CSE101 Circuit

Author: ICS Strategy Group

1. Basic logic gates and their truth tables

Inputs A B	C A AND B
0 0	0
0 1	0
1 0	0
1 1	1

Inputs A B	A OR B
0 0	0
0 1	1
1 0	1
1 1	1

Inputs A B	A XOR B
0 0	0
0 1	1
1 0	1
1 1	0

Inputs A	NOT A
0	1
1	0

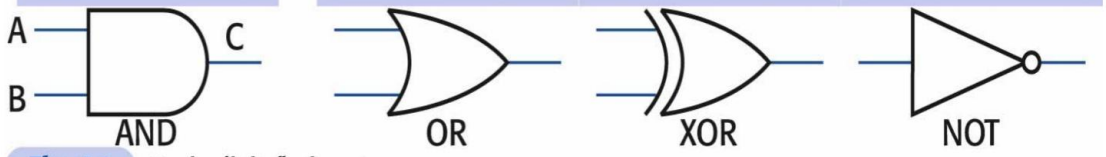


Fig. 4.2 Basic digitallogic gates.

可以把 XOR 理解为不等号方便记忆

2. Alternative notations(替代符号)

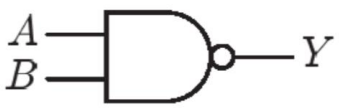
and: ∧, &

or: ∨

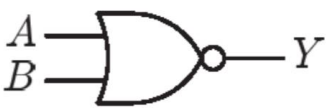
xor: |

not: ¬, -

3. N~ gate example

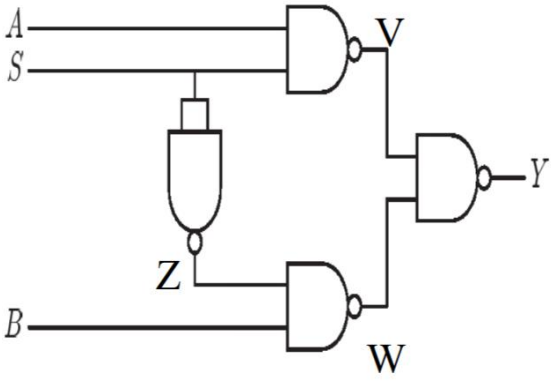


NAND gate, $Y =$ **not** $(A$ **and** $B)$



NOR gate, $Y =$ **not** $(A$ **or** $B)$

4. Selector circuit



- This circuit implements the function:
  - $Y =$ **not** $(V$ **and** $W)$,
    where,
  - $V =$ **not** $(A$ **and** $S)$,
  - $W =$ **not** $(Z$ **and** $B)$,
  - $Z =$ **not** $(S$ **and** $S)$
    $=$ **not** S.

- Combining altogether we get:
  $Y =$ **not** $($**not** $(A$ **and** $S)$ **and not** $($**not** (S) **and** $B))$.

A B S	A and S	V	Z	Z and B	W	V and W	Y
0 0 0	0	1	1	0	1	1	0
0 0 1	0	1	0	0	1	1	0
0 1 0	0	1	1	1	0	0	1
0 1 1	0	1	0	0	1	1	0
1 0 0	0	1	1	0	1	1	0
1 0 1	1	0	0	0	1	0	1
1 1 0	0	1	1	1	0	0	1
1 1 1	1	0	0	0	1	0	1

- One may give a short definition of the function from the truth table:

If S = 1 then Y = A

if S = 0 then Y = B

Or **Y = (S&A)V(-S&B)**

5. Data selector

Control W X Y Z	Data A B C D	OUT
0 0 0 1	a b c d	a
0 0 1 0	a b c d	b
0 1 0 0	a b c d	c
1 0 0 0	a b c d	d

