Question

- Compiler:
  1. understand what the program say.（into intermedia）
  2. 优化，减少指令的数量
- 实现部分功能的microarchitecture算不算实现了ISA?
  yes你选择的这些功能就构成了你的微架构的ISA。

CH2

十六进制Hexadecimal Notation

- 2进制到16进制的转化：n digits $\rightarrow \lceil \frac{n}{4} \rceil$ digits.

  整数部分的转化：从低位开始连续四位$binary\ digits$变为一位$Hexadecimal\ digits$

  小数的转化：从小数点开始连续四位$binary\ digits$变为一位$Hexadecimal\ digits$

  $(3000.28)_h = (0011000000000000.00101)_b$

八进制Octal Notation

- 2进制到8进制的转化：$n\ digits \rightarrow \lceil \frac{n}{3} \rceil\ digits$

  整数部分的转化：从低位开始连续三位$binary\ digits$变为一位$Octal\ digits$

  小数的转化：从小数点开始连续四位$binary\ digits$变为一位$Octal\ digits$

  $(53.54)_o = (101011.1011)_b$

十进制Decimal Notation

- X进制转化成10进制

  > $b_n b_{n-1} \ldots b_0 = b_n * X^n + b_{n-1} * X^{n-1} + \ldots + b_0 * X^0$

- 10进制转化成X进制

  > 短除法

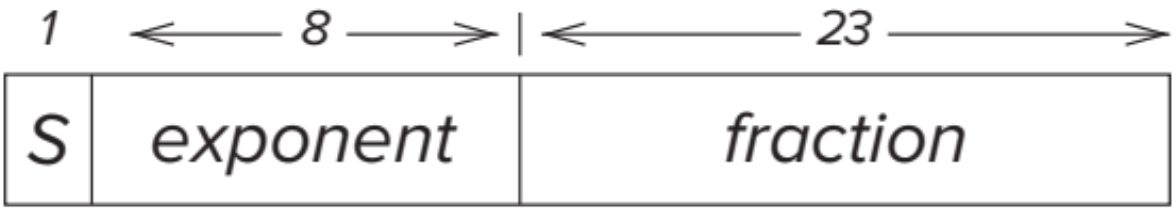
德摩根定律Demorgan's Law

当我们没有某些门时，需要转化。

$A + B = \overline{\overline{A} \cdot \overline{B}}$

$AB = \overline{\overline{A} + \overline{B}}$

浮点数表示IEEE754



The 32-bit floating point data type.

- **Greater Range, Less Precision**

  > 在**相同长度**的情况下，更高的精度往往是以更小的范围作为交换的。
  >
  > 因为$fraction$越大，$exponent$就越小

- **IEEE754**

  > | number of bits | S | Exponent | Fraction |
  > | --- | --- | --- | --- |
  > | 16 | 1 | 5 | 10 |
  > | **32** | **1** | **8** | **23** |
  > | 64 | 1 | 11 | 52 |

- **Normalized Form（32 bits）**

  > $N = (-1)^s * 1.fraction * 2^{exponent-127}, 1 \leq exponent \leq 254$
  >
  > Normalized Form最小能够表示的正数：$1.0 * 2^{-126}$
  >
  > Normalized Form最大能够表示的正数：$(2 - 2^{-23}) * 2^{127}$

- **Infinites or NaN**

  > - **Infinites**：$exponent = 255, fraction = 0$
  > - **NaN**：$exponent = 255, fraction \neq 0$

- **Subnormal Numbers/Denormalized number**

  > $exponent = 0, fraction \neq 0$
  >
  > $N = (-1)^s * 0.fraction * 2^{-126}, 1 \leq exponent \leq 254$
  >
  > 注意指数是-126！！！（记忆时只要记得比Normalized Form能够表示的最小的数小即可）

- $Zero$

  > $exponent = 0, fraction = 0$

Single precision		Double precision		Object represented
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	0
0	Nonzero	0	Nonzero	± denormalized number
1–254	Anything	1–2046	Anything	± floating-point number
255	0	2047	0	± infinity
255	Nonzero	2047	Nonzero	NaN (Not a Number)

CH3

Transistor

- 注意$source\ gate\ drain$极的位置

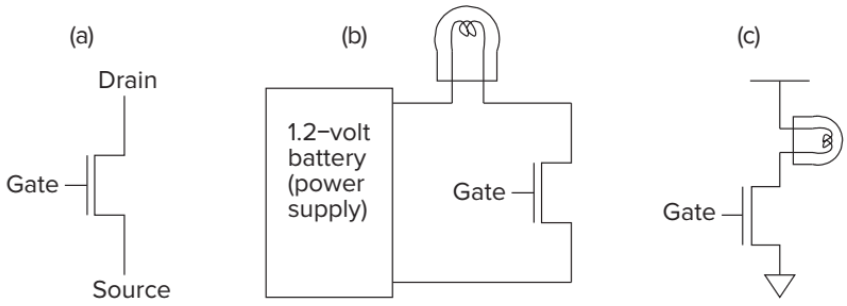


Figure 3.2 The N-type MOS transistor.

