ENGN4213/6213
Digital Systems and Microprocessors

Semester 1, 2023

Week 1, Lecture A:

Part 2: Revision of Combinational Logic Design

Australian
National
University

Contents to Revise

- MOS transistors and the CMOS logic family
- Combinational logic:
  - Boolean algebra,
  - Logic functions and basic theorems, truth tables
  - Sum of products, minterms
  - Circuit minimization, Karnaugh maps
- Number systems knowledge

Resources

- Your ENGN2218 notes (if/as required)
- Wakerly (5th edition) Ch. 1 (book intro)
- Wakerly Ch. 14.1: MOS, CMOS
- Wakerly Ch. 14.2- 14.3: electrical behavior of MOS
- Wakerly Ch. 3.1-3.3: combinational logic
- Wakerly Ch. 2.1-2.9: number systems
- Pre-recorded videos lecture: VL1: Non-ideal behaviour
- ENGN3213 old Reading Brick from 2008 (check wattle) has nice, easy-to-read, sections on number systems, MOS transistors and combinational logic.

Digital circuits

- Digital systems can be analysed in terms of logic values: 0/1, LOW/HIGH, FULL/EMPTY, ON/OFF, etc.
- In electronic circuits it is common to associate a **LOW voltage with a logic 0** and **a HIGH voltage with a logic 1** (this is called *positive* logic).
- Transistors can be made to switch between the extreme operating points (ON to OFF and vice versa). This forms the *basis for the design of digital circuits* which only use two voltage states.
  - This course focus on Complementary Metal-Oxide Semiconductor (CMOS) - based digital integrated circuit technologies.
- **Logic families:** clever arrangements of transistors allowing us to create circuits whose I/O behaviour corresponds to particular logic functions.
  - Truth tables: describe a simple circuit in terms of an input/output representation
  - Logic values are represented by distinct voltages.

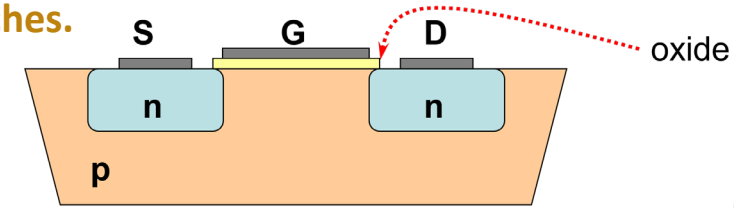
Digital circuits

CMOS inverter

Background: Metal Oxide Semiconductor (MOS) devices

- Three major MOS technology families:
  - PMOS (p-channel devices) Technologies implement p-channel transistors by *diffusing p-type dopants into an n-type silicon substrate* to form the source and the drain.
  - NMOS (n-channel devices) Technologies implement n-channel transistors by *diffusing n-type dopants into a p-type silicon substrate* to form the source and the drain.
  - CMOS (Complementary MOS) is a MOS technology in which both p-channel and n-channel devices are fabricated on the same die.
- MOSFET devices behave like **voltage-controlled switches.**

NMOS device:

