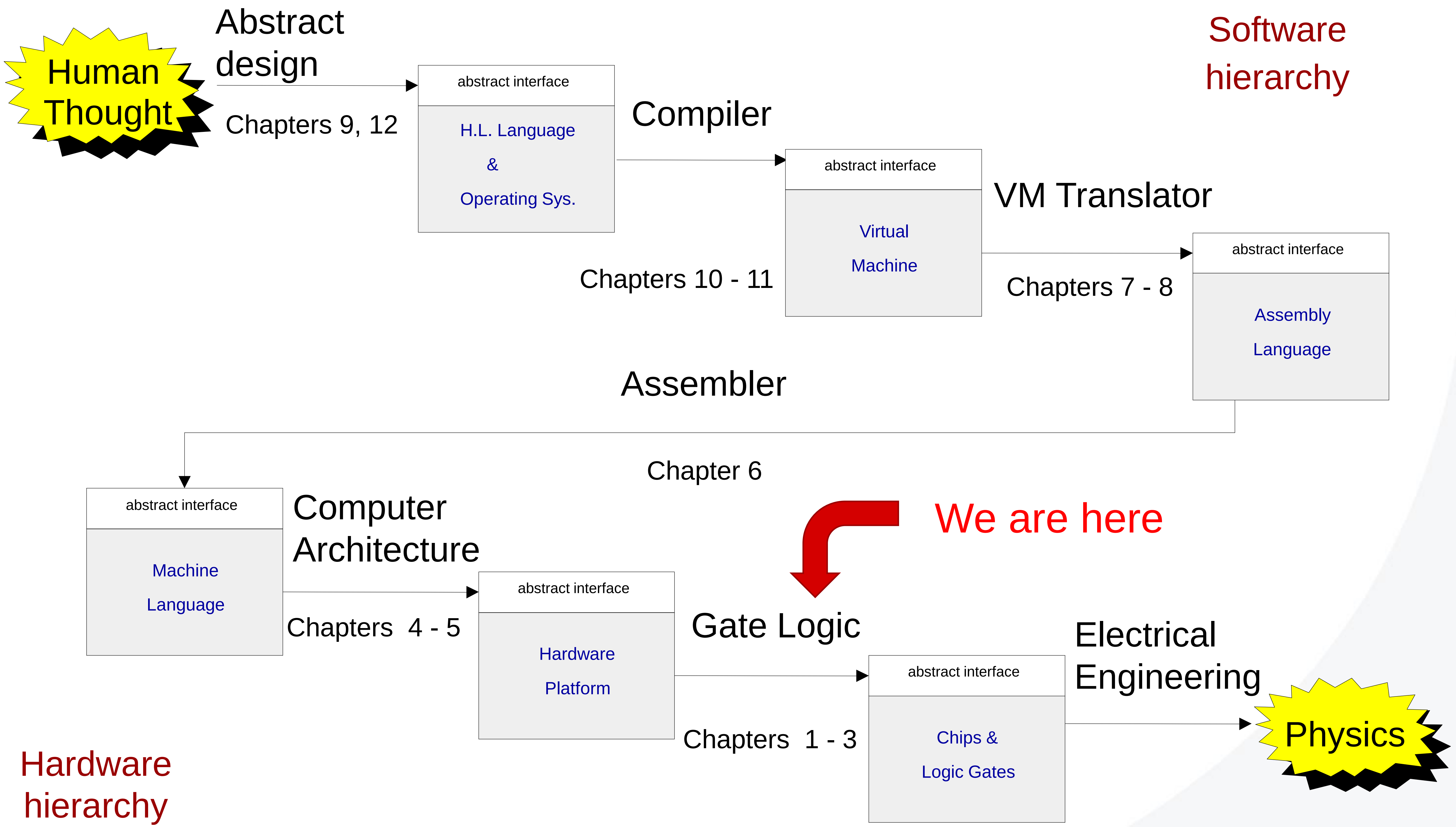
Computer Systems

Lecture 02: Logic Gates

THE UNIVERSITY of ADELAIDE

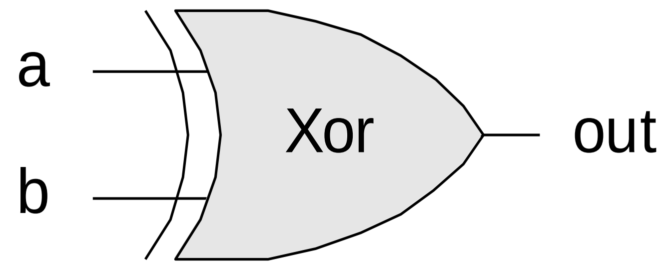
Our Journey



(Abstraction–implementation paradigm)

