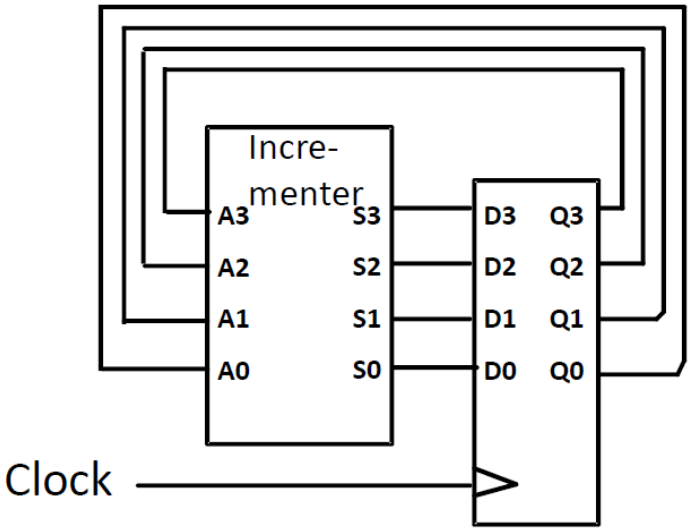
Counters: sequential circuits which "count" through a specific state sequence. They can count up, count down, or count through other fixed sequences. Two
distinct types are in common usage:

- Ripple Counters (for n bits, total worst case delay is n t_{PHL})

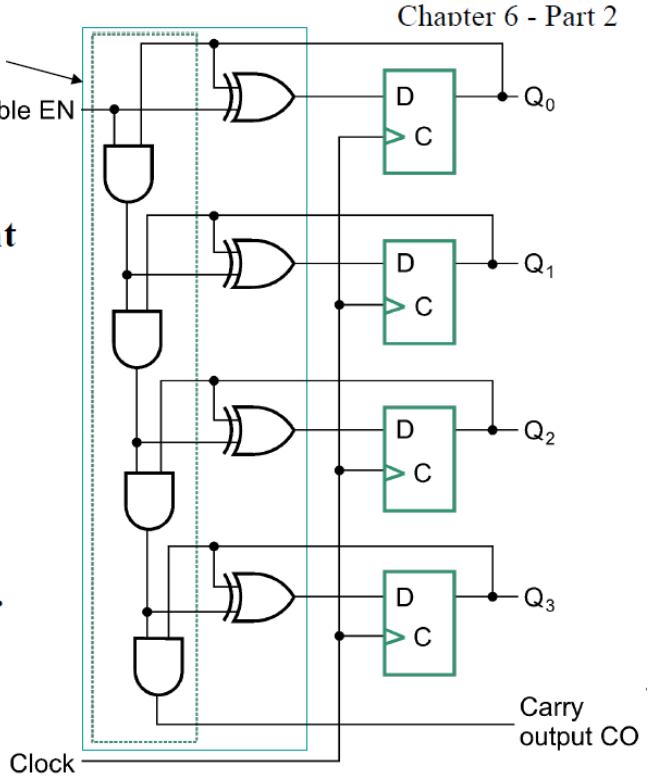
- Synchronous Counters
  - **To eliminate the "ripple" effects, use a common clock for each flip-flop and a combinational circuit to generate the next state.**
  - **Upward Counting Sequence**

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

=>

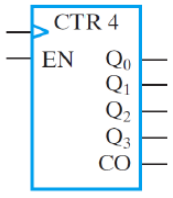


- **Internal details =>** Incrementer
- **Internal Logic**
  - **XOR complements each bit**
  - **AND chain causes complement of a bit if all bits toward LSB from it equal 1**
- **Count Enable**
  - **Forces all outputs of AND chain to 0 to "hold" the state**
- **Carry Out**
  - **Added as part of incrementer**
  - **Connect to Count Enable of additional 4-bit counters to form larger counters**

