310-2202 โครงสร้างของระบบคอมพิวเตอร์
(Computer Organization)

Topic 3: Digital logic

Damrongrit Setsirichok

Topic

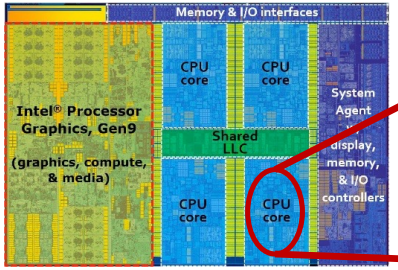
- Transistors & Logic Gates
- Combinational Logic Circuits
- Basic Storage Elements
- Sequential Logic Circuits

Transistor Count

- **Microprocessors contain millions of transistors**
  - **AMD Instinct: 146 Billion – 5nm technology**
  - **Apple M3 Max: 97 Billion – 3nm technology**
  - **Intel 7 Sapphire Rapids – 10nm technology**
- Transistor: Building Block of Computers
  - Logically, each transistor acts as a switch
- Combined to implement logic functions
  - AND, OR, NOT

https://en.wikipedia.org/wiki/Transistor_count

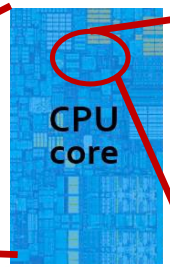
Approach: Bottom Up

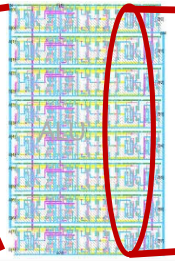


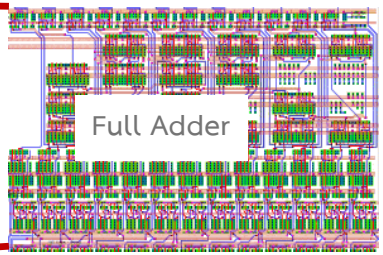
Integrated Circuit Design

100 Modules/ IC

0.25M~20G Devices



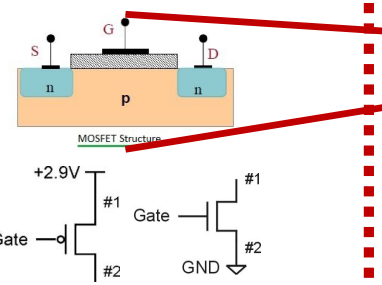




Register Transfer Level (RTL) Design

1K~10K Cells/Module

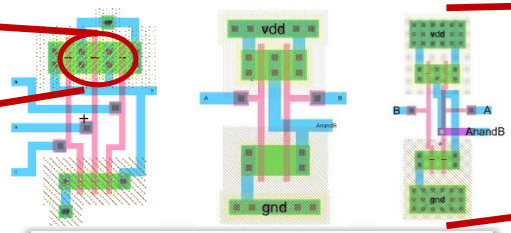
(100K Devices)



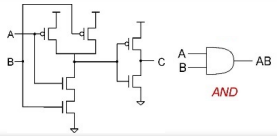
Transistor Physical Layout

Scheme for

Representing Information



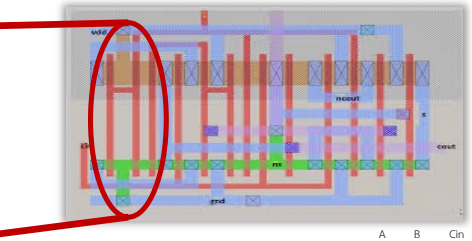
Gate Level Design

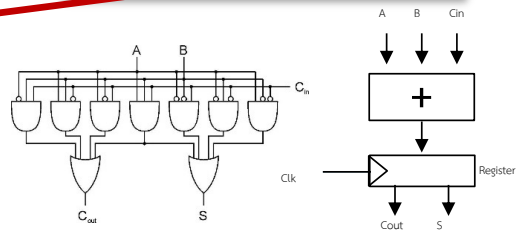


Circuit Level Design

(Transistor Level Design)

(2~8 Devices/Gate)





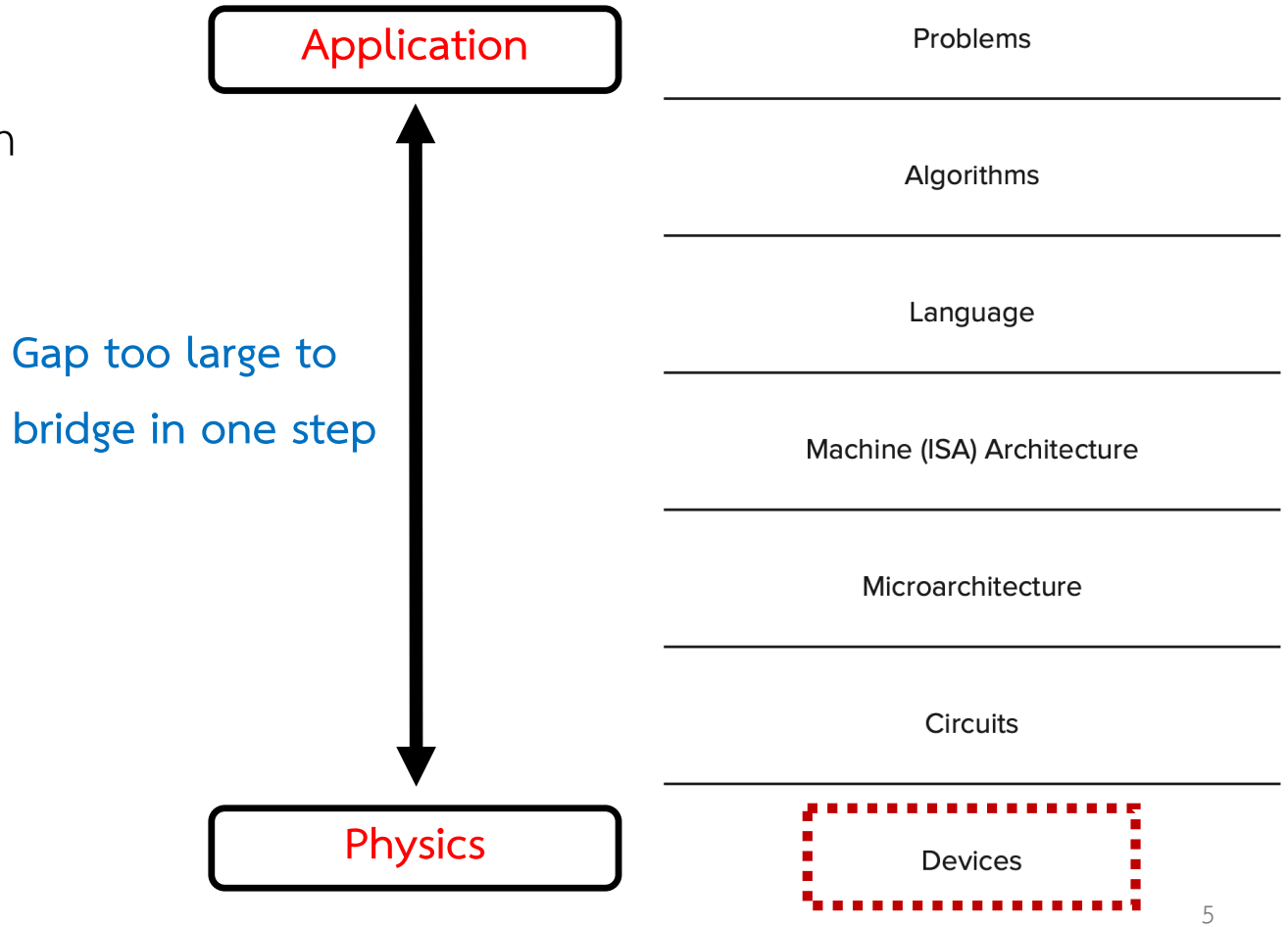
Register Transfer Level (RTL) Design

2~16 Gates/Cell

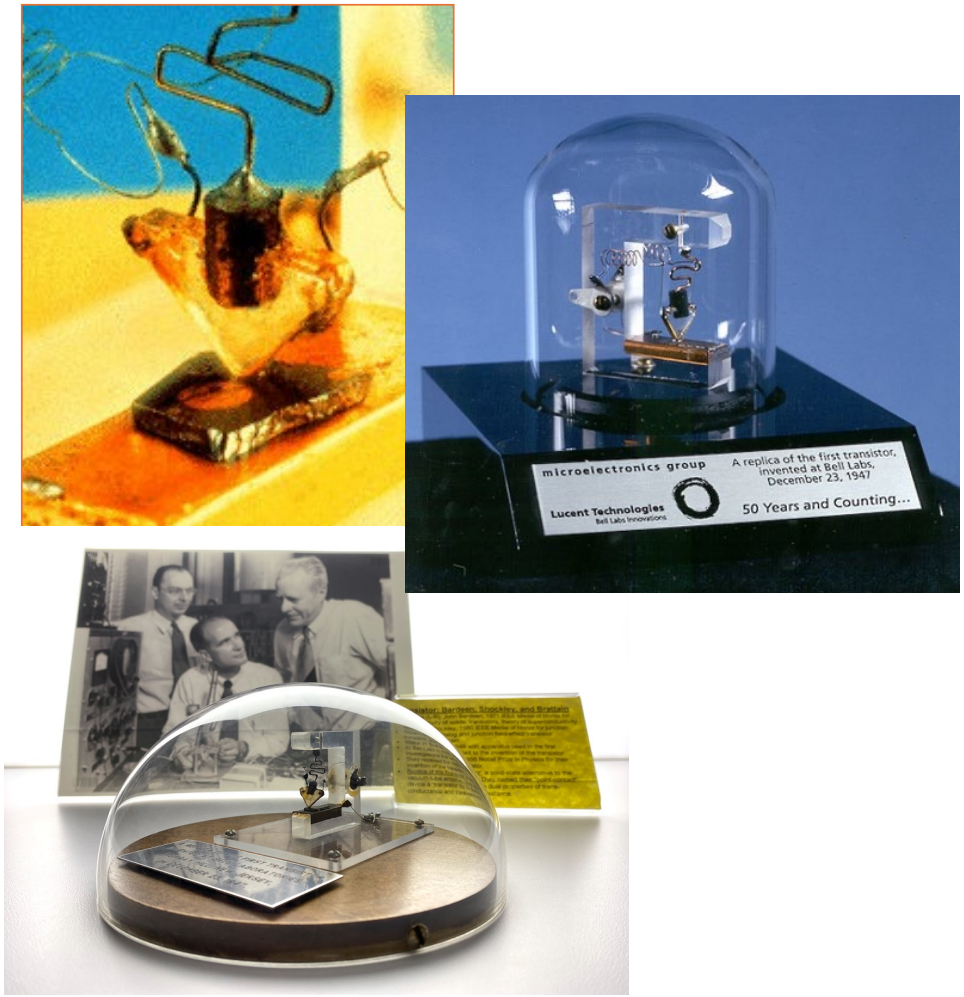
(16~64 Devices)

How Do We Get the Electrons to Do the Work?

- Statement of the Problem
- Algorithm
- Program
- ISA
- Microarchitecture
- Logic Circuit
- Devices

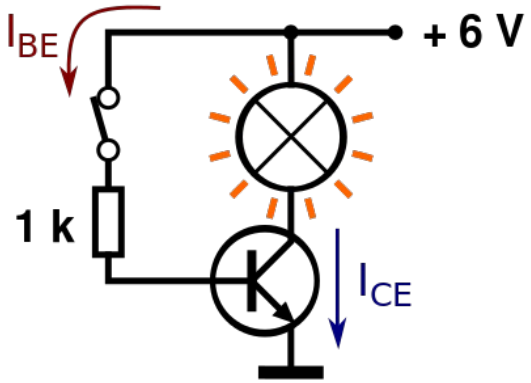


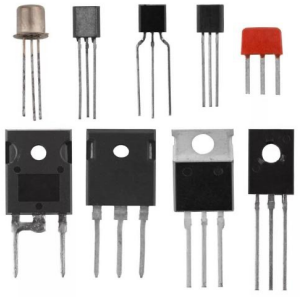
First use: Vacuum tubes as switches

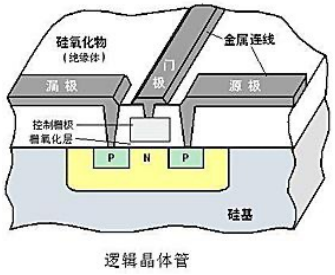


- **1945**: Bell sets up lab in the hopes of developing “solid state” components to replace existing electromechanical systems.
- **1947**: The Invention of the First Transistor-the point-contact transistor
- **1951**: Shockley develops junction transistor which can be manufactured in quantity.

Transistor as a switch







逻辑晶体管

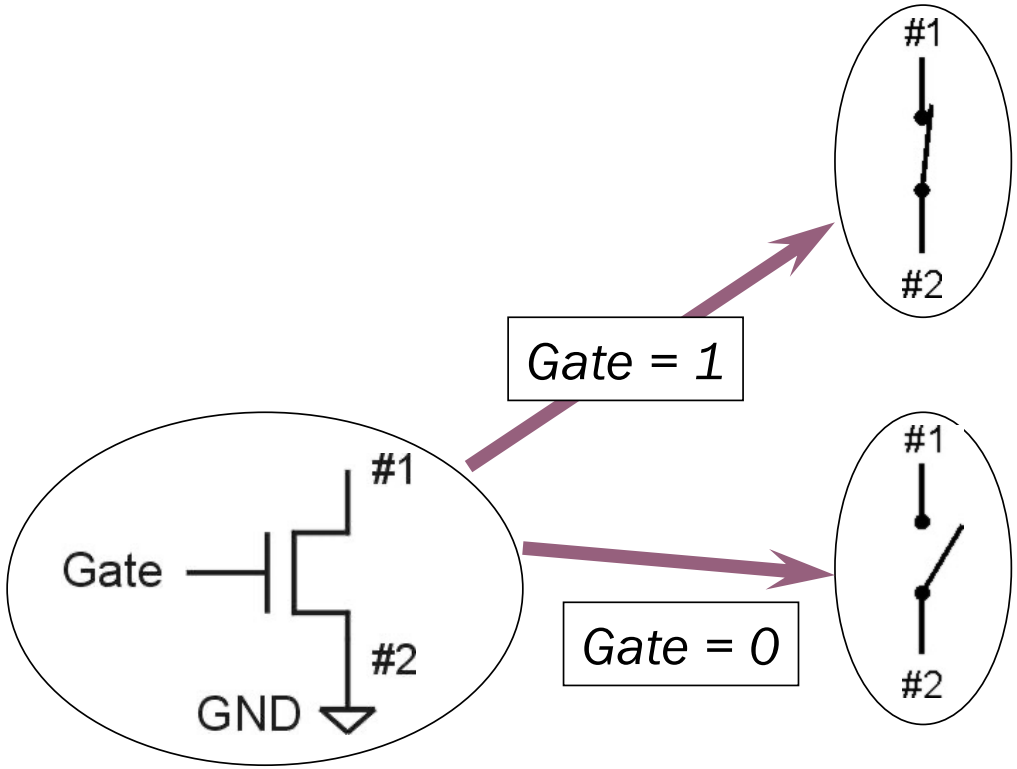
- Switch **open**:
  - No current through circuit
  - Light is off
  - V_{out} is +2.9 V
- Switch **closed**:
  - Short circuit across switch
  - Current flows
  - Light is on
  - V_{out} is 0 V

Switch-based circuits can easily represent two states:

on/off, open/closed, voltage/no voltage.

N-type MOS Transistor

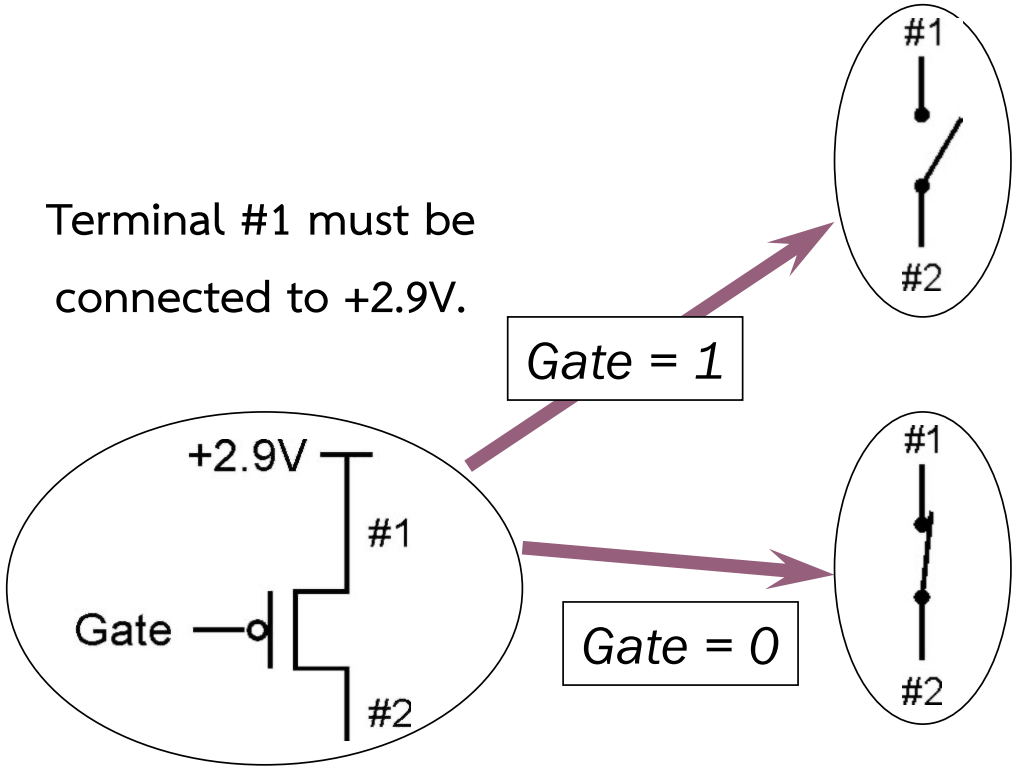
- MOS: Metal–Oxide–Semiconductor
  - two types: N-type and P-type
- N-type
  - when Gate has positive voltage, short circuit between #1 and #2 (switch closed)
  - when Gate has zero voltage, open circuit between #1 and #2 (switch open)

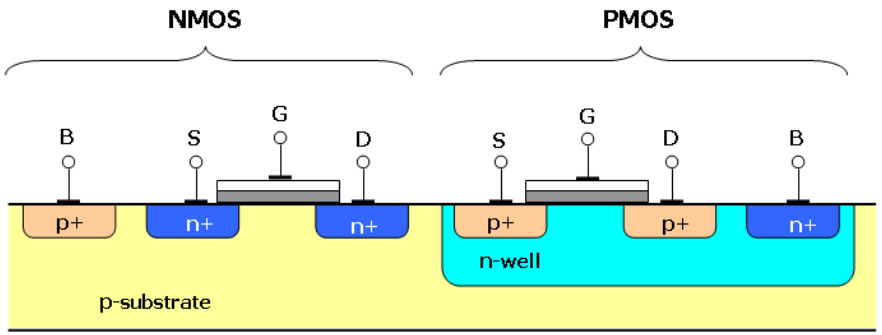


Terminal #2 must be connected to GND (0V).

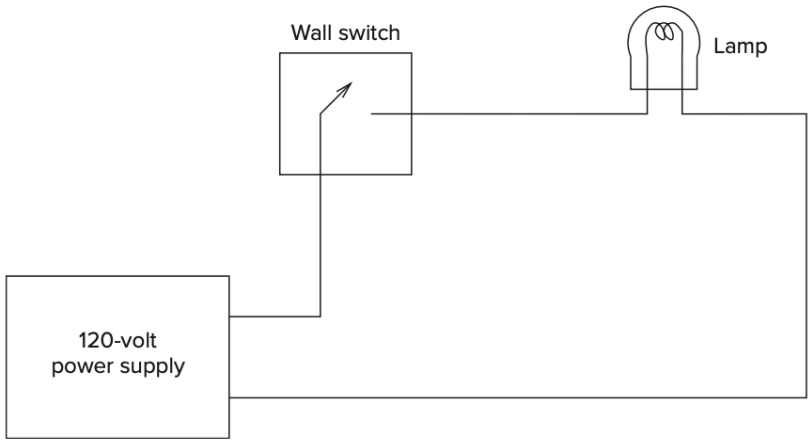
P-type MOS Transistor

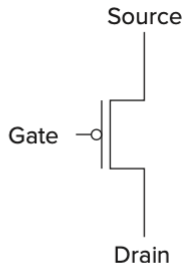
- **P-type** is complementary to N-type
  - when Gate has positive voltage, open circuit between #1 and #2 (switch open)
  - when Gate has zero voltage, short circuit between #1 and #2 (switch closed)



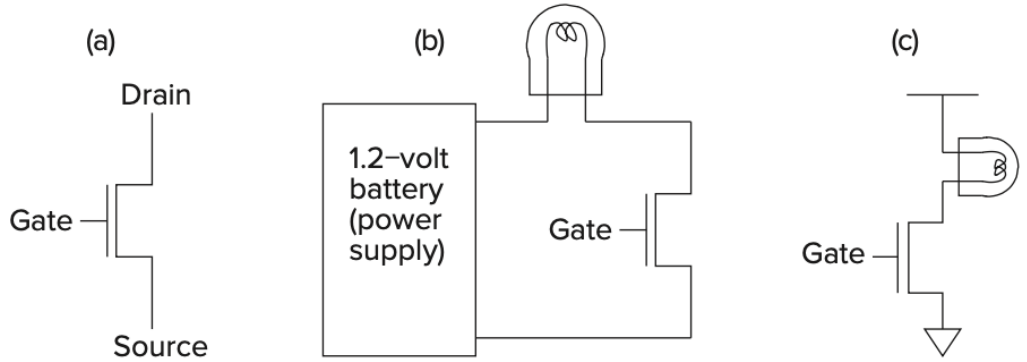


Transistor as a switch





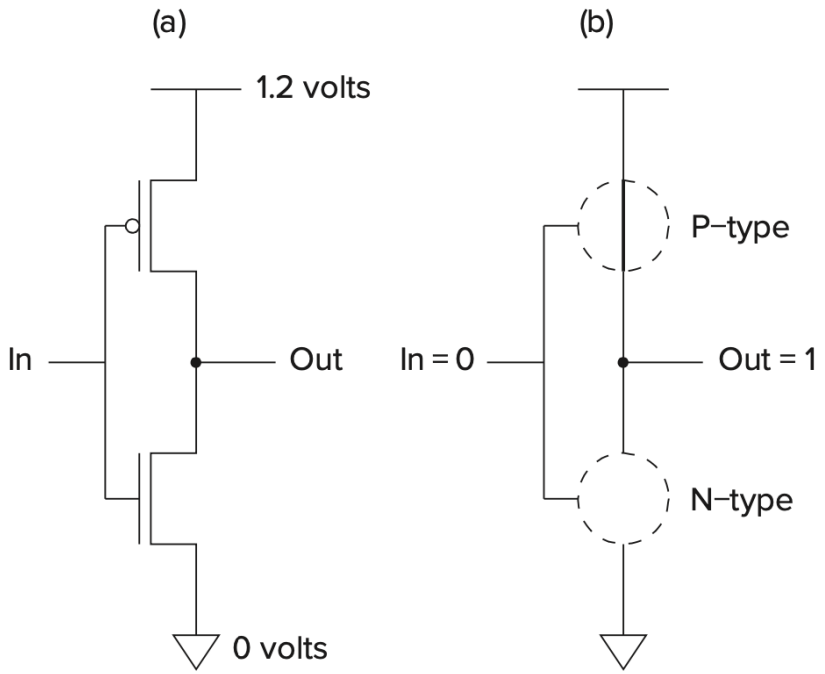
A P-type MOS transistor.