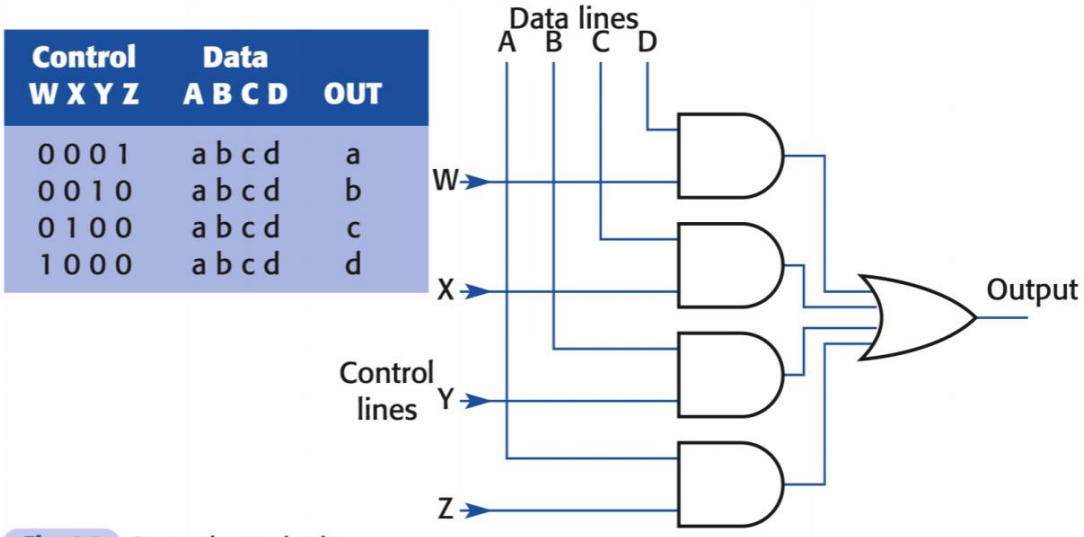


Fig. 4.5 Data selector circuit.

6. Two-line decoder

(Very Important, Lecture 23, P7~P14)

Two-line decoder

Selector Y X	Line d c b a
0 0	0 0 0 1
0 1	0 0 1 0
1 0	0 1 0 0
1 1	1 0 0 0

Fig. 4.6 A two-line decoder.

Data selector with two-line decoder

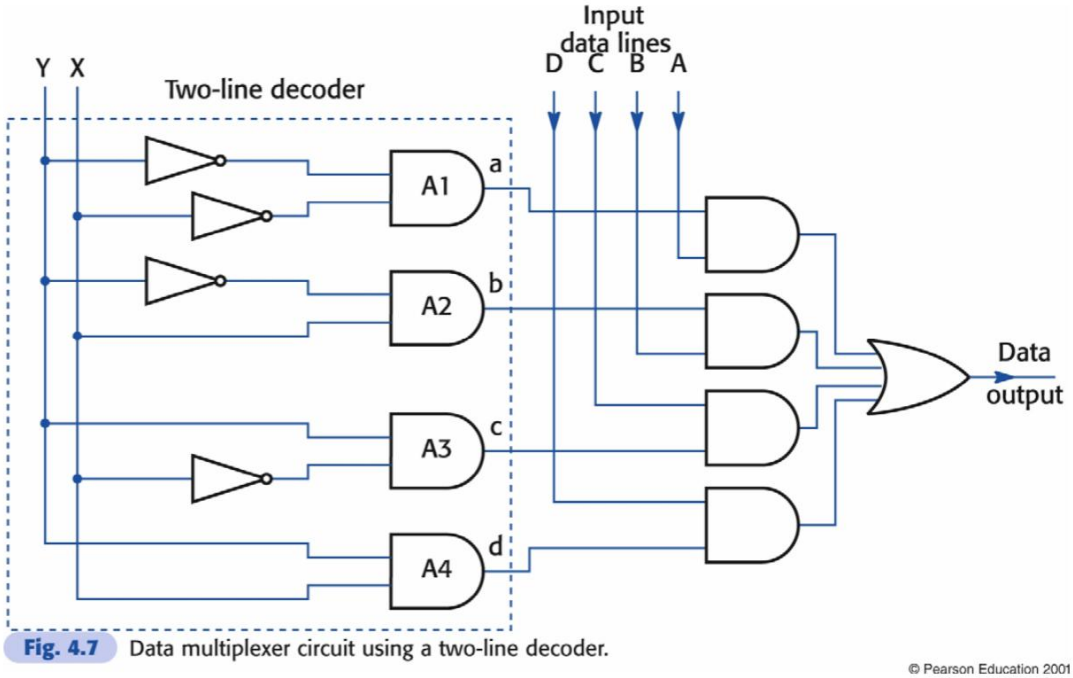


Fig. 4.7 Data multiplexer circuit using a two-line decoder.