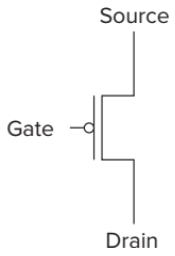


Figure 3.3 A P-type MOS transistor.

- 通断表（注意和通用技术三极管的区别）

	0	1
N-type	断	通
P-type	通	断

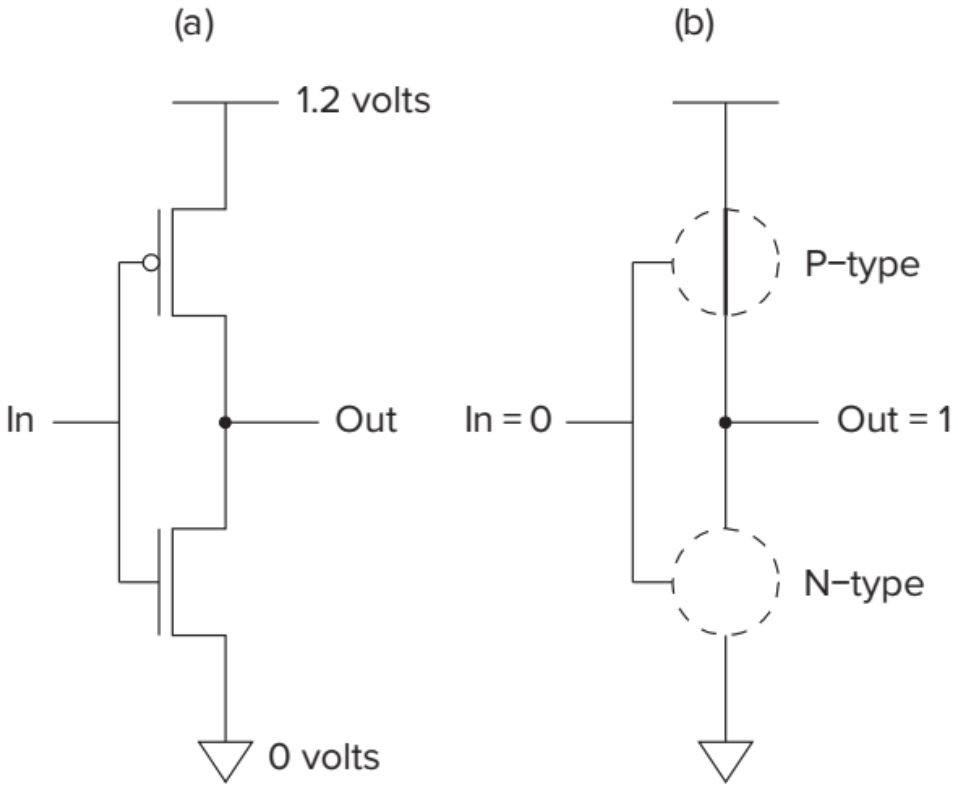
- 接地和接电池的符号
- *CMOS的概念complementary metal-oxide semiconductor*

  上下逻辑互补，只需要记半边

  在$output$上方的逻辑是$f(x)$，则下面的逻辑为$\overline{f(x)}$

Gate: NOT NOR XOR NAND

- **NOT Gate/Inverter**



(c)

In	Out
0 volts	1.2 volts
1.2 volts	0 volts

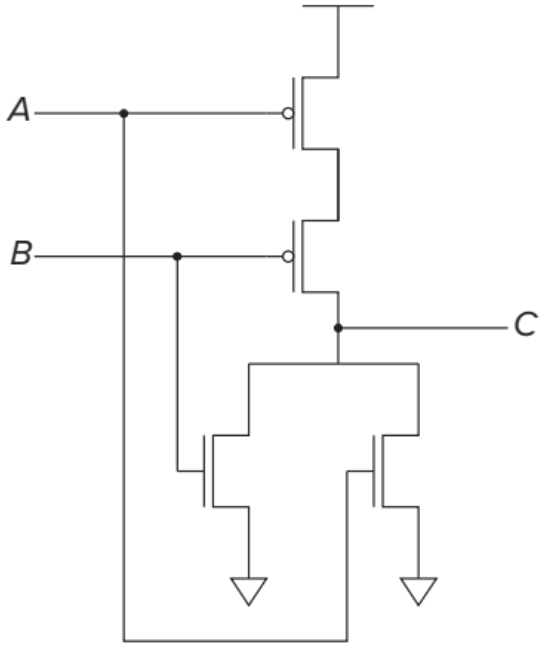
(d)

