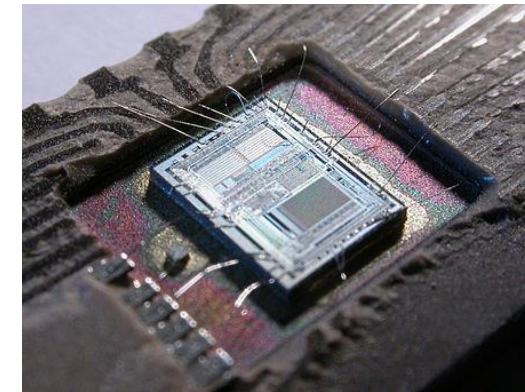
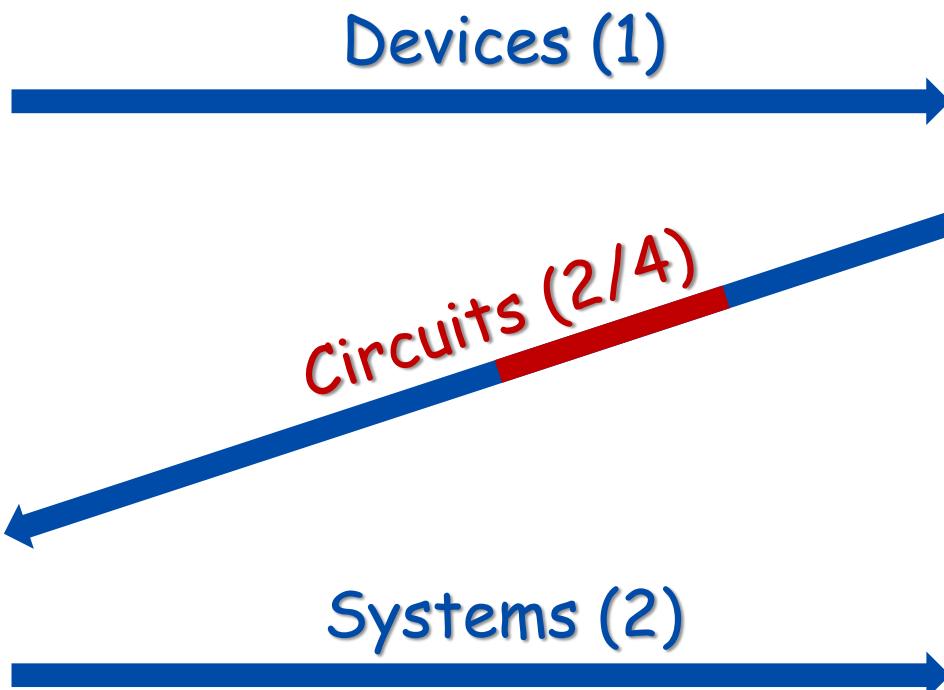


SI100B  
Introduction to Information  
Science and Technology  
(Electrical Engineering)

Lecture #4 (Digital)  
Sequential Logic Circuits

Instructor: Junrui Liang (梁俊睿)  
Oct. 15<sup>th</sup>, 2021

# The Theme Story



(Pictures are from the Internet)

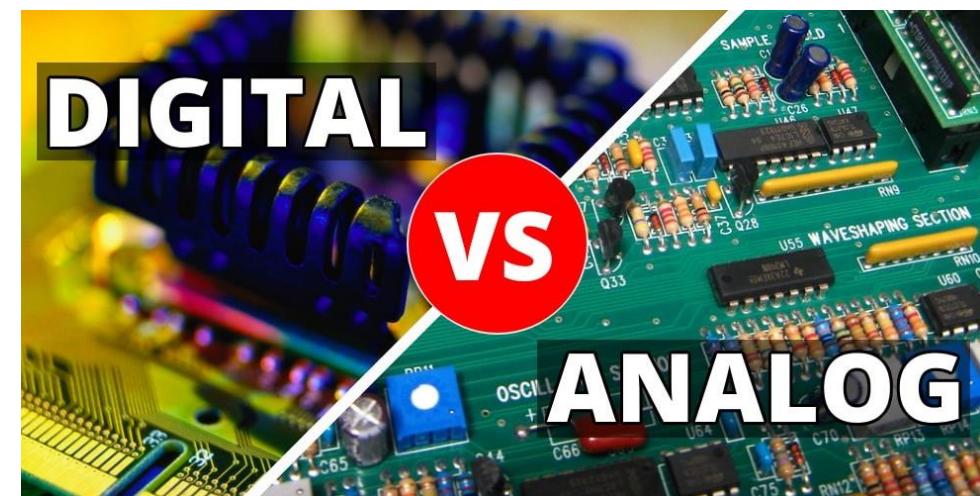
# Study Purpose of Lecture #4

- 哲学 (bao'an) 三问
  - Who are you?
  - Where are you from?
  - Where are you going?

To answer those questions  
throughout your life



- In this lecture, we ask
  - What is the fundamental difference between combinational and sequential logic circuits?
  - What is **latch** 锁存器? What is **flip-flop** 触发器?
  - How to use **finite state machine** 有限状态机 to build useful applications?

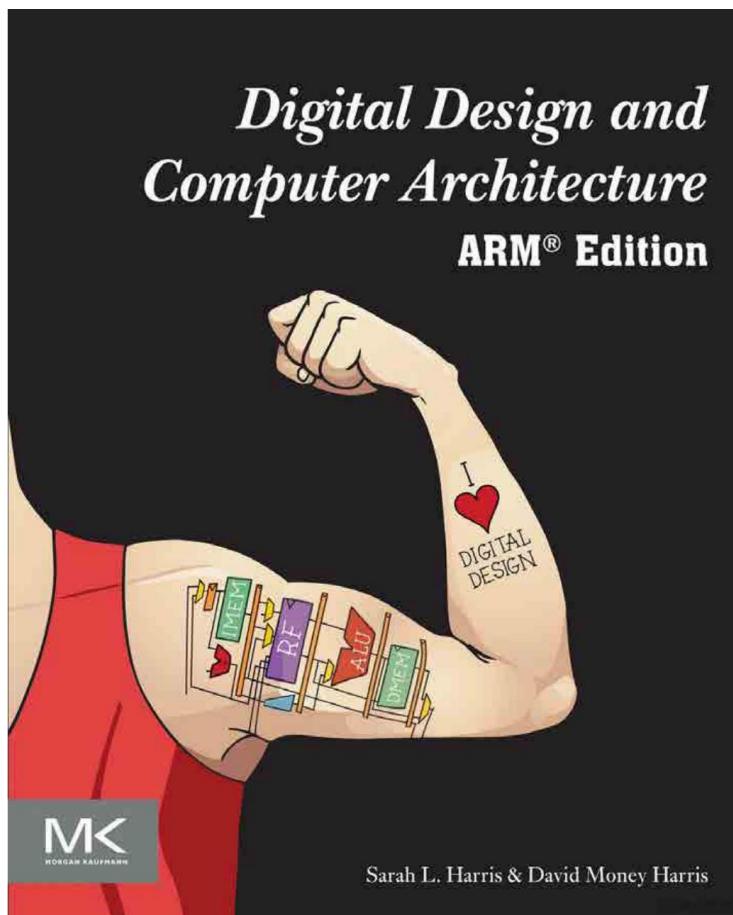


(Pictures are from the Internet)

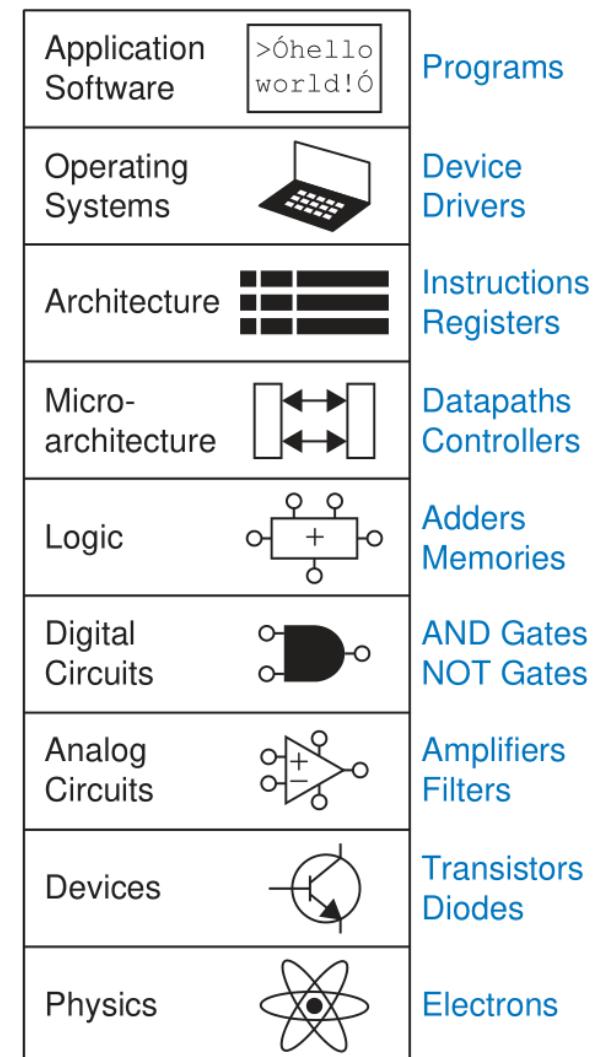
# Lecture Outline

- Different levels of abstraction (complementary)
- Computer configuration (complementary)
- Sequential logic circuits 时序逻辑电路
  - definition
- Latches and flip-flop 锁存器与触发器
  - The evolution of latches and flip-flops
  - Synchronous and asynchronous circuits 同步与异步电路
  - Example:  $4 \times 3$  memory
- Finite state machines (FSM) 有限状态机
  - Traffic light example

# Different levels of abstraction

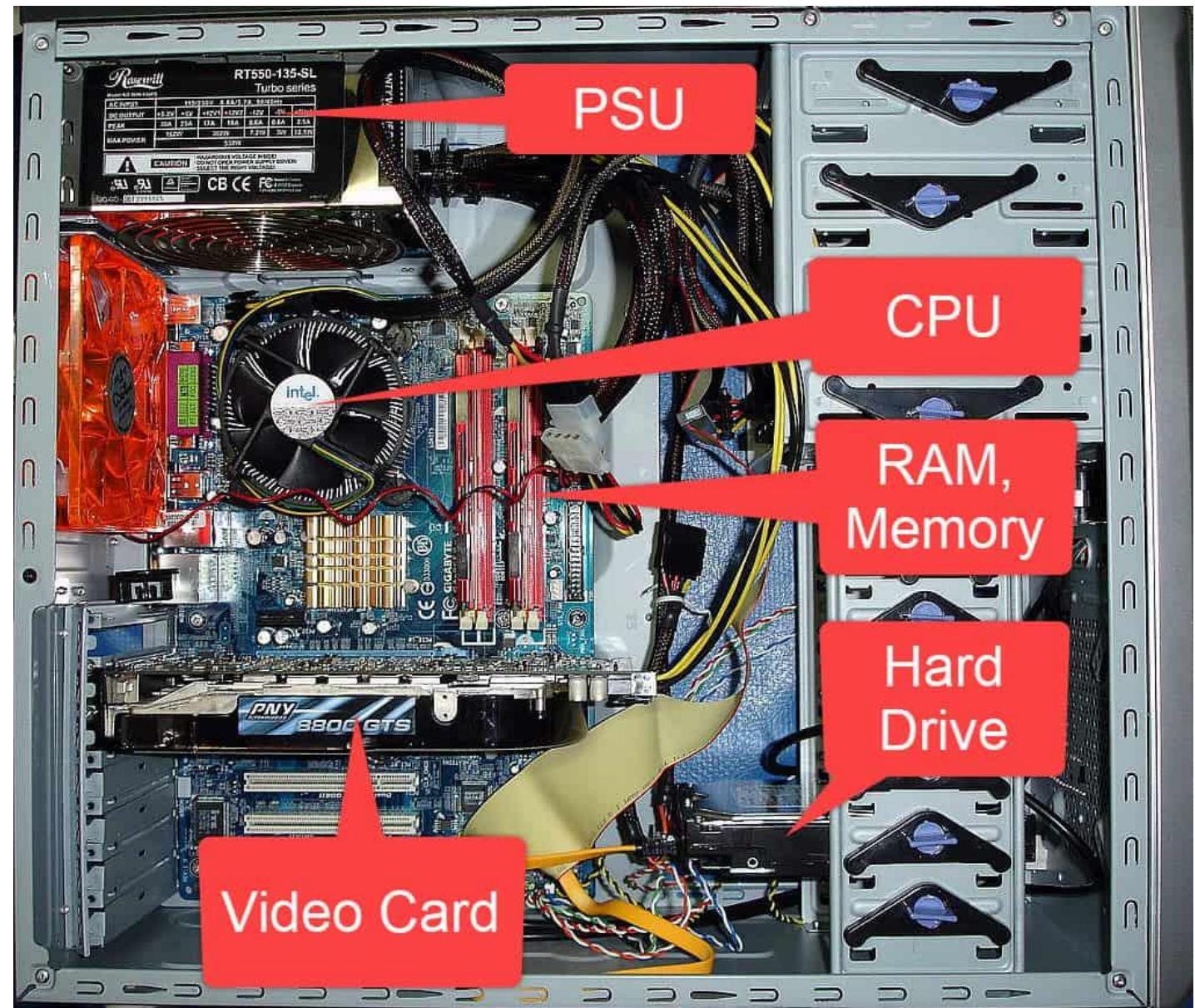
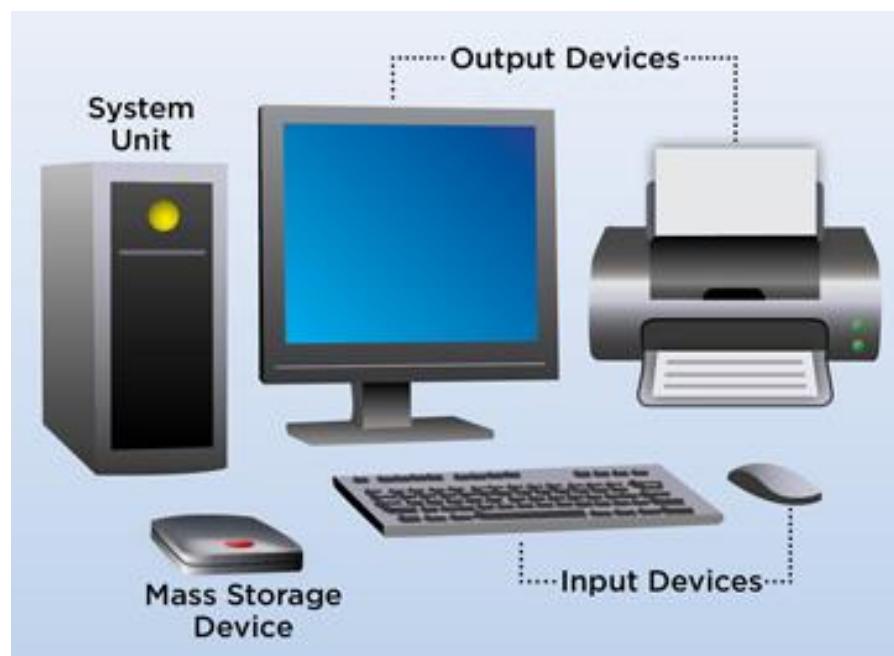