Review: Gate Logic

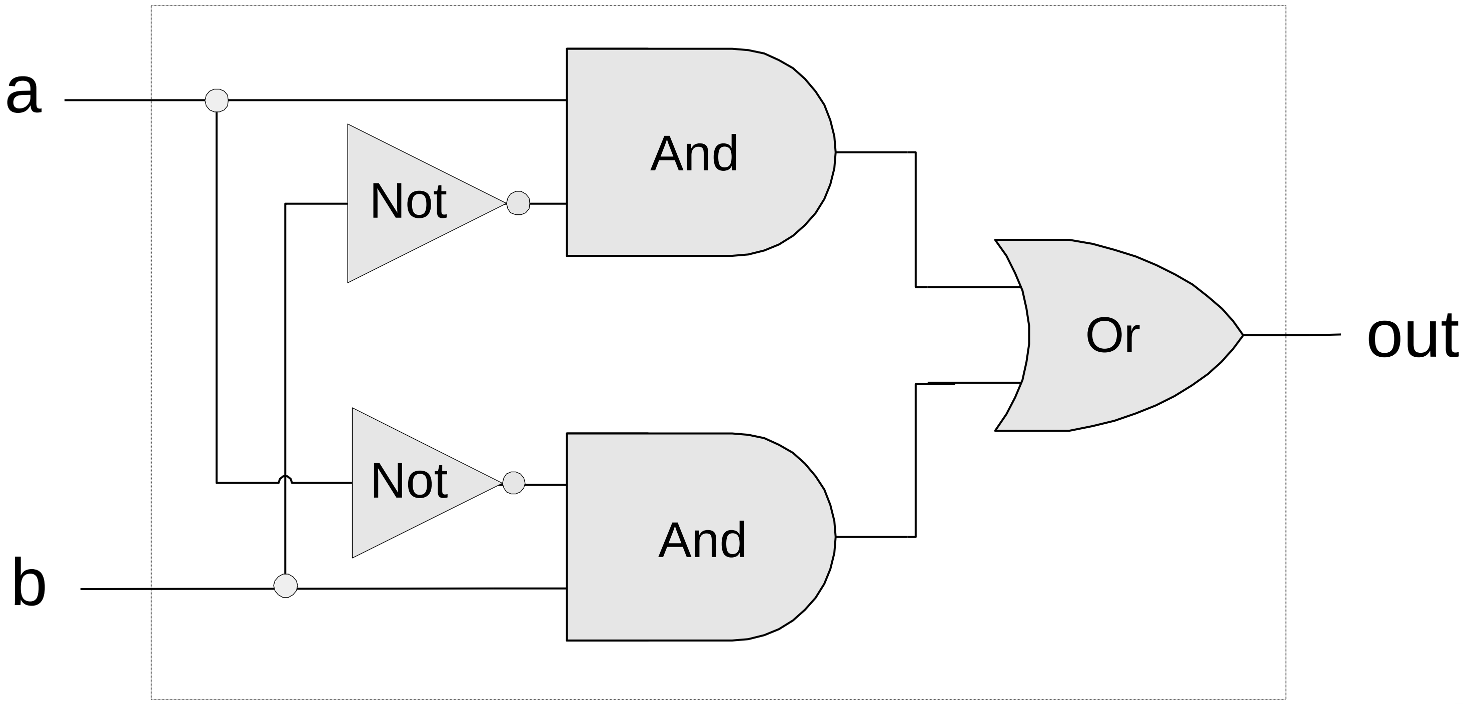
Interface



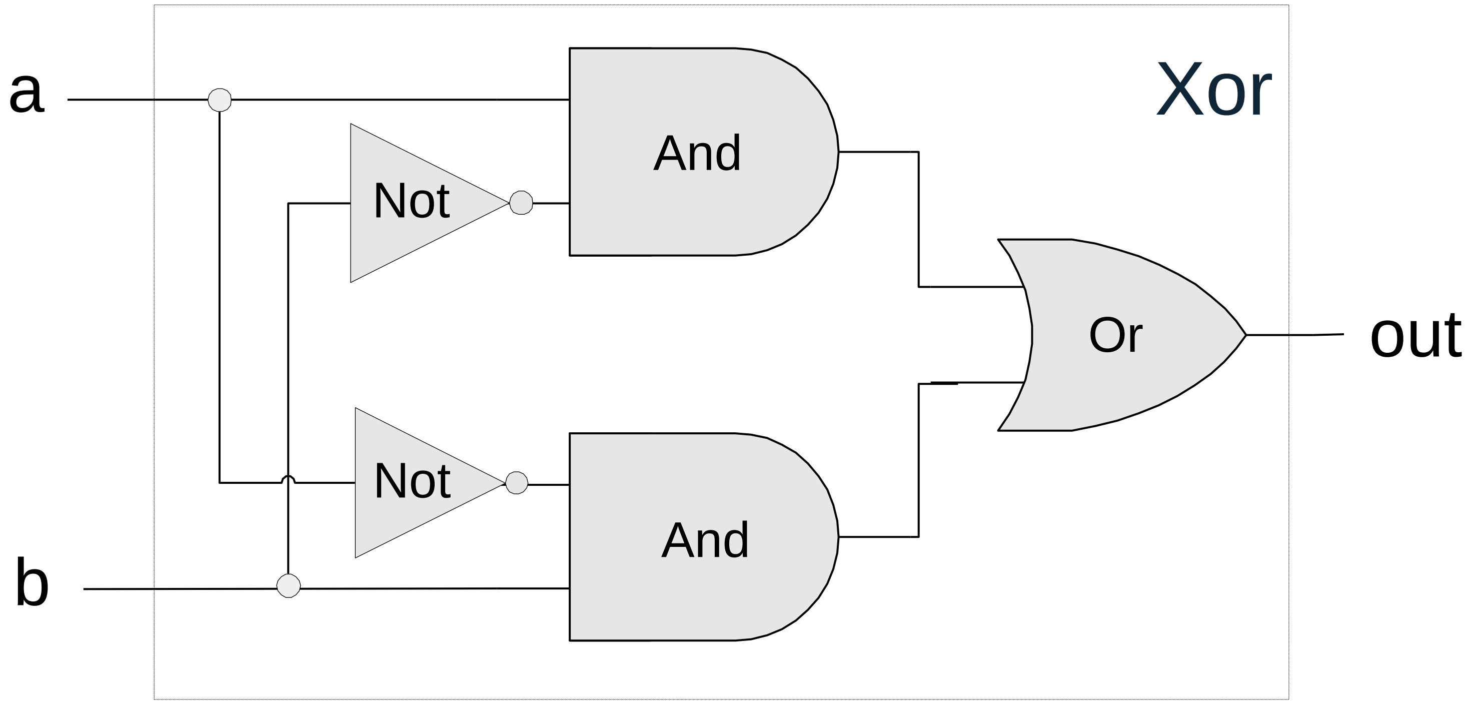
a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

