COMP110: Principles of Computing

4: Logic and memory

Learning outcomes

- **Distinguish** the basic types of logic gate
- **Use** logic gates to build simple circuits
- **Explain** how computer memory works

Logic gates

Boolean logic

Boolean logic

- Works with two values: TRUE and FALSE

Boolean logic

- Works with two values: TRUE and FALSE
- Foundation of the **digital computer**: represented in circuits as **on** and **off**

Boolean logic

- Works with two values: TRUE and FALSE
- Foundation of the **digital computer**: represented in circuits as **on** and **off**
- Representing as 1 and 0 leads to **binary notation**

Boolean logic

- Works with two values: TRUE and FALSE
- Foundation of the **digital computer**: represented in circuits as **on** and **off**
- Representing as 1 and 0 leads to **binary notation**
- One boolean value = one **bit** of information

Boolean logic

- Works with two values: TRUE and FALSE
- Foundation of the **digital computer**: represented in circuits as **on** and **off**
- Representing as 1 and 0 leads to **binary notation**
- One boolean value = one **bit** of information
- Programmers use boolean logic for conditions in `if` and `while` statements

Not

Not

NOT A is TRUE
if and only if
A is FALSE

Not

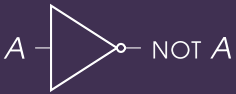
NOT *A* is TRUE
if and only if
A is FALSE

A	NOT *A*
FALSE	TRUE
TRUE	FALSE

Not

NOT *A* is TRUE
if and only if
A is FALSE

A	NOT *A*
FALSE	TRUE
TRUE	FALSE



And

And

A AND *B* is TRUE
if and only if
both *A* **and** *B* are TRUE

And

A AND *B* is TRUE
if and only if
both *A* **and** *B* are TRUE

A	*B*	*A* AND *B*
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

And

A AND *B* is TRUE
if and only if
both *A* **and** *B* are TRUE

A	*B*	*A* AND *B*
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

Or

Or

A OR *B* is TRUE
if and only if
either *A* **or** *B*, **or both,** are TRUE

Or

A OR *B* is TRUE
if and only if
either *A* **or** *B*, **or both,** are TRUE

A	*B*	*A* AND *B*
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

Or

A OR *B* is TRUE
if and only if
either *A* **or** *B*, **or both,** are TRUE

A	*B*	*A* AND *B*
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

Socrative `FALCOMPED`

What is the value of

$$A \text{ AND } (B \text{ OR } C)$$

when

$$A = \text{TRUE}$$
$$B = \text{FALSE}$$
$$C = \text{TRUE}$$

?

Socrative FALCOMPED

What is the value of

$$(\textsc{not}\ A)\ \textsc{and}\ (B\ \textsc{or}\ C)$$

when

$$A = \textsc{True}$$
$$B = \textsc{False}$$
$$C = \textsc{True}$$

?

Socrative `FALCOMPED`

For what values of A, B, C, D is

$$A \text{ AND NOT } B \text{ AND NOT } (C \text{ OR } D) = \text{TRUE}$$

?

Socrative FALCOMPED

What is the value of

$$A \text{ OR NOT } A$$

?

Socrative `FALCOMPED`

What is the value of

$$A \text{ AND NOT } A$$

?

Socrative FALCOMPED

What is the value of

$$A \text{ OR } A$$

?

Socrative `FALCOMPED`

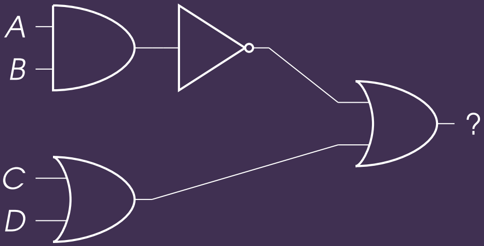
What is the value of

$$A \text{ AND } A$$

?

Socrative `FALCOMPED`

What expression is equivalent to this circuit?



Writing logical operations

Writing logical operations

Operation	Python	C family	Mathematics
NOT A	`not a`	`!a`	$\neg A$ or $\overline{A}$

Writing logical operations

Operation	Python	C family	Mathematics
NOT A	`not a`	`!a`	$\neg A$ or $\overline{A}$
A AND B	`a and b`	`a && b`	$A \wedge B$

Writing logical operations

Operation	Python	C family	Mathematics
NOT A	`not a`	`!a`	$\neg A$ or $\overline{A}$
A AND B	`a and b`	`a && b`	$A \wedge B$
A OR B	`a or b`	`a || b`	$A \vee B$

Writing logical operations

Operation	Python	C family	Mathematics
NOT A	`not a`	`!a`	$\neg A$ or $\overline{A}$
A AND B	`a and b`	`a && b`	$A \wedge B$
A OR B	`a or b`	`a || b`	$A \vee B$

Other operators can be expressed by combining these

Exclusive Or

Exclusive Or

A XOR *B* is TRUE
if and only if
either *A* **or** *B*, **but not both,** are TRUE

Exclusive Or

A XOR *B* is TRUE
if and only if
either *A* **or** *B*, **but not both,** are TRUE

A	*B*	*A* AND *B*
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

Exclusive Or

A XOR *B* is TRUE
if and only if
either *A* **or** *B*, **but not both,** are TRUE

A	*B*	*A* AND *B*
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

Socrative `FALCOMPED`

How can A XOR B be written using the operations AND, OR, NOT?