7. How to design a circuit from a truth table

(Very Important)

Truth Table -> Short form -> Logic Expression -> Implementation

Example:

Truth Table:

Truth table

i_1	i_2	i_3	i_4	O_1
1	1	0	0	1
1	1	0	1	1
0	0	1	0	1
1	0	1	0	1
0	1	0	1	1
.....				0
	...			0
.....				0

Step 1: Get the short form (* means 0 or 1)

Short form

i_1	i_2	i_3	i_4	O_1
1	1	0	*	1
*	0	1	0	1
0	1	0	1	1

Step 2: Get the logic expression

$O_1 =$ $(i_1$ **and** i_2 **and** $(\textbf{not}\ i_3))$

or

$((\textbf{not}\ i_2)$ **and** i_3 **and** $(\textbf{not}\ i_4))$

or

$((\textbf{not}\ i_1)$ **and** i_2 **and** $(\textbf{not}\ i_3)$ **and** $i_4)$.

Step 3: Implement

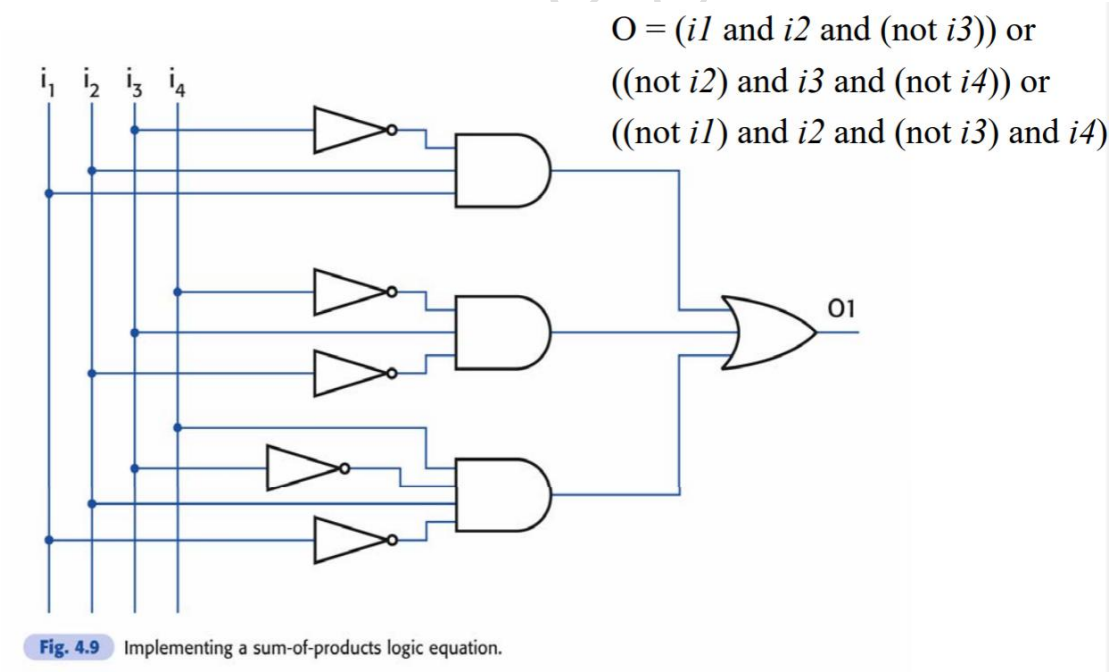
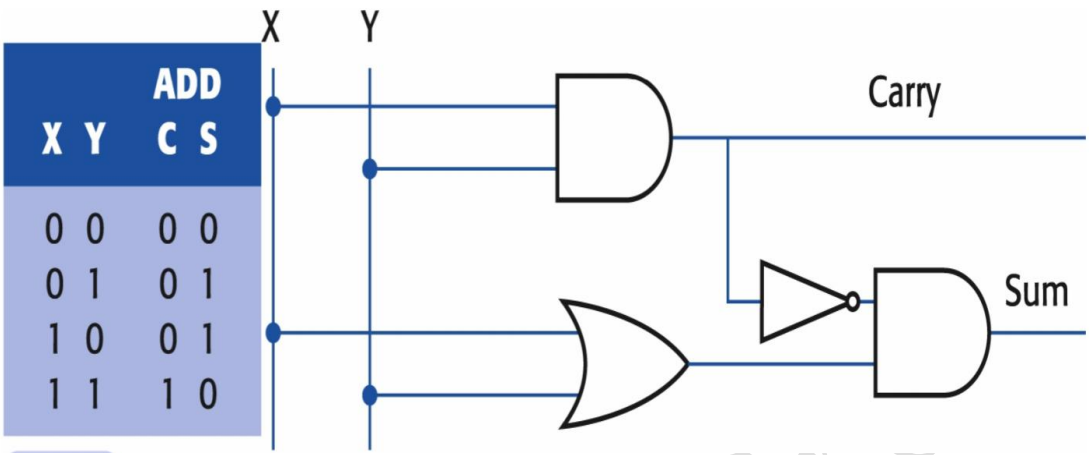


