Second Week
Combinational Systems

Definition and Analysis

Overview

- Combinational systems overview
- Switching algebra
- Logic operations
- Logic mnemonics
- Minimal sets of logic operations
- Multi-input logic operations
- Combinational systems analysis
- Logic expressions
- Documenting combinational systems

Combinational Logic

- One or more digital signal inputs
- One or more digital signal outputs
- Outputs are only functions of current input values (ideal) plus logic propagation delays

$$O_1(t+\Delta t) = F_1(I_1(t),\ldots I_m(t))$$
$$\vdots$$
$$O_n(t+\Delta t) = F_n(I_1(t),\ldots I_m(t))$$

Combinational Logic (cont.)

- Combinational logic has no memory!
  - Outputs are only function of current input combination
  - Nothing is known about past events
  - Repeating a sequence of inputs always gives the same output sequence
- Sequential logic (covered later) does have memory
  - Repeating a sequence of inputs can result in an entirely different output sequence

Combinational Logic Example

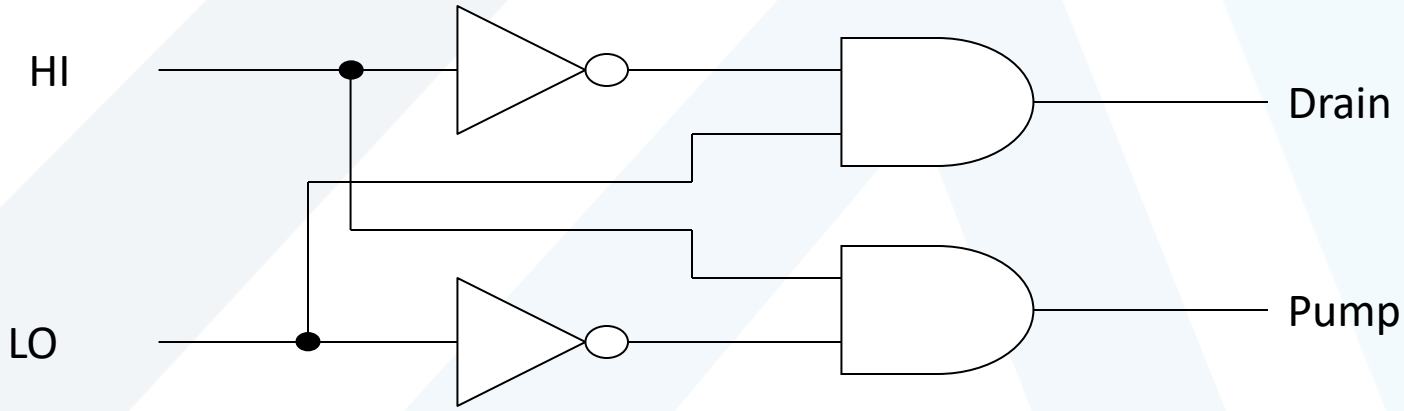
- Circuit controls the level of fluid in a tank
  - inputs are:
    - HI - 1 if fluid level is too high, 0 otherwise
    - LO - 1 if fluid level is too low, 0 otherwise
  - outputs are:
    - Pump - 1 to pump fluid into tank, 0 for pump off
    - Drain - 1 to open tank drain, 0 for drain closed
  - input to output relationship is described by a truth table

Combinational Logic Example (cont.)

HI	LO	Pump	Drain
0	0	0	0
0	1	1	0
1	0	0	1
1	1	x	x

Tank level is OK
Low level, pump more in
High level, drain some out
inputs cannot occur

Schematic Representation



Switching Algebra

- Based on Boolean Algebra
  - Developed by George Boole in 1854
  - Formal way to describe logic statements and determine truth of statements
- Only has two-values domain (0 and 1)
- Huntington’s Postulates define underlying assumptions

Huntington's Postulates

- Closure

  If X and Y are in set (0,1) then operations X+Y and X ·Y are also in set (0,1)

- Identity

  $X + 0 = X$ $\qquad$ $X \cdot 1 = X$

- Commutative

  $X + Y = Y + X$ $\qquad$ $X \cdot Y = Y \cdot X$

Huntington's Postulates (cont.)

- Distributive

  $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$

  $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

- Complement

$$X + \overline{X} = 1$$

$$X \cdot \overline{X} = 0$$

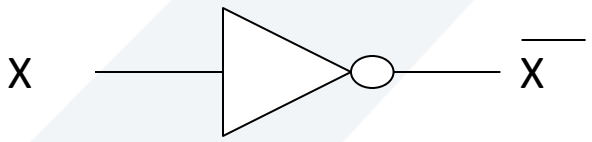
Note that for each property, one form is the dual of the other;

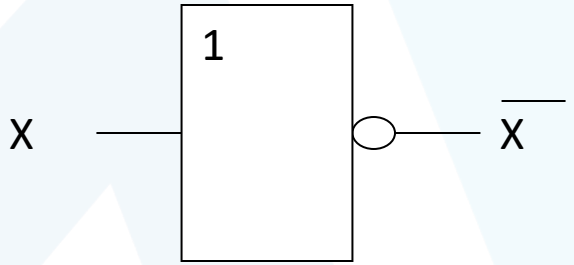
(0s to 1s, 1s to 0s, ·s to +s, +s to ·s)

Switching Algebra Operations - Not

- Unary complement or inversion operation
- Usually shown as overbar (X), other forms are ˜X, X’

X	$\overline{X}$
0	1
1	0

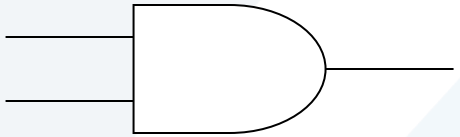


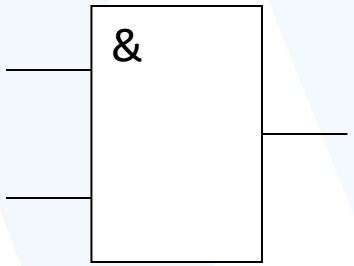


Switching Algebra Operations - AND

- Also known as the conjunction operation; output is true (1) only if all inputs are true
- Algebraic operators are '·', '&', '∧'

X	Y	X·Y
0	0	0
0	1	0
1	0	0
1	1	1

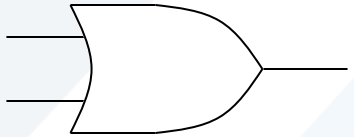


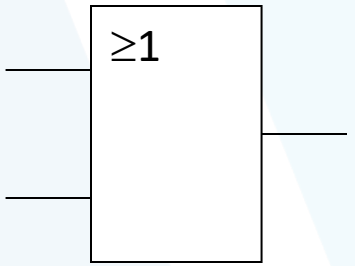


Switching Algebra Operations - OR

- Also known as the disjunction operation; output is true (1) if any input is true
- Algebraic operators are ‘+’, ‘|’, ‘∨’

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

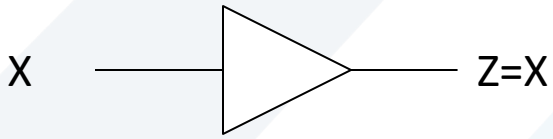


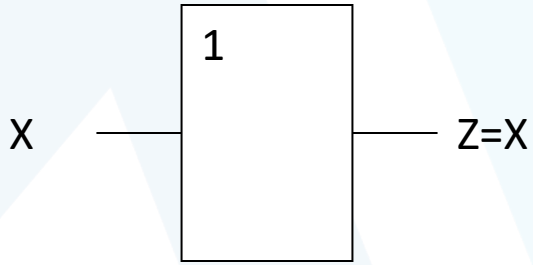


Additional Logic Operations

- For two inputs, there are 16 ways we can assign output values
  - Besides AND and OR, four others are useful
- The unary Buffer operation is useful in the real world

X	Z=X
0	0
1	1

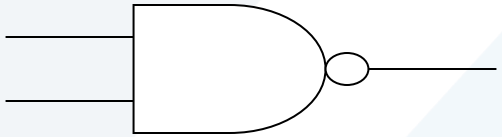


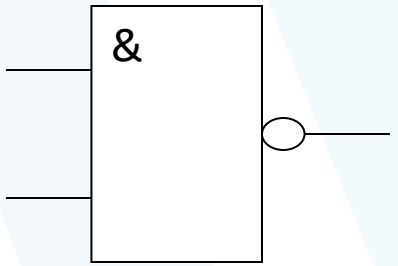


Additional Logic Operations - NAND

- NAND (NOT - AND) is the complement of the AND operation

X	Y	$\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

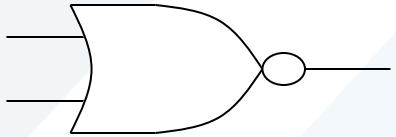


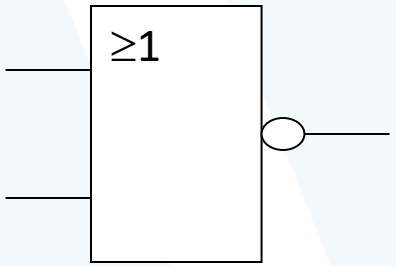


Additional Logic Operations - NOR

- NOR (NOT - OR) is the complement of the OR operation

X	Y	$\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

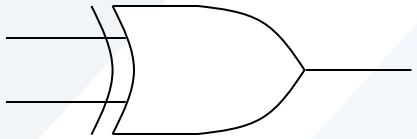


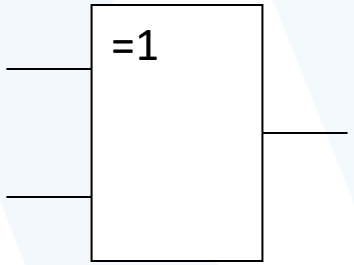


Additional Logic Operations -XOR

- Exclusive OR is similar to the inclusive OR (AKA OR) except output is 0 for 1,1 inputs
- Alternatively the output is 1 when modulo 2 input sum is equal to 1