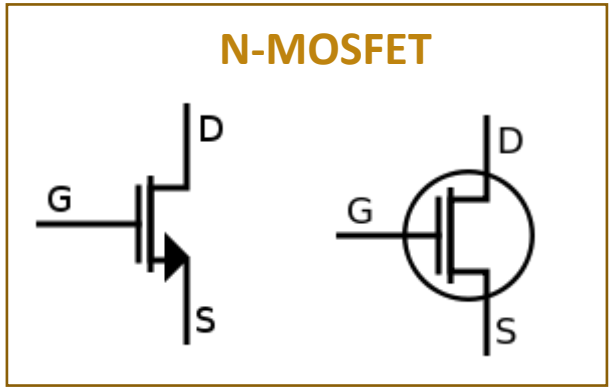


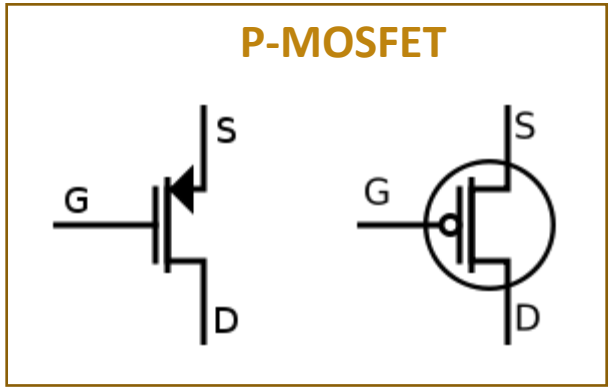
Digital circuits

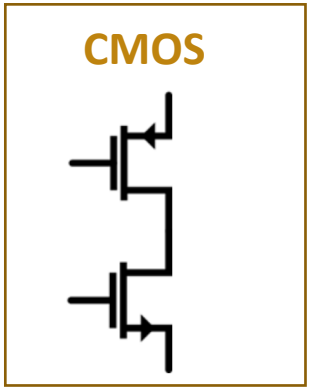
CMOS inverter

Background: Metal Oxide Semiconductor (MOS) devices

– Three major MOS technology families: *PMOS*, *NMOS*, *CMOS*







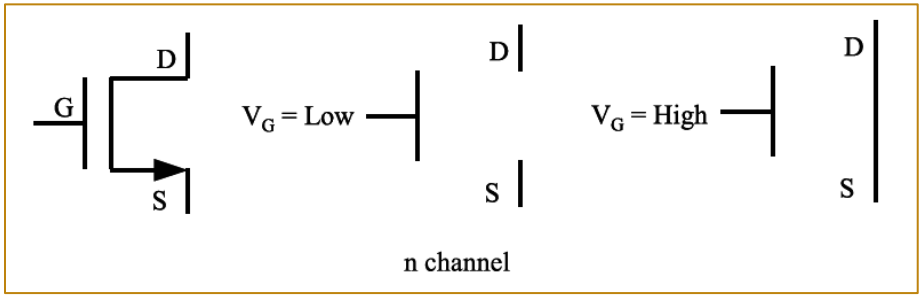
(variations of these symbols are also used in the literature)

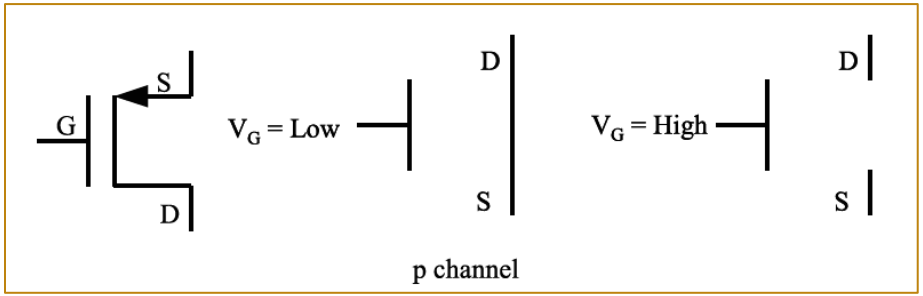
Digital circuits

CMOS inverter

Background: Metal Oxide Semiconductor (MOS) devices

MOSFET devices behave like **voltage-controlled switches**: *The gate voltage controls the resistance between source and drain.*

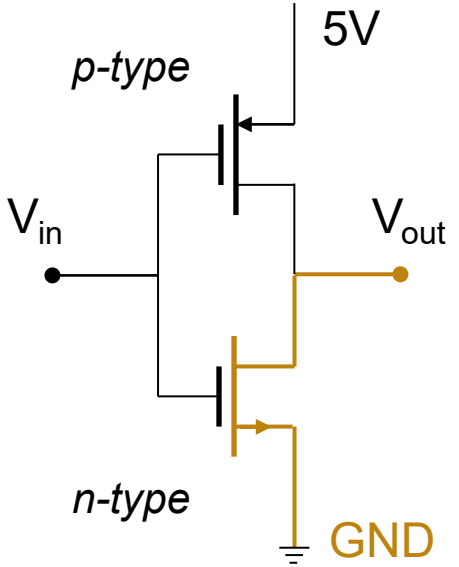




Digital circuits

CMOS inverter

CMOS Inverter: *basic building block of a CMOS device*



If $\boldsymbol{V_{in}}$ **is 5 V (HIGH)**

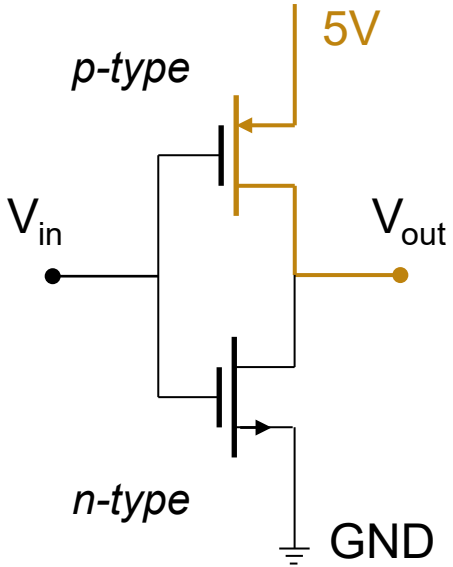
p-MOS has high resistance ($V_{GS} = 0\ V$) [open switch]
n-MOS has low resistance ($V_{GS} = 5\ V$) [closed switch]

Therefore, V_{out} is "shorted" to ground

$\boldsymbol{V_{out}}$ $\boldsymbol{=}$ **0 V (LOW)**

Digital circuits

CMOS inverter



If V_{in} **is 0V (LOW)**

p-MOS has low resistance ($V_{GS} = -5V$) [closed switch]
n-MOS has high resistance ($V_{GS} = 0V$) [open switch]