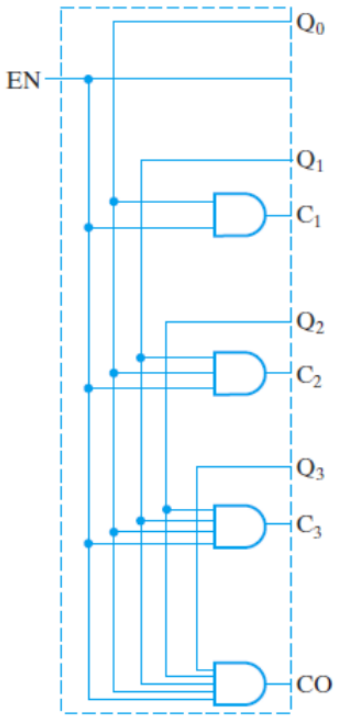


(a) Logic Diagram-Serial Gating

- **Carry chain**
  - **series of AND gates through which the carry "ripples"**
  - **Yields long path delays**
  - **Called *serial gating***
- **Replace AND carry chain with ANDs => in parallel**
  - **Reduces path delays**
  - **Called *parallel gating***
  - **Like carry lookahead**
  - **Lookahead can be used on COs and ENs to prevent long paths in large counters**
- **Symbol for Synchronous Counter**



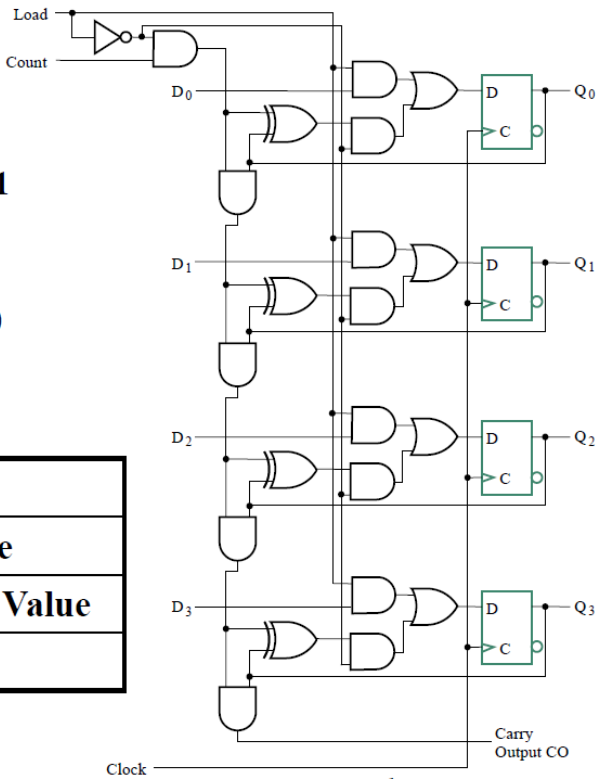
(c) Symbol



(b) Logic diagram—parallel gating

- Other Counters
  - Down Counter
  - Up-Down Counter
  - Parallel Load Counter
    - **Add path for input data**
      - **enabled for Load = 1**
    - **Add logic to:**
      - **disable count logic for Load = 1**
      - **disable feedback from outputs for Load = 1**
      - **enable count logic for Load = 0 and Count = 1**
    - **The resulting function table:**