In	Out
0	1
1	0

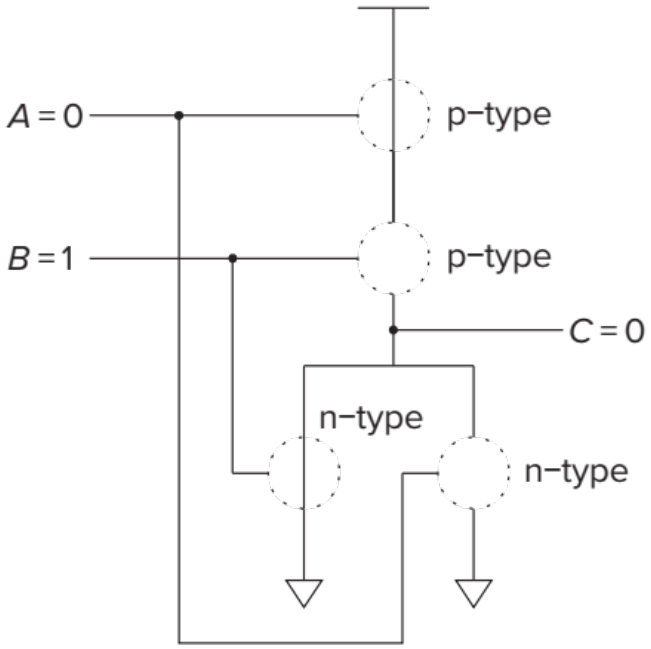
Figure 3.4 A CMOS inverter.

- **NOR**

(a)



(b)



(c)

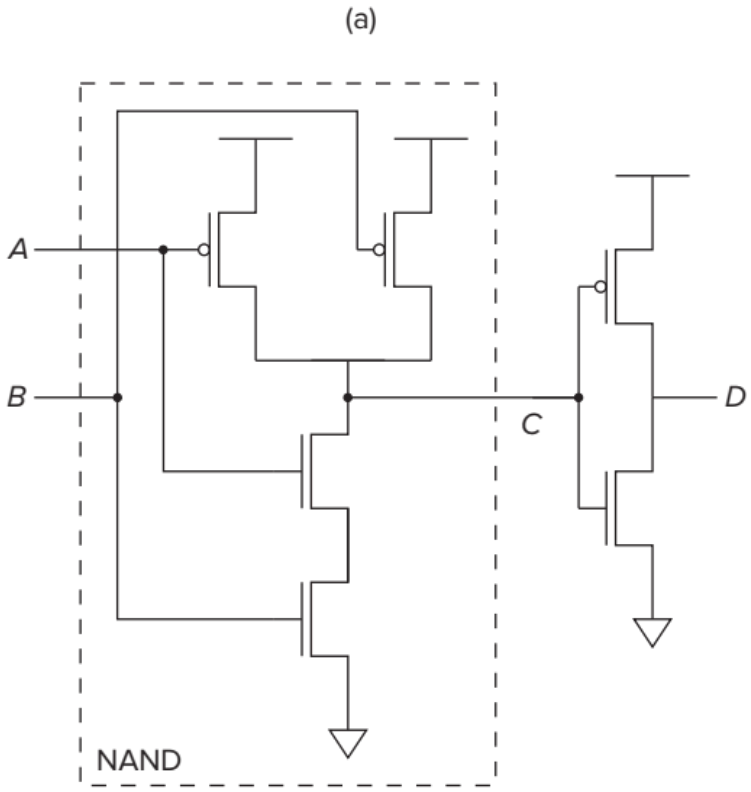
A	B	C
0 volts	0 volts	1.2 volts
0 volts	1.2 volts	0 volts
1.2 volts	0 volts	0 volts
1.2 volts	1.2 volts	0 volts

(d)

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

Figure 3.5 The NOR gate.

- **NAND**



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

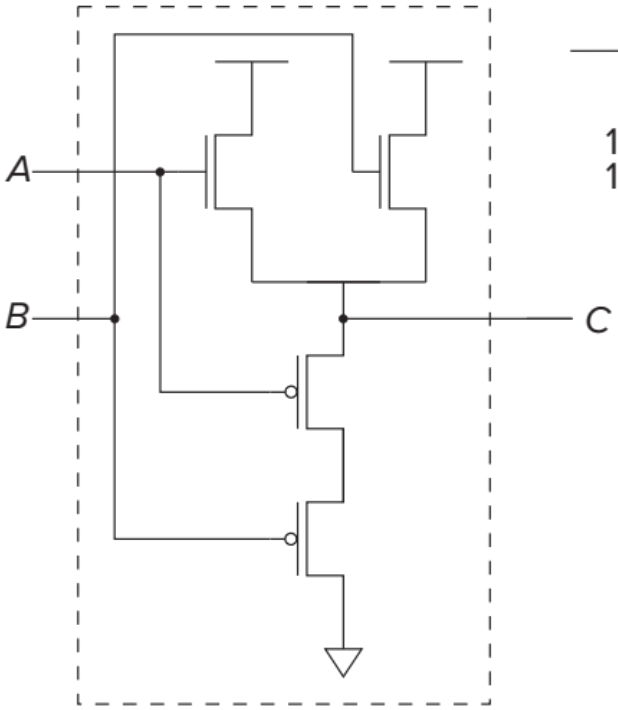
Figure 3.8 The AND gate.

补充问题P67：如何用晶体管制作一个三输入AND?