Therefore, V_{out} is "shorted" to the 5V rail

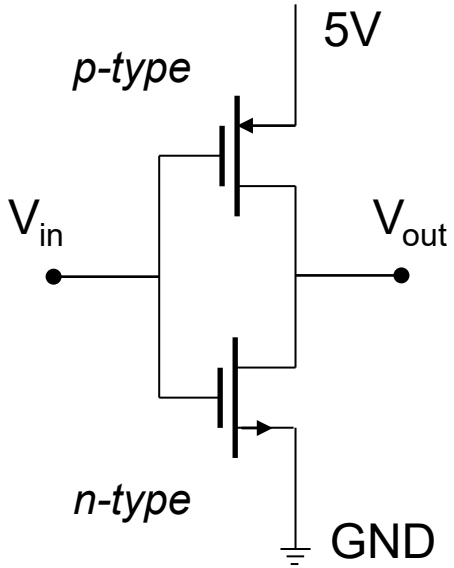
$V_{out} = 5V$ **(HIGH)**

Inverter Truth Table

IN	OUT
1	0
0	1

Digital circuits

CMOS inverter



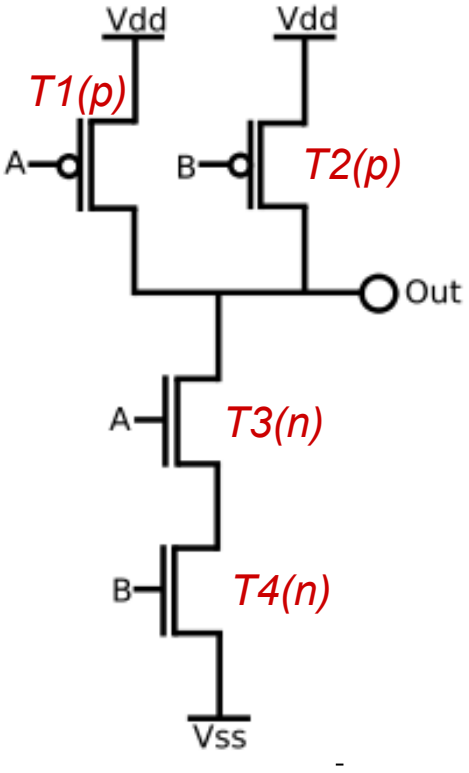
-

Other important general remarks about CMOS logic:

- The transistors are ideally operating at the **edge of the ohmic and saturation region** (i.e. maximum conductance)
- The circuit uses **no power to remain in a certain state**, this is a great advantage over BJTs
- Due to the oxide insulator, the **input impedance** of a CMOS logic circuit is very high (*higher noise immunity*)

Digital circuits

NAND Gate



A=LOW B=LOW ON: T1,T2 OFF: T3,T4 Out = HIGH	A=HIGH B=LOW ON: T2,T3 OFF: T1,T4 Out = HIGH	A=LOW B=HIGH ON: T1,T4 OFF: T2,T3 Out = HIGH	A=HIGH B=HIGH ON: T3,T4 OFF: T1,T2 Out = LOW

NAND
Truth Table

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

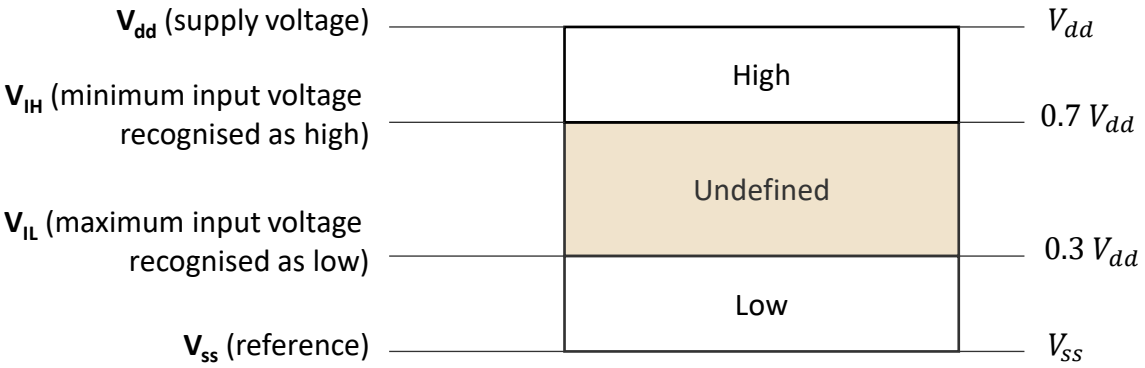
Digital circuits

Logic Levels (Wakerly Section 14.3)

- How are circuit voltages translated into logic levels?
  - A logic gate function as desired only in certain voltage
    - $V_{IH(min)}$: The *minimum* voltage level at the *input* to be recognized as *High*.
    - $V_{IL(max)}$: The *maximum* voltage level at the *input* to be recognized as *Low*.
    - $V_{OH(min)}$: The *minimum* voltage at the *output* when in state *High*.
    - $V_{OL(max)}$: The *maximum* voltage at the *output* when in state *Low*.

