Electronics Systems (938II)

Lecture 4.1

Programmable Logic Devices – Principles and PAL

Luca Crocetti

Introduction

- **PLD** = **P**rogrammable **L**ogic **D**evice

  - Programming HW, not SW!

  - Configurable logic elements that can implement different circuits
    - Configuration #1 → circuit/logic function #1
    - Configuration #2 → circuit/logic function #2
    - …

  - Not different pieces of HW, but the same piece of HW used in different ways!

Introduction

- **PLD**

  - We have already seen some examples of programmable connections
    - Fuses/anti-fuses
    - MOS transistors
    - …

  - And how to program logic functions?

Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression
  - Operators
    - $+$ → OR (gate)
    - $\cdot$ → AND (gate)
    - $\overline{}$ → NOT (gate)
  - Forms
    - **S**um-of-**P**roducts (**SP**)
    - **P**roduct-of-**S**ums (**PS**)

Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression

  - Operators
    - $+$ → OR (gate)
    - $\cdot$ → AND (gate)
    - $\overline{}$ → NOT (gate)

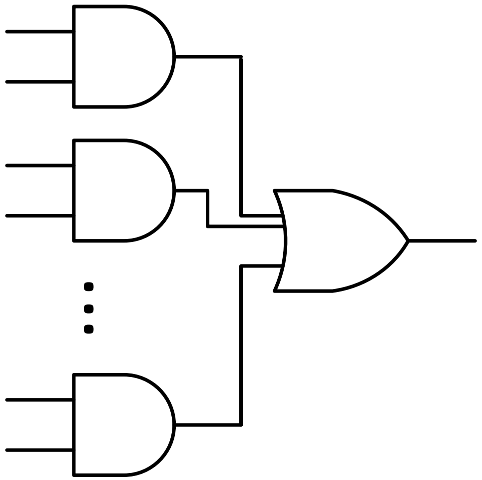
  - Forms
    - **S**um-of-**P**roducts (**SP**) → AND gates feeding an OR gate
    - **P**roduct-of-**S**ums (**PS**)

Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression

  - Operators
    - $+$ → OR (gate)
    - $\cdot$ → AND (gate)
    - $\overline{}$ → NOT (gate)

  - Forms
    - **S**um-of-**P**roducts (**SP**) → AND gates feeding an OR gate →
    - **P**roduct-of-**S**ums (**PS**)



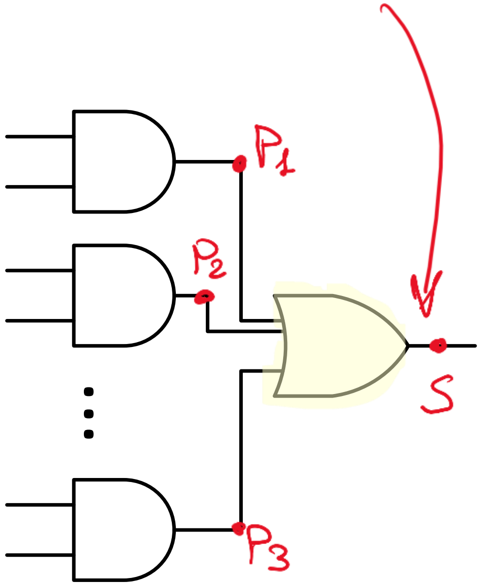
Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression

  - Operators
    - $+$ → OR (gate)
    - $\cdot$ → AND (gate)
    - $\overline{}$ → NOT (gate)

  - Forms
    - **S**um-of-**P**roducts (**SP**) → AND gates feeding an OR gate →
    - **P**roduct-of-**S**ums (**PS**)

$S = P_1 + P_2 + \dots + P_3$

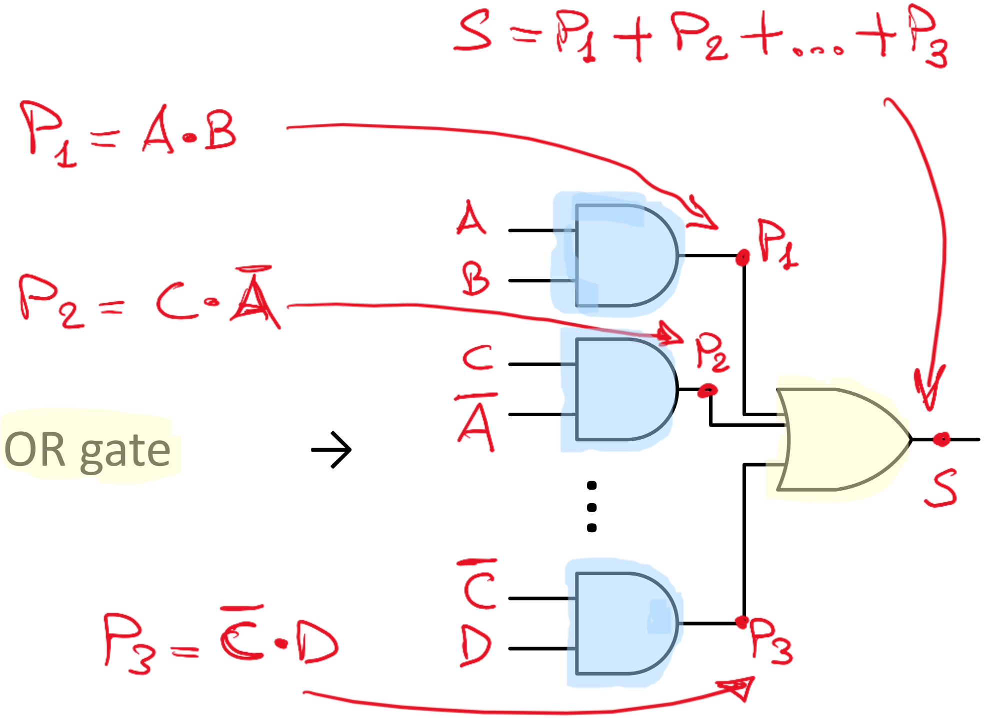


Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression

  - Operators
    - $+$ → OR (gate)
    - $\cdot$ → AND (gate)
    - $\overline{}$ → NOT (gate)

  - Forms
    - **S**um-of-**P**roducts (**SP**) → AND gates feeding an OR gate →
    - **P**roduct-of-**S**ums (**PS**)



Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression

  - Operators
    - $+$ → OR (gate)
    - $\cdot$ → AND (gate)
    - $\overline{}$ → NOT (gate)

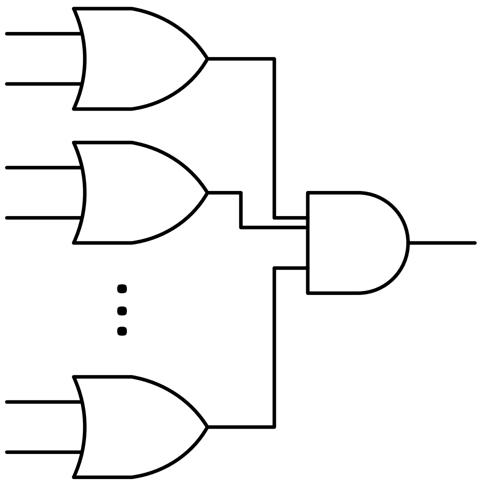
  - Forms
    - **S**um-of-**P**roducts (**SP**)
    - **P**roduct-of-**S**ums (**PS**) → OR gates feeding an AND gate

Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression

  - Operators
    - $+$ → OR (gate)
    - $\cdot$ → AND (gate)
    - $\overline{}$ → NOT (gate)

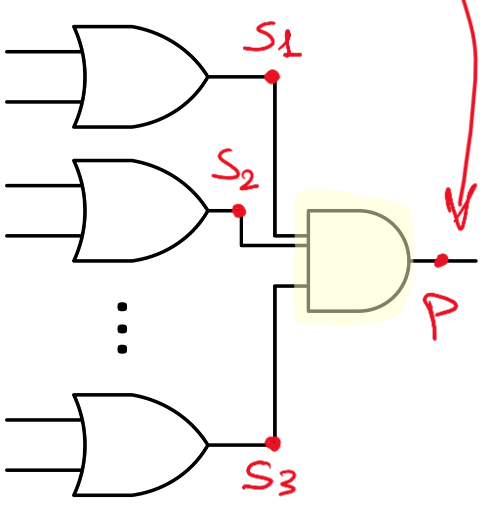
  - Forms
    - **S**um-of-**P**roducts (**SP**)
    - **P**roduct-of-**S**ums (**PS**) → OR gates feeding an AND gate →



Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression
  - Operators
    - $+$ → OR (gate)
    - $\cdot$ → AND (gate)
    - $\overline{}$ → NOT (gate)
  - Forms
    - **S**um-of-**P**roducts (**SP**)
    - **P**roduct-of-**S**ums (**PS**) → OR gates feeding an AND gate →

$P = S_1 \cdot S_2 \cdot \ldots \cdot S_3$

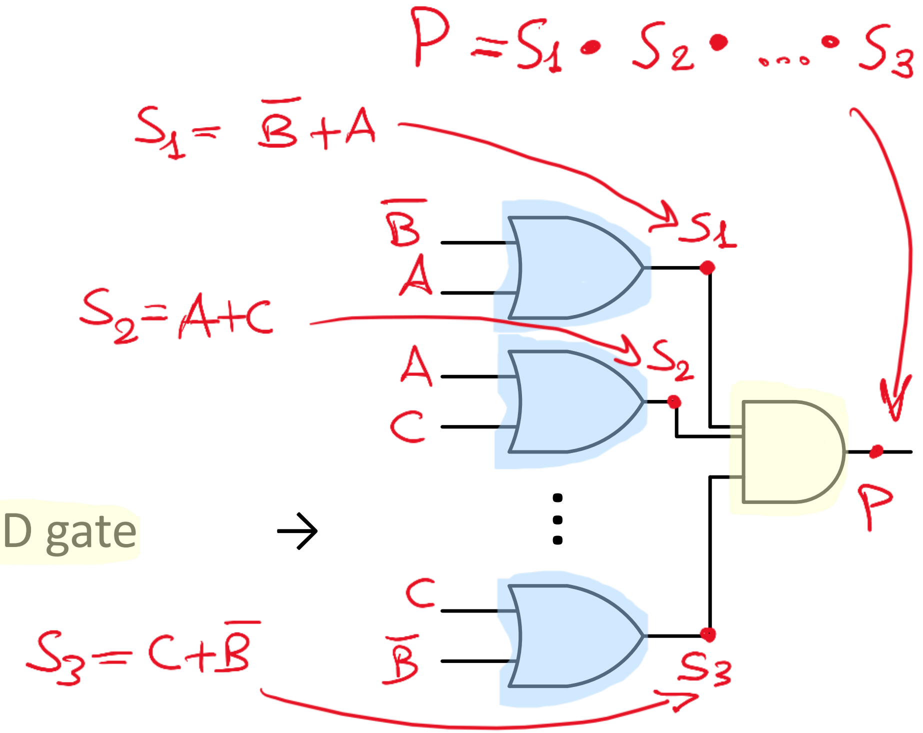


Principles of programmable logic

- Any combinational logic function can be represented by a Boolean expression

  - Operators
    - + → OR (gate)
    - · → AND (gate)
    - ¯ → NOT (gate)

  - Forms
    - **S**um-of-**P**roducts (**SP**)
    - **P**roduct-of-**S**ums (**PS**) → OR gates feeding an AND gate →



Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a,b) = a \cdot b$
    - AND gate/function

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a, b) = a \cdot b$
    - AND gate/function
    - Truth table →

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

    - KM

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a, b) = a \cdot b$
    - AND gate/function
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

    - KM

a \ b	0	1
0	0	0
1	0	1

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a, b) = a \cdot b$
    - AND gate/function
    - Truth table
    - KM

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

		b	
		0	1
a	0	0	0
	1	0	1

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a,b) = a \cdot b$
    - AND gate/function
    - Truth table
    - KM

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

a \ b	0	1
0	0	0
1	0	1

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case

- Example #1
  - $f_1(a, b) = a \cdot b$
  - AND gate/function
  - Truth table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

  - KM

a \ b	0	1
0	0	0
1	0	1

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a, b) = a \cdot b$
    - AND gate/function
    - Truth table
    - KM

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

		b	
		0	1
a	0	0	0
	1	0	1

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a,b) = a \cdot b$
    - AND gate/function
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

    - KM

a \ b	0	1
0	0	0
1	0	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a, b) = a \cdot b$
    - AND gate/function
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

    - KM

a \ b	0	1
0	0	0
1	0	1

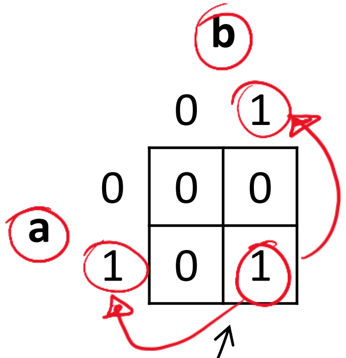
- How to
  1. **Find all the largest group(s) of contiguous 1s**
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a, b) = a \cdot b$
    - AND gate/function
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

    - KM



- How to
  1. Find all the largest group(s) of contiguous 1s
  2. **For each group, select only the inputs whose value does not change**
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case

- Example #1
  - $f_1(a, b) = a \cdot b$
  - AND gate/function
  - Truth table →

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

  - KM

a \ b	0	1
0	0	0
1	0	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. ~~**Complement surviving inputs whose value is 0**~~
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case

- Example #1
  - $f_1(a, b) = a \cdot b$
  - AND gate/function
  - Truth table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

  - KM

$P_0 = a \cdot b$

a \ b	0	1
0	0	0
1	0	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. **Multiply the surviving inputs → products P_i**
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case

- Example #1
  - $f_1(a, b) = a \cdot b$
  - AND gate/function
  - Truth table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

  - KM

$P_0 = a \cdot b$

	b	
a	0	1
0	0	0
1	0	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. **Sum the products**

Principles of programmable logic

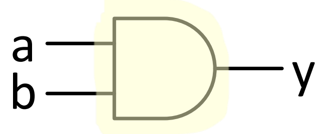
- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #1
    - $f_1(a, b) = a \cdot b$
    - AND gate/function
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

    - KM

a \ b	0	1
0	0	0
1	0	1

Logic circuit



Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #2
    - $f_2(a, b) = ??$
    - Truth table
    - KM

a	b	y
0	0	0
0	1	0
1	0	1
1	1	1

$$
\begin{array}{cc|c|c}
 & & \multicolumn{2}{c}{\mathbf{b}} \\
 & & 0 & 1 \\
\hline
\mathbf{a} & 0 & 0 & 0 \\
\hline
 & 1 & 1 & 1 \\
\hline
\end{array}
$$

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #2
    - $f_2(a,b) = ??$
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	1
1	1	1

    - KM

a \ b	0	1
0	0	0
1	1	1

- How to
  1. **Find all the largest group(s) of contiguous 1s**
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case

- Example #2
  - $f_2(a, b) = ??$
  - Truth table

a	b	y
0	0	0
0	1	0
1	0	1
1	1	1

  - KM

a \ b	0	1
0	0	0
1	1	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. **For each group, select only the inputs whose value does not change**
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #2
    - $f_2(a, b) = ??$
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	1
1	1	1

    - KM

a \ b	0	1
0	0	0
1	1	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. ~~**Complement surviving inputs whose value is 0**~~
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #2
    - $f_2(a, b) = ??$
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	1
1	1	1

    - KM

$P_0 = a$

	b = 0	b = 1
a = 0	0	0
a = 1	1	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. **Multiply the surviving inputs → products P_i**
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #2
    - $f_2(a,b) = \boldsymbol{a}$
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	1
1	1	1

    - KM

$P_0 = a \cdot b$

a \ b	0	1
0	0	0
1	1	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. **Sum the products**