**Grasp your spoon!**

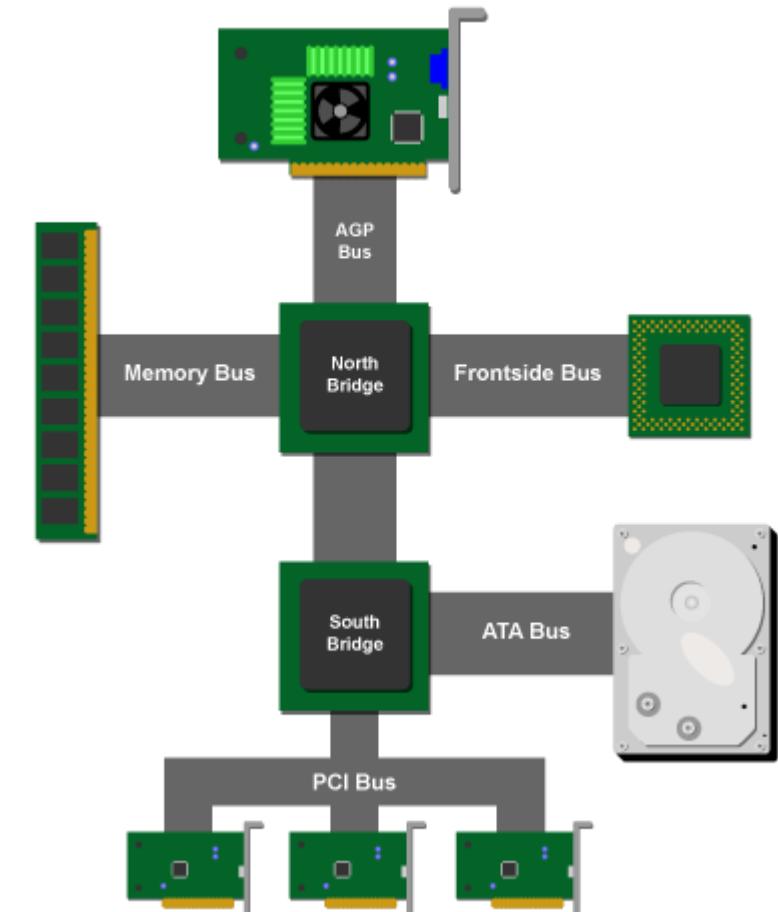
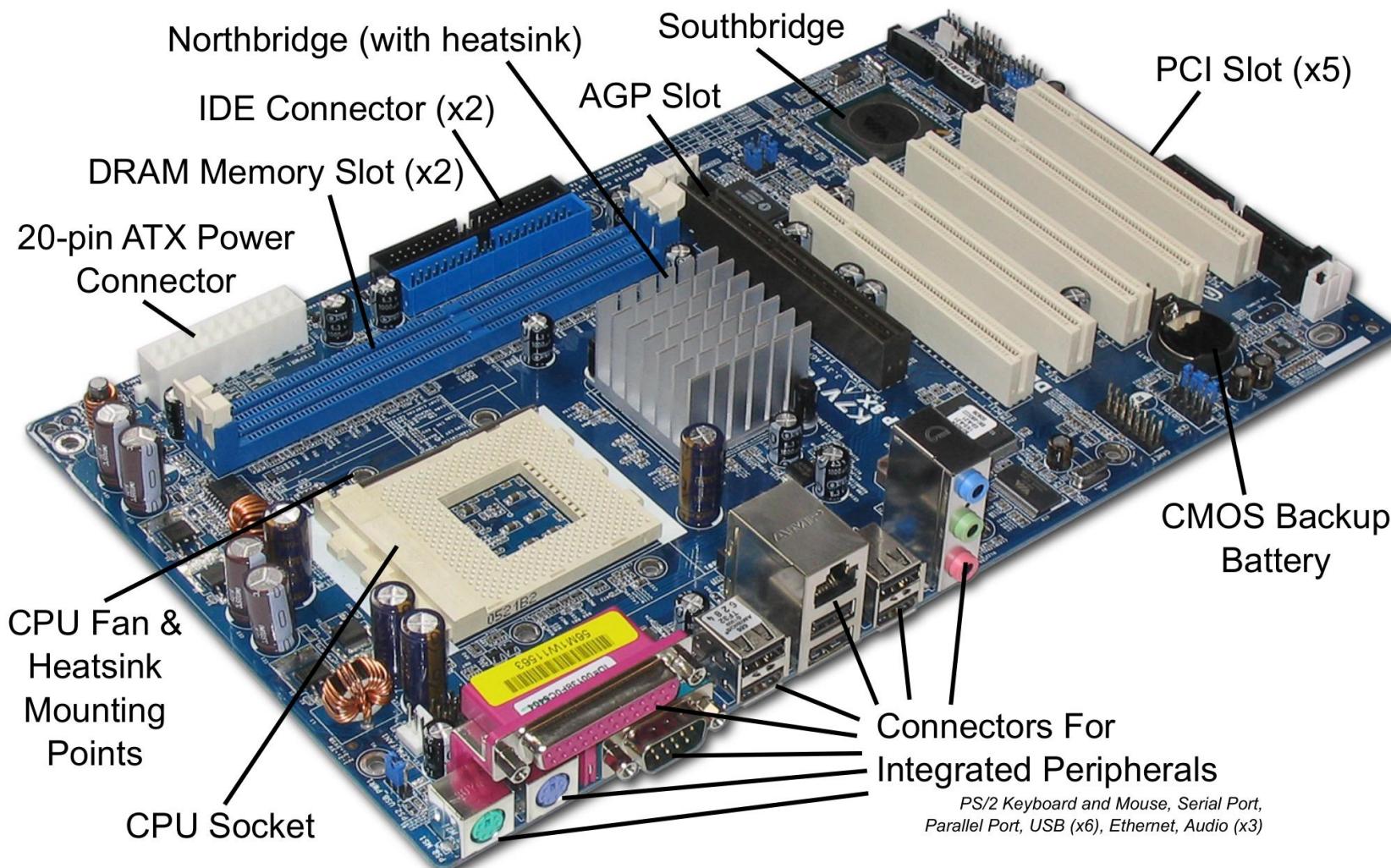


**Figure 1.1 Levels of abstraction  
for an electronic computing system**

# Inside a PC (personal computer)



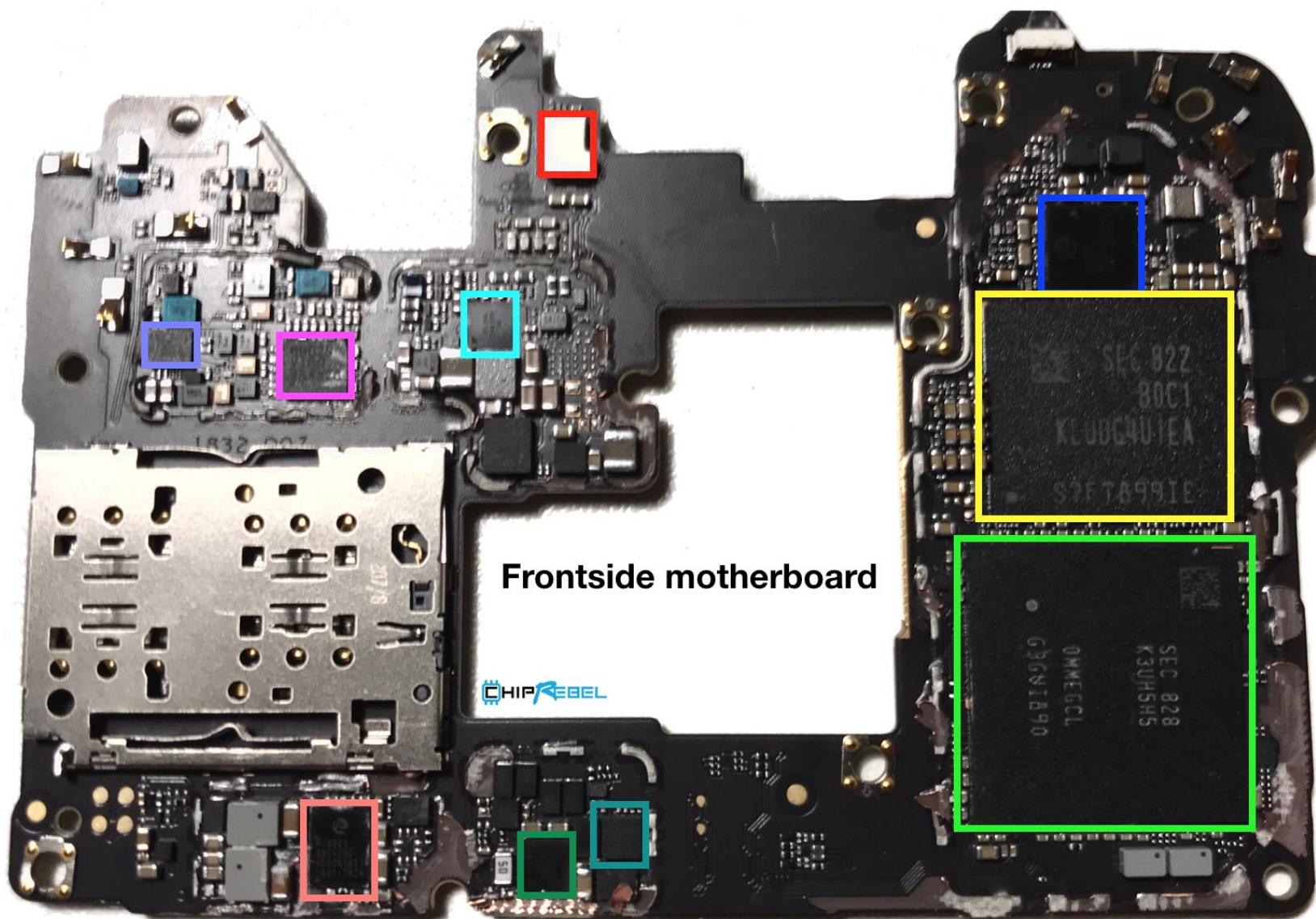
# A PC motherboard 主板



# Inside a cellphone (Huawei Mate 20 teardown)



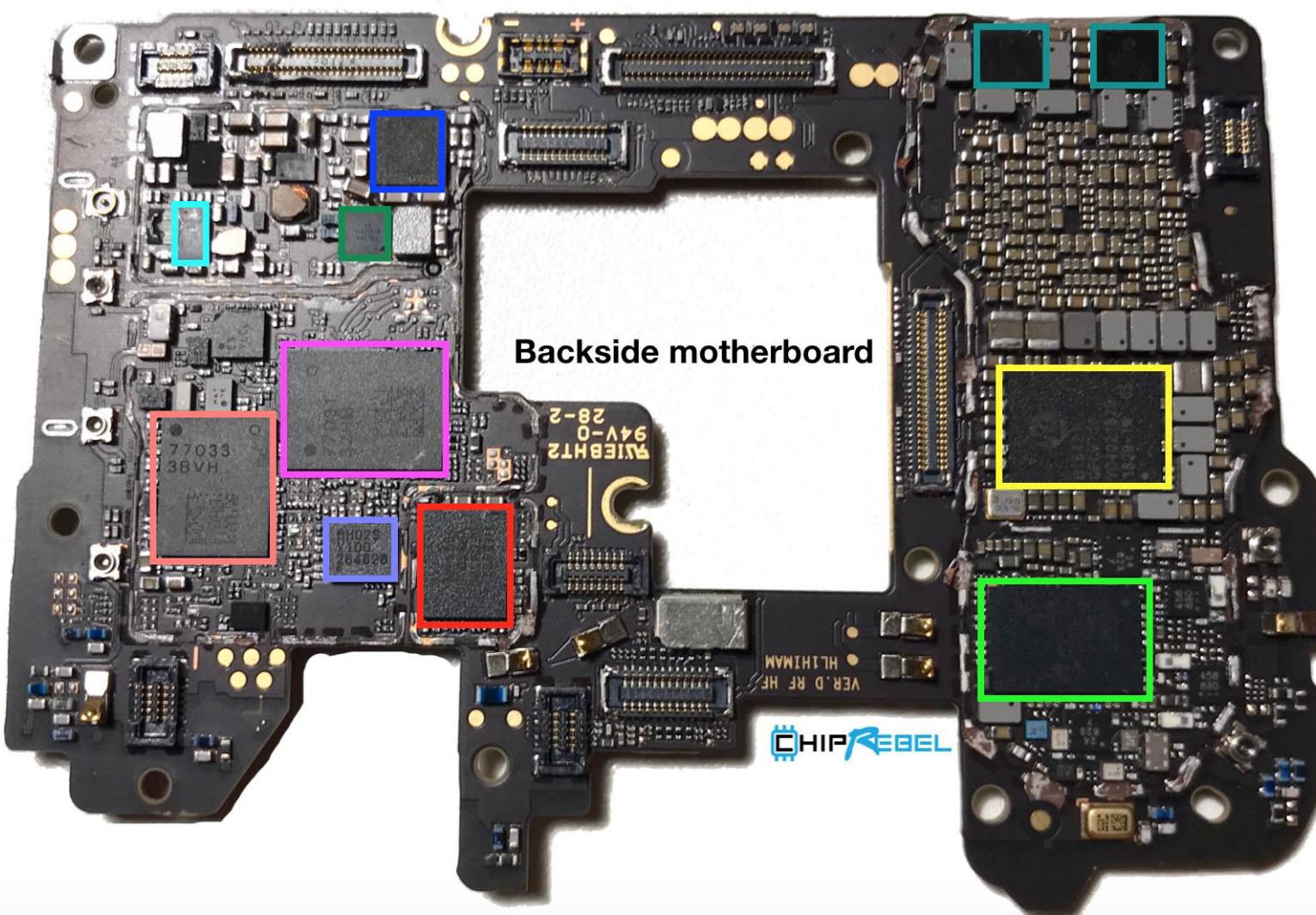
# Inside a cellphone (frontside)



## Huawei Mate 20 Chip package markings:

- SEC 828 K3UH5H5 which is Samsung's LPDDR4X SDRAM chip on top of the Kirin 980 AP. Package-on-Package
- SEC 822 BOC1 KLUDG4U1EA eUFS Samsung 128GB NAND Flash memory
- Hi 6403 Audio Codec
- 6H02s Y100
- 35L36A B1AE1815
- 9498UK T83206 Z1D808
- GJ9BK
- Hi 6523 Battery Charger
- M12x6Y2

