

L5 卡诺图 (Karnaugh Maps)

一、组合逻辑复习

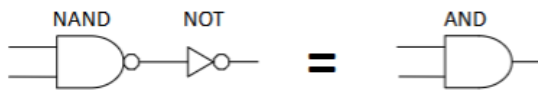
1. 组合逻辑电路特性

- 对输入数据 (0/1) 即时响应, 与历史事件无关。
- 几乎所有组合逻辑电路均可仅用 “与非门 (NAND)” 或 “或非门 (NOR)” 实现。

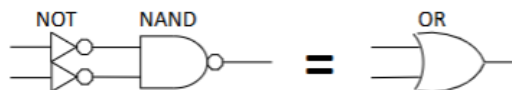
2. 门电路转换

- 与门 (AND) : 由与非门 (NAND) 和非门 (NOT) 组成。
- 或门 (OR) : 由与非门 (NAND) 和非门 (NOT) 组成。
- 非门 (NOT) : 可由与非门直接实现。

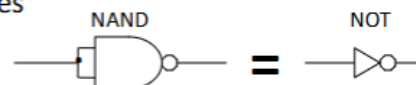
■ AND gates made from NAND and NOT gates



■ OR gates made from NAND and NOT gates



■ NOT gates made from NAND gates



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

3. 标准表达式

- 标准 SOP (积之和)** : 每个乘积项包含表达式域内的所有变量。

例: $\overline{A}BC + \overline{A}\overline{B}C + ABC$ (3 变量, 每项均含 A、B、C)。

作用: 用于构建真值表和卡诺图化简。

- 标准 POS (和之积)** : 每个和项包含表达式域内的所有变量。

例: $(A + B + C)(A + B + \overline{C})(\overline{A} + B + C)$ 。

4. 表达式转换

- SOP 与 POS 互转:**

- 确定 SOP 中乘积项对应的二进制数;
- 找出所有未包含的二进制数;
- 为每个未包含的二进制数写等效和项, 组合为 POS 形式 (反之同理)。

例: 3 变量 SOP 含二进制 000、010、011、101、111, 未含 001、100、110, 对应 POS 为

$(A + B + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + C)$ 。

- SOP 转真值表:**

- 构建所有输入组合的真值表;
- 将 SOP 转为标准形式, 确定对应二进制数;
- 对应二进制行填 1, 其余填 0。

二、卡诺图基础

1. 定义与作用

- 与真值表等价，均表示输入变量所有可能值及对应输出，但以单元格数组形式呈现。
- 提供系统化化简布尔表达式的方法，可得到最简 SOP/POS 表达式。

2. 变量与单元格数量

- 单元格数 = 2^n (n 为变量数)：2 变量 (4 格)、3 变量 (8 格)、4 变量 (16 格)、5 变量 (32 格，分层结构)。

3. 单元格相邻性

- 相邻单元格仅存在 1 个变量的差异 (0→1 或 1→0)。
- 边缘单元格相邻 (如 3 变量卡诺图中 101 与 001 相邻，可视为“圆柱形”结构)。
例：单元格 010 (3 变量) 相邻于 000、011、110，不相邻于 001、111 等。

3 and 4 - variable Karnaugh map

Variables A, B and C

A and B values are displayed on the left side of the Karnaugh map and C on the top.

Combining AB and C in the appropriate cells effectively form a truth table.

Only one variable transition to move to neighbouring cell

AB \ C	0	1
00		
01		
11		
10		



AB \ C	0	1
00	$\overline{A}\overline{B}\overline{C}$	$\overline{A}\overline{B}C$
01	$\overline{A}B\overline{C}$	$\overline{A}BC$
11	$AB\overline{C}$	ABC
10	$A\overline{B}\overline{C}$	$A\overline{B}C$

3 variable Karnaugh map

AB \ CD	00	01	11	10
00	$\overline{A}\overline{B}\overline{C}\overline{D}$	$\overline{A}\overline{B}\overline{C}D$	$\overline{A}\overline{B}C\overline{D}$	$\overline{A}\overline{B}CD$
01	$\overline{A}B\overline{C}\overline{D}$	$\overline{A}B\overline{C}D$	$\overline{A}BC\overline{D}$	$\overline{A}BCD$
11	$AB\overline{C}\overline{D}$	$AB\overline{C}D$	$ABC\overline{D}$	$ABCD$
10	$A\overline{B}\overline{C}\overline{D}$	$A\overline{B}\overline{C}D$	$A\overline{B}C\overline{D}$	$A\overline{B}CD$