Principles of programmable logic

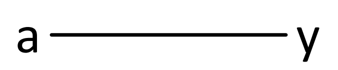
- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #2
    - $f_2(a, b) = \boldsymbol{a}$
    - Truth table

a	b	y
0	0	0
0	1	0
1	0	1
1	1	1

    - KM

a \ **b**	0	1
0	0	0
1	1	1

Logic circuit



Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #3
    - $f_3(a, b) = ??$
    - Truth table
    - KM

a	b	y
0	0	1
0	1	1
1	0	0
1	1	0

a \ b	0	1
0	1	1
1	0	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #3
    - $f_3(a,b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	0

    - KM

a \ b	0	1
0	1	1
1	0	0

- How to
  1. **Find all the largest group(s) of contiguous 1s**
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #3
    - $f_3(a,b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	0

    - KM

a \ b	0	1
0	1	1
1	0	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. **For each group, select only the inputs whose value does not change**
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #3
    - $f_3(a,b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	0

    - KM

$\bar{a}$

a \ b	0	1
0	1	1
1	0	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. **Complement surviving inputs whose value is 0**
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #3
    - $f_3(a, b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	0

    - KM

$P_0 = \bar{a}$

$\bar{a}$

a \ b	0	1
0	1	1
1	0	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. **Multiply the surviving inputs → products P_i**
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case

$P_0 = \bar{a}$

- Example #3
  - $f_3(a, b) = \bar{\boldsymbol{a}}$
  - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	0

  - KM

$\bar{a}$

a \ b	0	1
0	1	1
1	0	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. **Sum the products**

Principles of programmable logic

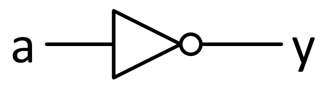
- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #3
    - $f_3(a,b) = \bar{\boldsymbol{a}}$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	0

    - KM

		b	
		0	1
a	0	1	1
	1	0	0

Logic circuit



Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #4
    - $f_4(a, b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	1

    - KM

a \ b	0	1
0	1	1
1	0	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #4
    - $f_4(a, b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	1

    - KM

a \ b	0	1
0	1	1
1	0	1

- How to
  1. **Find all the largest group(s) of contiguous 1s**
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #4
    - $f_4(a, b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	1

    - KM

a \ b	0	1
0	1	1
1	0	1

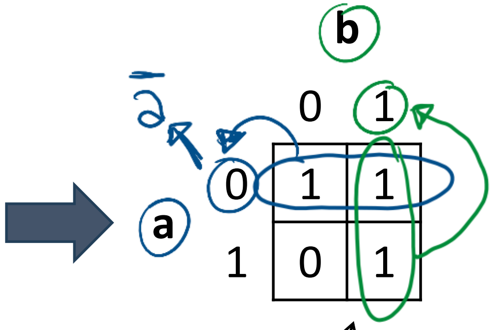
- How to
  1. Find all the largest group(s) of contiguous 1s
  2. **For each group, select only the inputs whose value does not change**
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #4
    - $f_4(a, b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	1

    - KM



- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. **Complement surviving inputs whose value is 0**
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #4
    - $f_4(a, b) = ??$
    - Truth table

a	b	y
0	0	1
0	1	1
1	0	0
1	1	1

    - KM

$P_0 = \bar{a}$

$P_1 = b$

a \ b	0	1
0	1	1
1	0	1

$\bar{a}$

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. **Multiply the surviving inputs → products P_i**
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #4
    - $f_4(a,b) = \bar{\boldsymbol{a}} + \boldsymbol{b}$
    - Truth table →
    - KM

$$\begin{cases} P_0 = \bar{a} \\ P_1 = b \end{cases}$$

a	b	y
0	0	1
0	1	1
1	0	0
1	1	1

K-map:

a \ b	0	1
0	1	1
1	0	1

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. **Sum the products**

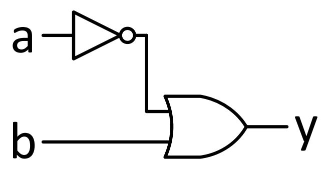
Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #4
    - $f_4(a,b) = \boldsymbol{\overline{a}} + \boldsymbol{b}$
    - Truth table
    - KM

a	b	y
0	0	1
0	1	1
1	0	0
1	1	1

		b	
		0	1
a	0	1	1
	1	0	1

Logic circuit



Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #5
    - $f_5(a,b) = ??$
    - Truth table

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

    - KM

a \ bc	00	01	11	10
0	0	1	1	0
1	0	0	1	0

Gray code: only one bit must change between two adjacent values

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #5
    - $f_5(a, b) = ??$
    - Truth table
    - KM

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

bc

a	00	01	11	10
0	0	1	1	0
1	0	0	1	0

- How to
  1. **Find all the largest group(s) of contiguous 1s**
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #5
    - $f_5(a, b) = ??$
    - Truth table
    - KM

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

a \ bc	00	01	11	10
0	0	1	1	0
1	0	0	1	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. **For each group, select only the inputs whose value does not change**
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #5
    - $f_5(a,b) = ??$
    - Truth table
    - KM

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$\bar{a}$

a \ bc	00	01	11	10
0	0	1	1	0
1	0	0	1	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. **Complement surviving inputs whose value is 0**
  4. Multiply the surviving inputs → products P_i
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #5
    - $f_5(a, b) = ??$
    - Truth table
    - KM

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$P_0 = \bar{a} \cdot c$

$P_1 = b \cdot c$

a \ bc	00	01	11	10
0 ($\bar{a}$)	0	1	1	0
1	0	0	1	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. **Multiply the surviving inputs → products P_i**
  5. Sum the products

Principles of programmable logic

- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #5
    - $f_5(a,b) = (\boldsymbol{\bar{a}} \cdot \boldsymbol{c}) + (\boldsymbol{b} \cdot \boldsymbol{c})$
    - Truth table →
    - KM

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$P_0 = \bar{a} \cdot c$

$P_1 = b \cdot c$

a \ bc	00	01	11	10
0	0	1	1	0
1	0	0	1	0

- How to
  1. Find all the largest group(s) of contiguous 1s
  2. For each group, select only the inputs whose value does not change
  3. Complement surviving inputs whose value is 0
  4. Multiply the surviving inputs → products P_i
  5. **Sum the products**

Principles of programmable logic

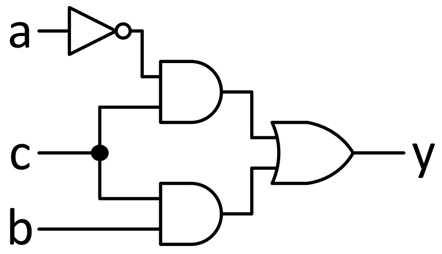
- Example(s)
  - **Karnaugh map** (**KM** or **K-map**) = graphical method to express logic functions as SP or PS
    - Showing SP case
  - Example #5
    - $f_5(a,b) = (\bar{\boldsymbol{a}} \cdot \boldsymbol{c}) + (\boldsymbol{b} \cdot \boldsymbol{c})$
    - Truth table →
    - KM

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

bc

a	00	01	11	10
0	0	1	1	0
1	0	0	1	0

Logic circuit



Principles of programmable logic

- Example(s)
  - Karnaugh map (KM or K-map) = graphical method to express logic functions as SP or PS
    - Showing SP case
    - Similar approach can be used for PS
    - However, the goal is not to learn how to use Karnaugh maps, but …

Principles of programmable logic

- Example(s)
  - Karnaugh map (KM or K-map) = graphical method to express logic functions as SP or PS
    - Showing SP case
    - Similar approach can be used for PS
    - However, the goal is not to learn how to use Karnaugh maps, but …

- … demonstrating that

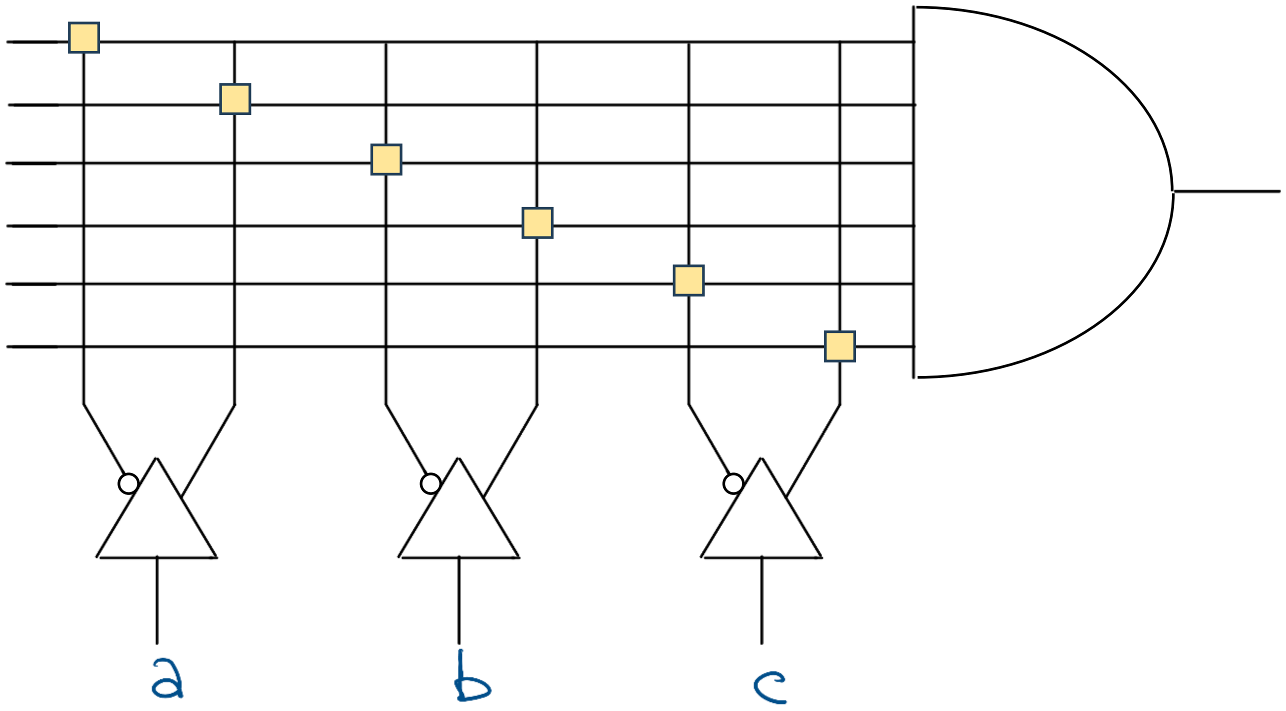
any logic function can be implemented as
a network of AND and OR (and NOT) gates!

PAL

- Earliest PLDs were called **P**rogrammable **A**rray **L**ogic (**PAL**)
  - Programmable network of OR, AND, and NOT gates
    - Pre-defined network
    - Programmable connections among gates
  - PROM to program (and re-program) connections