# Inside a cellphone (backside)

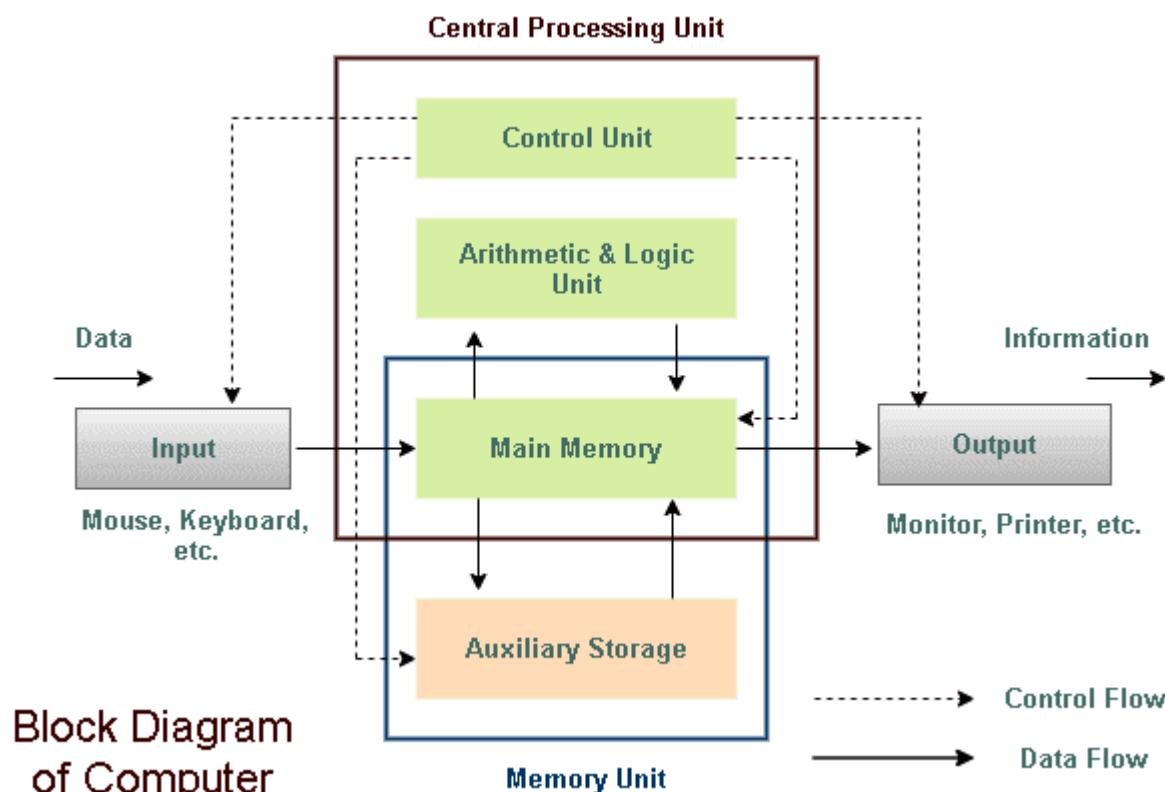


## Huawei Mate 20 Chip package markings:

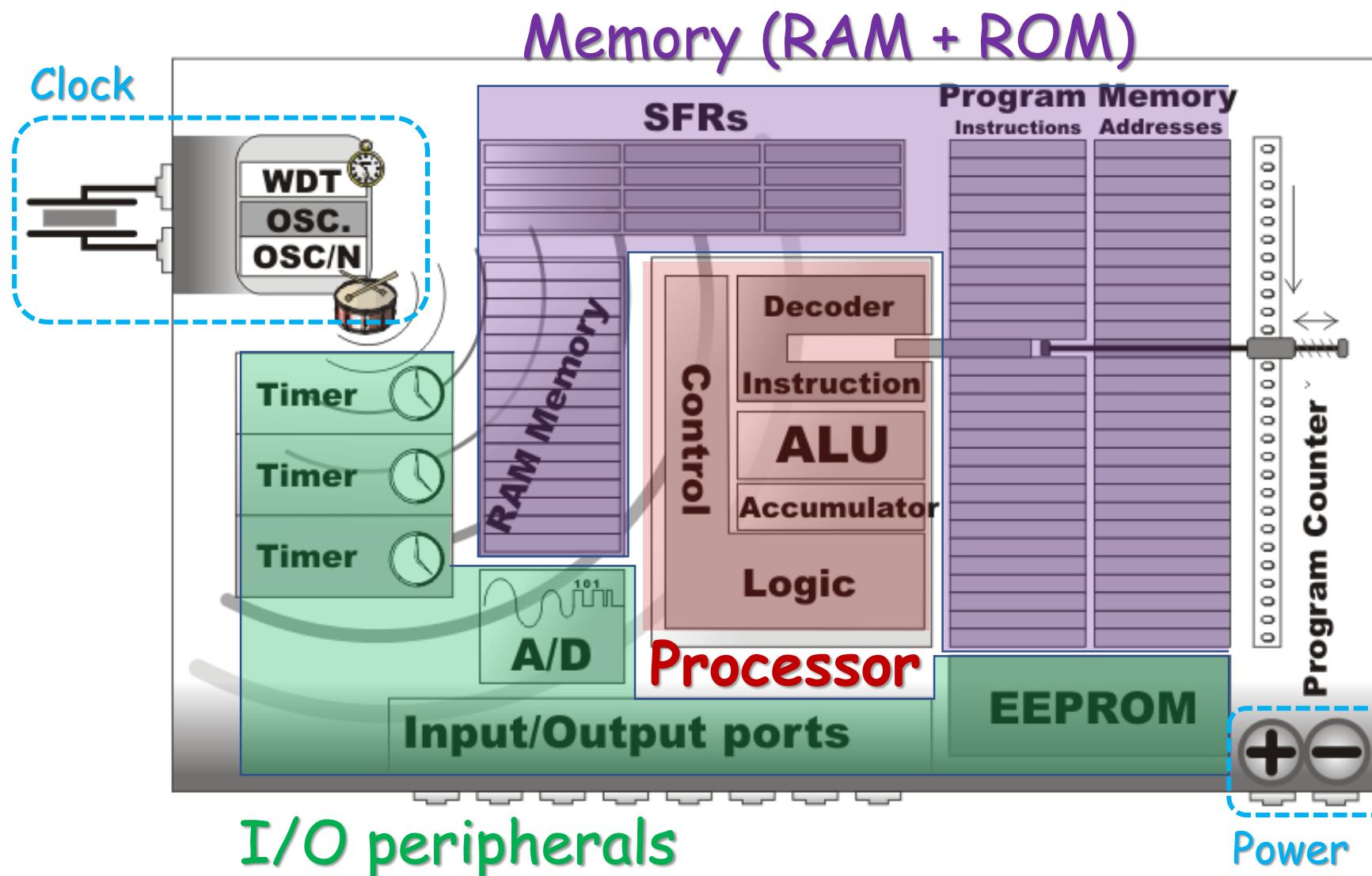
- Hi 1103 GFCV110  
HDGTC1830 WiFi chipset
- Hi 6421 6FCV7-16  
YD4021829 PMIC
- Hi 6422 Envelope tracker
- NXP 80T37 3403 SSD827  
NFC controller
- 35L36A B1AE1815
- 77031 32HM Qorvo Front  
End Module mid/high  
band
- 77033 38HV Qorvo Front  
End Module low/very low  
band
- 8H02S V100 284828
- Hi 6363 GFC V100  
YP8203830 RF  
Transceiver
- Qorvo Rf 8129 envelope  
tracker

# Computer Systems - Von Neumann Architecture 冯诺依曼结构

- Three major parts
  - CPU 中央处理器 (Central Processing Unit)
    - CU 控制单元 (Control Unit)
    - ALU 算术逻辑单元 (Arithmetic Logical Unit)
    - Register 寄存器
  - Memory 存储器
  - I/O 输入输出设备 (Input/Output)

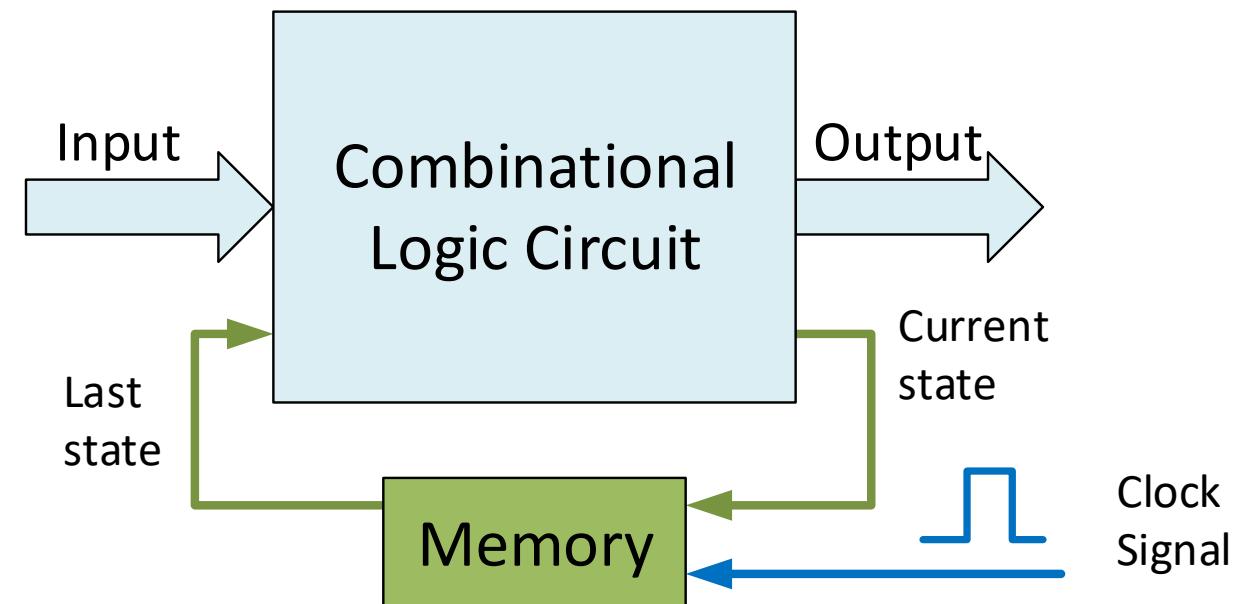
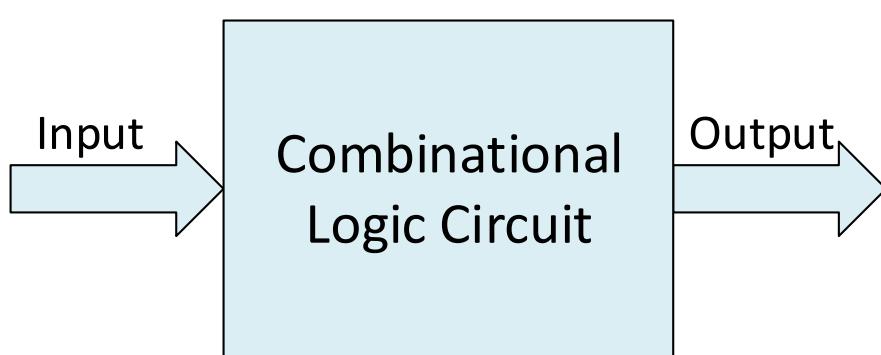


# How does a CPU work?



# Combinational Logic 组合逻辑 vs. Sequential Logic 时序逻辑

- Combinational logic circuit
  - Output depends on
    - Present input values
  - No memory
- Sequential Logic circuit
  - Output depends on
    - Present inputs
    - The history of past inputs
  - With memory

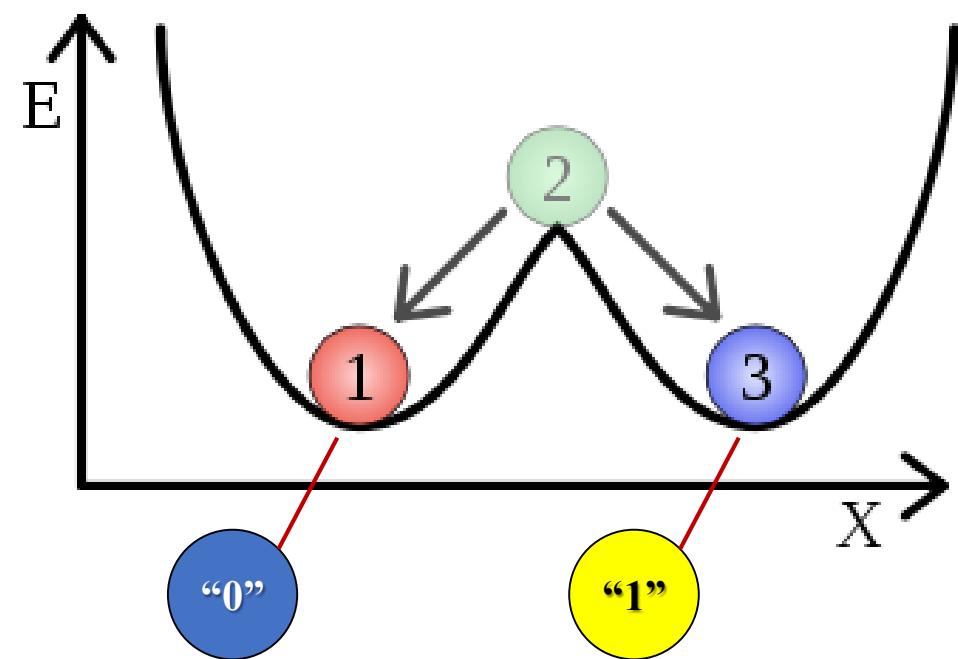


(Pictures are from the Internet)

# The original latch 门闩



- Bistable 双稳态 element



## Why designing latches?

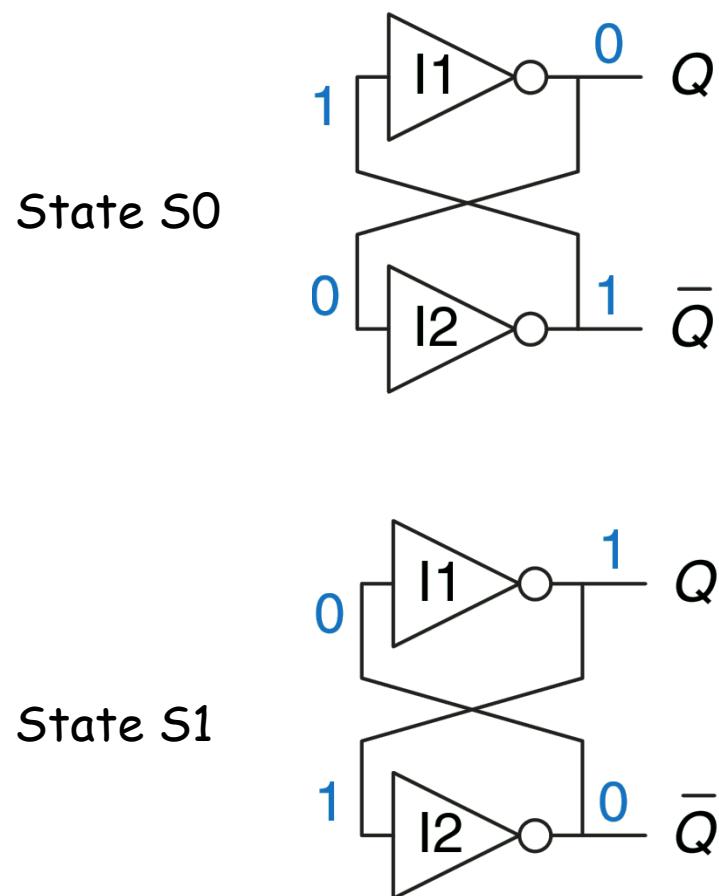
- The fundamental building block of memory is a bistable element, an element with two stable states (either "0" or "1").

Memorizing the state at either 1 or 3 position

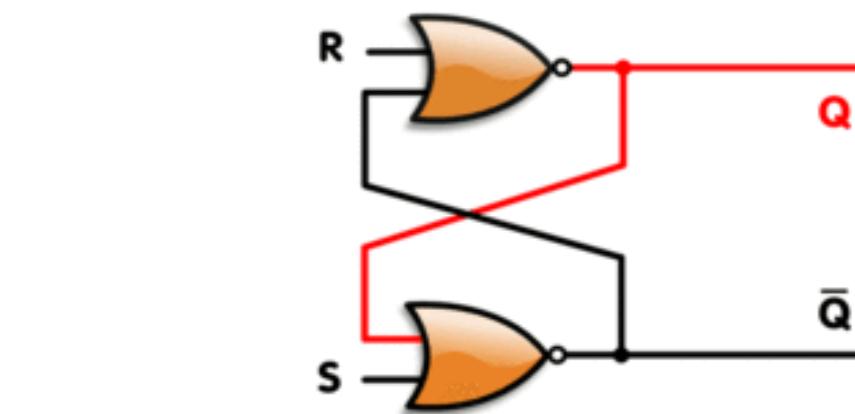
(Pictures are from the Internet)

# Digital latch 数字锁存器

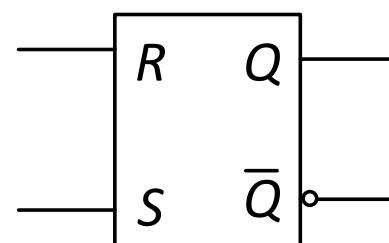
- Cross-coupled inverter pair



- SR latch (with input)
  - S: set (set output to 1) 置1
  - R: reset (set output to 0) 置0



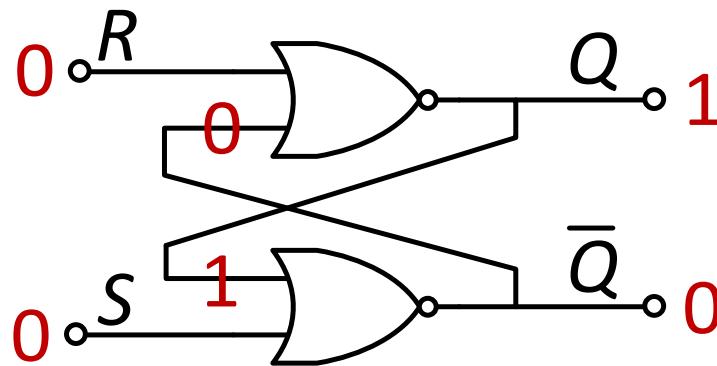
- Symbol



(Pictures are from the Internet)

# SR latch (when S=0, R=0)

- $Q=1$

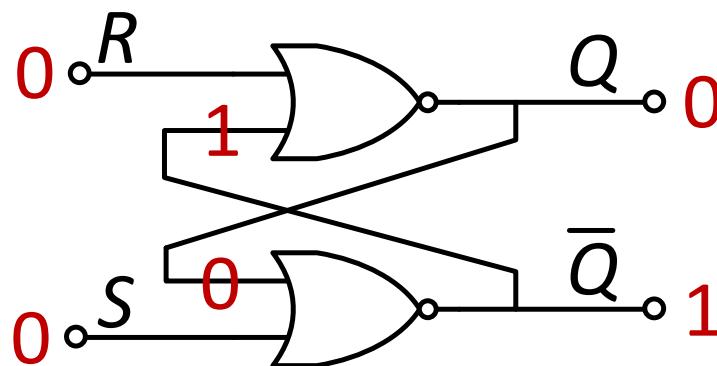


- Truth table

$S$	$R$	$Q$	$\bar{Q}$
0	0	$Q_{prev}$	$\bar{Q}_{prev}$

-- Hold state 保持状态

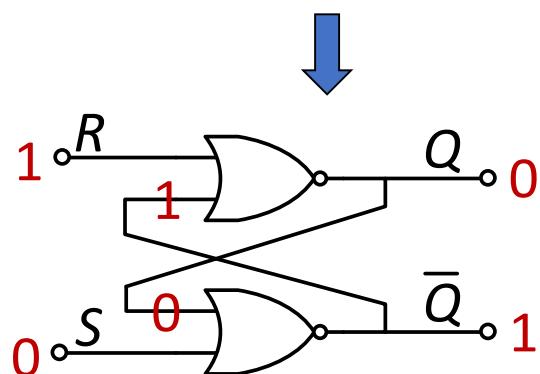
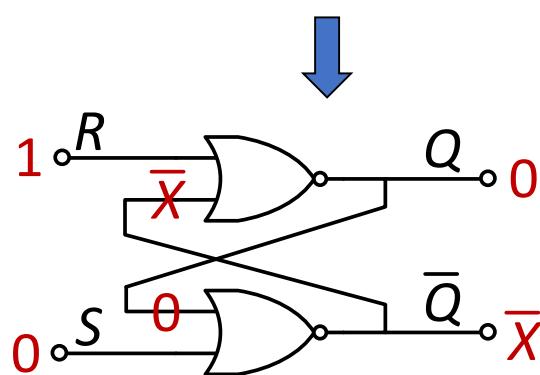
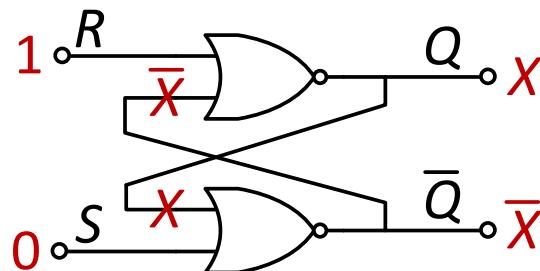
- $Q=0$



