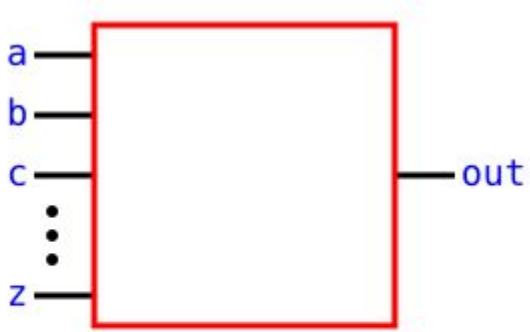
Gates for Digital Circuits

Instructor: Dr. Vinicius Prado da Fonseca (vpradodafons@online.mun.ca)

Digital Circuits

- A digital system, for example a computer, is constructed using components called gates
- Simple digital circuits
  - take one or more binary inputs
  - produce a binary output (0 or 1)
- n-input gate has 2^n possible outputs
- Behavior can be described with a truth table with 2^n entries
- Usually input-output: left-right, top-bottom.
- input letters, beginning of alphabet (a, b, c...)
- Output, Y, F, f



Digital Circuits

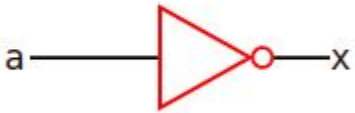
- Defined by a truth table
- Notes on notations:
  - and
    - written “ . ” or “ab”
    - verilog “&”
  - or
    - written “ + ”
    - verilog “ | ”
  - not
    - written “ ~” or “ ` ” or “ – ”
    - verilog “~”

a	b	NAND	AND
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

One Input Gates

- NOT gate
- Inverse of the input
- Notation: a', ~a, ā

Gate: NOT Boolean: $\bar{a}$ Verilog: ~a

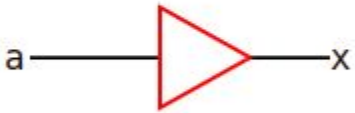


a	x
0	1
1	0

One Input Gates

- Buffer gate
- Copy the input to the output
- Same as a wire but sometimes is required to deliver a correct current (analog level)

Gate: BUF Boolean: *a* Verilog: a



a	x
0	0
1	1

One Input Gates

- Buffers (BUF) are used to restore logic levels.
- NOT gate changes a 0 to a 1, and a 1 to a 0. NOT gates provide logical negation.
- These gates implement all the unary operations.

Two Input Gates

- First operation AND
- Output is true if both inputs are true, high
- Y = A • B, Y = AB, or Y = A ∩ B (intersection)

Gate: AND Boolean: $a \cdot b$ or ab Verilog: a & b

a	b	x
0	0	0
0	1	0
1	0	0
1	1	1

Two Input Gates

- OR operation
- Output is true if any input is true, high
- Y = A + B or Y = A ∪ B (union)

Gate: OR Boolean: $a + b$ Verilog: a | b

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

Two Input Gates

- The NOT gate and the AND and OR gates are sufficient to implement any truth table.
- In digital circuits, NOR and NAND gates are more commonly used since they are cheaper to build.
- They can be constructed with four transistors, while AND gates and OR gates require six transistors (CMOS).
- NAND or NOR can be used to construct an AND, an OR and a NOT gate.
- Therefore either NAND or NOR can be used to implement any truth table.

Two Input Gates

- NAND Operation
- NOT AND

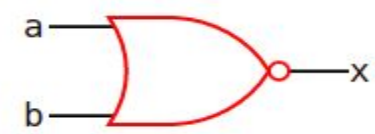
Gate: NAND Boolean: $\overline{a \cdot b}$ or $\overline{ab}$ Verilog: ~(a & b)

a	b	x
0	0	1
0	1	1
1	0	1
1	1	0

Two Input Gates

- NOR operation
- NOT OR
- In digital circuits, NOR and NAND gates are more commonly used since they are cheaper to build, less transistors

Gate: NOR Boolean: $\overline{a+b}$ Verilog: `~(a | b)`



a	b	x
0	0	1
0	1	0
1	0	0
1	1	0

Two Input Gates

- XOR (exclusive OR, pronounced “ex-OR”)
- indicated by A ⊕ B
- TRUE if A or B, but not both, odd number of TRUE inputs

Gate: XOR Boolean: $a \oplus b$ Verilog: `(a ^ b)`

a	b	x
0	0	0
0	1	1
1	0	1
1	1	0

Two Input Gates

- XNOR gate that performs the inverse of an XOR

Gate: XNOR Boolean: $\overline{a \oplus b}$ Verilog: `~(a ^ b)`

a	b	x
0	0	1
0	1	0
1	0	0
1	1	1

Three input gates

- All of the gates except NOT can have more than two inputs
  - Full descriptions in the notes

Gate: AND Boolean: $a \cdot b \cdot c$ or abc Verilog: a & b & c

a	b	c	x
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

Three input gates

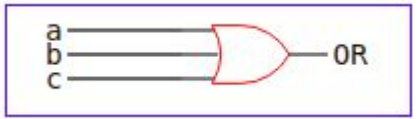
- Same behaviour
- XOR, less intuitive. True if odd number of inputs.