4 variable Karnaugh map

Cell Adjacency (expanded)

Cell Adjacency

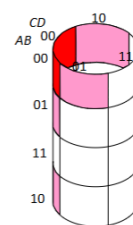
- The cells in the Karnaugh map are arranged so that there is only a single-variable change between adjacent cells.
- Example:
 - cell 010 is adjacent to 000, 011 and 110
 - cell 010 is NOT adjacent to 001, 111, 100 or 101
- Note that edge cells are adjacent to cells in the opposite edge.
- Example:
 - 101 is adjacent to 001 as well as 100 and 111.

AB \ C	0	1
00		
01		
11		
10		

3 variable Karnaugh map

- In a 4 variable Karnaugh map adjacency is more complicated, but the same rules apply.
- One could think of a Karnaugh map as being a continuous cylinder (or a kind of cylinder) with all edges linking together.

AB \ CD	00	01	11	10
00				
01				
11				
10				



三、卡诺图映射方法

1. 标准 SOP 映射

- 每个乘积项对应二进制数的单元格填 1，其余留空 (默认 0)。
例：3 变量 SOP $\overline{A}\overline{B}C + \overline{A}BC + AB\overline{C} + ABC$ 对应二进制 001、011、110、111，在卡诺图对应位置填 1。

2. 非标准 SOP 映射

- 先将非标准项扩展为标准 SOP，再按标准方法映射。
例： $\overline{A} + \overline{A}BC$ 扩展为 $\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC$ 后映射。

3. POS 映射

- 与 SOP 相反，对 POS 中的和项对应二进制数的单元格填 0，分组规则与 SOP 一致，但变量互补性相反。

4. 真值表直接映射

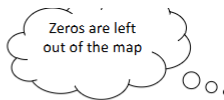
- 将真值表中输出为 1 (SOP) 或 0 (POS) 的行，对应到卡诺图的单元格填 1 或 0。

Mapping a Standard SOP

Each "1" is placed in a cell corresponding to a SOP term.

For example, for a term $\bar{A}\bar{B}C$ a "1" goes to the cell 101 in a 3-variable diagram.

$$\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$



AB \ C	0	1
00	1	
01		
11	1	
10	1	1

Have a go at filling in this Karnaugh map...

Example: Mapping a Non-Standard SOP on 3-variable K-map

Map the following expression:

$$\begin{aligned} &\bar{A} + \bar{A}\bar{B} + \bar{A}BC \\ &\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ &\bar{A}\bar{B}\bar{C} \quad \bar{A}\bar{B}C \quad \bar{A}BC \\ &\bar{A}\bar{B}\bar{C} \quad \bar{A}\bar{B}C \\ &\bar{A}\bar{B}\bar{C} \\ &\bar{A}\bar{B}C \end{aligned} \quad \begin{aligned} &\bar{A} + \bar{A}\bar{B} + \bar{A}BC \\ &\downarrow \quad \downarrow \quad \downarrow \\ &000 \quad 100 \quad 110 \\ &001 \quad 101 \\ &010 \\ &011 \end{aligned}$$

AB \ C	0	1
00	1	1
01	1	1
11	1	
10	1	1

四、卡诺图化简 (SOP 为例)

1. 分组规则

1. 分组大小必须为 1、2、4、8、16 (2^k) 个单元格;
2. 组内每个单元格必须与至少 1 个同组单元格相邻;
3. 优先构建最大可能的组;
4. 每个 1 必须至少属于 1 个组;
5. 允许重复分组 (但组不可完全相同)。

2. 变量消除法则

- 组内变量同时存在原变量和反变量时, 该变量被消除, 仅保留不变的变量。
例: 4 变量卡诺图中 8 单元格的组, 消除 A、C、D, 仅保留 B; 4 单元格的组消除 B、D, 保留 $\bar{A}C$ 。