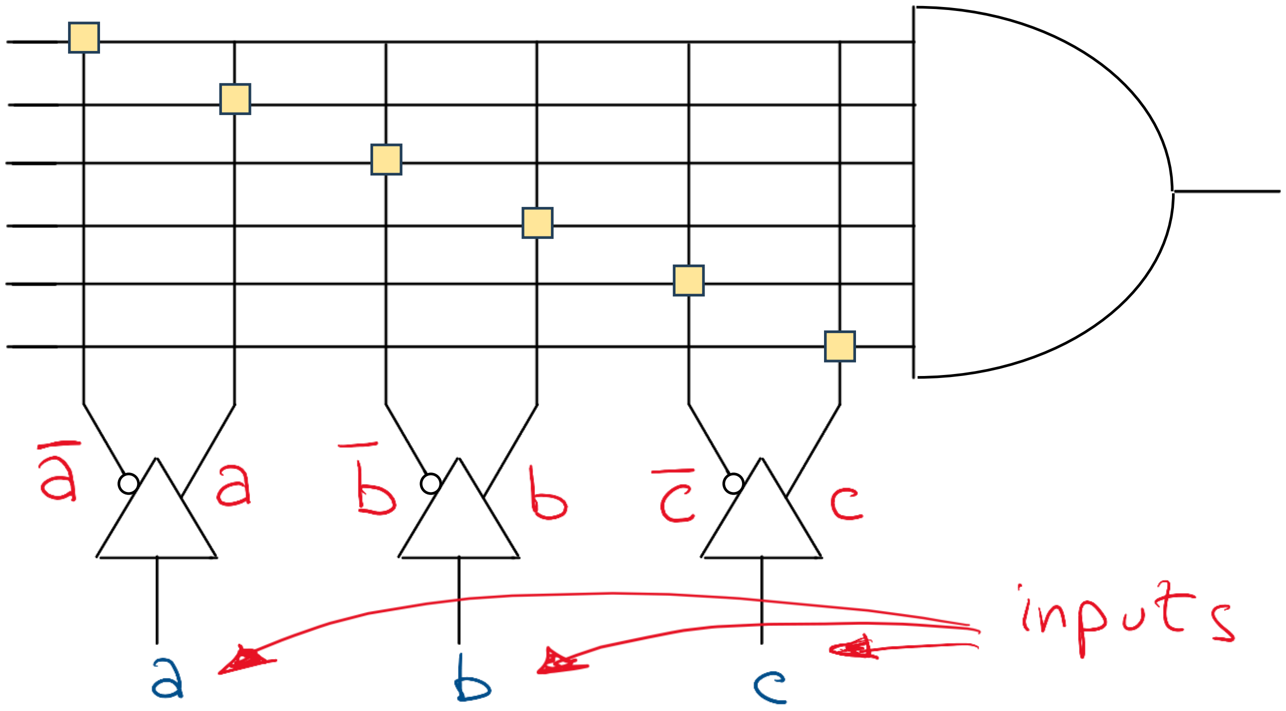
PAL

- Architecture outline



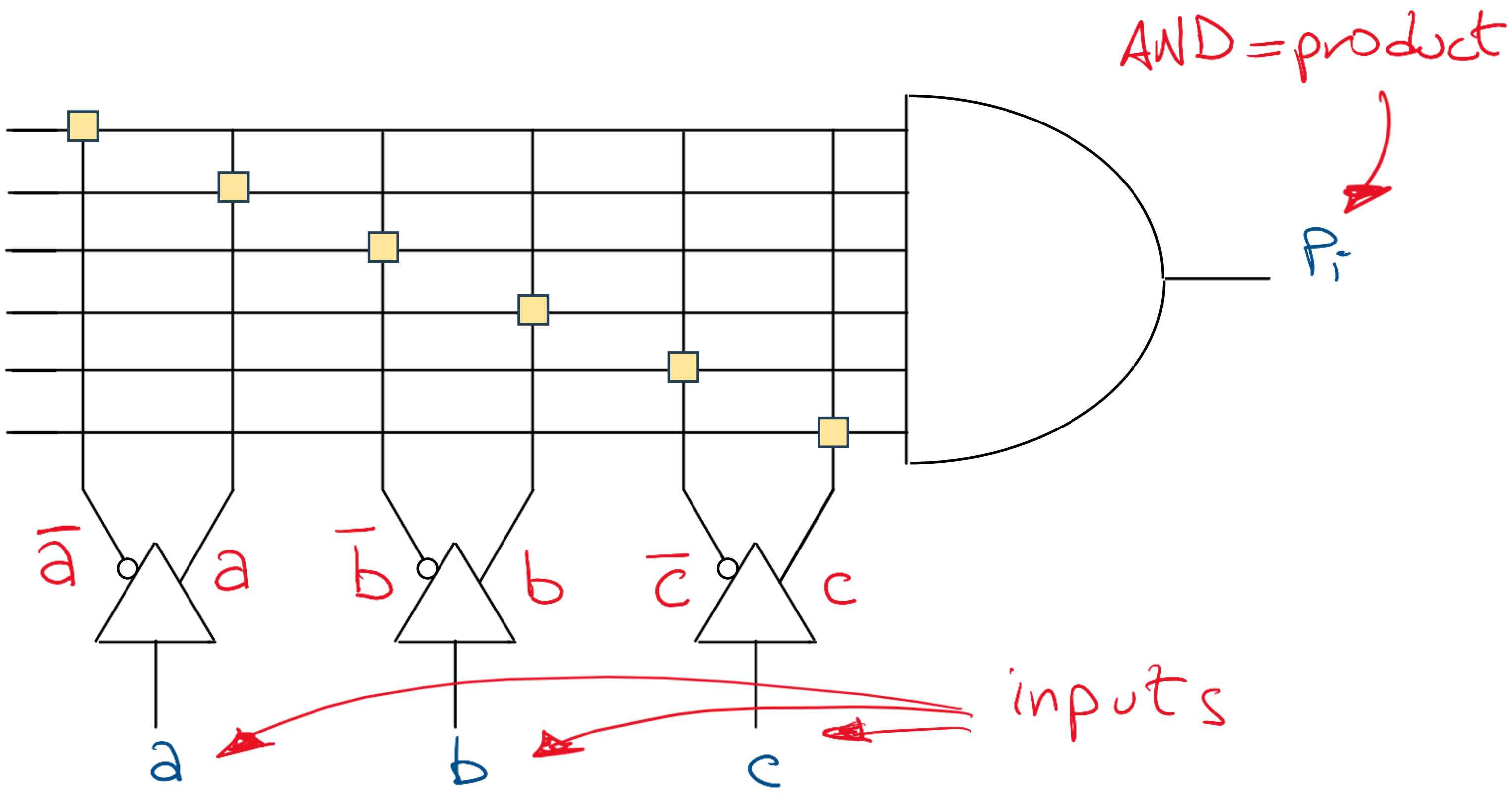
PAL

- Architecture outline



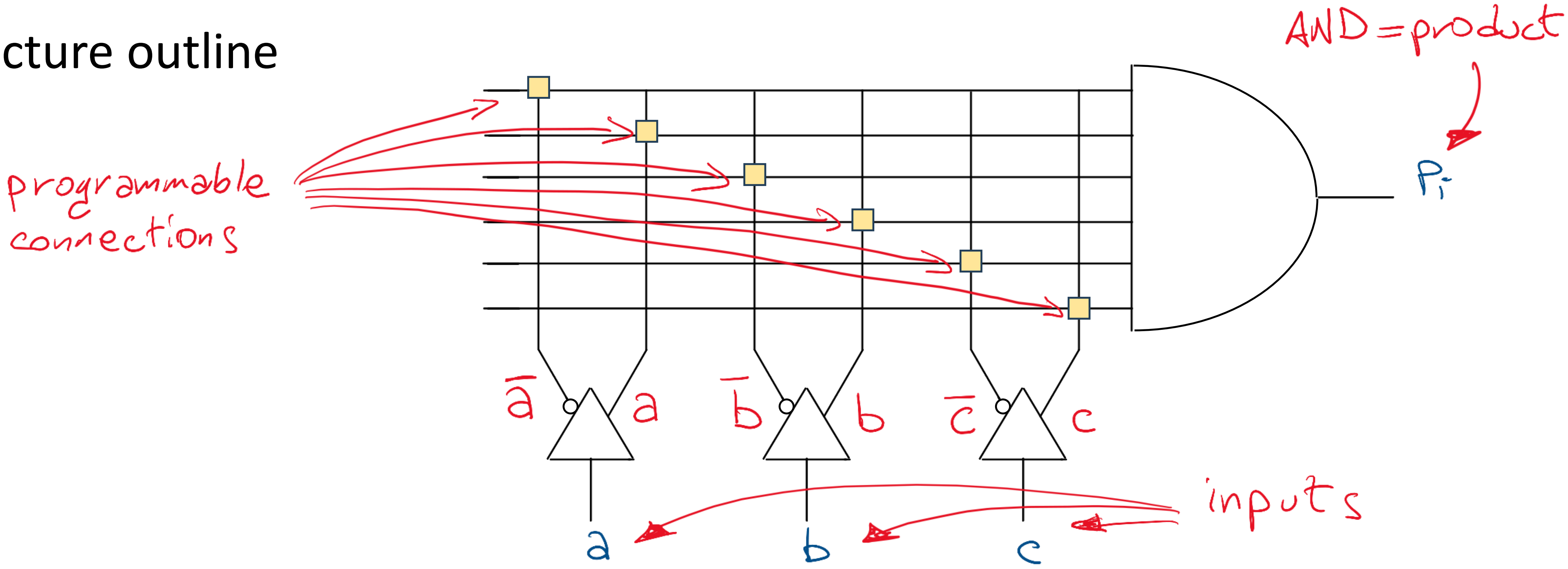
PAL

- Architecture outline



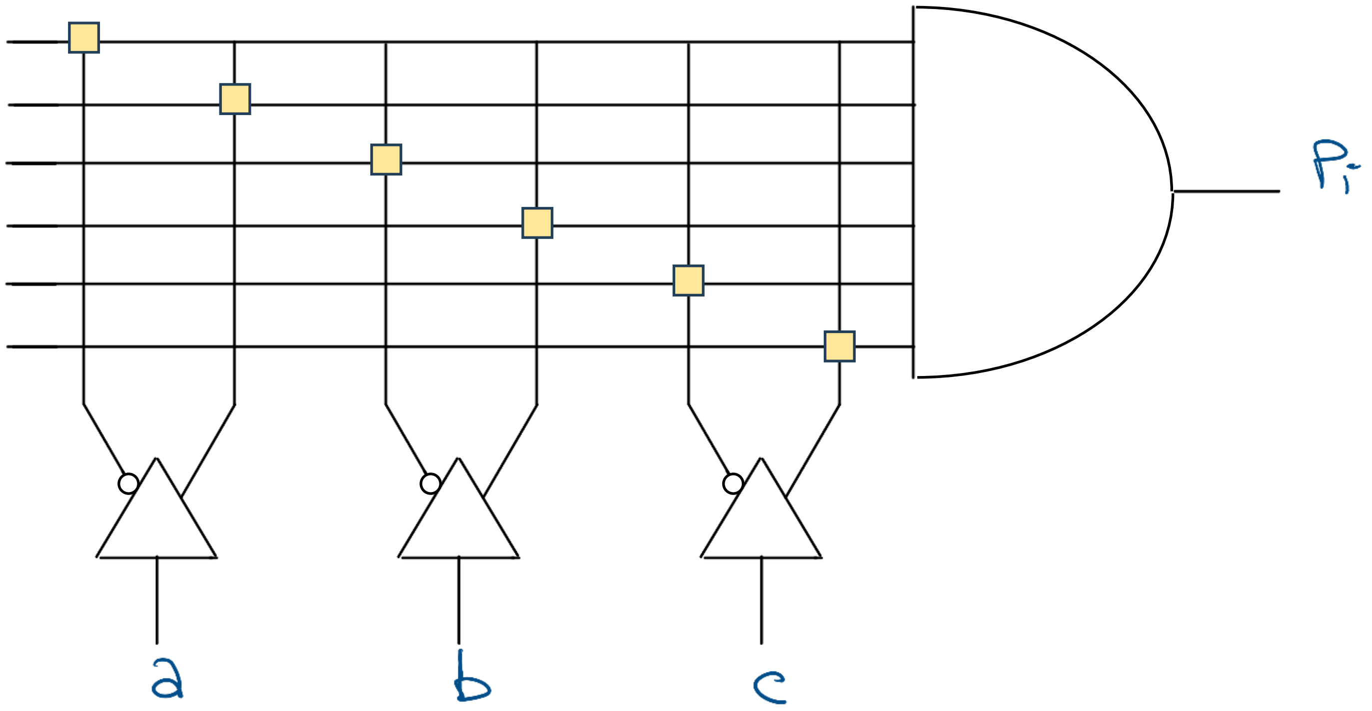
PAL

- Architecture outline



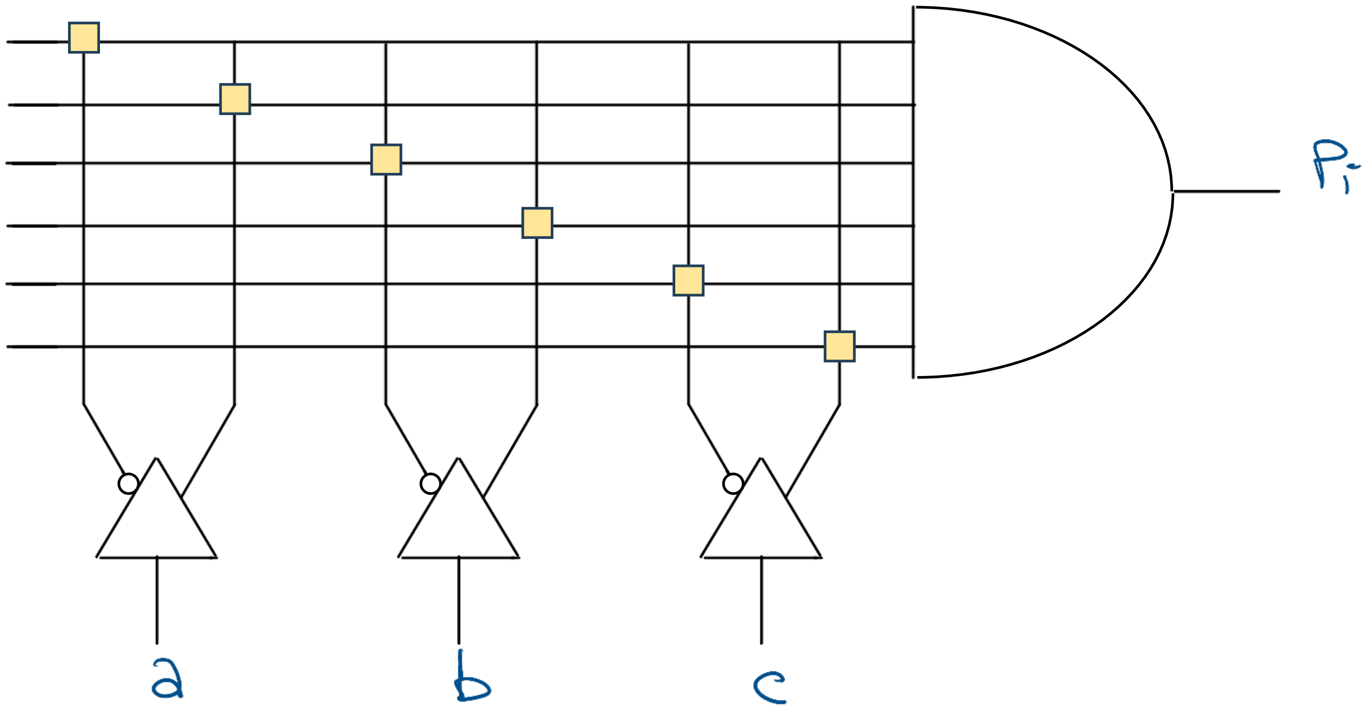
PAL

- Architecture outline
  - Do you see familiarity with previous examples?



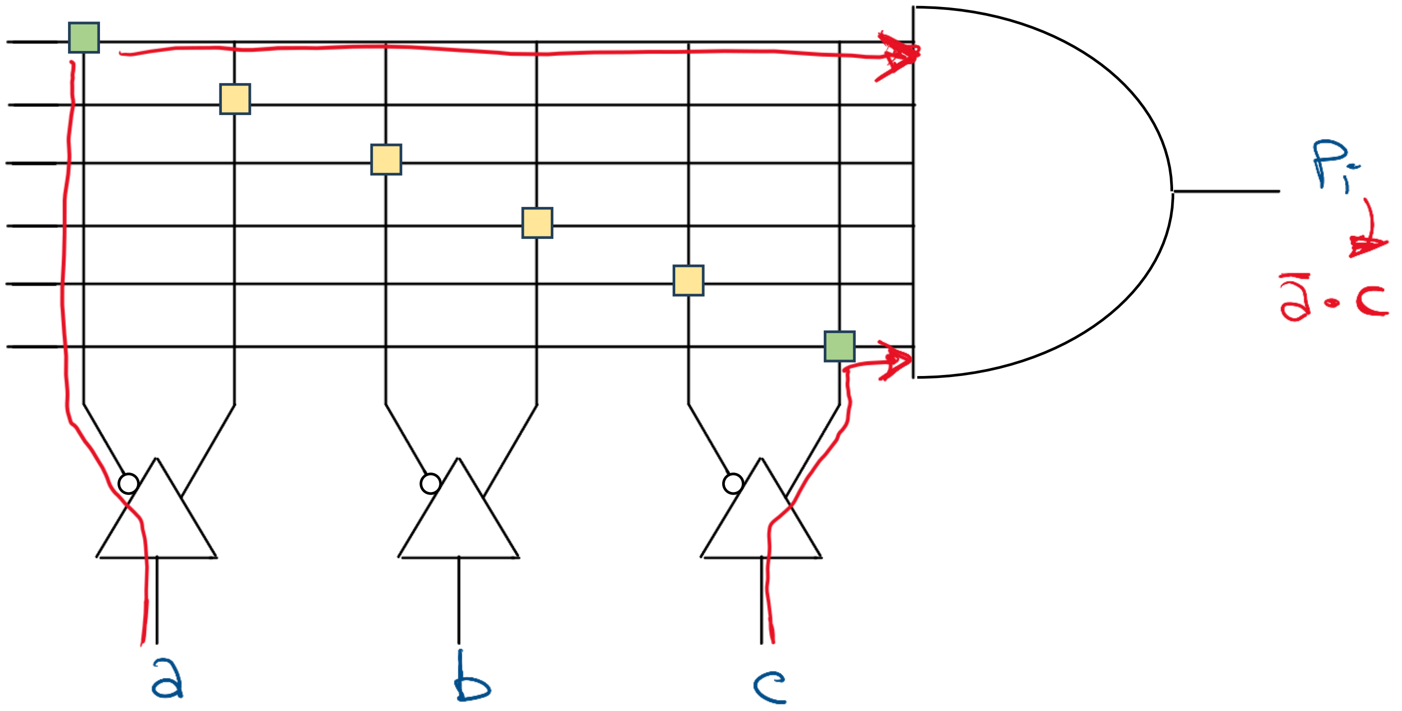
PAL

- Architecture outline
  - Do you see familiarity with previous examples?
    - Recalling example #5
      - $f_5(a, b) = (\boldsymbol{\bar{a} \cdot c}) + (\boldsymbol{b \cdot c})$



PAL

- Architecture outline
  - Do you see familiarity with previous examples?
    - Recalling example #5
      - $f_5(a, b) = (\bar{a} \cdot c) + (\boldsymbol{b} \cdot \boldsymbol{c})$