Load	Count	Action
0	0	Hold Stored Value
0	1	Count Up Stored Value
1	X	Load D



  - Divide-by-n Counter

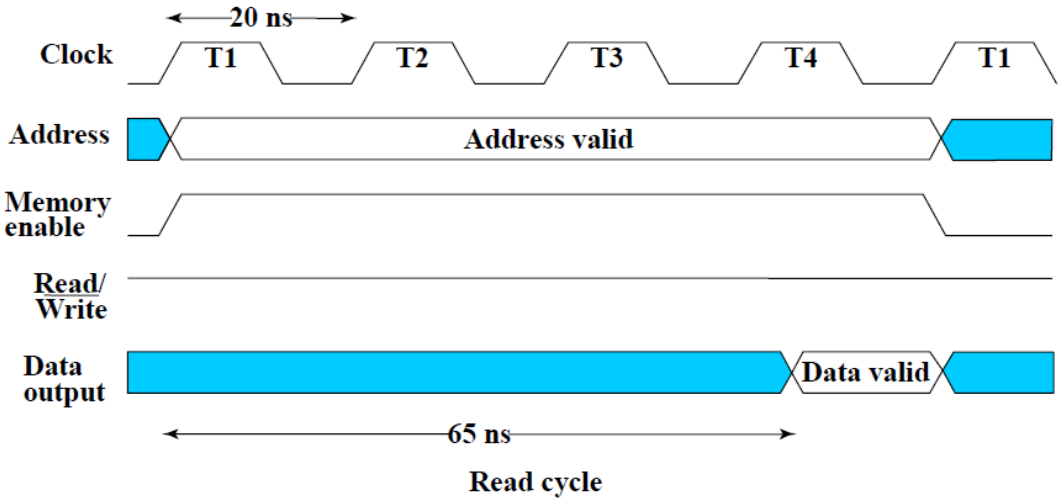
6.3 Control of Register Transfers

---

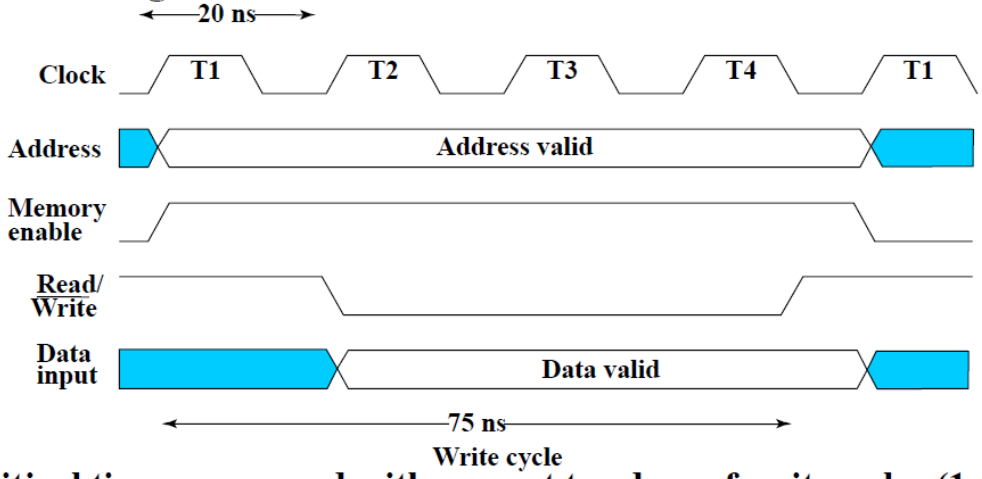
7 Memory Basics

7.1 Memory Definitions

- Memory ─ A collection of storage cells together with the necessary circuits to transfer information to and from them.
- Memory Organization ─ the basic architectural structure of a memory in terms of how data is accessed.
- Random Access Memory (RAM) ─ a memory organized such that data can be transferred to or from any cell (or collection of cells) in a time that is not dependent upon the particular cell selected.
- Memory Address ─ A vector of bits that identifies a particular memory element (or collection of elements).
- Typical data elements are: bit, byte, word
- Memory Data ─ a bit or a collection of bits to be stored into or accessed from memory cells.
- Memory Operations ─ operations on memory data supported by the memory unit. Typically, *read* and *write* operations over some data element (bit, byte, word, etc.).
  - **Most basic memories are asynchronous**
    - **Storage in latches or storage of electrical charge**
    - **No clock**
  - **Controlled by control inputs and address**
  - **Timing of signal changes and data observation is critical to the operation**
  - **Read timing:**