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

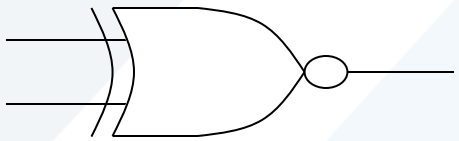


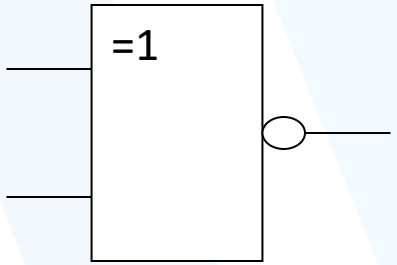


Additional Logic Operations - XNOR

- Exclusive NOR is the complement of the XOR operation
- Alternatively the output is 1 when modulo 2 input sum is not equal to 1

X	Y	$\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	1

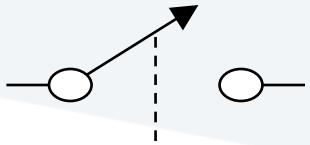




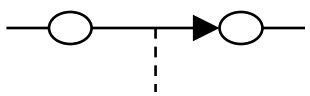
Logic Mnemonics

- Another method for understanding logic operations is to represent them as switches
  - Output = 1 is there is a current path thru switch
  - Practical application in relay logic
- Another method is Venn diagrams
  - Shaded area is where output is 1 or true