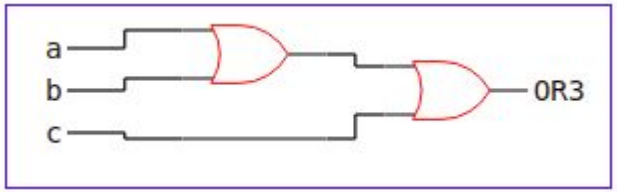
Gate: XOR Boolean: $a \oplus b \oplus c$ Verilog: a ^ b ^ c

a	b	c	x
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

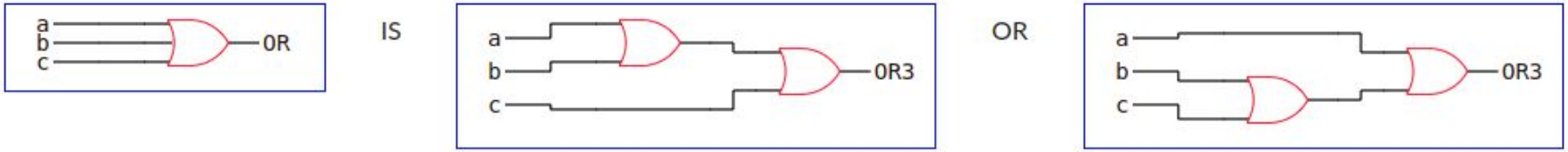
Building gates from two input gates

- Three input OR gate can be built using two input OR gates
- (a + b) + c

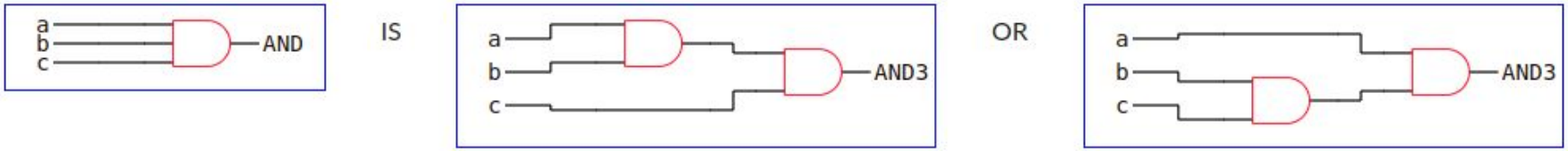




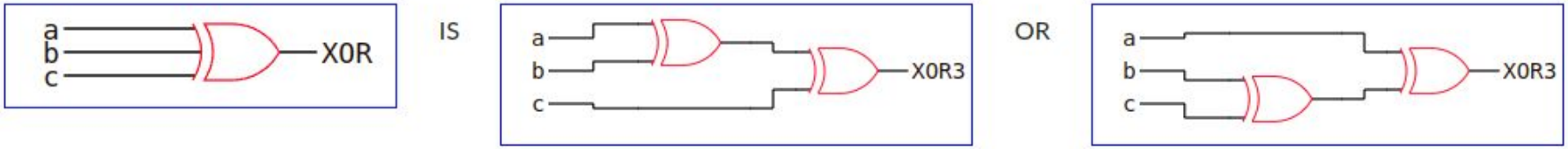
Three input OR gate can be built using two input OR gates.



Three input AND gate can be built using two input AND gates.