- **Why We Can't Simply Connect P-Type to Ground**

  这跟晶体管的具体物理性质有关：$P-type$空穴导通通常接在上面，$N-type$电子导通通常接在下面，否则会有大概$0.5V$的误差。

(a)

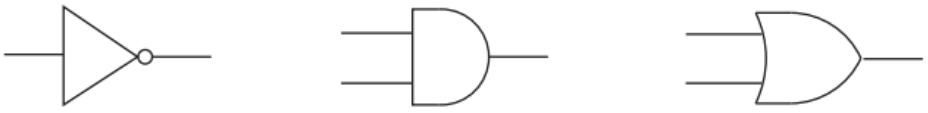


(b)

A	B	C
0 volts	0 volts	1.0 volts
0 volts	1.2 volts	0.7 volts
1.2 volts	0 volts	0.7 volts
1.2 volts	1.2 volts	0.7 volts

Figure 3.7 An OR gate (not really!).

- **Simplied Version**



(a) Inverter (b) AND gate (c) OR gate

(d) NAND gate (e) NOR gate

Figure 3.9 Basic logic gates.

- **Notation**

  - $NOT\ A = \overline{A}$ (读做A bar)
  - $A\ AND\ B = AB$
  - $A\ OR\ B = A + B$
  - $A\ NAND\ B = \overline{AB}$
  - $A\ NOR\ B = \overline{A + B}$

- **Gates with More Than Two Inputs**

(a)

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

(b)

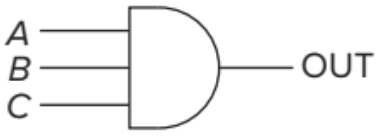


Figure 3.10 A three-input AND gate.

Higher level of abstraction: Mux Decoder Adder

- **Decoder**

  其实就是枚举$0, 1$的组合，每个不同的组合需要一个与门。

  Input Number: n

  Output Number: $\leq 2^n$

(a) (b)

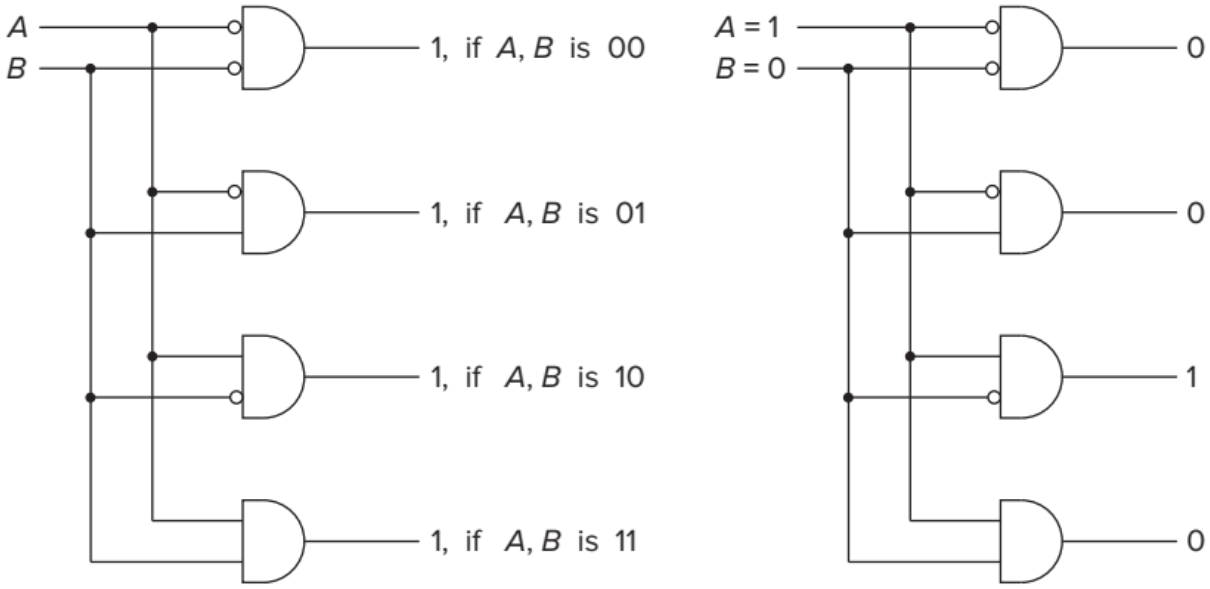


Figure 3.11 A two-input decoder.

- **MUX**（multiplexer）

  > - The function of a mux is to select one of the inputs (A or B) and connect it to the output.
  > - Inputs: A, B
  > - Select line: S
  > - Outputs: C
  > - Essence：Decoder + OR