Fig. 4.9 Implementing a sum-of-products logic equation.

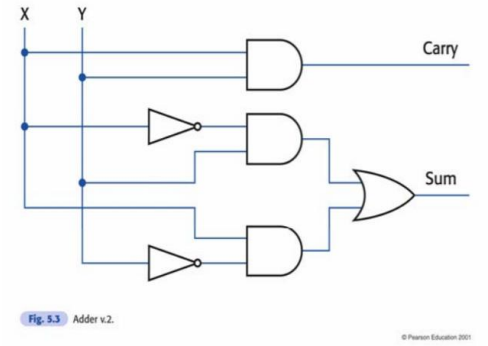
8. Half adder

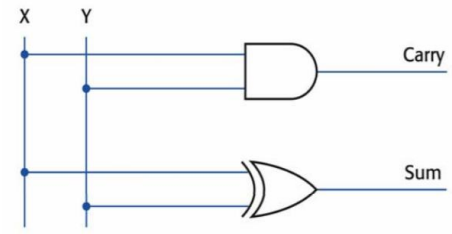
Carry means overflow, Sum means the result

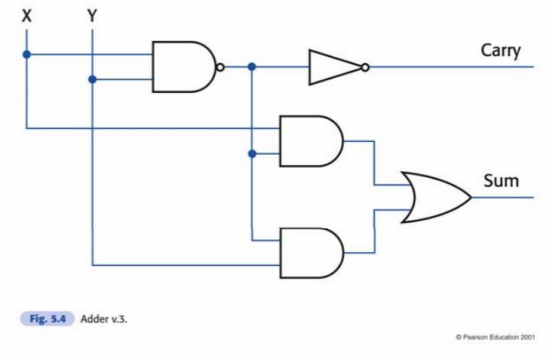
- Binary addition. For the addition of single-bit binary numbers: (Half-)adder.



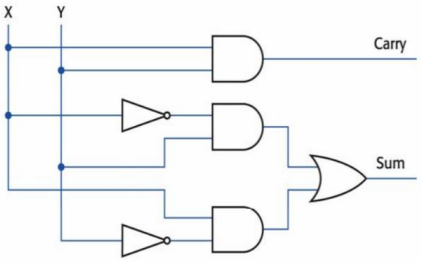
Other possible designs for half adder







个人比较推崇记住这种：



因为这样的话可以直接通过上面根据真值表画电

路的方法画出来，只需要把两个输出结果分开做就好了。

9. Full adder

Carry means overflow, Sum means the result

- For the addition of multi-bit binary numbers one needs to deal with carry-in from previous stage, that means the adder should have three inputs.

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

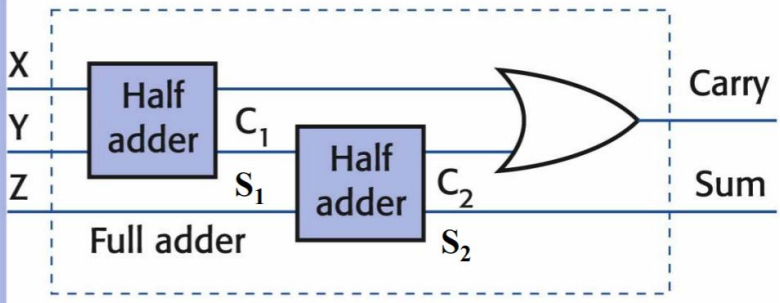


Fig. 5.6 Full adder.