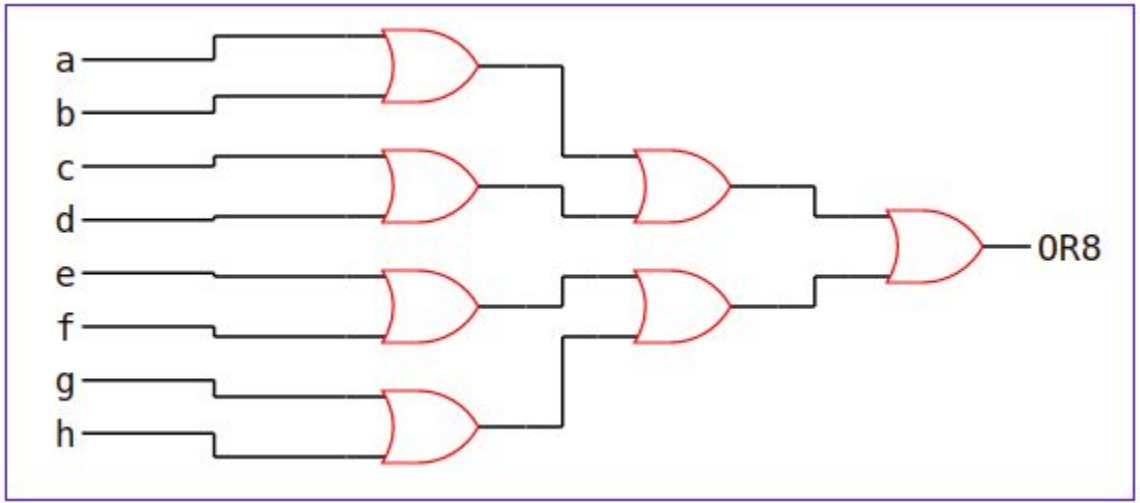


Three input XOR gate can be built using two input XOR gates.



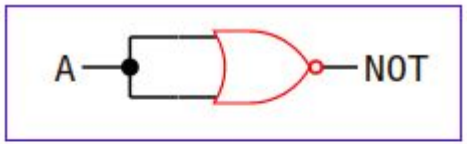
Building gates from two input gates

- N-input OR gate can be built with a tree of two input OR gates



Building gates from two input gates

- A NOT gate can be constructed with a NOR gate.

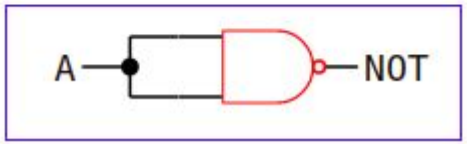


a	b	x
0	0	1
0	1	0
1	0	0
1	1	0

A	N
0	1
1	0

Building gates from two input gates

- A NOT gate can be constructed with a NAND gate.

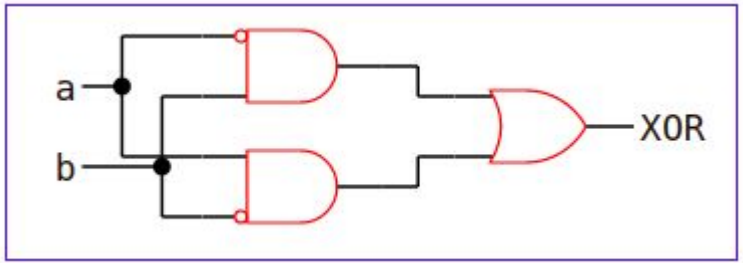


a	b	x
0	0	1
0	1	1
1	0	1
1	1	0

A	N
0	1
1	0

Building gates from two input gates

- An XOR gate can be made with:



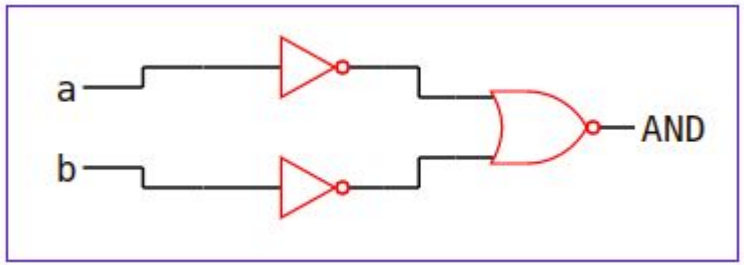
a	b	g1	g2	XOR
0	0	0	0	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

Building gates from two input gates

- Based on De Morgan's law (Unit 2) we can build an AND gate using two NOT gates and NOR gate:

  a.b = (a.b)'' = (a'+ b')'

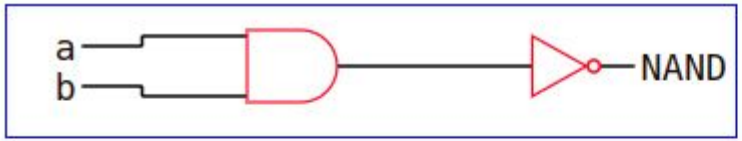
- Also works for OR built by NAND gate.



a	b	Na	Nb	AND
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Building gates from two input gates

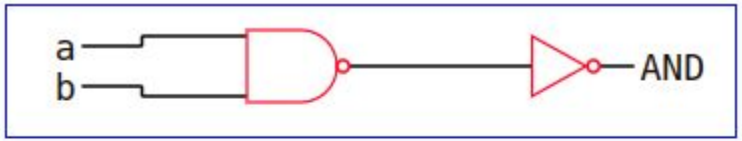
- A NAND gate can be built with an AND gate and NOT gate.



a	b	NAND	AND
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Building gates from two input gates

- An AND gate can be built with an NAND gate and a NOT gate.
- In CMOS circuits, an AND gate must be built using NAND and NOT gates.



a	b	NAND	AND
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Combinational circuits

- Gates used for combinational circuits
- Interconnected set of gates
- Output at any time is a function of the inputs at that time
- More on Unit 2

Questions?

Next: 04 - Gates From Transistors