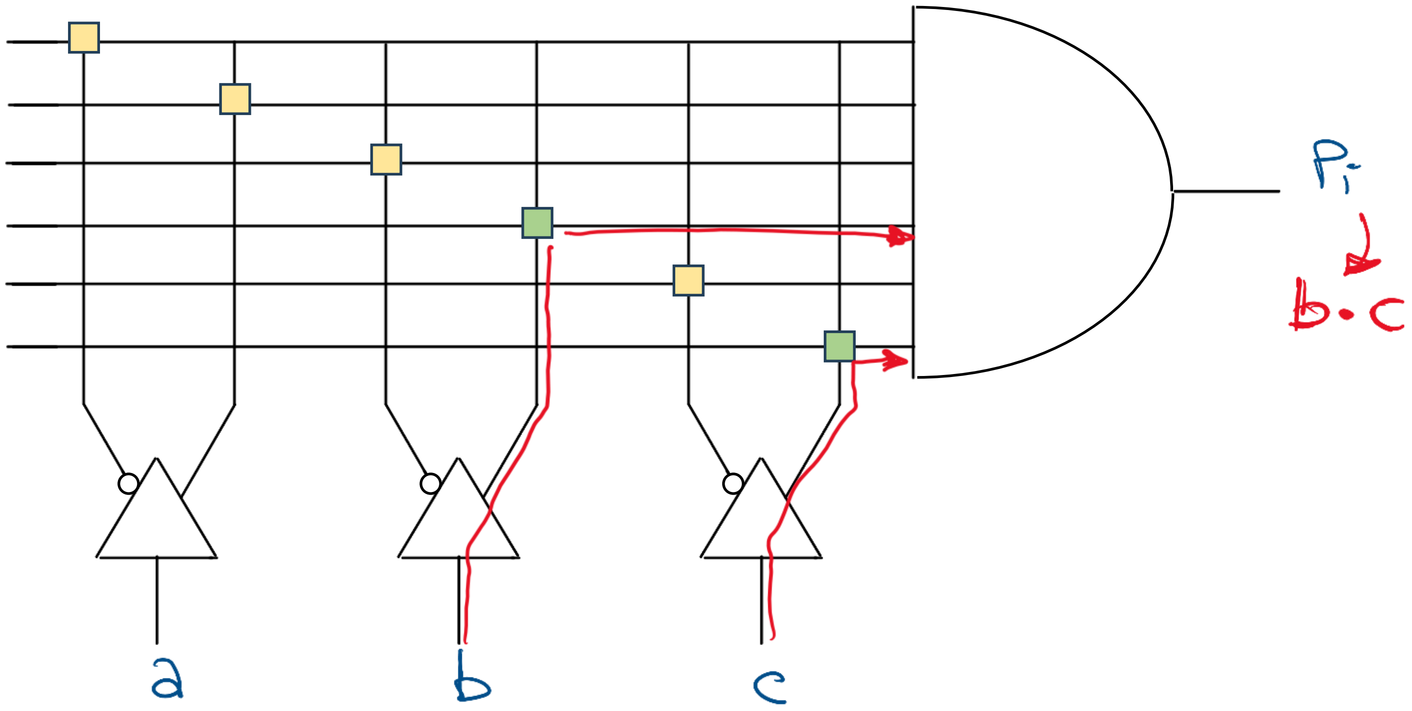


= non-programmed node (open circuit)

= programmed node (short circuit)

PAL

- Architecture outline
  - Do you see familiarity with previous examples?
    - Recalling example #5
      - $f_5(a,b) = (\bar{\boldsymbol{a}} \cdot \boldsymbol{c}) + (\boldsymbol{b} \cdot \boldsymbol{c})$

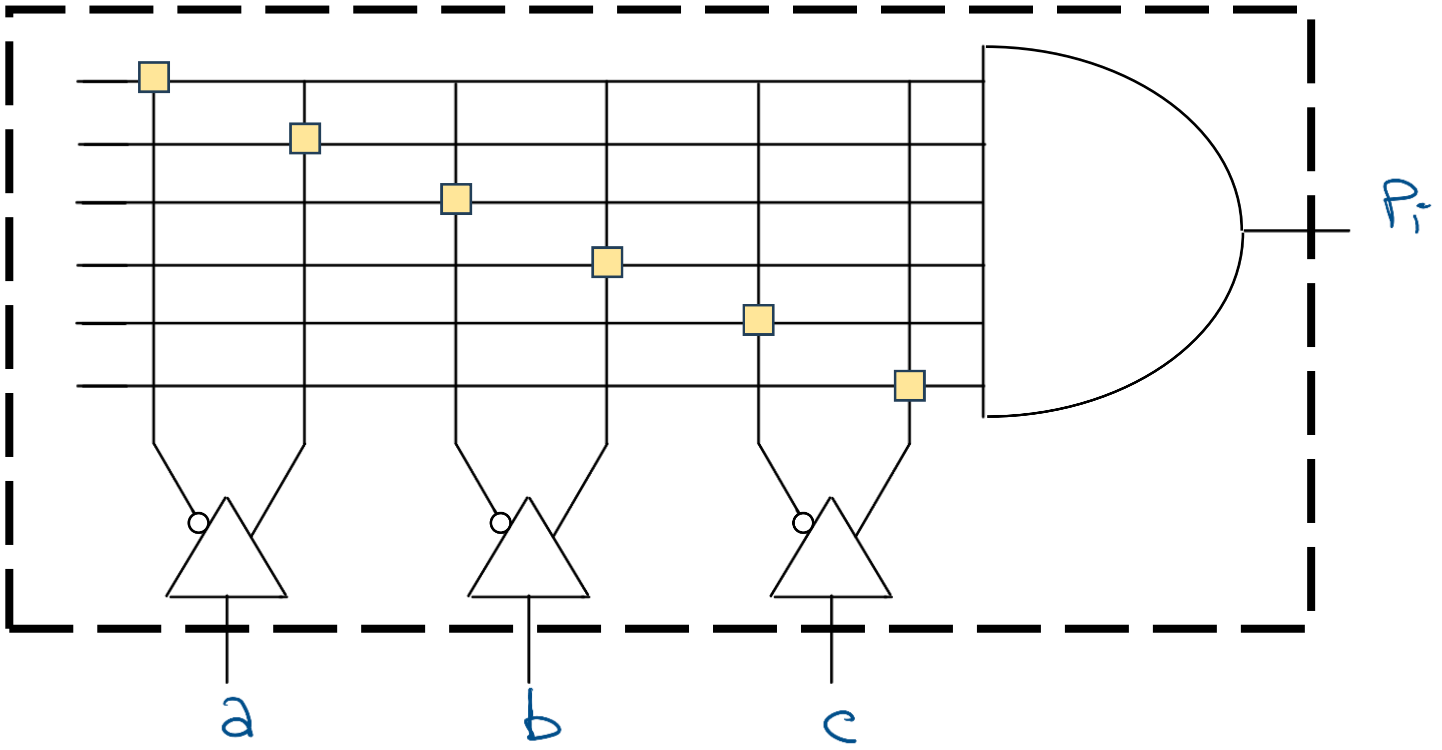


= non-programmed node (open circuit)

= programmed node (short circuit)

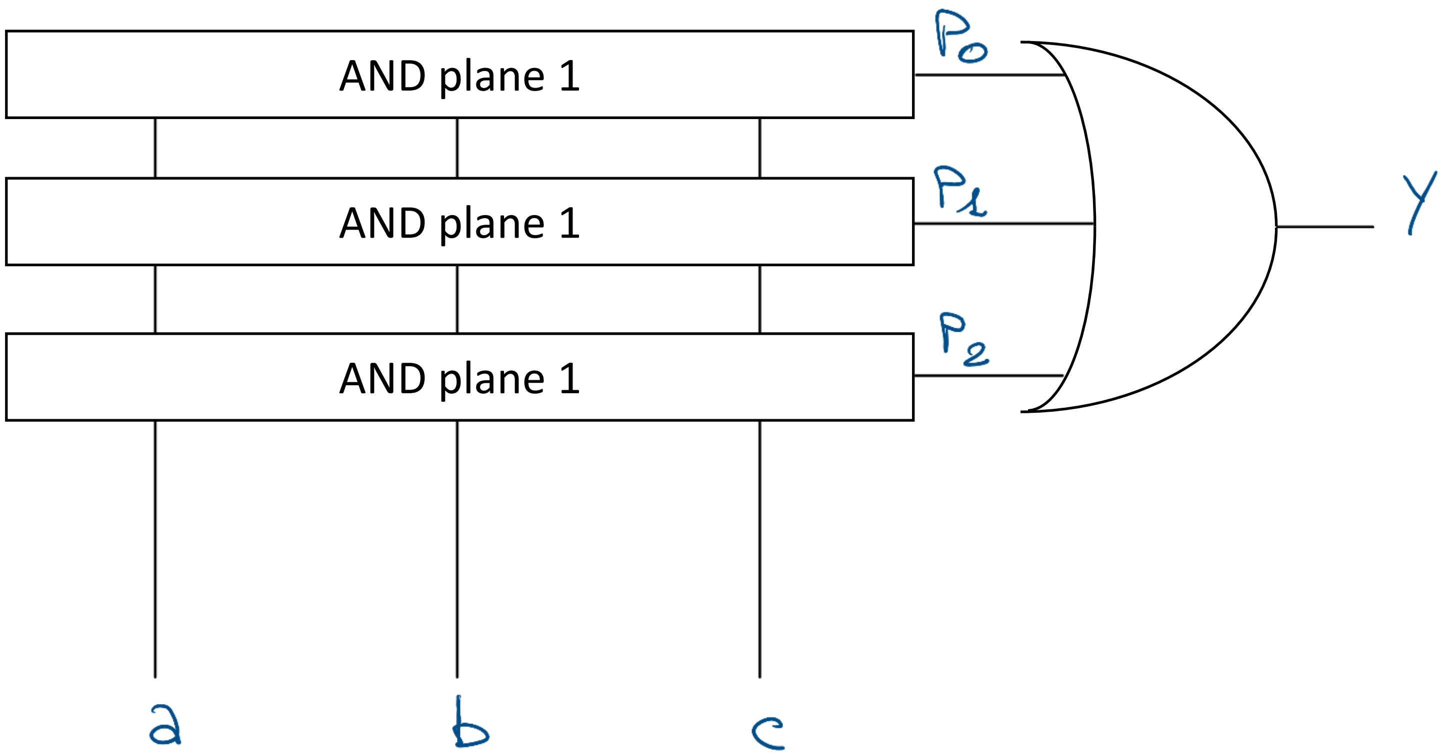
PAL

- Architecture outline
  - **AND plane**



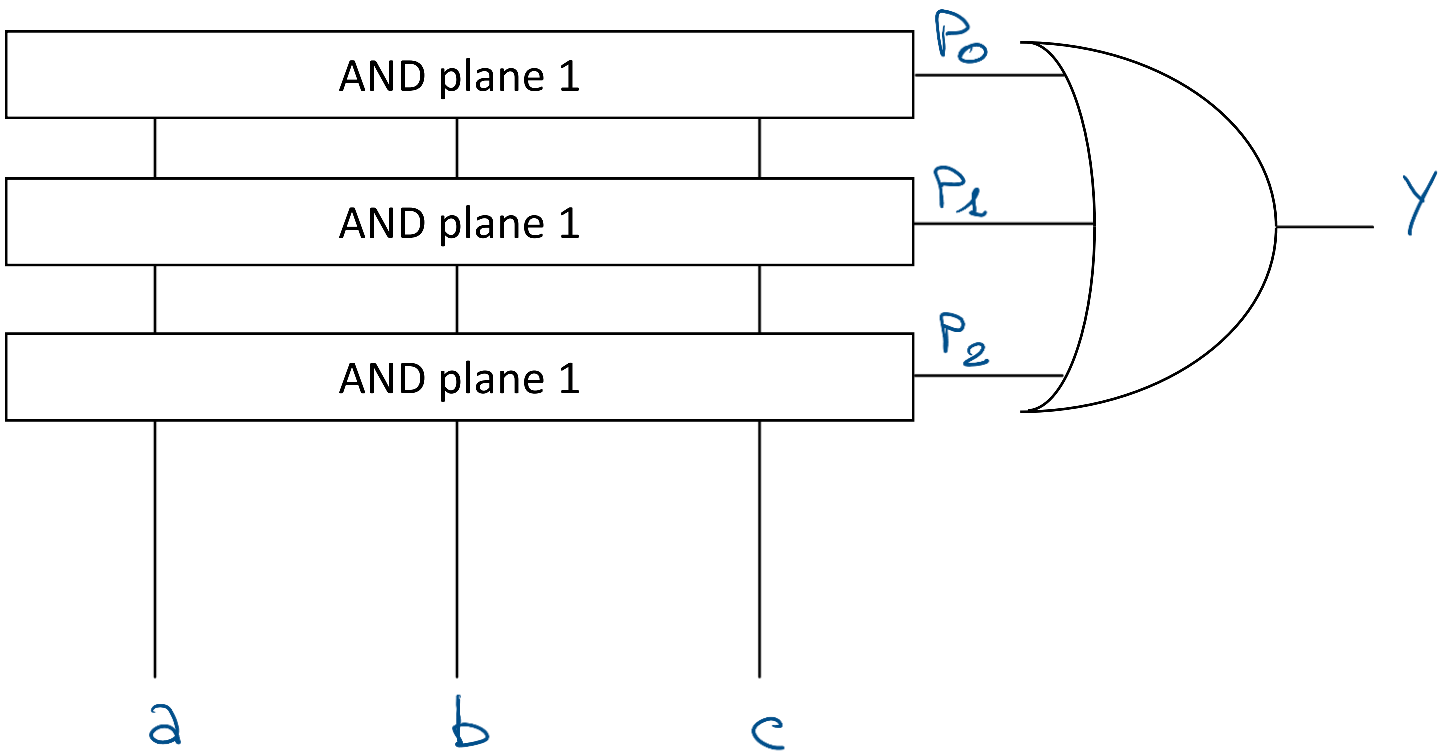
PAL

- Architecture outline



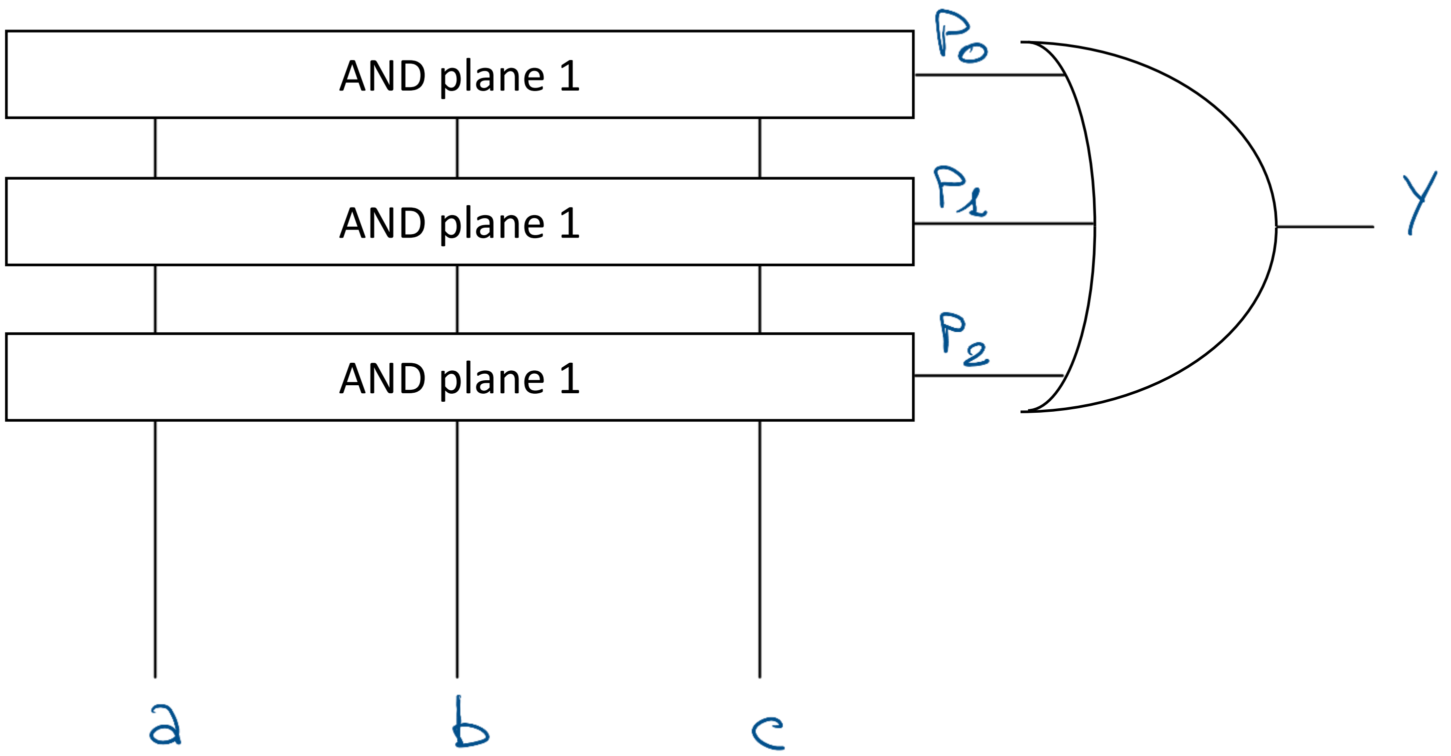
PAL

- Architecture outline
  - Do you see familiarity with previous examples?