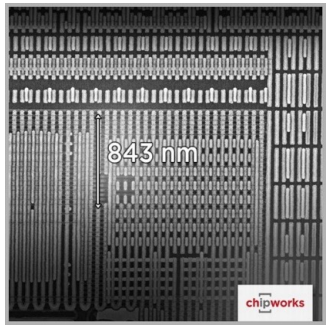
The N-type MOS transistor.

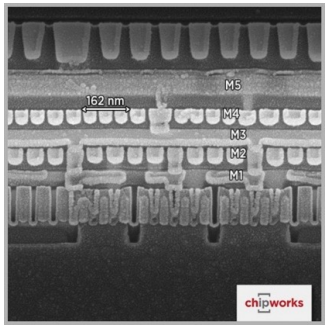
CMOS Circuit

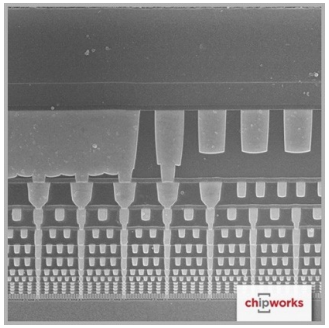
- **CMOS: Complementary Metal-Oxide Semiconductor**
- Contain both N-type and P-type MOS transistors
  - For all inputs, make sure that output is either connected to **GND** or **+**, **not both**

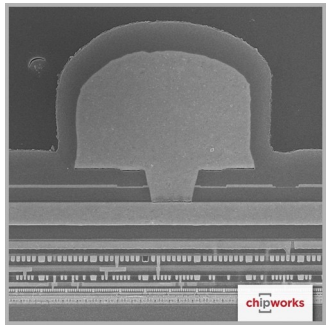


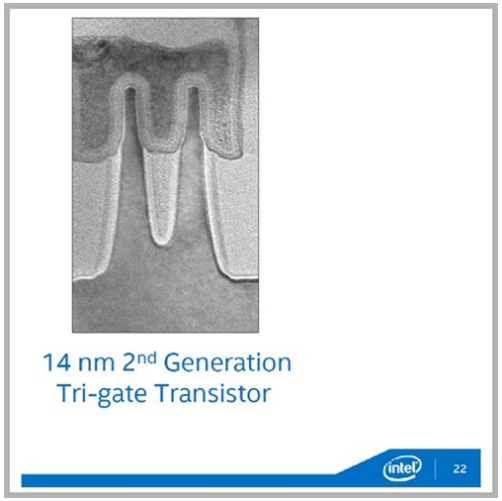
A transistor under a microscope

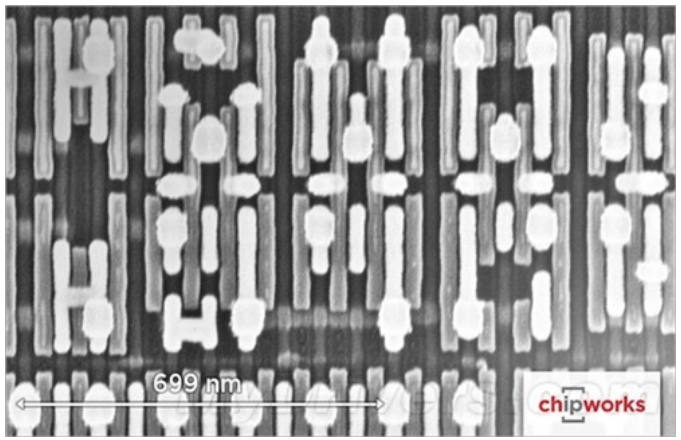


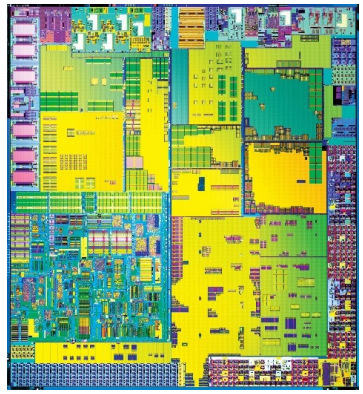








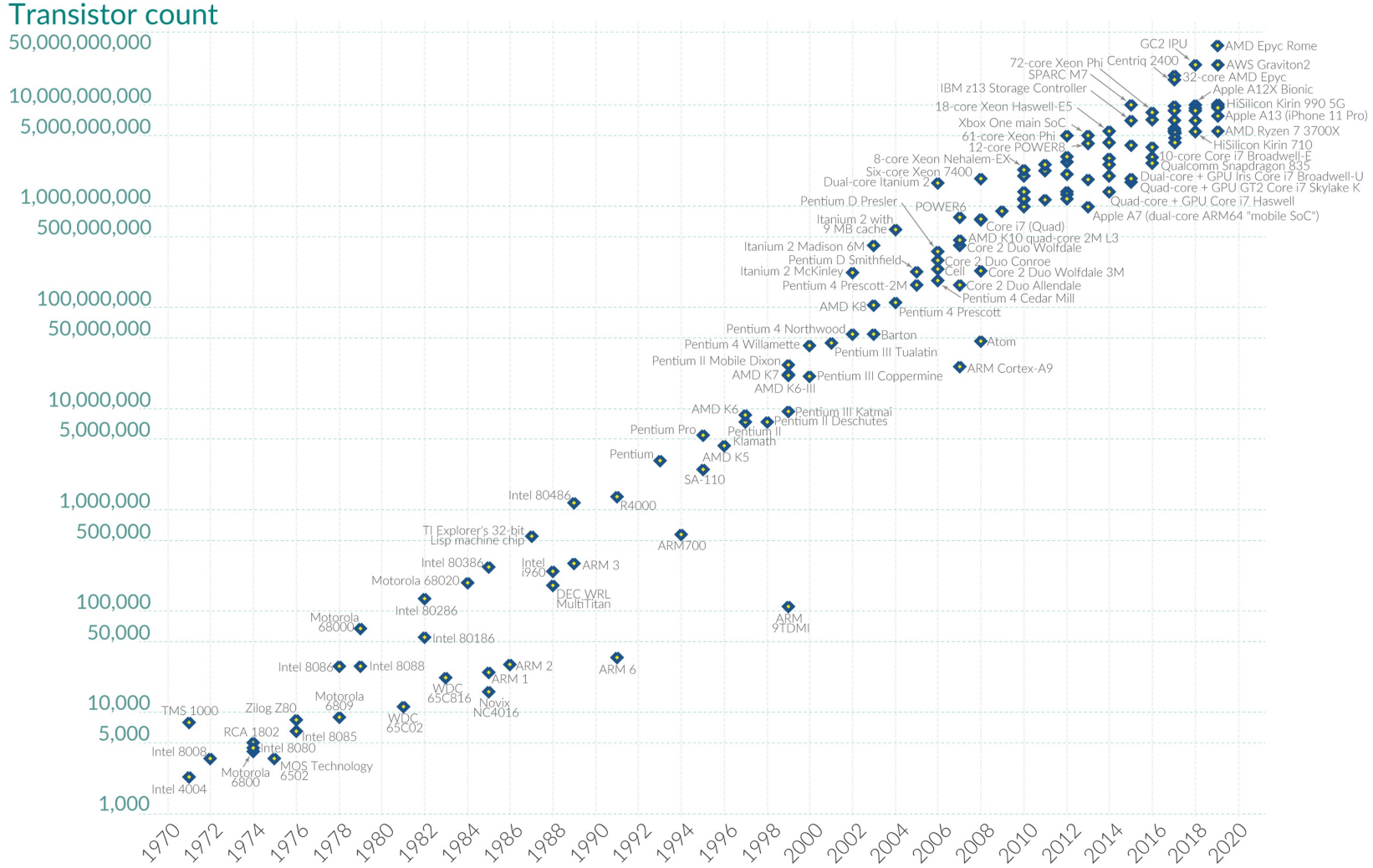




Moore's Law: The number of transistors on microchips doubles every two years

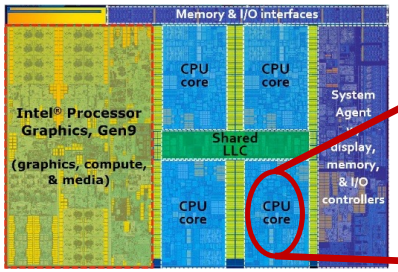
Our World in Data

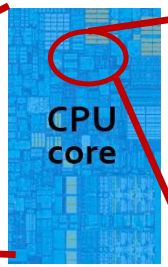
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

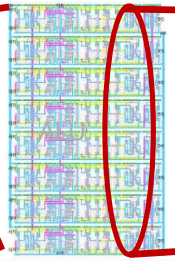


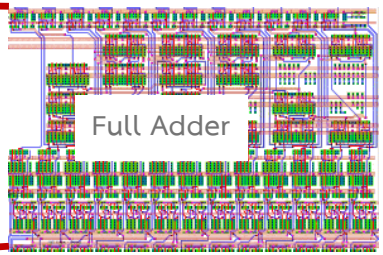
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

Approach: Bottom Up









Integrated Circuit Design

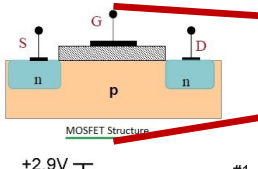
100 Modules/ IC

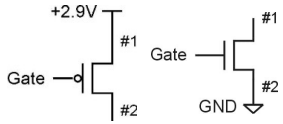
0.25M~20G Devices

Register Transfer Level (RTL) Design

1K~10K Cells/Module

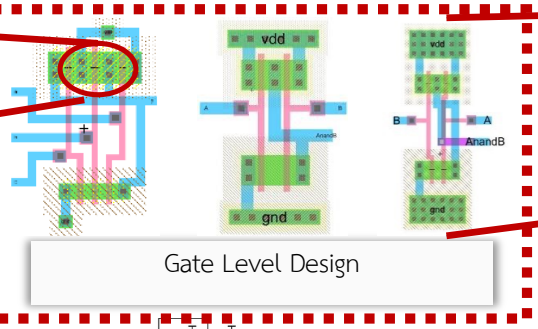
(100K Devices)

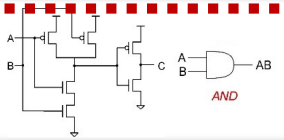




Transistor Physical Layout

Scheme for Representing Information

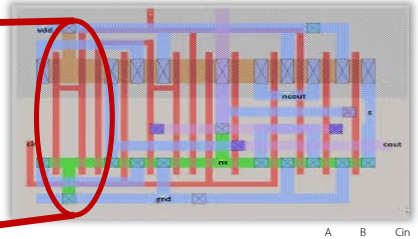


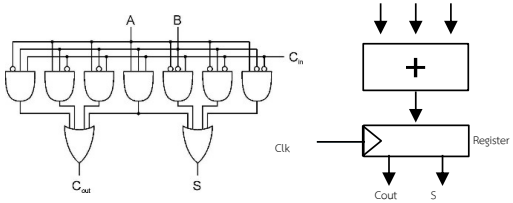


Circuit Level Design

(Transistor Level Design)

(2~8 Devices/Gate)



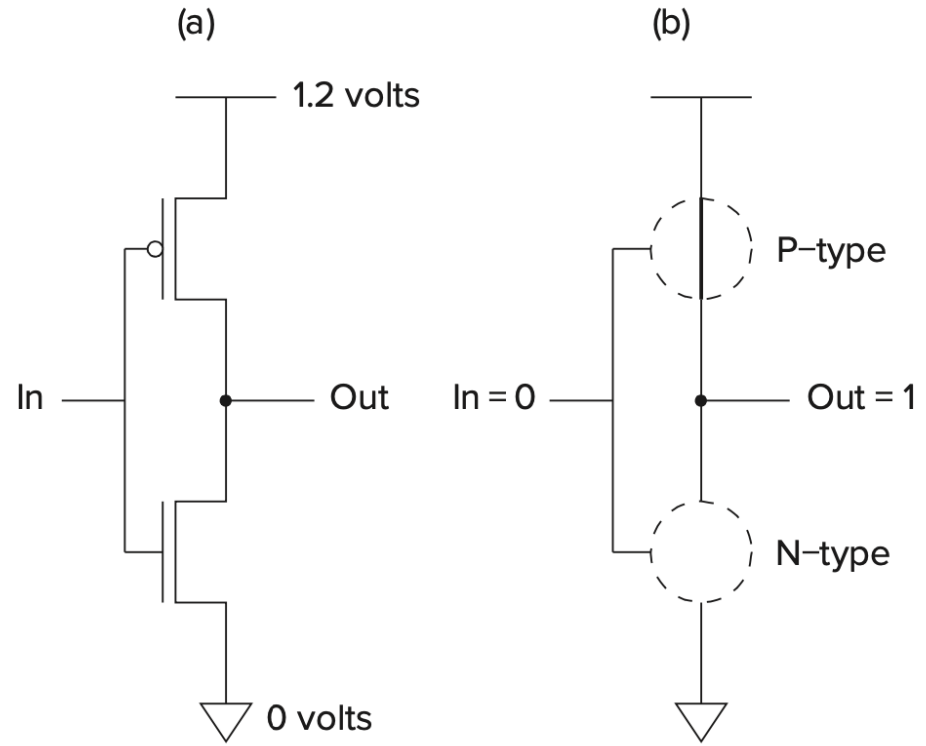


Register Transfer Level (RTL) Design

2~16 Gates/Cell

(16~64 Devices)

Inverter (NOT Gate)



(c)

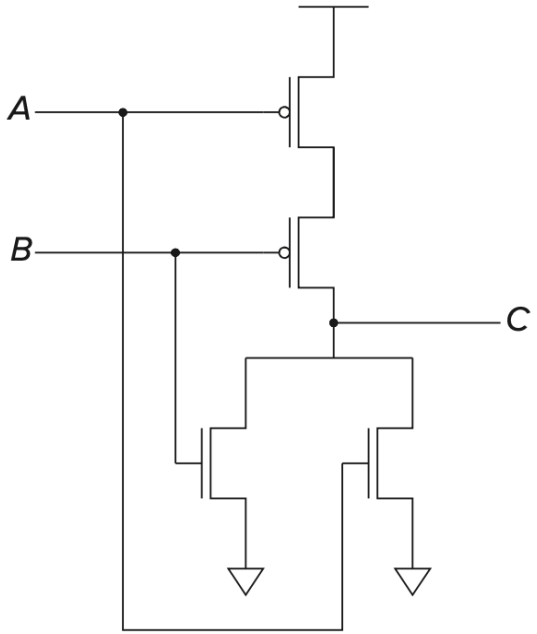
In	Out
0 volts	1.2 volts
1.2 volts	0 volts

(d)

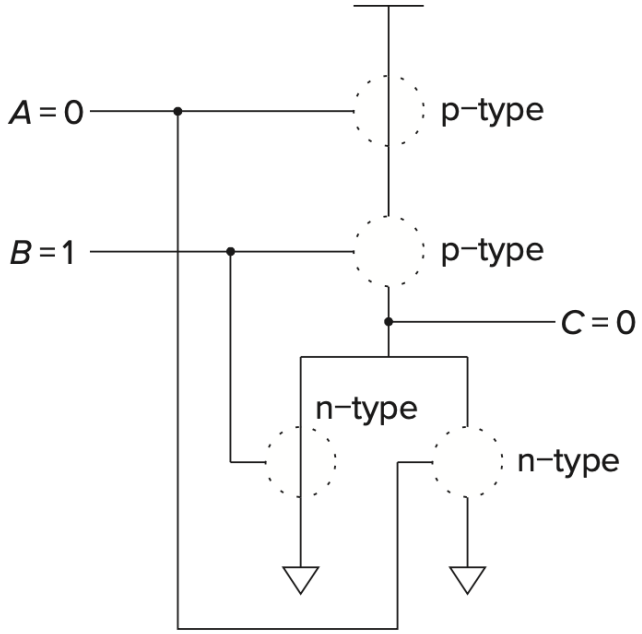
In	Out
0	1
1	0

A CMOS inverter.

(a)



(b)



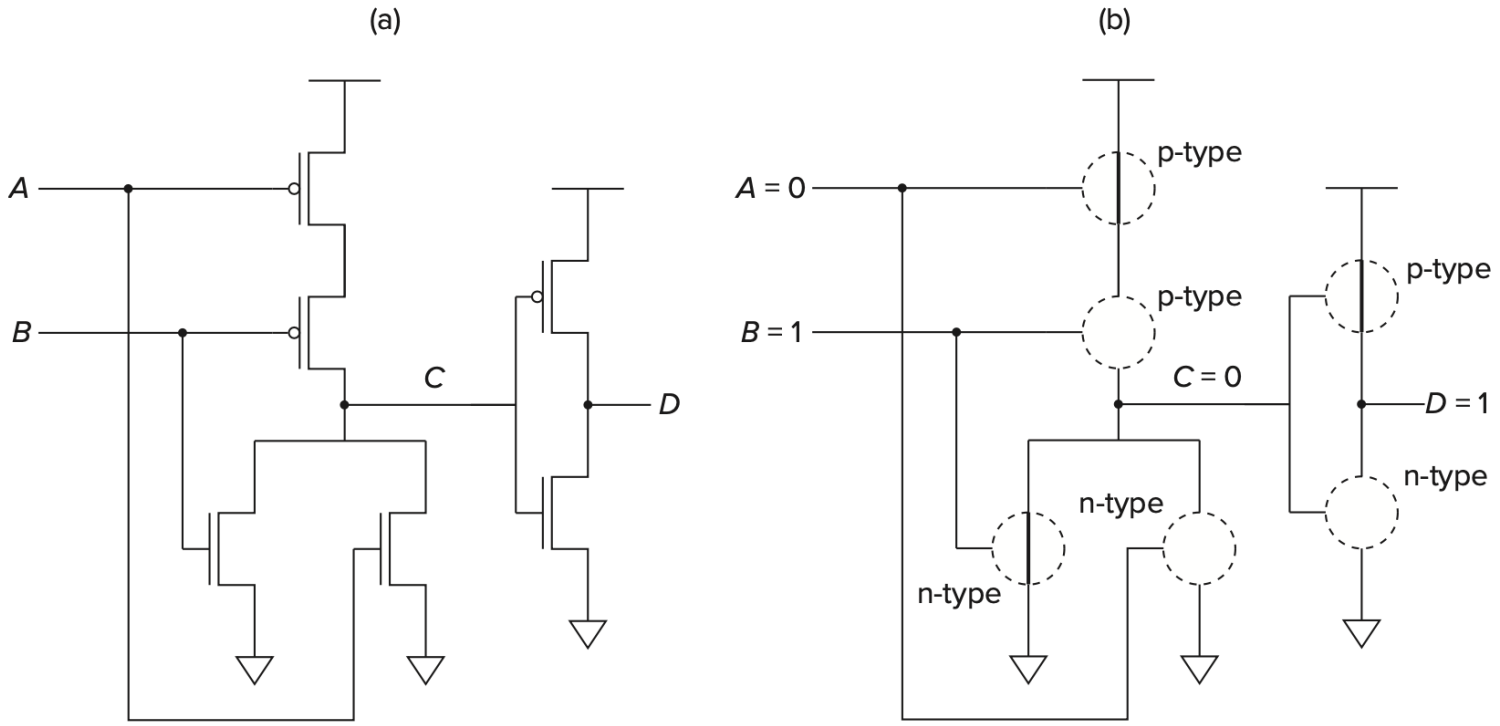
(c)

A	B	C
0 volts	0 volts	1.2 volts
0 volts	1.2 volts	0 volts
1.2 volts	0 volts	0 volts
1.2 volts	1.2 volts	0 volts

(d)

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

Figure 3.5 **The NOR gate.**

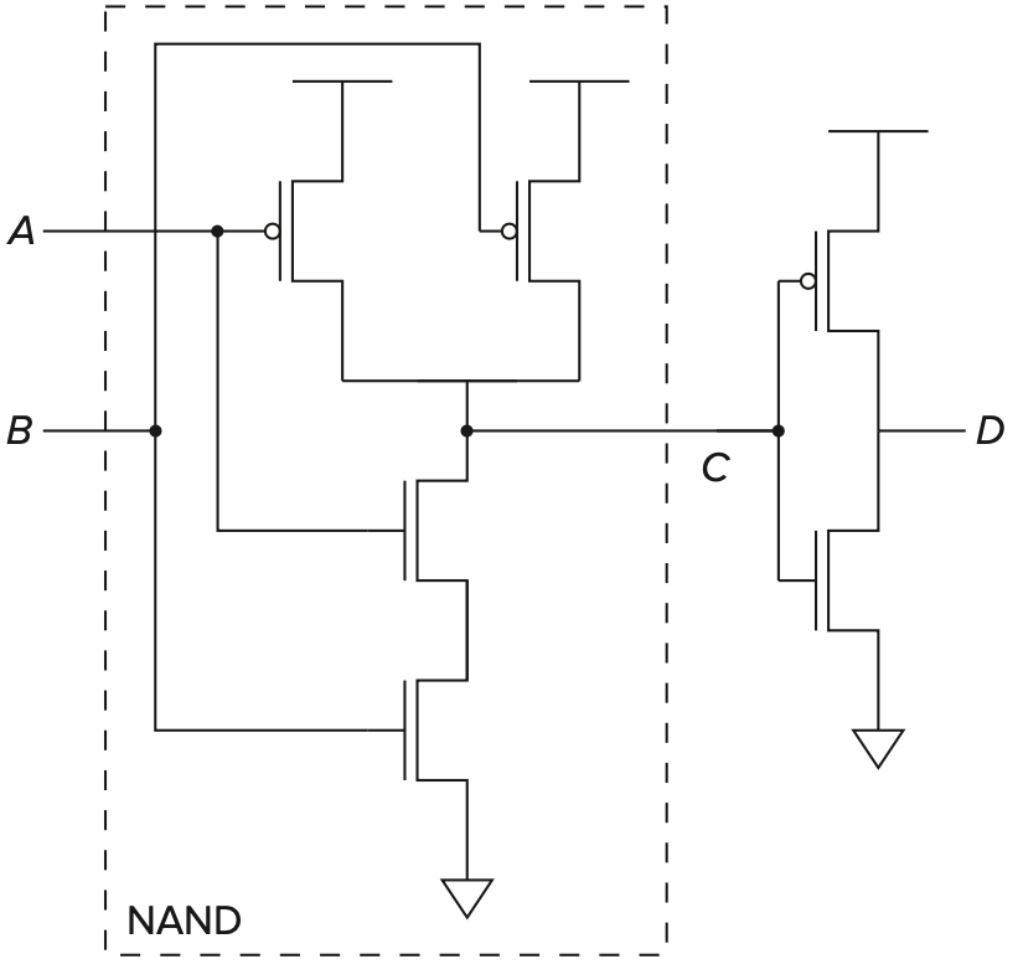


(c)