(Pictures are from the Internet)

# SR latch (when S=0, R=1)

- Truth table



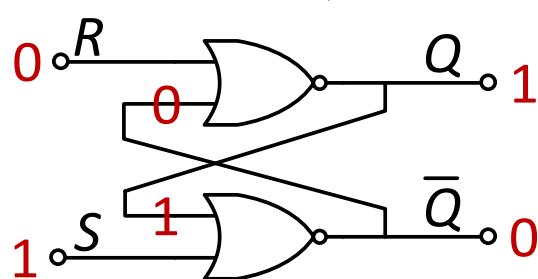
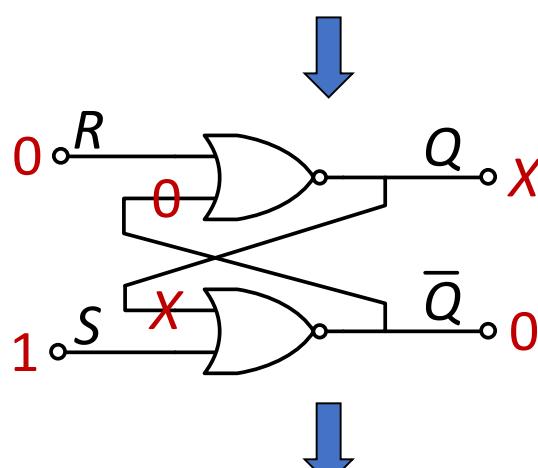
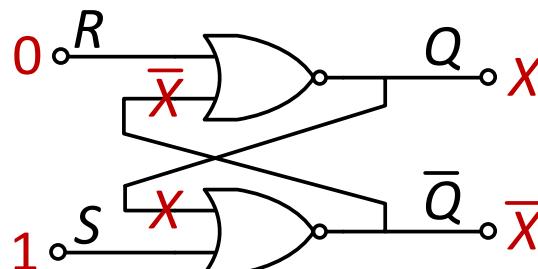
$S$	$R$	$Q$	$\bar{Q}$
0	0	$Q_{prev}$	$\bar{Q}_{prev}$
0	1	0	1

-- Hold state 保持状态  
-- Reset 重置 (清零)

(Pictures are from the Internet)

# SR latch (when S=1, R=0)

- Truth table



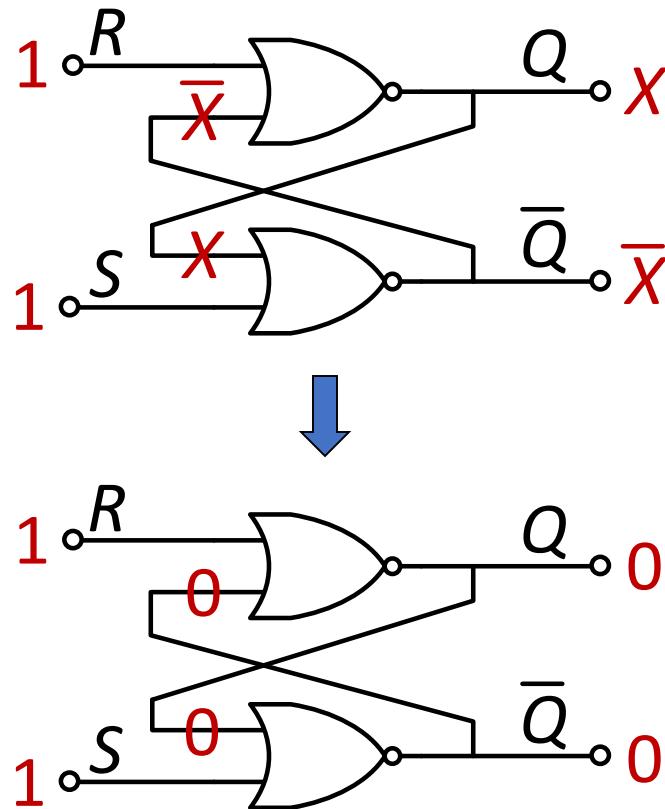
S	R	Q	$\bar{Q}$
0	0	$Q_{prev}$	$\bar{Q}_{prev}$
0	1	0	1
1	0	1	0

-- Hold state 保持状态  
 -- Reset 重置 (清零)  
 -- Set 置位 (置一)

(Pictures are from the Internet)

# SR latch (when $S=1, R=1$ )

- Truth table



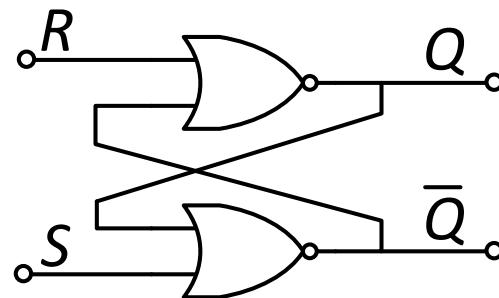
$S$	$R$	$Q$	$\bar{Q}$
0	0	$Q_{prev}$	$\bar{Q}_{prev}$
0	1	0	1
1	0	1	0
1	1	0	0

-- Hold state 保持状态  
-- Reset 重置 (清零)  
-- Set 置位 (置一)  
-- Not allow!

(Pictures are from the Internet)

# Timing control

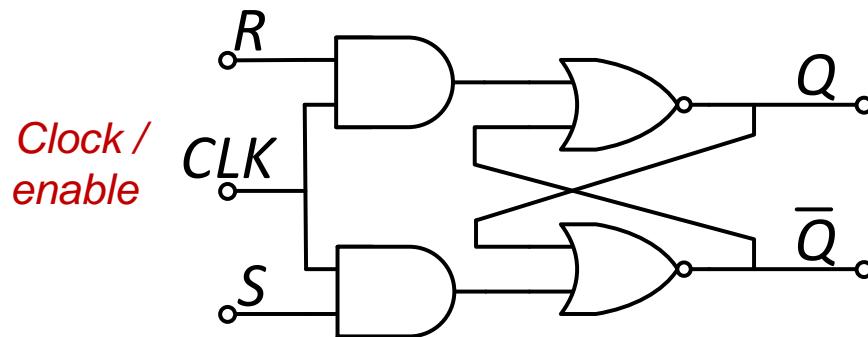
- Transparent latch



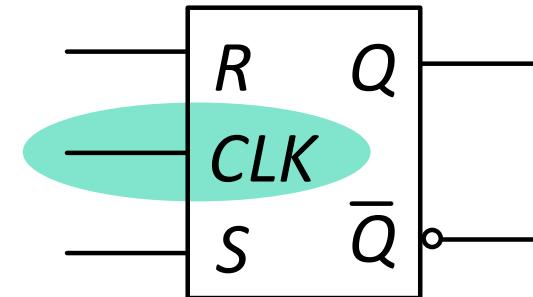
- Operation

E/C	Action
0	No action (keep state)
1	The same as non-clocked SR latch

- Gated SR Latch (level-sensitive)



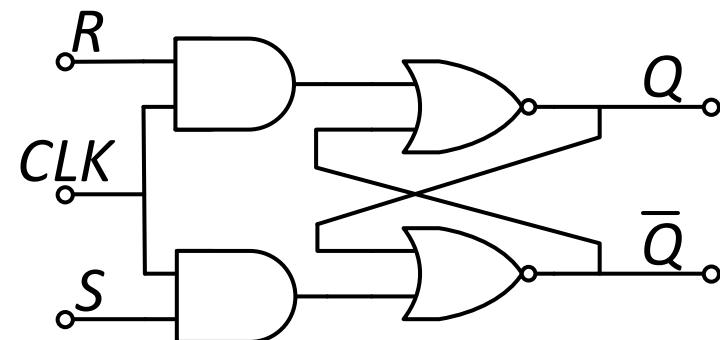
- Symbol



(Pictures are from the Internet)

# State determination → D latch

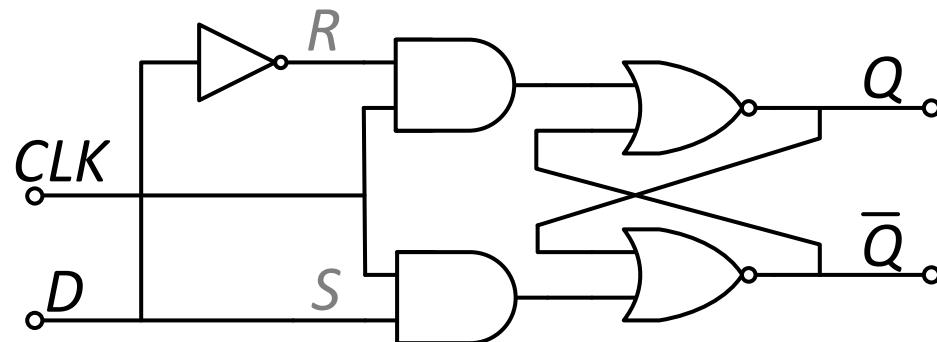
- Avoid the  $R=1, S=1$  confusion



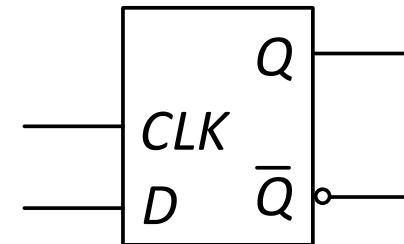
- Operation

$CLK$	$D$	$\bar{D}$	$S$	$R$	$Q$	$\bar{Q}$
0	X	$\bar{X}$	0	0	$Q_{prev}$	$\bar{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

- Modification → D latch

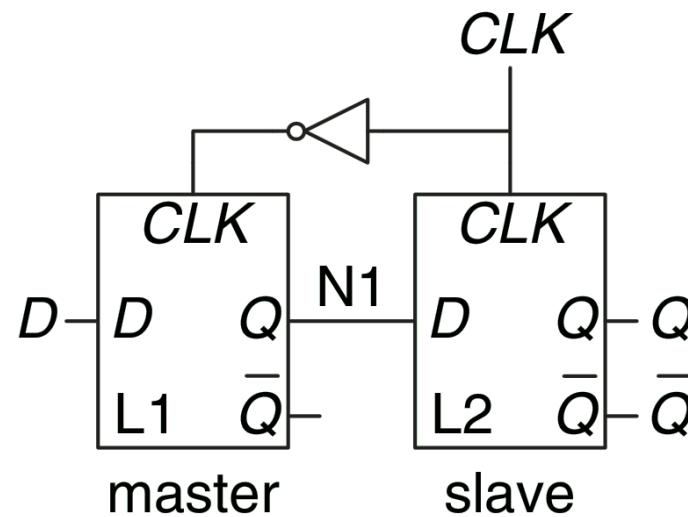


- Symbol

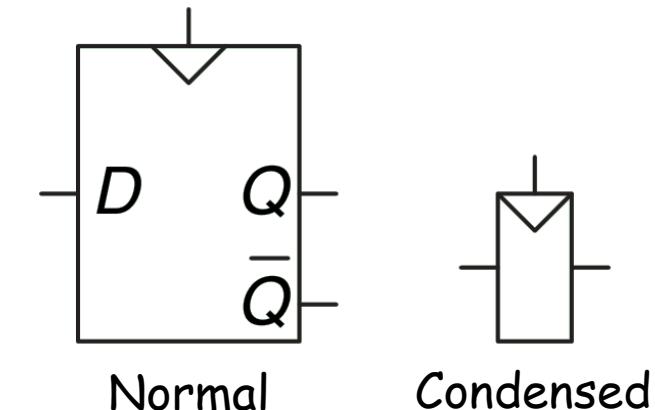


# More accurate timing control → edge-triggering

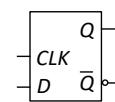
- D flip-flop 触发器



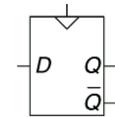
- Symbols



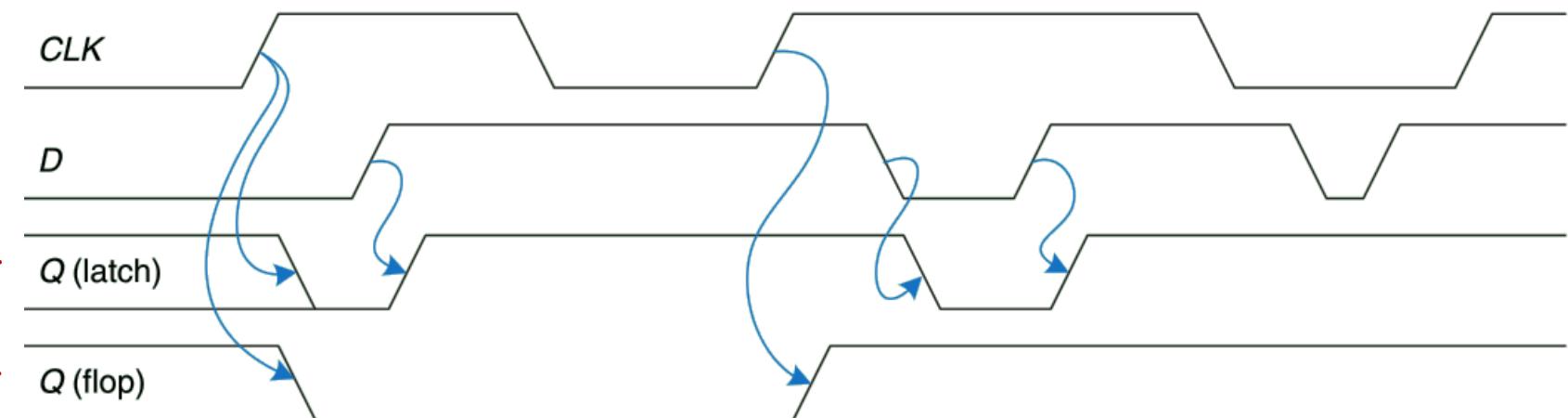
- Timing waveform



D latch output

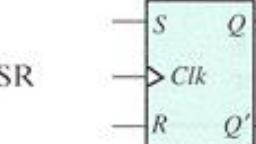


D flip-flop output

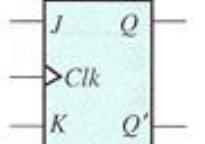


# Other flip-flops

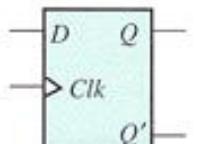
- SR (Set & Reset)

FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC TABLE	CHARACTERISTIC EQUATION	EXCITATION TABLE																																			
SR		<table border="1"> <thead> <tr> <th>S</th> <th>R</th> <th>Q(next)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>NA</td> </tr> </tbody> </table>	S	R	Q(next)	0	0	Q	0	1	0	1	0	1	1	1	NA	$Q(next) = S + R'Q$ $SR = 0$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q(next)</th> <th>S</th> <th>R</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>X</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>X</td> <td>0</td> </tr> </tbody> </table>	Q	Q(next)	S	R	0	0	0	X	0	1	1	0	1	0	0	1	1	1	X	0
S	R	Q(next)																																					
0	0	Q																																					
0	1	0																																					
1	0	1																																					
1	1	NA																																					
Q	Q(next)	S	R																																				
0	0	0	X																																				
0	1	1	0																																				
1	0	0	1																																				
1	1	X	0																																				

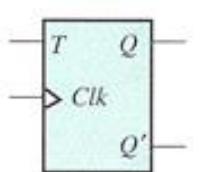
- JK (most versatile)

JK		<table border="1"> <thead> <tr> <th>J</th> <th>K</th> <th>Q(next)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>Q'</td> </tr> </tbody> </table>	J	K	Q(next)	0	0	Q	0	1	0	1	0	1	1	1	Q'	$Q(next) = JQ' + K'Q$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q(next)</th> <th>J</th> <th>K</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>X</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>X</td> </tr> <tr> <td>1</td> <td>0</td> <td>X</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>X</td> <td>0</td> </tr> </tbody> </table>	Q	Q(next)	J	K	0	0	0	X	0	1	1	X	1	0	X	1	1	1	X	0
J	K	Q(next)																																					
0	0	Q																																					
0	1	0																																					
1	0	1																																					
1	1	Q'																																					
Q	Q(next)	J	K																																				
0	0	0	X																																				
0	1	1	X																																				
1	0	X	1																																				
1	1	X	0																																				