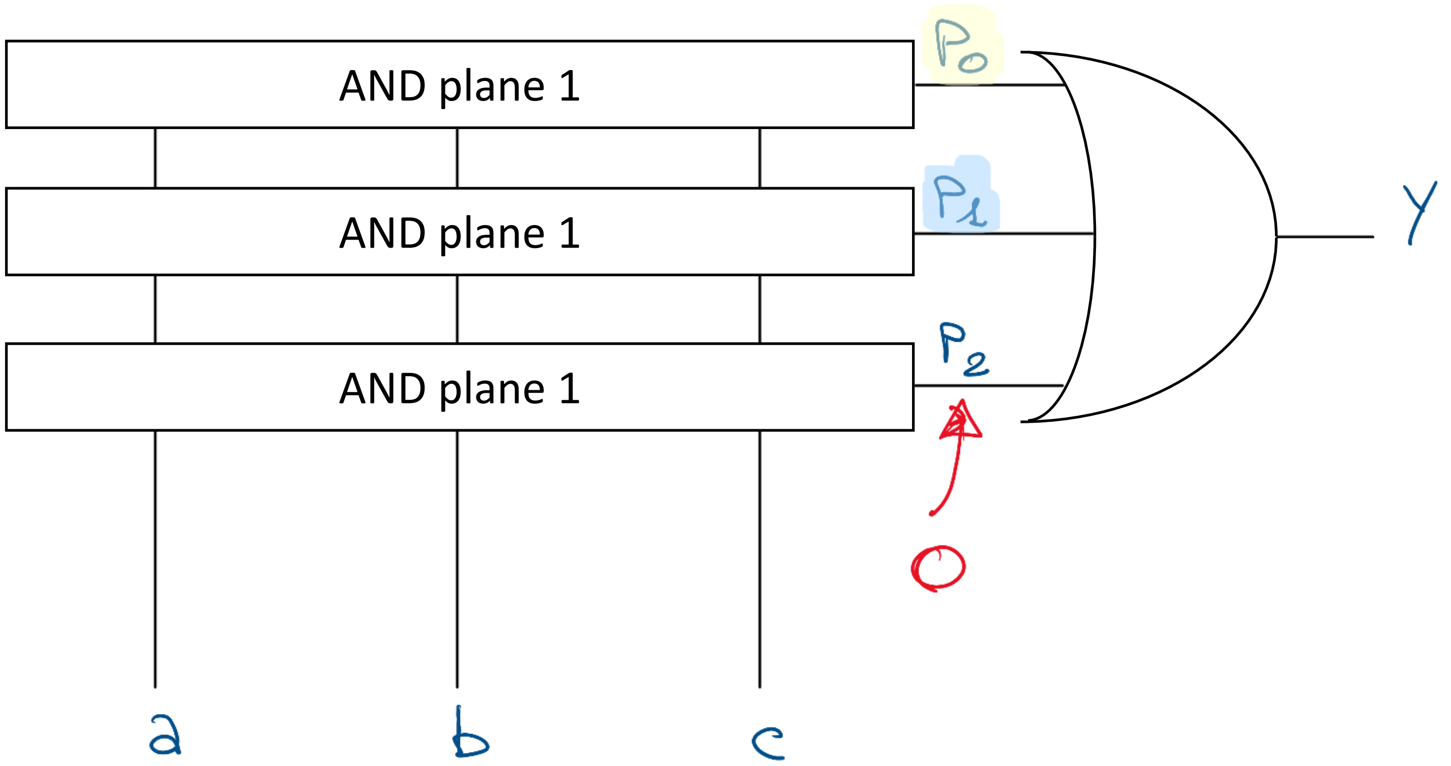
PAL

- Architecture outline
  - Do you see familiarity with previous examples?
    - Recalling example #5
      - $f_5(a,b) = (\boldsymbol{\bar{a}} \cdot \boldsymbol{c}) + (\boldsymbol{b} \cdot \boldsymbol{c})$



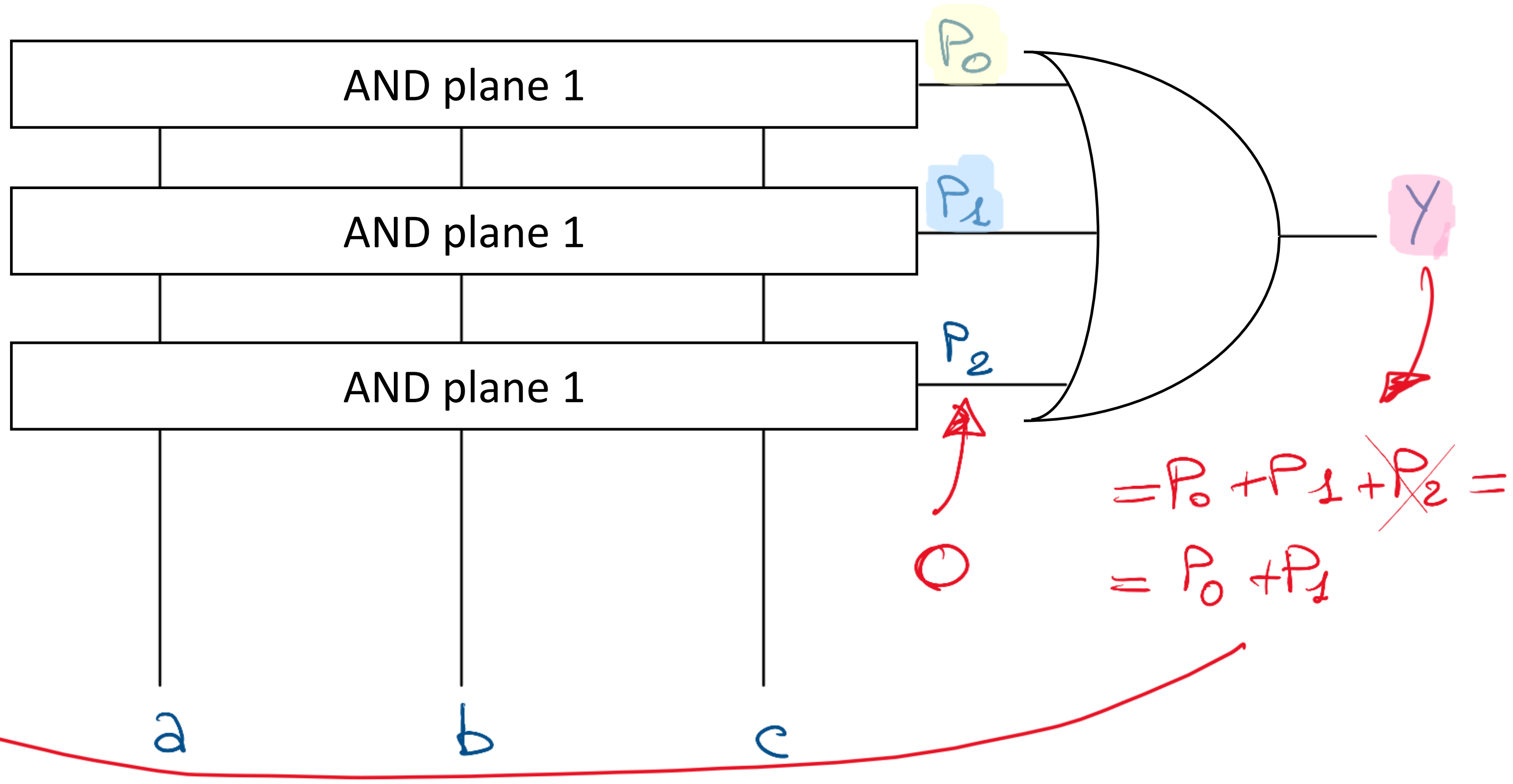
PAL

- Architecture outline
  - Do you see familiarity with previous examples?
    - Recalling example #5
      - $f_5(a,b) = \underbrace{(\overline{a} \cdot c)}_{P_0} + \underbrace{(b \cdot c)}_{P_1}$



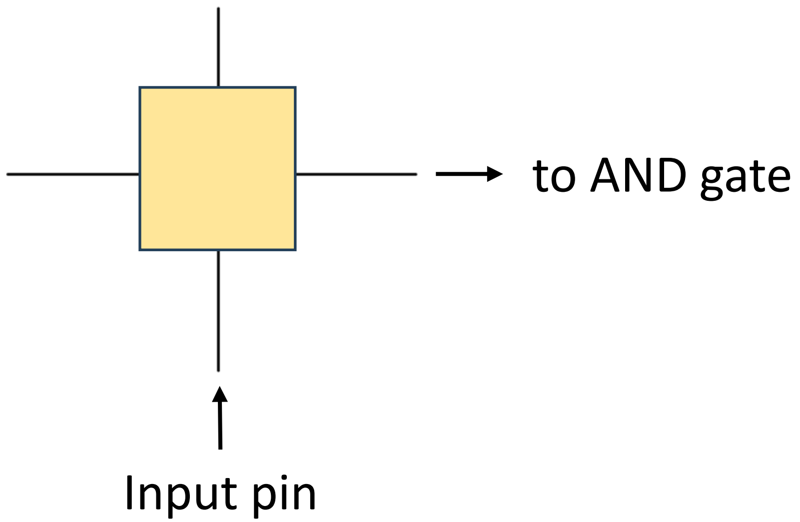
PAL

- Architecture outline
  - Do you see familiarity with previous examples?
    - Recalling example #5
      - $f_5(a,b) = \underbrace{(\bar{a} \cdot c)}_{P_0} + \underbrace{(b \cdot c)}_{P_1}$



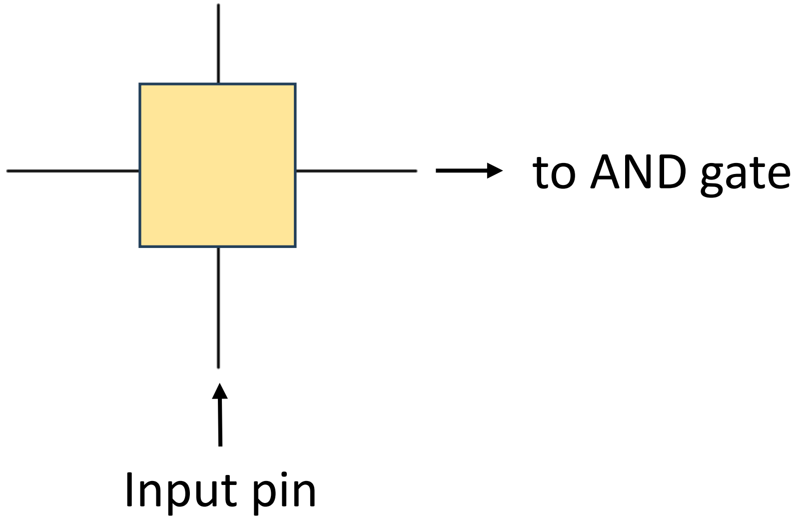
PAL

- Programmable connection(s)
  - Connecting input to AND gate
  - Determining the type of PAL/PLD
    - **OTP** = **O**ne-**T**ime **P**rogrammable
      - Fuse/Anti-fuse
    - **Reprogrammable**
      - UV-EPROM style (FAMOS)
      - Flash memory style (Flash transistor)



PAL

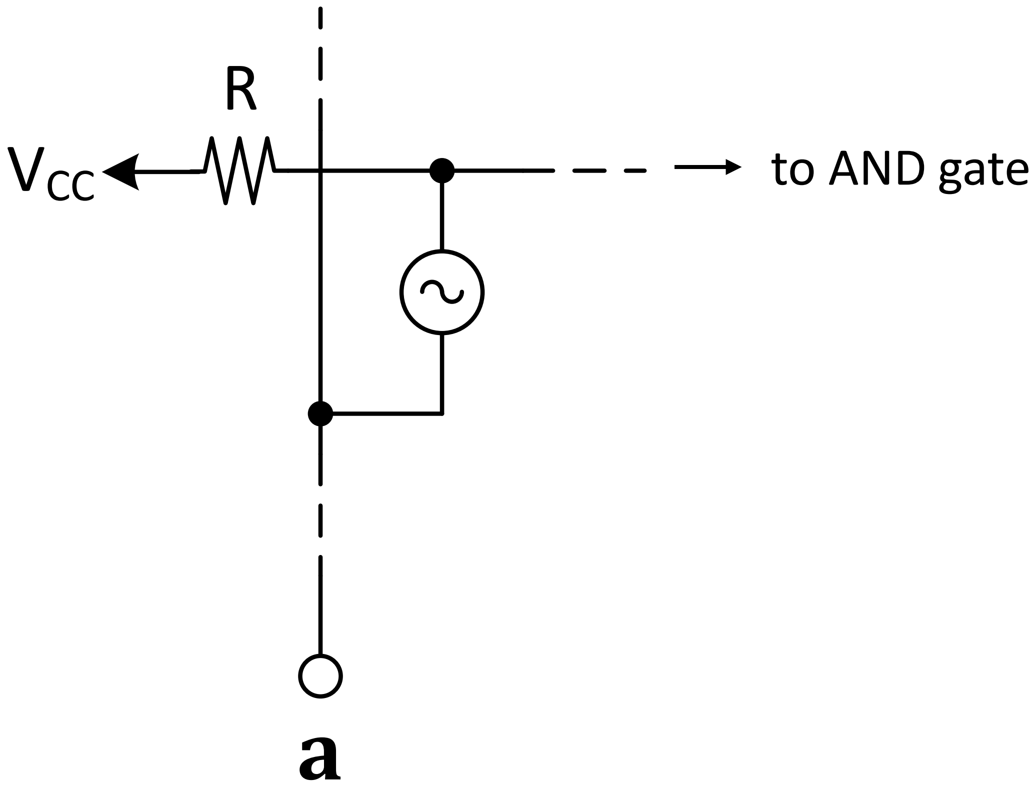
- Programmable connection(s)
  - Connecting input to AND gate
  - Determining the type of PAL/PLD
    - **OTP** = **O**ne-**T**ime **P**rogrammable
      - Fuse/Anti-fuse
    - **Reprogrammable**
      - UV-EPROM style (FAMOS)
      - Flash memory style (Flash transistor)