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

Figure 3.6 **The OR gate.**

(a)

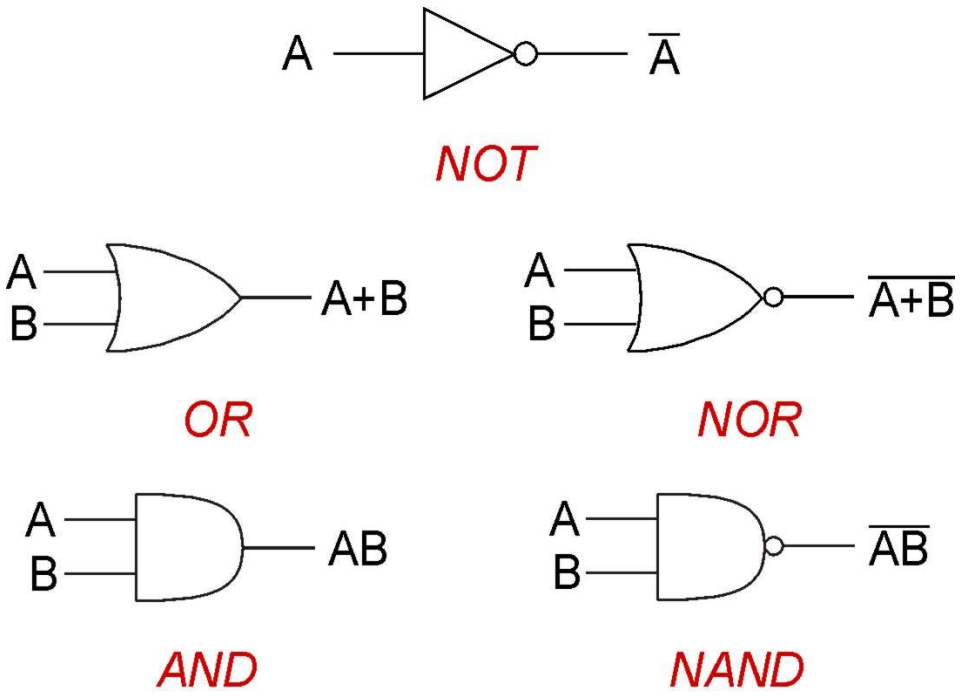


(b)

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

Figure 3.8 **The AND gate.**

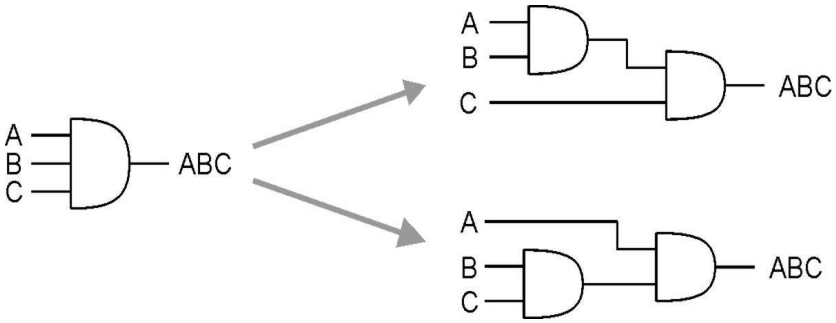
Basic logic gates



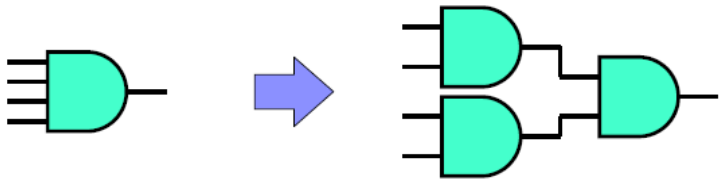
- More complicated gates from transistors possible ex. XOR

More than 2 Inputs?

- AND/OR can take any number of inputs.
  - AND = 1 if all inputs are 1.
  - OR = 1 if any input is 1.
  - Similar for NAND/NOR.
- Can implement with multiple two-input gates, or with single CMOS circuit.

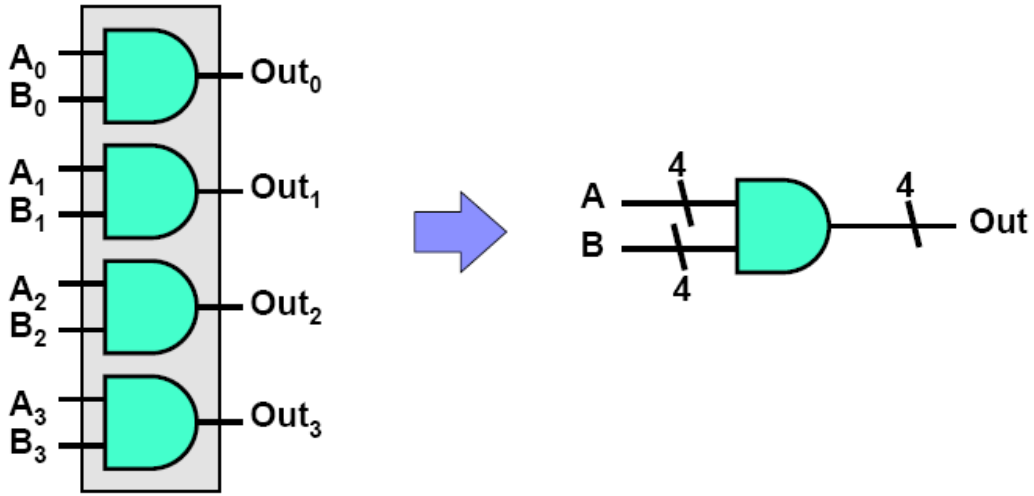


A	B	C	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



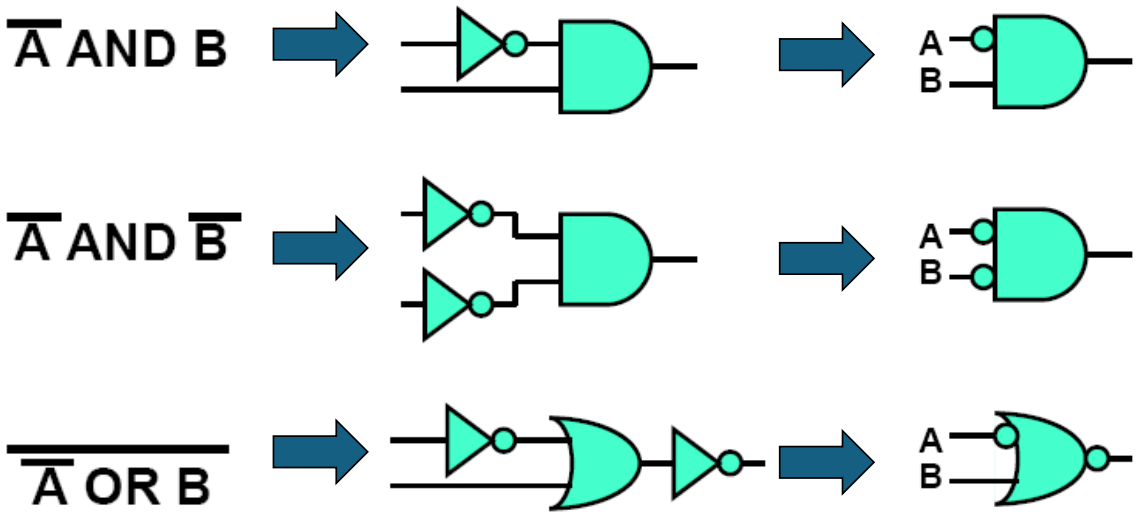
Visual Shorthand for Multi-bit Gates

- Use a cross-hatch mark to group wires
  - Example: calculate the AND of a pair of 4-bit numbers
  - A3 is “high-order” or “most-significant” bit
  - If “A” is 1000, then $A_3 = 1$, $A_2 = 0$, $A_1 = 0$, $A_0 = 0$



Shorthand for Inverting Signals

- Invert a signal by adding either
  - A before/after a gate
  - A "bar" over letter



Combinational Logic Circuits

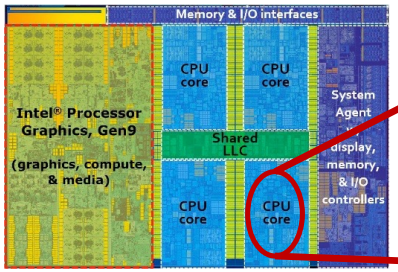
Combinational Logic Circuits

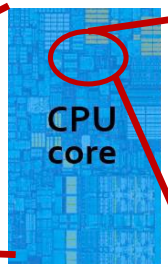
- Outputs are strictly dependent on the combination of input values
- Stateless
- Building Functions from Logic Gates
- **Decoder**
- **Mux: Multiplexer**
- **A One-Bit Adder**

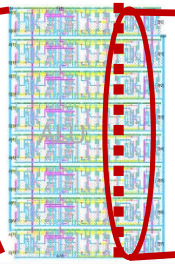
Sequential Logic Circuits

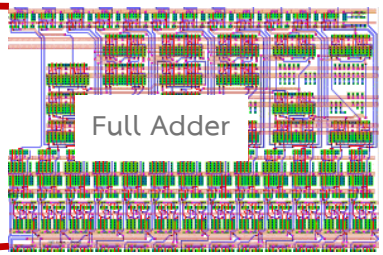
- Output depends on the sequence of inputs (past and present)
- Stores information (state) from past inputs

Approach: Bottom Up









Integrated Circuit Design

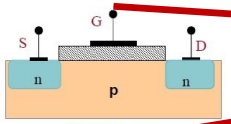
100 Modules/ IC

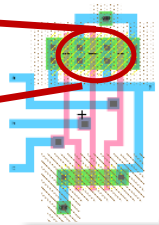
0.25M~20G Devices

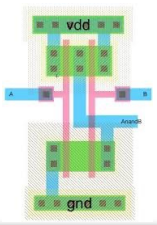
Register Transfer Level (RTL) Design

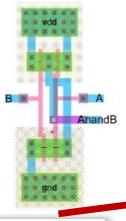
1K~10K Cells/Module

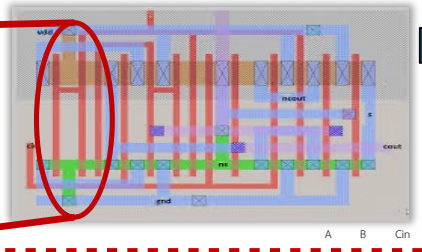
(100K Devices)

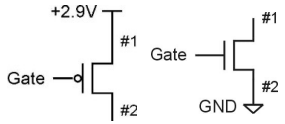








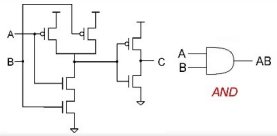




Transistor Physical Layout

Gate Level Design

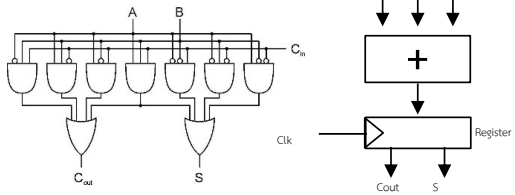
Scheme for Representing Information



Circuit Level Design

(Transistor Level Design)

(2~8 Devices/Gate)



Register Transfer Level (RTL) Design

2~16 Gates/Cell

(16~64 Devices)

Decoder

- n inputs, 2^n outputs
  - Exactly one output is 1 for each possible input pattern

2-bit decoder

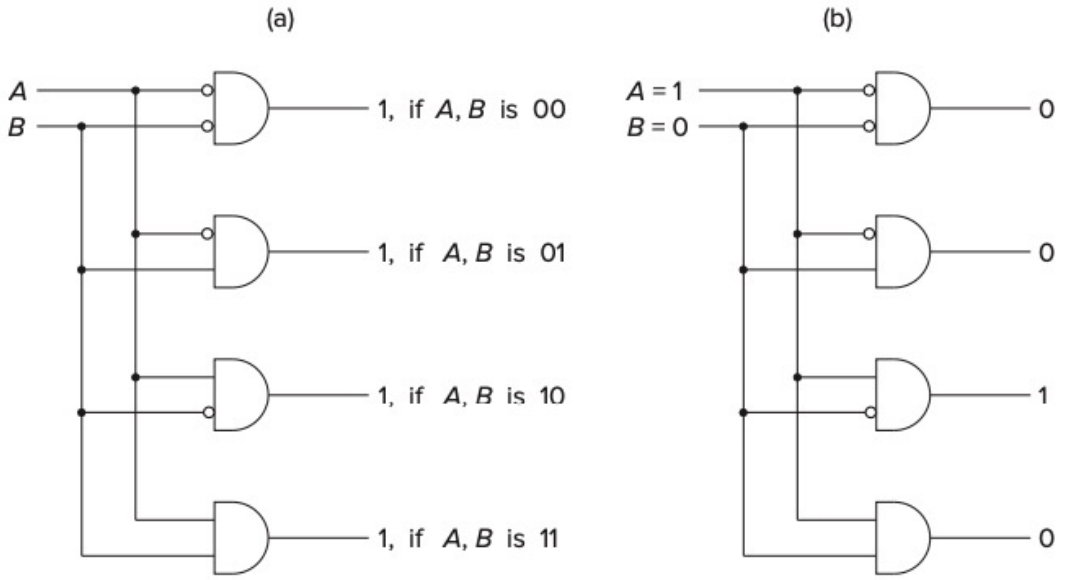


Figure 3.11 A two-input decoder.

Multiplexer (MUX)

- *n*-bit selector and 2^n inputs, one output
  - output equals one of the inputs, depending on selector

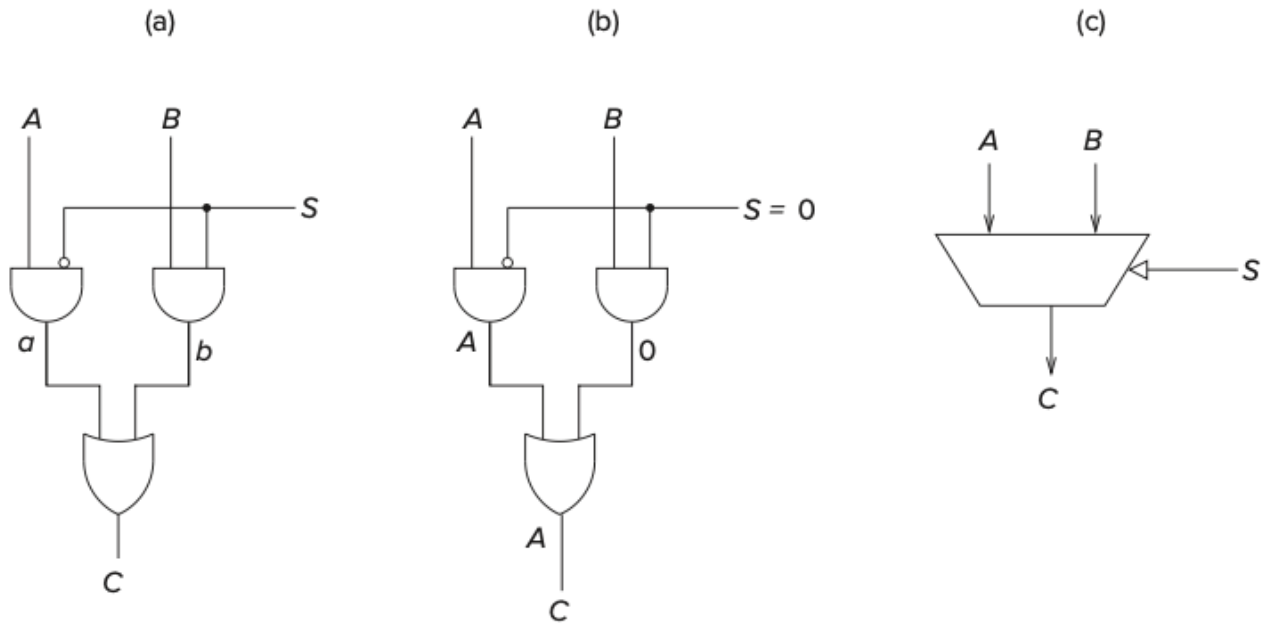


Figure 3.12 **A 2-to-1 mux.**

4-Input Multiplexer

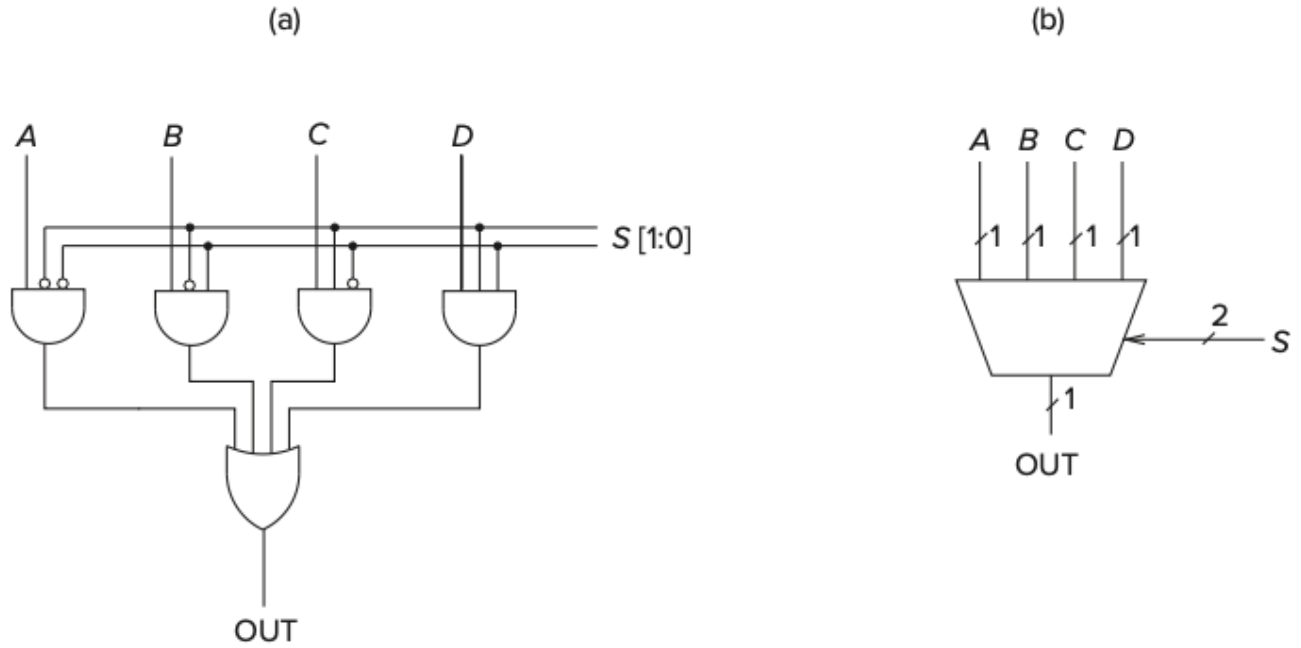
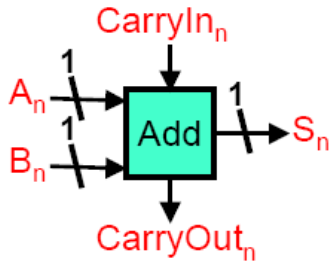
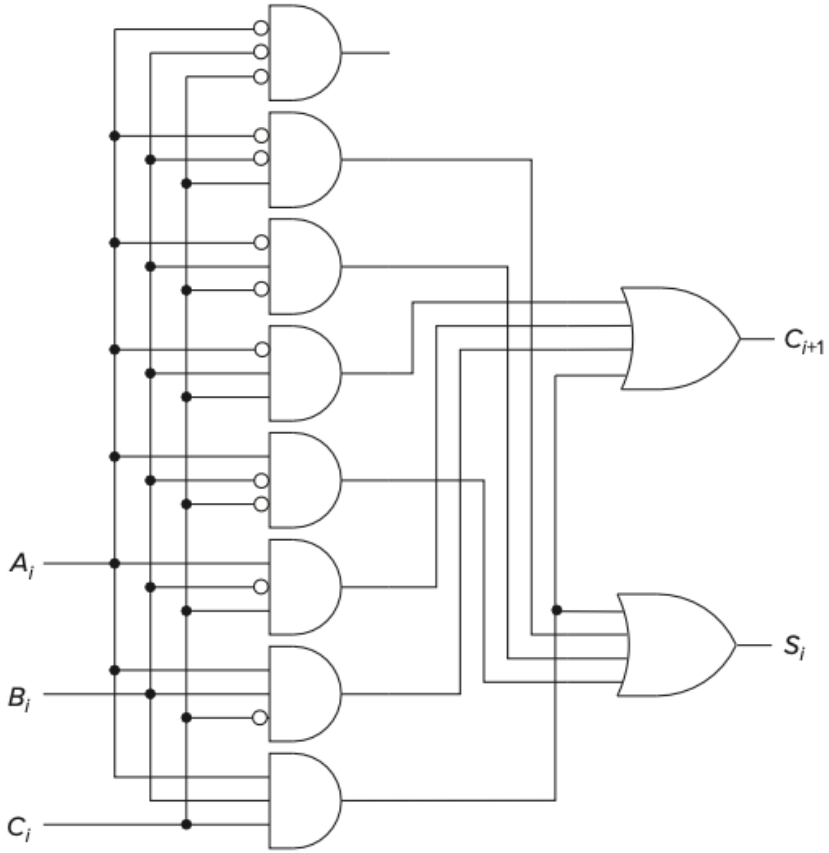


Figure 3.13 **A four-input mux.**

1-Bit Adder (Full Adder)

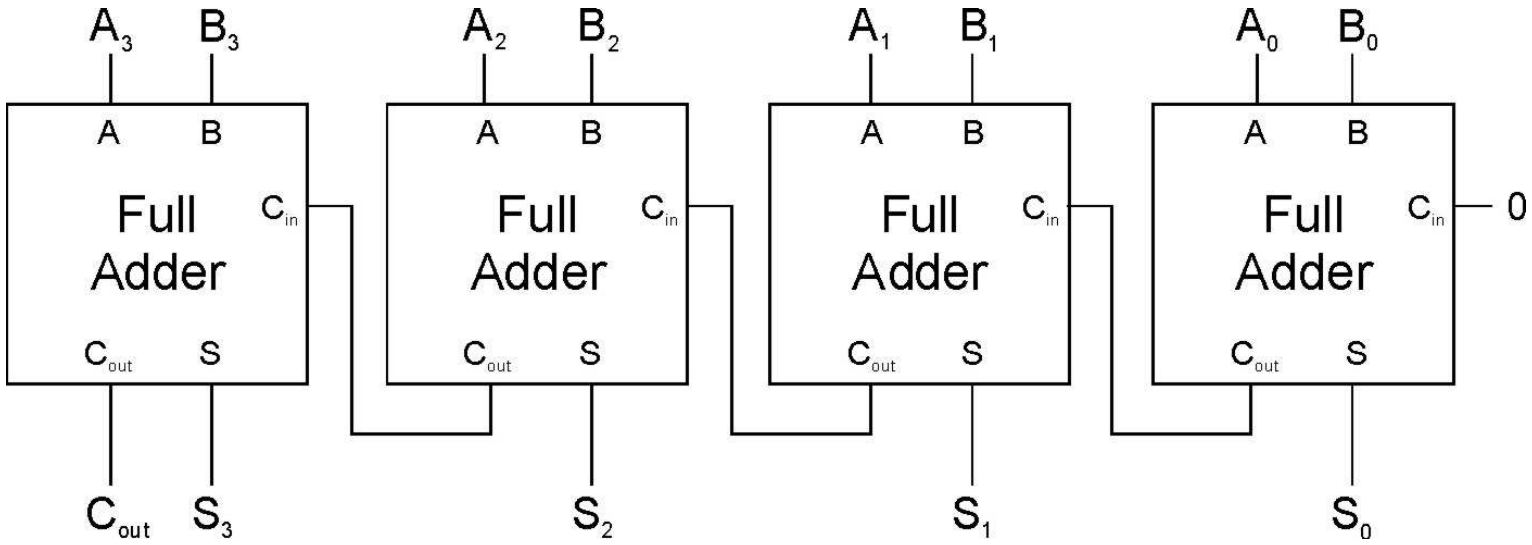
- Add **2-bit and carry-in**, produce one-bit sum and carry-out.



