中国科学技术大学
University of Science and Technology of China

计算系统概论A
Introduction to Computing Systems
(CS1002A.03)

Chapter 3
Digital Logic Structures

陈俊仕
cjuns@ustc.edu.cn
2022 Fall

计算机科学与技术学院
School of Computer Science and Technology

Previously

Microprocessors contain billions of transistors

- Intel Core 2 Duo due(2006)/core i7(2015): 0.291/1.9 billion
- AMD Barcelona(2006)/Ryzen(2017): 0.463/4.8 billion
- IBM Power6: 0.79 billion

Transistor: Building Block of Computers

- Logically, each transistor acts as a switch

Combined to implement logic functions

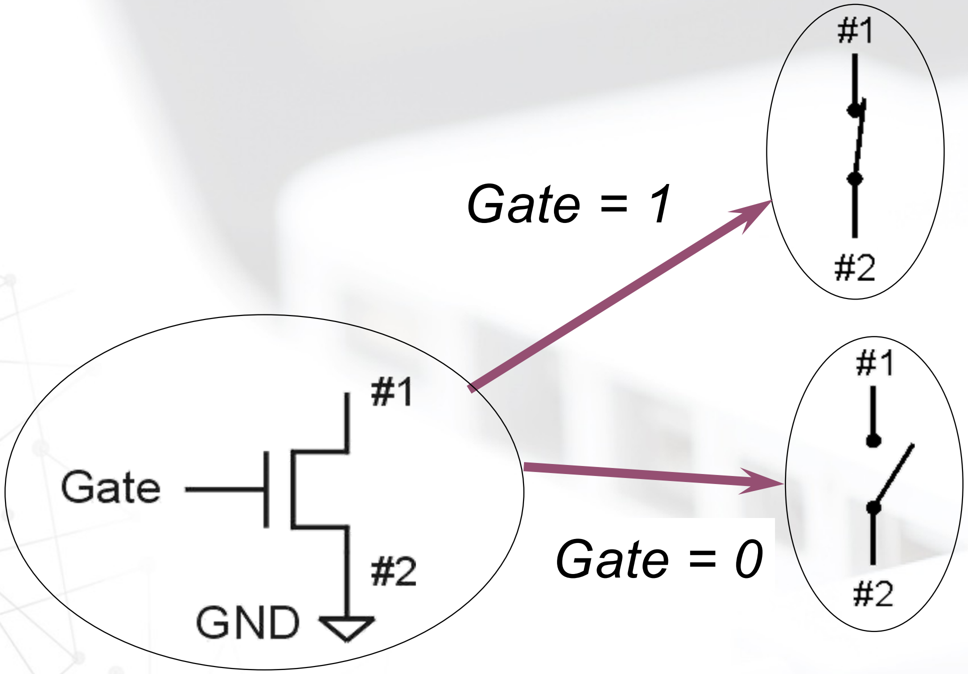
- AND, OR, NOT

Building Functions from Logic Gates

- Combinational Logic Circuit
- Sequential Logic Circuit

Previously: Transistor

- **N-type MOS**

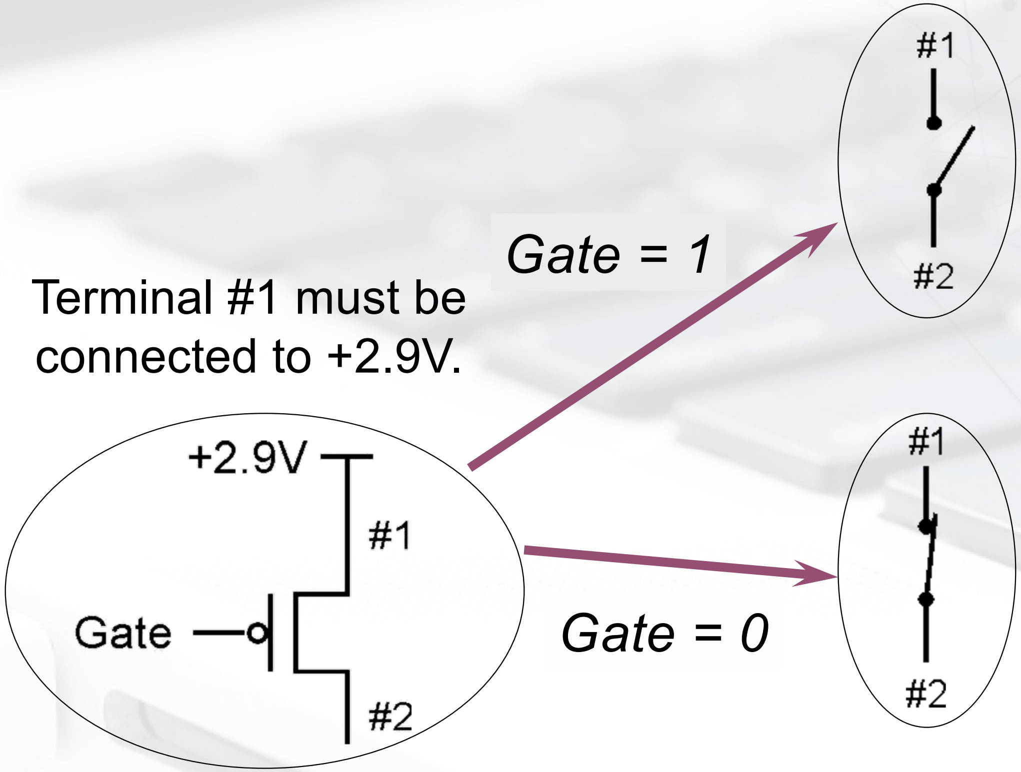


Gate = 1

Gate = 0

Terminal #2 must be connected to GND (0V).

- **P-type MOS**



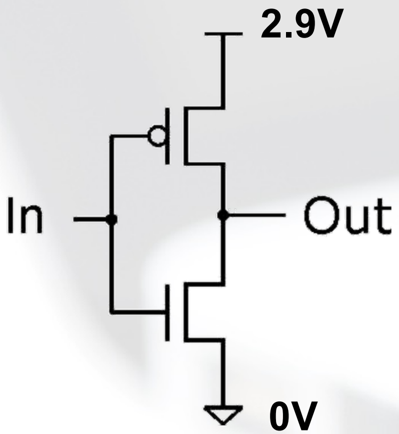
Terminal #1 must be connected to +2.9V.

Gate = 1

Gate = 0

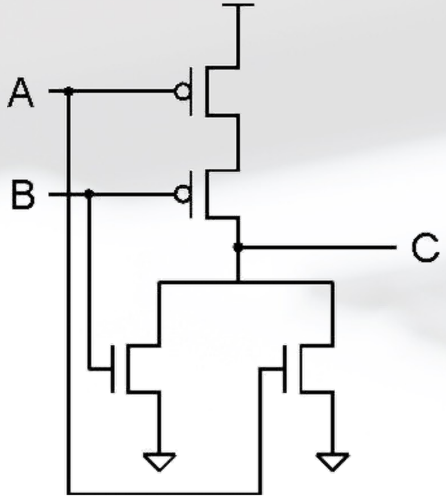
Previously: Logic Gates

- **NOT vs. NOR vs. OR**



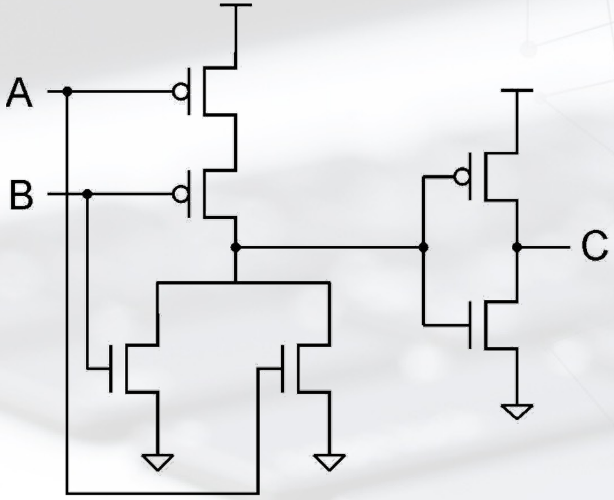
NOT

In	Out
0	1
1	0



NOR

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

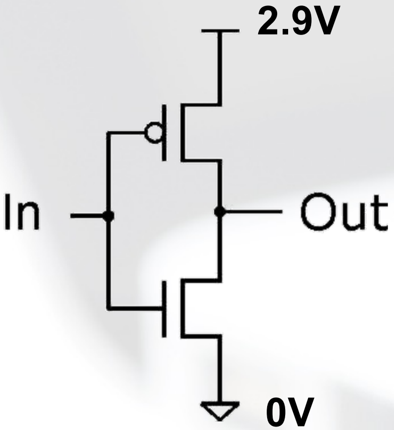


OR

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

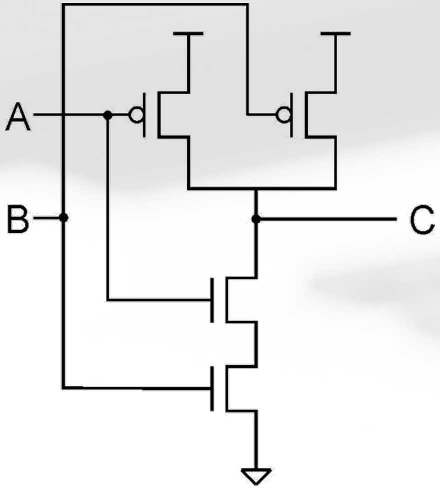
Previously: Logic Gates

- NOT vs. NAND vs. AND



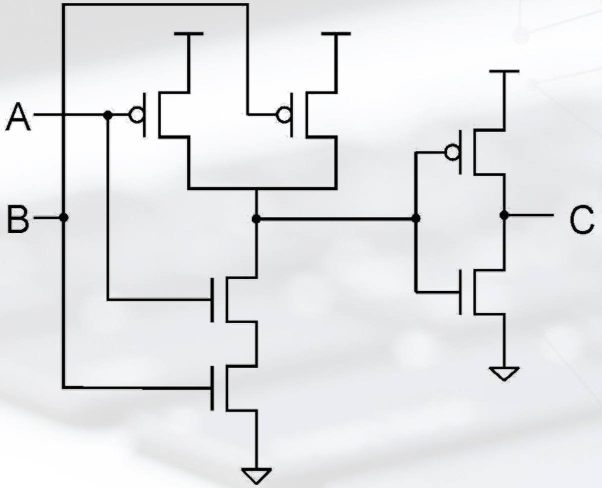
NOT

In	Out
0	1
1	0



NAND

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

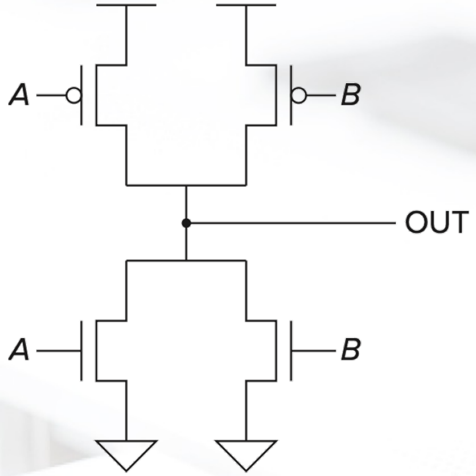


AND

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

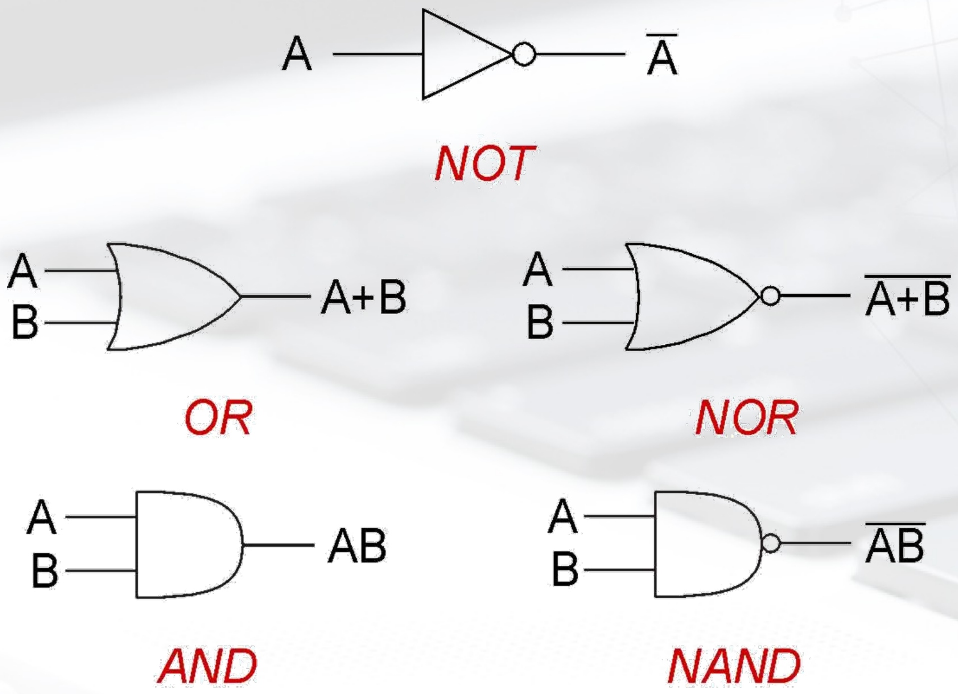
Problem 3.7

- The following circuit has a major flaw. Can you identify it?