Similar to memory (ROM):
- Word Line = input pin
- Bit Line = AND gate input

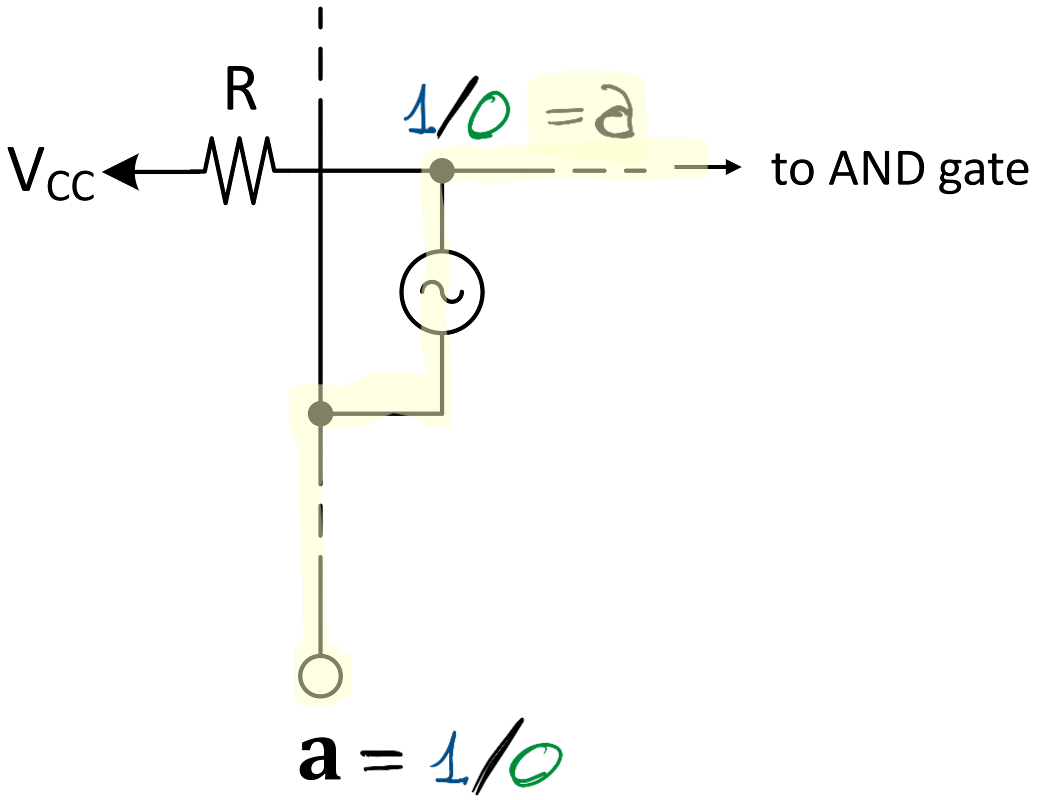
PAL

- OTP PAL
  - **Fuse**



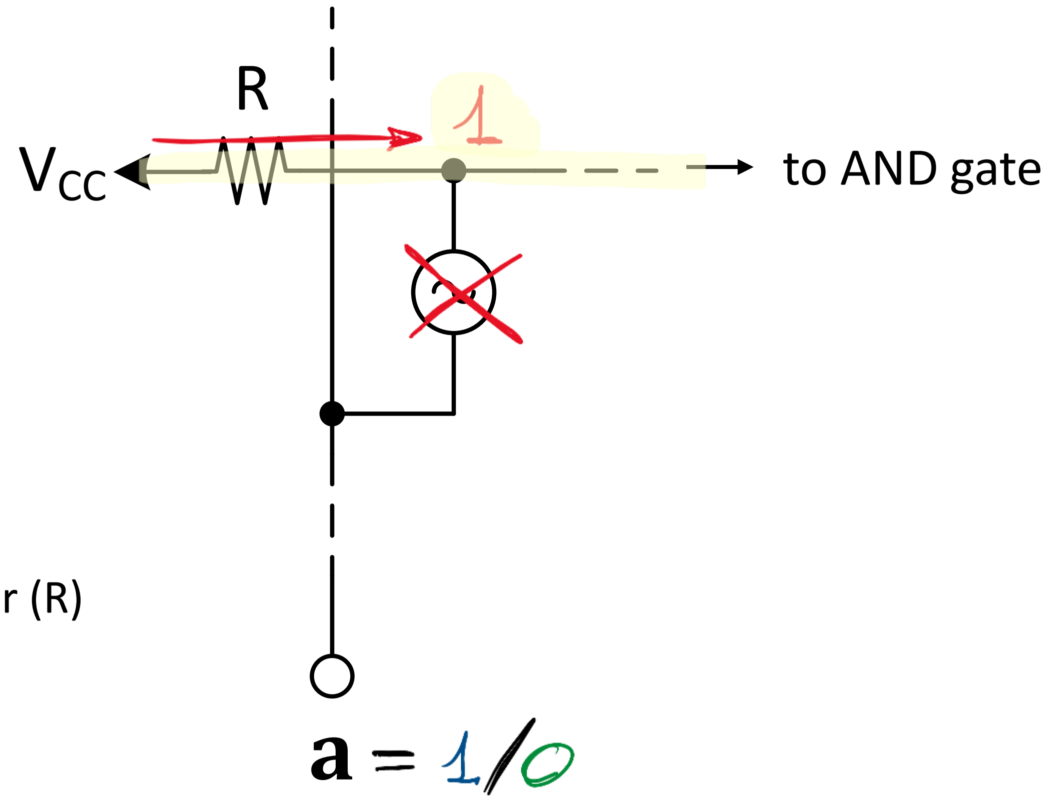
PAL

- OTP PAL
  - **Fuse**
    - Default (Fuse not vaporized)
      - Input pin (**a**) connected to the AND gate



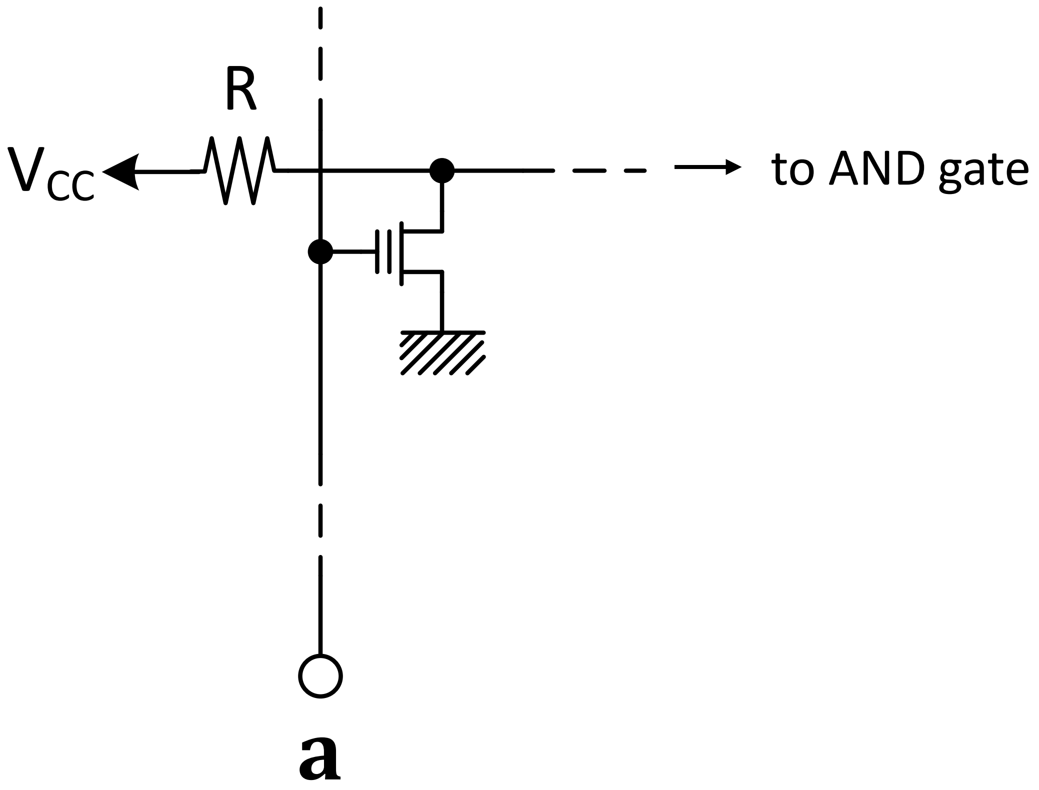
PAL

- OTP PAL
  - **Fuse**
    - Default (Fuse not vaporized)
      - Input pin (**a**) connected to the AND gate
    - Fuse vaporized (after programming)
      - Link to the input pin broken
      - AND gate input connected only to the pull-up resistor (R)
        - **Always logic 1**
        - **Not altering the AND gate functionality!!!**



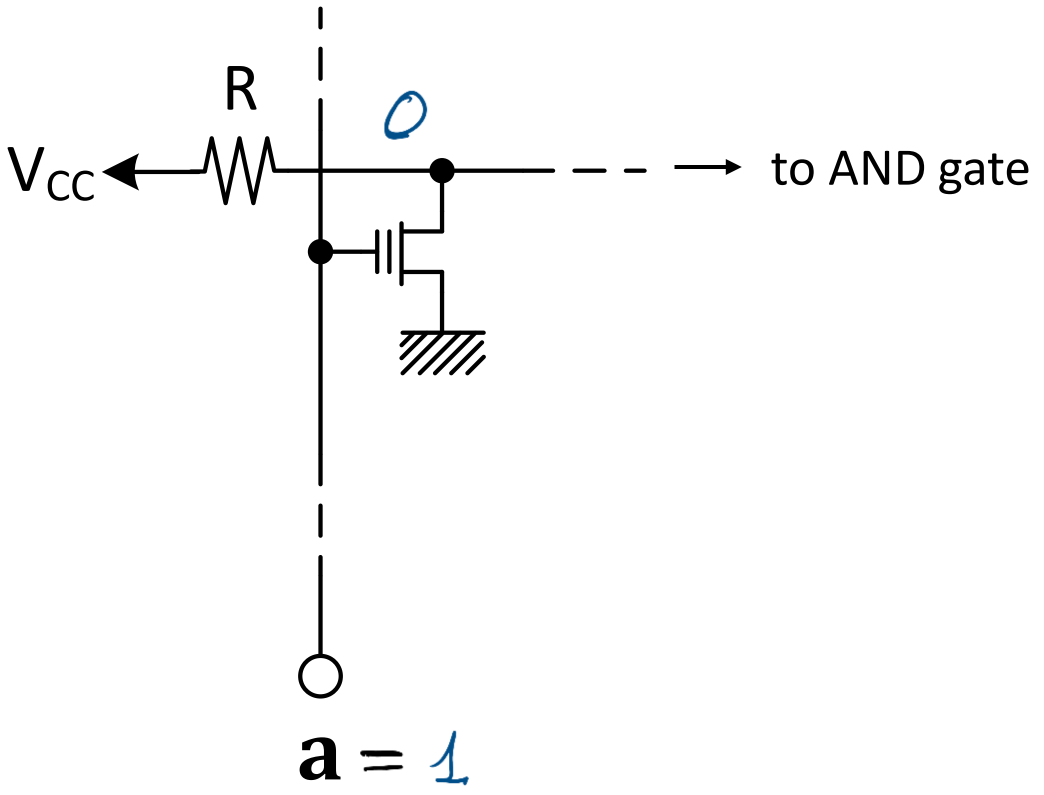
PAL

- Reprogrammable PAL
  - **UV erasable** (**FAMOS**)



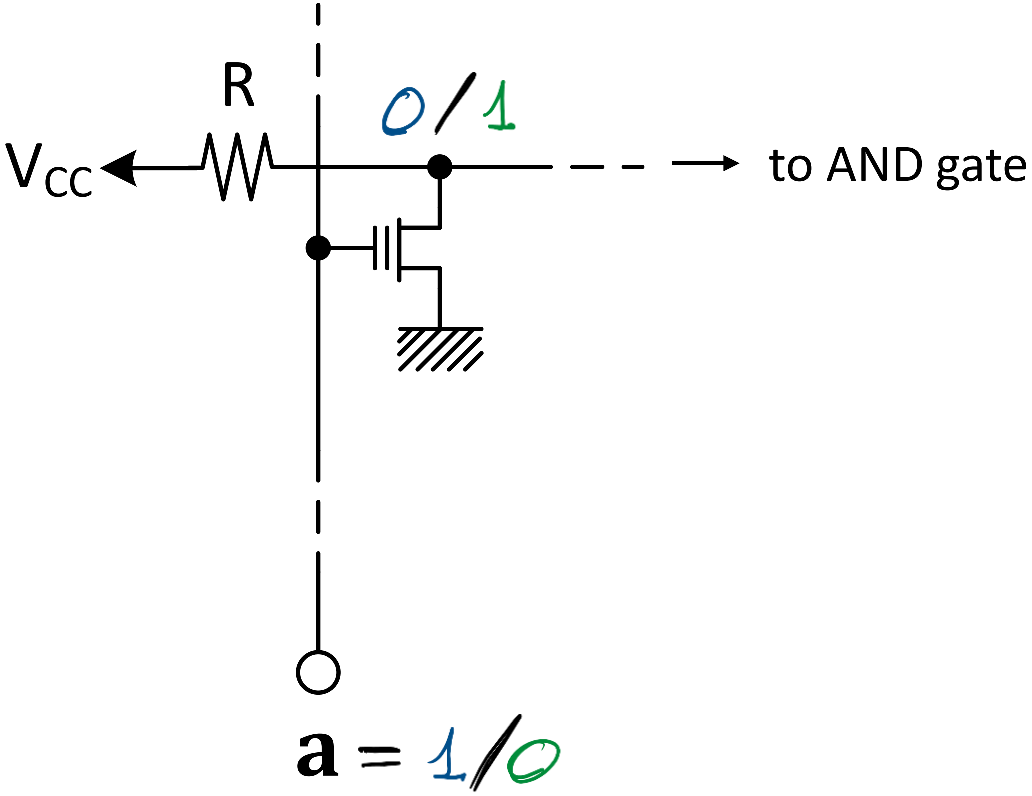
PAL

- Reprogrammable PAL
  - **UV erasable (FAMOS)**
    - Default (FAMOS not programmed)
      - If input pin (**a**) = logic 1 ($V_{CC} > V_T$) → transistor **ON**
        - AND gate input = logic 0



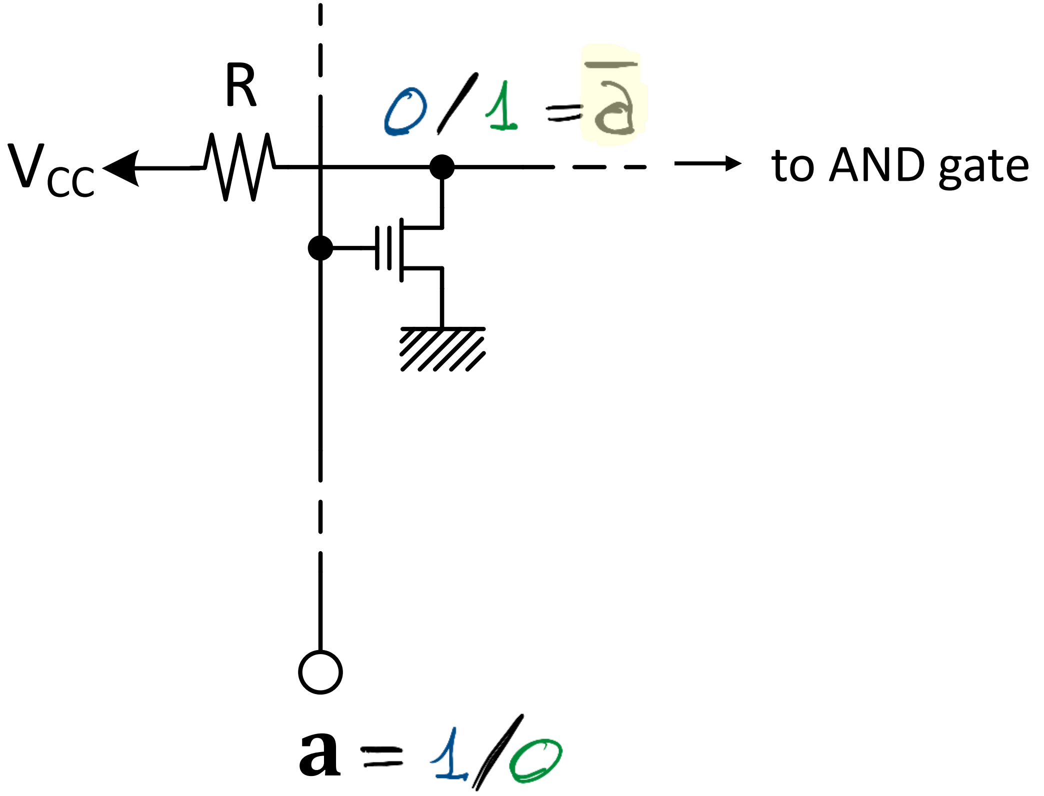
PAL

- Reprogrammable PAL
  - **UV erasable (FAMOS)**
    - Default (FAMOS not programmed)
      - If input pin (**a**) = logic 1 ($V_{CC} > V_T$) → transistor **ON**
        - AND gate input = logic 0
      - If input pin (**a**) = logic 1 ($0\ V < V_T$) → transistor **OFF**
        - AND gate input = logic 1