Negative gates

Negative gates

NAND , NOR , XNOR
are the **negations** of
AND , OR , XOR

Negative gates

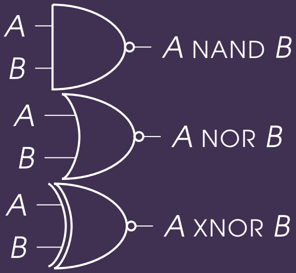
NAND , NOR , XNOR
are the **negations** of
AND , OR , XOR

$$A \text{ NAND } B = \text{NOT } (A \text{ AND } B)$$
$$A \text{ NOR } B = \text{NOT } (A \text{ OR } B)$$
$$A \text{ XNOR } B = \text{NOT } (A \text{ XOR } B)$$

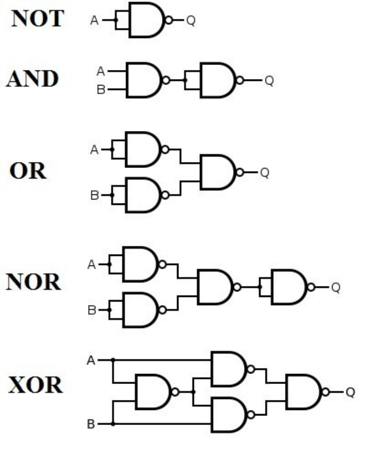
Negative gates

NAND , NOR , XNOR
are the **negations** of
AND , OR , XOR

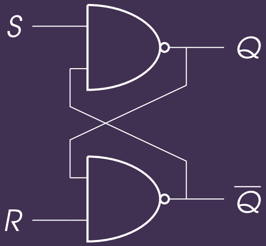
$$A \text{ NAND } B = \text{NOT } (A \text{ AND } B)$$
$$A \text{ NOR } B = \text{NOT } (A \text{ OR } B)$$
$$A \text{ XNOR } B = \text{NOT } (A \text{ XOR } B)$$



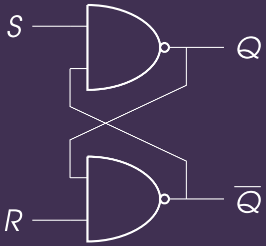
Any logic gate can be constructed from NAND gates



What does this circuit do?

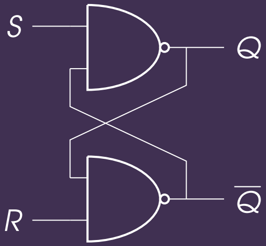


What does this circuit do?



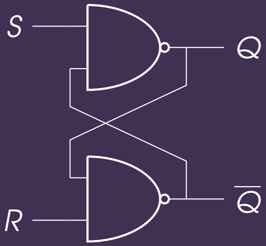
- This is called a **NAND latch**

What does this circuit do?



- This is called a **NAND latch**
- It "remembers" a single boolean value

What does this circuit do?



- This is called a **NAND latch**
- It "remembers" a single boolean value
- Put a few billion of these together (along with some control circuitry) and you've got **memory**!

NAND gates

NAND gates

- All arithmetic and logic operations, as well as memory, can be built from NAND gates

NAND gates

- All arithmetic and logic operations, as well as memory, can be built from NAND gates
- So an entire computer can be built just from NAND gates!

NAND gates

- All arithmetic and logic operations, as well as memory, can be built from NAND gates
- So an entire computer can be built just from NAND gates!
- Play the game: `http://nandgame.com`

NAND gates

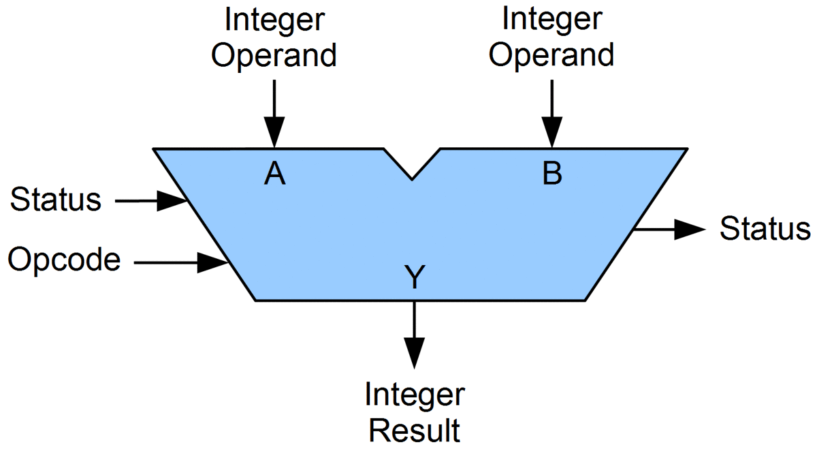
- All arithmetic and logic operations, as well as memory, can be built from NAND gates
- So an entire computer can be built just from NAND gates!
- Play the game: `http://nandgame.com`
- NAND gate circuits are **Turing complete**