Previously: Basic Gates

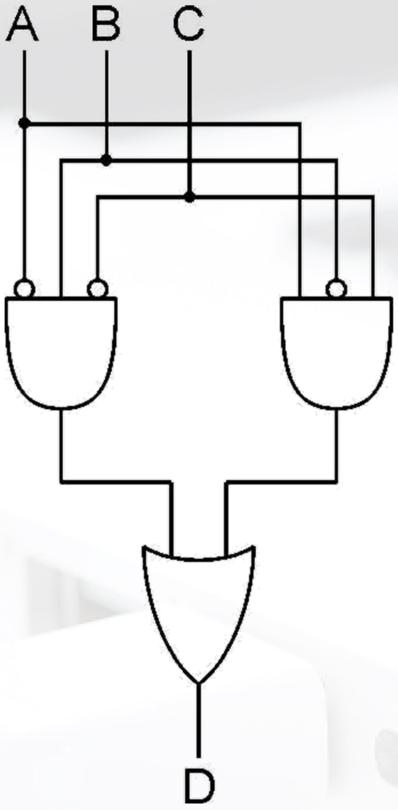
- **From Now on……Gates**
  - Covered transistors mostly so that you know they exist
  - Note: “Logic Gate” not related to “Gate” of transistors
- **Will study implementation in terms of gates**
  - Circuits that implement Boolean functions
- **More complicated gates from transistors possible**
  - XOR, Multiple-input AND-OR-Invert (AOI) gates



Previously: Logical Completeness

- AND, OR, NOT can implement ANY truth table

INPUT			OUTPUT
A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

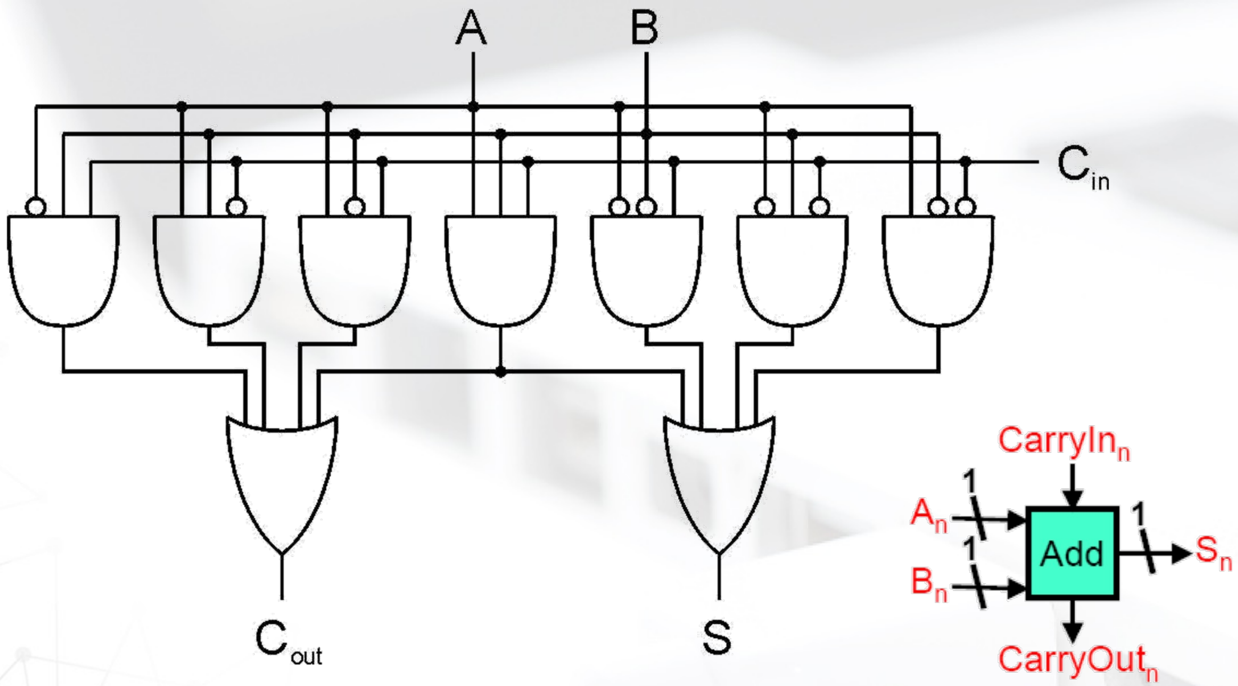


1. AND combinations that yield a "1" in the truth table.
2. OR the results of the AND gates.

Previously: Combinational Logic Circuits

■ A One-Bit Adder (aka. a Full Adder)

- Add two bits and carry-in, produce one-bit sum and carry-out.

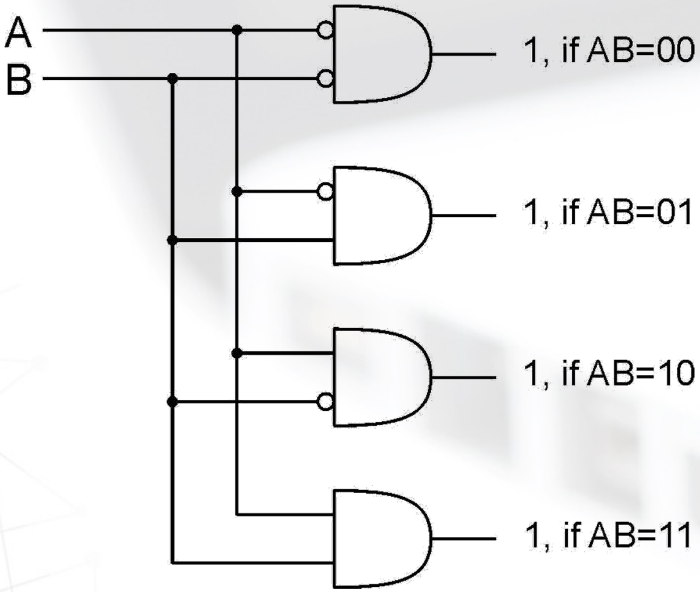


The truth table of a one-bit adder

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

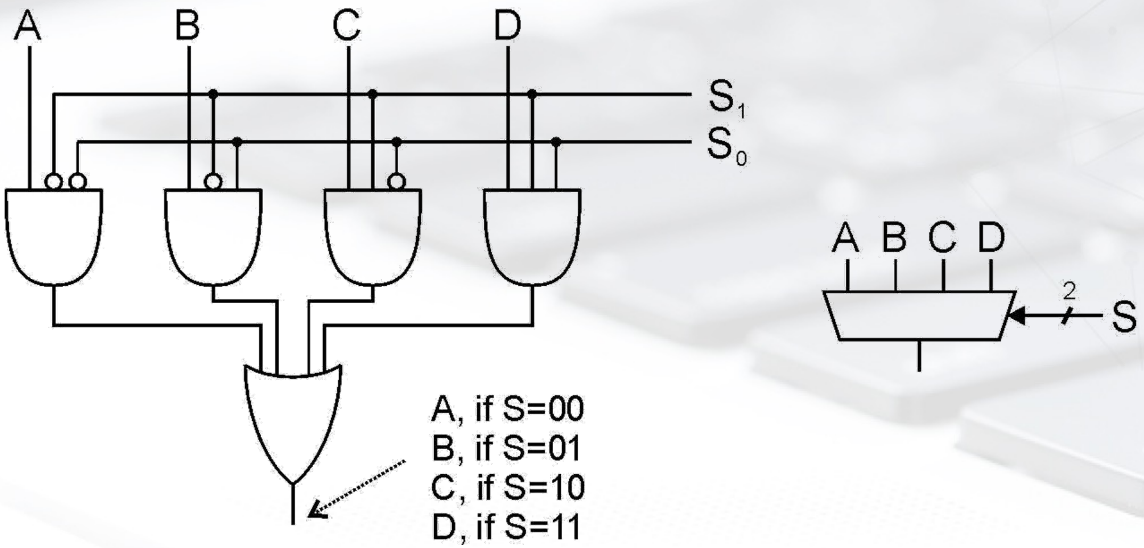
Previously: Combinational Logic Circuits

- ***Decoder***



2-bit decoder

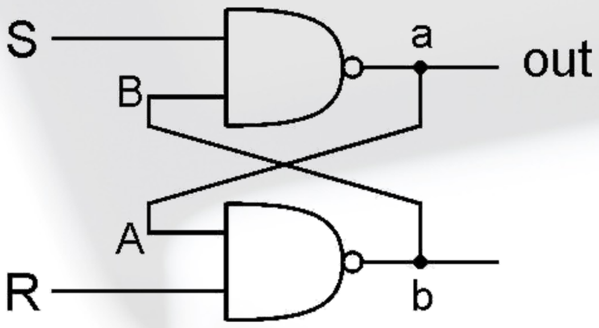
- ***Multiplexer (MUX)***



4-to-1 MUX

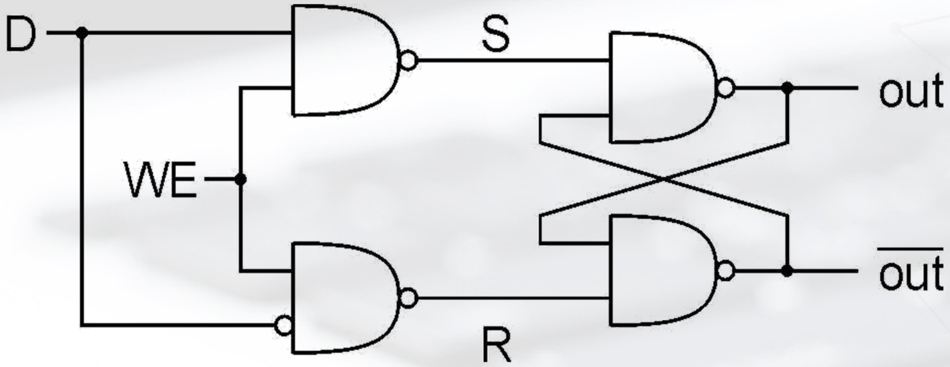
Previously: Basic Storage Elements

■R-S Latch



S	R	out	out’	
1	1	0/1	0/1	*hold*
1	0	0/1	0/0	
0	1	0/1	1/1	
0	0	x	x	

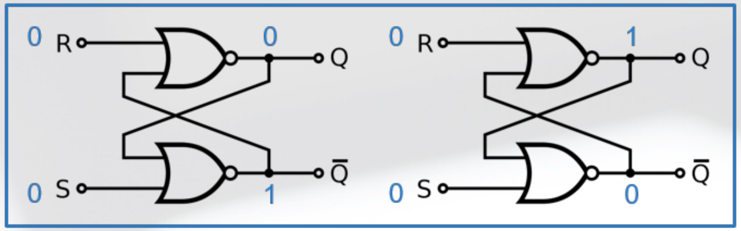
■The Gated D-Latch



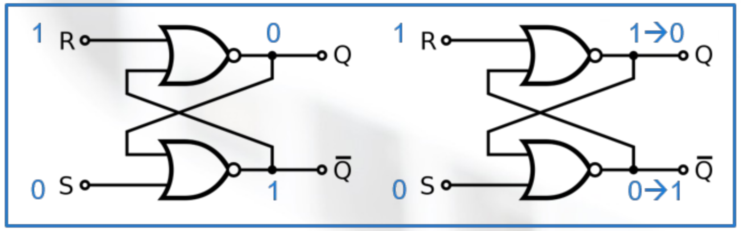
D	WE	S	R	out
0	0	1	1	*hold*
0	1	1	0	0
1	0	1	1	*hold*
1	1	0	1	1

Previously: Basic Storage Elements

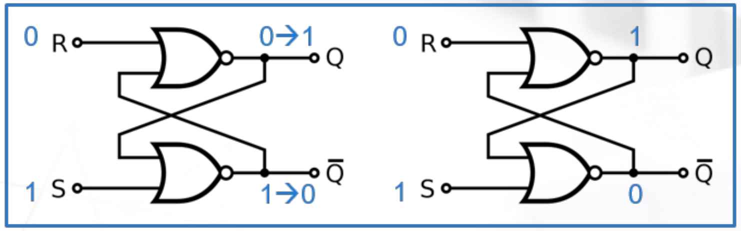
$(R, S) = (0, 0)$→ Hold the previous state as $(Q_{next}, \overline{Q_{next}}) = (Q, \overline{Q})$



$(R, S) = (1, 0)$→ Reset as $(Q_{next}, \overline{Q_{next}}) = (0, 1)$

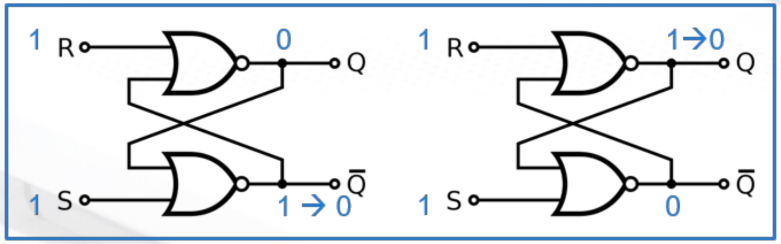


$(R, S) = (0, 1)$→ Set as $(Q_{next}, \overline{Q_{next}}) = (1, 0)$



S	R	out	out’	
0	0	0/1	0/1	*hold*
0	1	0/1	0/0	
1	0	0/1	1/1	
1	1	x	x	

$(R, S) = (0, 1)$→ Not allowed as $(Q_{next}, \overline{Q_{next}}) = (0, 0)$ that is not logical



Outline

1 The Transistor

2 Logic Gates

3 Combinational Logic Circuits

4 Basic Storage Elements

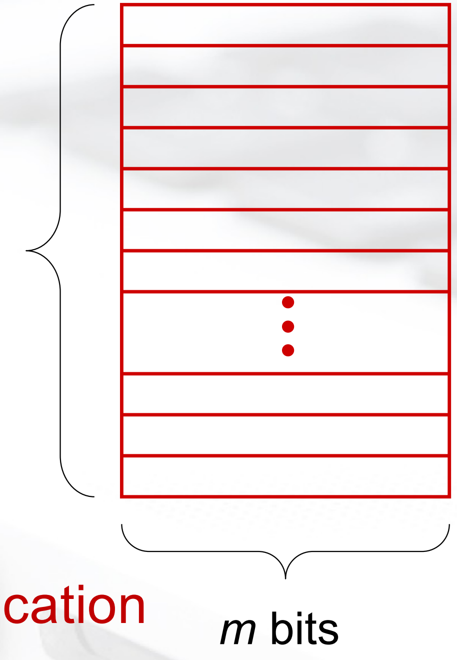
5 The Concept of Memory

6 Sequential Logic Circuits

7 Preview of Coming Attractions: From Logic to Data Path

Memory

- Now that we know how to store bits, we can build a memory – a logical $k \times m$ array of stored bits.