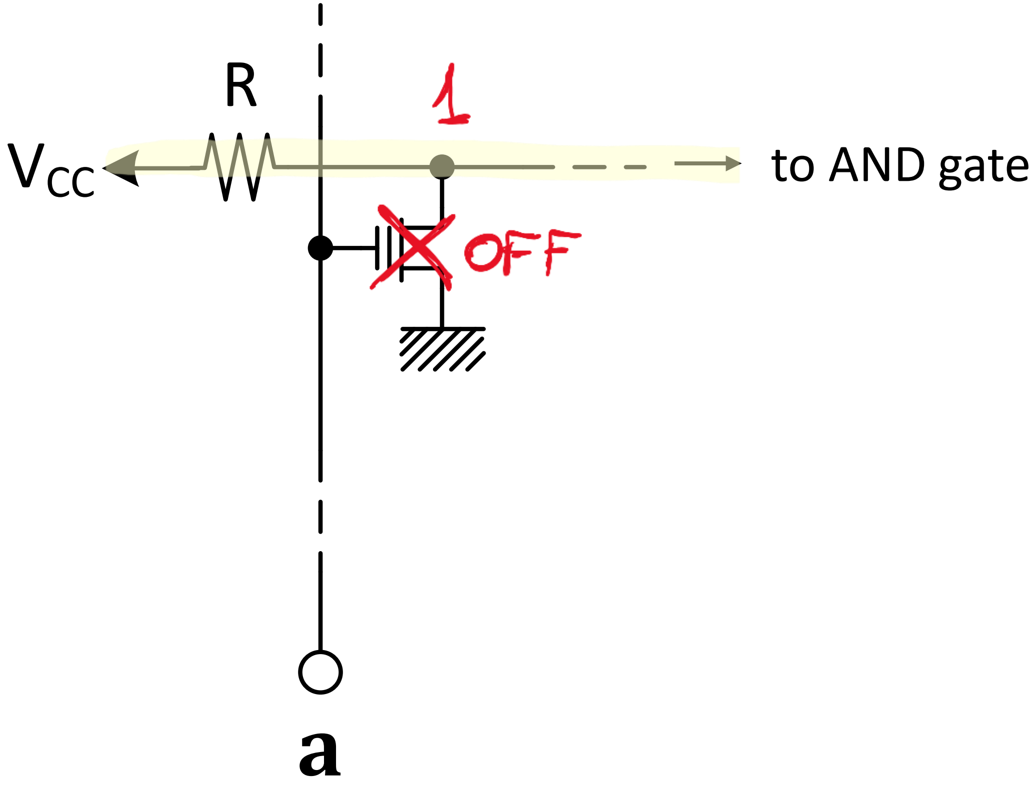
PAL

- Reprogrammable PAL
  - **UV erasable (FAMOS)**
    - Default (FAMOS not programmed)
      - If input pin (**a**) = logic 1 ($V_{CC} > V_T$) → transistor **ON**
        - AND gate input = logic 0
      - If input pin (**a**) = logic 1 ($0\ V < V_T$) → transistor **OFF**
        - AND gate input = logic 1
      - **Attention! Inverting logic / active-low logic!!!**



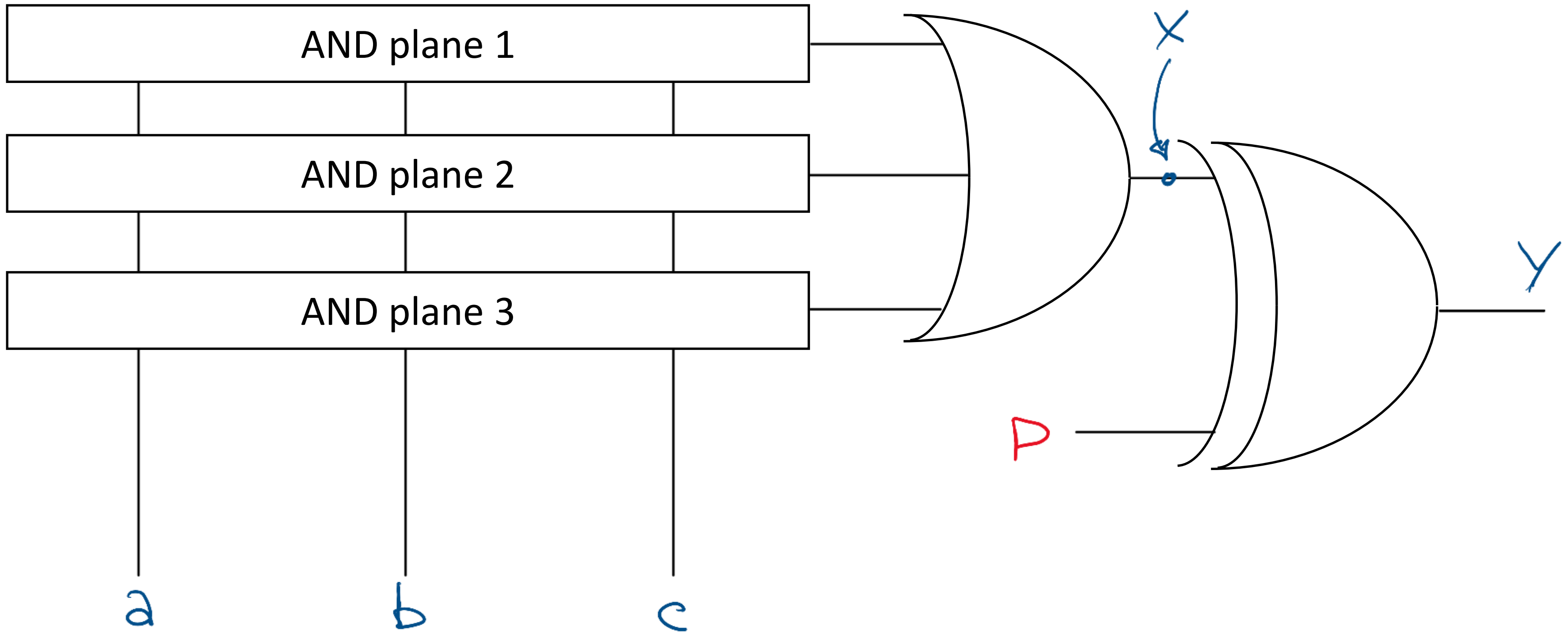
PAL

- Reprogrammable PAL
  - **UV erasable (FAMOS)**
    - Default (FAMOS not programmed)
      - If input pin (**a**) = logic 1 ($V_{CC} > V_T$) → transistor **ON**
        - AND gate input = logic 0
      - If input pin (**a**) = logic 1 ($0\ V < V_T$) → transistor **OFF**
        - AND gate input = logic 1
      - **Attention! Inverting logic / active-low logic!!!**
    - FAMOS programmed ($V_T' > V_{CC}$)
      - Transistor always OFF
      - **AND gate input always logic 1** (through pull-up resistor to V_{CC})
        - **Not altering AND gate functionality!!!**



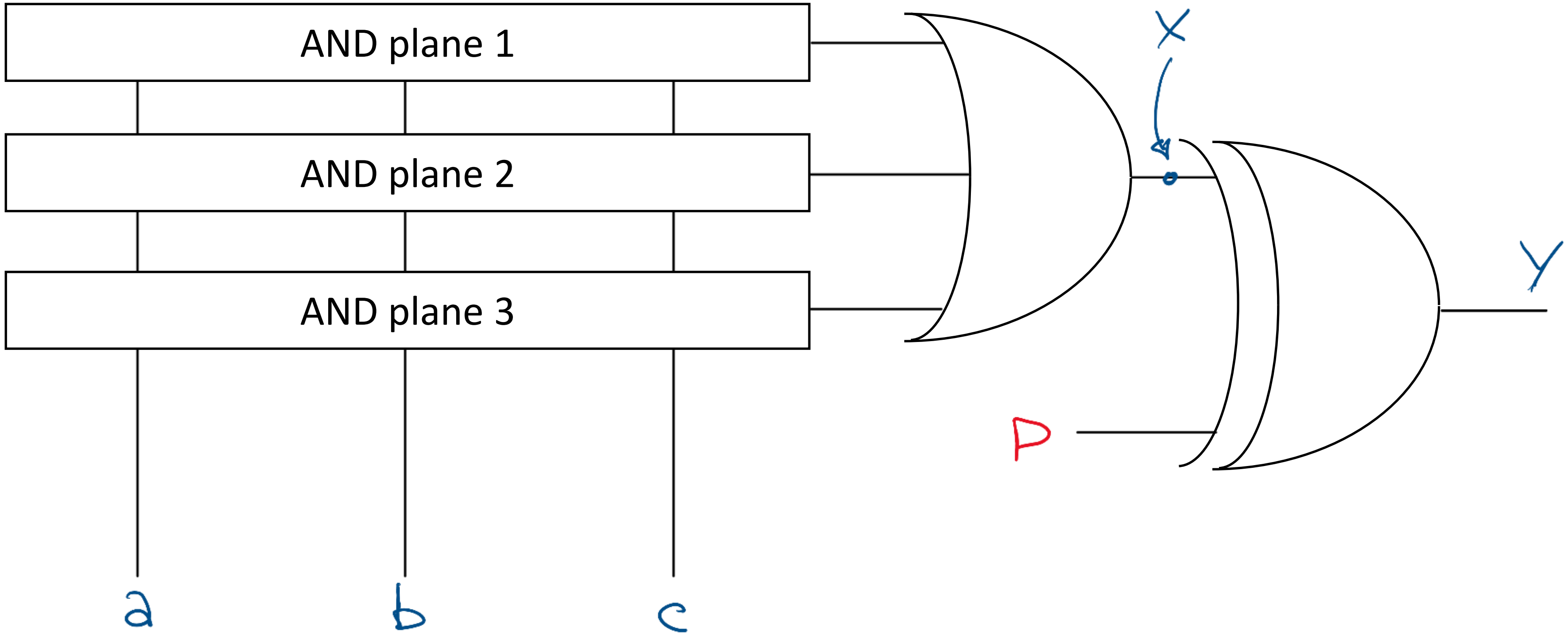
Evolution of PAL

- Output polarity configuration: active-high / active-low



Evolution of PAL

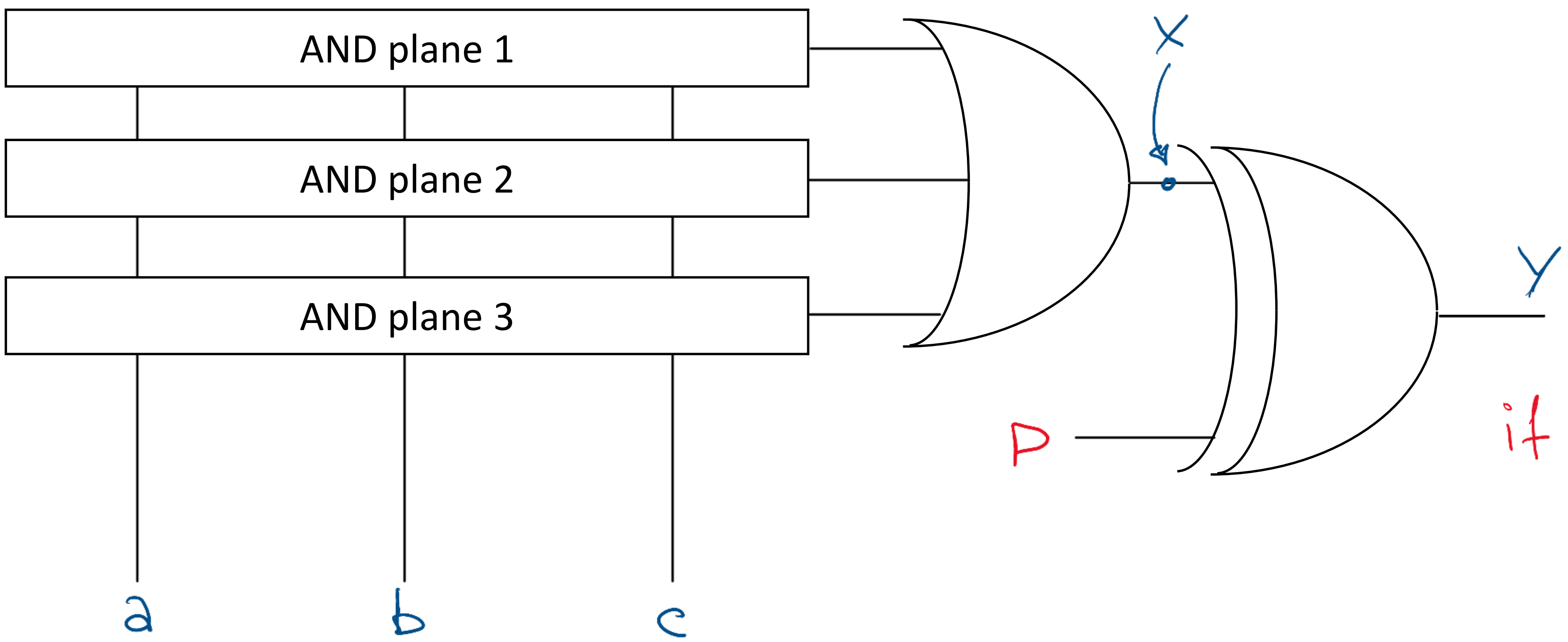
- Output polarity configuration: active-high / active-low