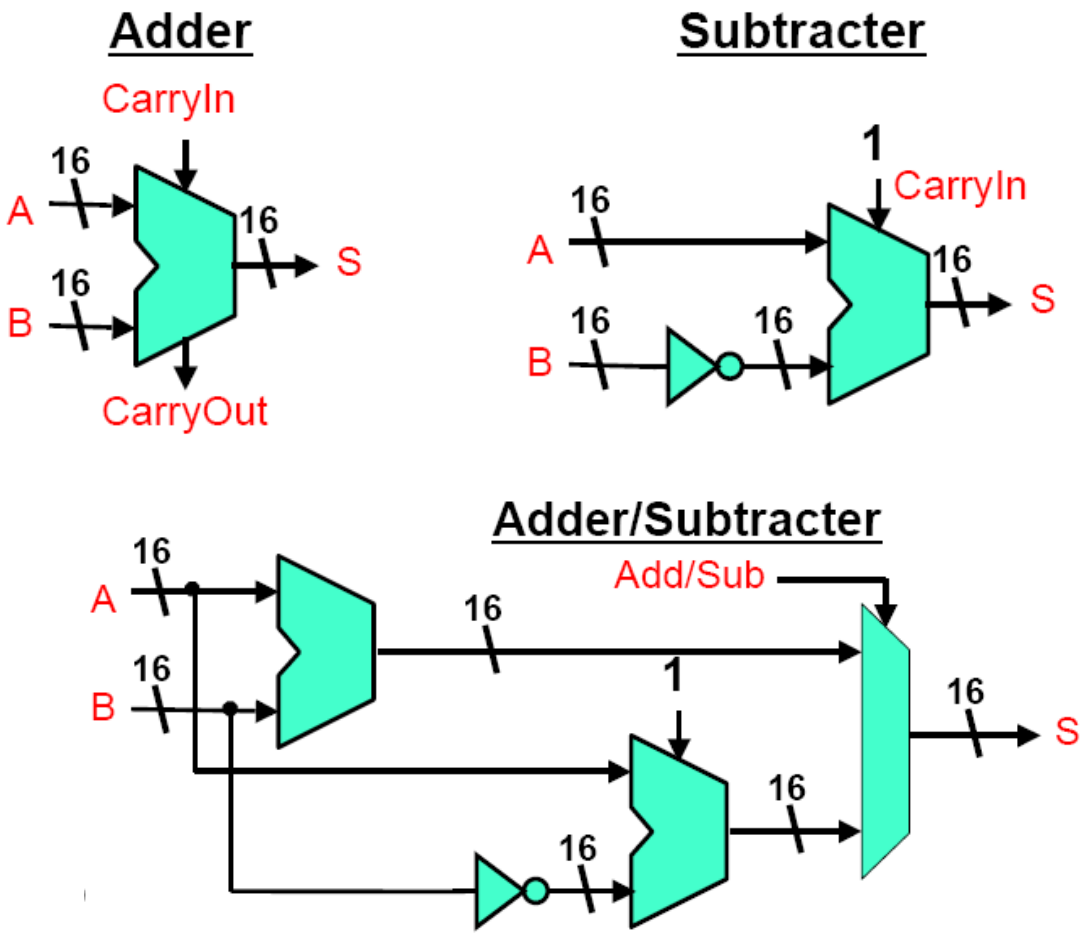


A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

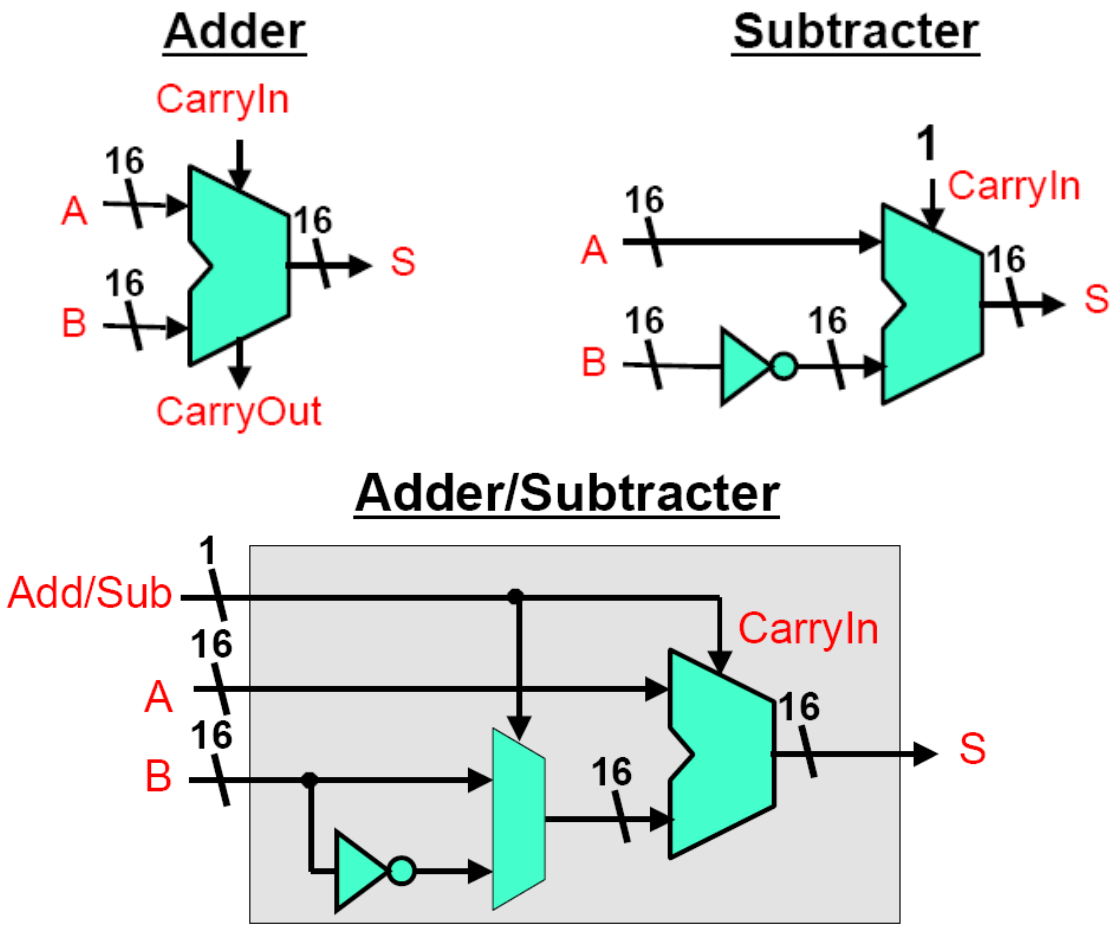
4-bit Adder



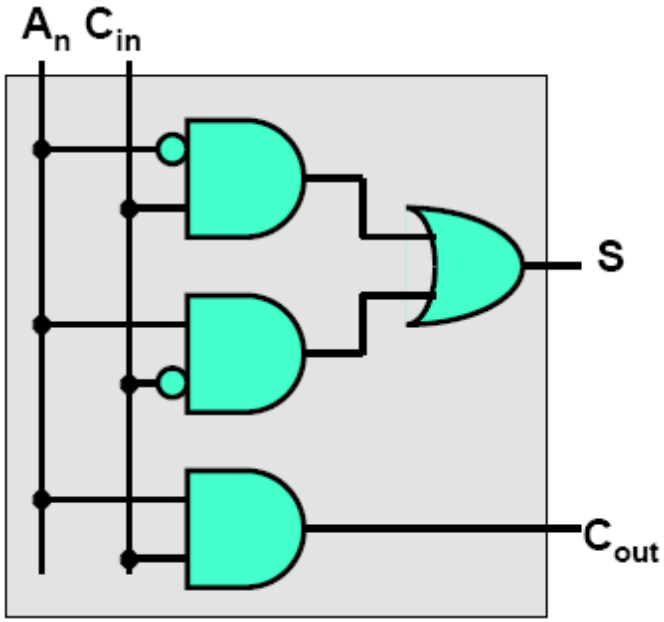
Adder/Subtracter - Approach #1



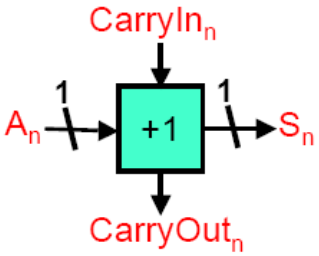
Adder/Subtracter - Approach #2



One-bit Incrementor



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



Basic Storage Elements

Combinational Logic Circuits

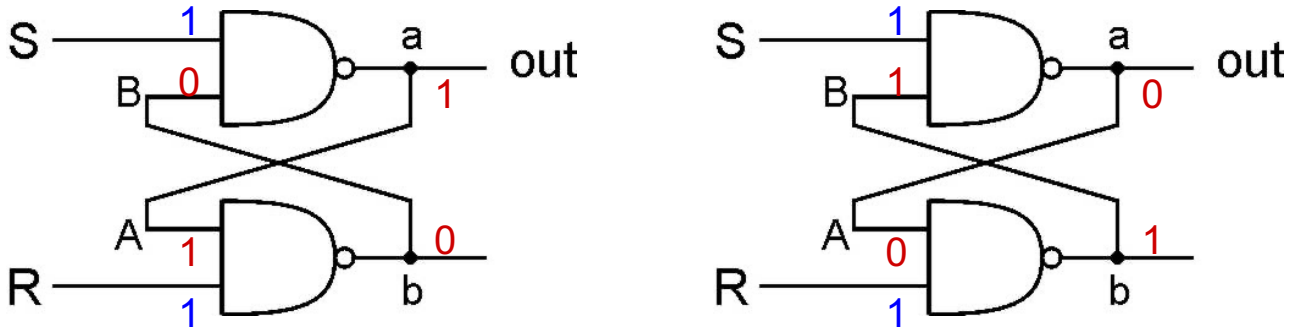
- Always gives the same output for a given set of inputs
  - **Ex: adder always generates sum and carry, regardless of previous inputs**

Sequential Logic Circuits

- Output **depends on the sequence of inputs** (past and present)
- Stores information (state) from past inputs
  - A given input might produce different outputs, depending on the stored information
  - Ex: ATM
- Useful for building “memory” elements and “state machines”

The R-S Latch: A simple example of a storage element

- Store 1-bit of information (0 or 1)
- R is used to "reset" or "clear" the element – set it to 0
- S is used to "set" the element – set it to 1

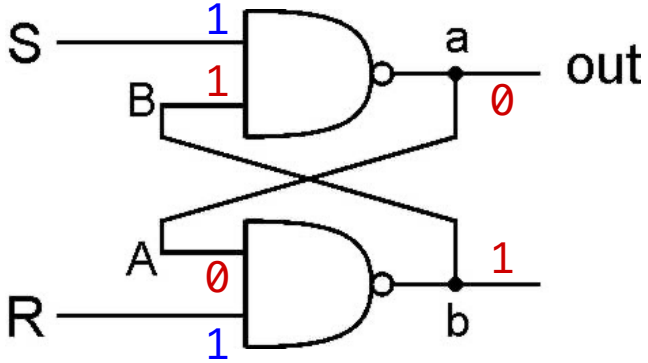


** If both R and S are one, out could be either zero or one

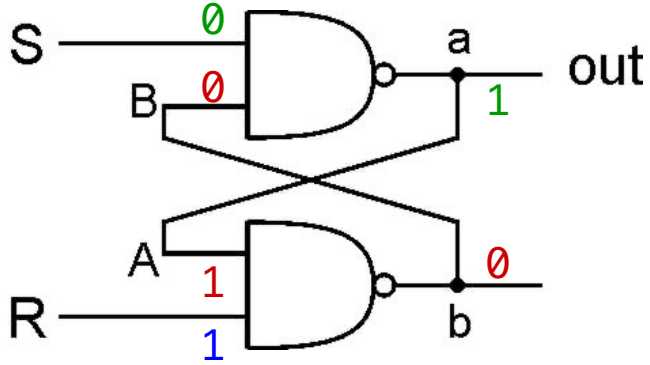
- The Quiescent State holds its previous value

Setting the R-S Latch

- Suppose we start with output = 0, then change S to 0



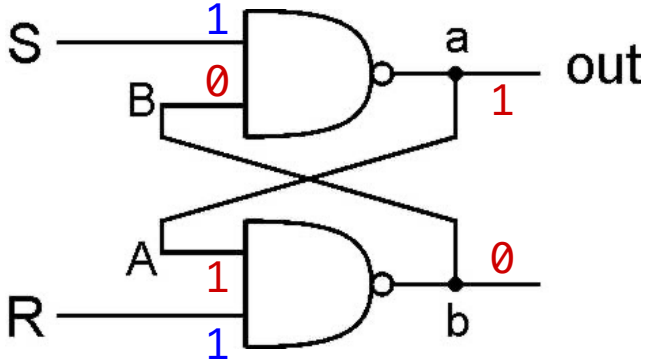
Output changes to one.



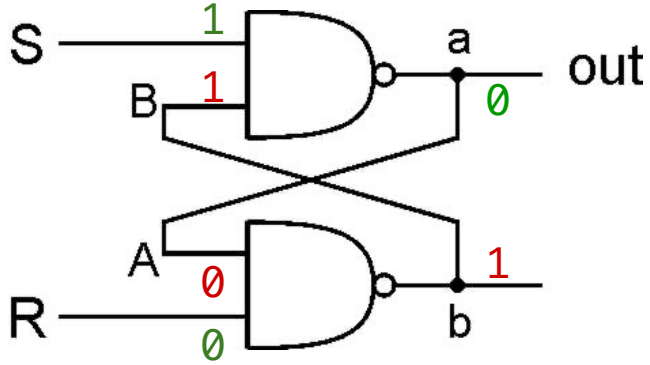
*** Then set S=1 to "store" value in quiescent state.*

Clearing the R-S Latch

- Suppose we start with output = 1, then change R to 0



Output changes to one.



*** Then set R=1 to "store" value in quiescent state.*

R-S Latch Summary

R = S = 1

- hold current value in latch

S=1 and R = change from 1 → 0

- set value to 0

R=1 and S = change from 1 → 0

- set value to 1

R = S = 0

both outputs equal one

final state determined by electrical properties of gates

Gated D-Latch

- Two inputs: D (data) and WE (write enable)
  - when **WE = 1**, latch is set to value of D ; **S = NOT(D), R = D**
  - when **WE = 0**, latch holds previous value ; **S = R = 1**

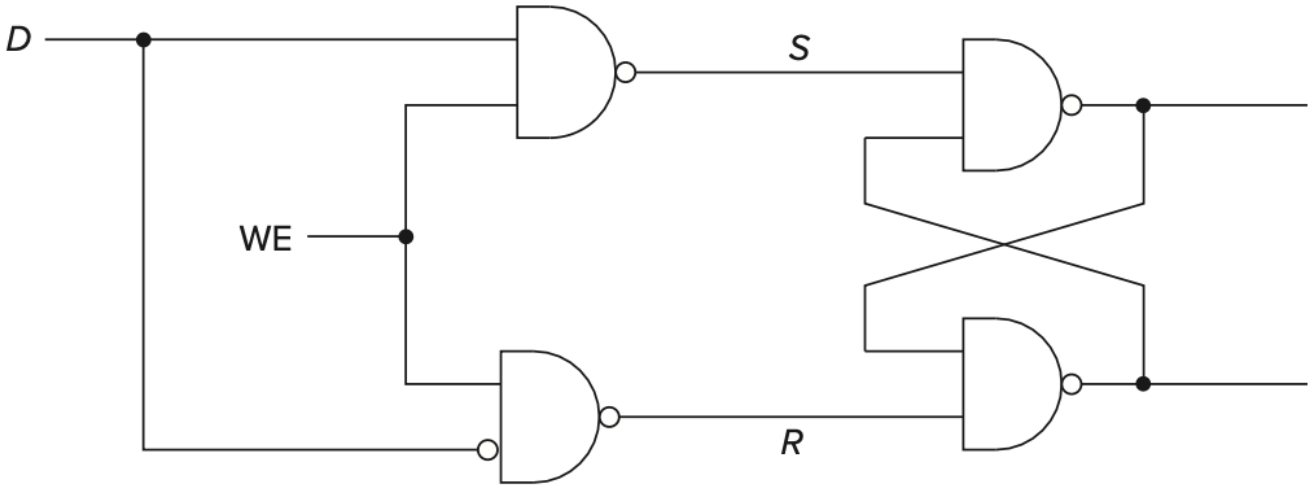
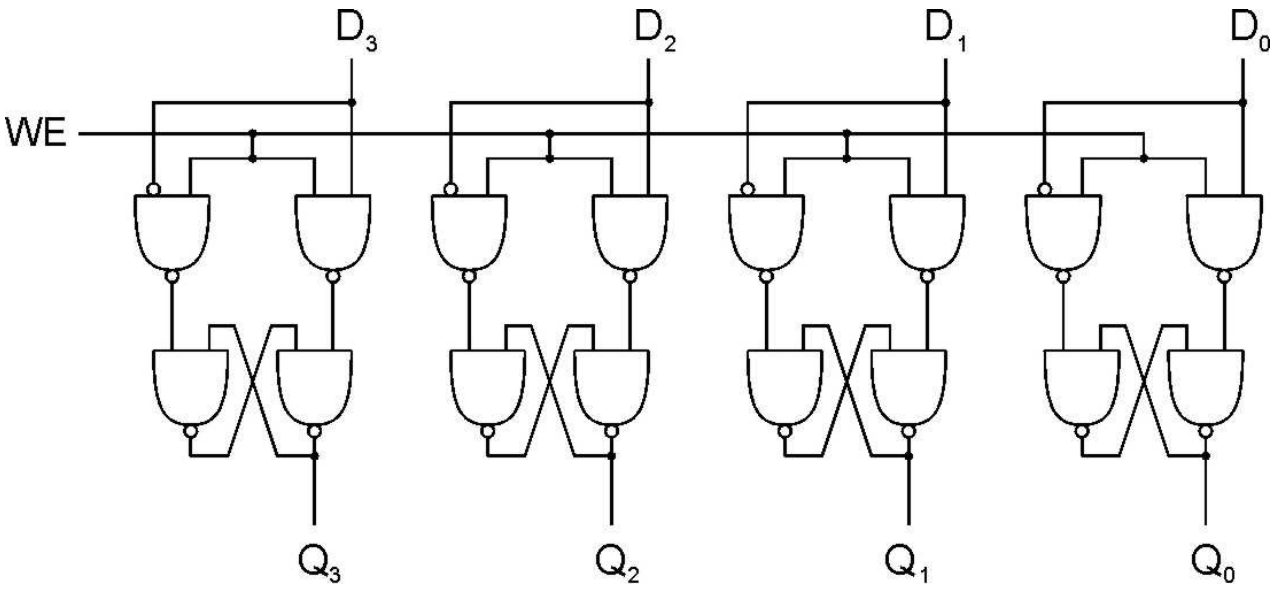
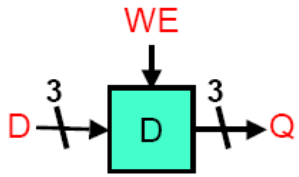


Figure 3.19 A gated D latch.