3. 化简示例


- 表达式 $\bar{A}\bar{B}C\bar{D} + \bar{A}C\bar{D} + \bar{B}C\bar{D} + \bar{A}BC\bar{D}$ 映射后分组, 化简结果为 $\bar{D} + \bar{B}C$ 。

Karnaugh map simplification of SOP expressions

We are now asked to do the reverse process and to simplify a SOP expression (to minimise it)

This is done by grouping the "1" cells in the Karnaugh map according to the following rules:

- A group of cells **must** contain 1, 2, 4, 8 or 16 cells
- Each cell in a group must be adjacent to one or more cells in that group
- Always** include the largest possible number of cells in a group
- Each "1" in the map **must be included in at least one group**
- The "1s" already in a group **can** be included in another group as long as the groups are not identical



AB \ C	0	1
00		1
01	1	
11	1	1
10		

Example: Find a minimum SOP expression

Rule: Eliminate variables that are in the same group in both complemented and uncomplemented forms.

Group of 8 cells:

- A, C and D are all present therefore only B remains.


Groups of 4 cells:

- B and D are present therefore only $\bar{A}C$ remains.

Group of 2 cells:

- B is the only one present with its own complement therefore $\bar{A}C\bar{D}$

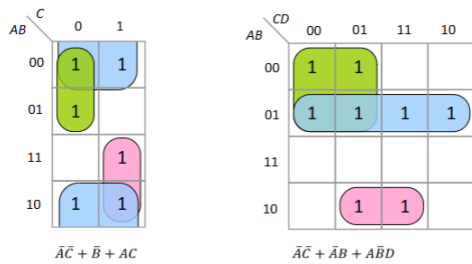
$$B + \bar{A}C + \bar{A}C\bar{D}$$



AB \ CD	00	01	11	10
00			1	1
01	1	1	1	1
11	1	1	1	1
10			1	

Examples of SOP groupings

Have a go!

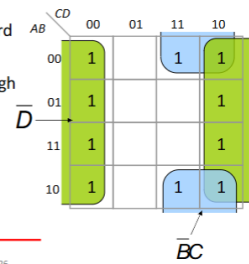


Example: SOP minimisation

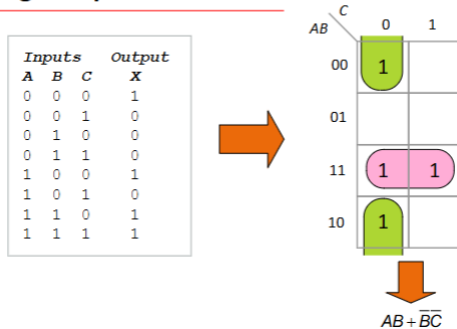
$$BCD + \bar{A}BCD + \bar{A}BC\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + \bar{A}BC\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D}$$

1. First convert term into standard form
2. Plot the expression on Karnaugh map
3. Group the "1s"
4. Find the answer

$$\bar{D} + \bar{B}C$$



Mapping Truth table directly into Karnaugh map



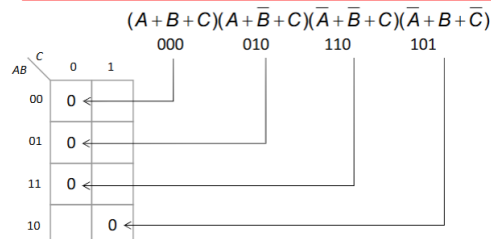
关于 POS

POS and Karnaugh maps

- POS expressions are treated in exactly the same way as the SOPs in connection to Karnaugh maps. **Except that in case of POS you work with 0's rather than 1's.**
- The same tricks for grouping apply. So the reduction of a POS could be done in exactly the same way as SOP, but variables have a reverse complementarity.
- Same rules for "don't care" cases apply.

Mapping POS expression on a Karnaugh map

(Try deriving the POS from the Karnaugh map, then convert it to the equivalent SOP and compare to the SOP derived from the Karnaugh map!!)