Digital circuits

Logic Levels (Wakerly Section 14.3)



- A logic gate will generate an output greater than $V_{OH(min)}$ or lower than $V_{OL(max)}$
- For typical CMOS, $V_{OH(min)} \approx 4.4\ V; V_{OL(max)} \approx 0.3V$. Voltages may differ amongst logic families
- For unambiguous communication of logic level from one gate to another we need

$$V_{IH(min)} < V_{OH(min)} \text{ and } V_{IL(max)} > V_{OL(max)}$$

Digital circuits

Logic Levels

- **Noise margin**: Measure of the extent to which a logic circuit can tolerate noise or unwanted spurious signals.

$$V_{NH} = V_{OH(min)} - V_{IH(min)}$$
$$V_{NL} = V_{IL(max)} - V_{OL(max)}$$

- V_{NH} and V_{NL} give the maximum noise voltage that will not cause incorrect or ambiguous communication of logic level in the High/Low states.

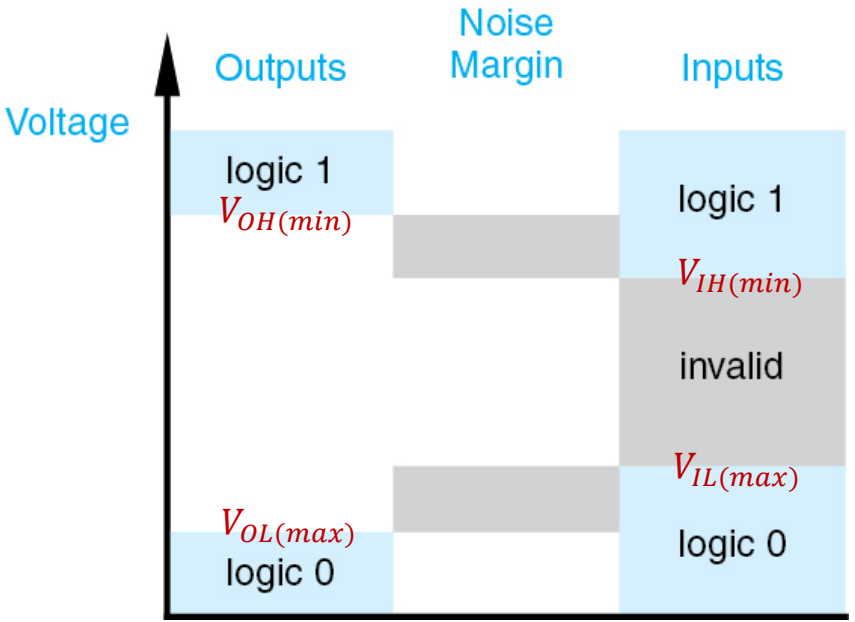


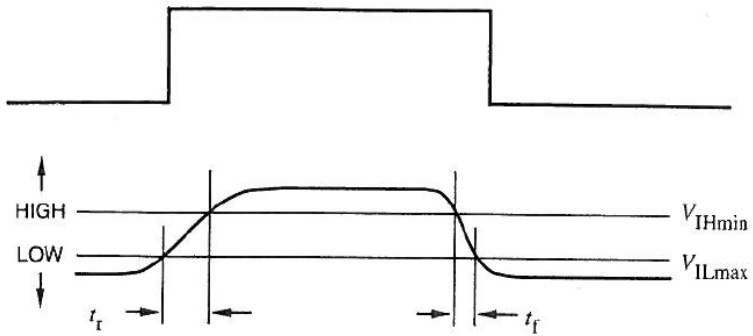
Figure 14.1 Chapter 14: Diigtal Circuits, John F. Wakerly, "*Digital Design, Principles and Practices*", 5th ed., Pearson/Prentice Hall.

Digital circuits

Transition Time

– Transition time is the *amount of time that the output of a logic circuit takes to change from one state to another*. It is ideally instantaneous, but in practice finite *rise* and *fall times* apply.

- Nonzero transition times are caused mainly by **capacitance in the circuit and/or the load** driven by the logic gate. They can be calculated with simple RC circuit analysis. (see Wakerly 14.4).
- Rise and fall times are calculated as the time it takes for the output to cross the *undefined* logic region. t_{rise} and t_{fall} are generally not equal.



Ideal transition

Real transition

Digital circuits

Transition Time

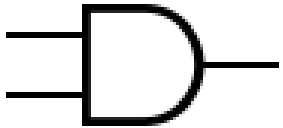
Implications of non-instantaneous transitions:

- The designer must consider what **load** they are driving and/or the **fanout** of their logic gates (fanout = *the number of logic gate inputs that can be satisfactorily driven by a logic gate output*) as greater loads will mean slower transitions.
- A chain of logic gates, each with a nonzero transition time, will introduce a **propagation delay**.
  - *Propagation delay of a signal path is the amount of time that it takes for a change in the input signal to produce a change in the output signal*. This is often defined as the time between the midpoint of input and output transitions.
  - If *different paths* in a digital circuit *have different propagation delays*, this can result in **hazards**: unwanted temporary transitions in the output.
- Slow transitions and propagation delays can affect the ability of a circuit to operate at a certain frequency.