Switch/Venn Equivalents

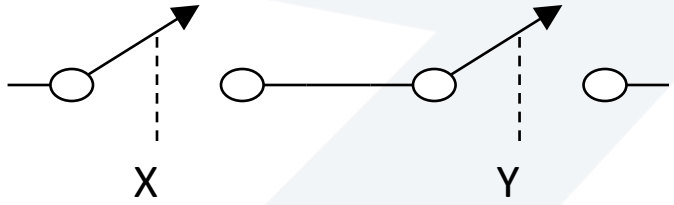


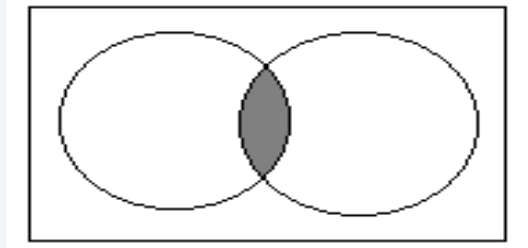
Input = 0
switch open



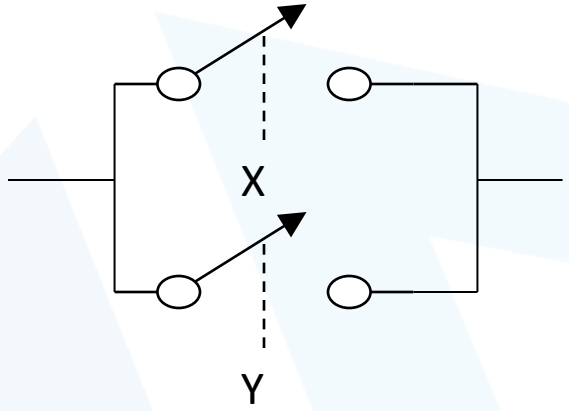
Input = 0
switch closed

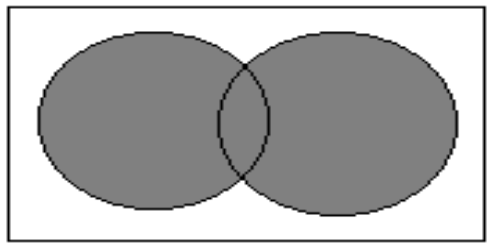
AND



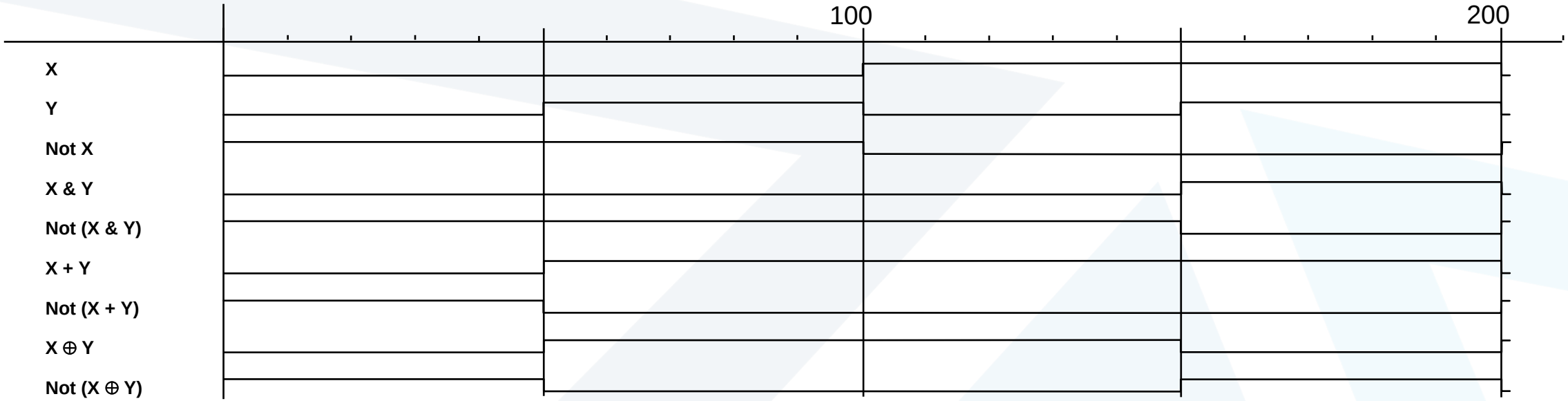


OR





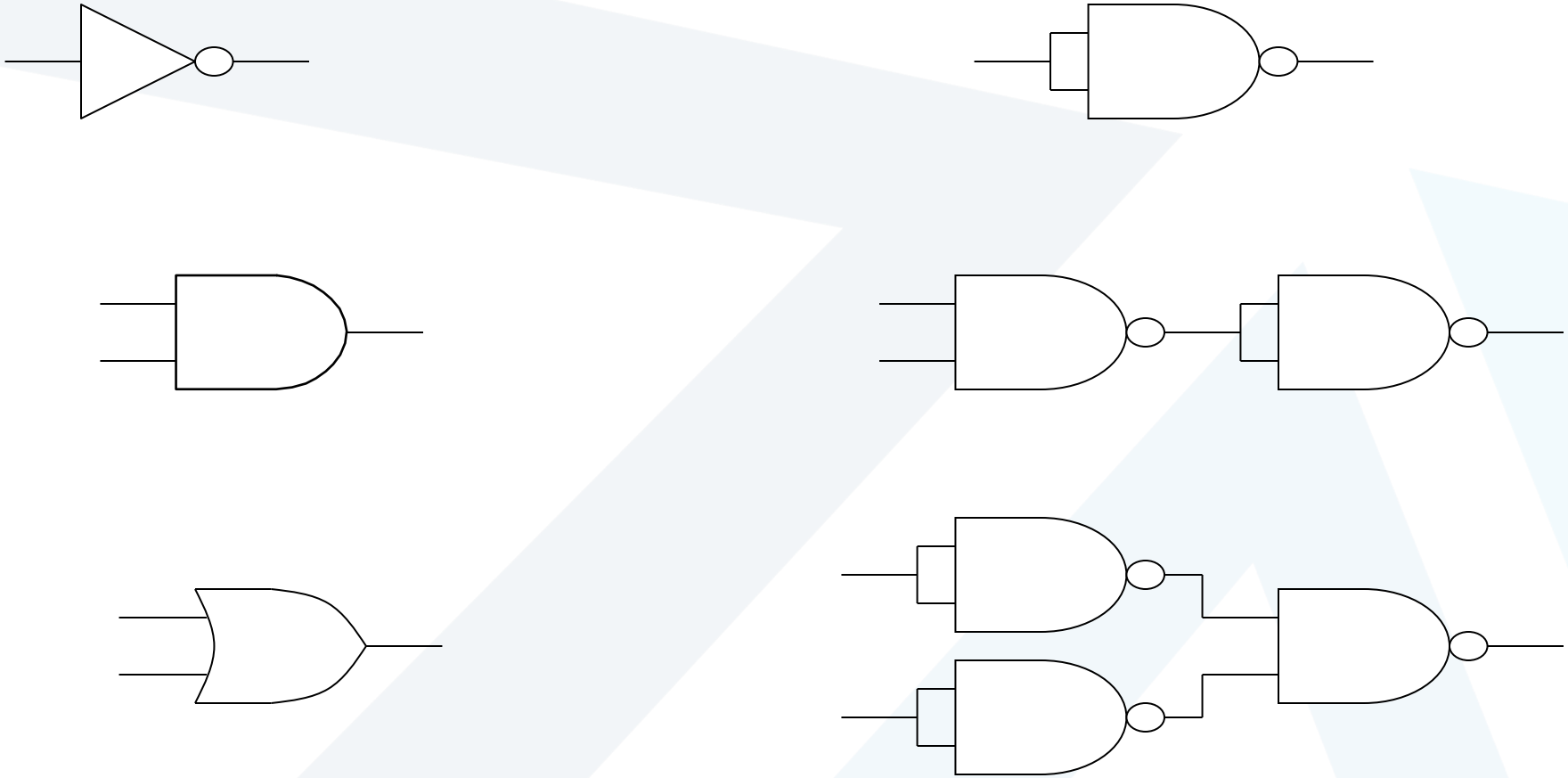
Logic Functions - Waveform View



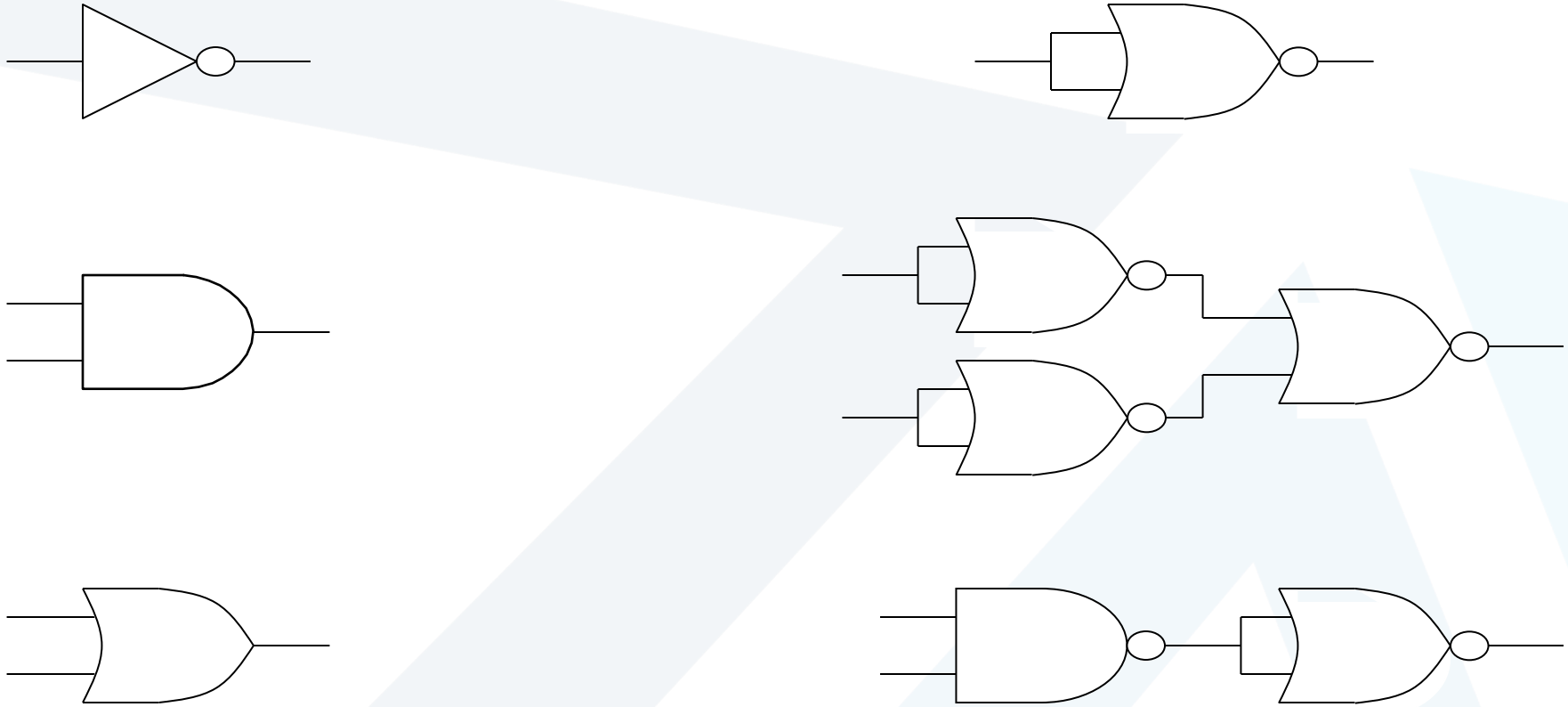
Minimal Logic Operator Sets

- AND , OR, NOT are all that's needed to express any combinational logic function as switching algebra expression
  - operators are all that were originally defined
- Two other minimal logic operator sets exist
  - Just NAND gates
  - Just NOR gates
- We can demonstrate how just NANDs or NORs can do AND, OR, NOT operations

NAND as a Minimal Set



NOR as a Minimal Set



Multi-input Logic Operators

- To this point all operators have been unary or binary
- AND, OR, NAND, NOR, XOR, XNOR can all be extended to any number of inputs
- Truth table constructed by evaluation of operation on successive pairs of inputs

$$(A + B + C + D) \equiv (((A + B) + C) + D)$$

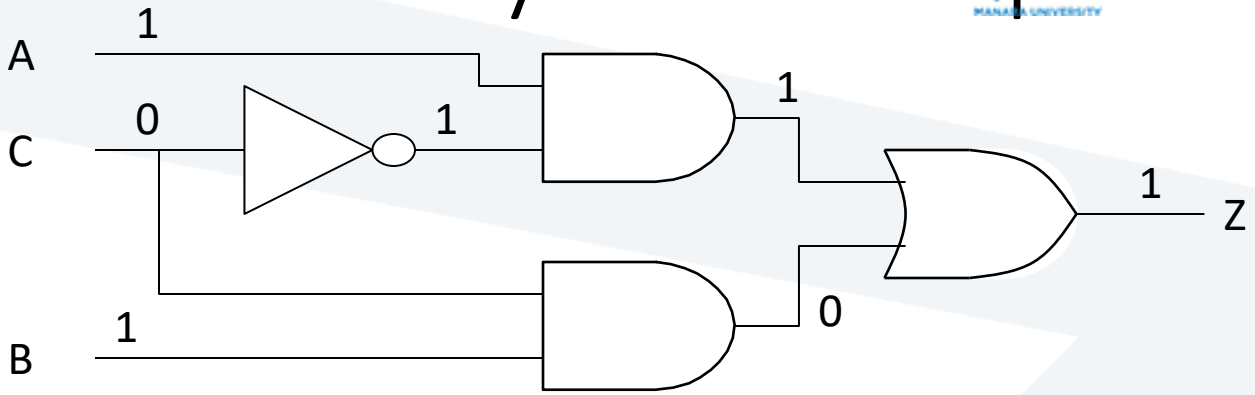
Combinational Circuit Analysis

- Combinational circuit analysis starts with a schematic and answers the following questions:
  - What is the truth table(s) for the circuit output function(s)
  - What is the logic expression(s) for the circuit output function(s)

Literal Analysis

- Literal analysis is process of manually assigning a set of values to the inputs, tracing the results, and recording the output values
  - For ‘n’ inputs there are 2^n possible input combinations
  - From input values, gate outputs are evaluated to form next set of gate inputs
  - Evaluation continues until gate outputs are circuit outputs
- Literal analysis only gives us the truth table

Literal Analysis - Example



A	B	C	Z
0	0	0	x
0	0	1	x
0	1	0	x
0	1	1	x
1	0	0	x
1	0	1	x
1	1	0	1
1	1	1	x

Assign input values

Determine gate outputs and propagate

Repeat until we reach output

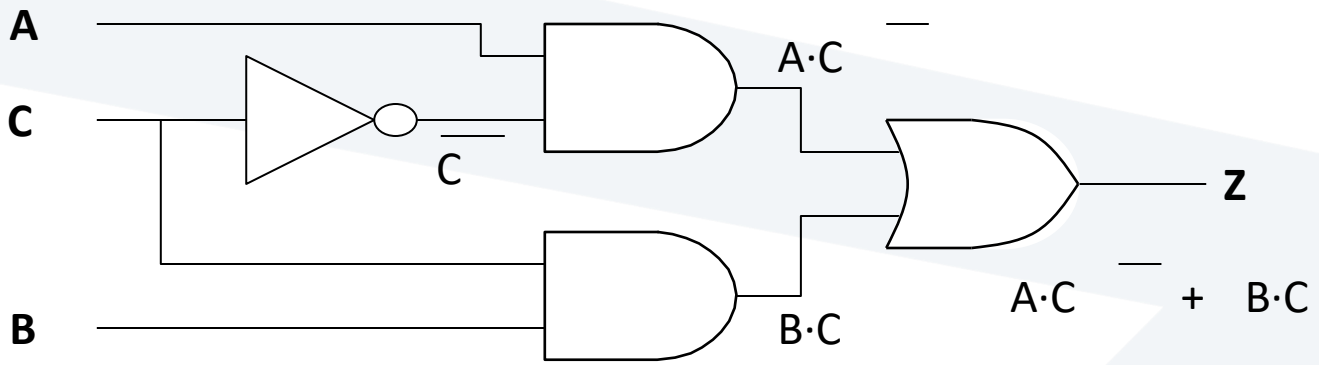
Symbolic Analysis

- Like literal analysis we start with the circuit diagram
  - Instead of assigning values, we determine gate output expressions instead
  - Intermediate expressions are combined in following gates to form complex expressions
  - We repeat until we have the output function and expression
- Symbolic analysis gives both the truth table and logic expression

Symbolic Analysis (cont.)

- Note that we are constructing the truth table as we go
  - truth table has a column for each intermediate gate output
  - intermediate outputs are combined in the truth table to generate the complex columns
- Symbolic analysis is more work but gives us complete information

Symbolic Analysis - Example



Generate intermediate expression

Create associated TT column

Repeat till output reached

A	B	C	$\overline{C}$	$A \cdot \overline{C}$	$B \cdot C$	$Z = A \cdot \overline{C} + B \cdot C$
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	1	0	1	1	0	1
1	1	1	0	0	1	1

Logic Expressions

- Terms and Definitions
  - Logic Expression - a mathematical formula consisting of logical operators and variables
  - Logic Operator - a function that gives a well defined output according to switching algebra
  - Logic Variable - a symbol representing the two possible switching algebra values of 0 and 1
  - Logic Literal - the values 0 and 1 or a logic variable or it's complement

Logic Expressions - Precedence

- Like standard algebra, switching algebra operators have a precedence of evaluation
  - NOT operations have the highest precedence
  - AND operations are next
  - OR operations are lowest
- Parentheses explicitly define the order of operator evaluation
  - If in doubt, USE PARENTHESES!

Canonical Forms

- Two standard (canonical) forms
  - Canonical sum form
    - AKA disjunctive normal form or sum-of-products
    - OR of AND terms
  - Canonical product form
    - AKA conjunctive normal form or product-of-sums
    - AND or OR terms
- In both forms, each first-level operator corresponds to one row of truth table
- 2nd-level operator combines 1st-level results

Canonical Forms (cont.)