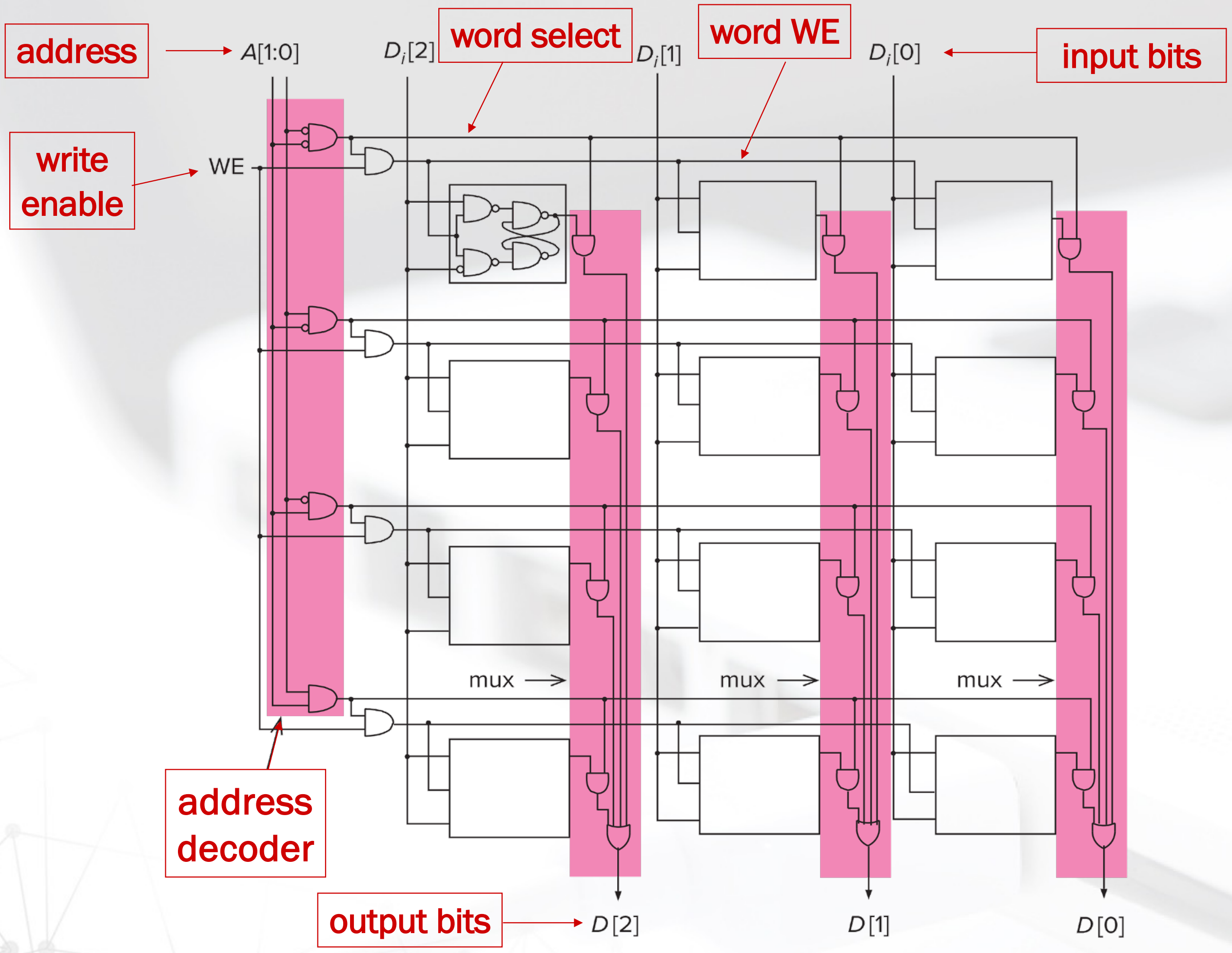
Address Space:
number of locations
(usually a power of 2)

$k = 2^n$
locations

Addressability:
number of bits per location
(e.g., byte-addressable)

m bits

A 2^2 x 3 Memory


address

write enable

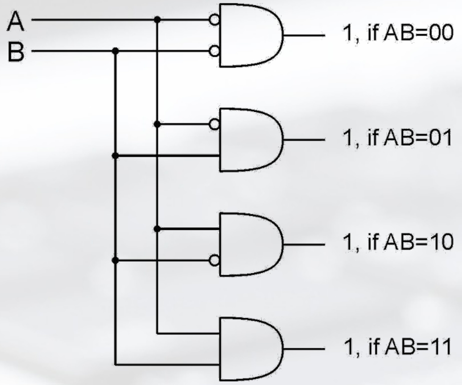
word select

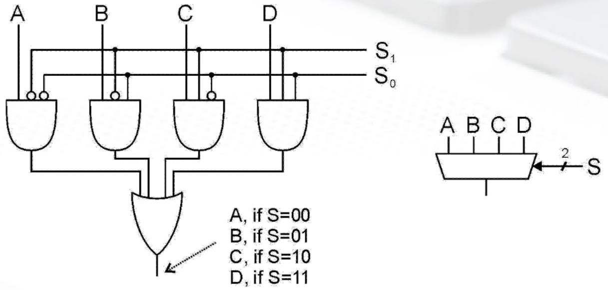
word WE

input bits

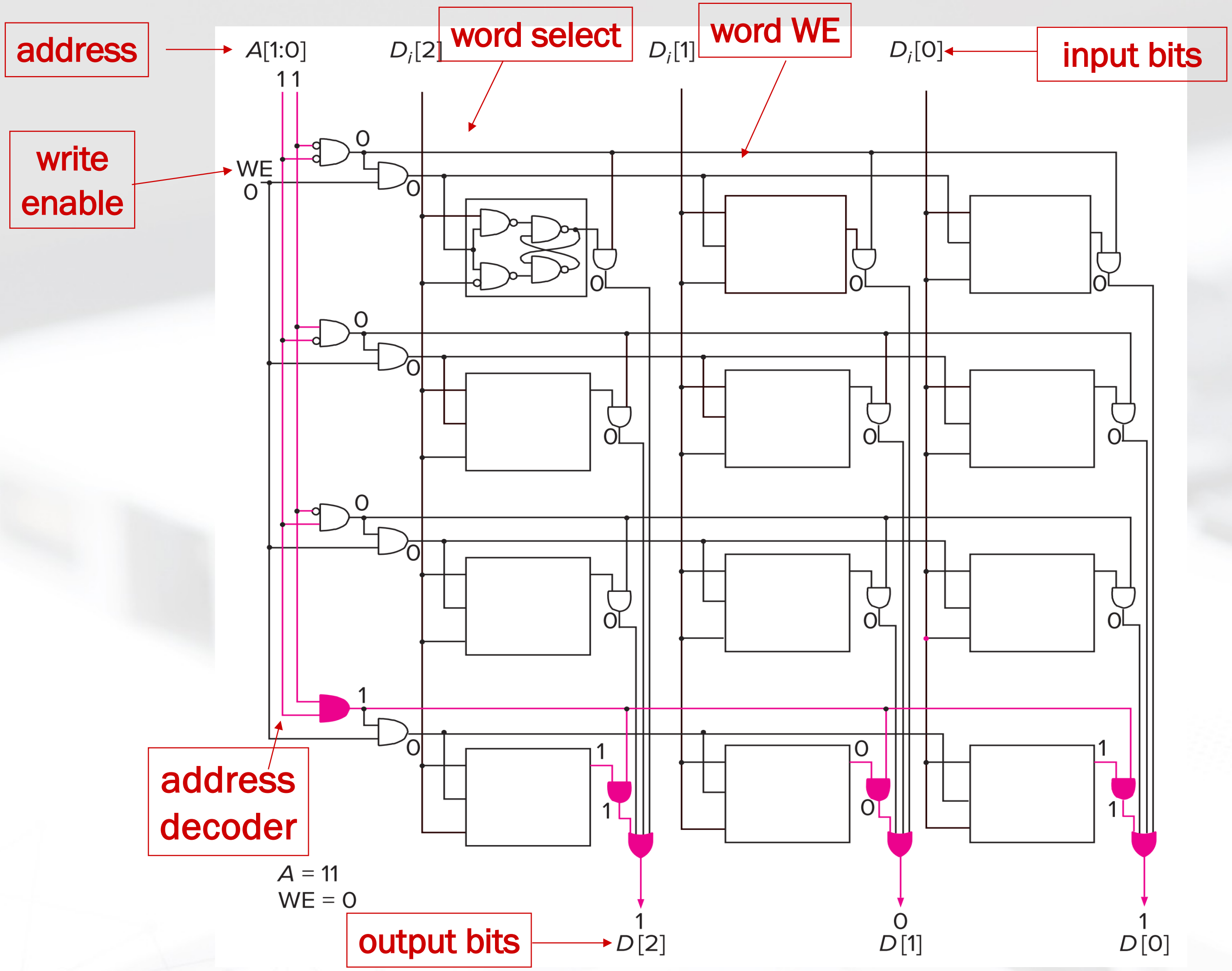
address decoder

output bits





Reading location 3 in our 2^2 x 3 memory



address

write enable

word select

word WE

input bits

address decoder

output bits

More Memory Details

- This is a not the way actual memory is implemented.
  - fewer transistors, much more dense, relies on electrical properties
- But the logical structure is very similar.
  - address decoder
  - word select line
  - word write enable
- Two basic kinds of RAM (Random Access Memory)
  - Static RAM (SRAM)
    - fast, not very dense (bit-cell is a latch)
  - Dynamic RAM (DRAM)
    - slower but denser, bit storage must be periodically refreshed
    - each bit-cell is a capacitor (like a leaky bucket) that decays

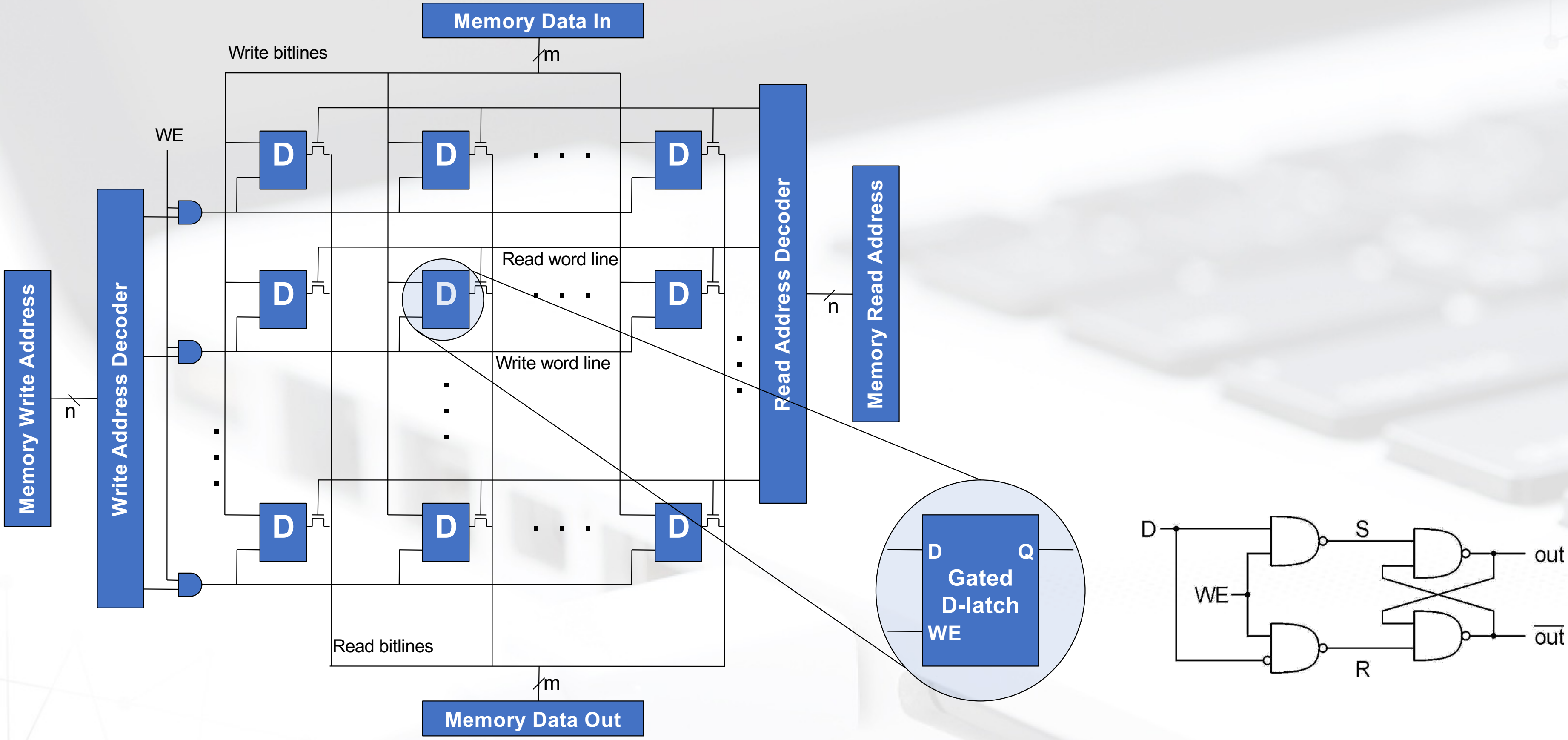
Also, non-volatile memories: ROM, PROM, flash, …

More Memory Details

- **This is a not the way actual memory is implemented.**
  - fewer transistors, much more dense, relies on electrical properties
- **But the logical structure is very similar.**
  - address decoder
  - word select line
  - word write enable
- **Two basic kinds of RAM (Random Access Memory)**
  - Static RAM (SRAM)
    - — fast, not very dense (bit-cell is a latch)
  - Dynamic RAM (DRAM)
    - — slower but denser, bit storage must be periodically refreshed
    - — each bit-cell is a capacitor (like a leaky bucket) that decays

Also, non-volatile memories: ROM, PROM, flash, …

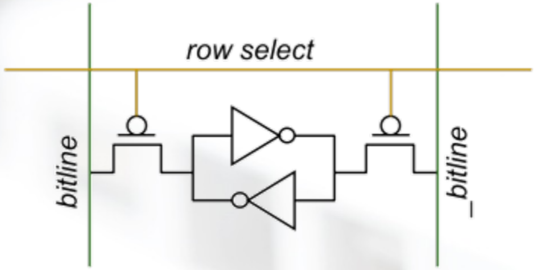
SRAM Memory

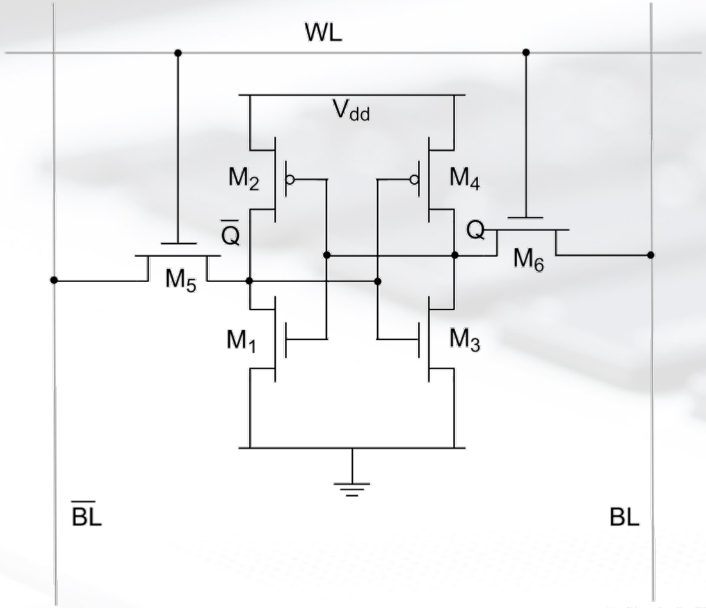

Memory Data In
Write bitlines
m
WE
D
D
D
Read word line
Write word line
Memory Write Address
n
Write Address Decoder
Read Address Decoder
n
Memory Read Address
Read bitlines
m
Memory Data Out
D
Q
Gated
D-latch
WE
D
WE
S
R
out
$\overline{\text{out}}$