二、POS 化简实例 (3 变量: A、B、C)

步骤 1: 确定原始标准 POS 表达式

以如下标准 POS 表达式为例 (每个和项含全部 3 个变量):

$$(A+B+C)(A+B+\bar{C})(\bar{A}+B+C)(\bar{A}+\bar{B}+C)$$

步骤 2: 映射卡诺图 (填 "0" 单元格)

POS 表达式的每个和项对应输出为 0 的输入组合 (和项为 0 时, 所有变量均需满足 "原变量 = 0, 反变量 = 1")。

- 分析和项对应的二进制输入 (3 变量共 $2^3 = 8$ 种组合):

1. $(A+B+C) \rightarrow$ 当 $A=0, B=0, C=0$ 时和项为 0 \rightarrow 二进制 "000";
2. $(A+B+\bar{C}) \rightarrow$ 当 $A=0, B=0, C=1$ 时 $\bar{C}=0$, 和项为 0 \rightarrow 二进制 "001";
3. $(\bar{A}+B+C) \rightarrow$ 当 $A=1, B=0, C=0$ 时 $\bar{A}=0$, 和项为 0 \rightarrow 二进制 "100";
4. $(\bar{A}+\bar{B}+C) \rightarrow$ 当 $A=1, B=1, C=0$ 时 $\bar{A}=0, \bar{B}=0$, 和项为 0 \rightarrow 二进制 "110"。

- 绘制 3 变量卡诺图（行：AB，顺序 00/01/11/10；列：C，顺序 0/1），在上述 4 个二进制对应的单元格填“0”：

AB \ C	0 (C=0)	1 (C=1)
00	0 (000)	0 (001)
01	1	1
11	0 (110)	1
10	0 (100)	1

步骤 3：对“0”单元格分组（遵循 SOP 相同分组规则）

按“最大组、 2^k 大小、相邻性”原则分组：

- 第 1 组：000 (AB=00,C=0) 和 001 (AB=00,C=1) → 组大小 = 2 (C 变化, AB=00 不变)；
- 第 2 组：100 (AB=10,C=0) 和 110 (AB=11,C=0) → 组大小 = 2 (B 变化, A=1、C=0 不变)。

步骤 4：生成最简和项（体现“变量反向互补”）

POS 分组后，保留组内“状态不变的原变量”构成和项（与 SOP 保留“原 / 反变量构成乘积项”相反）：

- 第 1 组 (AB=00 不变)：A=0 对应原变量“A”，B=0 对应原变量“B” → 和项为 $(A + B)$ (当 A=0 且 B=0 时，和项为 0，覆盖该组 2 个“0”单元格)；
- 第 2 组 (A=1、C=0 不变)：A=1 对应反变量 \bar{A} (原变量 A=1 时， $\bar{A} = 0$)，C=0 对应原变量“C” → 和项为 $(\bar{A} + C)$ (当 $\bar{A} = 0$ 且 C=0 时，和项为 0，覆盖该组 2 个“0”单元格)。

步骤 5：得到最简 POS 表达式

将两组生成的和项相乘，即最简结果：

$$(A + B)(\bar{A} + C)$$

五、“无关项” (Don't Care) 条件

1. 定义

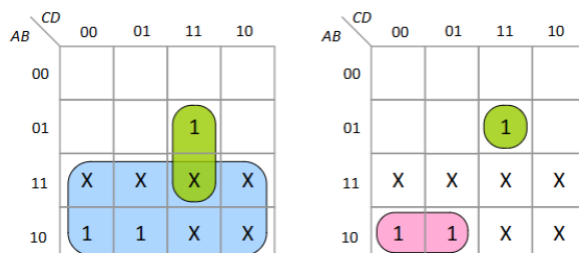
- 某些输入组合不可能出现，其输出值（0 或 1）不影响电路功能，用“X”表示。

2. 化简应用

- X 可视为 1 或 0 参与分组，以构建更大的组，实现更简化的表达式。
例：含 X 的分组可合并更多单元格，使化简结果更简洁。

“Don't care” conditions

- In this case of “don't care” condition, one can use X to represent 1 or 0 in K-map.
- Result using “Xs”: $A + BCD$ not using them: $A\bar{B}\bar{C} + \bar{A}BCD$