(Truth table)

Implementation



In Workshops You Will Do:




```
CHIP Xor {
    IN a, b;
    OUT out;

    PARTS:
    Not(in=a,out=na);
    Not(in=b,out=nb);
    And(a=na,b=b,out=c);
    And(a=a,b=nb,out=d);
    Or(a=c,b=d,out=out);

}
```

All Boolean functions of 2 variables

Function	x	**0**	**0**	**1**	**1**
	y	**0**	**1**	**0**	**1**
Constant 0 $\leftrightarrow x \cdot \bar{x}$	0	0	0	0	0
x And y	$x \cdot y$	0	0	0	1
x And Not y	$x \cdot \bar{y}$	0	0	1	0
x	x	0	0	1	1
Not x And y	$\bar{x} \cdot y$	0	1	0	0
y	y	0	1	0	1
x Xor y (Add / Difference)	$x \cdot \bar{y} + \bar{x} \cdot y$	0	1	1	0
x Or y	$x + y$	0	1	1	1
x Nor y	$\overline{x + y}$	1	0	0	0
x Xnor y (Equivalence)	$x \cdot y + \bar{x} \cdot \bar{y}$	1	0	0	1
Not y	$\bar{y}$	1	0	1	0
x Or Not y (If y then x)	$x + \bar{y}$	1	0	1	1
Not x	$\bar{x}$	1	1	0	0
Not x Or y (If y then x)	$\bar{x} + y$	1	1	0	1
x Nand y	$\overline{x \cdot y}$	1	1	1	0
Constant 1 $\leftrightarrow x + \bar{x}$	1	1	1	1	1

THE UNIVERSITY of ADELAIDE

Canonical Form: <u>representation of a Boolean function in terms of And, Or, and Not</u>

We can construct a canonical representation of any Boolean function with:

- **And (symbol "dot product" Example: $x \cdot y$)**
- **Or (symbol "plus sign" Example: $x + y$)**
- **Not (symbol "bar on top" Example: $\overline{x}$)**

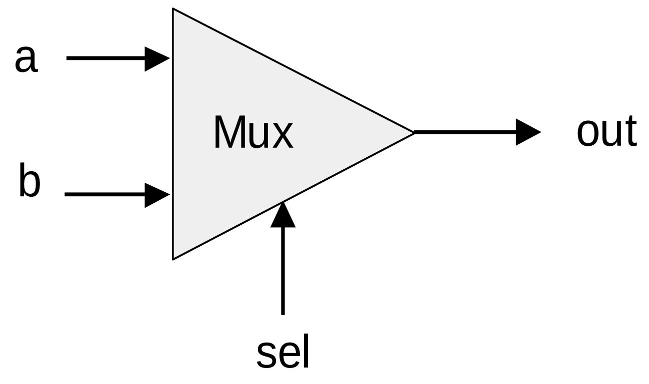
How?

- **<u>Sum of Product (SoP)</u> $\leftarrow$ focus in this course**
- **Product of Sum (PoS)**
- **Karnaugh Map (K-map)**

…

Canonical Form – <u>Sum of Product (SoP)</u>

a	b	sel	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



Exercise at the end!

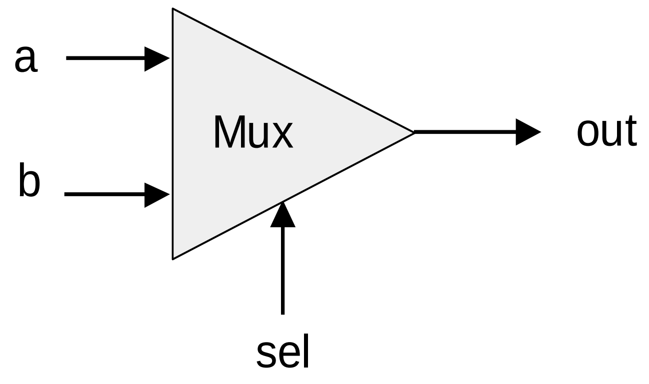
$$out = (\bar{a}.b.sel) + (a.\bar{b}.\overline{sel}) + (a.b.\overline{sel}) + (a.b.sel)$$

1. Find all lines with out = 1, ignore all lines with out = 0.
2. Write inputs in product (And), if the input is 1 write as it is, otherwise, write in bar (Not) form.
3. Repeat 2. for each line with out = 1.
4. Use sum (Or) to connect each term.

THE UNIVERSITY of ADELAIDE

Canonical Form – Product of Sum (PoS)

a	b	sel	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

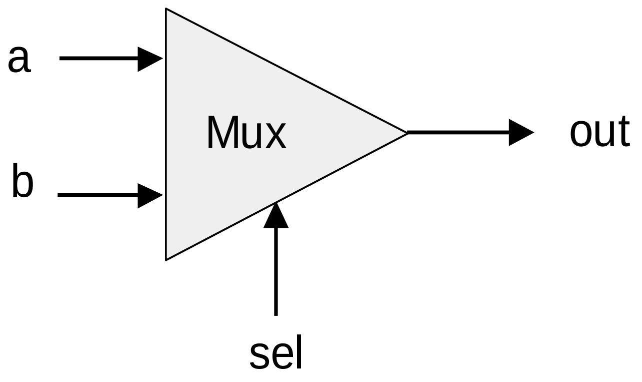


$$\mathbf{out} = (\mathbf{a} + \mathbf{b} + \mathbf{sel}).(\mathbf{a} + \mathbf{b} + \overline{\mathbf{sel}}).$$
$$(\mathbf{a} + \overline{\mathbf{b}} + \mathbf{sel}).(\overline{\mathbf{a}} + \mathbf{b} + \overline{\mathbf{sel}})$$

1. Find all lines with out = 0, ignore all lines with out = 1.
2. Write inputs in sum (Or), if the input is 0 write as it is, otherwise, write in bar (Not).
3. Repeat 2. for each line with out = 0.
4. Use product (And) to connect each term.