Input Number: $\leq 2^n$

Select Line Size: n

Output Number: 1

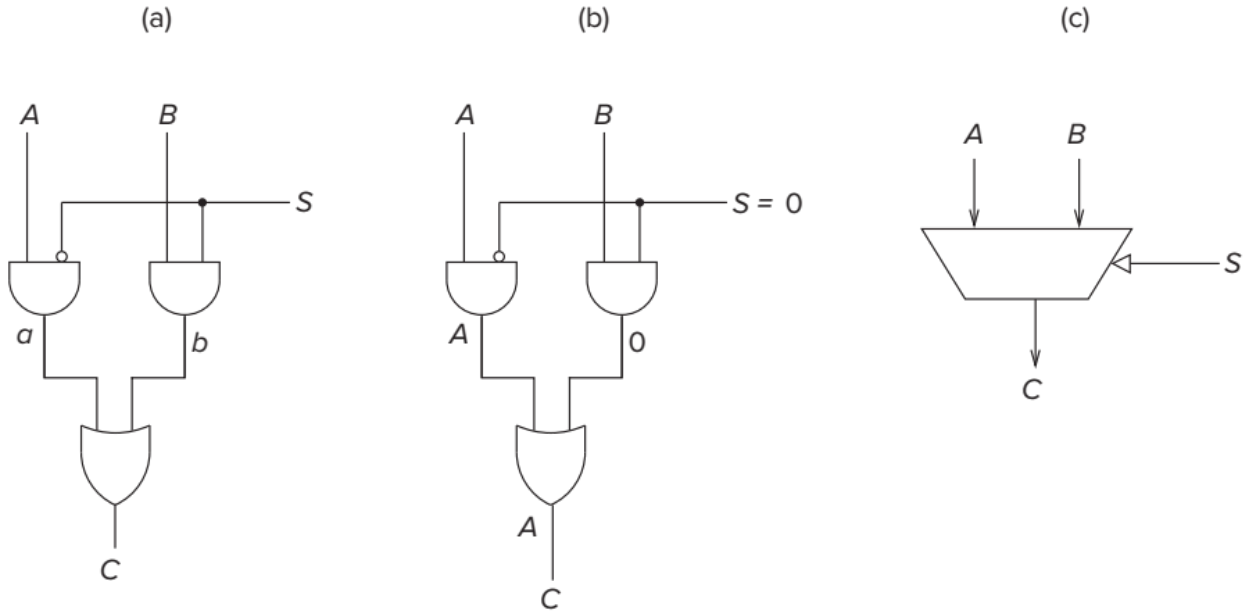


Figure 3.12 A 2-to-1 mux.

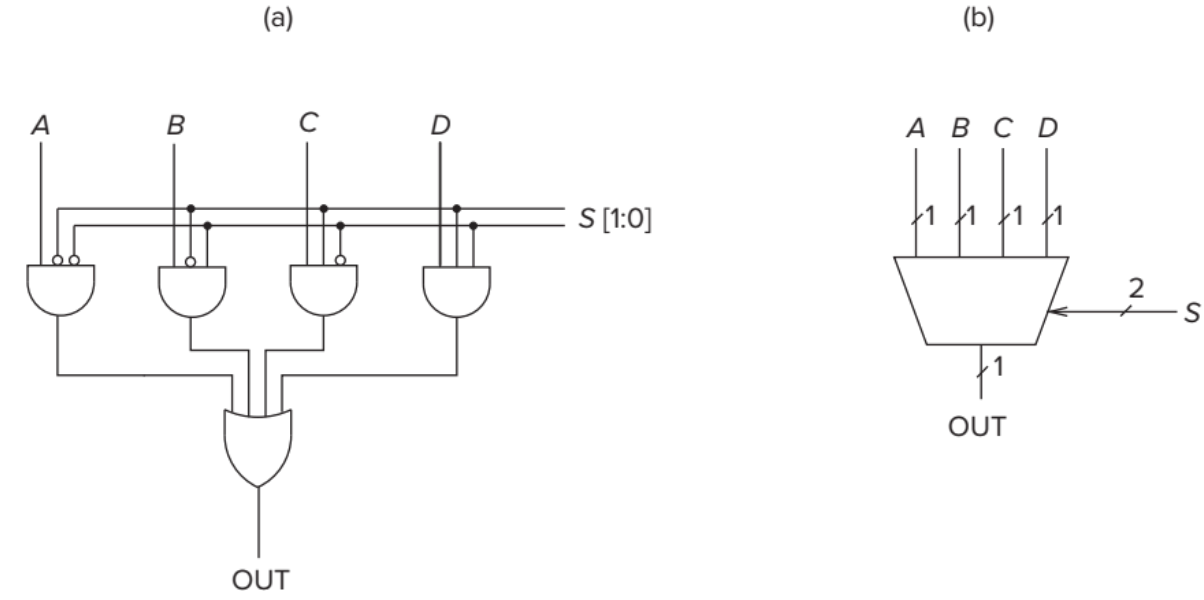


Figure 3.13 A four-input mux.

补充问题P69：如何用门制作一个8输入的mux？你需要多宽的$select\ line$

- **Adder**

  - One-Bit Adder (Full Adder)

  Half Adder仅有两个$input$不考虑上一位的进位

A_i	B_i	C_i	C_{i+1}	S_i
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

The truth table for a one-bit adder.