- D (Data)

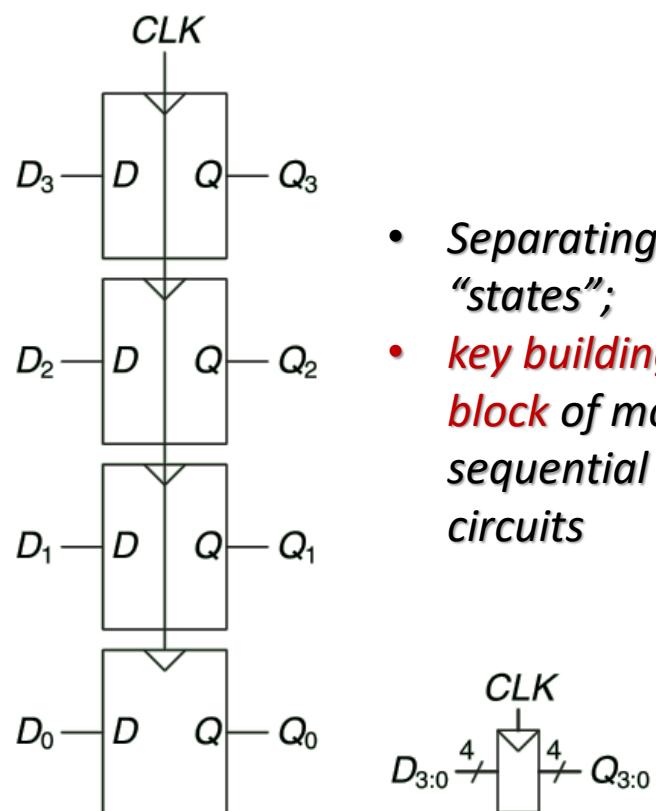
D		<table border="1"> <thead> <tr> <th>D</th> <th>Q(next)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> </tbody> </table>	D	Q(next)	0	0	1	1	$Q(next) = D$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q(next)</th> <th>D</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Q	Q(next)	D	0	0	0	0	1	1	1	0	0	1	1	1
D	Q(next)																								
0	0																								
1	1																								
Q	Q(next)	D																							
0	0	0																							
0	1	1																							
1	0	0																							
1	1	1																							

- T (Triggered/Toggle)

T		<table border="1"> <thead> <tr> <th>T</th> <th>Q(next)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Q</td> </tr> <tr> <td>1</td> <td>Q'</td> </tr> </tbody> </table>	T	Q(next)	0	Q	1	Q'	$Q(next) = TQ' + T'Q$	<table border="1"> <thead> <tr> <th>Q</th> <th>Q(next)</th> <th>T</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	Q	Q(next)	T	0	0	0	0	1	1	1	0	1	1	1	0
T	Q(next)																								
0	Q																								
1	Q'																								
Q	Q(next)	T																							
0	0	0																							
0	1	1																							
1	0	1																							
1	1	0																							

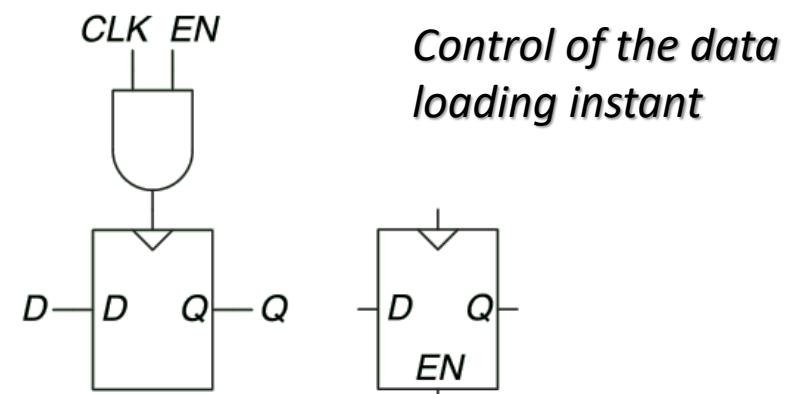
# Flip-flop derivations

- Register 寄存器

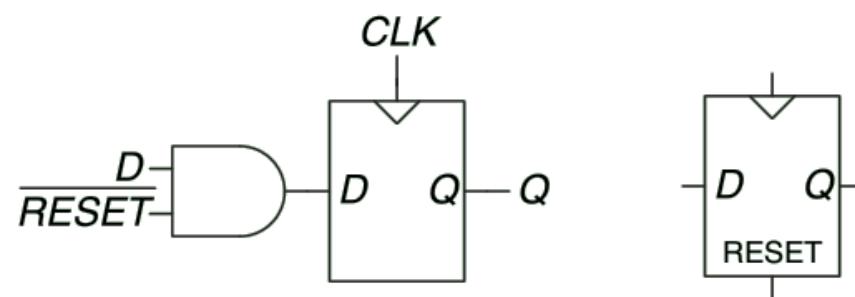


- Separating the “states”;
- key building block of most sequential circuits

- Enabled flip-flop



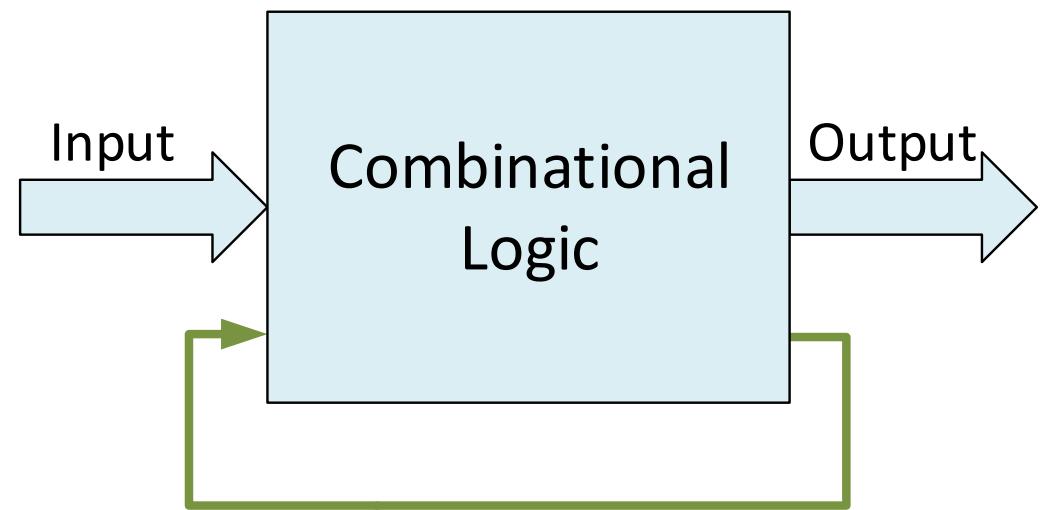
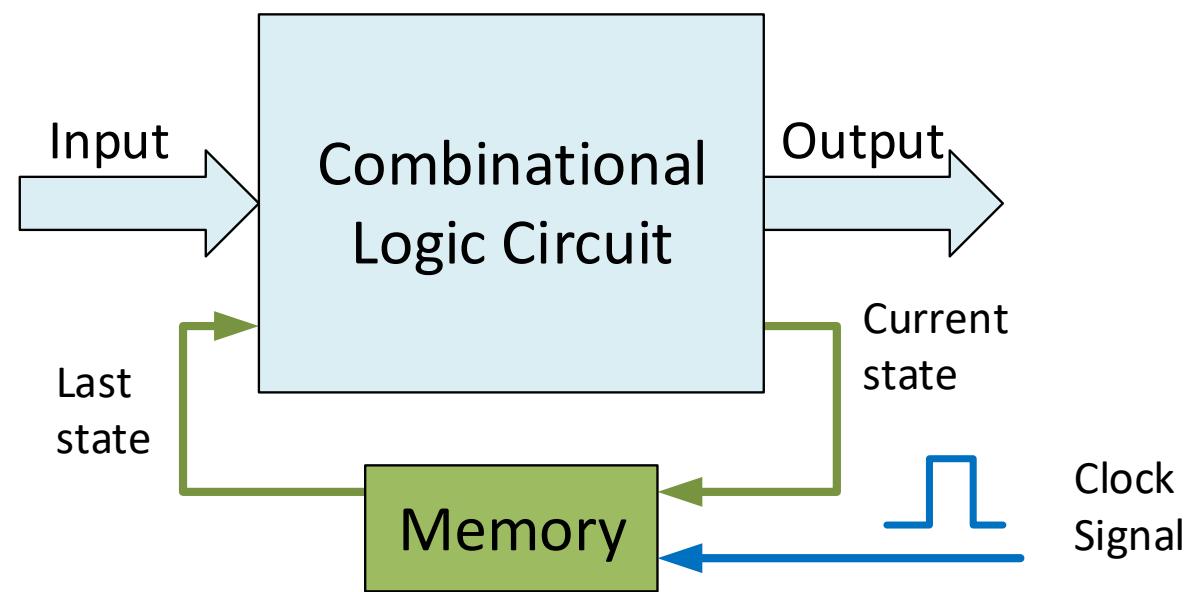
- Resettable flip-flop



*Block the effect of input D (reset=0)*

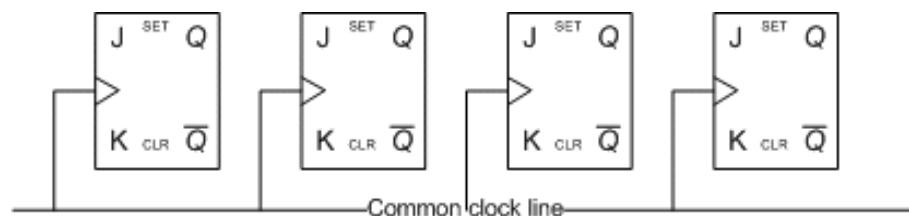
# Synchronous and asynchronous circuits

- Synchronous circuit
- Asynchronous circuit or self-timed circuit



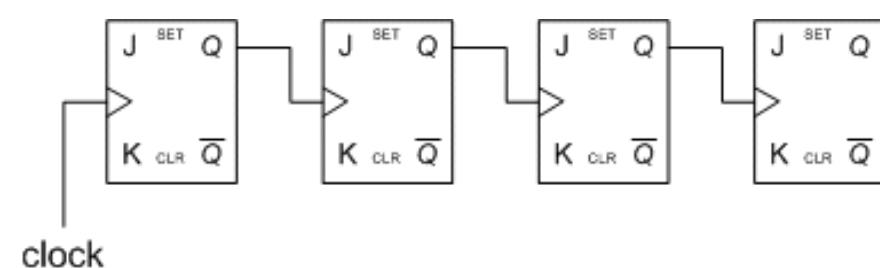
# Synchronous and asynchronous circuits

- Synchronous circuit



- Common **clock signal**
- Output only change at the **edge** of clock pulse
- clock signal should be long enough so that the **critical path** can settle before next clock edge
- Easy design

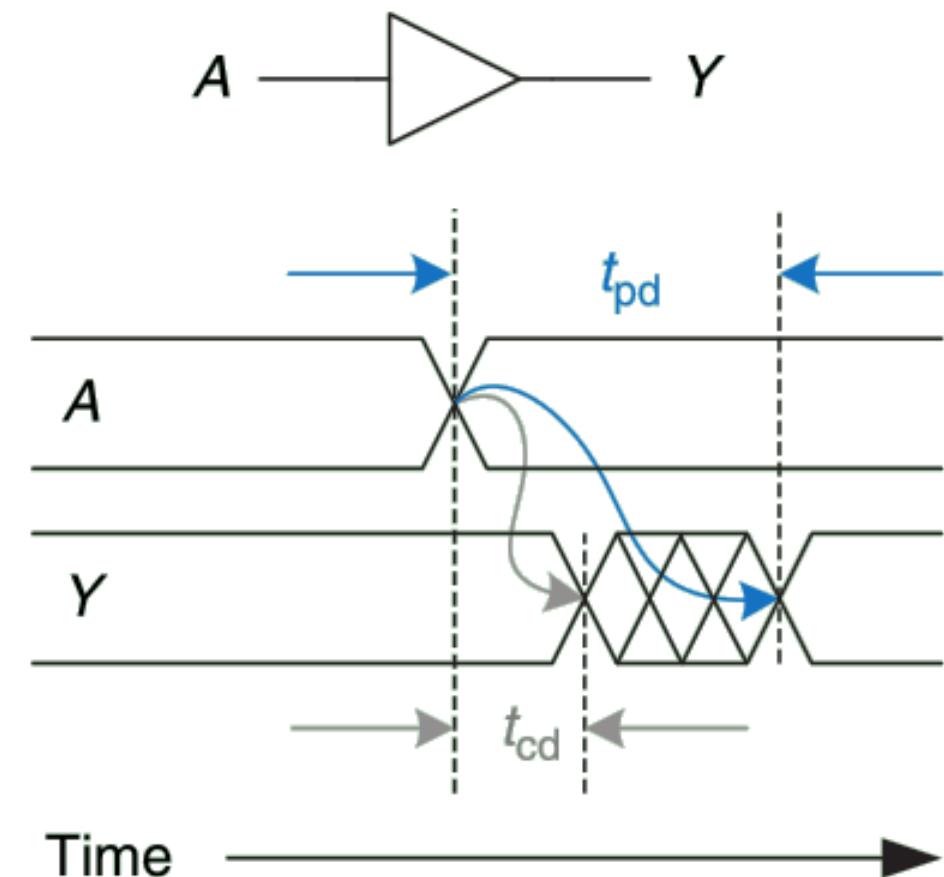
- Asynchronous circuit or self-timed circuit



- Not governed by global clock
- Resulting state can be sensitive to the relative arrival times of inputs at gates, the **race condition**

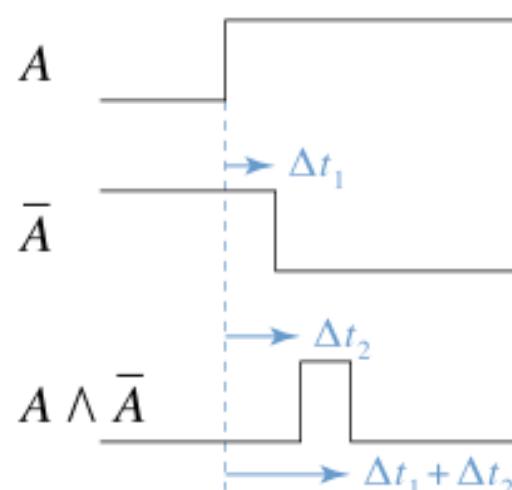
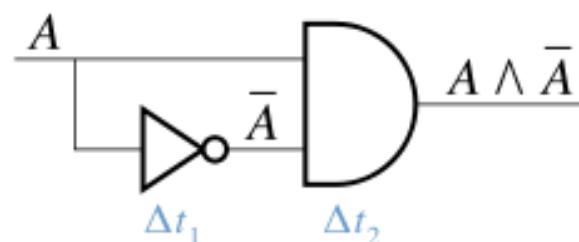
## Review: delay in combinational circuit

- contamination delay  $t_{cd}$ 
  - $Y$  (output) starts to change after the change of  $A$  (input)
- propagation delay  $t_{pd}$ 
  - $Y$  (output) definitely settles in new value



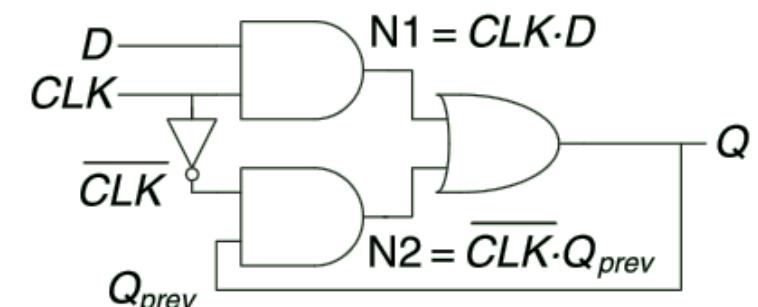
# Racing condition 竞争冒险

- In combinational circuit

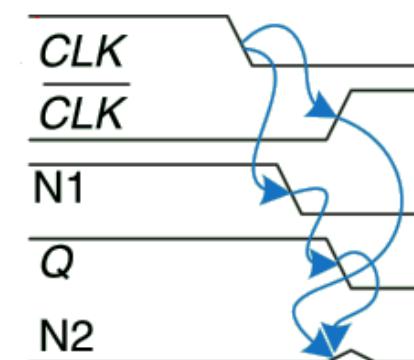


- In asynchronous sequential circuit

$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$



- $D = 1$   
 $CLK = 1; \overline{CLK} = 0$   
 $Q_{prev} = 1 * 1 + 0 * X = 1$
- $CLK = 1 \rightarrow 0; \overline{CLK} = 1$   
 $Q = 1 * 0 + 1 * 1 = 1$
- Suppose the delay through the inverter from  $CLK$  to  $\overline{CLK}$  is rather long compared to the delays of the AND and OR gates
- eventually  $Q=0$  because of the race condition**