Combinational Logic

Combinational Logic

Logic gates, symbols and truth tables



AND

$A \cdot B$

$A \wedge B$

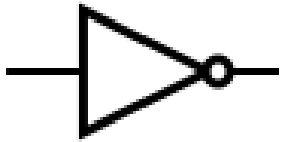
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR

$A + B$

$A \vee B$

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1



NOT

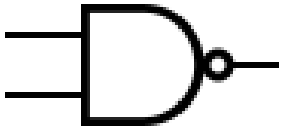
A'

$\overline{A}$

A	Out
0	1
1	0

Combinational Logic

Logic gates, symbols and truth tables



NAND

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

$$A \overline{\wedge} B$$

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR

$$\overline{A + B}$$

$$A \overline{\vee} B$$

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Combinational Logic

Logic gates, symbols and truth tables

XOR

$A \oplus B$

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

$\overline{A \oplus B}$

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Combinational Logic

Boolean algebra theorems

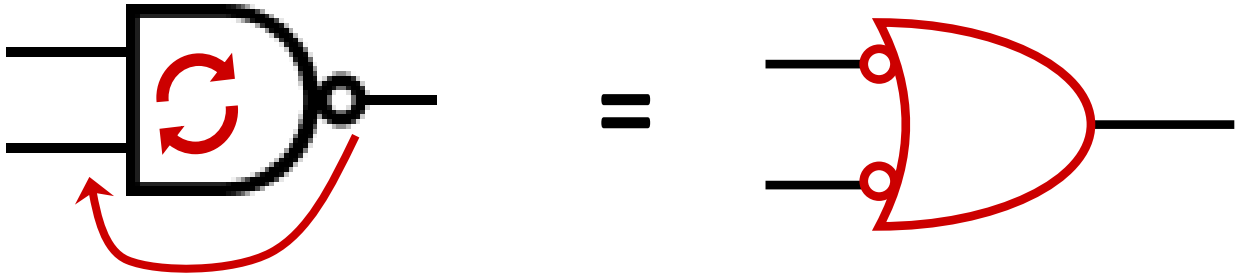
Theorem	OR version	AND version (dual)
Identity and Null Element:	$X + 0 = X$ $X + 1 = 1$	$X \bullet 1 = X$ $X \bullet 0 = 0$
Idempotent laws:	$X + X = X$	$X \bullet X = X$
Involution law:	$(X')' = X$	
Complementary laws:	$X + X' = 1$	$X \bullet X' = 0$
Commutative law:	$X + Y = Y + X$	$X \bullet Y = Y \bullet X$
Associative law:	$(X + Y) + Z = X + (Y + Z)$	$(X \bullet Y) \bullet Z = X \bullet (Y \bullet Z)$
Distributive law:	$X(Y + Z) = XY + XZ$	$X + Y \bullet Z = (X + Y) \bullet (X + Z)$
De Morgan's Theorem	$(X + Y + Z + \ldots)' = X' \bullet Y' \bullet Z' \ldots$	$(X \bullet Y \bullet Z \bullet \cdots)' = X' + Y' + Z' + \ldots$
Generalised De Morgan	$[f(X_1, X_2, \ldots XN, +, \bullet)]' = f(X_1', X_2', \ldots XN', \bullet, +)$	

Combinational Logic

Boolean Algebra: **De Morgan's theorem**

- Probably the most commonly used Boolean algebra result

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



- "Pull bubble through and swap AND and OR."
- Very useful for visual manipulation of circuits

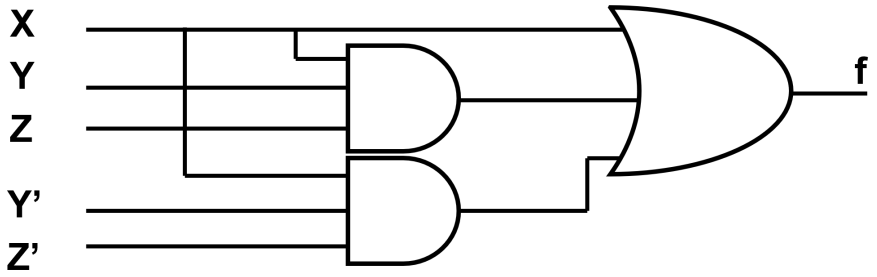
Combinational Logic

Canonical Expression: **Sum of Products**

- A **Sum of Products** (SoP) is a useful way to write a logical expression.
- A SoP contains:
  - Only OR (sum) operations at the "outermost" level
  - Each term in the sum is a product of literals, e.g.:

$$f(X, Y, Z) = X + X \bullet Y \bullet Z + X \bullet Y' \bullet Z'$$

- A SoP expression can be readily implemented as a two-level circuit, where a series of AND gates feed into a common OR gate.