Canonical Form – Karnaugh Map (K-map)

a	b	sel	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



sel \ a/b	00	01	11	10
0	0	0	1	1
1	0	1	1	0

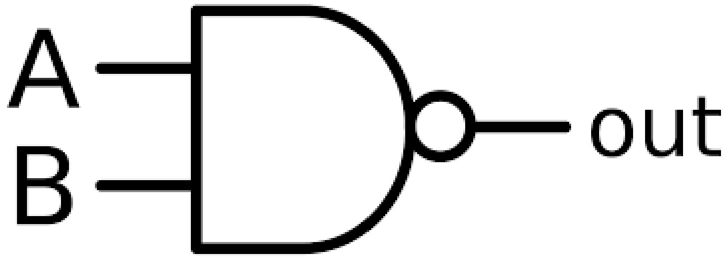
$$out = (a.\overline{sel}) + (b.sel)$$

K-map is one of the systematic way to simplify the Boolean expression.
In this course, we will not ask for simplification of Boolean expressions.
For curiosity please read: https://www.allaboutcircuits.com/textbook/digital/chpt-8/logic-simplification-karnaugh-maps/

How to construct And, Or and <u>Not</u> from Nand

From this

A	B	out
0	0	1
0	1	1
1	0	1
1	1	0

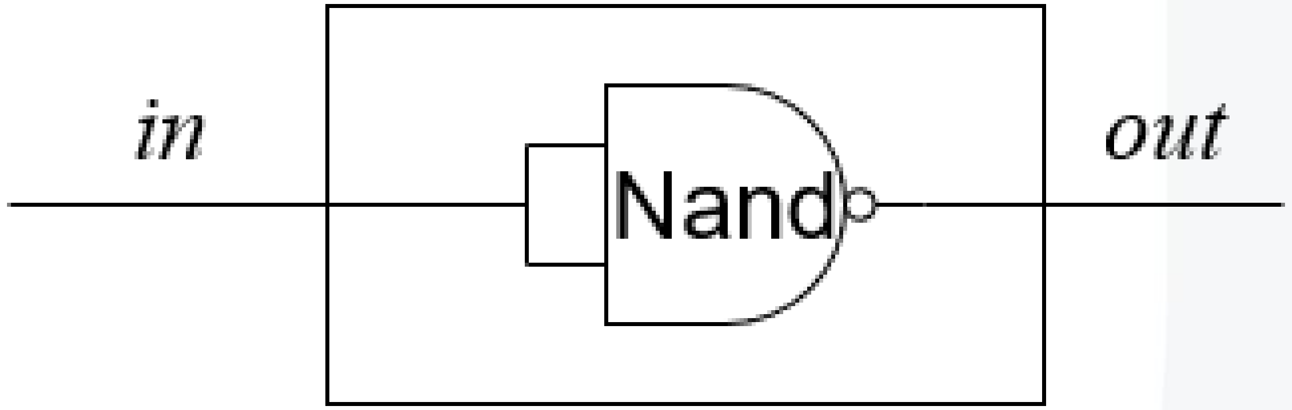


$$out = \overline{A \cdot B}$$

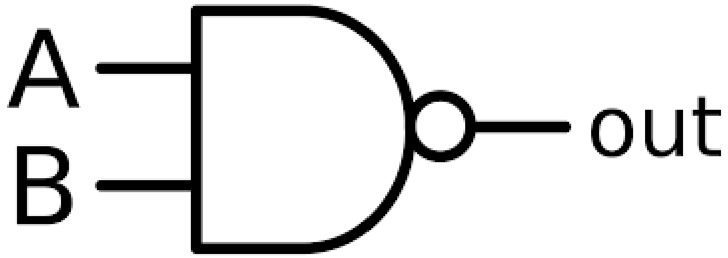
Formal proof in Textbook Appendix A1.3