- **Write timing:**



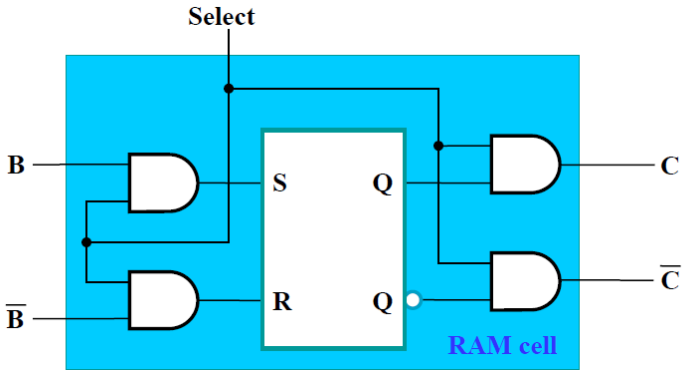
- **Critical times measured with respect to edges of write pulse (1-0-1):**
  - **Address must be established at least a specified time before 1-0 and held for at least a specified time after 0-1 to avoid disturbing stored contents of other addresses**
  - **Data must be established at least a specified time before 0-1 and held for at least a specified time after 0-1 to write correctly**

7.2 RAM Integrated Circuits

- Types of random access memory:
  - Static: information stored in latches
  - Dynamic: information stored as electrical charges on capacitors
- Dependence on Power Supply
  - Volatile: loses stored information when power turned off
  - Non-volatile: retains information when power turned off

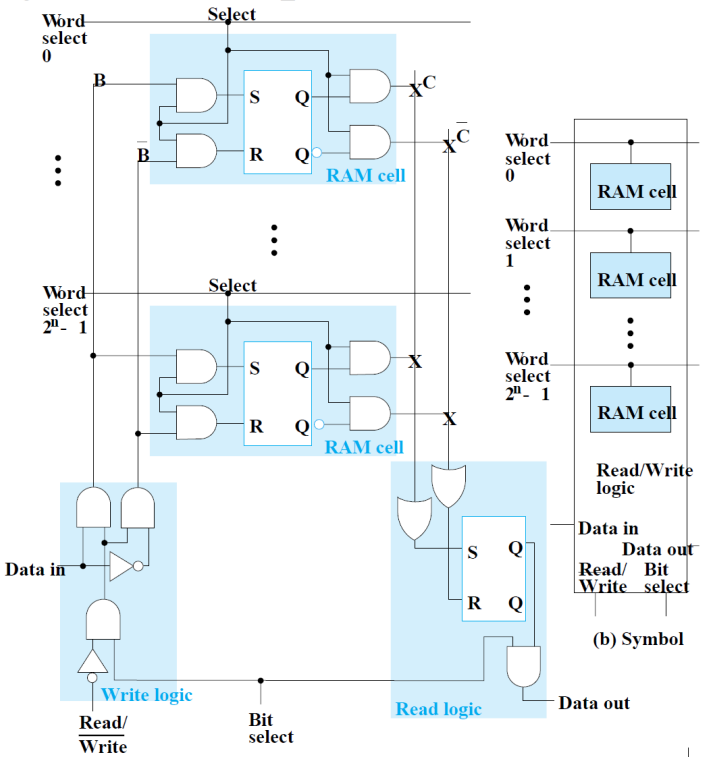
7.2.1 Static RAM

Storage cell: SR Latch, Select input for control, Dual Rail Data Input B and B', Dual Rail Data Output C and C'