Combinational Logic

Minterms

- Minterms are a very useful way to derive a logic expression expressed as a SoP from a truth table.
- A minterm is a product of literals in which
  - *Each input variable (literal) appears once*
  - *Each minterm is true (product=1) for exactly one combination of inputs*

Example:

x	y	z	minterm	F(x,y,z)
0	0	0	$x' \cdot y' \cdot z'$	1
1	0	0	$x \cdot y' \cdot z'$	0
0	1	0	$x' \cdot y \cdot z'$	0
1	1	0	$x \cdot y \cdot z'$	1
0	0	1	$x' \cdot y' \cdot z$	0
1	0	1	$x \cdot y' \cdot z$	0
0	1	1	$x' \cdot y \cdot z$	1
1	1	1	$x \cdot y \cdot z$	0

A sum of minterms describing the function F can be derived from the truth table (using the minterms for which F=1)

$$\boldsymbol{F = x' \cdot y' \cdot z' + x \cdot y \cdot z' + x' \cdot y \cdot z}$$

Canonical SOP representation: $\boldsymbol{F = \Sigma_m(0, 3, 6)}$

Combinational Logic

Maxterms and Product of Sums

- A completely equivalent, dual approach to SOP;
- Product-of-sums (POS) expression is a logical product of zz terms Product of Sums is a **product of maxterms**

$$\text{Example: } \boldsymbol{F = Z' \cdot (W + X + Y) \cdot (X + Y' + Z) \cdot (W' + Y' + Z).}$$

- It does not add anything new to this revision overview of combinational logic. You can read more about it in the Textbook: Wakerly, Section 3.1, if you are interested

Combinational Logic

What use is all this to us?

- Analysis
  - Interpreting the behavior of a combinational circuit in terms of a logic function

- Synthesis
  - Given a problem, come up with a logic function which describes it
  - Manipulate the function to use particular logic gates and/or minimise the number of components used.

Combinational Logic

Manipulation/Minimization of logic functions

Many practical reasons

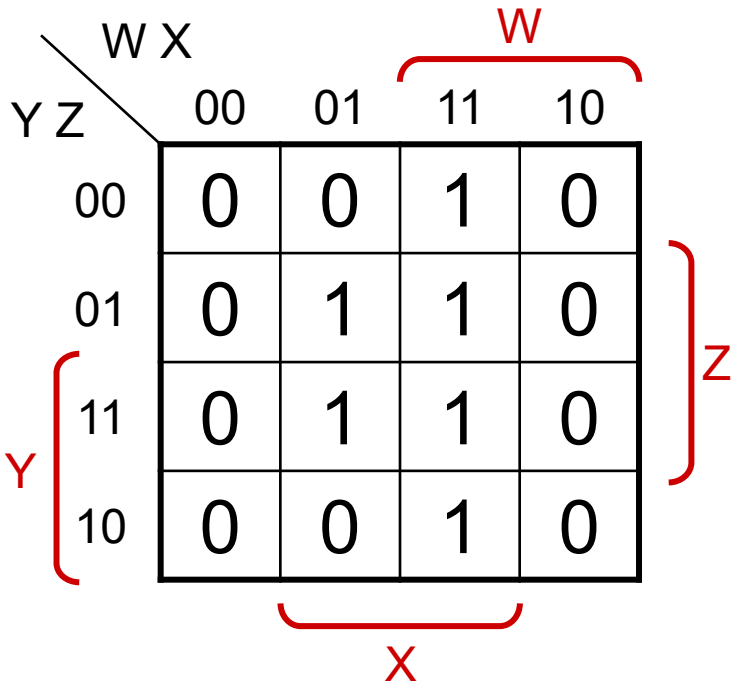
- Easier to understand/draw/visualize
- Regarding physical implementation:
  - Reduce the number of inputs / components
  - Reduce circuit footprint (space occupied)
  - Reduce cost
  - Reduce power consumption
  - Reduce propagation delays (sometimes)
  - Manipulating the expression may help the designer select logic gates which are simpler (e.g. NAND has fewer transistors than AND... seems strange but it is true!)

For simple functions it can be done by hand, for complex ones there are minimizer algorithms

Combinational Logic

Karnaugh maps

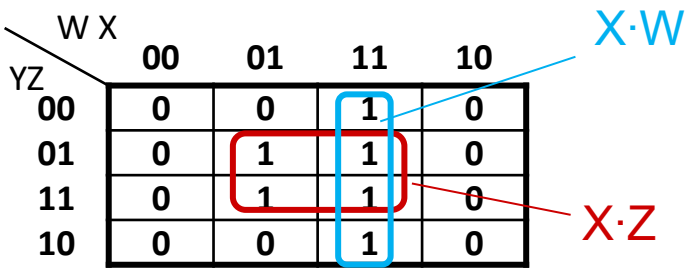
- A graphical method to perform minimization of a logic function
- They work reasonably well for functions of up to 4 variables, beyond that it gets a bit tricky
- A Karnaugh map is a table representation of a function's truth table.
  - The input variable values are written next to the table in Grey code and the function value is written inside the cells.
  - Grey code means that *each row/column differs from an adjacent one by exactly one logic shift*.