How to construct And, Or and Not from Nand

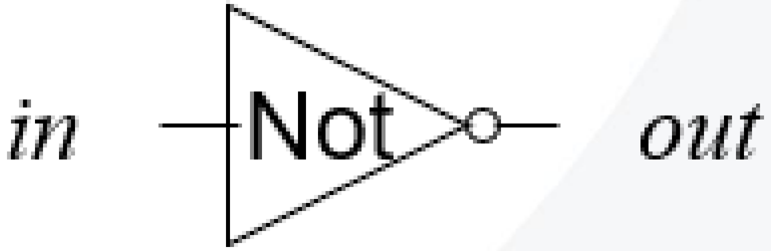
From this



A	B	out
0	0	1
~~0~~	~~1~~	~~1~~
~~1~~	~~0~~	~~1~~
1	1	0



Have A and B commonly connected, A and B always the same

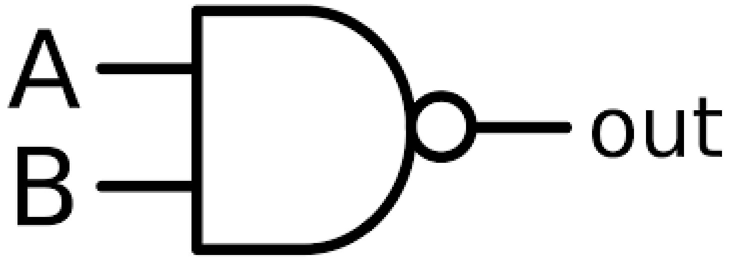


$$out = \overline{A \cdot B}$$

Formal proof in Textbook Appendix A1.3

How to construct And, Or and Not from Nand

From this



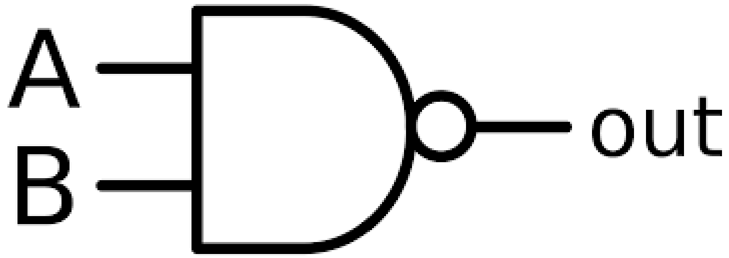
A	B	out
0	0	1
0	1	1
1	0	1
1	1	0

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

Formal proof in Textbook Appendix A1.3

How to construct And, Or and Not from Nand

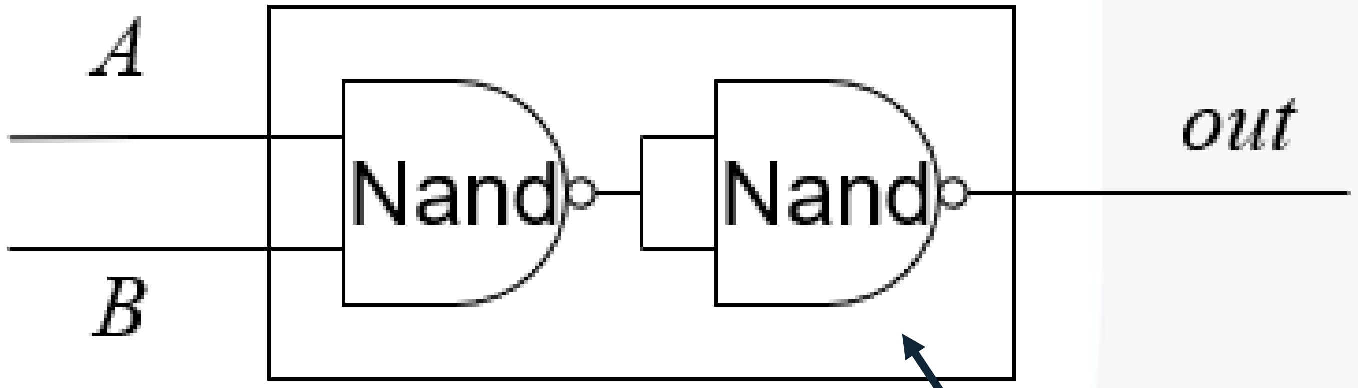
From this



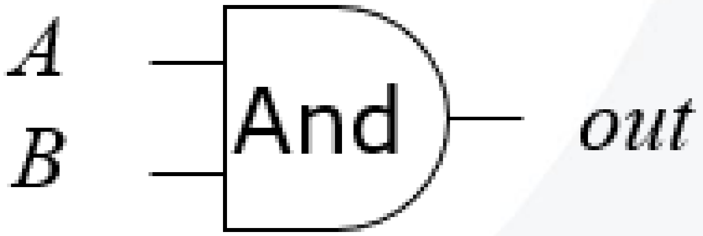
A	B	out	$\overline{out}$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