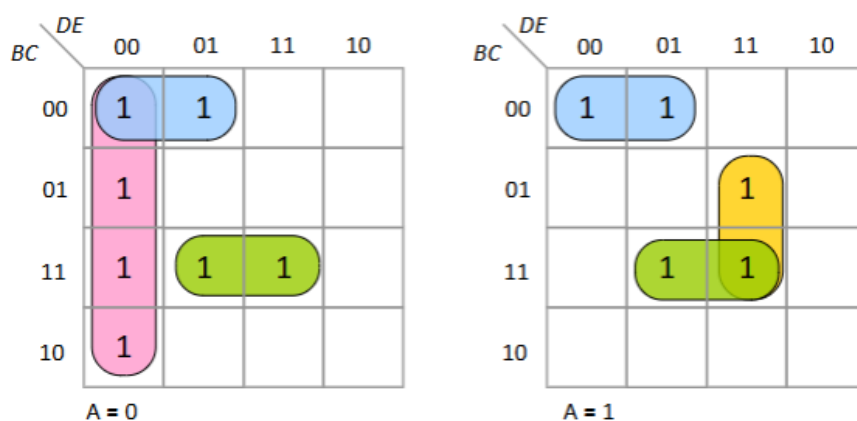


六、5 变量卡诺图

- 采用分层结构：底层为 $A=0$ ，顶层为 $A=1$ ，BC 为行、DE 为列，相邻性规则不变。

5-variable Karnaugh map

- Variables domain is A, B, C, D, E
- This is the case of layering the Karnaugh map with the bottom layer signifying $A=0$ and top layer $A=1$.
- The rules and properties of such maps are still the same.



L6 算术电路 (Arithmetic Circuits)

一、二进制加法

1. 加法规则

- $0 + 0 = 0$ (进位 0) ; $0 + 1 = 1$ (进位 0) ; $1 + 0 = 1$ (进位 0) ; $1 + 1 = 0$ (进位 1) ;
 $1 + 1 + 1 = 1$ (进位 1) 。

2. 多位数加法问题

- N 位加法器的真值表行数为 2^{2N} (如 2 位 16 行、5 位 1024 行) , 直接设计逻辑电路过于复杂, 需拆分组件。

二、加法器电路

1. 半加器 (Half-Adder, HA)

- 功能：实现 2 位二进制数加法，输出“和 (S)”与“进位出 (C_0)”。
- 真值表：

输入 a	输入 b	进位出 C_0	和 S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

2. 全加器 (Full-Adder, FA)

- 功能：实现 3 位二进制数加法（含进位入 C_i ），输出“和 (S)”与“进位出 (C_0)”。
- 真值表：

输入 a	输入 b	进位入 C_i	进位出 C_0	和 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

3. 进位 ripple 加法器 (Carry-Ripple Adder)

- 结构：1 个半加器（最低位，无进位入）+ (N-1) 个全加器（其余位，接收前一位进位）。
例：4 位加法器由 1 个 HA + 3 个 FA 组成，输出 4 位和 ($S_3S_2S_1S_0$) 和 1 位进位出 (C_0)。

4. 级联加法器

- 多个小位数加法器级联为大位数加法器，前一级进位出连接后一级进位入。
例：2 个 4 位加法器级联为 8 位加法器。

三、二进制减法与减法器

1. 减法规则

- $0 - 0 = 0$ (借位 0) ; $0 - 1 = 1$ (借位 1) ; $1 - 0 = 1$ (借位 0) ; $1 - 1 = 0$ (借位 0) ; 借位时低位从高位借 1 当 2。

2. 2 的补码 (Two's Complement)

- 作用：将减法转换为加法 ($A - B = A + (-B)$)，简化电路设计。
- 表示规则：
 - 最高位为符号位：0 (正数)、1 (负数)；
 - 正数：原码本身；
 - 负数：原码取反 (1 的补码) + 1。
例：4 位中，-2 的原码 0010 → 取反 1101 → 加 1 → 1110 (2 的补码)。

3. 减法器实现

- 用加法器构建：将被减数 B 取反，设置加法器进位入 $C_i=1$ (实现加 1)，输出前 N 位为结果。

2.2. Two's Complement

- Two's complement can be used to represent negative numbers.
- Left most bit; 0 indicates a positive number, 1 indicates a negative number.