## Rules of synchronous sequential circuit

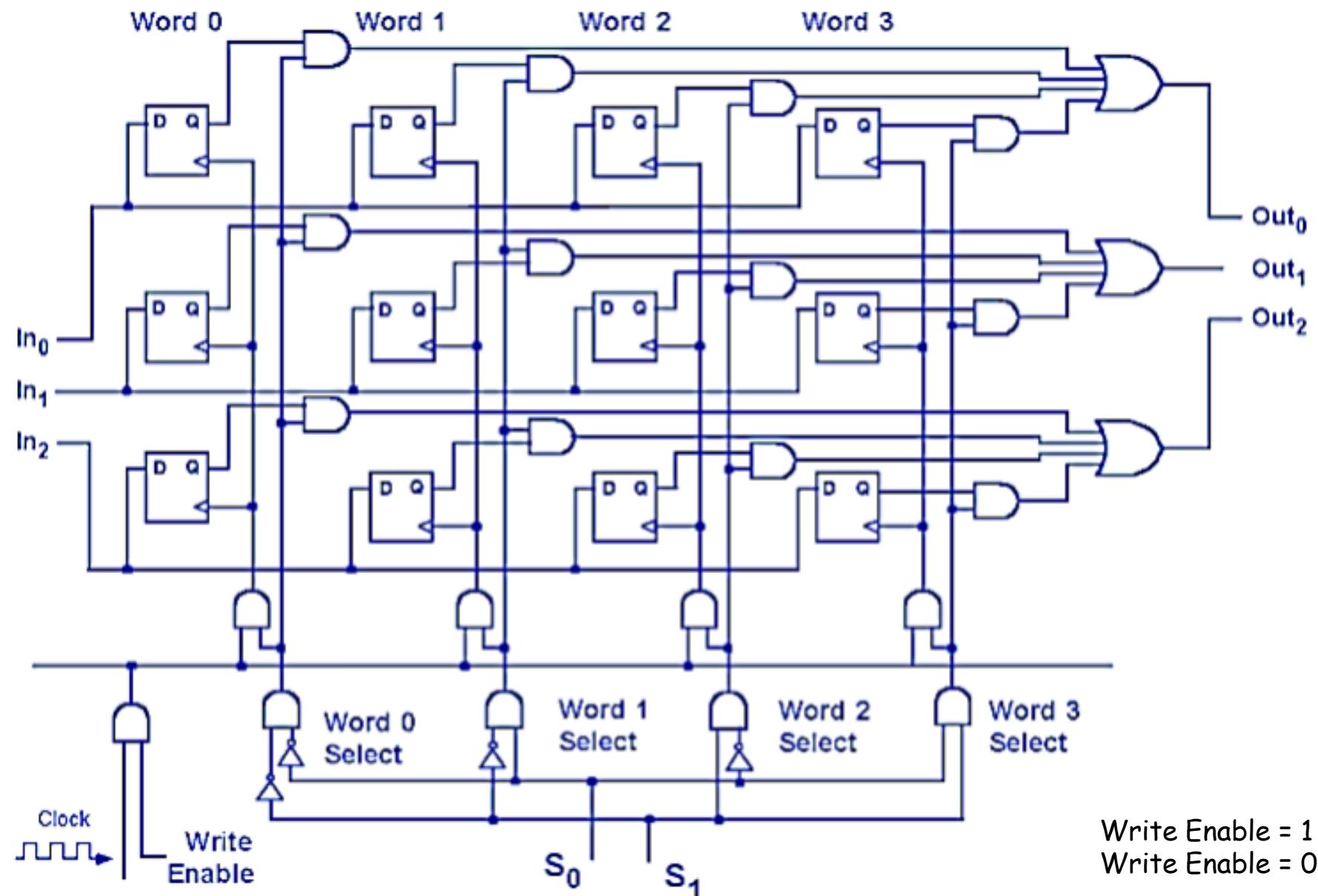
- Every circuit element is either a register or a **combinational circuit**
- At least one circuit element is a **register**
- All registers receive the **same clock signal**
- Every cyclic path contains **at least one register**

## Example: 4 × 3 memory

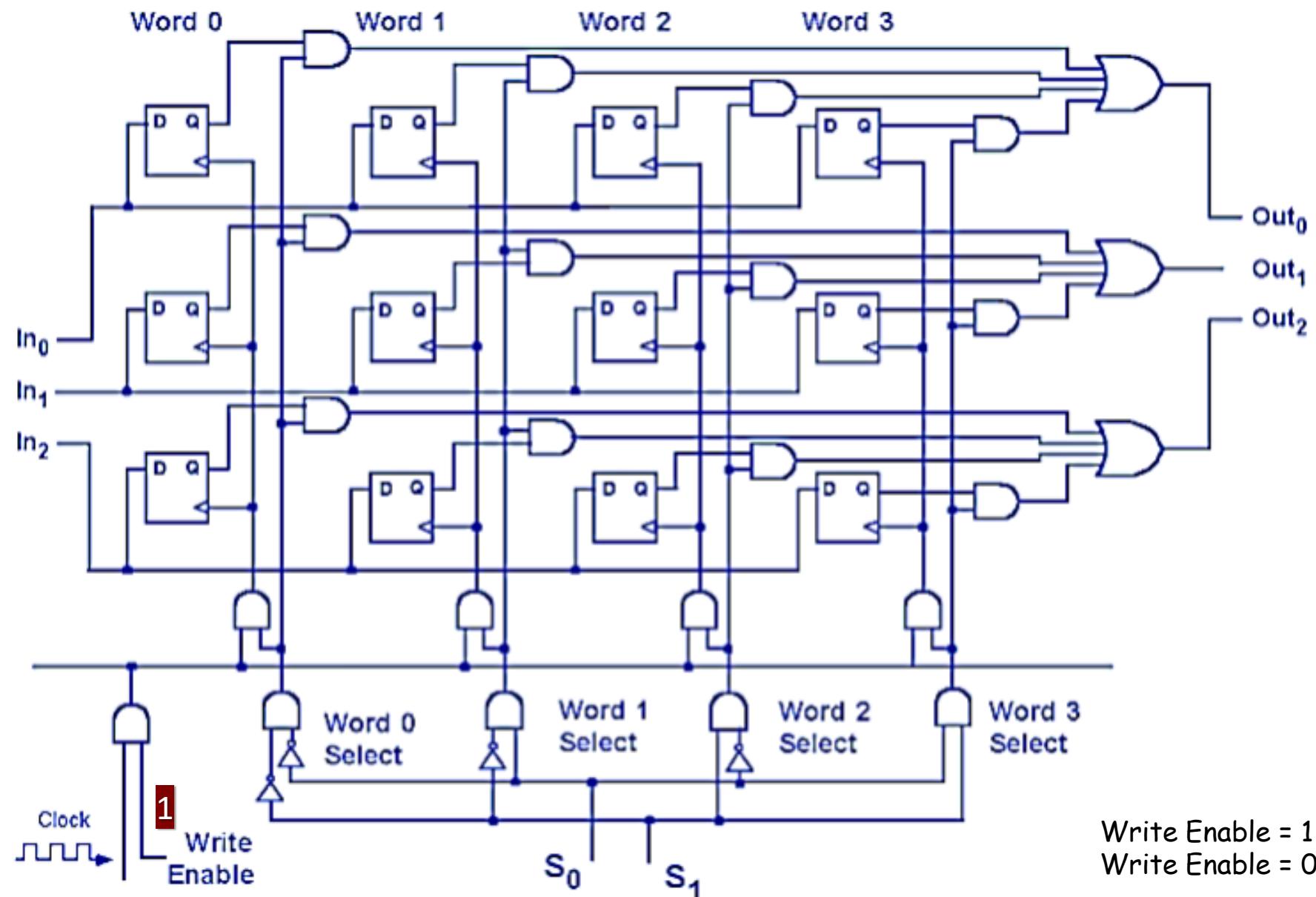
- 4 words and each can store 3 bits of information
- To represent 4 words need 2-bits for address
  - Address decoder performs the address decoding
- To store information we use D Flip-Flop for each bit (total 12, as each location as 3 bits and we have 4 total locations)
- Need select variable to chose if we want to read or write data and that is combined with clock signal (using some combinational logic)
- Also need some other combinational logic...