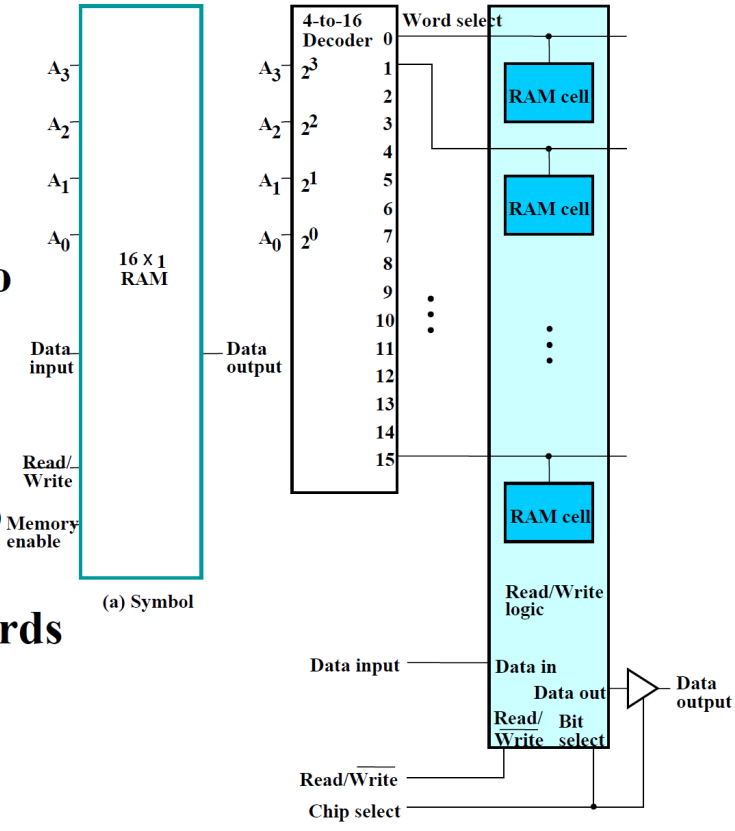
Static RAM — Bit Slice

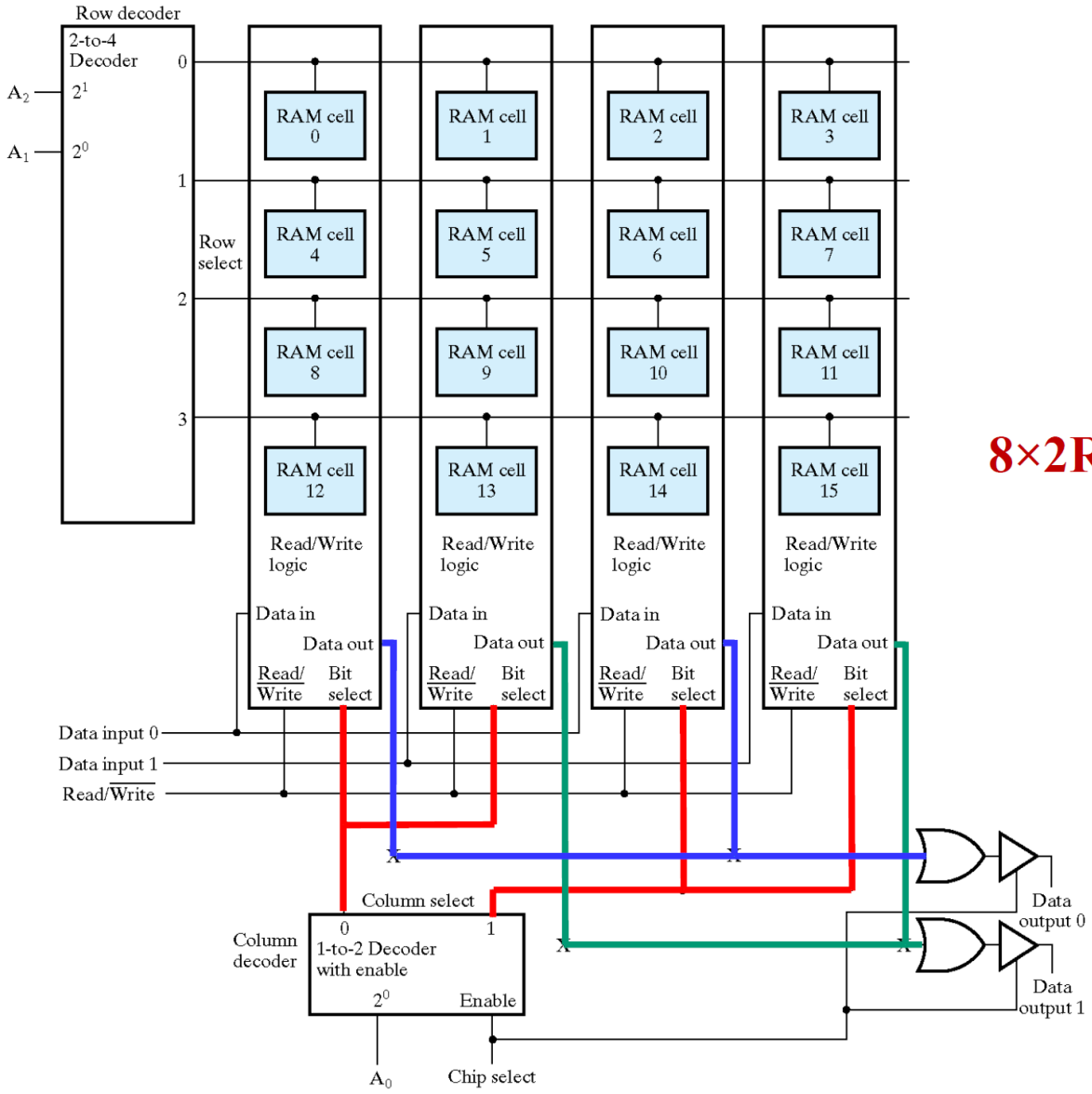
- Represents all circuitry that is required for 2^n 1-bit words
  - Multiple RAM cells
  - Control Lines:
    - Word select i – one for each word
    - Read / $\overline{\text{Write}}$
    - Bit Select
  - Data Lines:
    - Data in
    - Data out