Combinational Logic

Karnaugh maps

- To minimize a logic function, identify all **prime implicants:** *set of adjacent 1-valued cells* such that if we try to double its size, some 0s will be included.
- Prime implicants represent a product term of two (or more) of the input variables. A minimal form of the logic function is given by the *sum of the least number of prime implicants such that all the 1-valued cells are considered*
- **Example:**



$$F(W, X, Y, Z) = X \cdot W + X \cdot Z$$

- There are more complex details (Wakerly 3.3). Karnaugh maps are not strictly a course topic, so this basic revision is enough for our purposes.

Combinational Logic

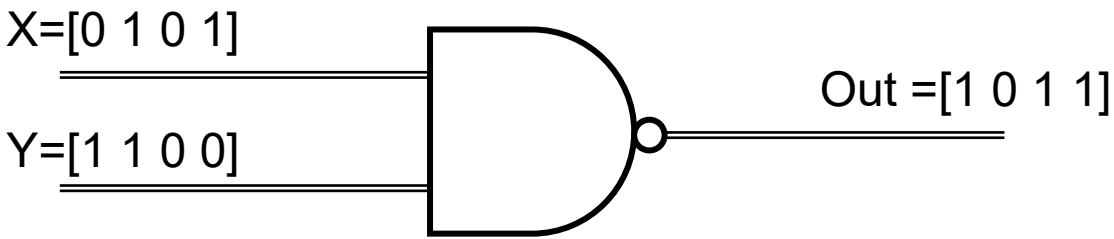
Karnaugh maps: Rule Summary

- Row and column assignments arranged such that adjacent terms change by only one bit (grey code): use 00,01,11,10 instead of 00,01,10,11
- Each map consists of 2^n cells, where n is the number of logic variables
- 2^n 1's can be circled (group) at a time: 1, 2, 4, 8, …
- No zeros can be circled.
- No diagonals.
- Groups should be as large as possible.
- Every 1 must be in at least one group.
- Overlapping and "wraps around itself" - i.e. the top and bottom, right and left edges are touching.
- Fewest number of groups possible.

Combinational Logic

Multibit Operations

- So far we have considered input signals to be a single 0/1 value, i.e., one bit.
- We will encounter designs with circuits involving buses, i.e. multiple bits fed in parallel.
- The general convention for combinational operations among multibit signals is that *operations are conducted on a bit-by-bit basis*.

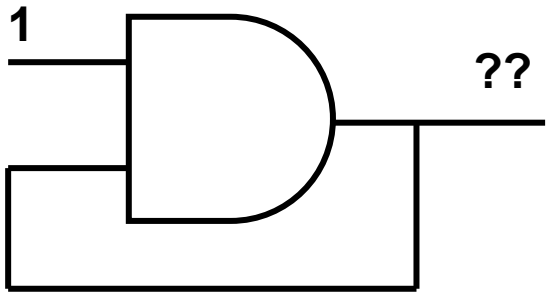


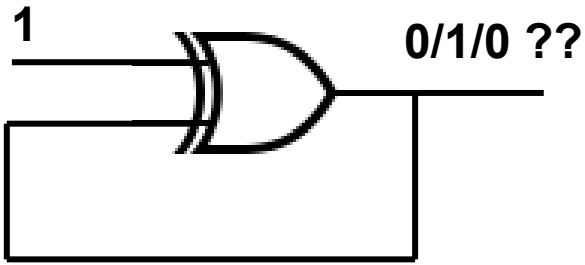
- The diagram above is more a shorthand drawing than a realistic description of a circuit. We can't achieve the result with a single NAND gate, the circuit will need to perform three separate combinational operations, either in parallel or by multiplexing.

Combinational Logic

Feedback in combinational circuits

- Generally undesirable, you can get yourself into *indeterminate or "oscillatory" situations* with very unpredictable behavior.
  - **<u>Avoid</u> using feedback in combinational designs**



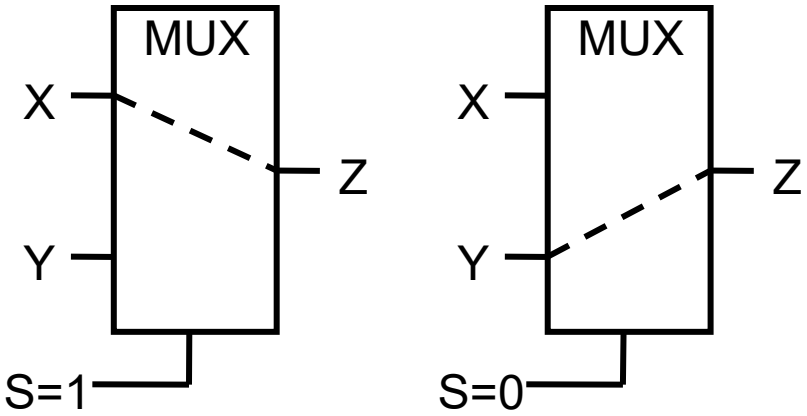


- We will see a meaningful use of feedback in the context of sequential logic later.