Canonical Sum Form
Sum of Products (OR of AND terms)

$$F = \left(\overline{A} \cdot \overline{B} \cdot \overline{C}\right) + \left(\overline{A} \cdot B \cdot C\right) + \left(A \cdot B \cdot \overline{C}\right) + \left(A \cdot B \cdot C\right)$$

Minterms

Canonical Product Form
Product of Sums (AND of OR terms)

$$F = \left(A + B + \overline{C}\right) \cdot \left(A + \overline{B} + C\right) \cdot \left(\overline{A} + B + C\right) \cdot \left(\overline{A} + B + \overline{C}\right)$$

Maxterms

Canonical Sum Form

- Each AND (product) term is a Minterm
  - ANDed product of literals in which each variable appears exactly once, in true or complemented form (but not both!)
  - Each minterm has exactly one ‘1’ in the truth table
  - When minterms are ORed together each minterm contributes a ‘1’ to the final function

Minterms and Canonical Sum

A	B	C	Minterms	$\overline{A}\cdot\overline{B}\cdot\overline{C}$	m_0	m_3	m_6	m_7	F
0	0	0	m_0 =	$\overline{A}\cdot\overline{B}\cdot C$	1	0	0	0	1
0	0	1	m_1 =	$\overline{A}\cdot B\cdot\overline{C}$	0	0	0	0	0
0	1	0	m_2 =	$\overline{A}\cdot B\cdot C$	0	0	0	0	0
0	1	1	m_3 =	$A\cdot\overline{B}\cdot\overline{C}$	0	1	0	0	1
1	0	0	m_4 =	$A\cdot\overline{B}\cdot C$	0	0	0	0	0
1	0	1	m_5 =	$A\cdot B\cdot\overline{C}$	0	0	0	0	0
1	1	0	m_6 =	$A\cdot B\cdot C$	0	0	1	0	1
1	1	1	m_7 =		0	0	0	1	1

$$F = \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot B\cdot C + A\cdot B\cdot\overline{C} + A\cdot B\cdot C$$

$$F(A,B,C) = m_0 + m_3 + m_6 + m_7$$

$$F(A,B,C) = \sum m(0,3,6,7)$$

Canonical Product Form

- Each OR (sum) term is a Maxterm
  - ORed product of literals in which each variable appears exactly once, in true or complemented form (but not both!)
  - Each maxterm has exactly one ‘0’ in the truth table
  - When maxterms are ANDed together each maxterm contributes a ‘0’ to the final function

Maxterms and Canonical Product

A	B	C	Maxterms		M_1	M_2	M_4	M_5	F
0	0	0	M_0 =	$A+B+C$	1	1	1	1	1
0	0	1	M_1 =	$A+B+\overline{C}$	0	1	1	1	0
0	1	0	M_2 =	$A+\overline{B}+C$	1	0	1	1	0
0	1	1	M_3 =	$A+\overline{B}+\overline{C}$	1	1	1	1	1
1	0	0	M_4 =	$\overline{A}+B+C$	1	1	0	1	0
1	0	1	M_5 =	$\overline{A}+B+\overline{C}$	1	1	1	0	0
1	1	0	M_6 =	$\overline{A}+\overline{B}+C$	1	1	1	1	1
1	1	1	M_7 =	$\overline{A}+\overline{B}+\overline{C}$	1	1	1	1	1

$$F = (A+B+\overline{C})\cdot(A+\overline{B}+C)\cdot(\overline{A}+B+C)\cdot(\overline{A}+B+\overline{C})$$

$$F(A,B,C) = M_1 \cdot M_2 \cdot M_4 \cdot M_5$$

$$F(A,B,C) = \prod M(1,2,4,5)$$

Documenting Combinational Systems

- Schematic (circuit) diagrams are a graphical representation of the combinational circuit
  - Best practice is to organize drawing so data flows left to right, control, top to bottom
- Two conventions exist to denote circuit signal connections
  - Only ‘T’ intersections are connections; others just cross over
  - Solid dots ‘●’ are placed at connection points (This is preferred)

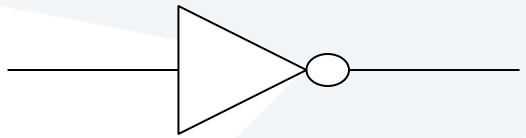
Signal Active States and Bubbles

- Primarily applies to control signals, used to denote when a condition is active or enabled
  - *Active State* - signal state (0 or 1) that indicates the assertion of some condition or action
    - Also called the excitation state
    - A signal is asserted when it is in the active state
    - A signal is negated when it is in the inactive state
    - Active-1 (active high) is when active state is logic 1
    - Active-0 (active low) is when active state is logic 0
  - Symbol pins without bubbles denote active-1
  - Symbol pins with bubbles denote active-0

Active States and Bubbles (cont.)

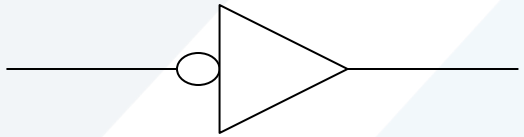
Inverter (NOT operator) has two different forms

Input asserted
active-1



Output asserted
active-0

Input asserted
active-0



Output asserted
active-1

Alternative Symbols

- The NOT example can be extended to all logic gates
  - Each logic gate has two equivalent symbols
    - The one we've seen so far for active-1 inputs
    - The alternate for active-0 inputs
  - In each case the gate operates the same
    - The only difference is how we interpret the values

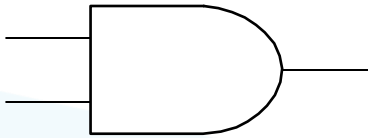
https://manara.edu.sy/

AND Gate Alternate Symbol

X	Y	X·Y
0	0	0
0	1	0
1	0	0
1	1	1

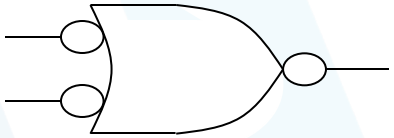
Active-1

X	Y	X·Y
F	F	F
F	T	F
T	F	F
T	T	T



Active-0

X	Y	X+Y
T	T	T
T	F	T
F	T	T
F	F	F

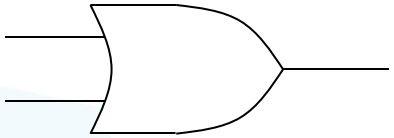


OR Gate Alternate Symbol

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

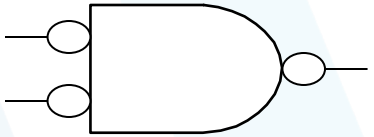
Active-1

X	Y	X+Y
F	F	F
F	T	T
T	F	T
T	T	T

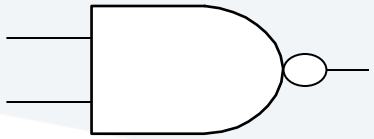
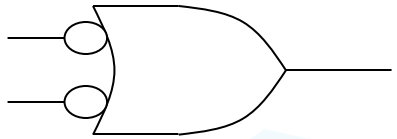
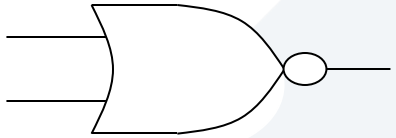
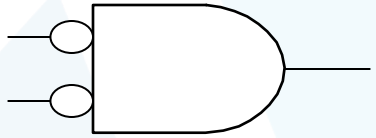
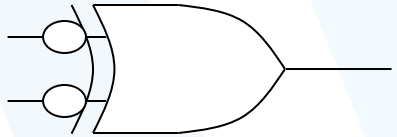
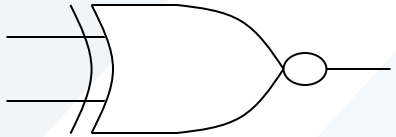


Active-0

X	Y	X·Y
T	T	T
T	F	F
F	T	F
F	F	F



Other Gate Alternative Symbols

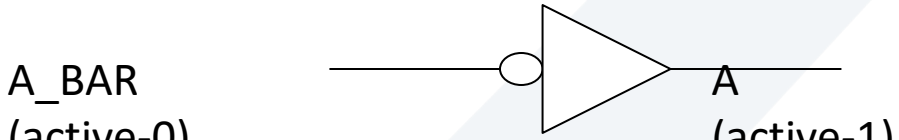
	Active-1	Active-0
NAND		
NOR		
XOR		
XNOR		

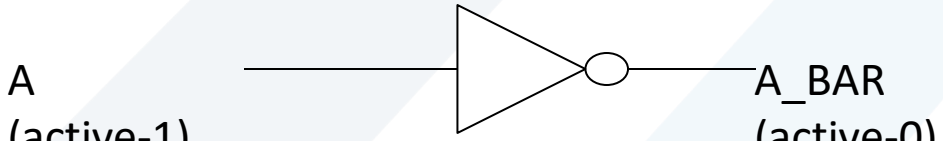
Signal Naming Conventions

- The problem is how to distinguish between active-1 and active-0 signals.
  - Barring a signal name to designate active-0 is not recommended
    - Is A act $\overline{A}$ or NOT A??
  - Use suffix of '_0'; (i.e A_0) after signal name
  - Use suffix of '_LO'
  - Use suffix of '_BAR' (Preferred)

Signal Naming (cont.)

Active-0 signal naming and symbol bubbles require some thought to interpret properly

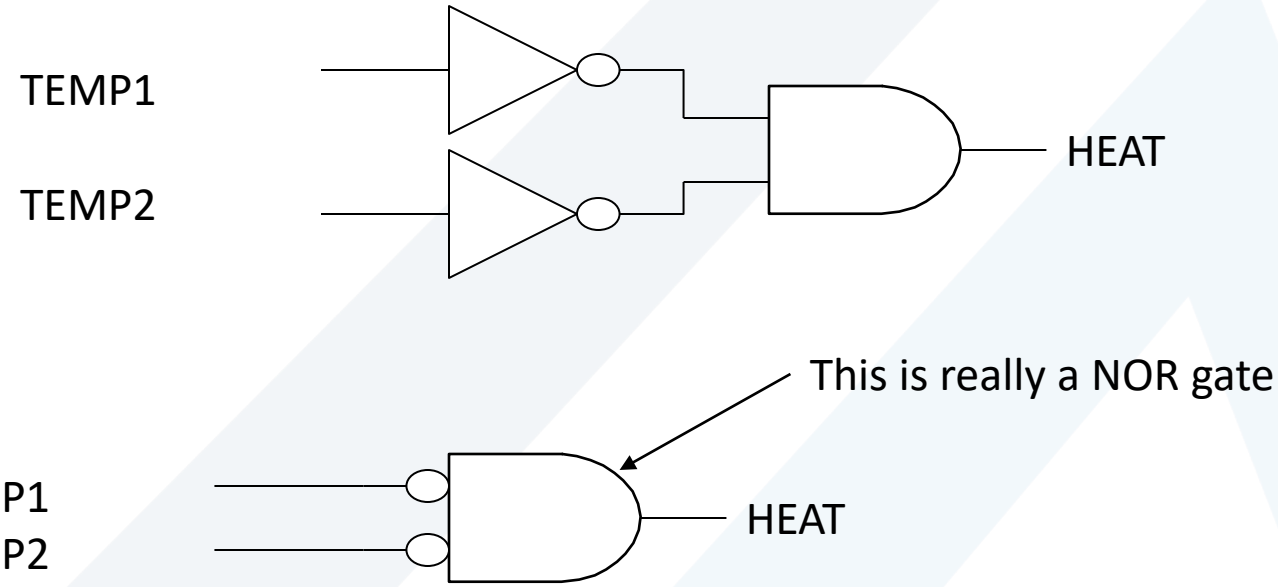




Naming and Alternate Symbols

Proper active-0 signal naming and usage of alternate symbols can clarify the circuit intent

$$HEAT = \overline{TEMP1} \cdot \overline{TEMP2}$$



جَامعة
المَنارة
MANARA UNIVERSITY

Combinational Systems

Definition and Analysis

Overview

- Combinational systems overview
- Switching algebra
- Logic operations
- Logic mnemonics
- Minimal sets of logic operations
- Multi-input logic operations
- Combinational systems analysis
- Logic expressions
- Documenting combinational systems

Combinational Logic

- One or more digital signal inputs
- One or more digital signal outputs
- Outputs are only functions of current input values (ideal) plus logic propagation delays

$$O_1(t+\Delta t) = F_1(I_1(t),\ldots I_m(t))$$
$$\vdots$$
$$O_n(t+\Delta t) = F_n(I_1(t),\ldots I_m(t))$$

Combinational Logic (cont.)