SRAM Memory

6-Transistor SRAM

- 3 operations
  - —read, write, hold

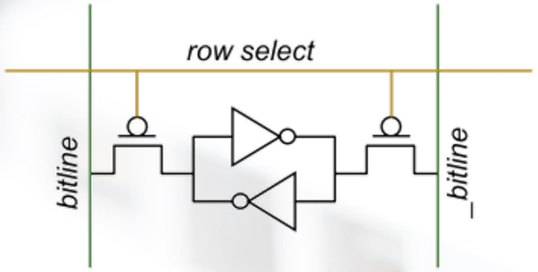


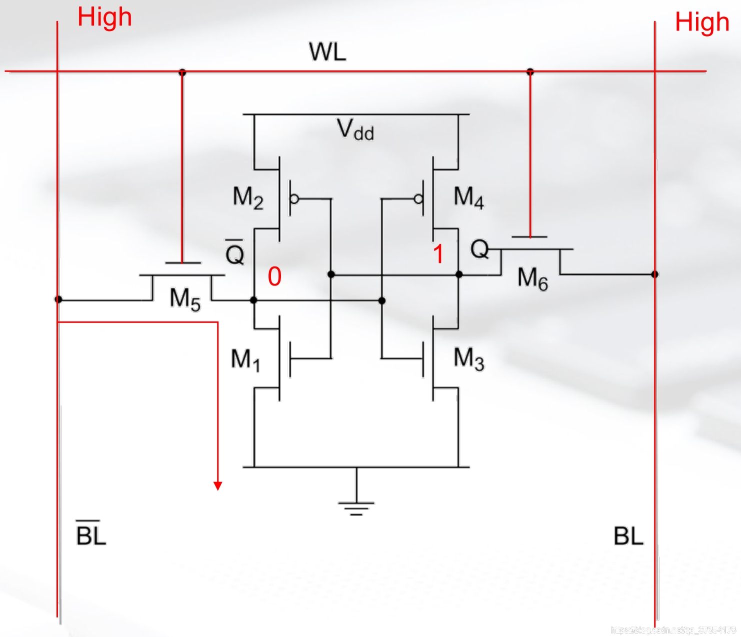


SRAM Memory

6-Transistor SRAM

- 3 operations
  - —read, write, hold



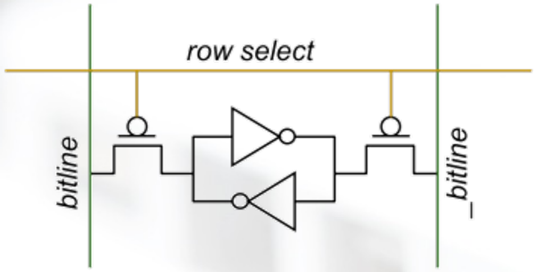


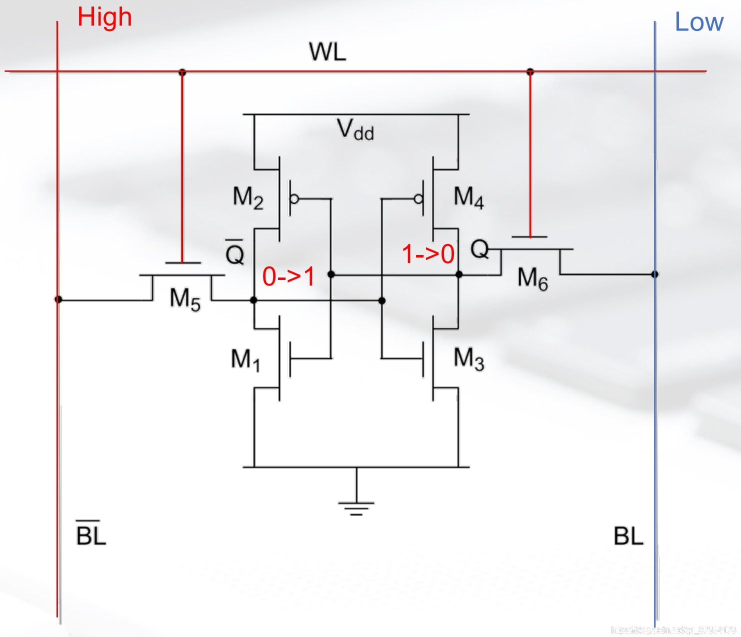
TODO：补充读的过程

SRAM Memory

6-Transistor SRAM

- 3 operations
  - —read, write, hold

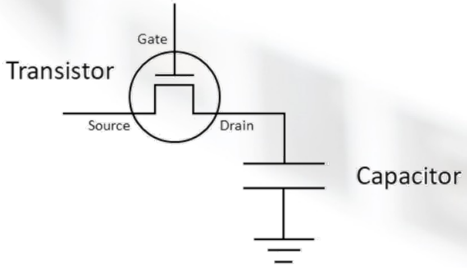


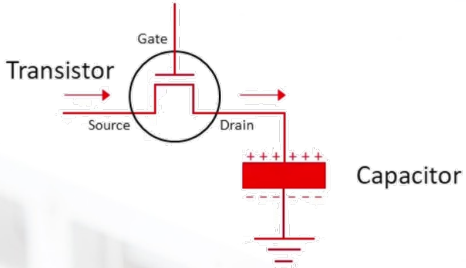


DRAM Memory

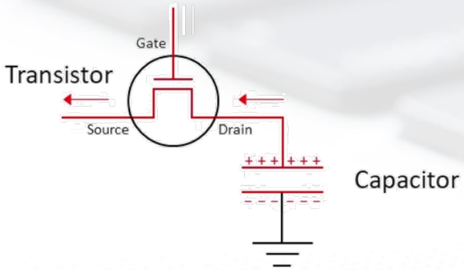
Basic Storage Element of DRAM

- one transistor + one capacitor
- slower but denser, bit storage must be periodically refreshed
- each bit-cell is a capacitor (like a leaky bucket) that decays



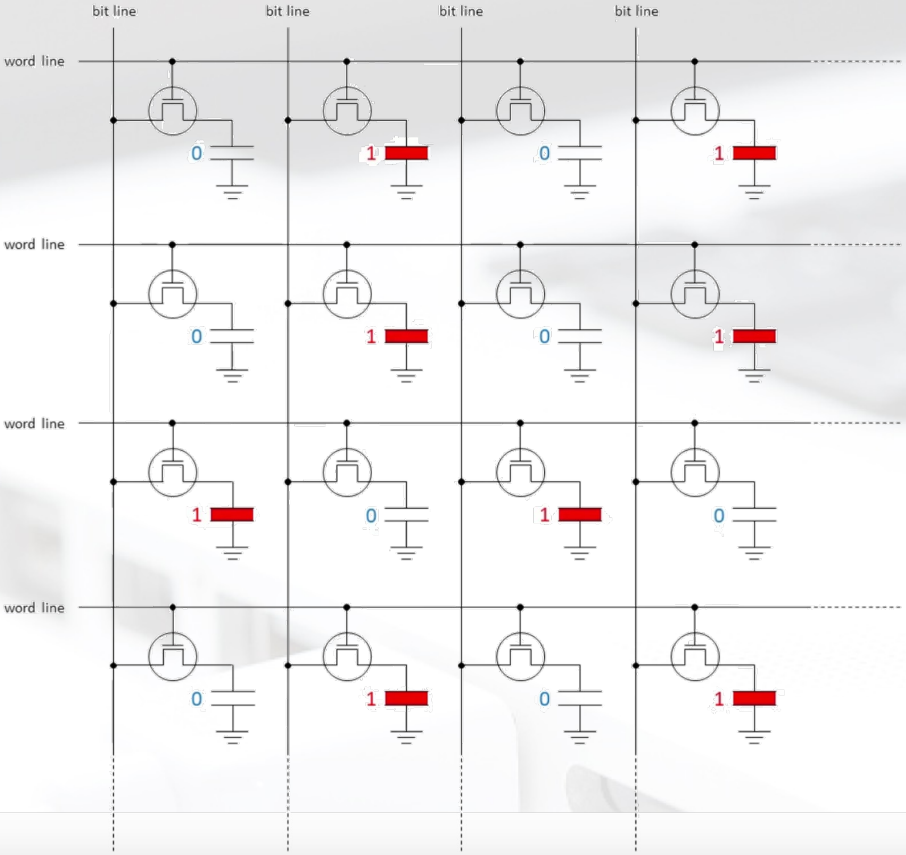


1



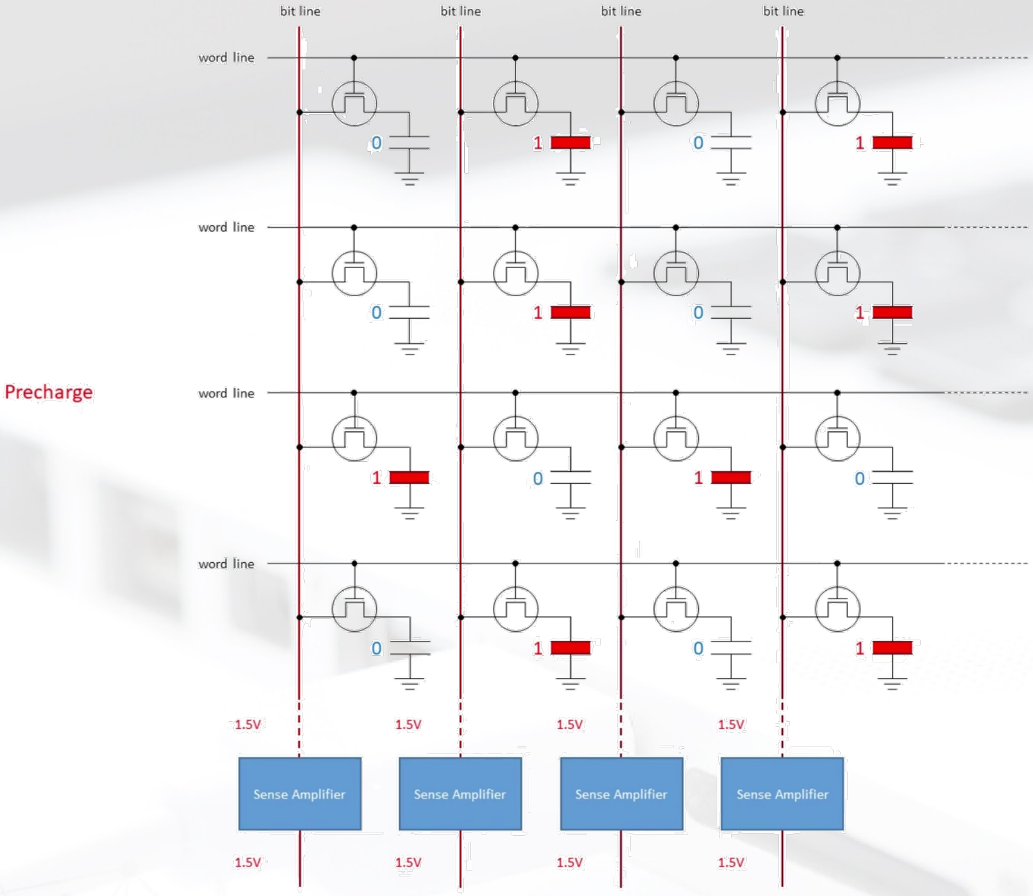
0

DRAM Memory


bit line
bit line
bit line
bit line
word line
0
1
0
1
word line
0
1
0
1
word line
1
0
1
0
word line
0
1
0
1

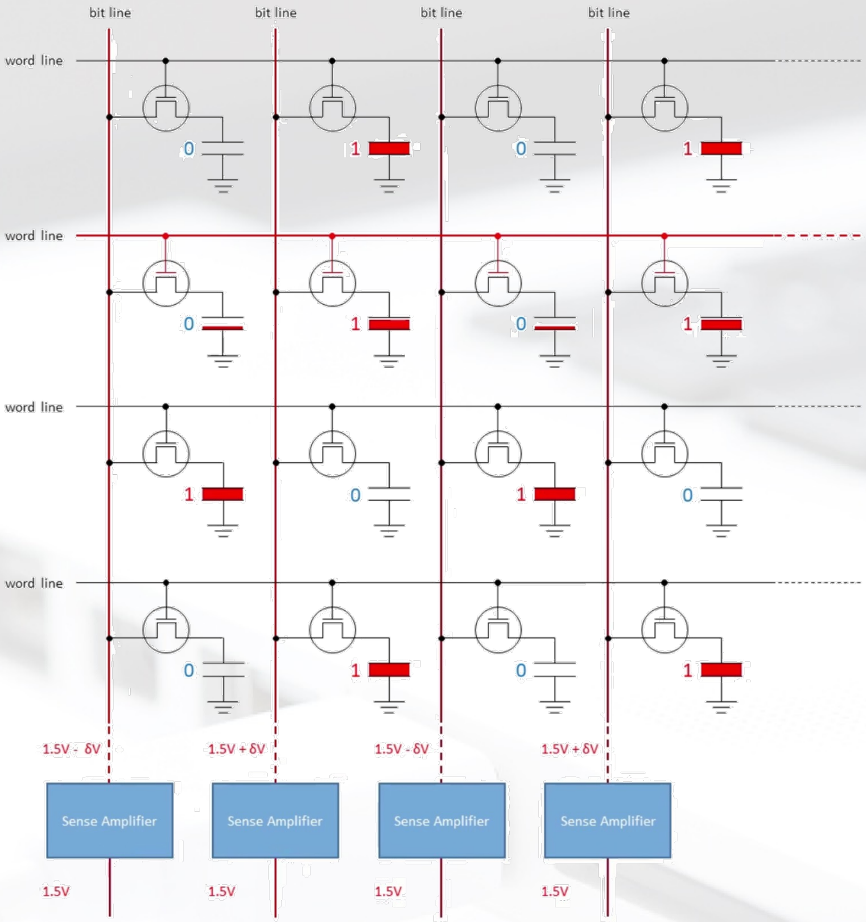
DRAM Memory

■ Precharge bit line



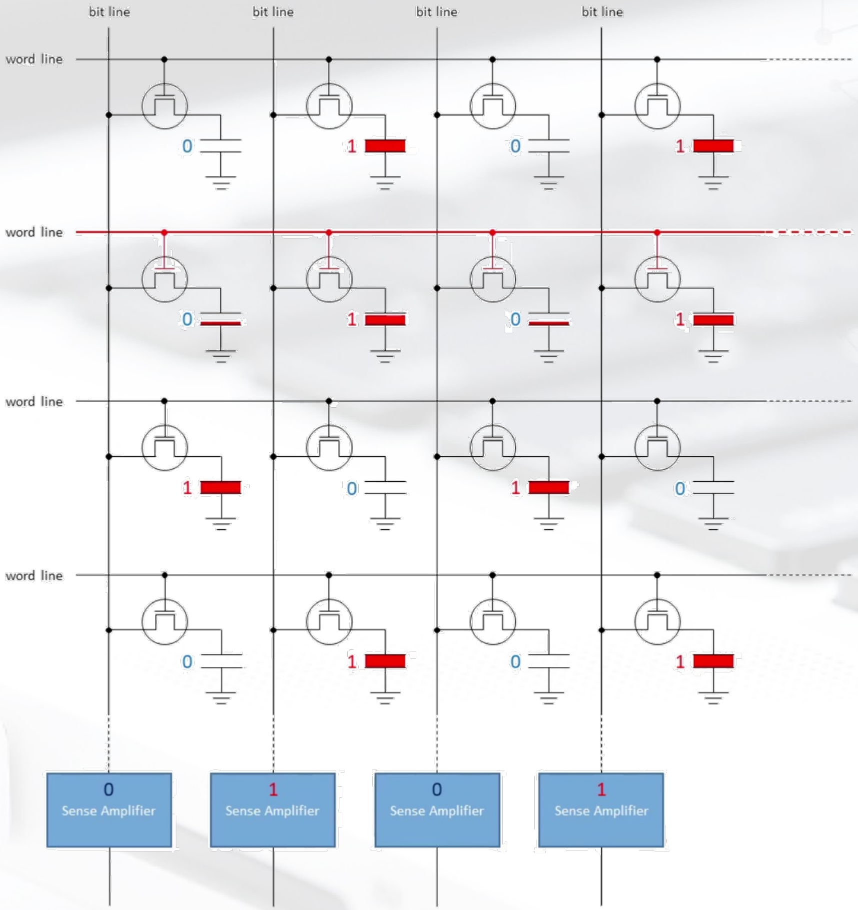
DRAM Memory

- **Select word line**



DRAM Memory

- **Read data with sense amplifier**
  - destructive read
    - —when a cell is read, it must then be re-written
  - Charge leakage
    - —requires that all cells are periodically refreshed