Register

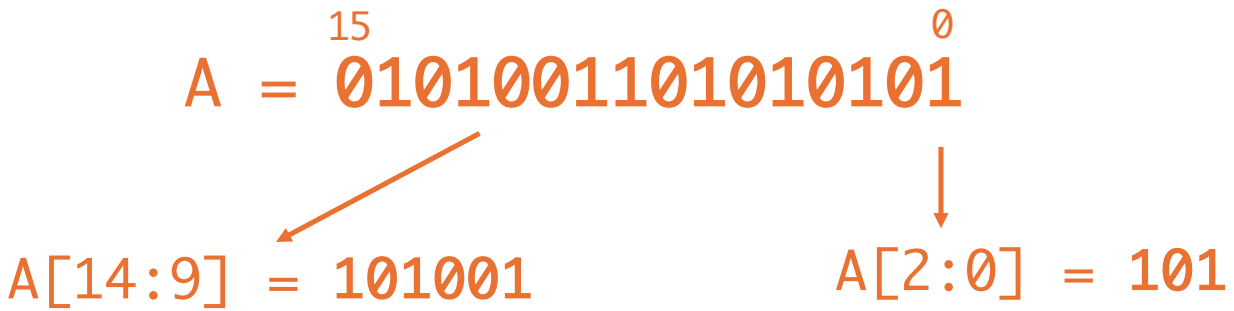
- Store a multi-bit value.
  - Collection of D-latches, all controlled by a common WE.
  - When WE=1, ***n***-bit value D is written to register





Representing Multi-bit Values

- Number bits from right (0) to left (n-1)
- Use brackets to denote range: **D[l : r]** denotes bit **l** to bit **r**, from *left* to *right*



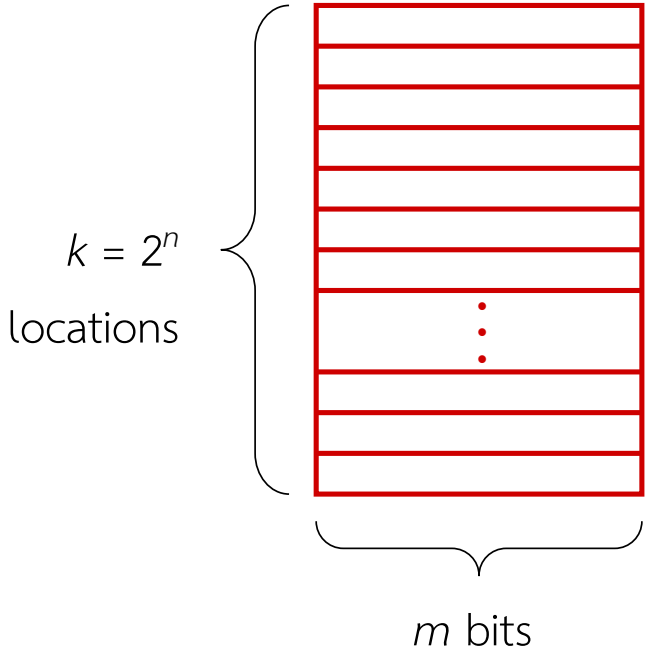
Memory

- A memory can be built – a logical ***k x m* array of stored bits.**

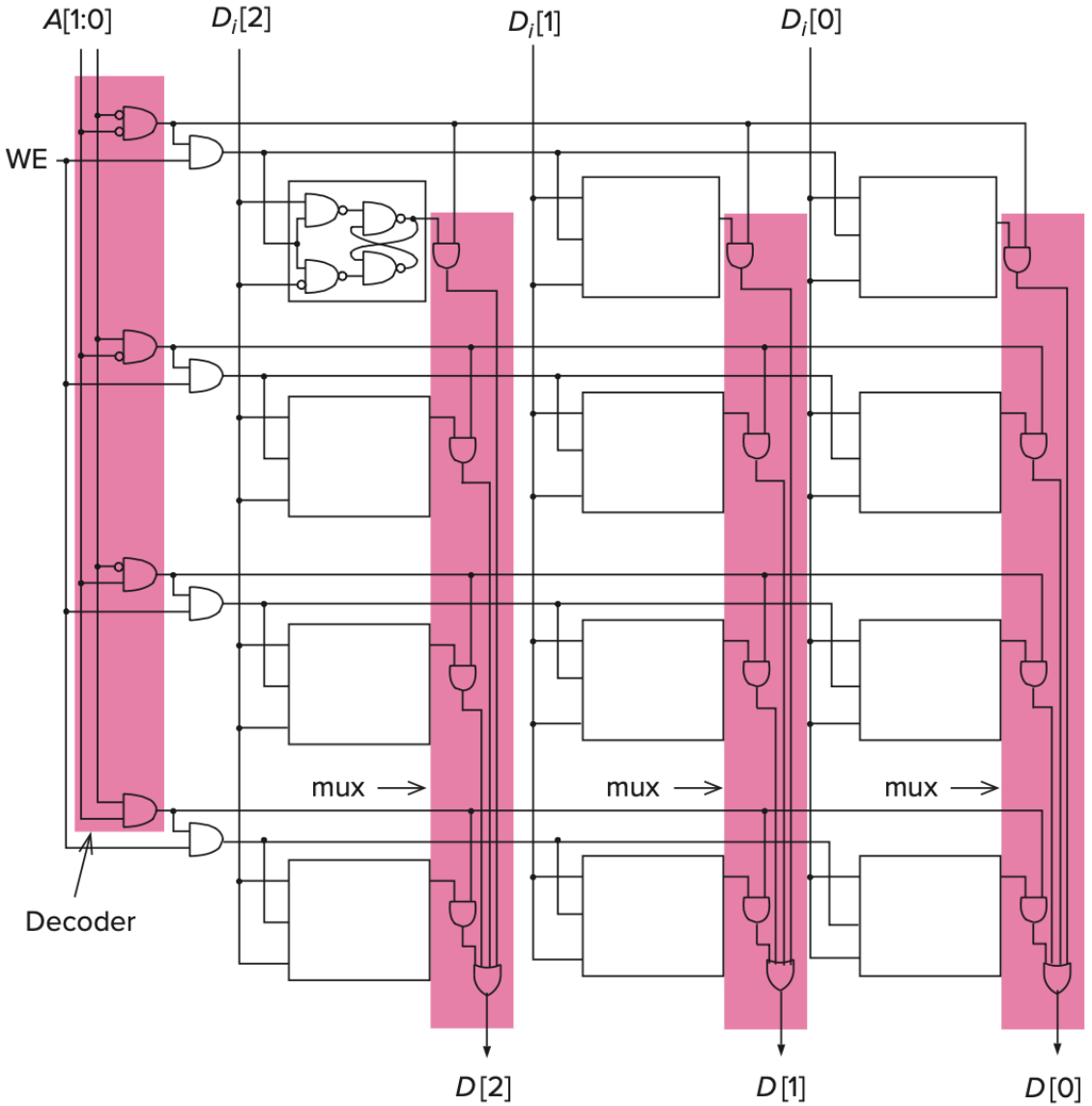
Address Space:

number of locations

(usually a power of 2)

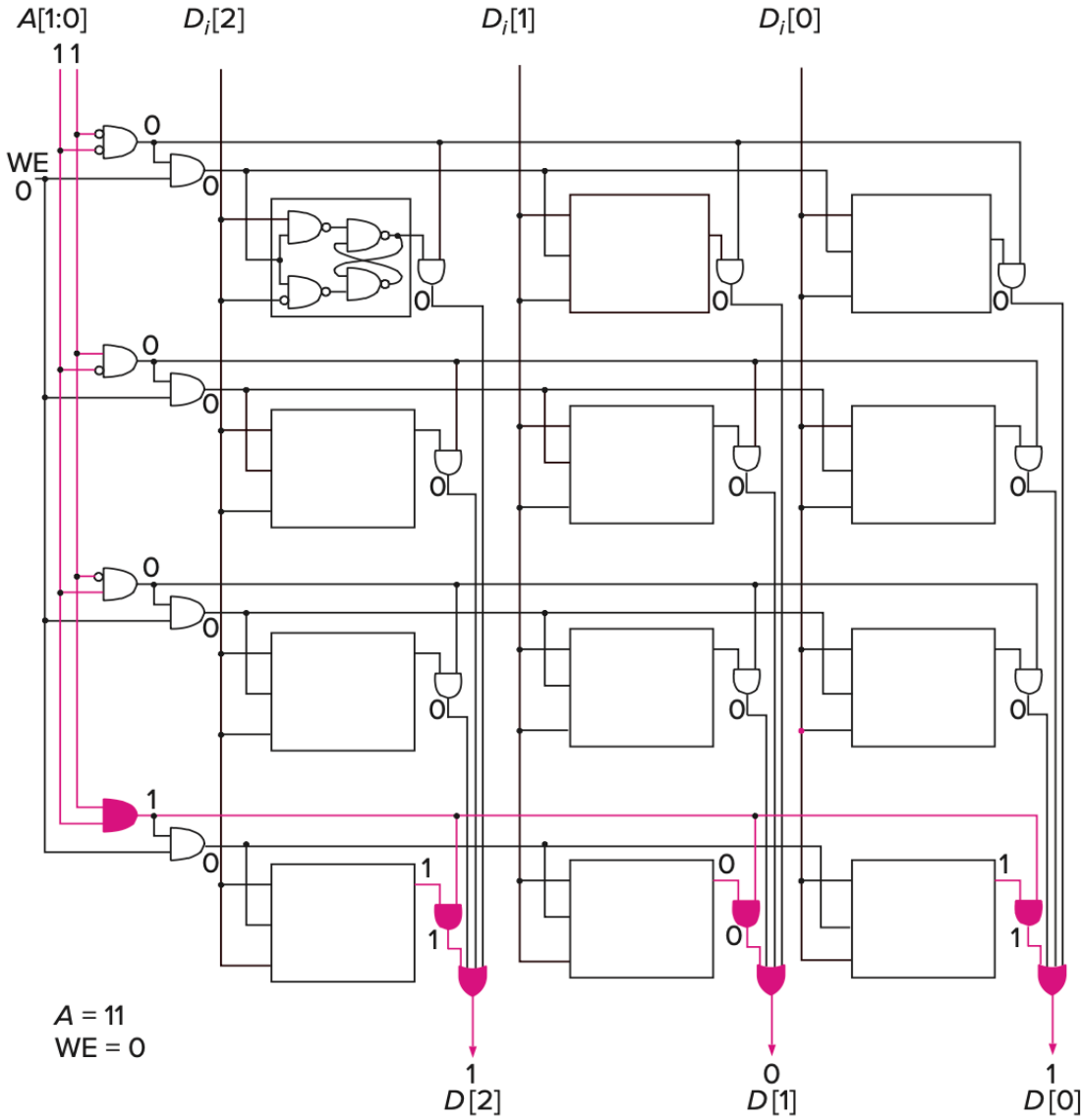


2^2-by-3-bit memory

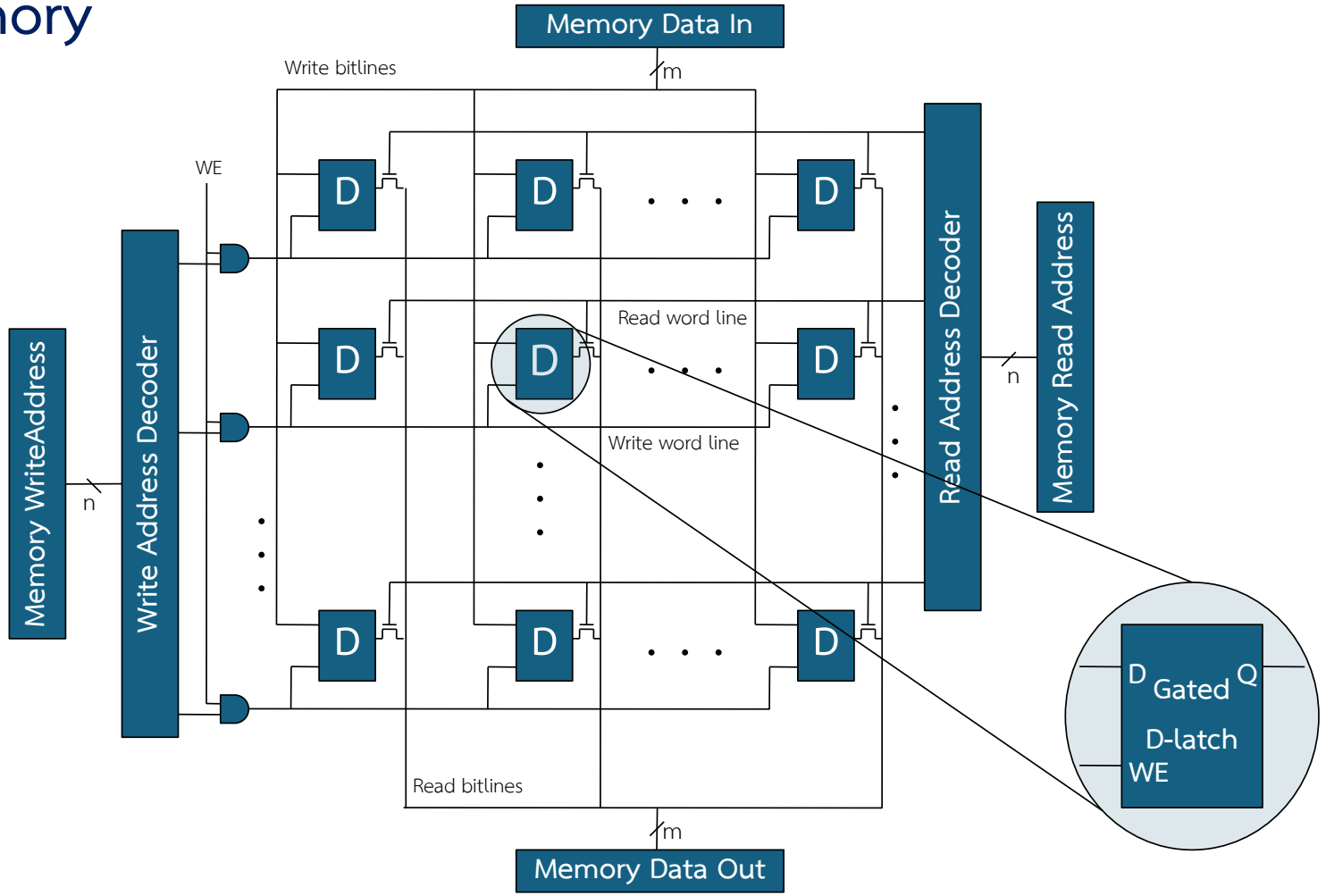


2^2-by-3-bit memory

Reading location 3rd



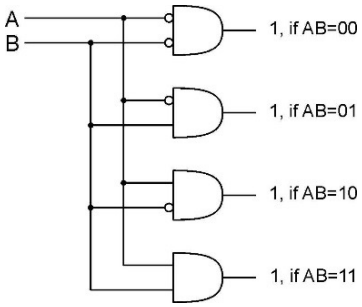
SRAM Memory

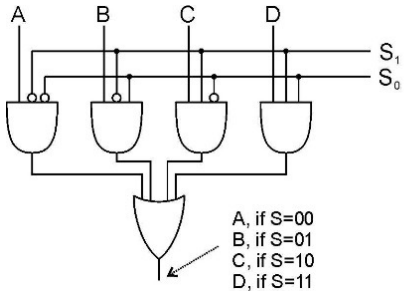

Memory Data In
m
Write bitlines
WE
D
Read word line
Write word line
Memory WriteAddress
n
Write Address Decoder
Read Address Decoder
n
Memory Read Address
D Gated Q
D-latch
WE
Read bitlines
m
Memory Data Out

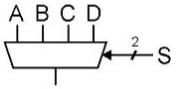
Wrap-up Memory

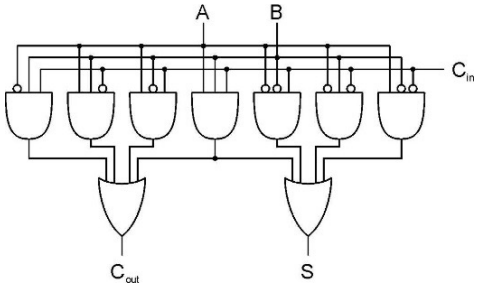
- This is a **not the way actual** memory is implemented.
  - fewer transistors, much more dense, relies on electrical properties
- **But the logical structure is very similar.**
  - **address decoder**
  - **word select line**
  - **word write enable**
- Two basic kinds of RAM (Random Access Memory)
  - Static RAM (SRAM): fast, not very dense **(bitcell 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

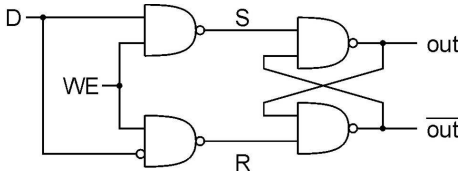
Wrap-up Memory and Basic Logical Structure

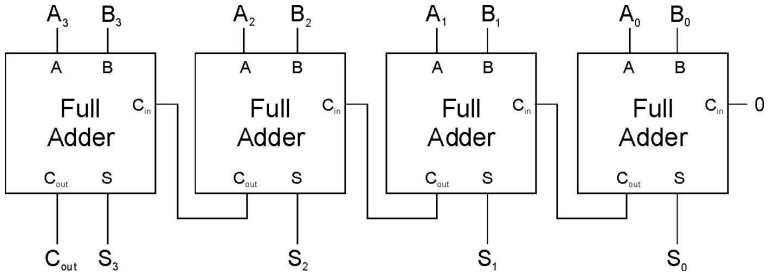


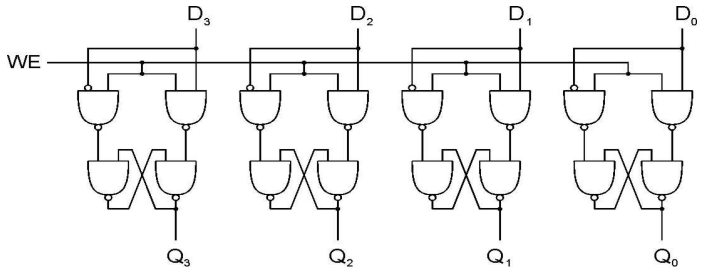


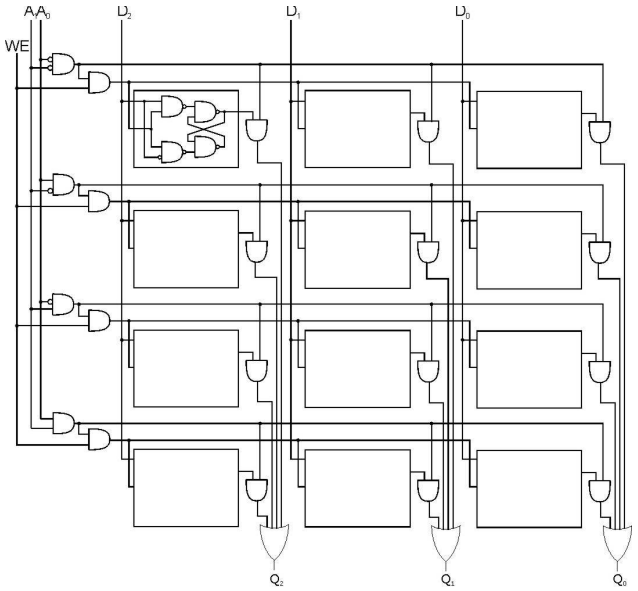










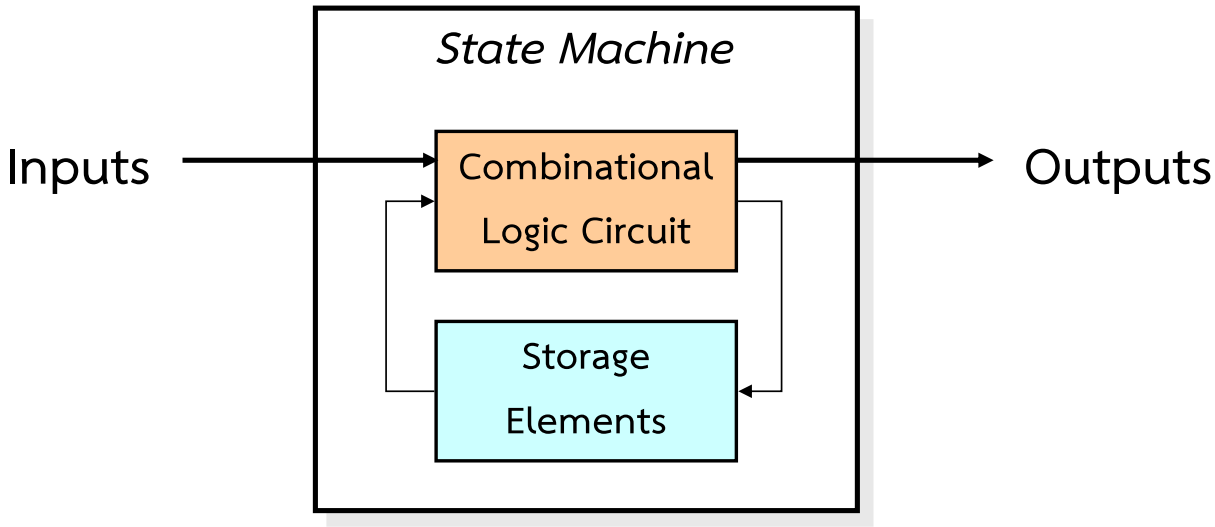


Sequential Logic Circuits

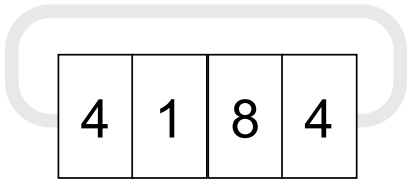
State Machine

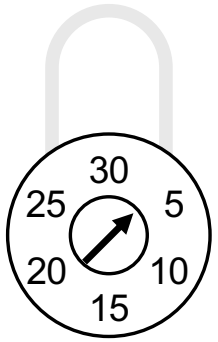
Another type of sequential circuit

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



Combinational vs. Sequential





Combinational

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

Sequential

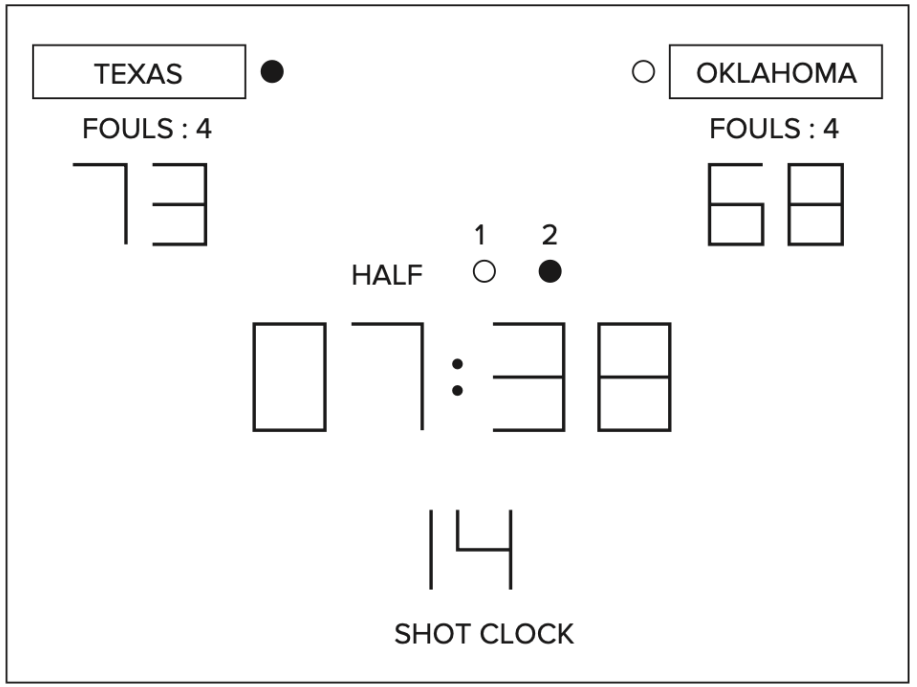
Success depends on the sequence of values

(EX: R13-L22-R3)

The Concept of State

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

- Ex: State of a game of basketball



State Diagram

** R13-L22-R3 to Unlock

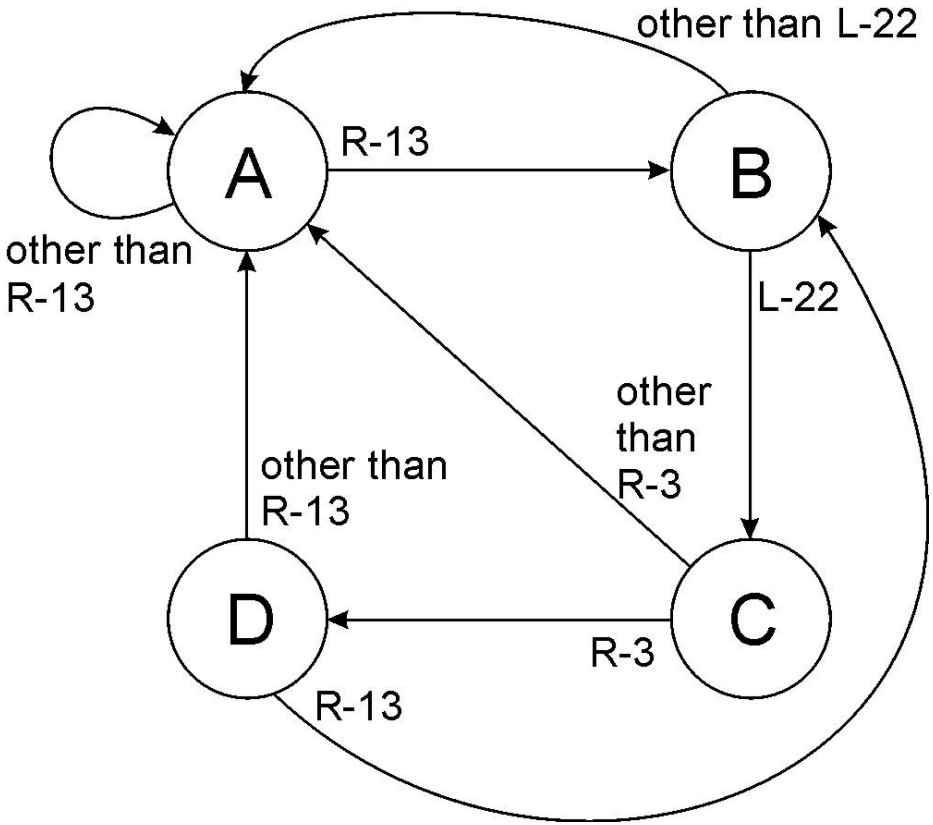
Back to Sequential Lock

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.