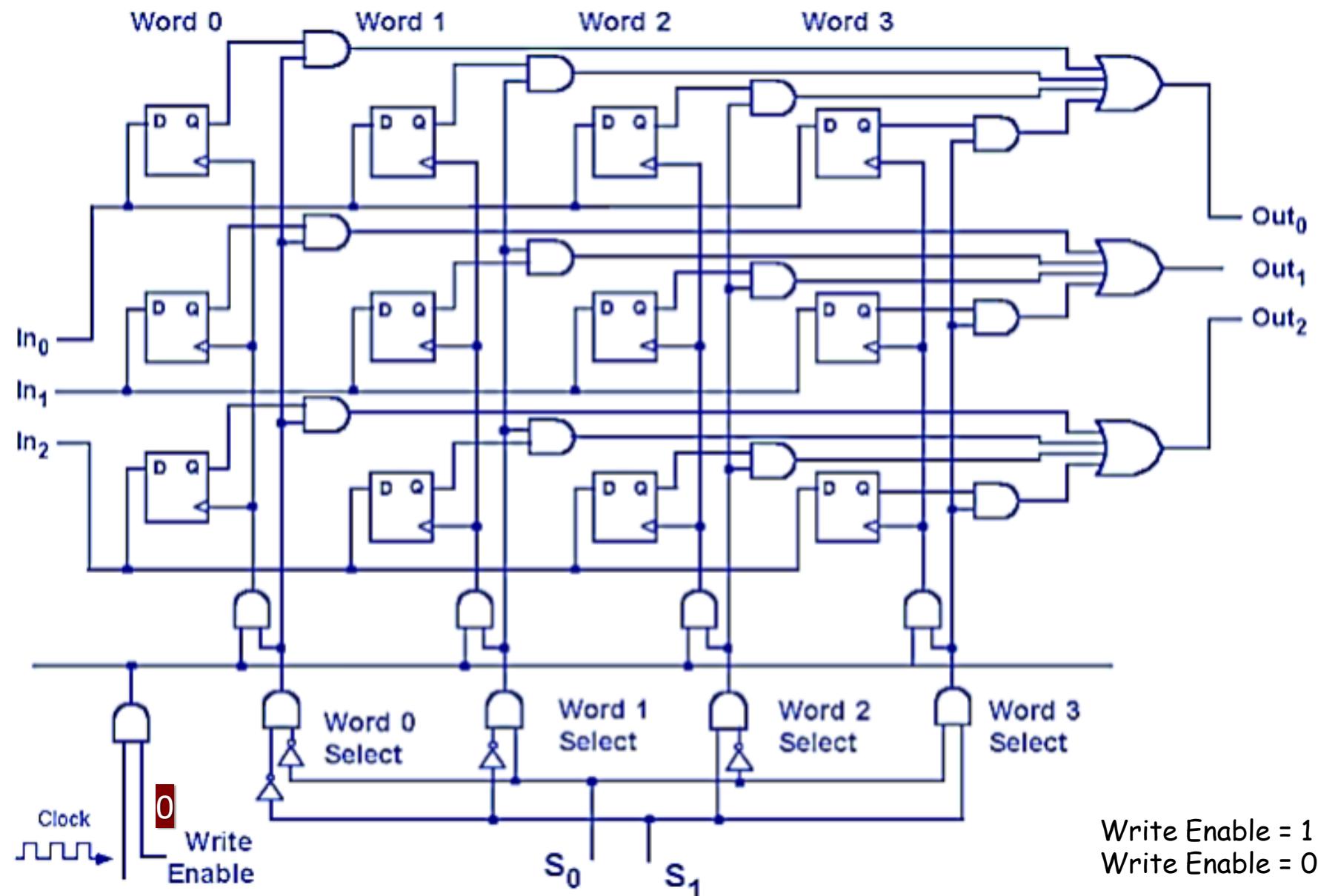
# Example: 4 × 3 memory



# Write operation to word 1



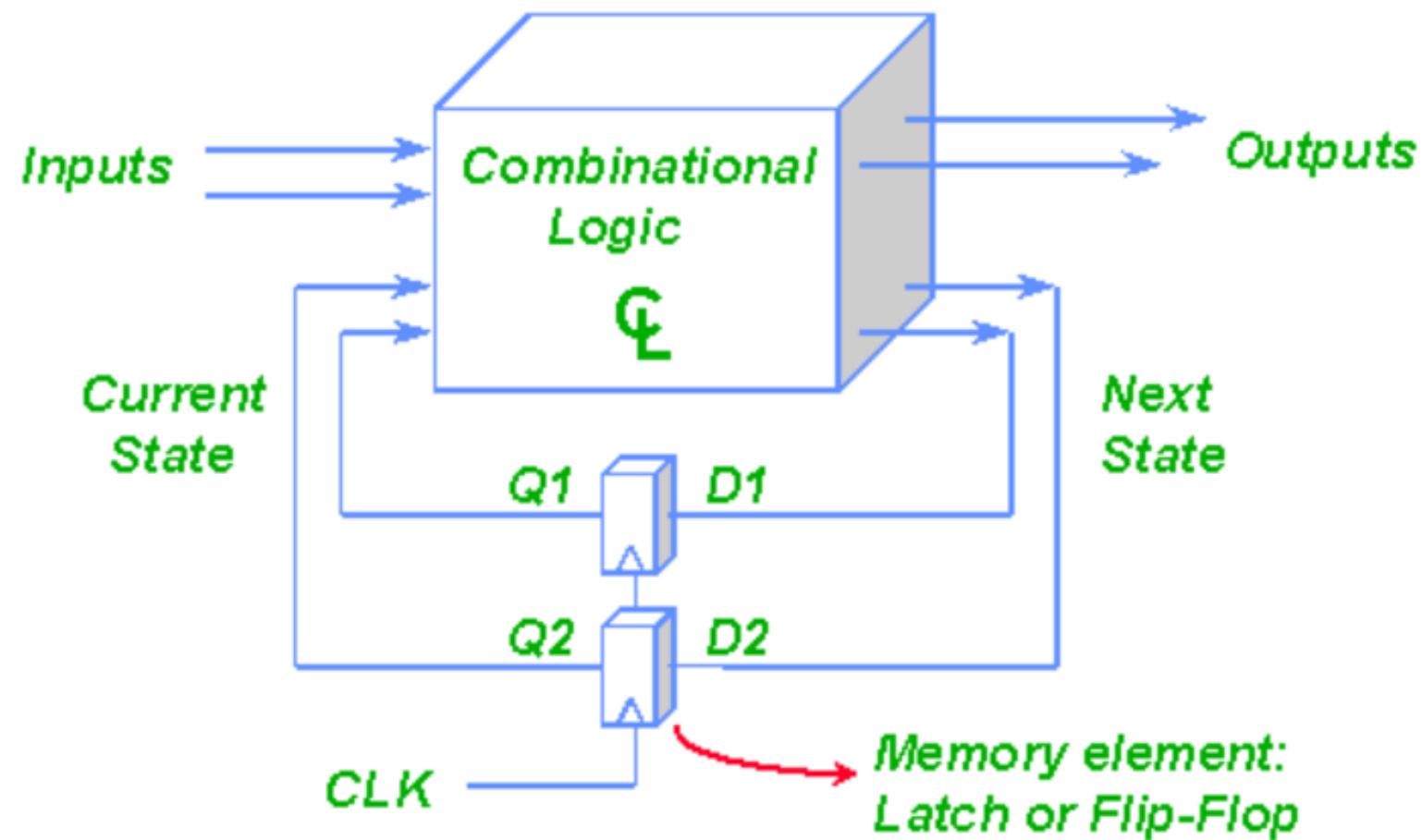
# Read operation from word 3



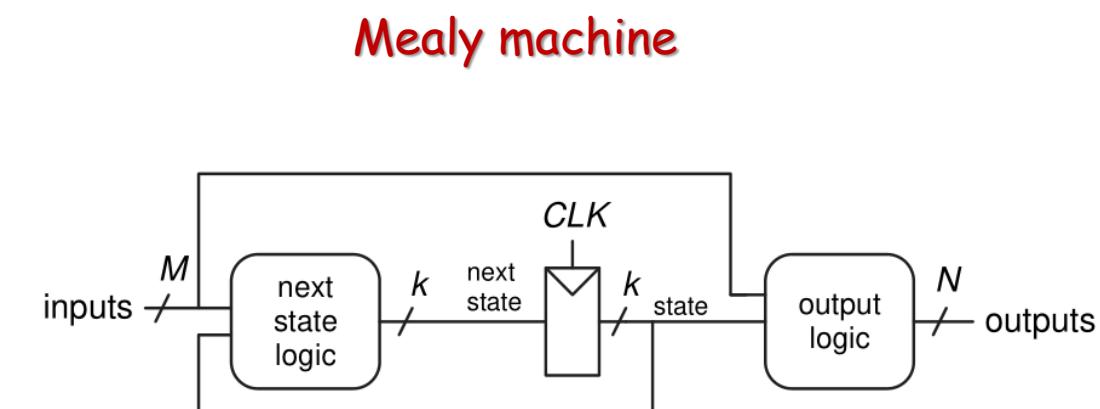
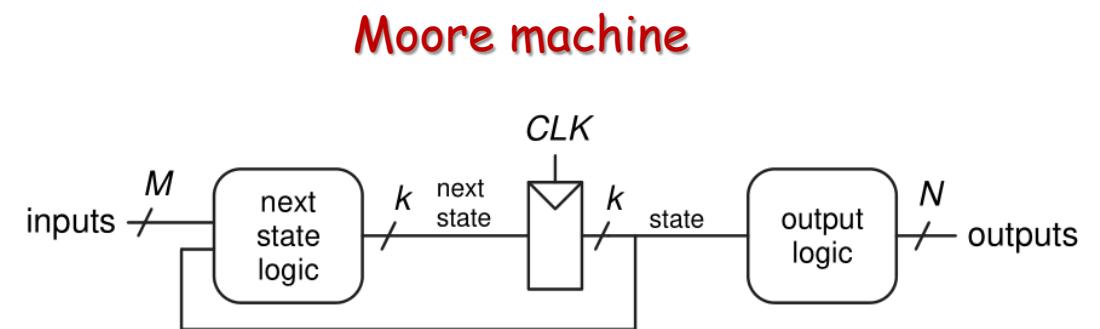
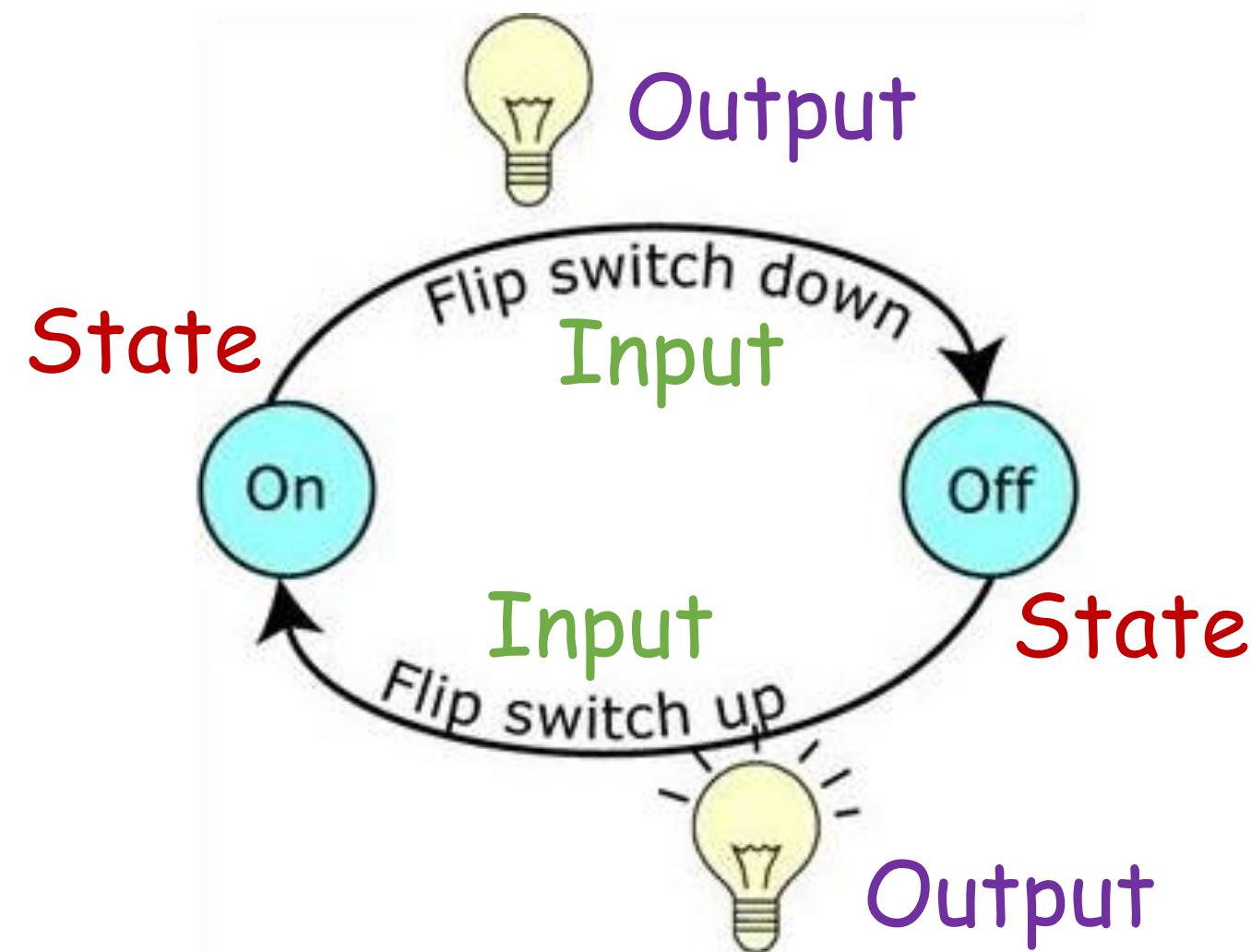
Write Enable = 1 allows memory write  
Write Enable = 0 allows memory read

## Finite-state machine

- FSM -- a **mathematical model of computation used to design both computer programs and sequential logic circuits**

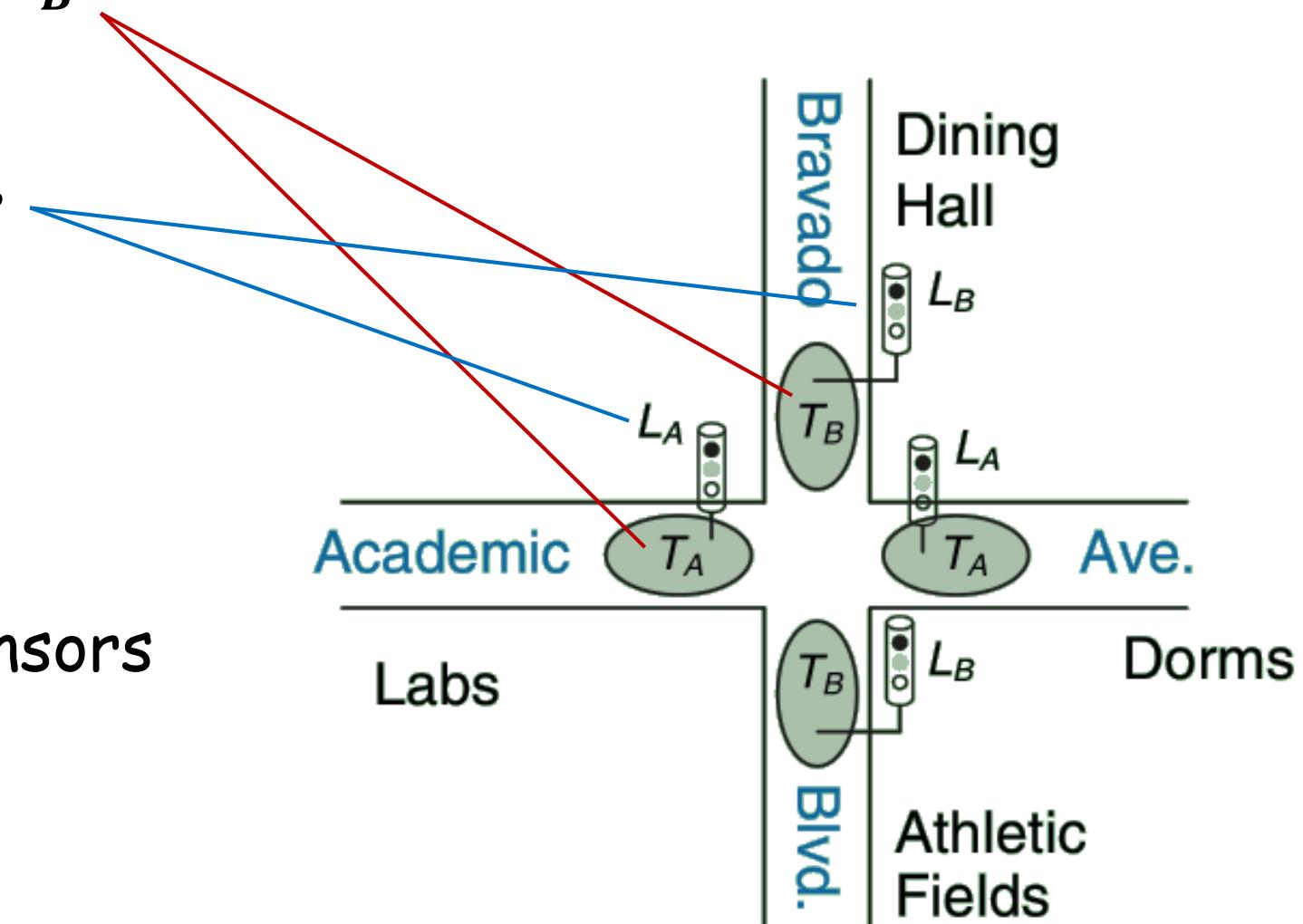


# The most seen FSM

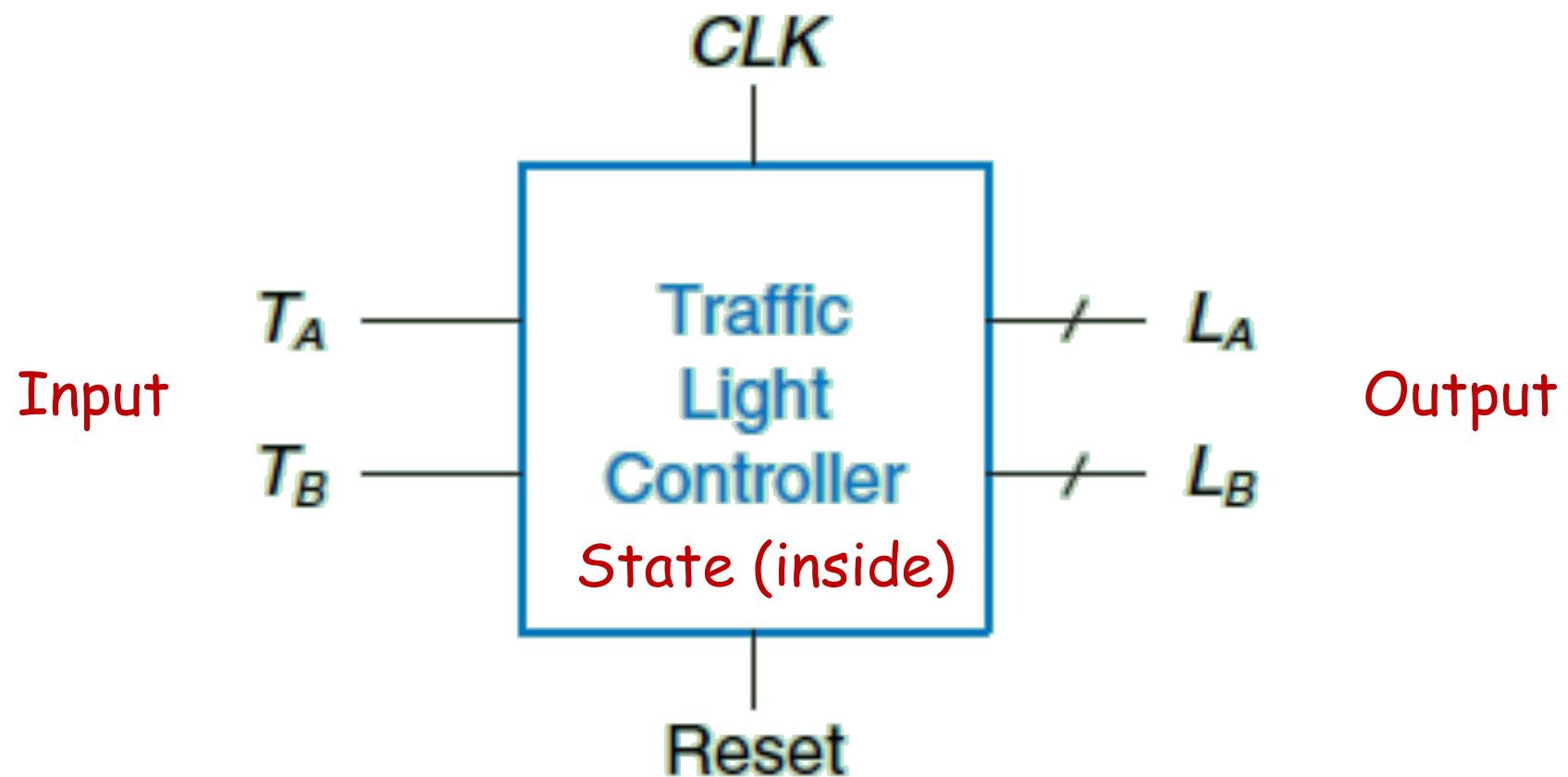


## Example: traffic light controller

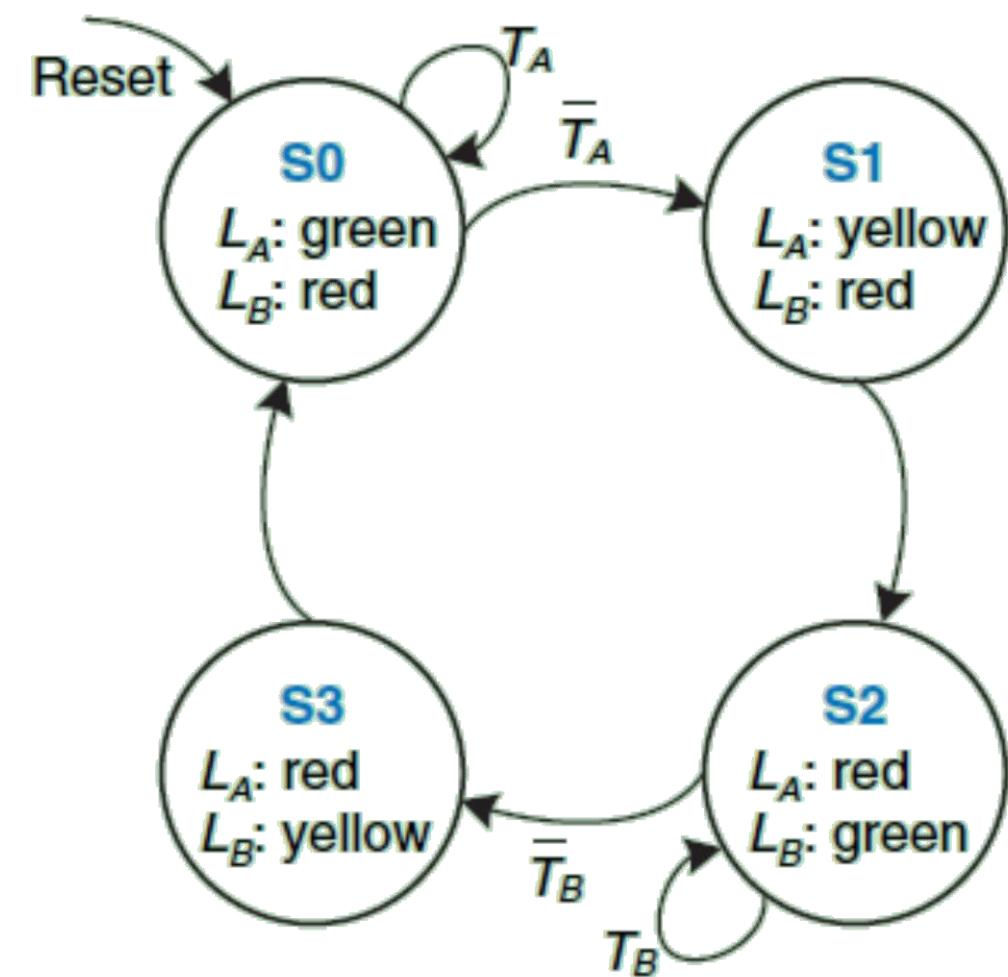
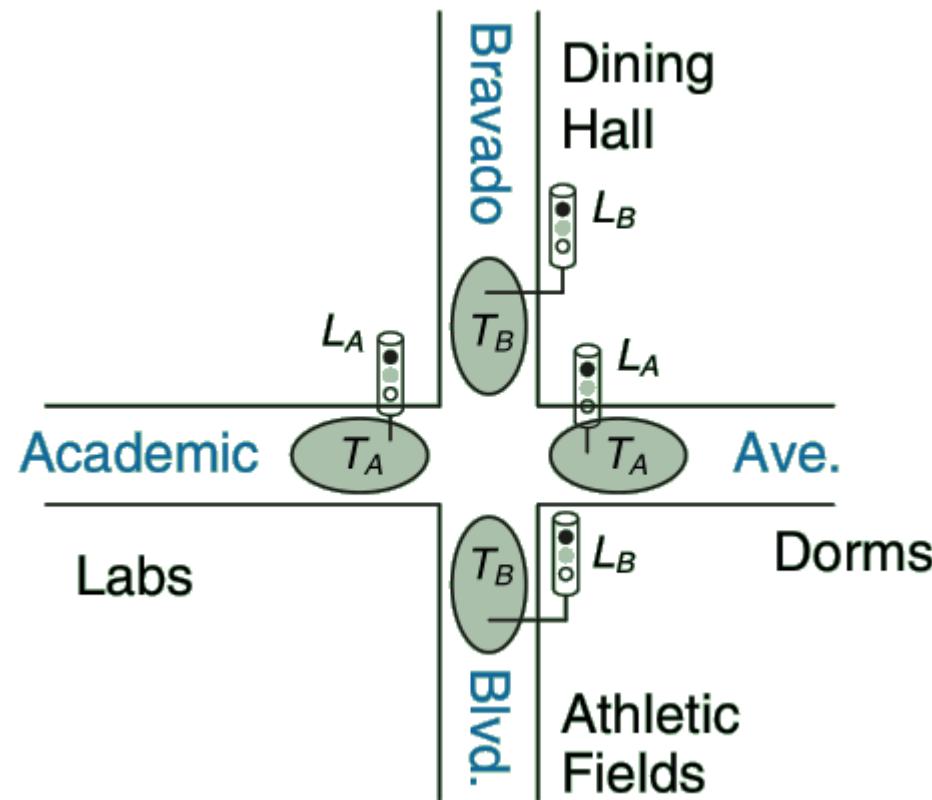
- Traffic sensors  $T_A$  and  $T_B$  ("1" busy, "0" empty)
- Traffic lights  $L_A$  and  $L_B$  (each light have red, yellow, and green)
- 5-second clock, at each rising edge, lights may change based on the sensors
- Reset button