2^n-Word × 1-Bit RAM IC

- To build a RAM IC from a RAM slice, we need:
  - Decoder — decodes the n address lines to 2^n word select lines
  - A 3-state buffer on the data output permits RAM ICs to be combined into a RAM with c × 2^n words



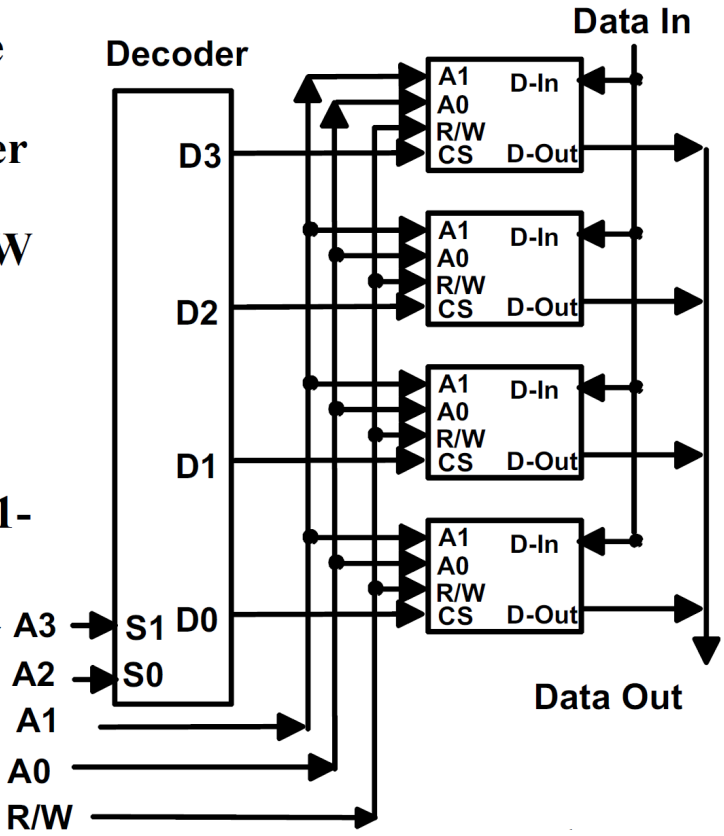


8×2RAM

8 x 2 RAM: 8 words, 2 bits

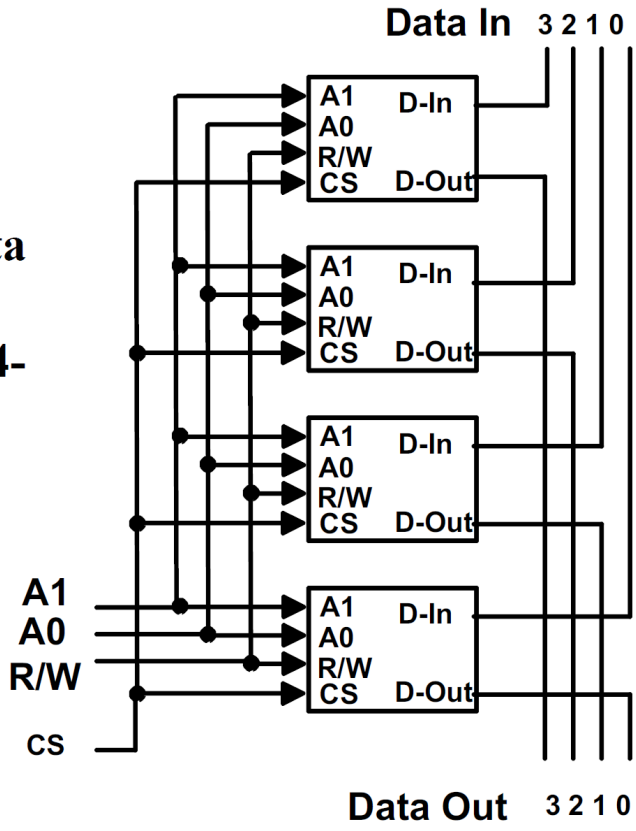
Making Larger Memories

- Using the CS lines, we can make larger memories from smaller ones by tying all address, data, and R/W lines in parallel, and using the decoded higher order address bits to control CS.
- Using the 4-Word by 1-Bit memory from before, we construct a 16-Word by 1-Bit memory. ⇒



Making Wider Memories

- To construct wider memories from narrow ones, we tie the address and control lines in parallel and keep the data lines separate.
- For example, to make a 4-word by 4-bit memory from 4, 4-word by 1-bit memories ⇒
- Note: Both 16x1 and 4x4 memories take 4-chips and hold 16 bits of data.