Finite State Machine

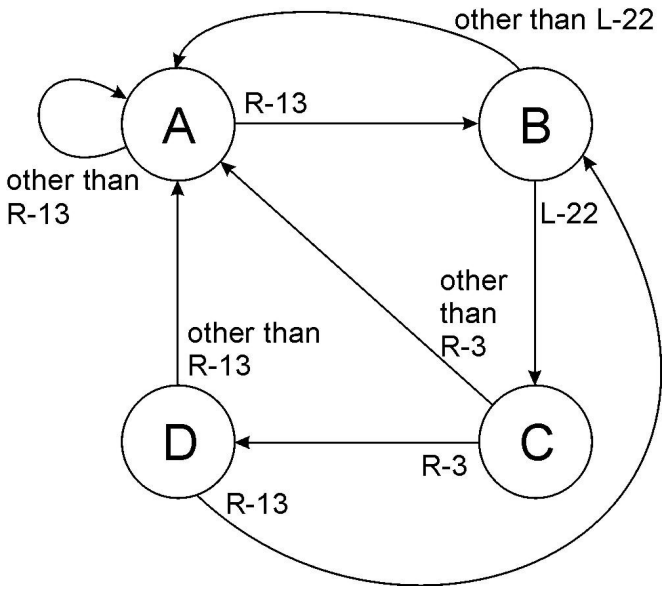
1. a finite number of states
2. a finite number of external inputs
3. a finite number of external outputs
4. an explicit specification of all state transitions
5. an explicit specification of what determines each external output value.

Often described by a state diagram

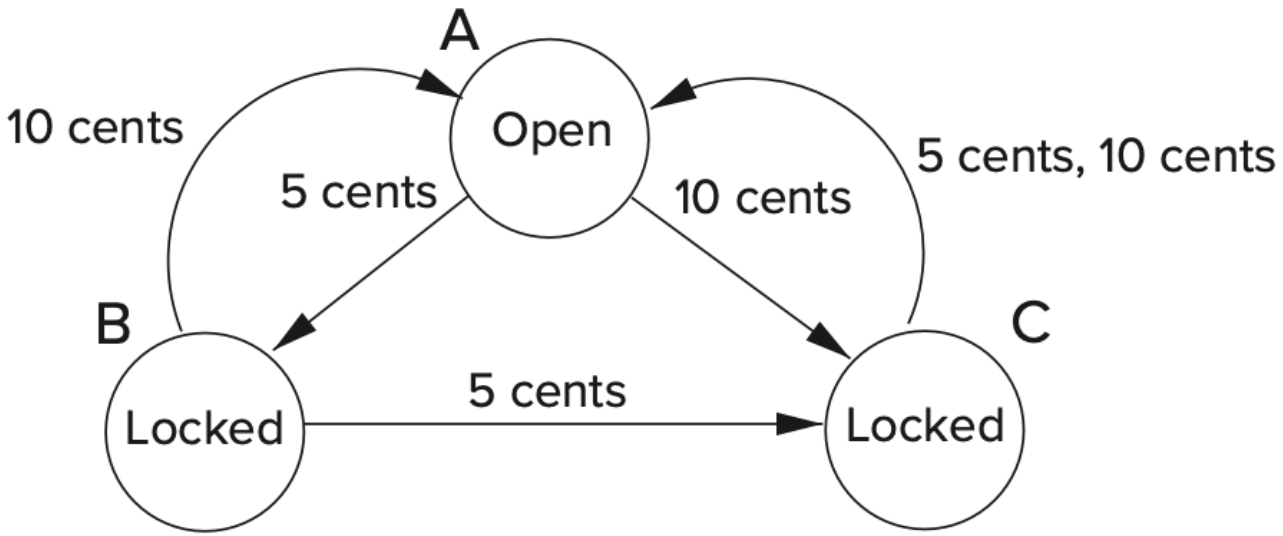
- Inputs may cause state transitions
- Outputs are associated with each state (or with each transition)

Recall Combination Lock

- Four states A, B, C, and D
- The external inputs are R13, L22, R3, and R-other-than-13, L-other-than-22, and R-other-than-3
- The external output is either the lock is open, or the lock is not open
- The explicit specifications of all state transitions are shown by the arrows in the state diagram



Tao Bin


A
Open
10 cents
5 cents
10 cents
5 cents, 10 cents
B
Locked
5 cents
C
Locked

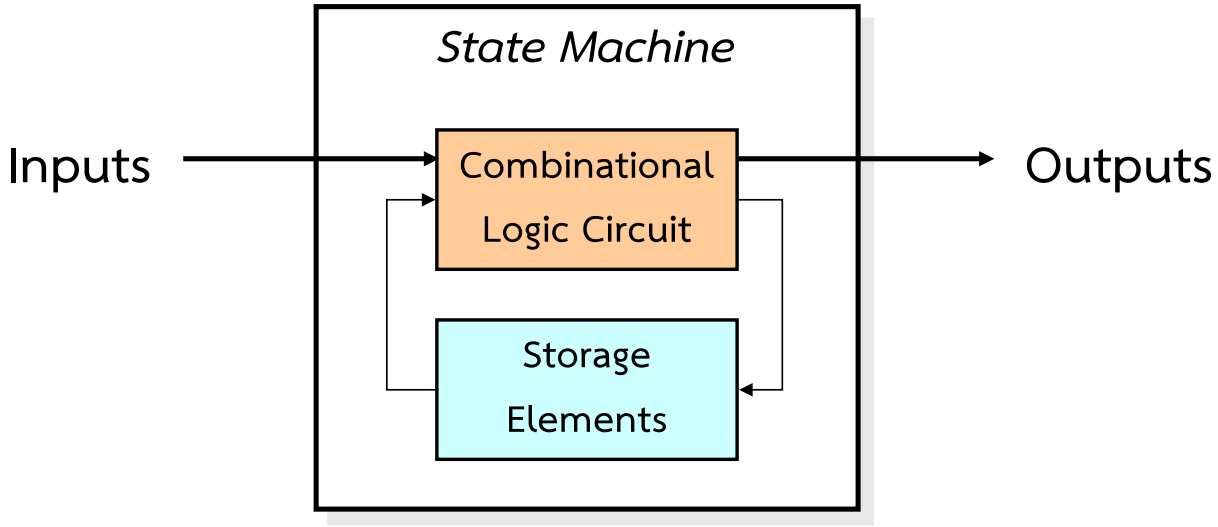
Implementing a Finite State Machine

Combinational logic

- Determine outputs and next state.

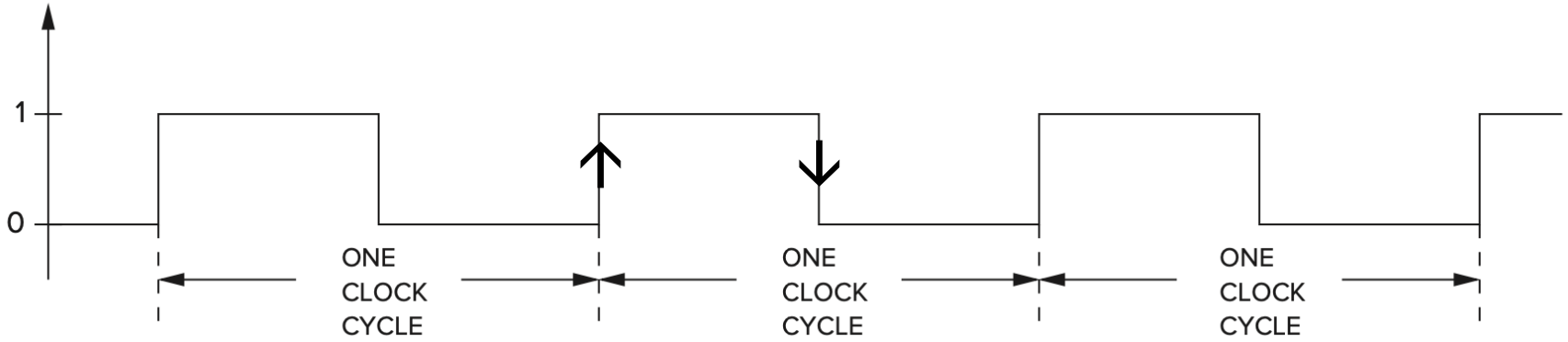
Storage elements

- Maintain state representation.

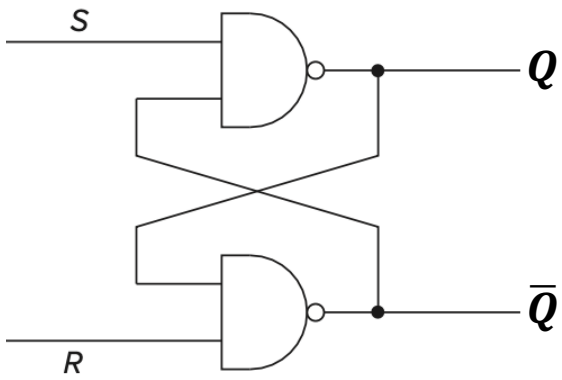


The Clock

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

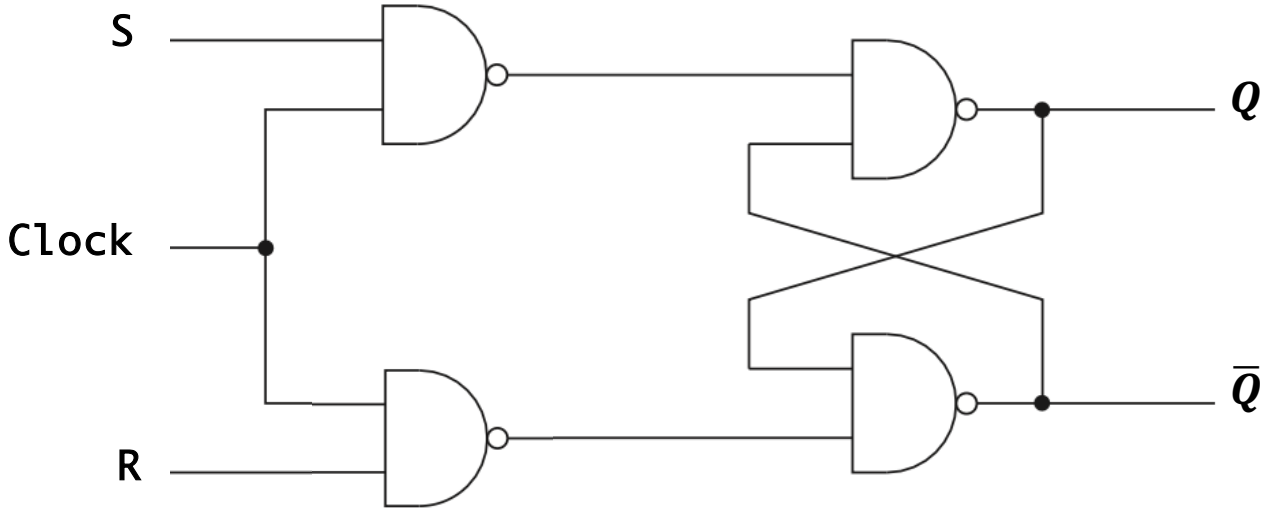


Recall R-S Latch



S	R	Q(t+1)
0	0	-
0	1	1
1	0	0
1	1	Q_t

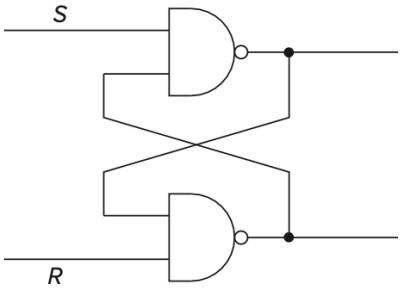
Clocked RS Flip-Flop

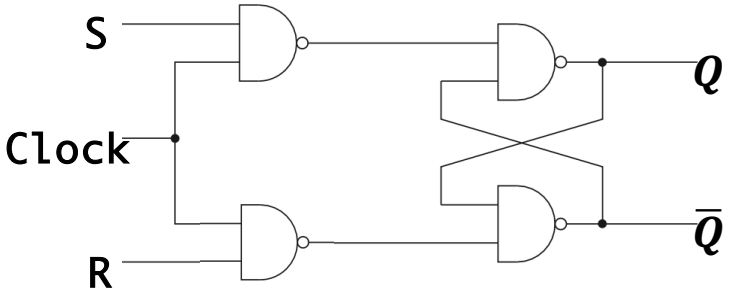


Clock	R	S	Q(t+1)
↓ or 0 or 1	X	X	Q_t (no change)
↑	0	0	Q_t (no change)
↑	0	1	1 (set)
↑	1	0	0 (reset)
↑	1	1	Not Allowed

Note

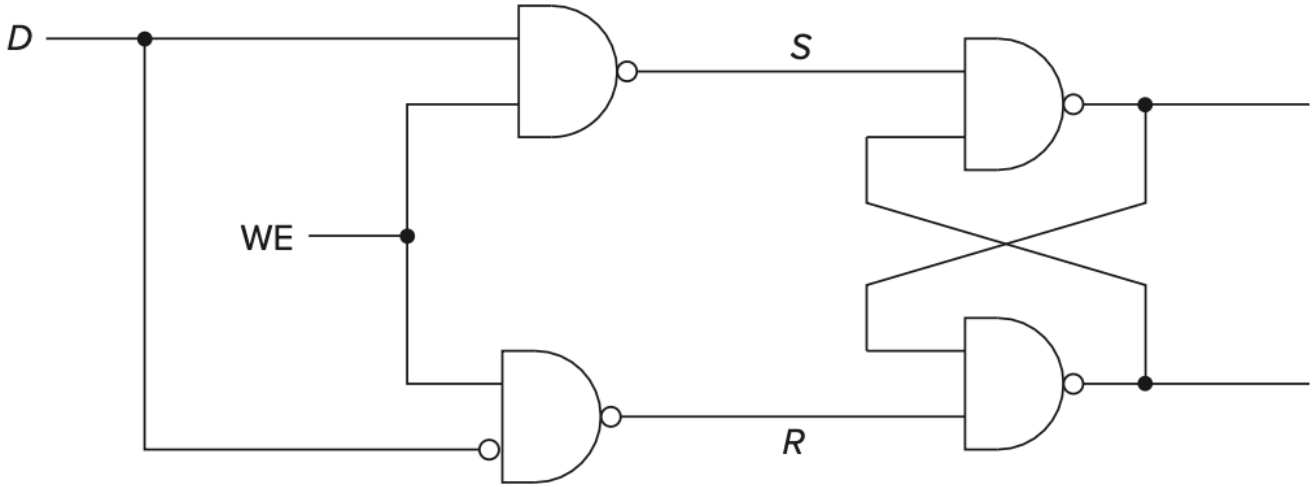
- The difference between **latch and flip flop**
  - **Latch is level-triggered** → outputs can change as soon as the inputs change)
  - **Flip-Flop is edge-triggered** → only changes state when a control signal goes from high to low or low to high



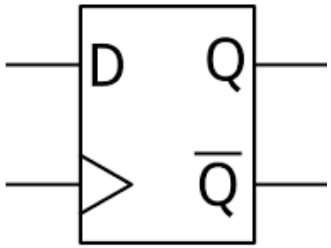


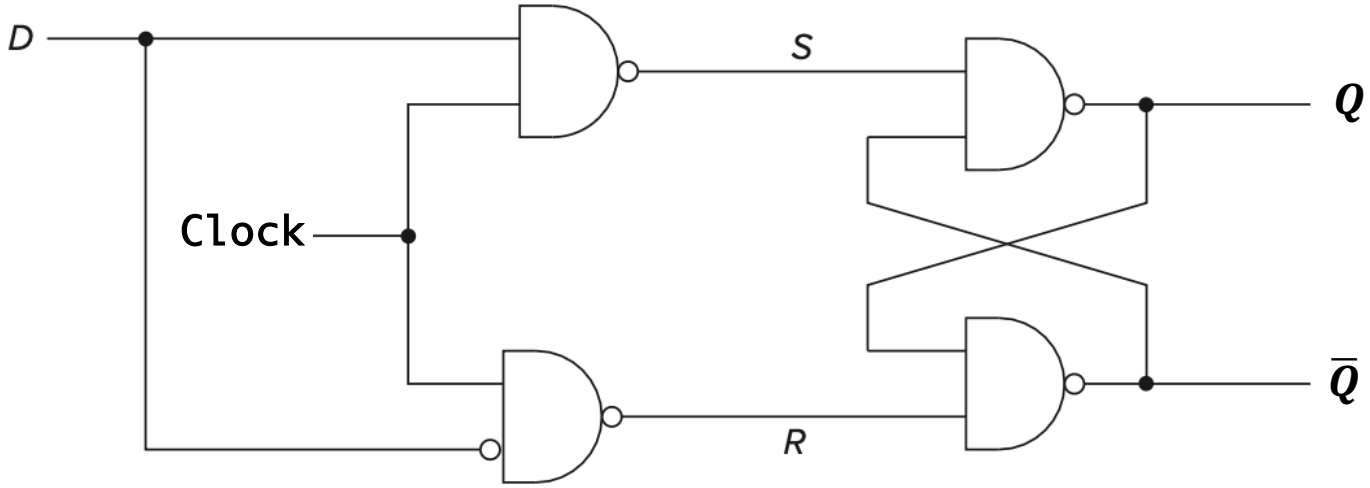
D-Latch

- Two inputs: **D (Data)** and **WE (Write Enable)**
  - when **WE = 1**, latch is set to value of D ; **S = NOT(D), R = D**
  - when **WE = 0**, latch holds previous value ; **S = R = 1**

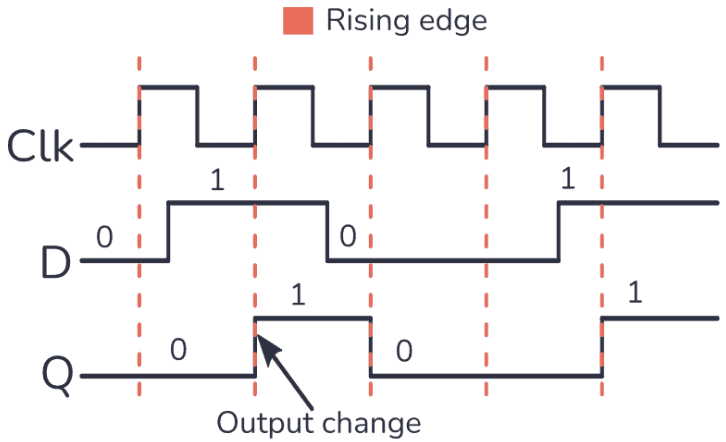


D Flip-Flop





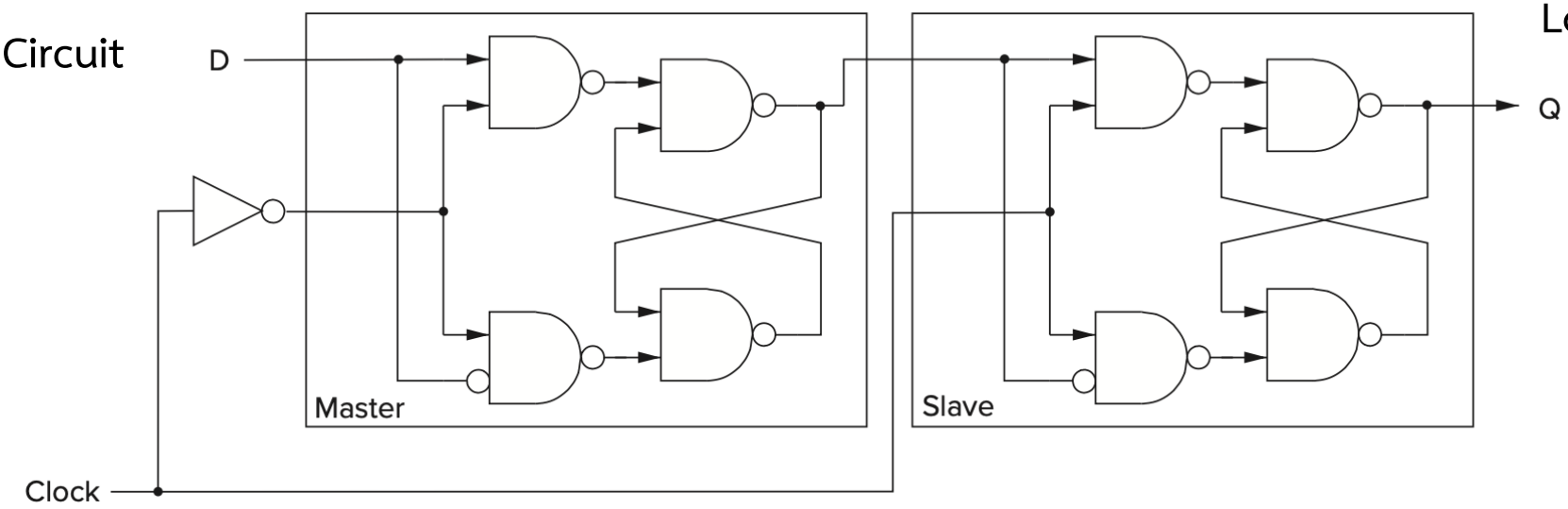
Clock	D	Q(t+1)
↓ or 0 or 1	X	Q_t (no change)
↑	0	0 (reset)
↑	1	1 (set)