## Black box view



# State transition diagram



# State table

State	Encoding $S_{1:0}$
S0	00
S1	01
S2	10
S3	11

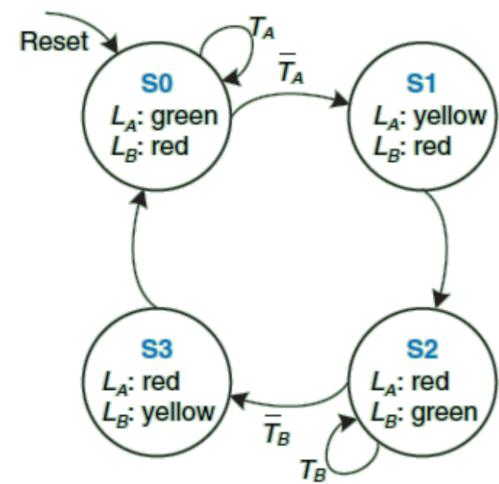
State encoding

State transition table

Current State $S$	Inputs $T_A$	Inputs $T_B$	Next State $S'$
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

Binary encoded state transition table

Current State $S_1$	Current State $S_0$	Inputs $T_A$	Inputs $T_B$	Next State $S'_1$	Next State $S'_0$
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0



# Output table

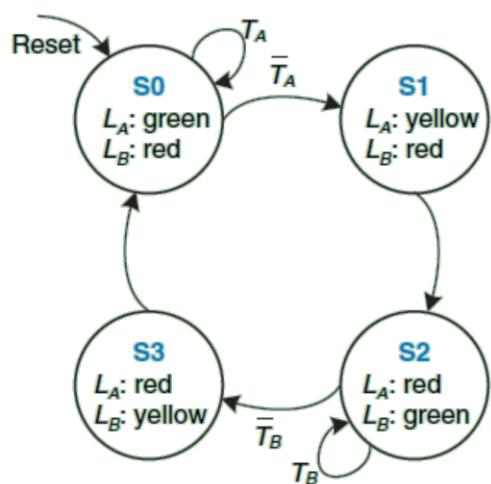
Output encoding

Output	Encoding $L_{1:0}$
green	00
yellow	01
red	10



Output table

Current State		Outputs			
$S_1$	$S_0$	$L_{A1}$	$L_{A0}$	$L_{B1}$	$L_{B0}$
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1



# Sum-of-products form

State table

Current State		Inputs		Next State	
$S_1$	$S_0$	$T_A$	$T_B$	$S'_1$	$S'_0$
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

Output table

Current State		Outputs			
$S_1$	$S_0$	$L_{A1}$	$L_{A0}$	$L_{B1}$	$L_{B0}$
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

$$\begin{aligned}
 S'_1 &= \bar{S}_1 S_0 + S_1 \bar{S}_0 \bar{T}_B + S_1 \bar{S}_0 T_B \\
 &= \bar{S}_1 S_0 + S_1 \bar{S}_0 \\
 &= S_1 \oplus S_0
 \end{aligned}$$

$$S'_0 = \bar{S}_1 \bar{S}_0 \bar{T}_A + S_1 \bar{S}_0 \bar{T}_B$$

$$L_{A1} = S_1$$

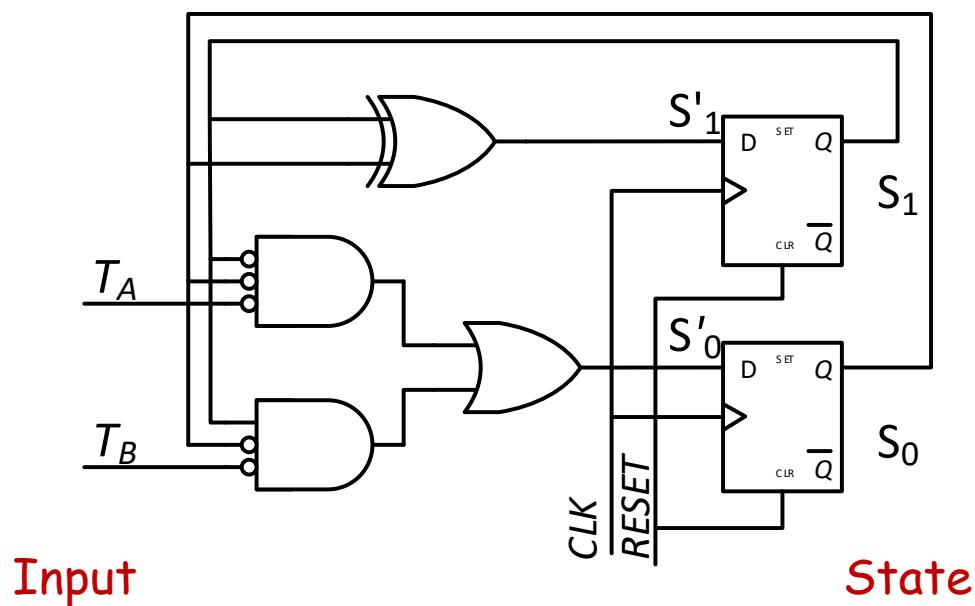
$$L_{A0} = \bar{S}_1 S_0$$

$$L_{B1} = \bar{S}_1$$

$$L_{B0} = S_1 S_0$$

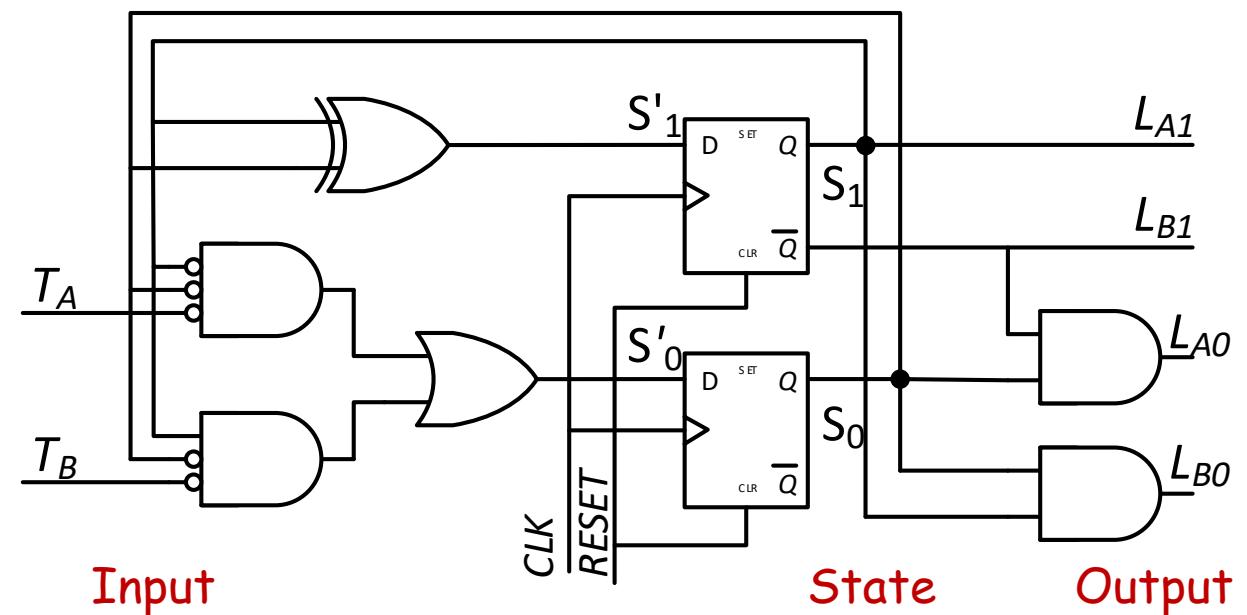
## State & output logic

- State logic (sequential)
  - Output Logic (combinational)



$$S'_1 = S_1 \oplus S_0$$

$$S'_0 = \bar{S}_1 \bar{S}_0 \bar{T}_A + S_1 \bar{S}_0 \bar{T}_B$$

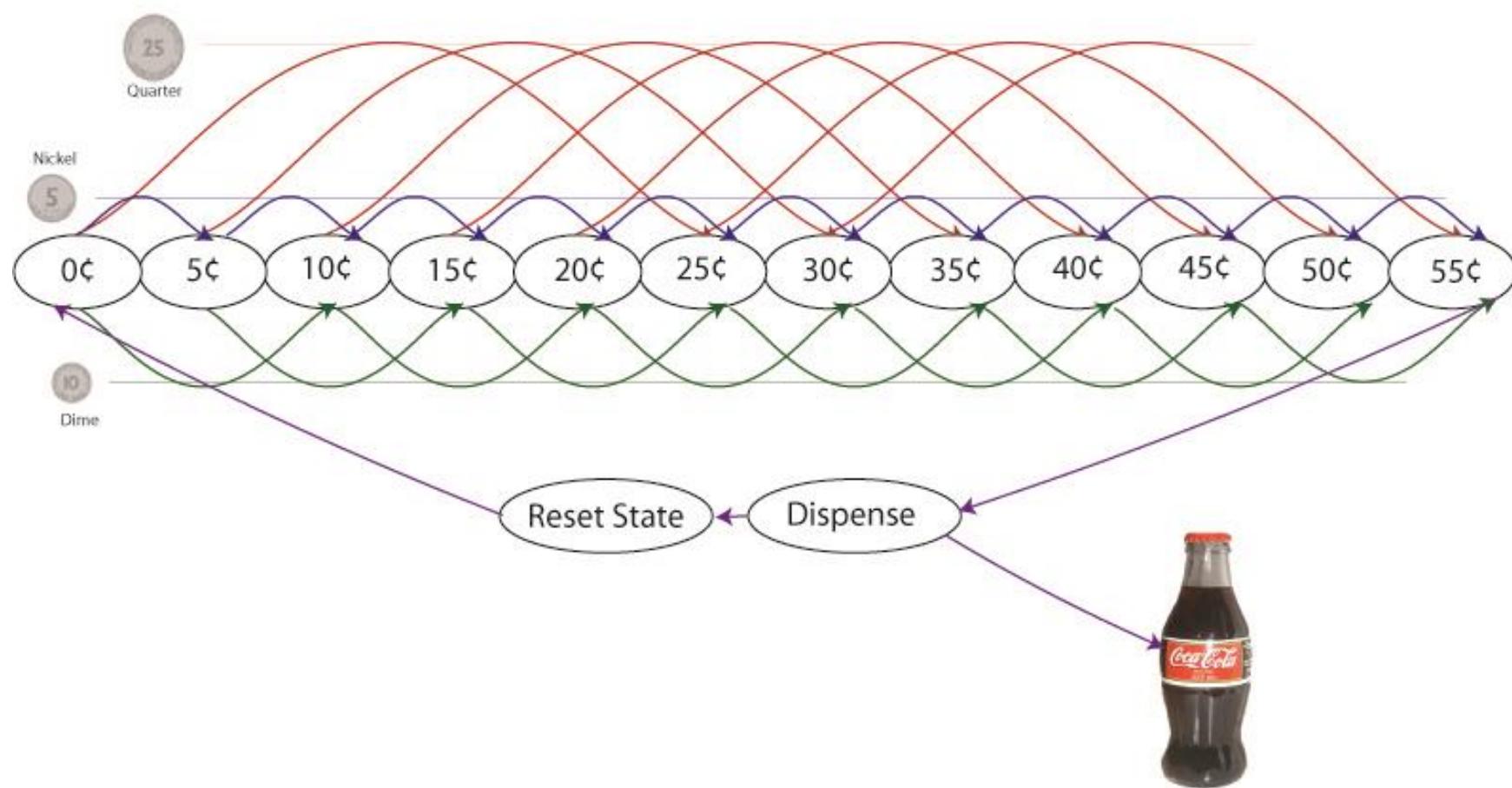


$$L_{A1} = S_1; \quad L_{B1} = \bar{S}_1$$

$$L_{A0} = \bar{S}_1 S_0; \quad L_{B0} = S_1 S_0$$

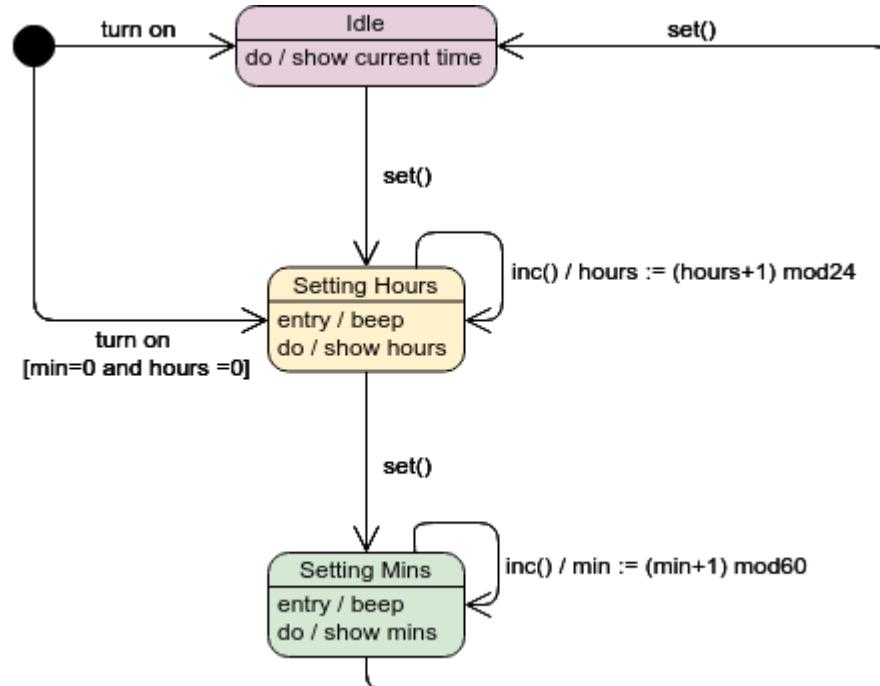
# FSM with more states .....

Finite State Machine:  
Soda Machine State Diagram



# FSM with more states .....

- Digital clock



- Microwave oven