P	X	Y
0	0	0
0	1	1
1	0	1
1	1	0

Evolution of PAL

- Output polarity configuration: active-high / active-low

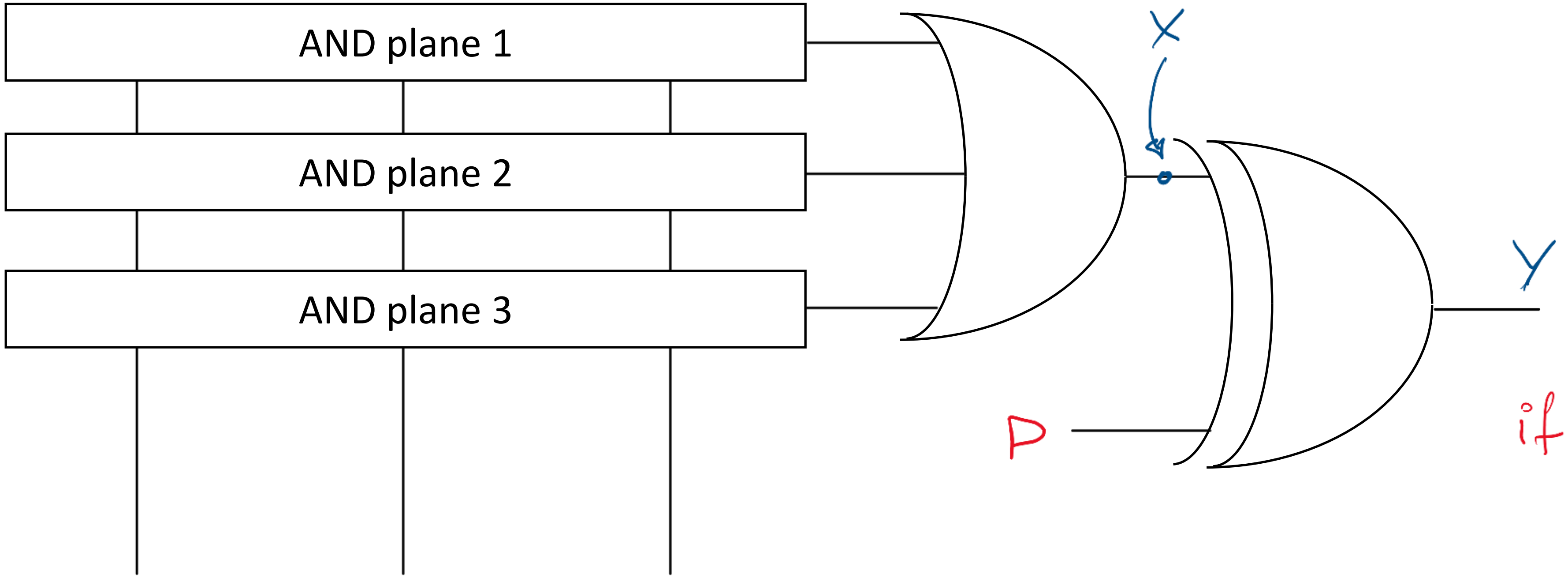


P	X	Y
0	0	0
0	1	1
1	0	1
1	1	0

if $P=0 \Rightarrow Y=X$

Evolution of PAL

- Output polarity configuration: active-high / active-low

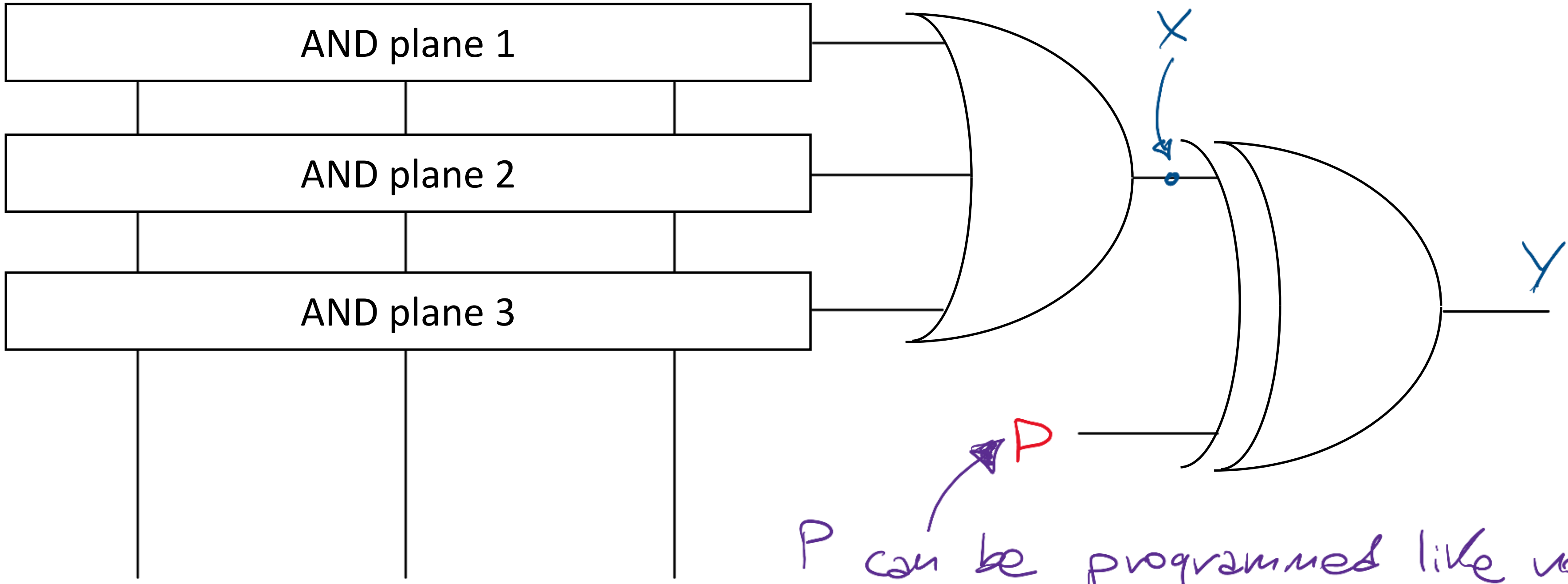


P	X	Y
0	0	0
0	1	1
1	0	1
1	1	0

if $P=1 \Rightarrow Y=\overline{X}$

Evolution of PAL

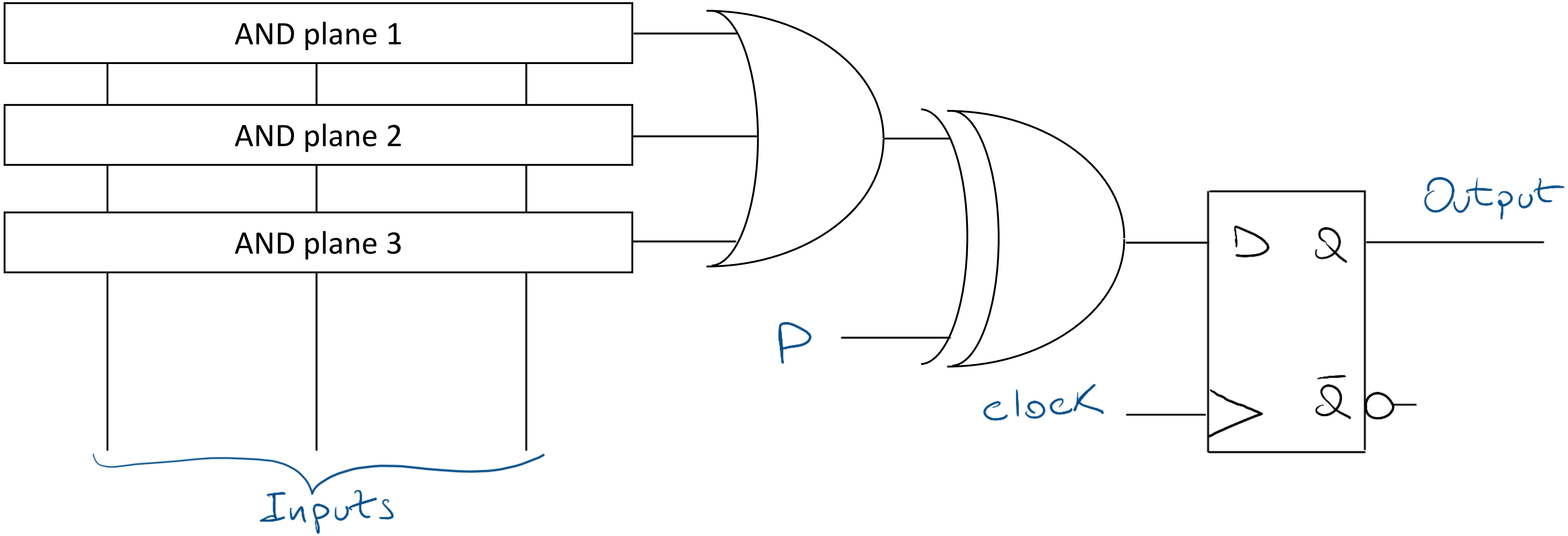
- Output polarity configuration: active-high / active-low



P can be programmed like nodes (fuse, FGMOS)

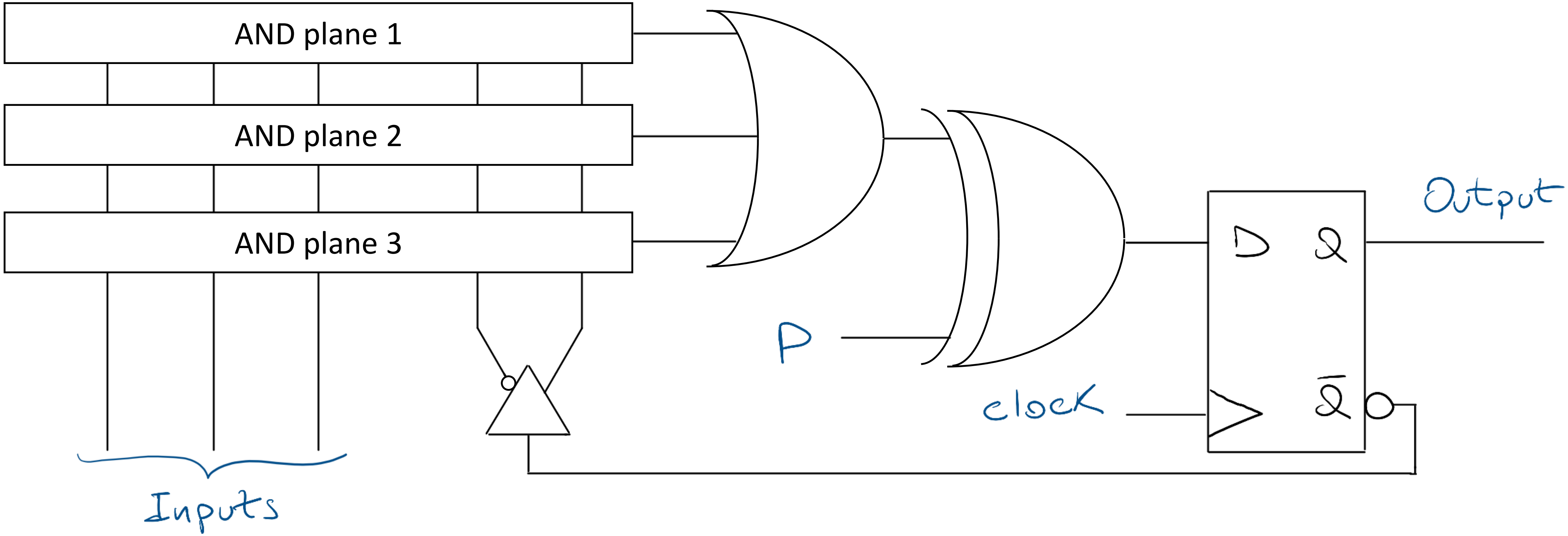
Evolution of PAL

- Registered output for sequential logic



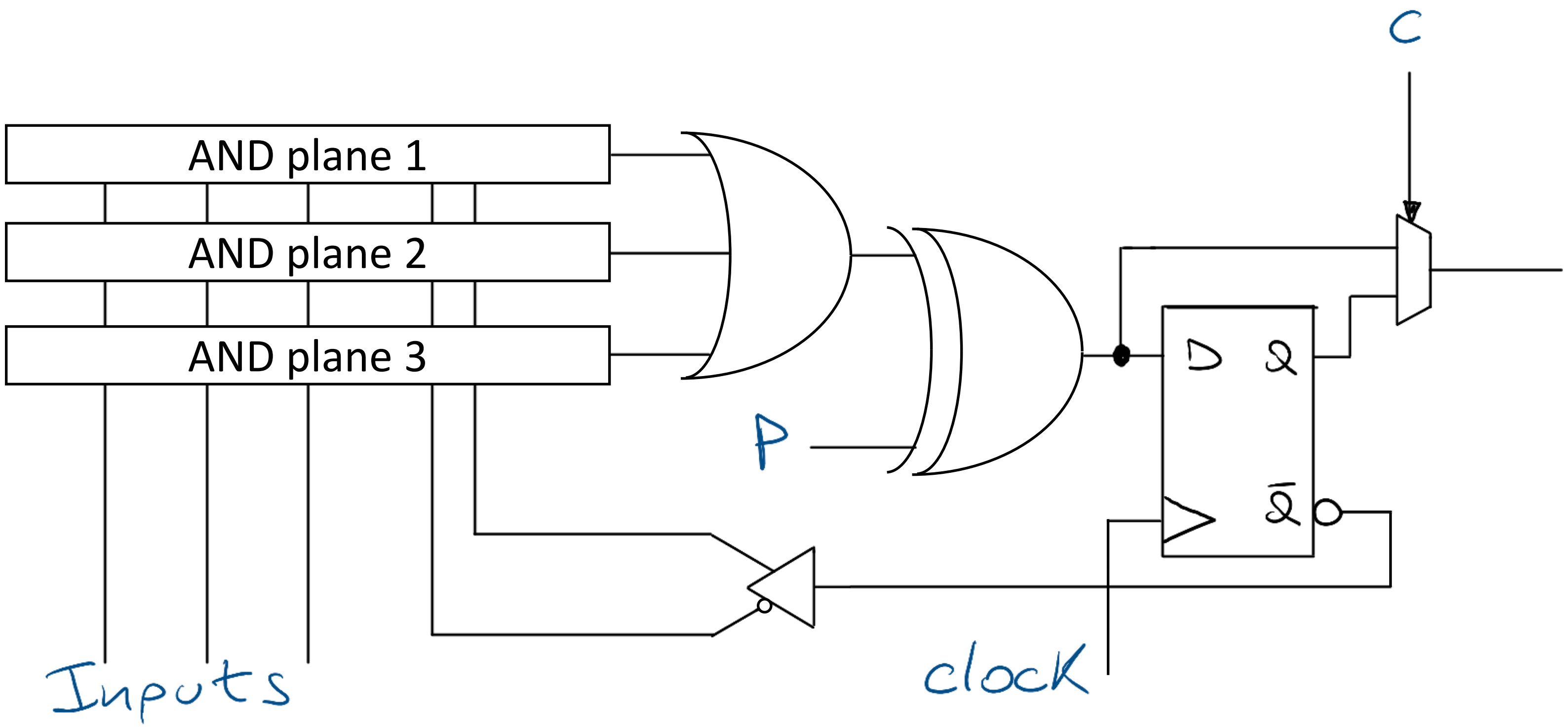
Evolution of PAL

- Feedback path for FSM implementation



Evolution of PAL

- MUX to select between combinational or sequential output



Evolution of PAL

- PAL has evolved by enriching its functionality/flexibility

- Evolution to more complex PLDs (CPLDs) to the latest and most powerful configurable device
  - **FPGA** = **F**ield-**P**rogrammable **G**ate **A**rray

PAL – Last notes

- How PALs were programmed
  - Dedicated equipment
  - Programming language
    - PALASM (PAL Assembler)
    - CUPL (Compiler for Universal Programmable Logic)
    - ABEL (Advanced Boolean Expression Language)

PAL – Last notes

- How PALs were programmed
  - Dedicated equipment
  - Programming language
    - **PALASM** (PAL Assembler)
    - CUPL (Compiler for Universal Programmable Logic)
    - ABEL (Advanced Boolean Expression Language)

```
PAL16R4 PAL                PAL DESIGN SPECIFICATION
CNT4SC
4 bit counter with synchronous clear
Michael Holley and Dave Pellerin
Clk  Clear  NC  NC  NC  NC  NC  NC  NC  GND
 OE  NC     NC /Q3 /Q2 /Q1 /Q0  NC  NC  VCC

  Q3 :=  Clear
      + /Q3 * /Q2 * /Q1 * /Q0
      +  Q3 *  Q0
      +  Q3 *  Q1
      +  Q3 *  Q2

  Q2 :=  Clear
      + /Q2 * /Q1 * /Q0
      +  Q2 *  Q0
      +  Q2 *  Q1

  Q1 :=  Clear
      + /Q1 * /Q0
      +  Q1 *  Q0

  Q0 :=  Clear
      + /Q0

FUNCTION TABLE
OE Clear Clk   /Q0 /Q1 /Q2 /Q3
------------------------------
 L   H    C     L   L   L   L
 L   L    C     H   L   L   L
 L   L    C     L   H   L   L
 L   L    C     H   H   L   L
 L   L    C     L   L   H   L
 L   H    C     L   L   L   L
------------------------------
```

PAL – Last notes

- How PALs were programmed
  - Dedicated equipment
  - Programming language
    - PALASM (PAL Assembler)
    - CUPL (Compiler for Universal Programmable Logic)
    - **ABEL** (Advanced Boolean Expression Language)

```
'' 4 to 1 multiplexer design with case construct
''      SEL0..SEL1 pin;
''      A,B,C,D    pin;
''      MUX_OUT    pin istype 'com';
''
''      SEL = [SEL1,SEL0];

equations

when SEL==0 then MUX_OUT = A; else
when SEL==1 then MUX_OUT = B; else
when SEL==2 then MUX_OUT = C; else
when SEL==3 then MUX_OUT = D;
```

PAL – Last notes

- How PALs were programmed
  - Dedicated equipment
  - Programming language
    - PALASM (PAL Assembler)
    - CUPL (Compiler for Universal Programmable Logic)
    - ABEL (Advanced Boolean Expression Language)

First examples of **HDL**

Thank you for your attention

Luca Crocetti
(luca.crocetti@unipi.it)