Outline

1 The Transistor

2 Logic Gates

3 Combinational Logic Circuits

4 Basic Storage Elements

5 The Concept of Memory

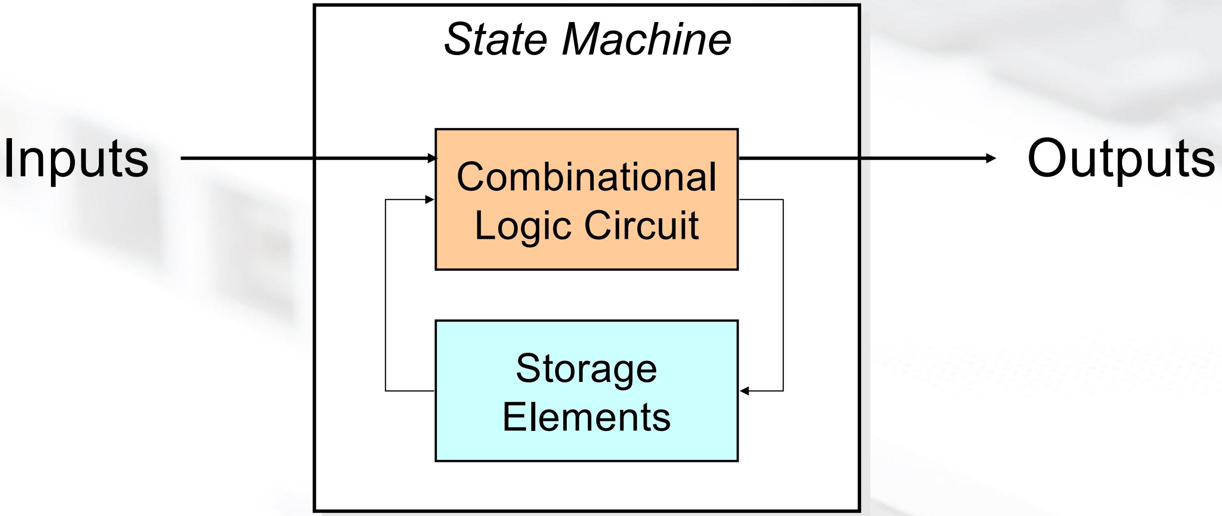
6 **Sequential Logic Circuits**

7 Preview of Coming Attractions: From Logic to Data Path

State Machine

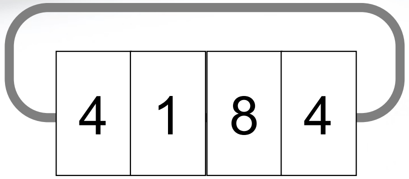
Another type of sequential circuit

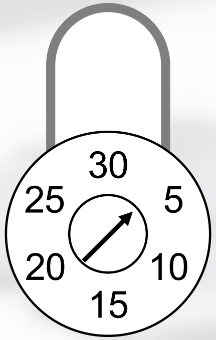
- Combines combinational logic with storage
- "Remembers" state, and changes output (and state) based on inputs and current state



Combinational vs. Sequential

Two types of “combination” locks





Combinational Lock
Success depends only on the values, not the order in which they are set.

Sequential Lock
Success depends on the sequence of values (e.g, R-13, L-22, R-3).

State

The state of a system is a snapshot of all the relevant elements of the system at the moment the snapshot is taken.

Examples:

- The state of a basketball game can be represented by the scoreboard.
  - — Number of points, time remaining, possession, etc.

- The state of a tic-tac-toe game can be represented by the placement of X’s and O’s on the board.

State of Sequential Lock

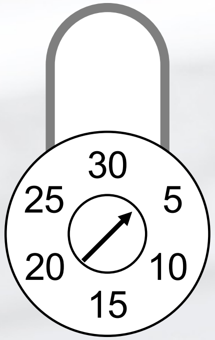
Our lock example has four different states, labelled A-D:

A: **The lock is not open,**
and no relevant operations have been performed.

B: **The lock is not open,**
and the user has completed the R-13 operation.

C: **The lock is not open,**
and the user has completed R-13, followed by L-22.

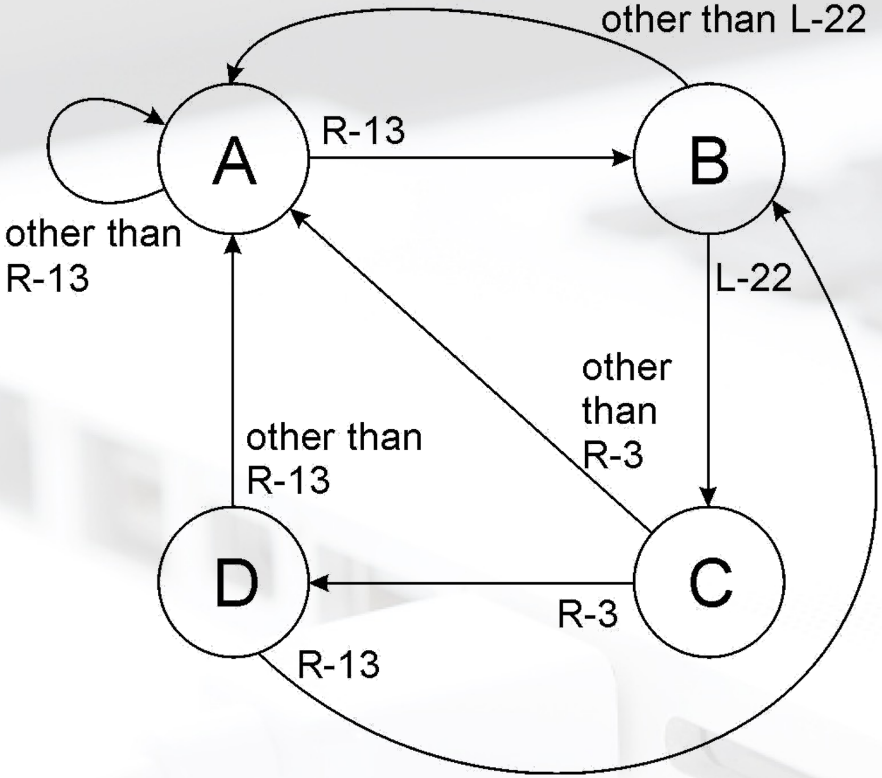
D: **The lock is open.**



Sequential Lock
Success depends on the sequence of values (e.g, R-13, L-22, R-3).

State Diagram of Sequential Lock

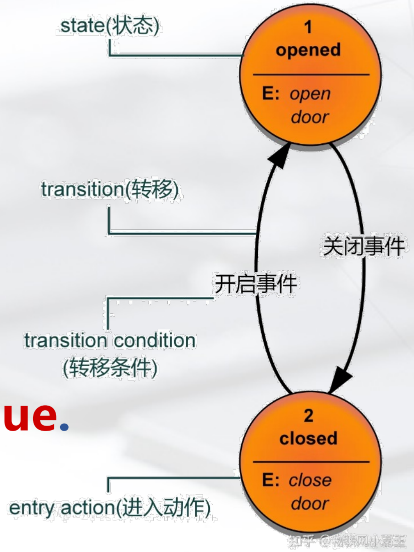
Shows states and actions that cause a transition between states.



Finite State Machine

A description of a system with the following components:

1. **A finite number of states**
2. **A finite number of external inputs**
3. **A finite number of external outputs**
4. **An explicit specification of all state transitions**
5. **An explicit specification of what causes each external output value.**



Often described by a state diagram.

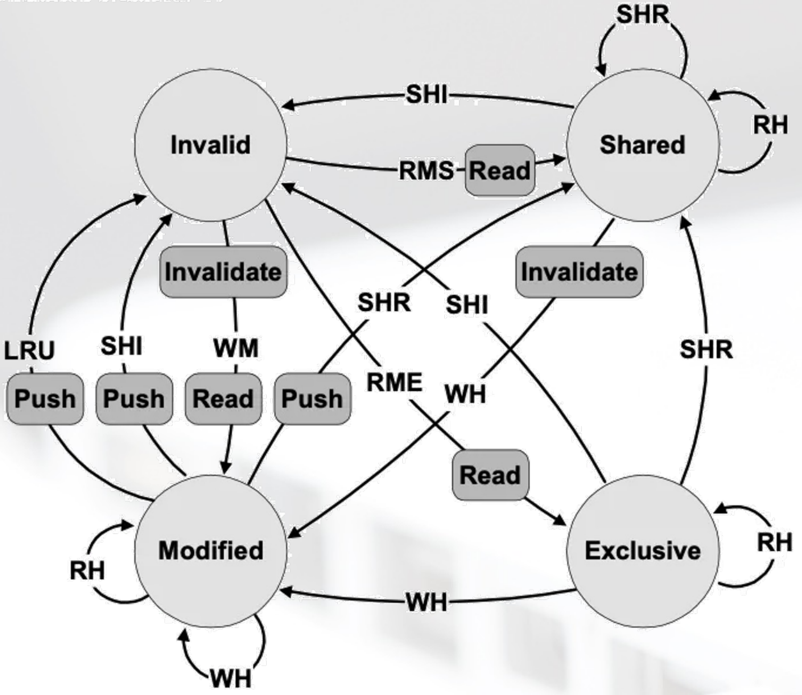
- Inputs may cause state transitions.
- Outputs are associated with each state (or with each transition).

FSM Exercise

- Even number of 0s
- Even number of 0s and even number of 1s
- Binary number dividable by 4
- String contains ‘110110’

FSM Example

MESI State Diagram

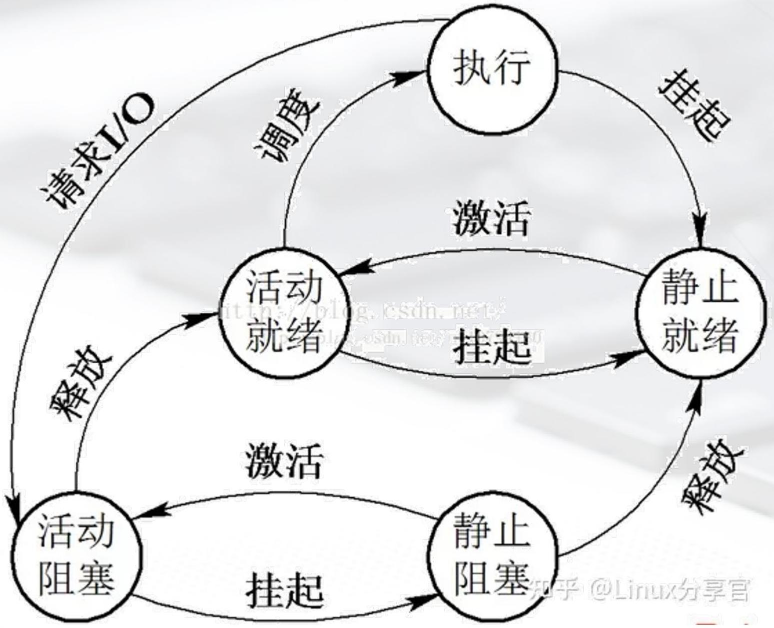


Events:

RH = Read Hit
RMS = Read miss, shared
RME = Read miss, exclusive
WH = Write hit
WM = Write miss
SHR = Snoop hit on read
SHI = Snoop hit on invalidate
LRU = LRU replacement

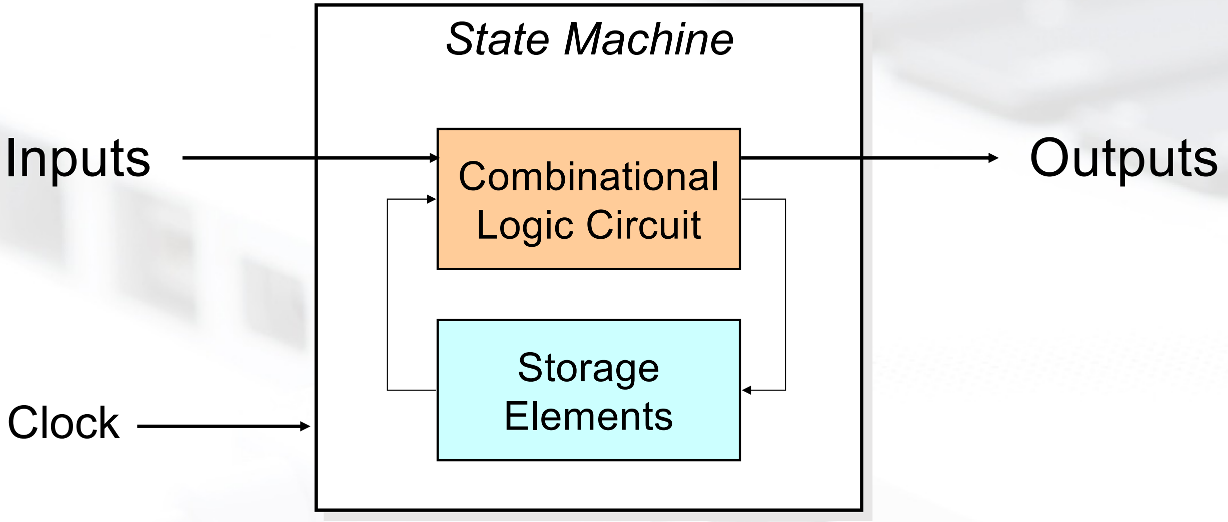
Bus Transactions:

Push = Write cache line back to memory
Invalidate = Broadcast invalidate
Read = Read cache line from memory



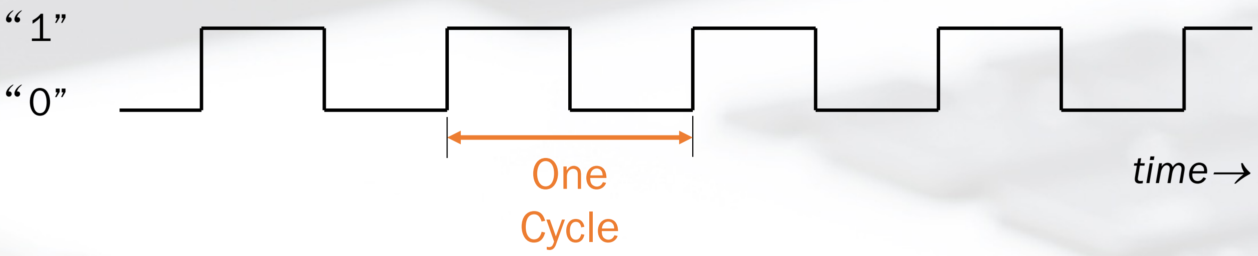
Implementing a Finite State Machine

- **Combinational logic**
  - Determine outputs and next state.
- **Storage elements**
  - Maintain state representation.