- Combinational logic has no memory!
  - Outputs are only function of current input combination
  - Nothing is known about past events
  - Repeating a sequence of inputs always gives the same output sequence
- Sequential logic (covered later) does have memory
  - Repeating a sequence of inputs can result in an entirely different output sequence

Combinational Logic Example

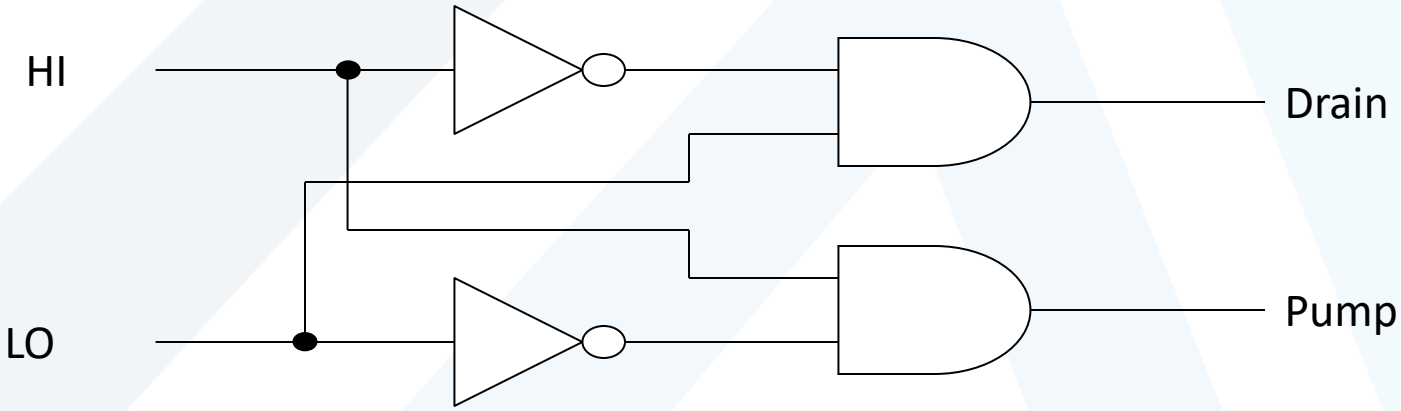
- Circuit controls the level of fluid in a tank
  - inputs are:
    - HI - 1 if fluid level is too high, 0 otherwise
    - LO - 1 if fluid level is too low, 0 otherwise
  - outputs are:
    - Pump - 1 to pump fluid into tank, 0 for pump off
    - Drain - 1 to open tank drain, 0 for drain closed
  - input to output relationship is described by a truth table

Combinational Logic Example (cont.)

HI	LO	Pump	Drain
0	0	0	0
0	1	1	0
1	0	0	1
1	1	x	x

Tank level is OK
Low level, pump more in
High level, drain some out
inputs cannot occur

Schematic Representation



Switching Algebra

- Based on Boolean Algebra
  - Developed by George Boole in 1854
  - Formal way to describe logic statements and determine truth of statements
- Only has two-values domain (0 and 1)
- Huntington's Postulates define underlying assumptions

Huntington's Postulates

- Closure

  If X and Y are in set (0,1) then operations X+Y and X ·Y are also in set (0,1)

- Identity

  $X + 0 = X$ $\qquad$ $X \cdot 1 = X$

- Commutative

  $X + Y = Y + X$ $\qquad$ $X \cdot Y = Y \cdot X$

Huntington's Postulates (cont.)

- Distributive

  $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$

  $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

- Complement

$$X + \overline{X} = 1$$

$$X \cdot \overline{X} = 0$$

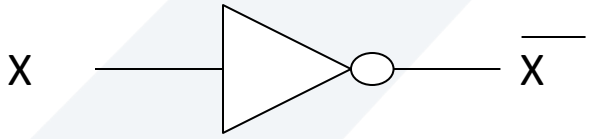
Note that for each property, one form is the dual of the other;

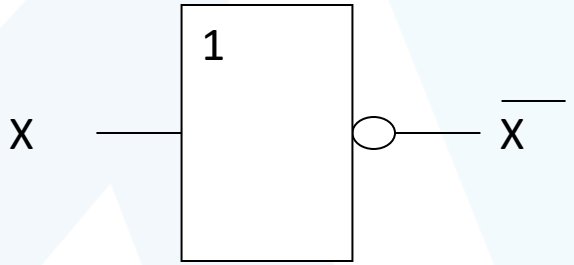
(0s to 1s, 1s to 0s, ·s to +s, +s to ·s)

Switching Algebra Operations - Not

- Unary complement or inversion operation
- Usually shown as overbar (X), other forms are ˜X, X’

X	$\overline{X}$
0	1
1	0

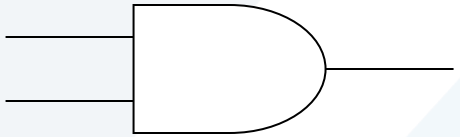


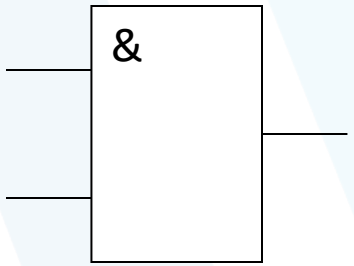


Switching Algebra Operations - AND

- Also known as the conjunction operation; output is true (1) only if all inputs are true
- Algebraic operators are '·', '&', '∧'

X	Y	X·Y
0	0	0
0	1	0
1	0	0
1	1	1

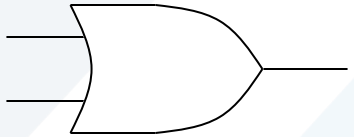


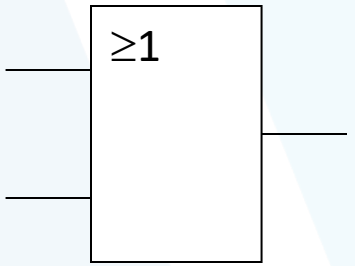


Switching Algebra Operations - OR

- Also known as the disjunction operation; output is true (1) if any input is true
- Algebraic operators are ‘+’, ‘|’, ‘$\vee$’

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

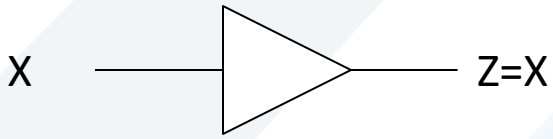


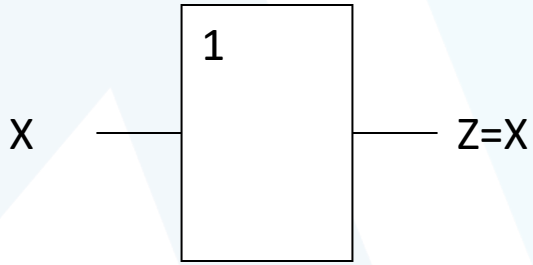


Additional Logic Operations

- For two inputs, there are 16 ways we can assign output values
  - Besides AND and OR, four others are useful
- The unary Buffer operation is useful in the real world

X	Z=X
0	0
1	1

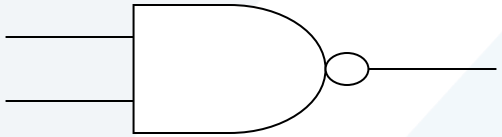


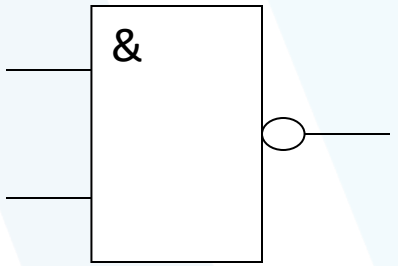


Additional Logic Operations - NAND

- NAND (NOT - AND) is the complement of the AND operation

X	Y	$\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

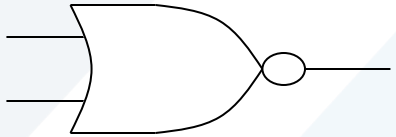


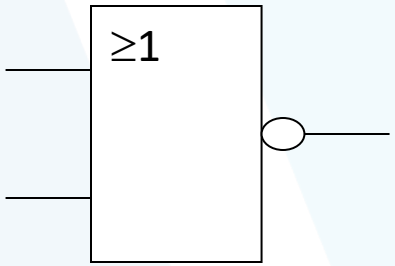


Additional Logic Operations - NOR

- NOR (NOT - OR) is the complement of the OR operation

X	Y	$\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

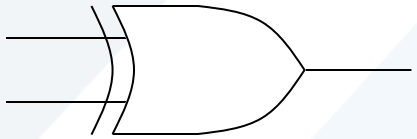


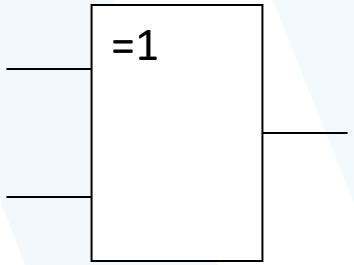


Additional Logic Operations -XOR

- Exclusive OR is similar to the inclusive OR (AKA OR) except output is 0 for 1,1 inputs
- Alternatively the output is 1 when modulo 2 input sum is equal to 1