因为二进制中三个 1 加一起，最多只可能为 11，即最多 overflow 一次，所以两次累加得到的 Carry 用 OR 连接输出。

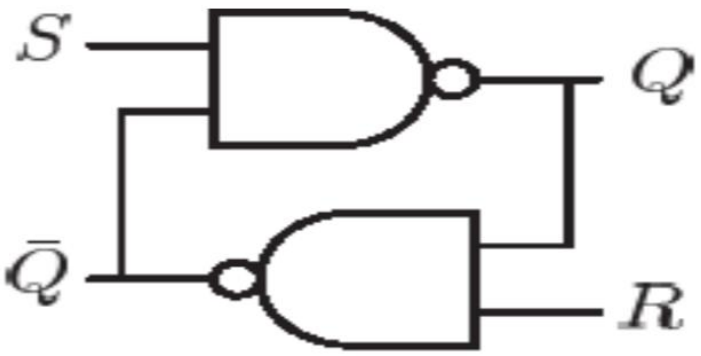
10. Sequential logic circuits

- **Combinational** (combinatorial) logic circuits.
  - In all Boolean circuits that we have seen so far, the output at any moment depends only on the input at the same moment.
  - **No memory** is there.
- **Sequential** logic circuits.
  - The output depends also on the **state** of the circuit. The state of the circuit is somehow stored within the circuit.
  - There is a **memory** inside.

重点在组合逻辑电路没有记忆区，但数列逻辑电路有记忆区间。

11. SR flip-flop

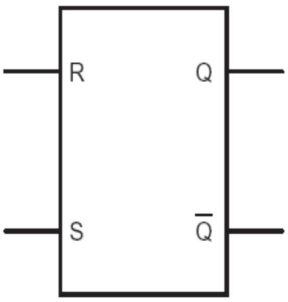
Has an illegal state.



R	S	Q
0	0	?
0	1	0
1	0	1
1	1	Q_{prev}

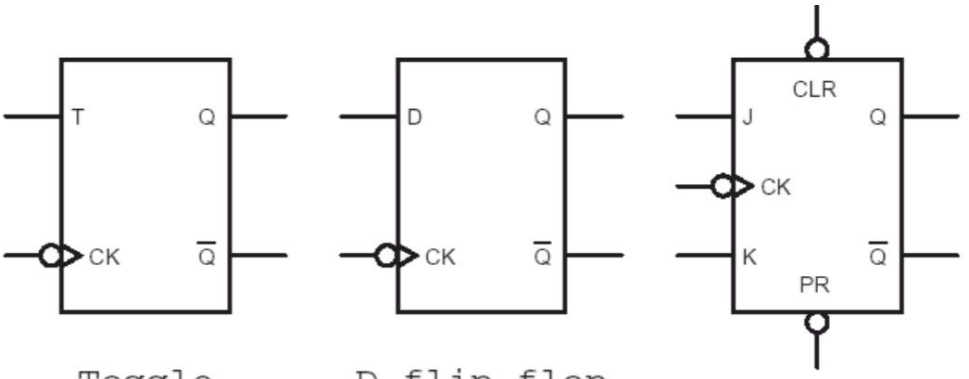
prev means previous
即：
当(R,S)从(0,1)变成(1,1)结果 Q 仍旧保持 0
当(R,S)从(1,0)变成(1,1)结果 Q 仍旧保持 1

SR representation



FlipFlop(SR)

12. Other types of flip-flops



Toggle D flip-flop J-K flip-flop

T	Q
0	Q_{prev}
1	$\bar{Q}_{prev}$

D	Q
0	0
1	1

J	K	Q
0	0	Q_{prev}
0	1	0
1	0	1
1	1	$\bar{Q}_{prev}$

- All these flip-flops has an input marked for **clock**. The new output occurs when the clock is pulsed, i.e. momentarily changed from 1 to 0.
- CLR (clear) and PR (preset) inputs are used to initialise the flip-flop to known value (0 and 1, respectively).

13. D flip-flops

Can copy data from one register to another.