Master / Slave flip-flop

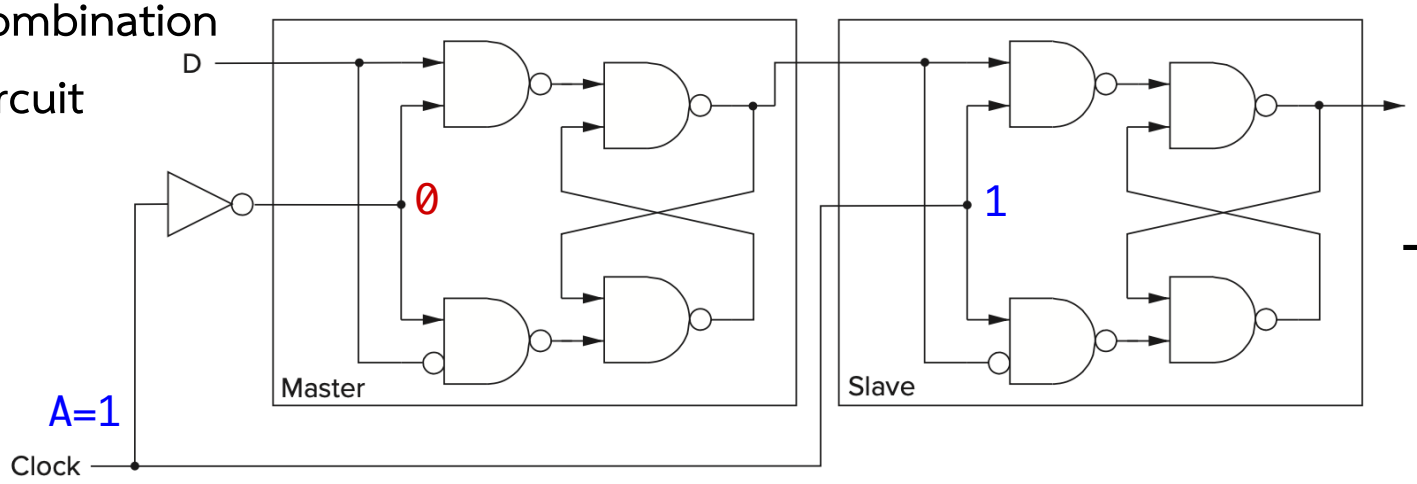
From Combination Logic Circuit

To Combination Logic Circuit



- A pair of gated D-latches, to isolate *next* state from *current* state
- Not write the next state values into the storage elements until the beginning of the next clock cycle

From Combination Logic Circuit



To Combination Logic Circuit

Clock	D	Q(t+1)
↓ or 0 or 1	X	Q_t (no change)
↑	0	0
↑	1	1

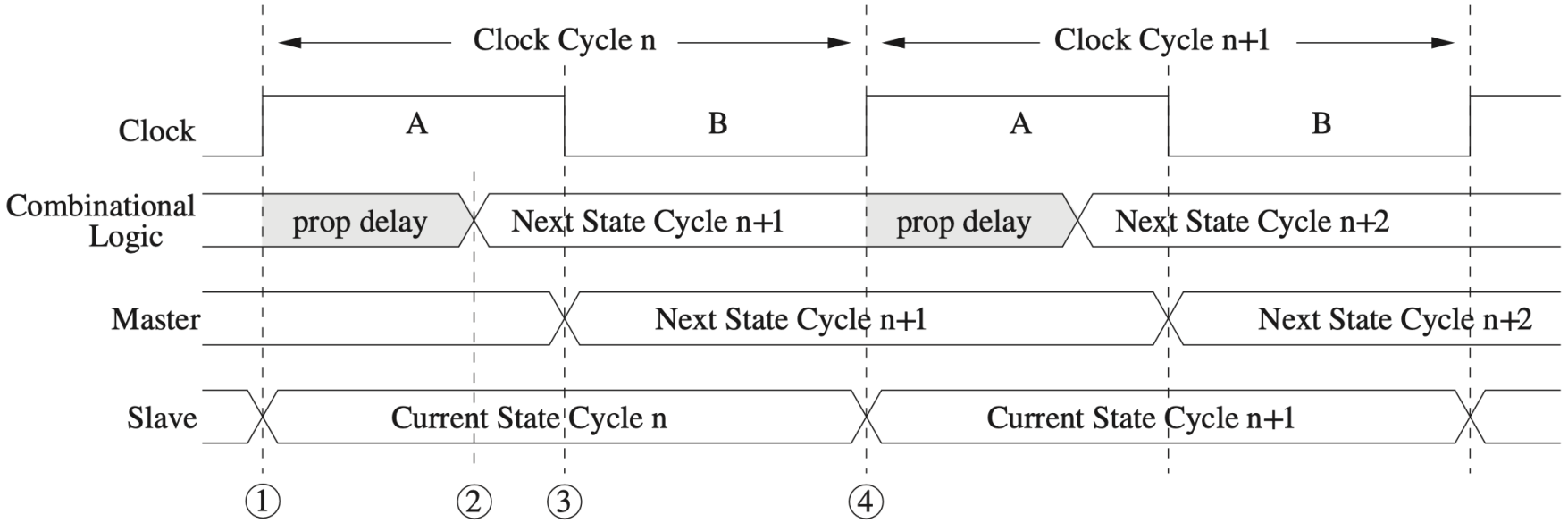
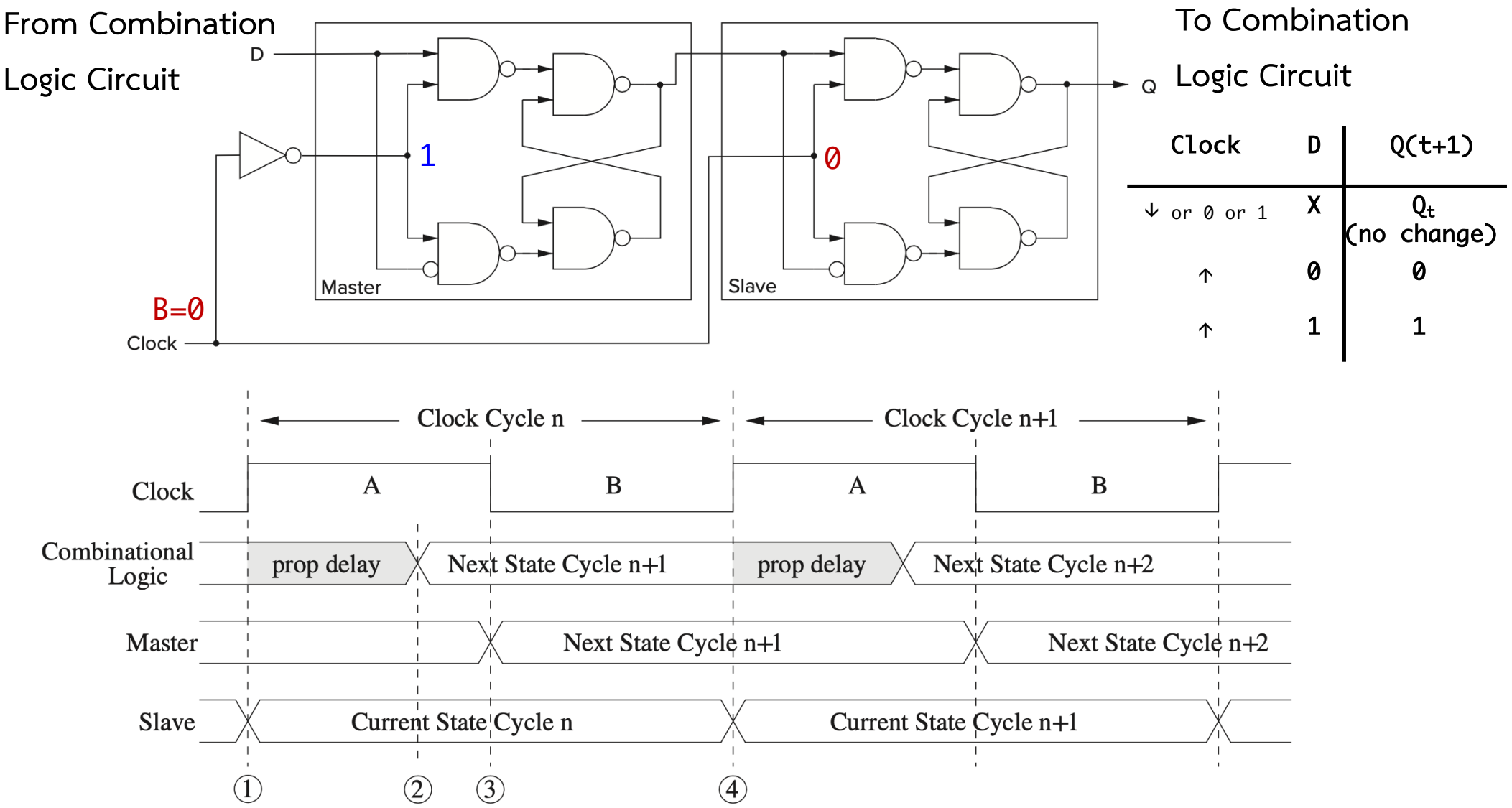


Figure 3.34 Timing diagram for a master/slave flip-flop.

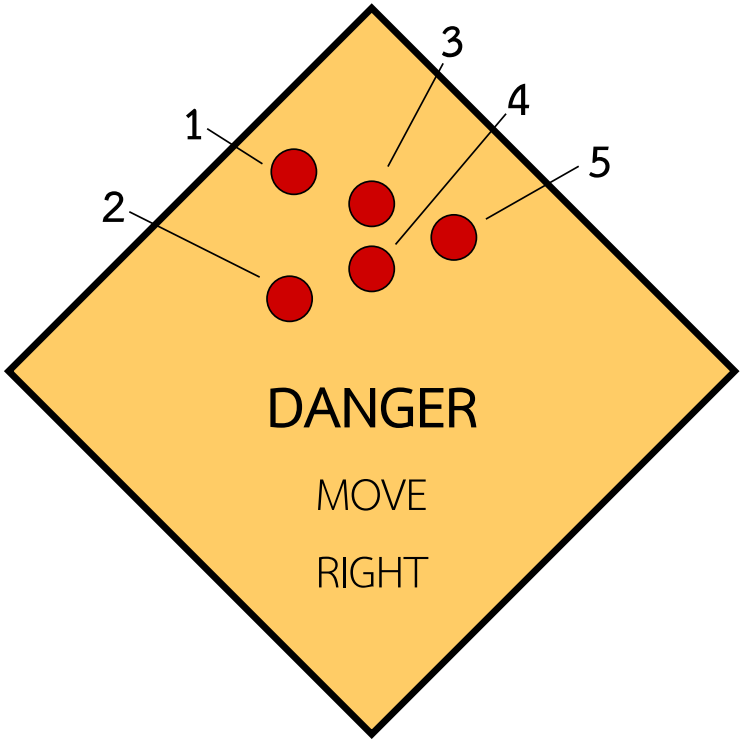


Clock	D	Q(t+1)
↓ or 0 or 1	X	Q_t (no change)
↑	0	0
↑	1	1

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

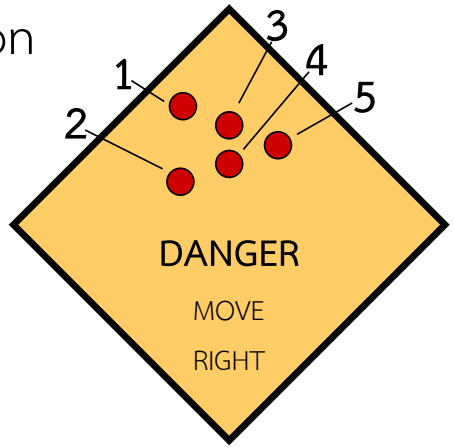
Example: A blinking traffic sign

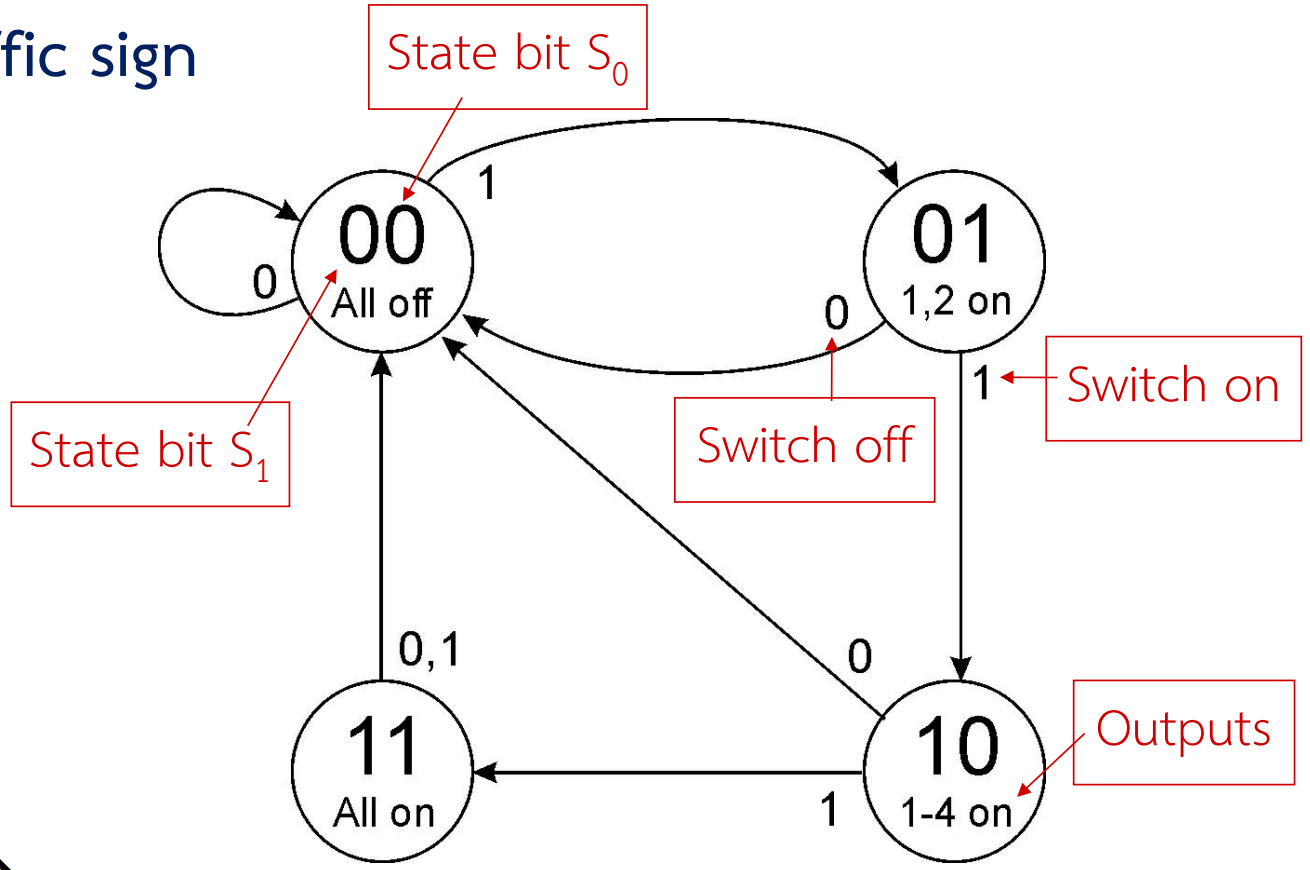
- No lights on
- 1 & 2 on
- 1, 2, 3, & 4 on
- 1, 2, 3, 4, & 5 on
- Repeat as long as **switch** is turned on



Example: A blinking traffic sign

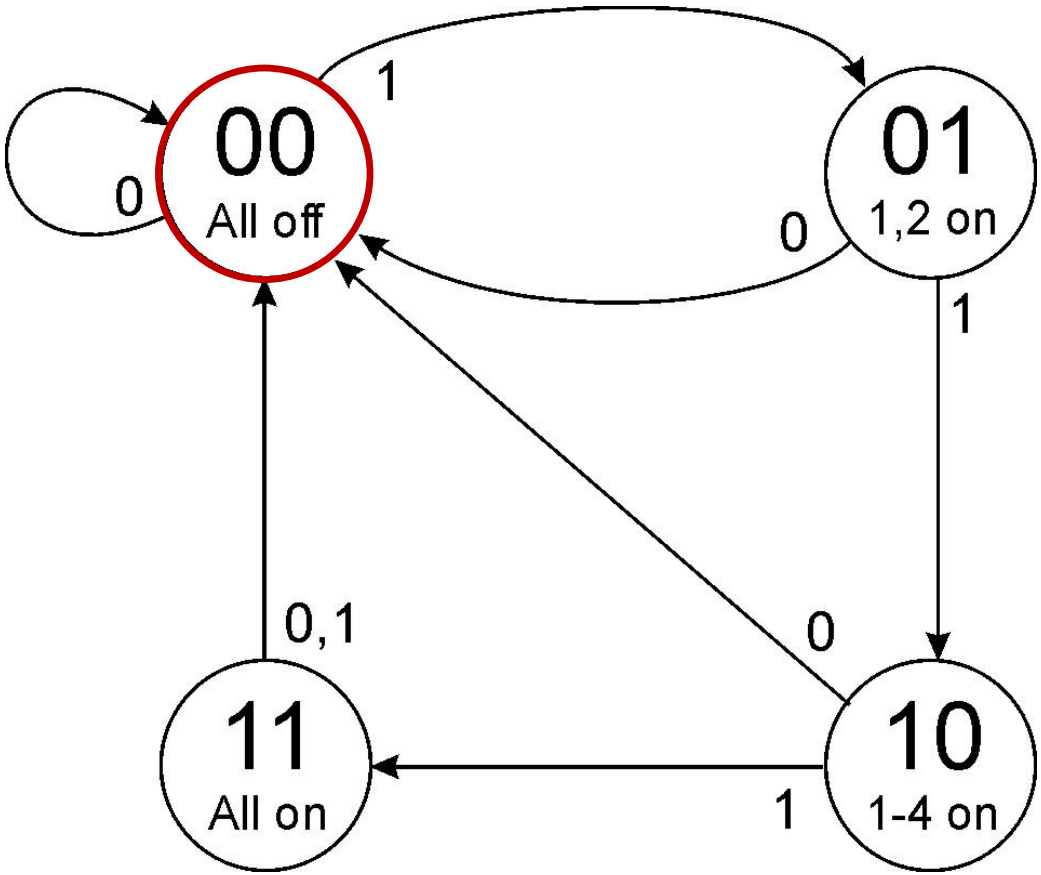
- No lights on
- 1 & 2 on
- 1, 2, 3, & 4 on
- 1, 2, 3, 4, & 5 on
- Repeat as long as **switch** is turned on





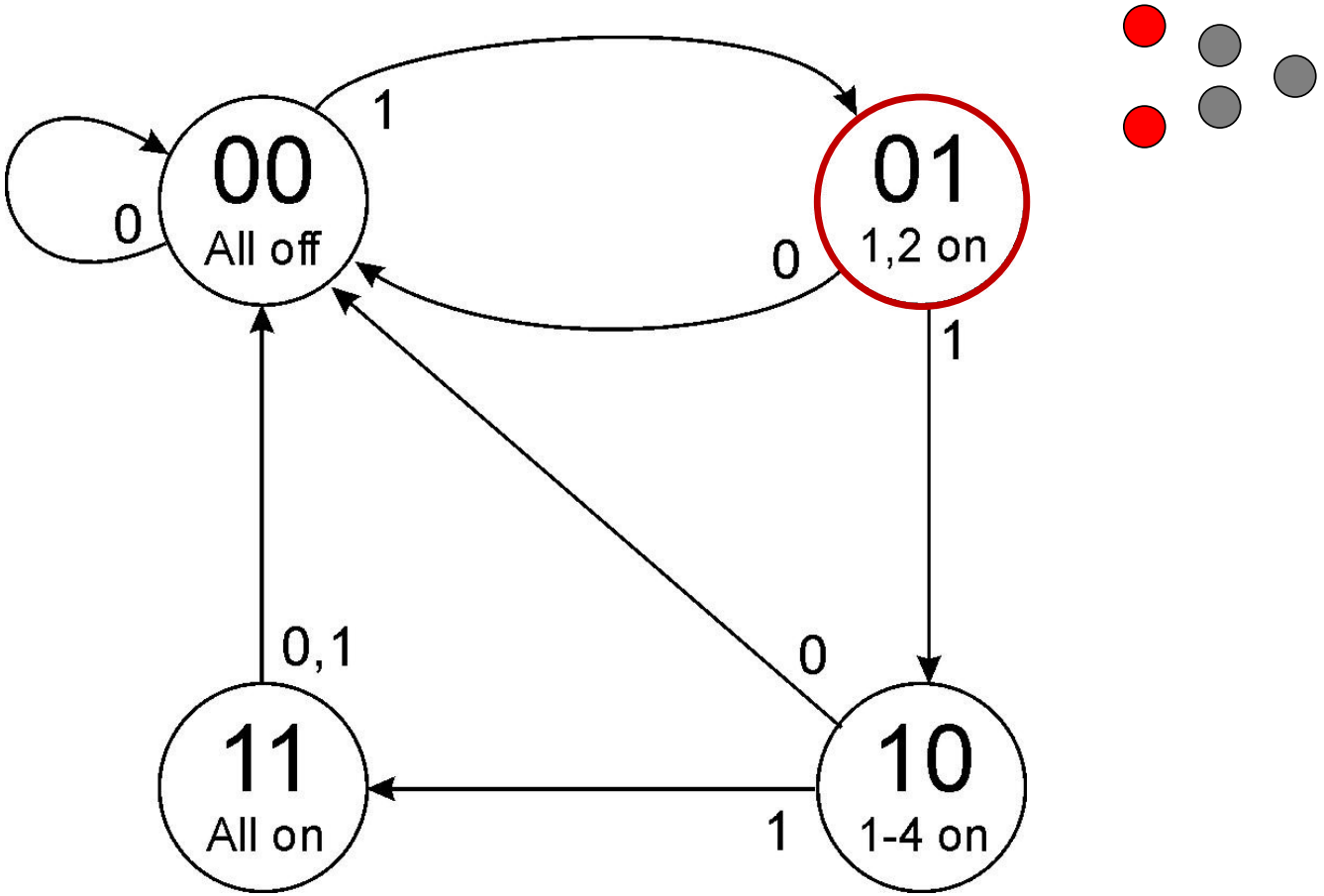
Transition on each clock cycle.