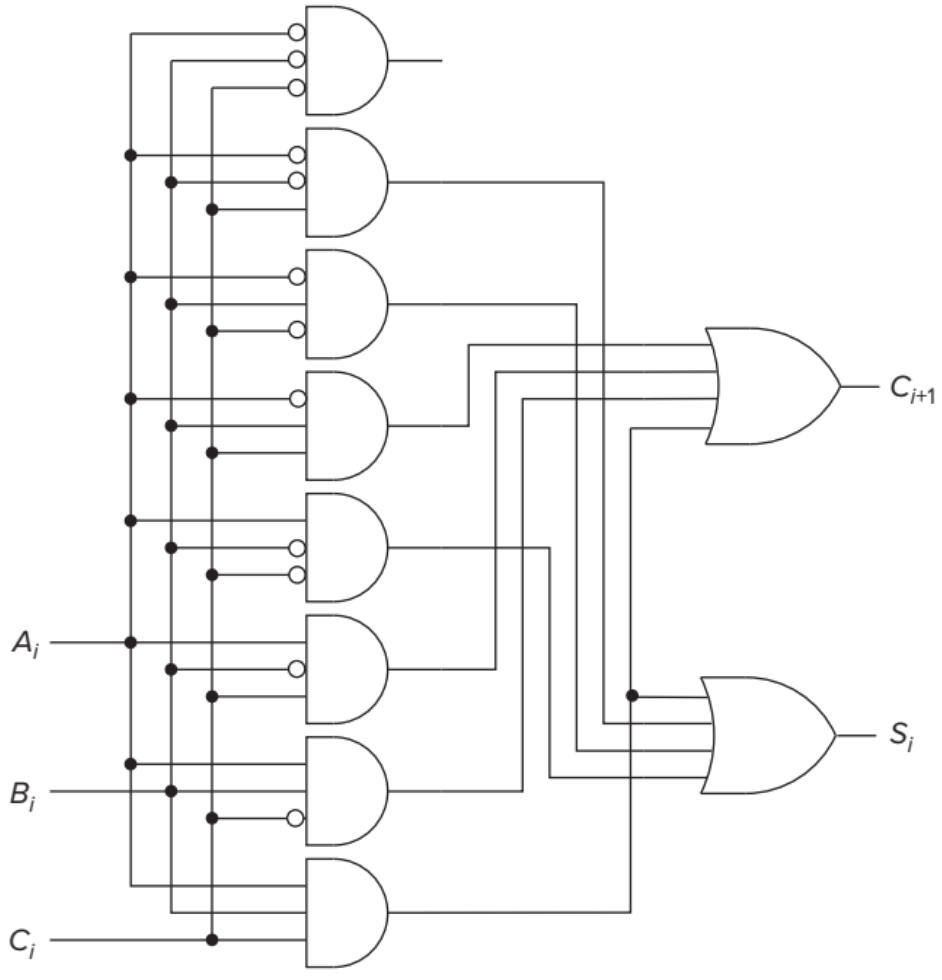


Figure 3.15 Gate-level description of a one-bit adder.

- 4-bit Adder

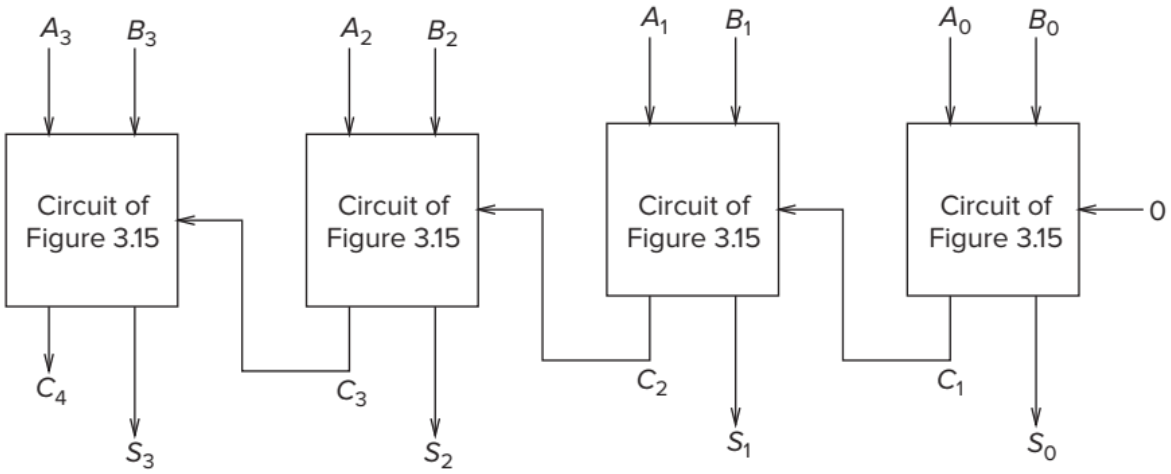


Figure 3.16 A circuit for adding two 4-bit binary numbers.