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

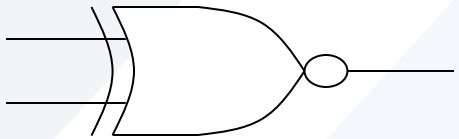


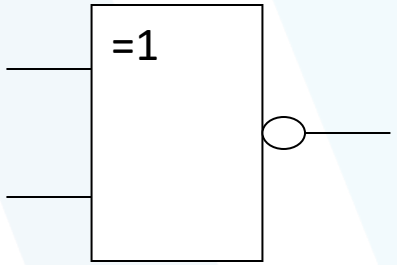


Additional Logic Operations - XNOR

- Exclusive NOR is the complement of the XOR operation
- Alternatively the output is 1 when modulo 2 input sum is not equal to 1

X	Y	$\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	1

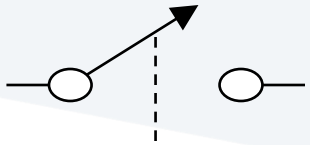




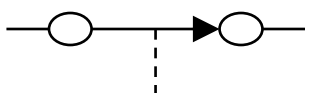
Logic Mnemonics

- Another method for understanding logic operations is to represent them as switches
  - Output = 1 is there is a current path thru switch
  - Practical application in relay logic
- Another method is Venn diagrams
  - Shaded area is where output is 1 or true