Traffic Sign State Diagram: State 00



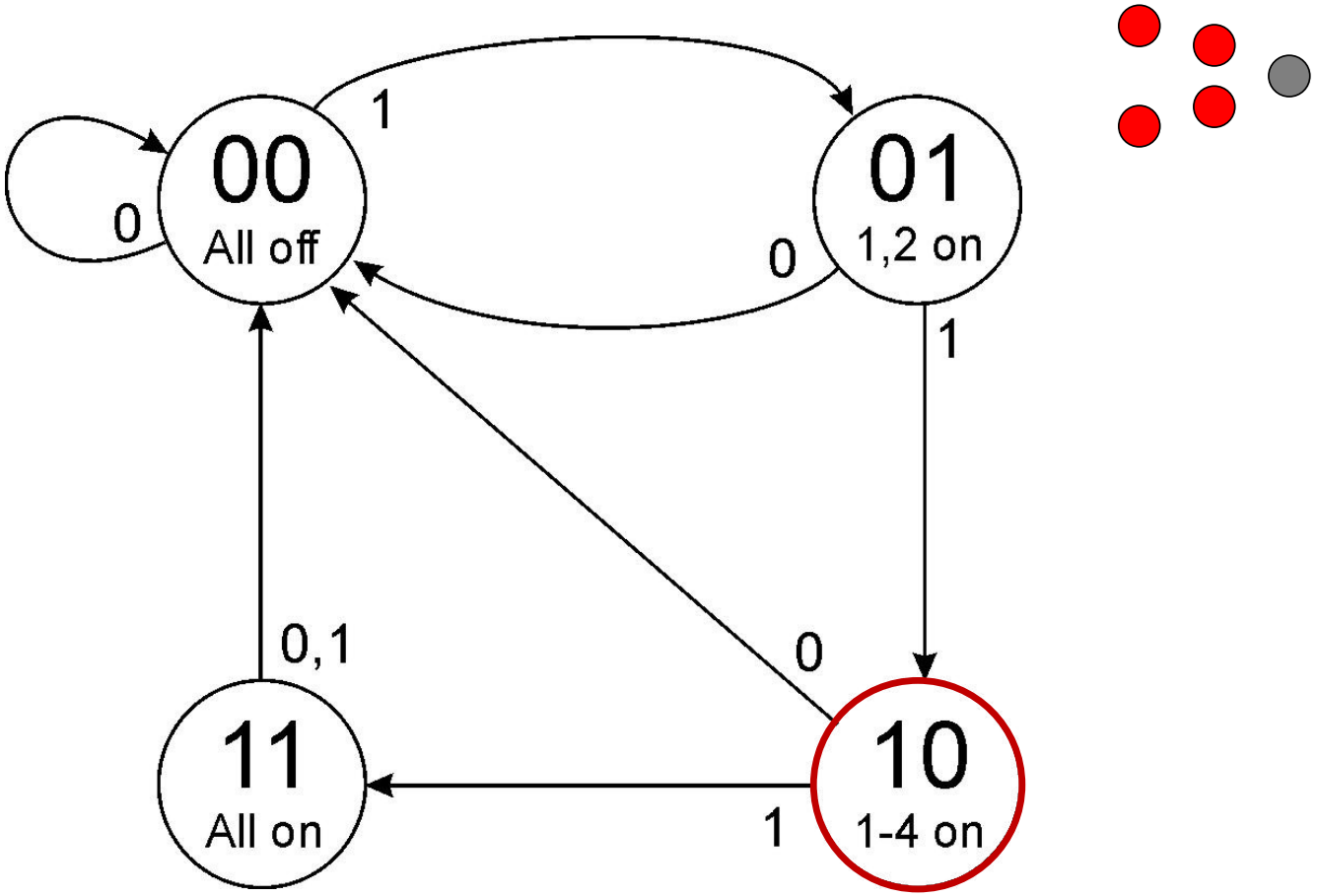
Transition on each clock cycle.

Traffic Sign State Diagram: State 01



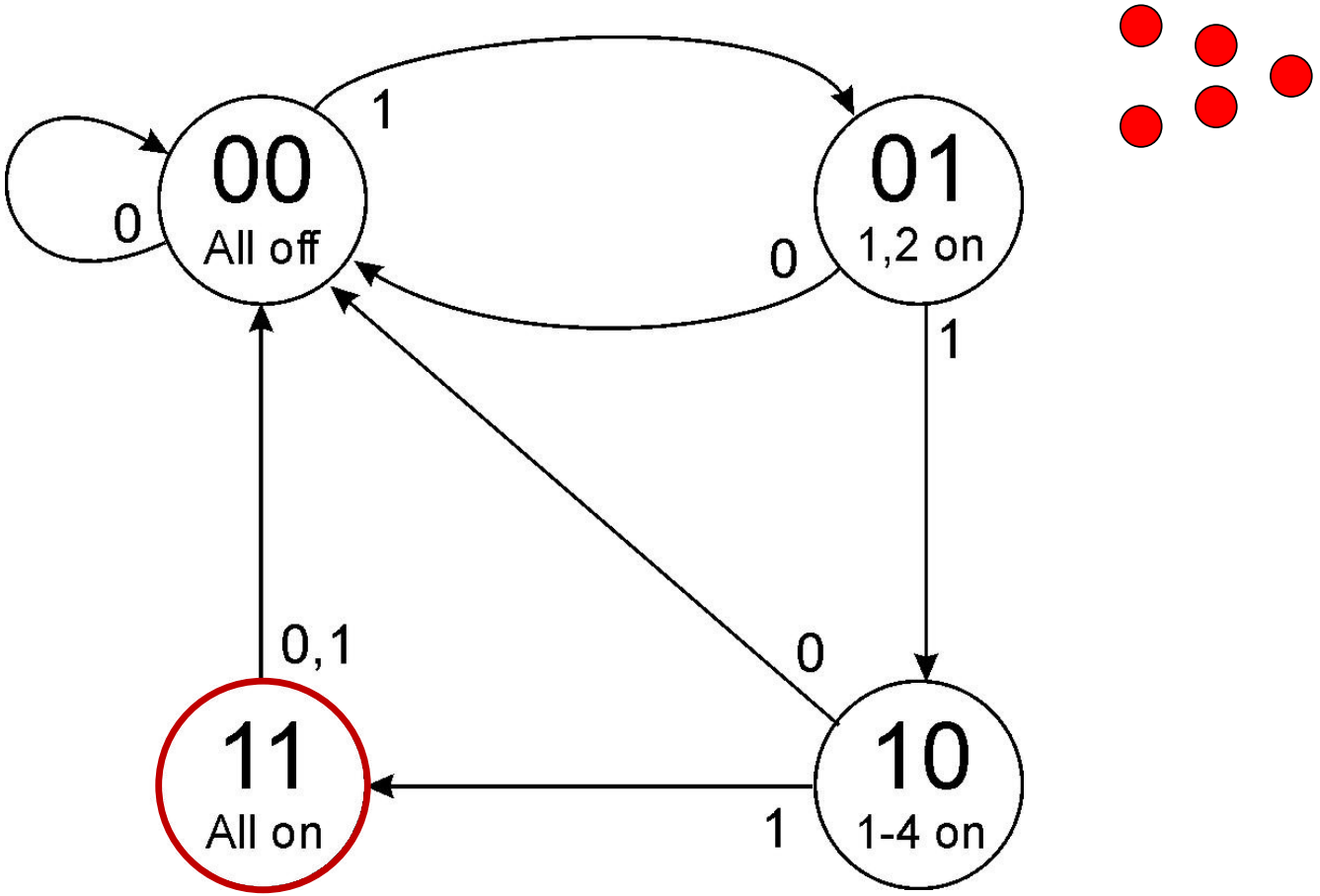
Transition on each clock cycle.

Traffic Sign State Diagram: State 10



Transition on each clock cycle.

Traffic Sign State Diagram: State 11



Transition on each clock cycle.

Traffic Sign Truth Tables

Outputs

(depend only on state: S_1S_0)

Lights 1 and 2

Lights 3 and 4

Light 5

S_1	S_0	W	X	V
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)

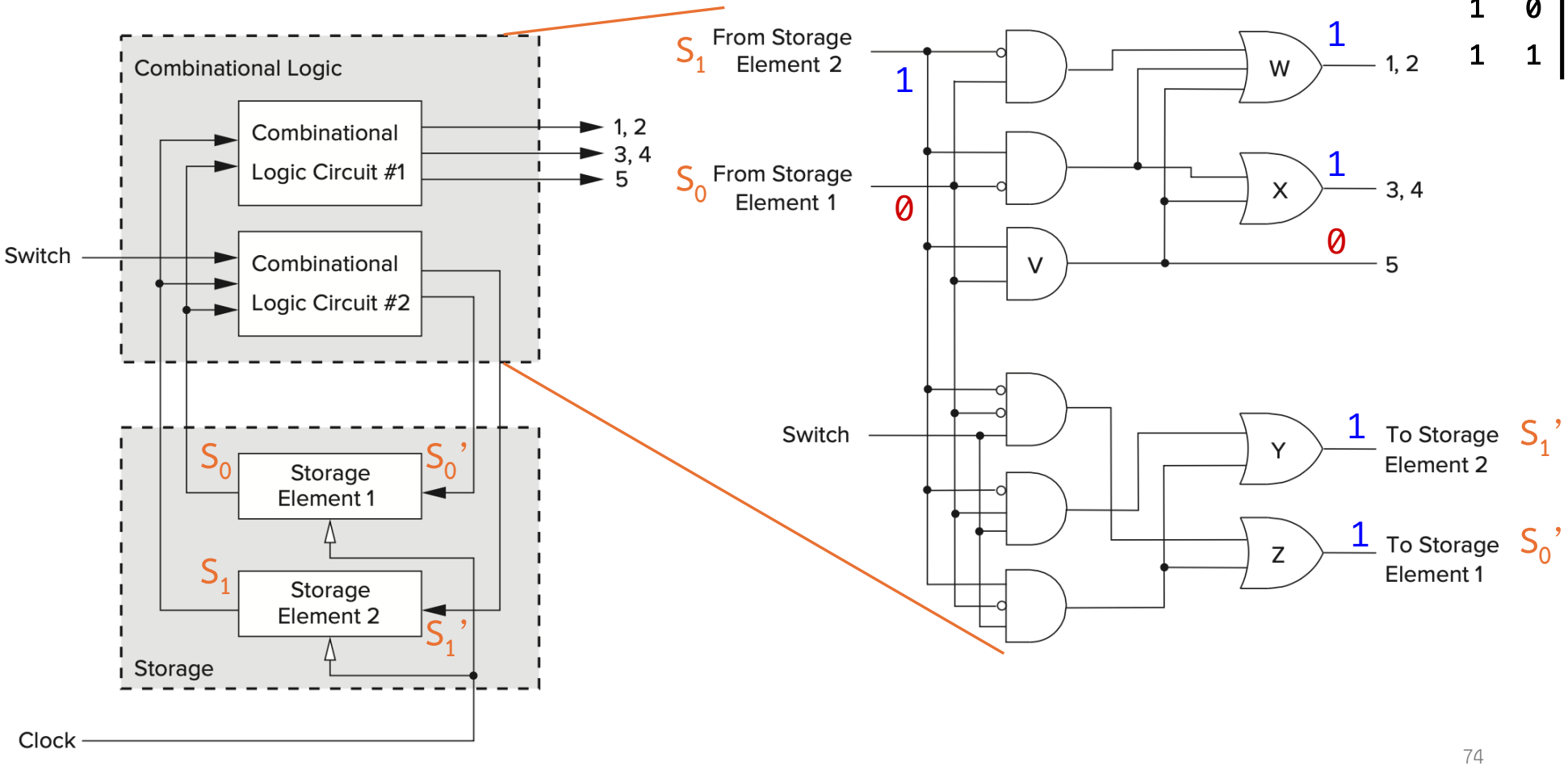
Switch

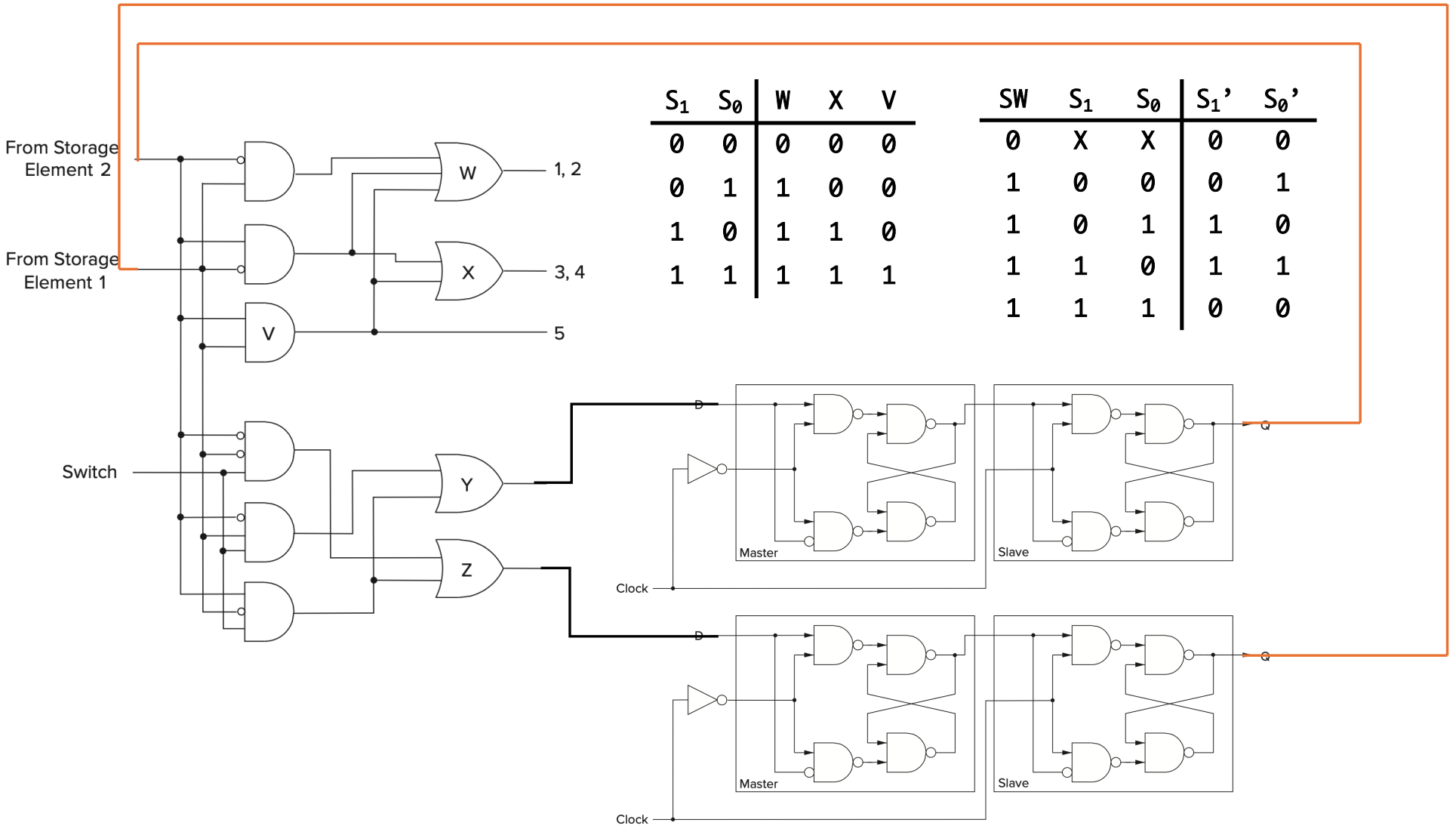
SW	S_1	S_0	S_1'	S_0'
0	X	X	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Whenever SW=0, next state is 00.

Sequential logic circuit for the danger sign controller.

S_1	S_0	W	X	V
0	0	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	1	1	1

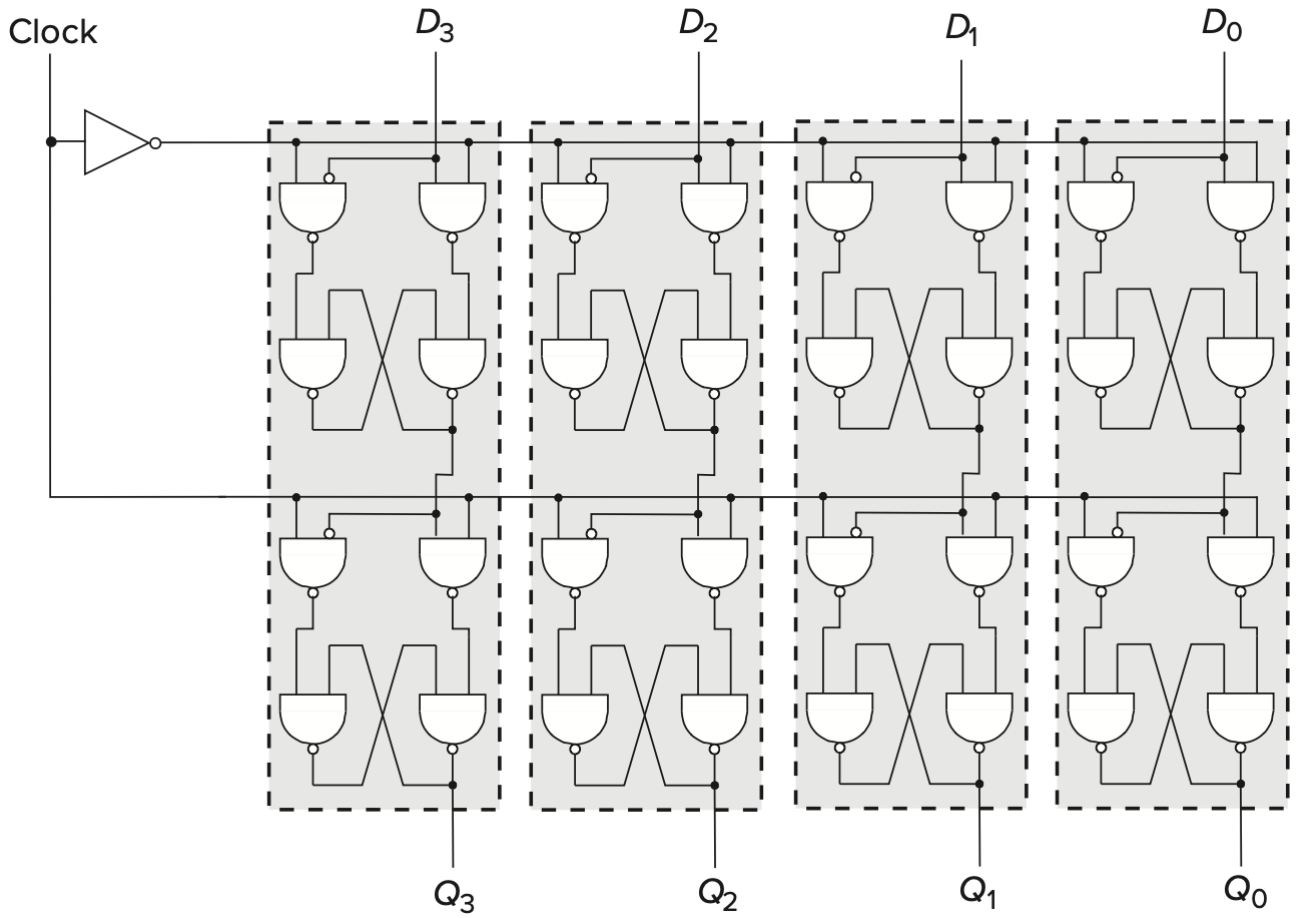




S_1	S_0	W	X	V
0	0	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	1	1	1

SW	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

4-bit Register



From Logic to Data Path

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

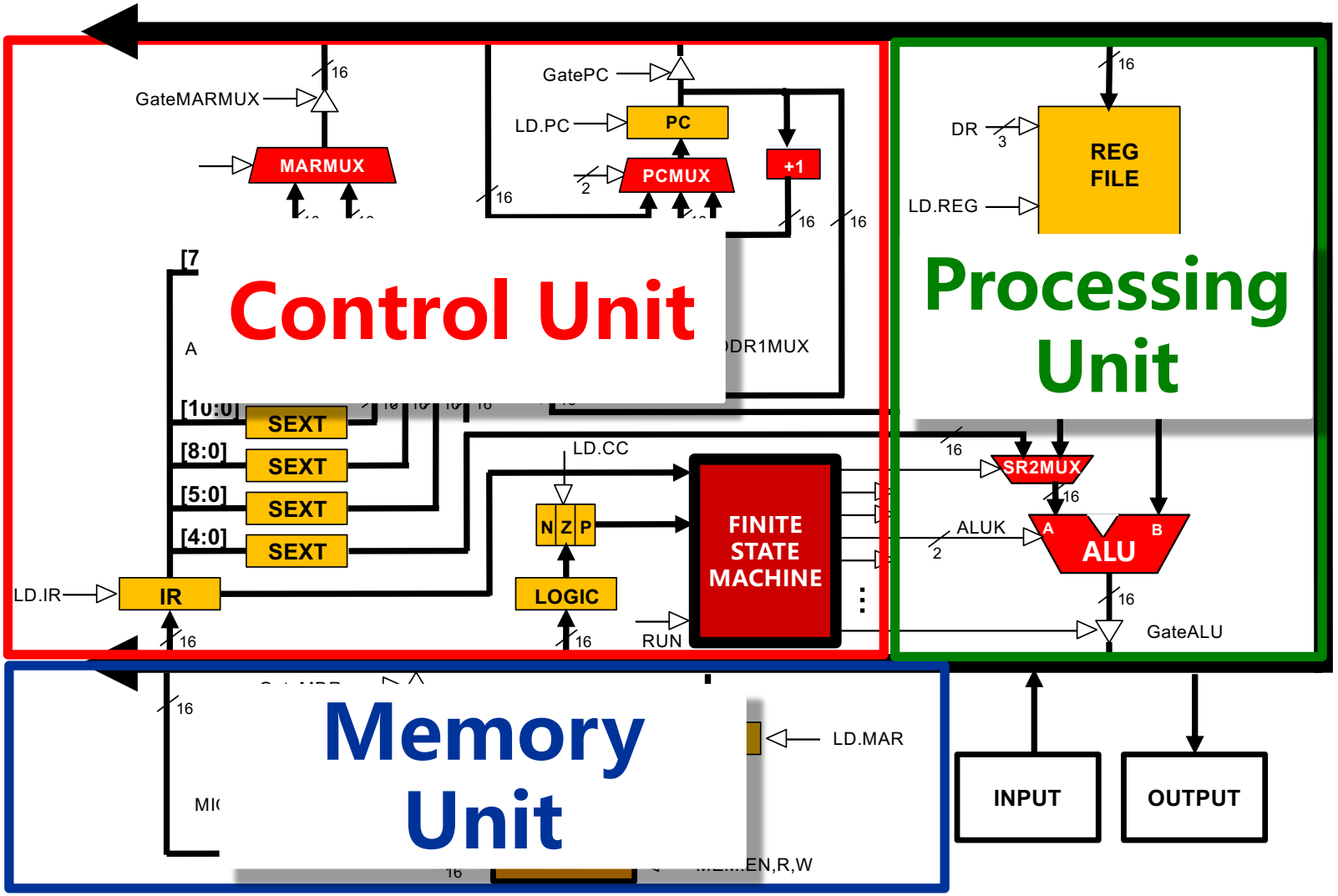
Combinational Logic

- 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)



The data path of the LC-3 computer

- Five MUXes
- An adder (shown as the ALU symbol with a + sign inside) and an ALU
- PC: Program Counter, IR: Instruction Register, MAR, and MDR are all 16-bit registers that store 16 bits of information each
- REG FILE consists of eight registers that each store 16 bits of information
- One bit of information can be stored in one flip-flop
- Three 1-bit registers, N, Z, and P (Negative Zero and Positive Flag)

Wrap-Up

- MOS transistors are used as switches to implement logic functions
- Logic functions are usually expressed with AND, OR, and NOT
- Basic digital logic
  - Transistors / Gates
  - Storage (latches, flip-flops, memory)
  - State machines
- Some simple circuits
  - Adder, subtracter, adder/subtracter, Incrementor
  - Traffic sign state machine

Homework

- Exercises:

  3.1, 3.2, 3.3, 3.5, 3.8, 3.10,

  3.25, 3.29, 3.30, 3.35, 3.40, 3.49