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

Add one Not gate to the out



Not gate we built in the last slide



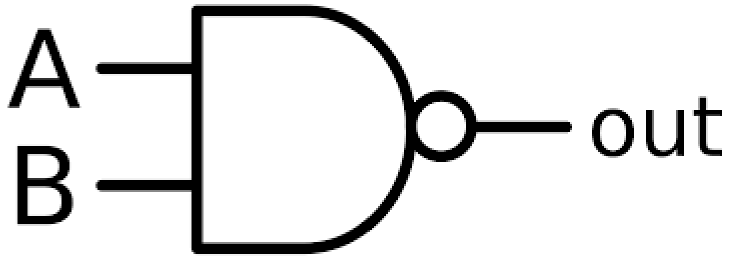
Formal proof in Textbook Appendix A1.3

THE UNIVERSITY of ADELAIDE

How to construct And, <u>Or</u> and Not from Nand

From this

A	B	out
0	0	1
0	1	1
1	0	1
1	1	0

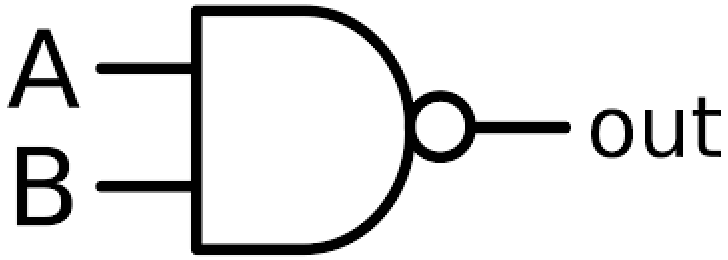


$$out = \overline{A \cdot B}$$

Formal proof in Textbook Appendix A1.3

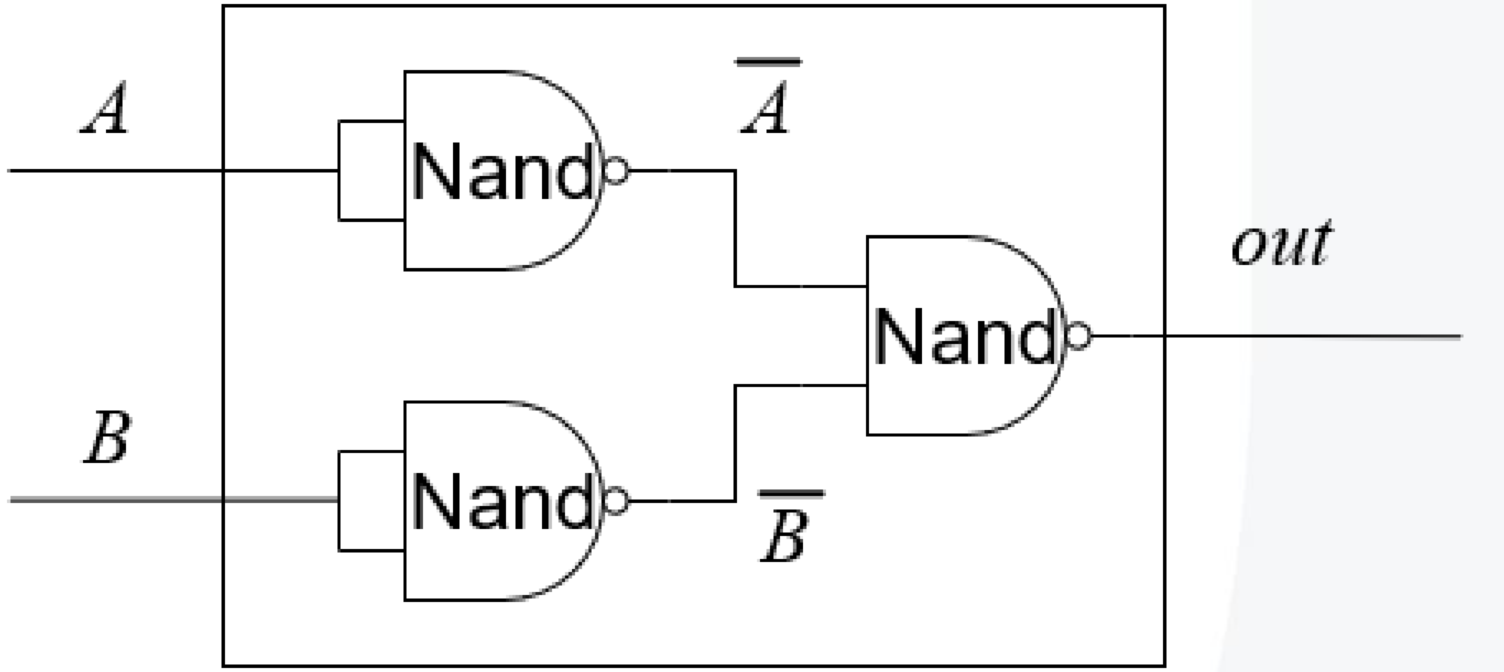
How to construct And, Or and Not from Nand

From this



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

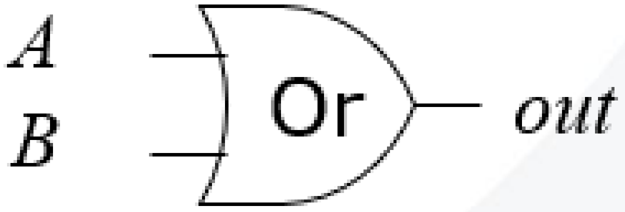
A	B	$\overline{A}$	$\overline{B}$	out
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0