The Programmable Logic Array (PLA)

Essence: $Decoder$ + OR。本质是通过实现真值表来实现某些特定的函数

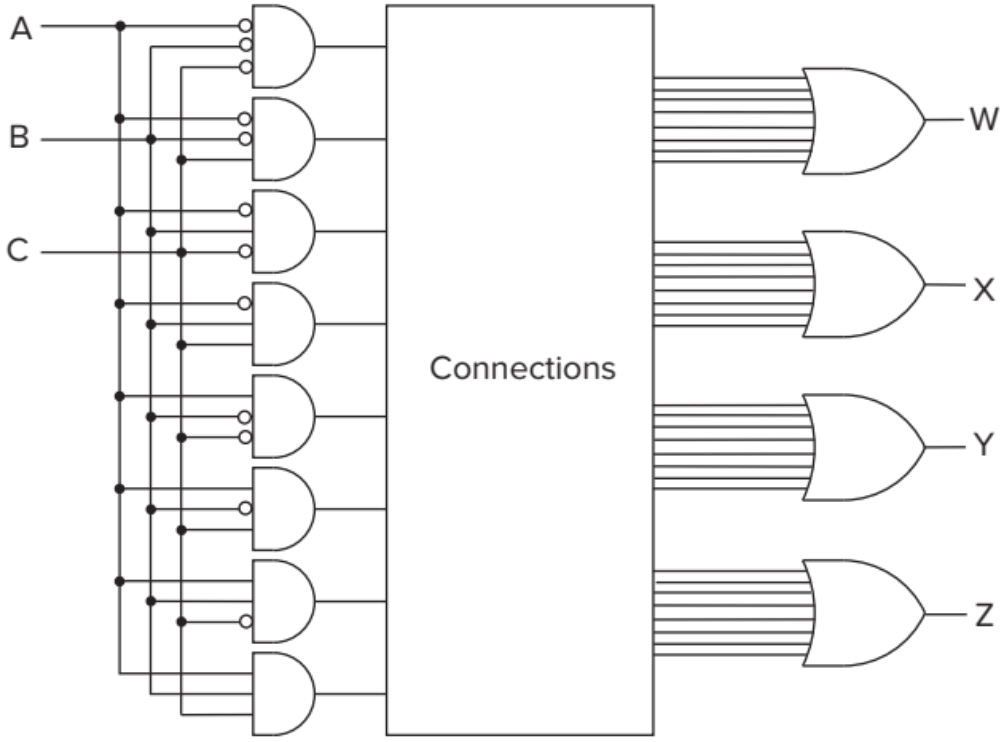


Figure 3.17 A programmable logic array.

Logical Completeness

We say that the set of gates $\{AND, OR, NOT\}$ is **logically complete** (PLA就行) because we can build a circuit to carry out the specification of any truth table we wish without using any other kind of gate.

补充问题P73：Is there any single two-input logic gate that is logically complete?$NAND$ 和 NOR是的

Latch

- **R-S Latch**

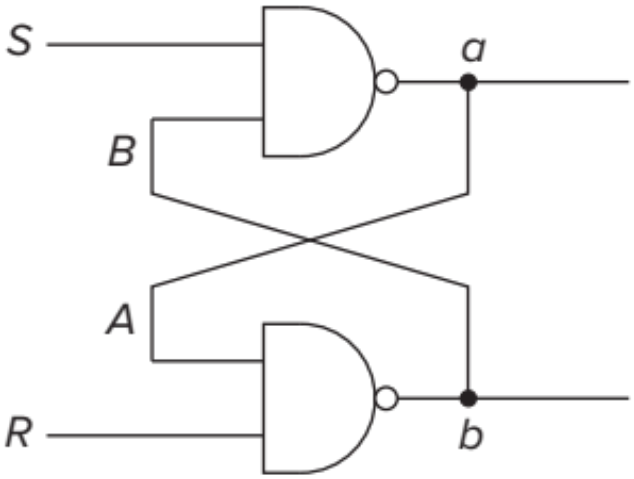


Figure 3.18 An R-S latch.

> 此时Q在上，但是另外一种Q在下(NOR)

R	S	State
1	1	The Quiescent State
1	0	Set (momentarily)
0	1	Reset (momentarily)
0	0	不允许的状态，将会导致不稳定

> **We should point out that if both S and R were allowed to be set to 0 at the same time**, the outputs a and b would both be 1, and the final state of the latch would depend on the electrical properties of the transistors making up the gates and not on the logic being performed. How the electrical properties of the transistors would determine the final state in this case is a subject we will have to leave for a later semester. :-(
>
> When a digital circuit is powered on, the latch can be in either of its two states, 0 or 1. It does not matter which state **since we never use that information until after we have set it to 1 or 0.**

- $D-Latch$

  $WE=0$, 保持态

  $WE=1,\ Q=D$

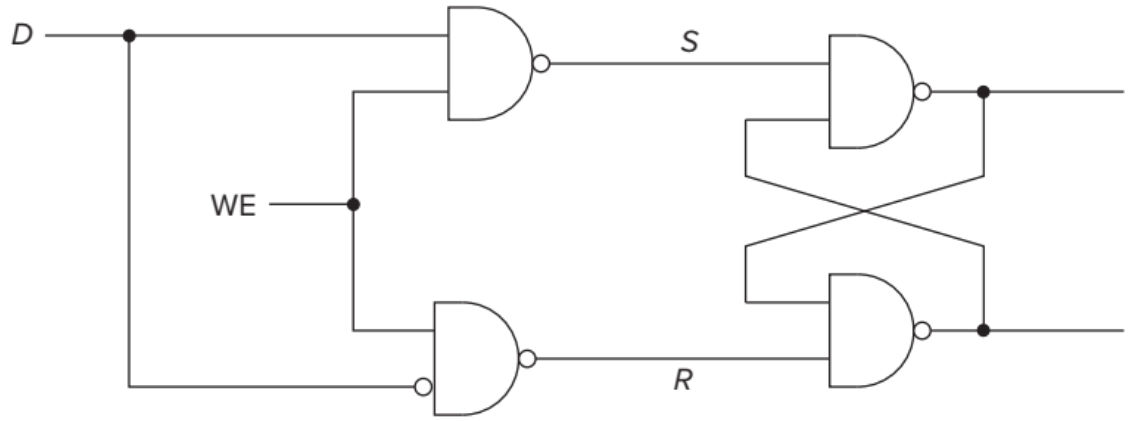


Figure 3.19 A gated D latch.