The Clock

Frequently, a **clock circuit** triggers transition from one state to the next.

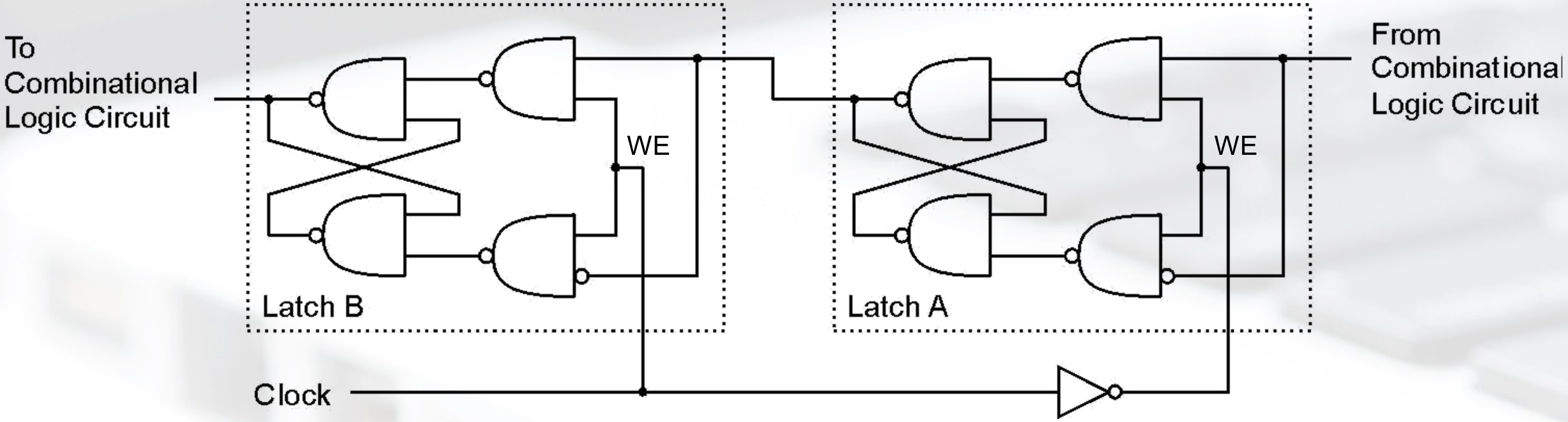


At the beginning of each clock cycle, state machine makes a transition, based on the current state and the external inputs.

- Not always required. In lock example, the input itself triggers a transition.

Storage: Master-Slave Flipflop

A pair of gated D-latches, to isolate *next* state from *current* state.

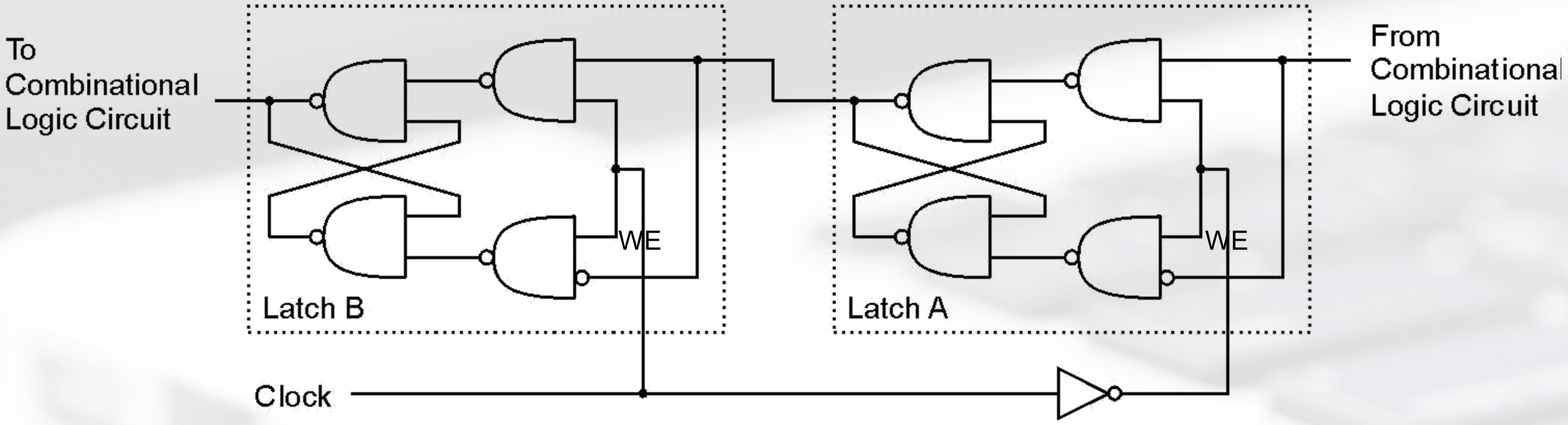


During 1st phase (clock=1), previously-computed state becomes *current* state and is sent to the logic circuit.

During 2nd phase (clock=0), *next* state, computed by logic circuit, is stored in Latch A.

Storage: Master-Slave Flipflop

A pair of gated D-latches, to isolate *next* state from *current* state.



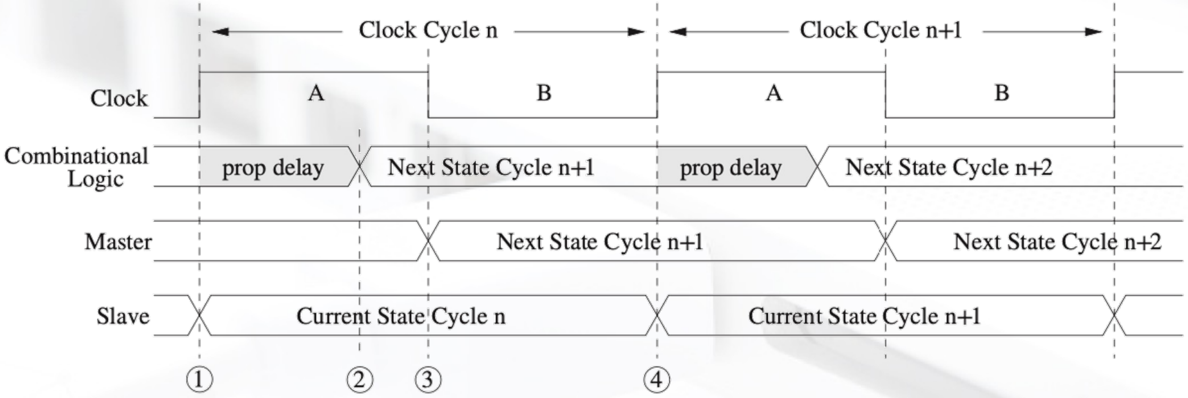


Figure 3.34 Timing diagram for a master/slave flip-flop.

Storage

Each master-slave flipflop stores one state bit.

The number of storage elements (flipflops) needed is determined by the number of states (and the representation of each state).

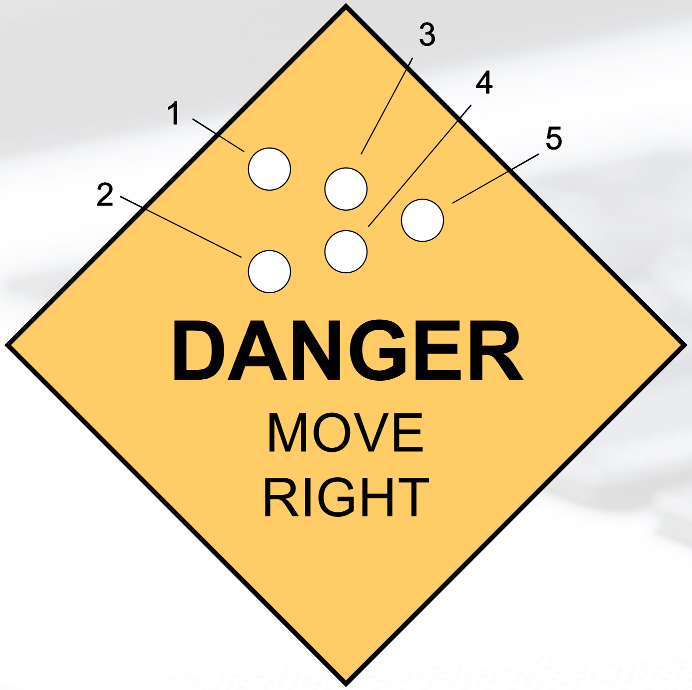
Examples:

- **Sequential lock**
  - —Four states – two bits
- **Basketball scoreboard**
  - —7 bits for each score, 5 bits for minutes, 6 bits for seconds, 1 bit for possession arrow, 1 bit for half, …

Complete Example

A blinking traffic danger sign

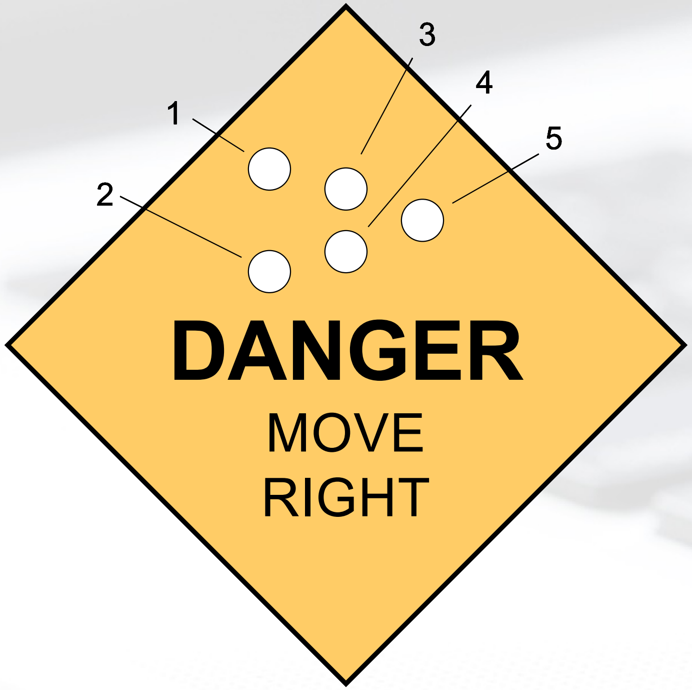
- No lights on
- 1 & 2 on
- 1, 2, 3, & 4 on
- 1, 2, 3, 4, & 5 on
- (repeat as long as switch is turned on)



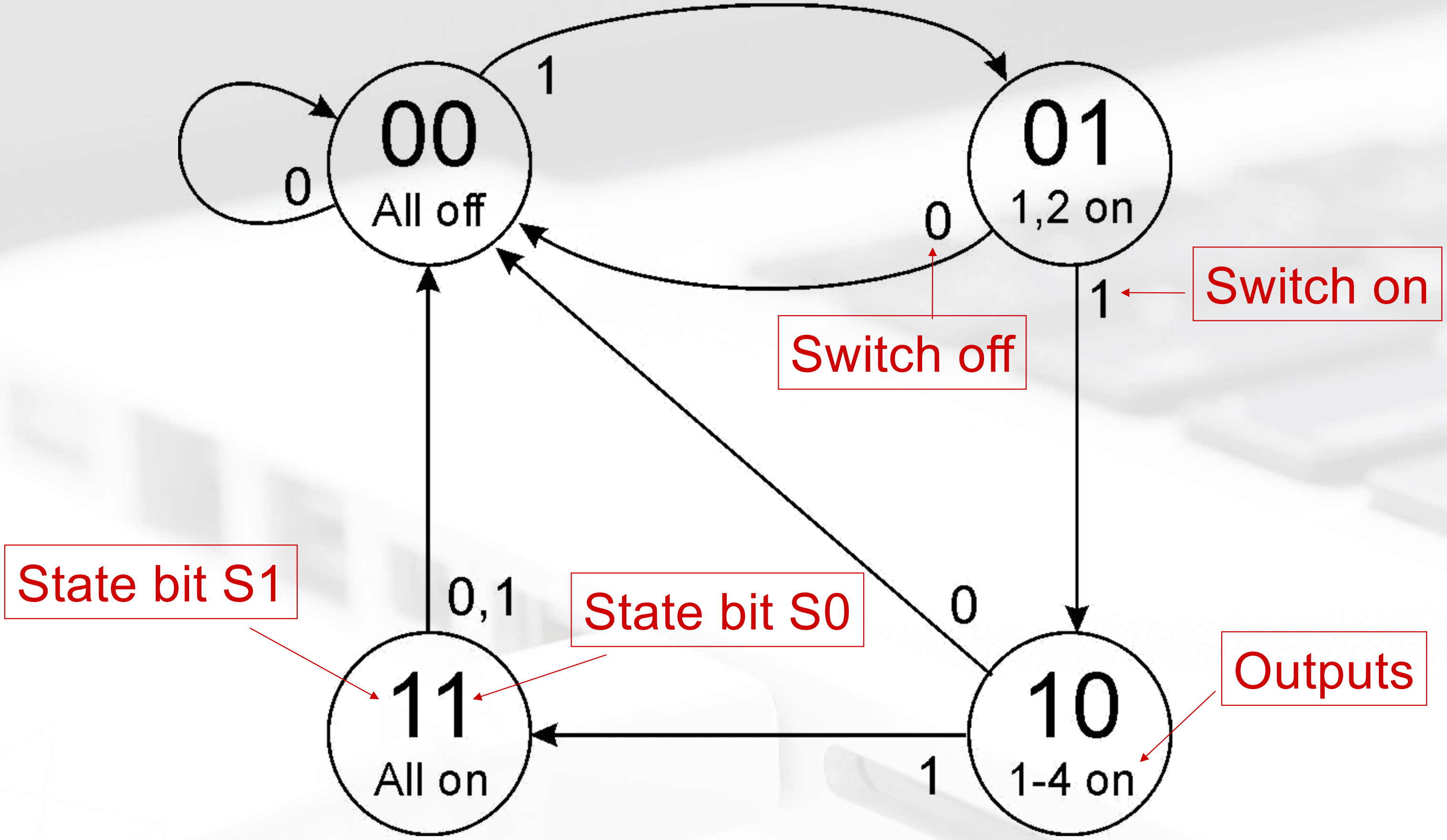
Complete Example

A blinking traffic danger sign

- No lights on
- 1 & 2 on
- 1, 2, 3, & 4 on
- 1, 2, 3, 4, & 5 on
- (repeat as long as switch is turned on)