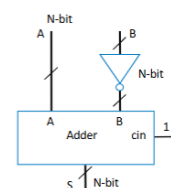
Positive numbers		Negative numbers
0	0000 → 1000	×
1	0001 → 1111	-1
2	0010 → 1110	-2
3	0011 → 1101	-3
4	0100 → 1100	-4
5	0101 → 1011	-5
6	0110 → 1010	-6
7	0111 → 1001	-7
×	1000 → 1000	-8

2.4. Subtractors

- Building a subtractor with an adder using two's complement

$$\begin{aligned} A - B &= A + (-B) \\ &= A + (\text{two's complement of } B) \\ &= A + (\text{one's complement of } B + 1) \\ &= A + \text{inverting bits of } (B) + 1 \end{aligned}$$

- To build a subtractor using an adder
 - Invert bits of B and setting carry-in to 1
 - The remainder is the first N-bit of output



2 的补码 (Two's Complement) 表示负数的原理与规则

一、核心原理

2 的补码是数字电子技术中表示有符号二进制数的标准方法，其核心价值在于：将负数表示为特定的二进制形式，从而把“减法运算”转化为“加法运算”（即 $A - B = A + (-B)$ ），无需额外设计减法电路，仅通过加法器即可实现加减统一运算，极大简化了算术电路的设计。

二、关键规则：符号位与数值表示

以你提供的4 位二进制数为例（位数决定表示范围，4 位补码可表示 -2^3 到 $2^3 - 1$ ，即 -8 到 7），其表示规则如下：

1. 符号位定义（最高位，Left most bit）

- 二进制数的最高位（第 3 位，从 0 开始计数）为符号位，直接标识数的正负：
 - 符号位 = 0 → 正数
 - 符号位 = 1 → 负数

2. 正数的 2 的补码表示

- 正数的 2 的补码 等于其自身的二进制原码（原码：直接用二进制表示数值的形式）。结合表格实例（4 位）：

十进制正数	二进制原码 (= 2 的补码)	符号位
1	0001	0 (正)
2	0010	0 (正)
3	0011	0 (正)
4	0100	0 (正)
5	0101	0 (正)
6	0110	0 (正)
7	0111	0 (正)

3. 负数的 2 的补码表示

- 负数的 2 的补码通过其“绝对值的二进制原码”经过两步计算得到：
 - 对绝对值的原码取“1 的补码”（即按位取反：0→1, 1→0）；
 - 对 1 的补码加 1，得到最终的 2 的补码。

结合表格实例（4 位，以十进制 -1 到 -8 为例）：

十进制负数	绝对值的原码	1 的补码 (按位取反)	2 的补码 (1 的补码 + 1)	符号位
-1	0001	1110	1111	1 (负)
-2	0010	1101	1110	1 (负)
-3	0011	1100	1101	1 (负)
-4	0100	1011	1100	1 (负)
-5	0101	1010	1011	1 (负)
-6	0110	1001	1010	1 (负)
-7	0111	1000	1001	1 (负)
-8	1000 (特殊)	——	1000	1 (负)

4. 特殊情况：-8 的表示

- 4 位二进制中，“1000” 是唯一的 “非对称” 表示：它没有对应的正数原码（正数最大为 0111=7），直接定义为 -8，这是由补码的位数限制导致的（n 位补码的负数范围比正数多 1 个，即 -2^{n-1} 到 $2^{n-1} - 1$ ）。

三、为什么 2 的补码能实现 “减变加”？

以 “计算 $7 - 3$ ” 为例（4 位补码）：

- 转化为加法： $7 - 3 = 7 + (-3)$ ；
- 查补码表：7 的补码 = 0111，-3 的补码 = 1101；
- 加法运算： $0111 + 1101 = 10100$ （4 位结果取前 4 位 “0100”，进位 “1” 丢弃）；
- 还原结果：“0100” 对应十进制 4，与 $7 - 3 = 4$ 一致。

这一特性正是文档中 “用加法器实现减法器” 的核心依据。

四、比较器电路

1. 相等比较器 (Equality Comparator)