Memory

A 2^2-by-3-Bit Memory：2^2个地址，每个地址3bit

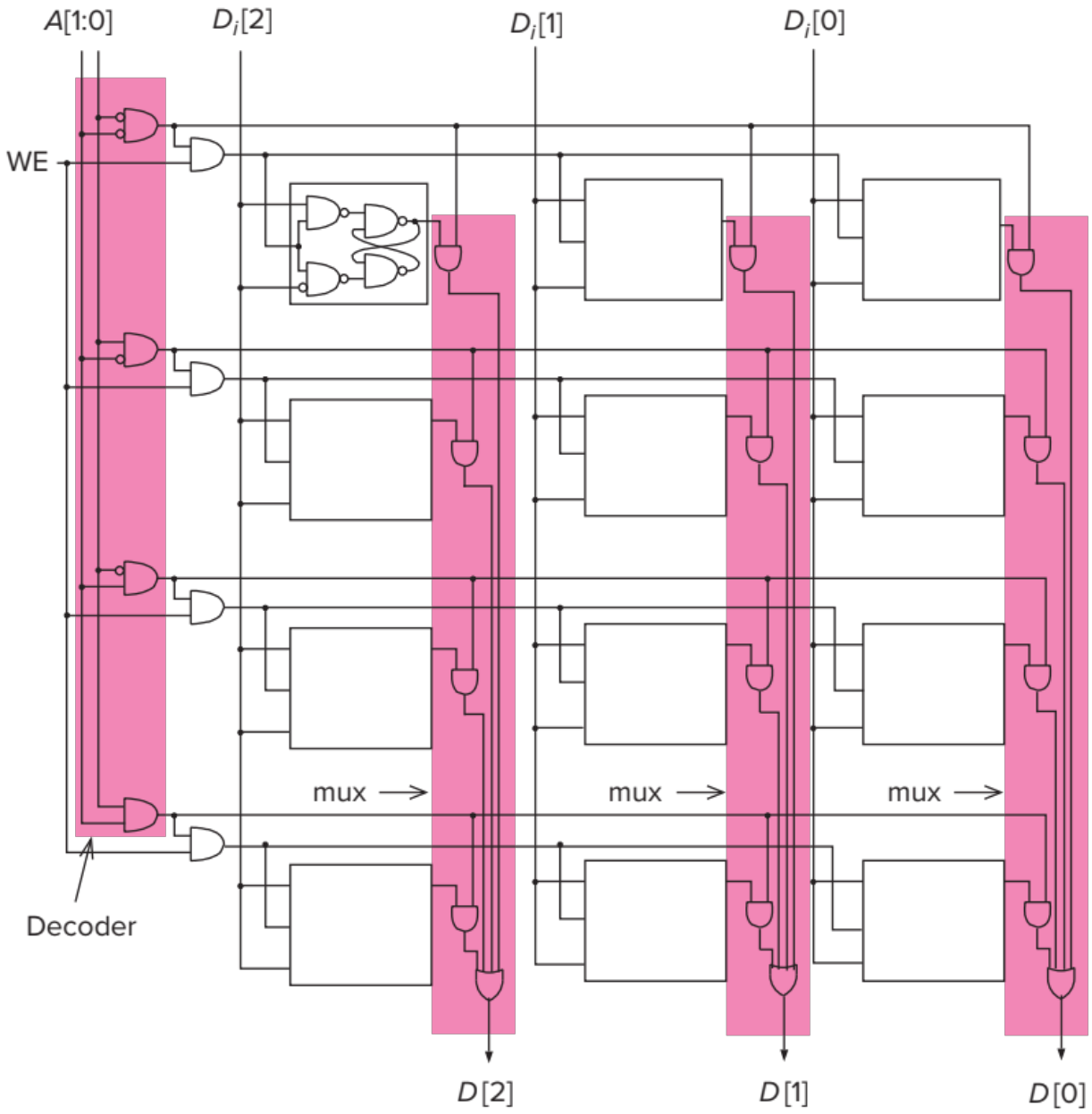


Figure 3.20 A 2^2-by-3-bit memory.

- Address Space

> The total number of uniquely identifiable locations as the memory's address space.
>
> e.g. With n bits of address, we can uniquely identify 2^n locations.

- Addressability

> The number of bits stored in each memory location is the memory's addressability.
>
> e.g. 64-bit addressable <=> addressability is 64

Finite State Machine(FSM)

Synchronous Finite State Machine