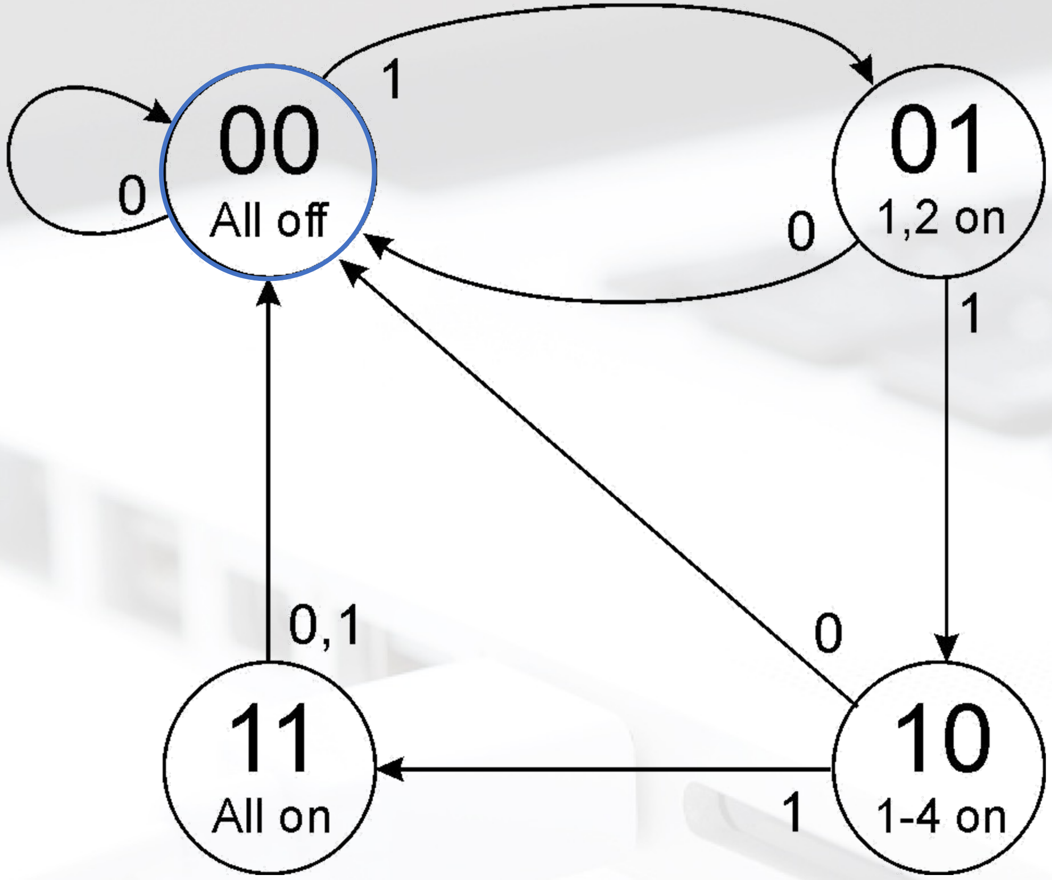


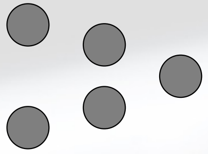
Traffic Danger Sign State Diagram



Transition on each clock cycle.

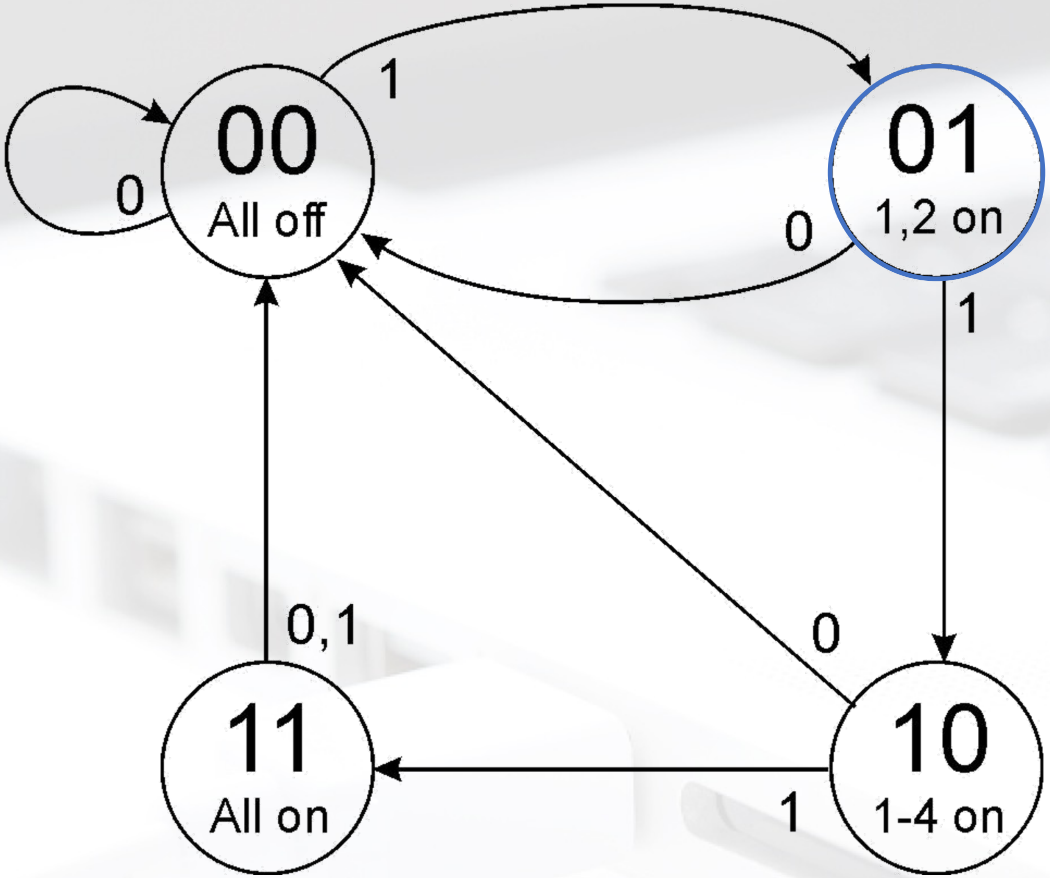
Traffic Danger Sign State Diagram: State 00

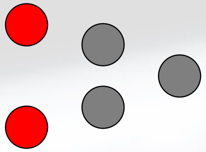




Transition on each clock cycle.

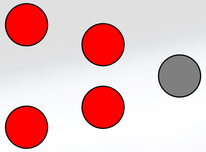
Traffic Danger Sign State Diagram: State 01

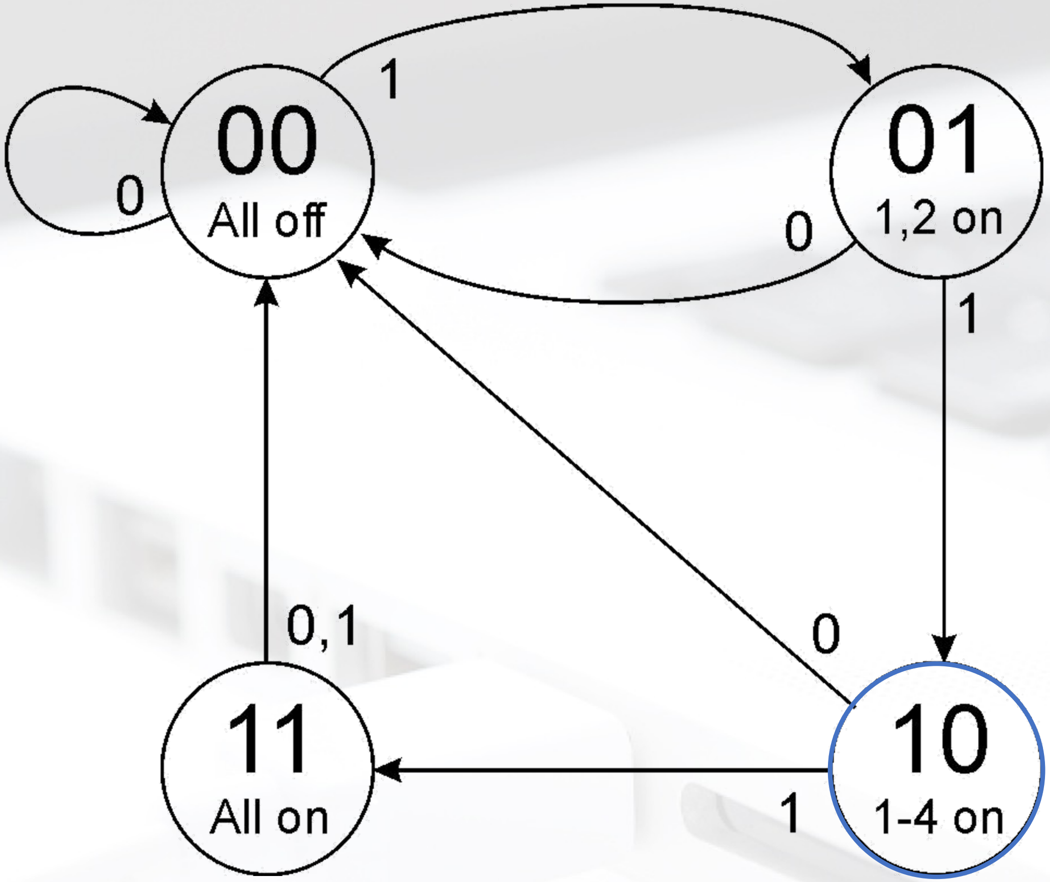




Transition on each clock cycle.

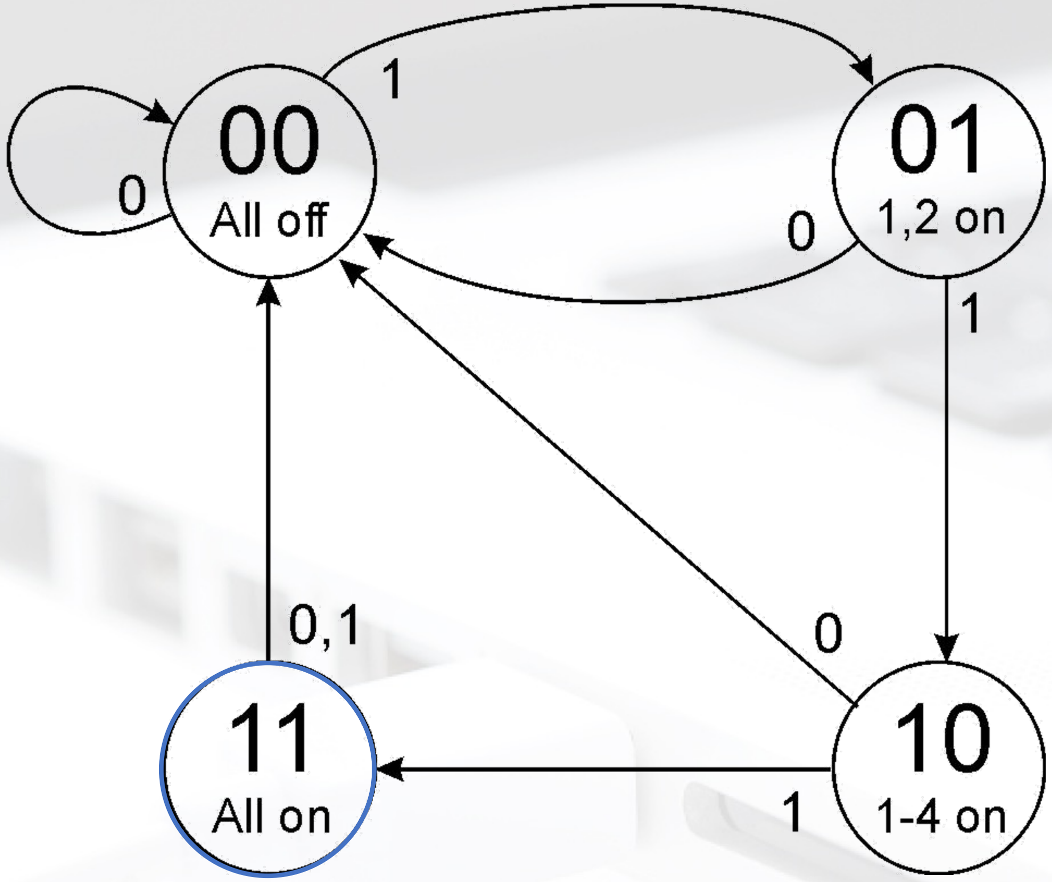
Traffic Danger Sign State Diagram: State 10

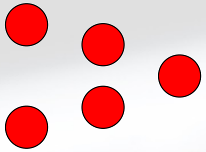




Transition on each clock cycle.

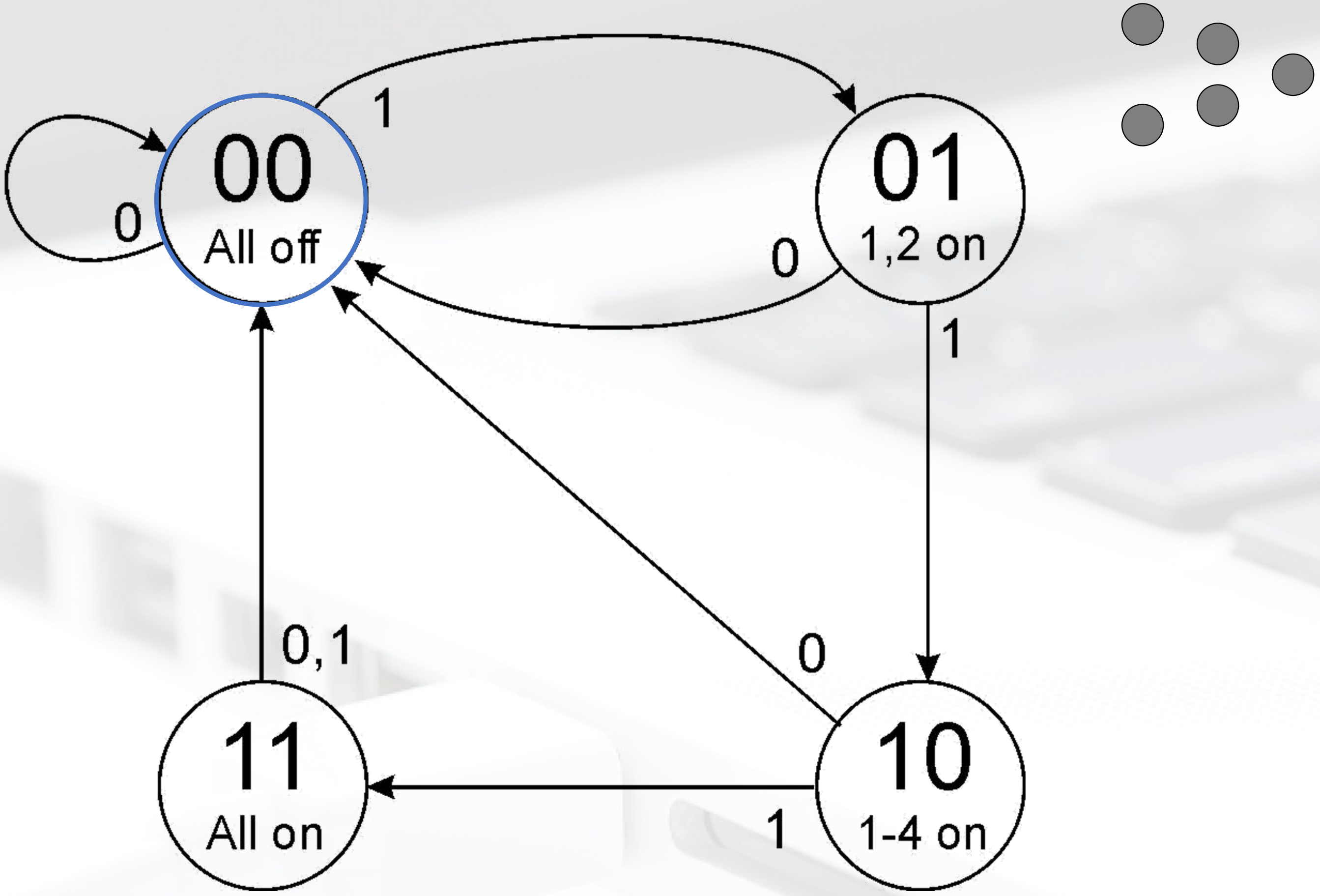
Traffic Danger Sign State Diagram: State 11





Transition on each clock cycle.