Switch/Venn Equivalents

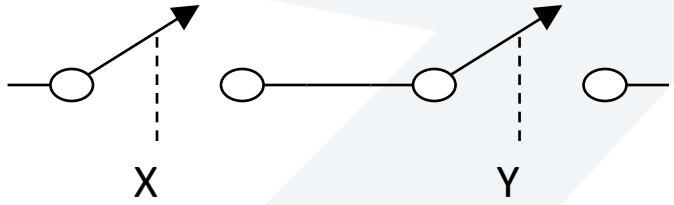


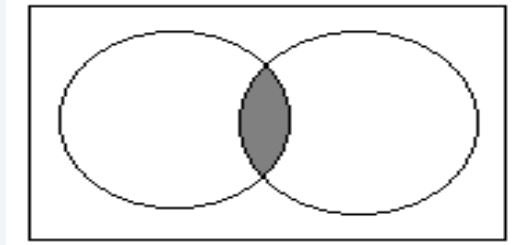
Input = 0
switch open



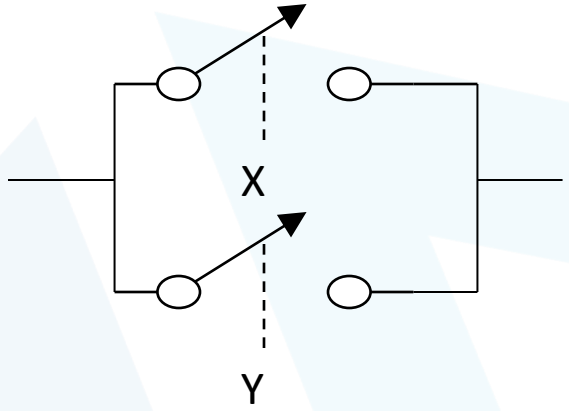
Input = 0
switch closed

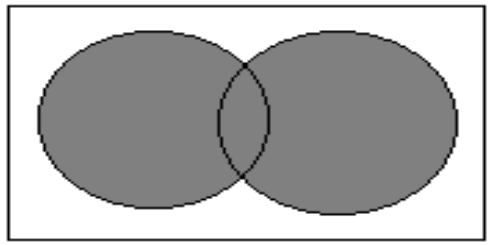
AND



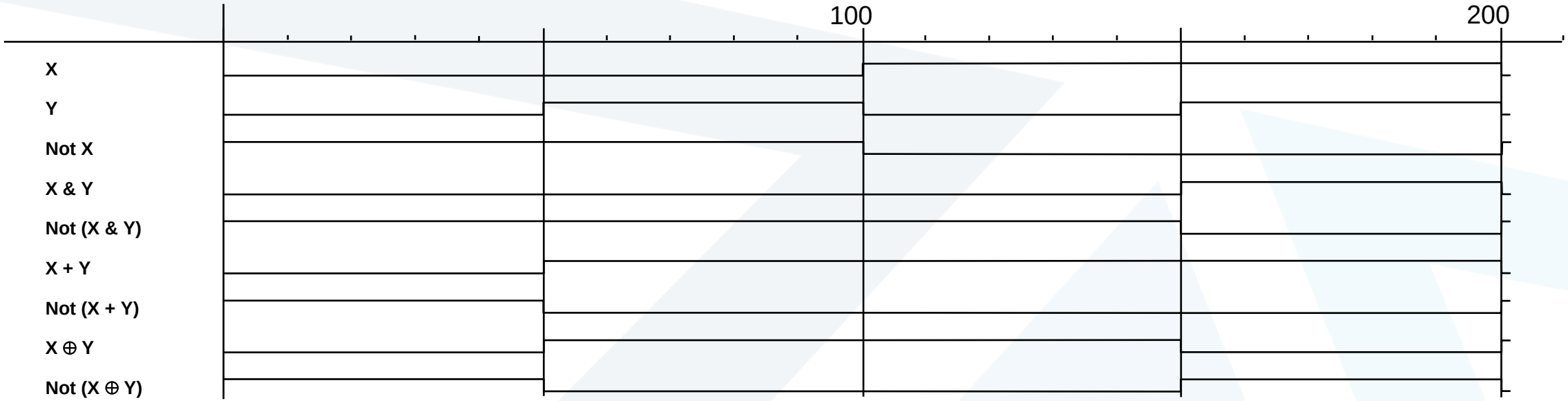


OR





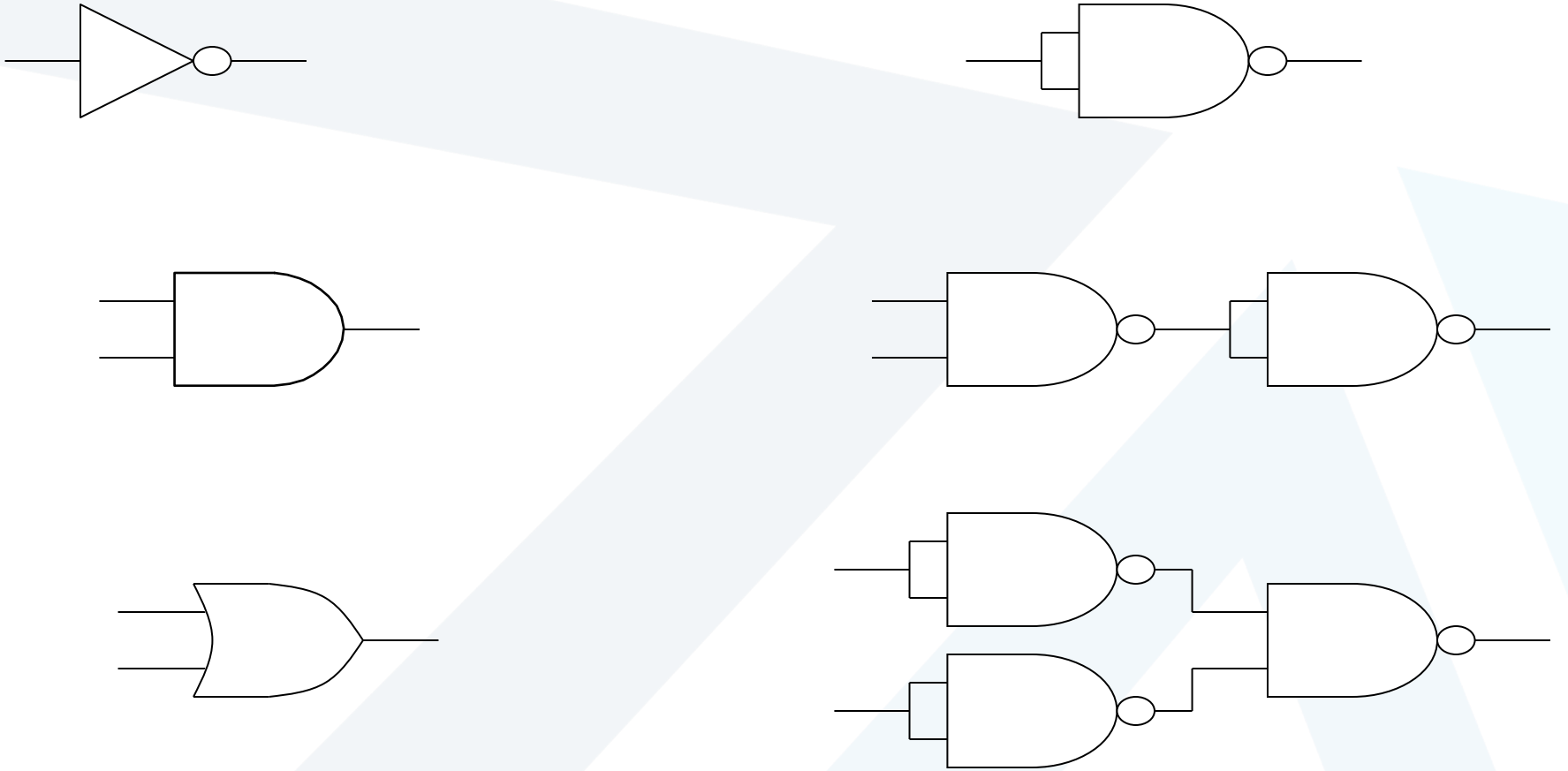
Logic Functions - Waveform View



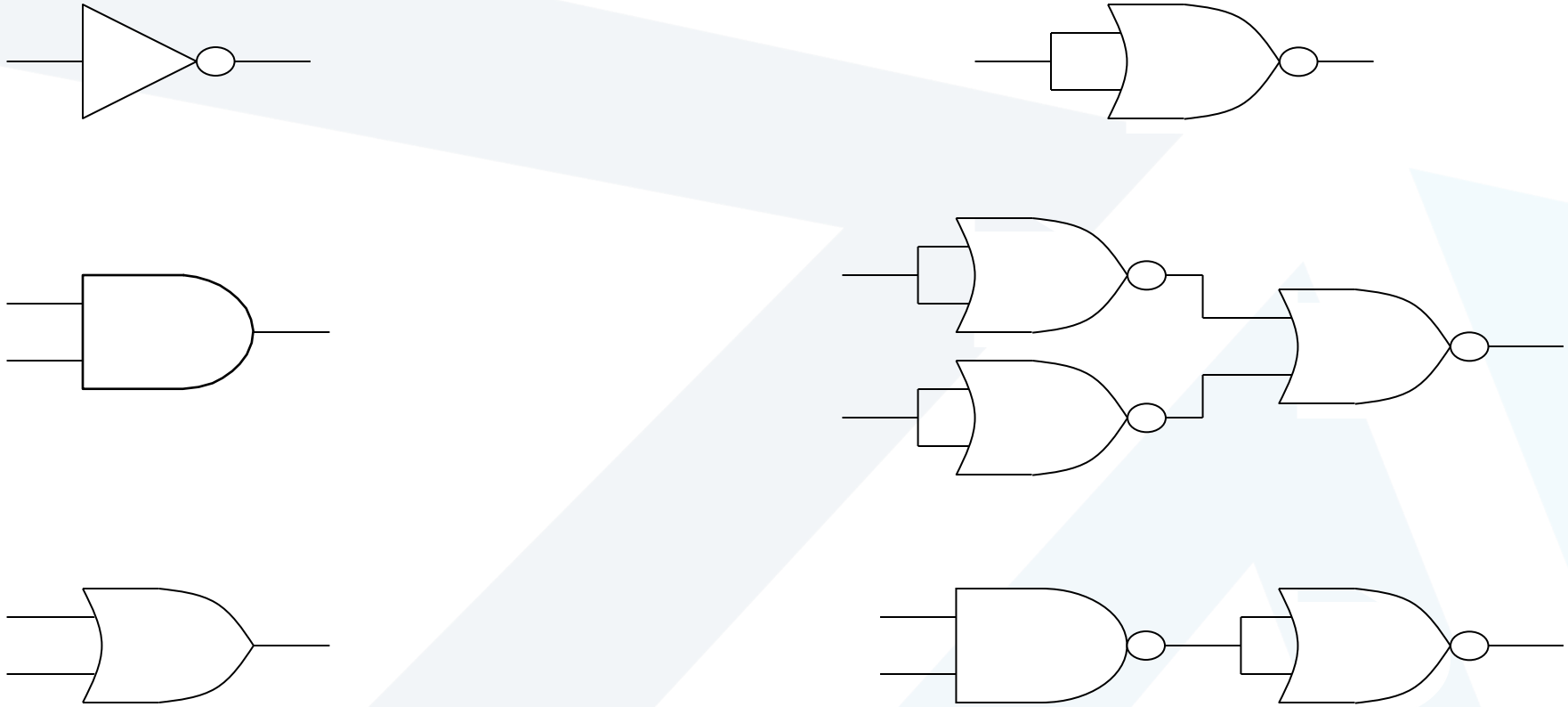
Minimal Logic Operator Sets

- AND , OR, NOT are all that's needed to express any combinational logic function as switching algebra expression
  - operators are all that were originally defined
- Two other minimal logic operator sets exist
  - Just NAND gates
  - Just NOR gates
- We can demonstrate how just NANDs or NORs can do AND, OR, NOT operations

NAND as a Minimal Set



NOR as a Minimal Set



Multi-input Logic Operators

- To this point all operators have been unary or binary
- AND, OR, NAND, NOR, XOR, XNOR can all be extended to any number of inputs
- Truth table constructed by evaluation of operation on successive pairs of inputs

$$(A + B + C + D) \equiv (((A + B) + C) + D)$$

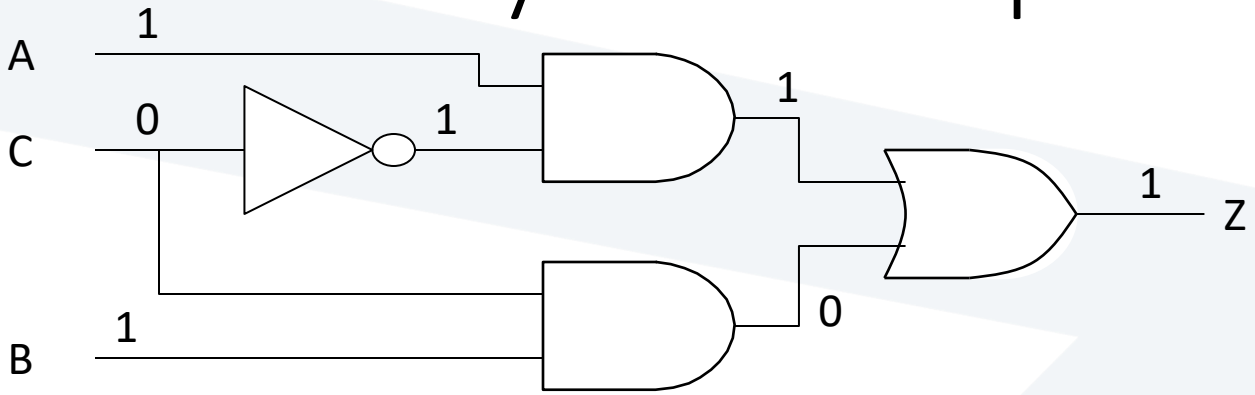
Combinational Circuit Analysis

- Combinational circuit analysis starts with a schematic and answers the following questions:
  - What is the truth table(s) for the circuit output function(s)
  - What is the logic expression(s) for the circuit output function(s)

Literal Analysis

- Literal analysis is process of manually assigning a set of values to the inputs, tracing the results, and recording the output values
  - For ‘n’ inputs there are 2^n possible input combinations
  - From input values, gate outputs are evaluated to form next set of gate inputs
  - Evaluation continues until gate outputs are circuit outputs
- Literal analysis only gives us the truth table

Literal Analysis - Example



A	B	C	Z
0	0	0	x
0	0	1	x
0	1	0	x
0	1	1	x
1	0	0	x
1	0	1	x
1	1	0	1
1	1	1	x

Assign input values

Determine gate outputs and propagate

Repeat until we reach output

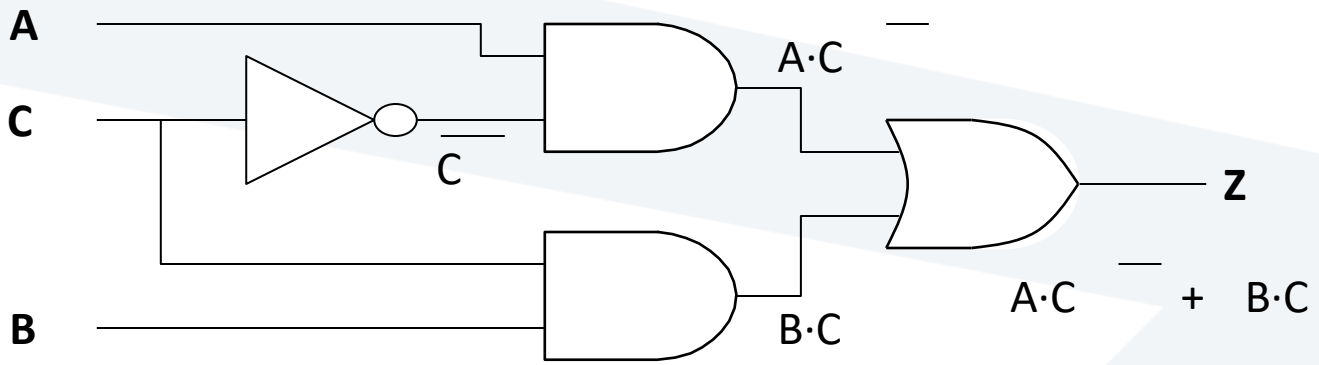
Symbolic Analysis

- Like literal analysis we start with the circuit diagram
  - Instead of assigning values, we determine gate output expressions instead
  - Intermediate expressions are combined in following gates to form complex expressions
  - We repeat until we have the output function and expression
- Symbolic analysis gives both the truth table and logic expression

Symbolic Analysis (cont.)

- Note that we are constructing the truth table as we go
  - truth table has a column for each intermediate gate output
  - intermediate outputs are combined in the truth table to generate the complex columns
- Symbolic analysis is more work but gives us complete information

Symbolic Analysis - Example



Generate intermediate expression

Create associated TT column

Repeat till output reached

A	B	C	$\overline{C}$	$A \cdot \overline{C}$	$B \cdot C$	$Z = A \cdot \overline{C} + B \cdot C$
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	1	0	1	1	0	1
1	1	1	0	0	1	1

Logic Expressions

- Terms and Definitions
  - Logic Expression - a mathematical formula consisting of logical operators and variables
  - Logic Operator - a function that gives a well defined output according to switching algebra
  - Logic Variable - a symbol representing the two possible switching algebra values of 0 and 1
  - Logic Literal - the values 0 and 1 or a logic variable or it's complement

Logic Expressions - Precedence

- Like standard algebra, switching algebra operators have a precedence of evaluation
  - NOT operations have the highest precedence
  - AND operations are next
  - OR operations are lowest
- Parentheses explicitly define the order of operator evaluation
  - If in doubt, USE PARENTHESES!

Canonical Forms

- Two standard (canonical) forms
  - Canonical sum form
    - AKA disjunctive normal form or sum-of-products
    - OR of AND terms
  - Canonical product form
    - AKA conjunctive normal form or product-of-sums
    - AND or OR terms
- In both forms, each first-level operator corresponds to one row of truth table
- 2nd-level operator combines 1st-level results

Canonical Forms (cont.)