NAND gates

- All arithmetic and logic operations, as well as memory, can be built from NAND gates
- So an entire computer can be built just from NAND gates!
- Play the game: `http://nandgame.com`
- NAND gate circuits are **Turing complete**
- The same is true of NOR gates

Arithmetic Logic Unit

Arithmetic Logic Unit



Arithmetic Logic Unit

Arithmetic Logic Unit

- Important part of the CPU

Arithmetic Logic Unit

- Important part of the CPU
- Inputs:
  - **Operand** words A, B
  - **Opcode**
  - **Status** bits

Arithmetic Logic Unit

- Important part of the CPU
- Inputs:
  - **Operand** words A, B
  - **Opcode**
  - **Status** bits
- Outputs:
  - **Result** word Y
  - **Status** bits

Arithmetic Logic Unit

- Important part of the CPU
- Inputs:
  - **Operand** words A, B
  - **Opcode**
  - **Status** bits
- Outputs:
  - **Result** word Y
  - **Status** bits
- Opcode specifies how Y is calculated based on A and B

ALU operations

Typically include:

ALU operations

Typically include:

- Add with carry

ALU operations

Typically include:

- Add with carry
- Subtract with borrow

ALU operations

Typically include:

- Add with carry
- Subtract with borrow
- Negate (2's complement)

ALU operations

Typically include:

- Add with carry
- Subtract with borrow
- Negate (2's complement)
- Increment, decrement

ALU operations

Typically include:

- Add with carry
- Subtract with borrow
- Negate (2's complement)
- Increment, decrement
- Bitwise AND, OR, NOT, . . .

ALU operations

Typically include:

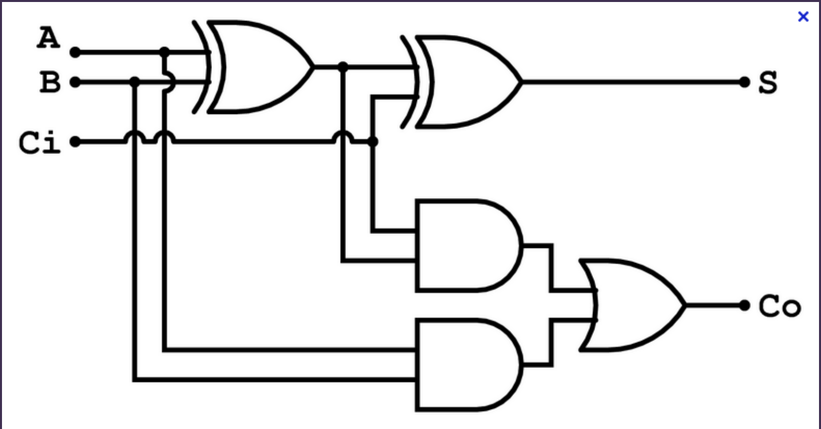
- Add with carry
- Subtract with borrow
- Negate (2's complement)
- Increment, decrement
- Bitwise AND, OR, NOT, ...
- Bit shifts

Adding 3 bits

A	B	C	$A + B + C$
0	0	0	00
0	0	1	01
0	1	0	01
0	1	1	10
1	0	0	01
1	0	1	10
1	1	0	10
1	1	1	11

FALMOUTH UNIVERSITY

1-bit adder

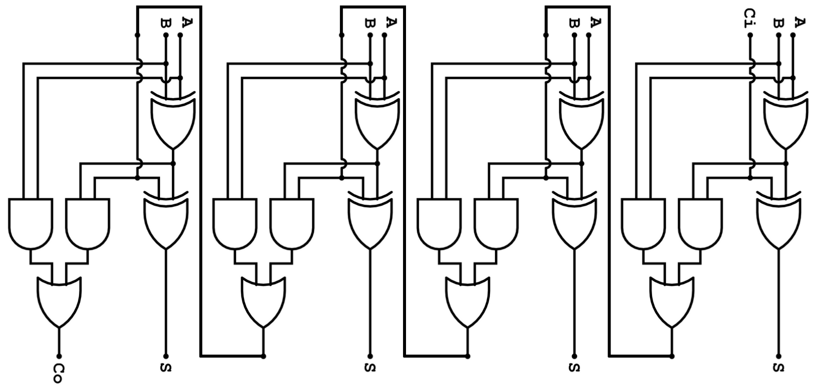


How does the 1-bit adder work?

Exercise:

- Write down the boolean expressions for S and Co
- Draw a truth table for these
- Compare the truth table to the addition table on a previous slide

n-bit adder



Exercise Sheet ii

Due **next Tuesday!**