Add one Not gate to each of the A and B

$x+y = \overline{\overline{x}.\overline{y}}$

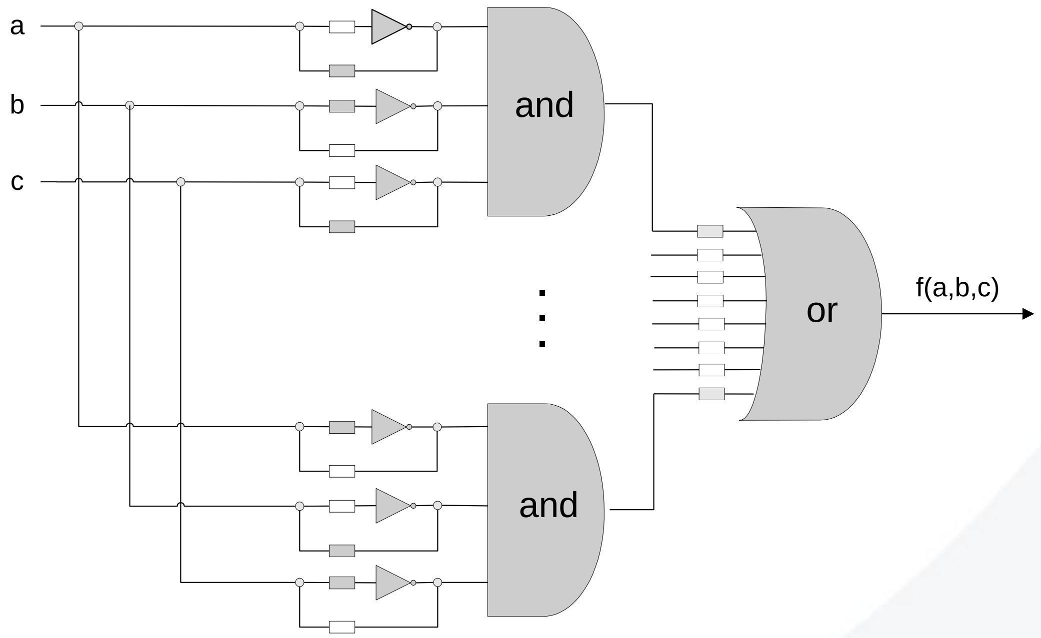
De Morgan's Theorem



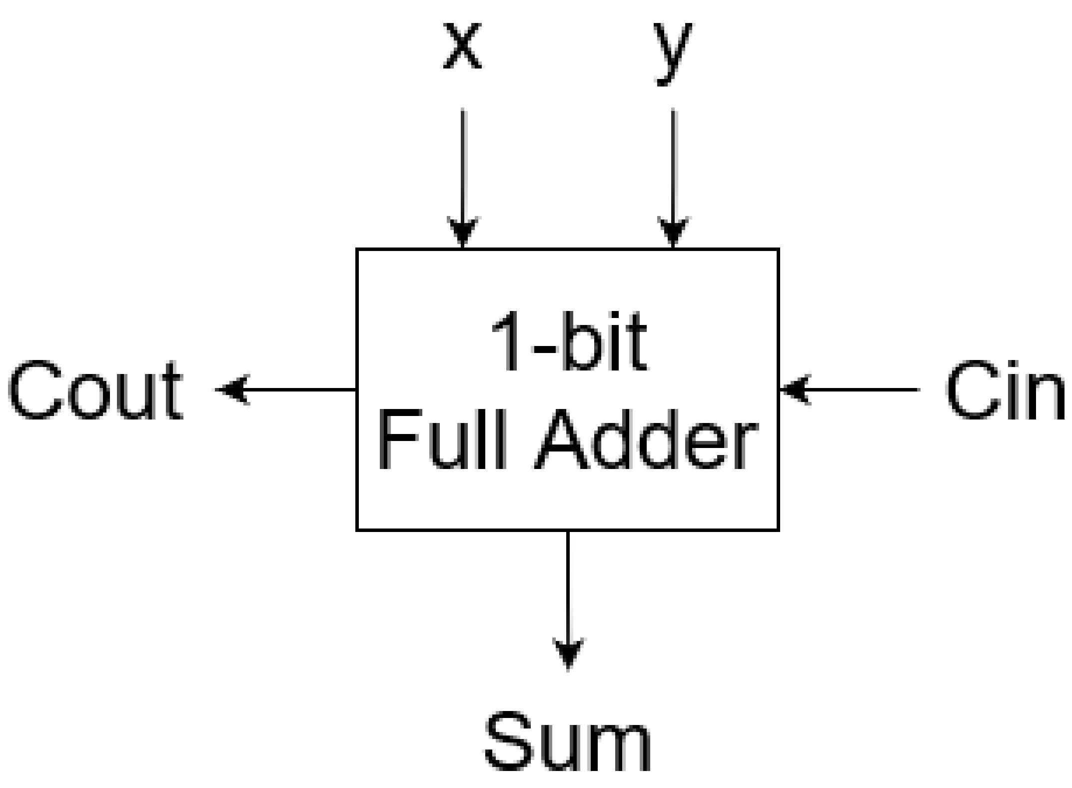
Formal proof in Textbook Appendix A1.3

Boolean Functions: Summary

- Each Boolean function has a canonical representation
- The canonical representation is expressed in terms of <u>And, Or, Not</u>
- And, Not, Or can be expressed in terms of <u>Nand alone</u> (or Nor)
- Every Boolean function can be realised by a standard circuit consisting of Nand gates only
- Mass production
- Universal building blocks, unique topology