Canonical Sum Form
Sum of Products (OR of AND terms)

$$F = \left(\overline{A} \cdot \overline{B} \cdot \overline{C}\right) + \left(\overline{A} \cdot B \cdot C\right) + \left(A \cdot B \cdot \overline{C}\right) + \left(A \cdot B \cdot C\right)$$

Minterms

Canonical Product Form
Product of Sums (AND of OR terms)

$$F = \left(A + B + \overline{C}\right) \cdot \left(A + \overline{B} + C\right) \cdot \left(\overline{A} + B + C\right) \cdot \left(\overline{A} + B + \overline{C}\right)$$

Maxterms

Canonical Sum Form

- Each AND (product) term is a Minterm
  - ANDed product of literals in which each variable appears exactly once, in true or complemented form (but not both!)
  - Each minterm has exactly one ‘1’ in the truth table
  - When minterms are ORed together each minterm contributes a ‘1’ to the final function

Minterms and Canonical Sum

A	B	C	Minterms	$\overline{A}\cdot\overline{B}\cdot\overline{C}$	m_0	m_3	m_6	m_7	F
0	0	0	m_0 =	$\overline{A}\cdot\overline{B}\cdot C$	1	0	0	0	1
0	0	1	m_1 =	$\overline{A}\cdot B\cdot\overline{C}$	0	0	0	0	0
0	1	0	m_2 =	$\overline{A}\cdot B\cdot C$	0	0	0	0	0
0	1	1	m_3 =	$A\cdot\overline{B}\cdot\overline{C}$	0	1	0	0	1
1	0	0	m_4 =	$A\cdot\overline{B}\cdot C$	0	0	0	0	0
1	0	1	m_5 =	$A\cdot B\cdot\overline{C}$	0	0	0	0	0
1	1	0	m_6 =	$A\cdot B\cdot C$	0	0	1	0	1
1	1	1	m_7 =		0	0	0	1	1

$$F = \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot B\cdot C + A\cdot B\cdot\overline{C} + A\cdot B\cdot C$$

$$F(A, B, C) = m_0 + m_3 + m_6 + m_7$$

$$F(A, B, C) = \sum m(0, 3, 6, 7)$$

Canonical Product Form

- Each OR (sum) term is a Maxterm
  - ORed product of literals in which each variable appears exactly once, in true or complemented form (but not both!)
  - Each maxterm has exactly one ‘0’ in the truth table
  - When maxterms are ANDed together each maxterm contributes a ‘0’ to the final function

Maxterms and Canonical Product

A	B	C	Maxterms		M_1	M_2	M_4	M_5	F
0	0	0	M_0 =	$A + B + C$	1	1	1	1	1
0	0	1	M_1 =	$A + B + \overline{C}$	0	1	1	1	0
0	1	0	M_2 =	$A + \overline{B} + C$	1	0	1	1	0
0	1	1	M_3 =	$A + \overline{B} + \overline{C}$	1	1	1	1	1
1	0	0	M_4 =	$\overline{A} + B + C$	1	1	0	1	0
1	0	1	M_5 =	$\overline{A} + B + \overline{C}$	1	1	1	0	0
1	1	0	M_6 =	$\overline{A} + \overline{B} + C$	1	1	1	1	1
1	1	1	M_7 =	$\overline{A} + \overline{B} + \overline{C}$	1	1	1	1	1

$$F = \left(A + B + \overline{C}\right) \cdot \left(A + \overline{B} + C\right) \cdot \left(\overline{A} + B + C\right) \cdot \left(\overline{A} + B + \overline{C}\right)$$

$$F(A, B, C) = M_1 \cdot M_2 \cdot M_4 \cdot M_5$$

$$F(A, B, C) = \prod M(1, 2, 4, 5)$$

Documenting Combinational Systems

- Schematic (circuit) diagrams are a graphical representation of the combinational circuit
  - Best practice is to organize drawing so data flows left to right, control, top to bottom
- Two conventions exist to denote circuit signal connections
  - Only ‘T’ intersections are connections; others just cross over
  - Solid dots ‘●’ are placed at connection points (This is preferred)

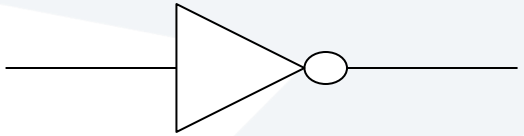
Signal Active States and Bubbles

- Primarily applies to control signals; used to denote when a condition is active or enabled
  - *Active State* - signal state (0 or 1) that indicates the assertion of some condition or action
    - Also called the excitation state
    - A signal is asserted when it is in the active state
    - A signal is negated when it is in the inactive state
    - Active-1 (active high) is when active state is logic 1
    - Active-0 (active low) is when active state is logic 0
  - Symbol pins without bubbles denote active-1
  - Symbol pins with bubbles denote active-0

Active States and Bubbles (cont.)

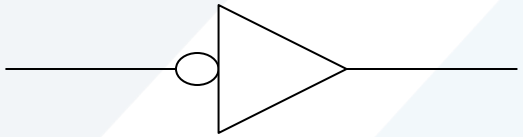
Inverter (NOT operator) has two different forms

Input asserted
active-1



Output asserted
active-0

Input asserted
active-0



Output asserted
active-1

Alternative Symbols

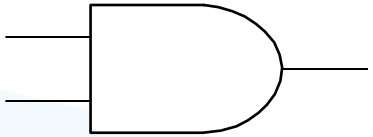
- The NOT example can be extended to all logic gates
  - Each logic gate has two equivalent symbols
    - The one we've seen so far for active-1 inputs
    - The alternate for active-0 inputs
  - In each case the gate operates the same
    - The only difference is how we interpret the values

AND Gate Alternate Symbol

X	Y	X·Y
0	0	0
0	1	0
1	0	0
1	1	1

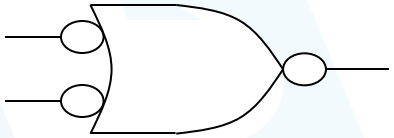
Active-1

X	Y	X·Y
F	F	F
F	T	F
T	F	F
T	T	T



Active-0

X	Y	X+Y
T	T	T
T	F	T
F	T	T
F	F	F

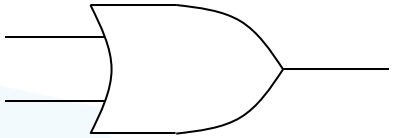


OR Gate Alternate Symbol

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

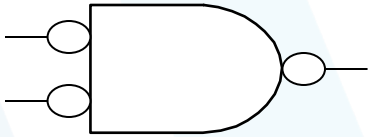
Active-1

X	Y	X+Y
F	F	F
F	T	T
T	F	T
T	T	T

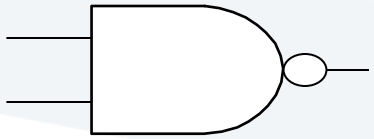
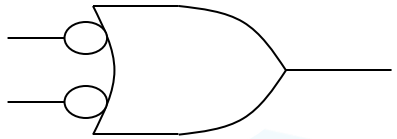
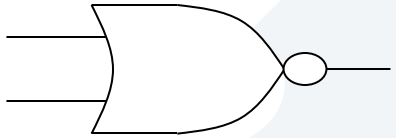
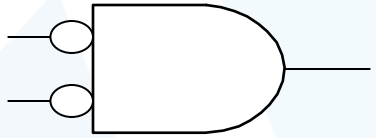
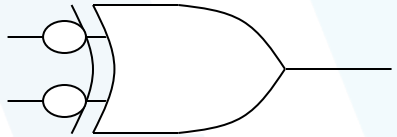
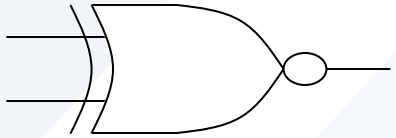


Active-0

X	Y	X·Y
T	T	T
T	F	F
F	T	F
F	F	F



Other Gate Alternative Symbols

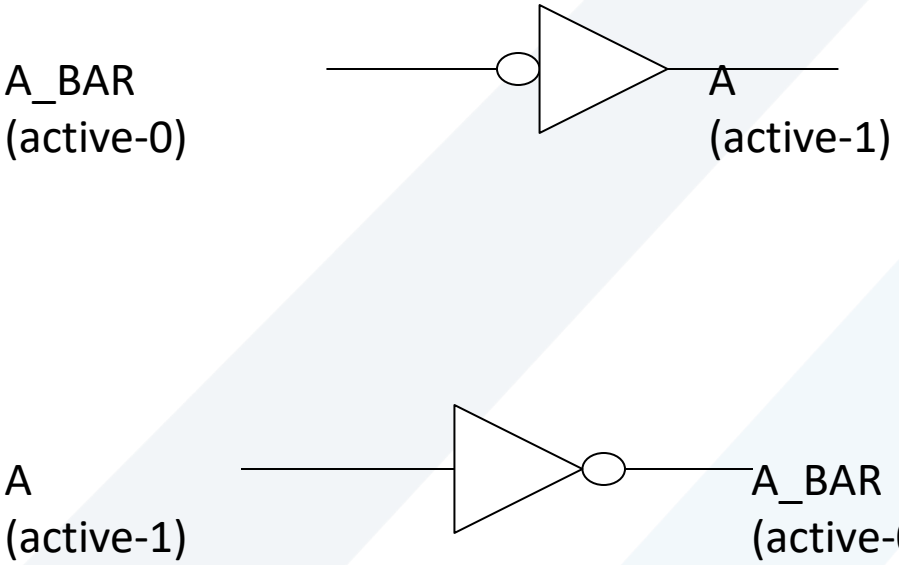
	Active-1	Active-0
NAND		
NOR		
XOR		
XNOR		

Signal Naming Conventions

- The problem is how to distinguish between active-1 and active-0 signals.
  - Barring a signal name to designate active-0 is not recommended
    - Is A act $\overline{A}$ or NOT A??
  - Use suffix of ‘_0’; (i.e A_0) after signal name
  - Use suffix of ‘_LO’
  - Use suffix of ‘_BAR’ (Preferred)

Signal Naming (cont.)

Active-0 signal naming and symbol bubbles require some thought to interpret properly



Naming and Alternate Symbols

Proper active-0 signal naming and usage of alternate symbols can clarify the circuit intent

$$\mathrm{HEAT} = \overline{\mathrm{TEMP1}} \cdot \overline{\mathrm{TEMP2}}$$