- 功能：**N 位输入 A、B，输出 1 当且仅当所有对应位相等 ($a_n = b_n$)。
- 例：4 位相等比较器需满足 $a_3 = b_3 \wedge a_2 = b_2 \wedge a_1 = b_1 \wedge a_0 = b_0$ 。

2. 幅度比较器 (Magnitude Comparator)

- 功能：**输出 $A > B$ 、 $A = B$ 、 $A < B$ 三种结果。
- 比较逻辑：**高位优先，从最高位开始比较，若相等则比较下一位，直至出现不等或比较完所有位。
- 结构：**分阶段设计，每级接收高位比较结果，输出当前级结果至低位。

■ **N-bit equality comparator:** Outputs 1 if two N -bit numbers are equal

■ Design using combinational design process

■ **Step 1** - Capture the function

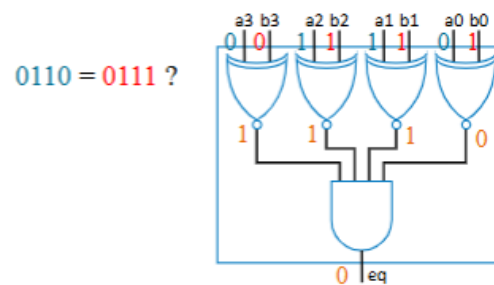
- E.g. 4-bit equality comparator with inputs A and B
- $a_3 = b_3$ and $a_2 = b_2$ and $a_1 = b_1$ and $a_0 = b_0$
- Two bits are equal if both 1 or both 0

■ **Step 2a** - Create equations

$$eq = (a_3b_3 + \overline{a_3}\overline{b_3}) \cdot (a_2b_2 + \overline{a_2}\overline{b_2}) \cdot (a_1b_1 + \overline{a_1}\overline{b_1}) \cdot (a_0b_0 + \overline{a_0}\overline{b_0})$$

$$eq = (.) \cdot (.) \cdot (.) \cdot (.)$$

■ **Step 2b** - Implement as circuit



■ **N-bit magnitude comparator:** Two N -bit inputs A and B, outputs whether $A > B$, $A = B$, or $A < B$, for

■ **Step 1** - Capture the function

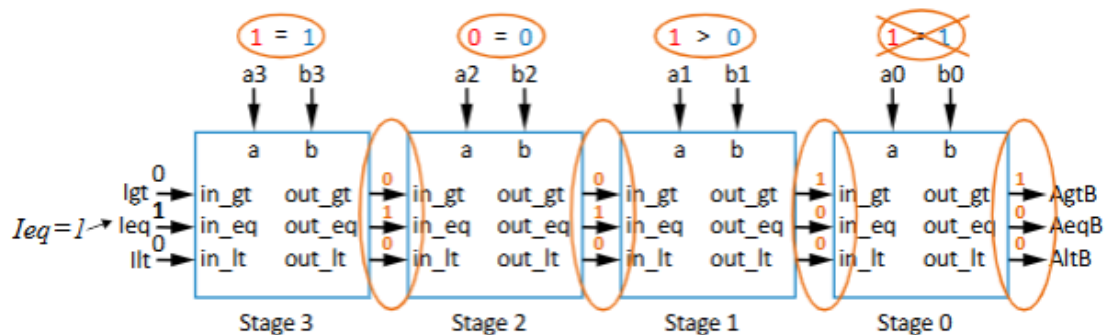
- E.g. 4-bit magnitude comparator with inputs A and B
- First compare a_3 and b_3 . If equal, compare a_2 and b_2 and so on.
- Stop if comparison not equal.

■ **Step 2a, 2b** - Create equations and implement as circuit

- Each bit pair called a *stage*
- Each stage has 3 inputs taking results of higher stage, outputs new results to lower stage

■ How does it work?

$$1011 = 1001 ?$$



■ Each stage:

■ $out_gt = in_gt + (in_eq \cdot a \cdot \bar{b})$

■ $out_lt = in_lt + (in_eq \cdot \bar{a} \cdot b)$

■ $out_eq = in_eq \cdot (a \oplus b)$

Have a go at
building these
three circuits!