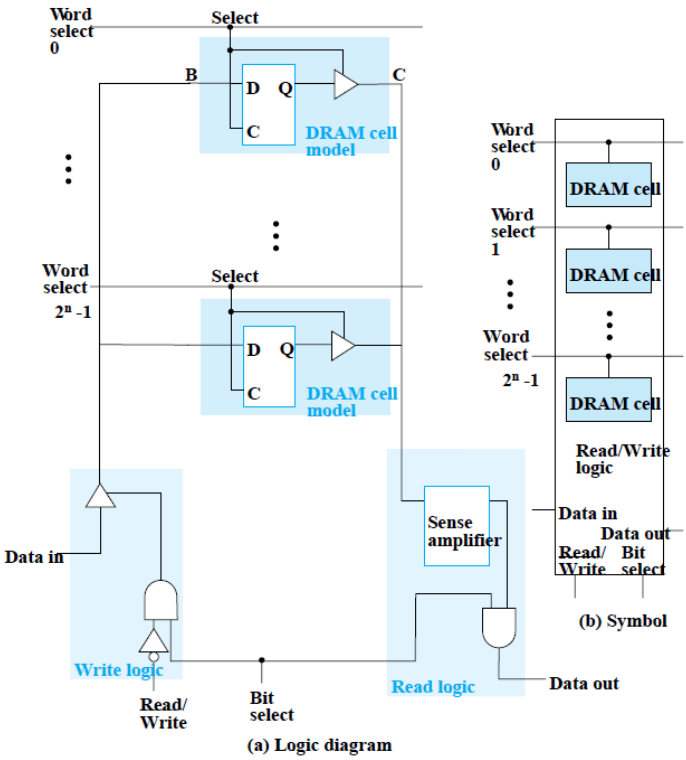


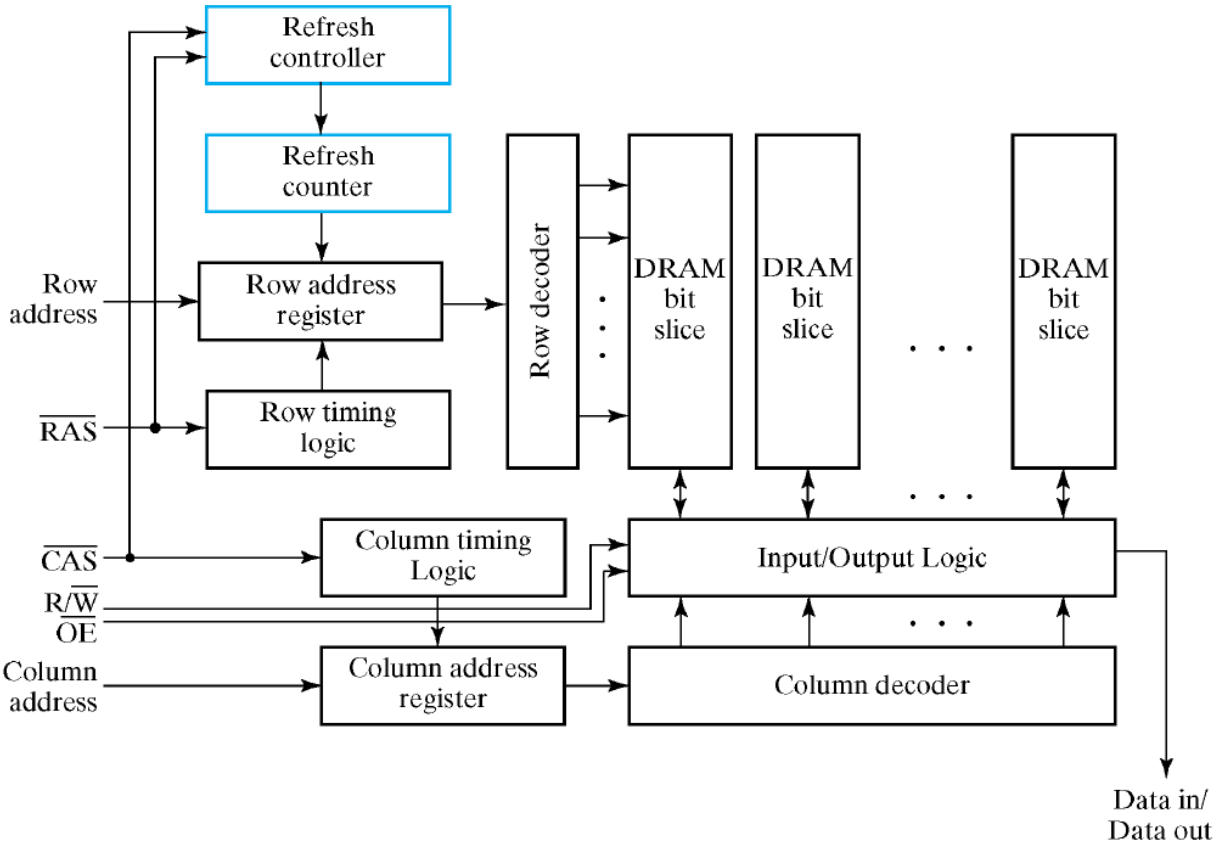
7.2.2 Dynamic RAM (DRAM)

- Basic Principle: Storage of information on capacitors (电容器)
- Charge and discharge of capacitor to change stored value
- Use of transistor as “switch” to Store charge and Charge or discharge

Dynamic RAM - Bit Slice

- C is driven by 3-state drivers
- Sense amplifier is used to change the small voltage change on C into H or L
- In the electronics, B, C, and the sense amplifier output are connected to make destructive read into non-destructive read





DRAM Types:

- Synchronous DRAM (SDRAM)
- Double Data Rate SDRAM (DDR SDRAM)
- RAMBUS DRAM (RDRAM)