Traffic Danger Sign State Diagram: State 00



Transition on each clock cycle.

Traffic Danger Sign Truth Tables

Outputs
(depend only on state: S1S0)

Lights 1 and 2

Lights 3 and 4

Light 5

S_1	S_0	Z	Y	X
0	0	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	1	1	1

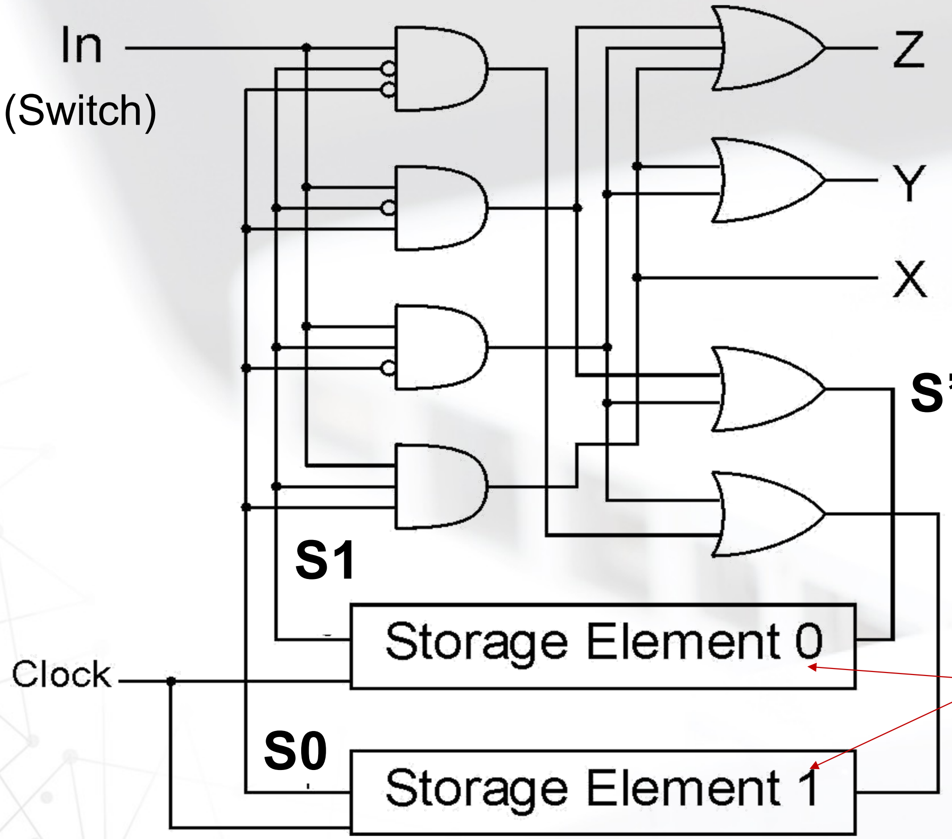
Next State: S1’S0’
(depend on state and input)

Switch

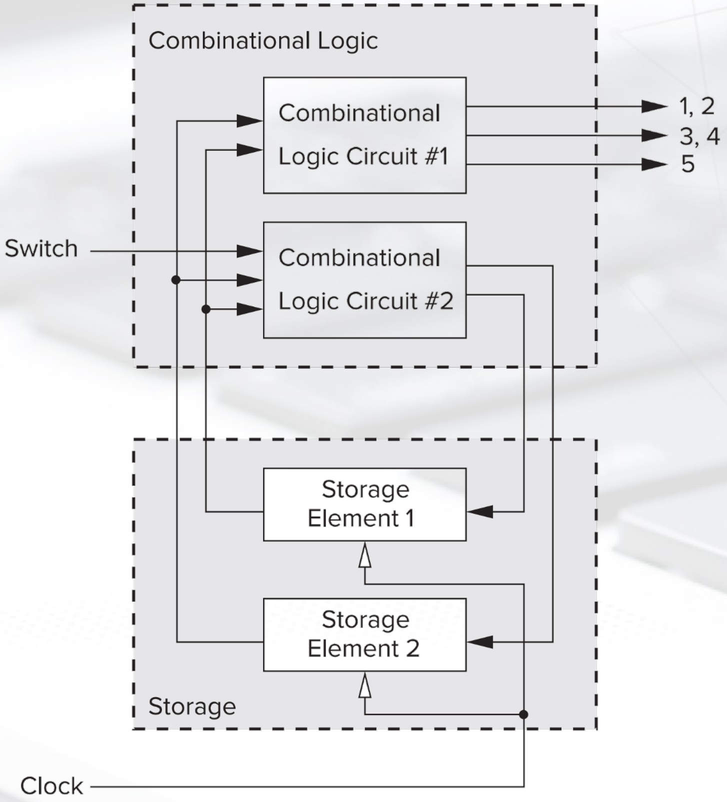
In	S_1	S_0	S_1’	S_0’
0	X	X	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Whenever In=0, next state is 00.

Traffic Danger Sign Logic



Master-slave flipflop



Outline

1 The Transistor

2 Logic Gates

3 Combinational Logic Circuits

4 Basic Storage Elements

5 The Concept of Memory

6 Sequential Logic Circuits

7 Preview of Coming Attractions: From Logic to Data Path

From Logic to Data Path

The data path of a computer is all the logic used to process information.

- See the data path of the LC-3 on next slide.

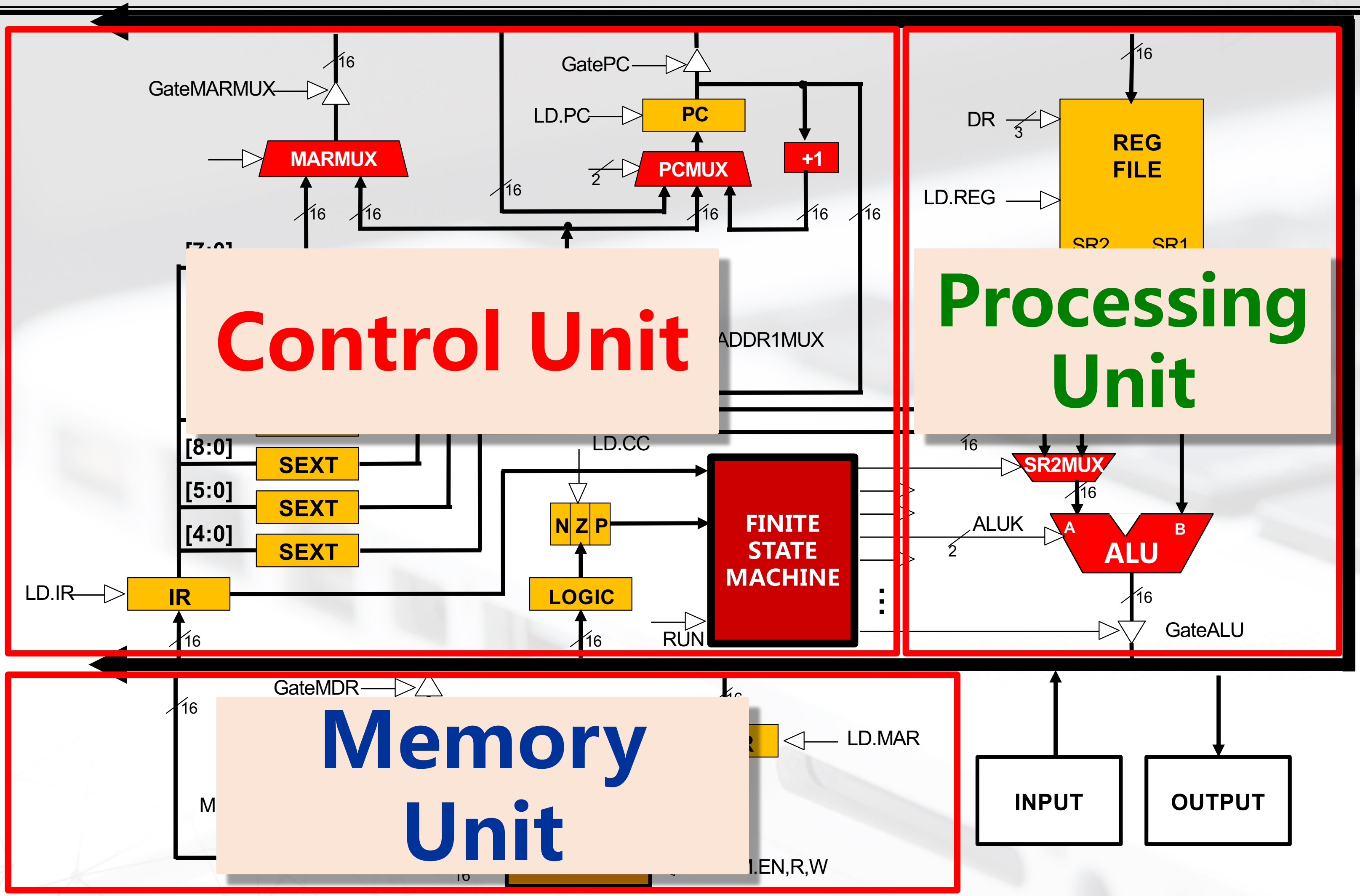
Combinational Logic

- Decoders -- convert instructions into control signals
- Multiplexers -- select inputs and outputs
- ALU (Arithmetic and Logic Unit) -- operations on data

Sequential Logic

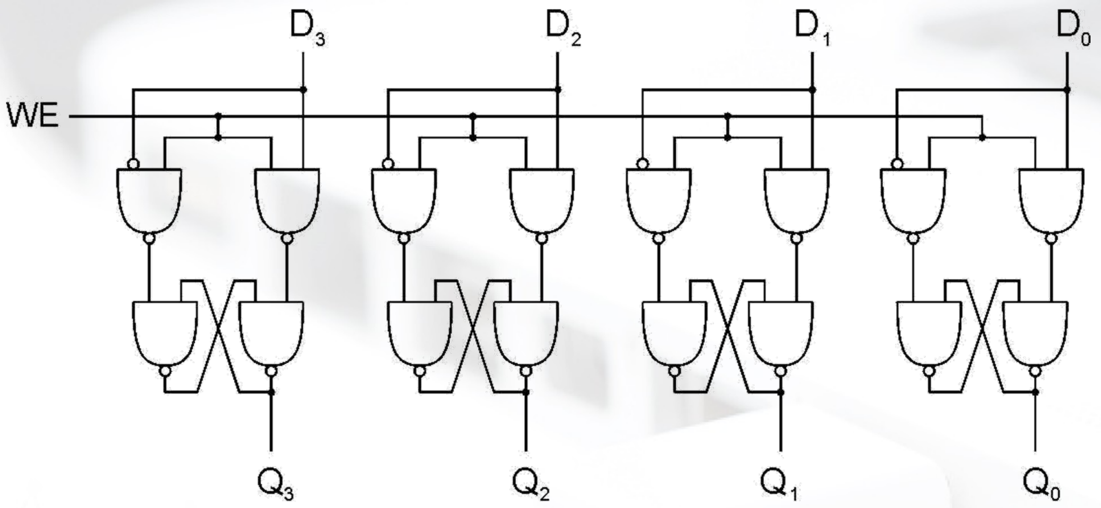
- State machine -- coordinate control signals and data movement
- Registers and latches -- storage elements

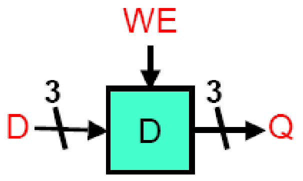
LC-3 Data Path Overview (Microarchitecture)



Register

- **A register stores a multi-bit value.**
  - We use a collection of D-latches, all controlled by a common WE.
  - When WE=1, n-bit value D is written to register.





Register

A four-bit register

- We use a collection of flip-flops instead of D-latches
- Read the contents of a register throughout a clock cycle
- And store a new value in the register at the end of that same clock cycle

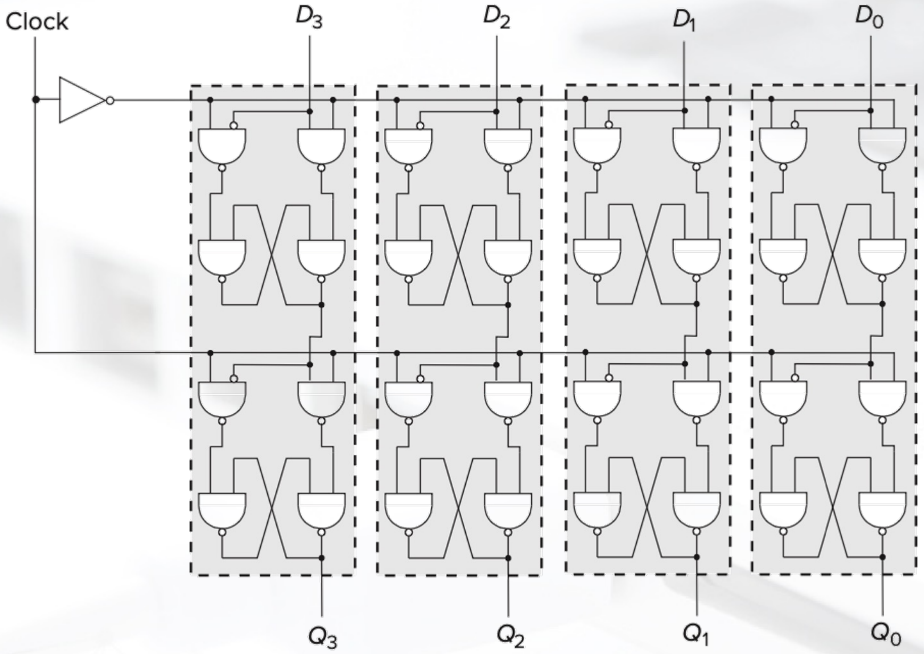


Figure 3.36 A four-bit register.

Summary

- We’ve touched on basic digital logic
  - Transistors
  - Gates
  - Storage (latches, flip-flops, memory)
  - State machines
- Built some simple circuits
  - adder, subtracter, adder/subtracter, Incrementer
  - Counter (consisting of register and incrementer)
  - Hard-coded traffic sign state machine
  - Programmable traffic sign state machine
- Up next: a computer as a (simple?) state machine

LC-3 Data Path