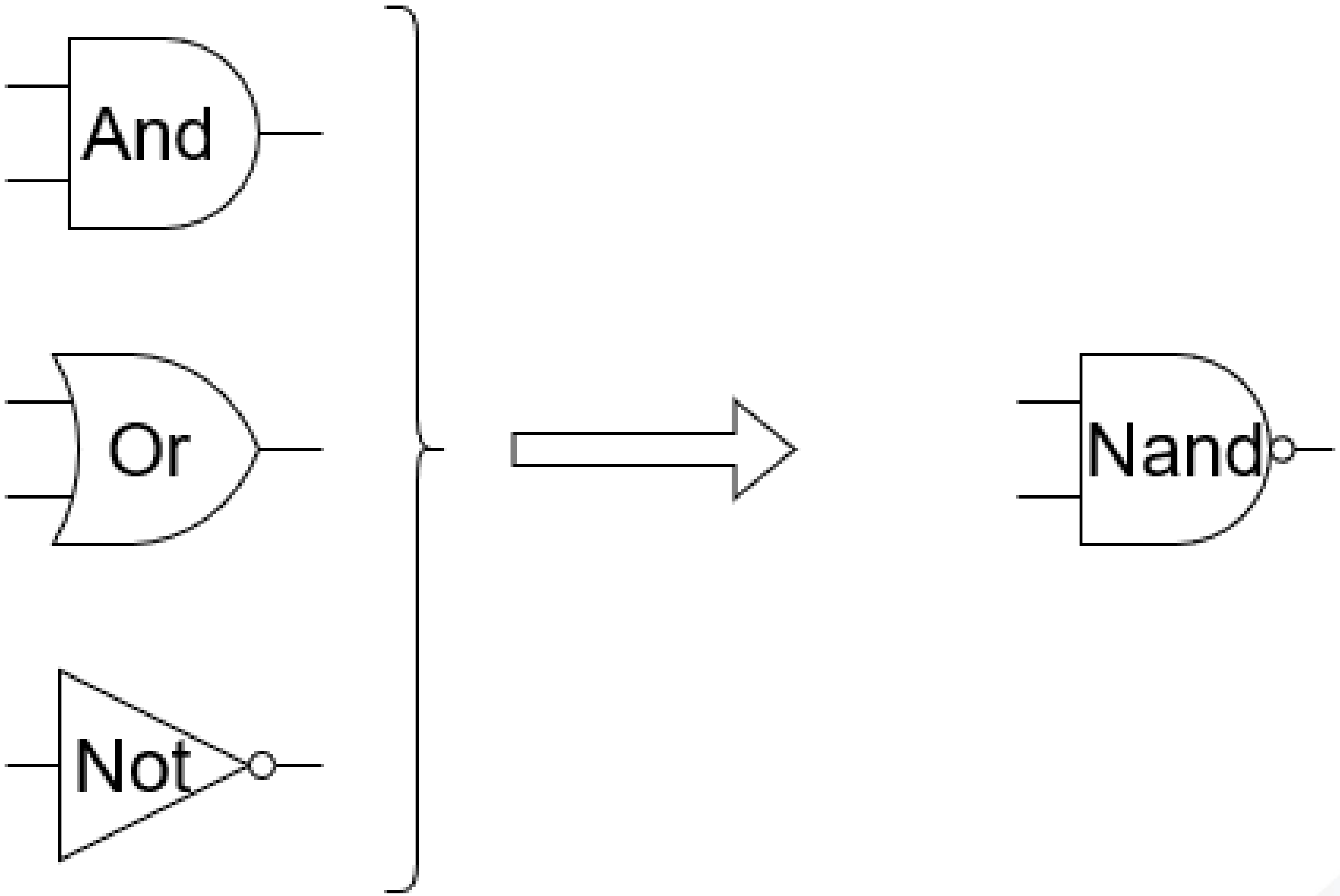


Exercise: Canonical Form Boolean Expression for 1-bit Full Adder

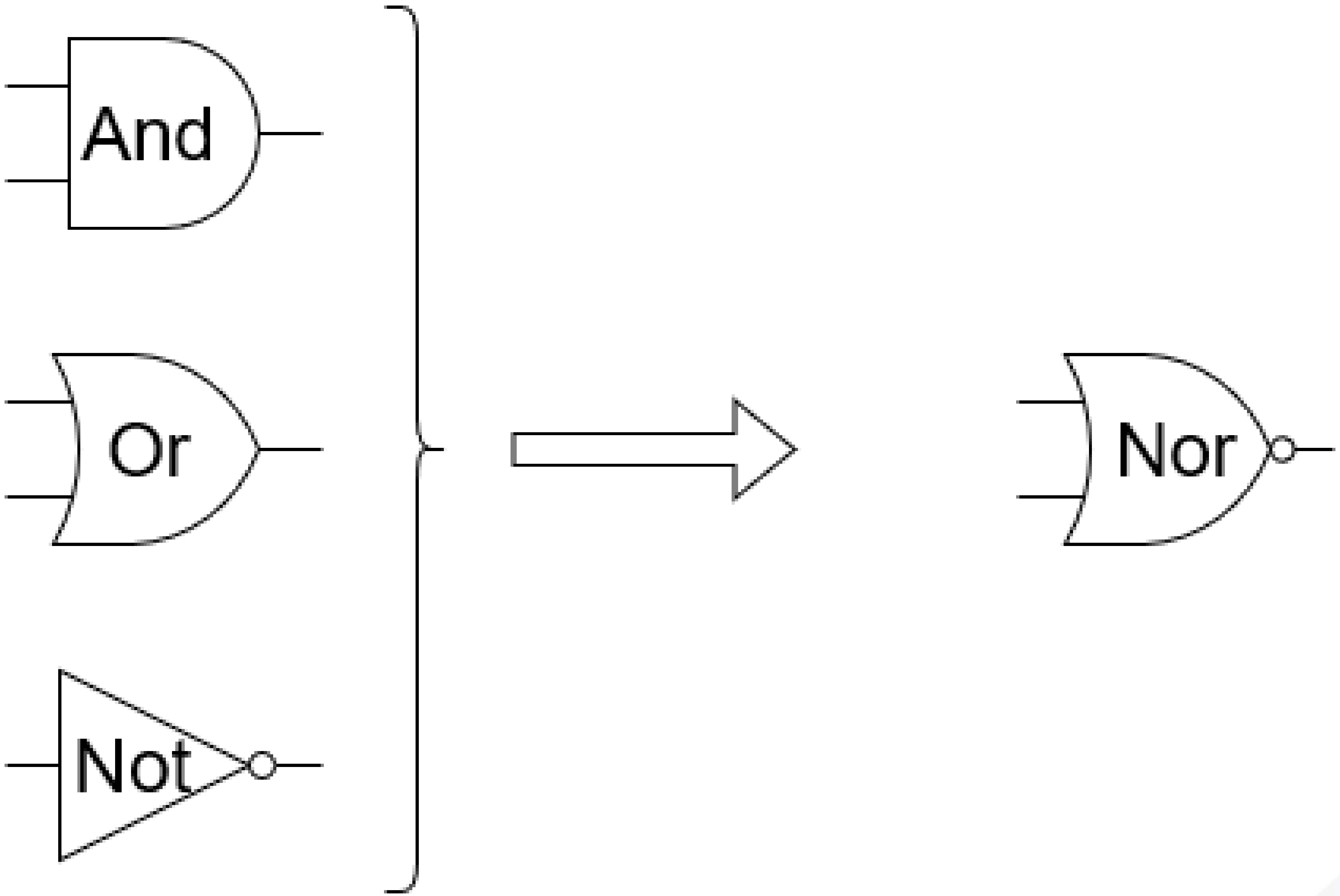


x	y	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Exercise: Build Nand using And, Or, Not



Exercise: Build Nor using And, Or, Not



This Week

- Review Chapter 1 of the Textbook
- Start Assignment 1 (available from Wednesday week 1)
- Workshops start in week 2
- Week 2 quiz due before Thursday lecture.