Combinational Logic

Common Designs: **Multiplexer**

- A multiplexer (MUX) has *multiple inputs and passes a specific one to the output* depending on a selector signal.
  - Below is a 1-bit multiplexer (2 inputs, 1 selector bit, 1 output)



X	Y	S	Z
0	xx	1	0
1	xx	1	1
xx	0	0	0
xx	1	0	1

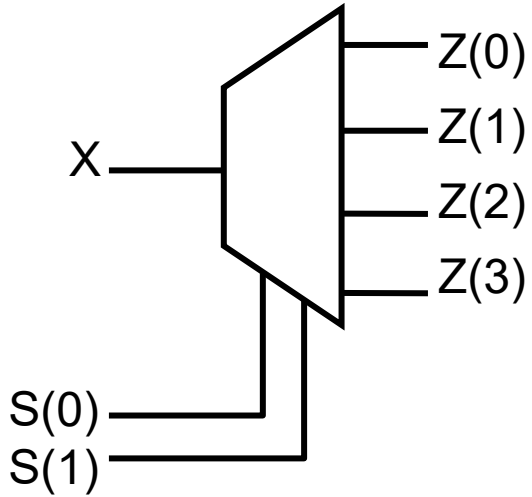
xx = "don't care"

Combinational Logic

Common Designs: **Demultiplexer**

– A demultiplexer (DMUX) is exactly the opposite of a MUX: it *routes the input to one of many possible outputs*

- Below is a 1-to-4 demultiplexer (1 input, 2 selector bits, 4 outputs)

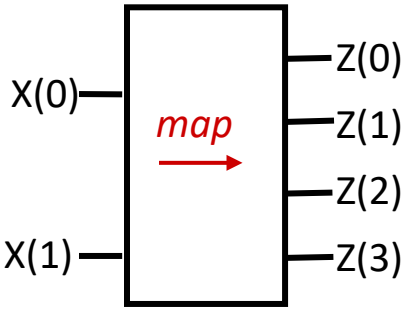


X	S(0)	S(1)	Z(0)	Z(1)	Z(2)	Z(3)
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1

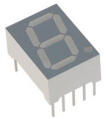
Combinational Logic

Common Designs: **Decoder**

- A decoder is a *multiple-input, multiple-output* logic device that *maps a set of input words (series of bits) to a set of output words*
  - Below you see a 2-to-4 decoder which converts a binary number into 4 available 4-bit words (chosen arbitrarily)



X(0)	X(1)	Z(0)	Z(1)	Z(2)	Z(3)
0	0	0	0	0	1
0	1	1	0	1	1
1	0	0	1	1	0
1	1	1	1	0	1



An example of a decoder is the logic circuit which drives a **7-segment display**: a 4-bit binary number is mapped to the appropriate sequence to light the segments necessary to display the number in decimal form

Combinational Logic

Common Designs: **Adder**

– An adder is a circuit which *outputs the sum of binary numbers presented as inputs*.

  • Below you see the example of a 1-bit **simple adder:**

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Do you recognise this logic function?

– A so-called ***full adder*** has an extra output bit (**carry out**) to *signal when the sum would require an extra bit,* and an extra input bit called **carry in** *useful in hierarchical adder designs.*

Combinational Logic

Summing up:

- Combinational logic circuits can be constructed to implement boolean functions.
- Can describe a discrete mathematical function which maps n inputs to m outputs as

$$(Y_1, Y_2, \dots, Y_n) = F(X_1, X_2, \dots X_m)$$

- The output is an instantaneous function of the input apart from a *short propagation delay*
- Logic functions can be manipulated using *boolean algebra*. The function does not unambiguously determine the circuit!
- **Minimization** is an important design step.
- **Feedback is not allowed** in combinational circuits for digital applications.
- A number of common (and useful!) circuits are based on combinational logic.

Number Systems (Self read)

(Wakerly Chapter 2; ENGN3213 Reading Brick from 2008)

Essential knowledge

- Decimal (!!), binary and hexadecimal systems
- Conversions between systems
  - Practice with recognising visually 0 to 15 (binary 0000 to 1111, hex 0 to F)

    Ex. d14 = $1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$ = b1110 = $14 \cdot 16^0$ = hE

- Addition operations between binary numbers

  Ex.

  $$\begin{array}{r} {\scriptstyle 1}\quad {\scriptstyle 1} \\ 10100111 + \\ 00110100 = \\ \hline 11011011 \end{array}$$

- Describing long binary strings using hexadecimal numbers

  Ex. b000111110000 = h1F0

THANK YOU

Acknowledgement

These presentation slides are the modified version of slides prepared by Dr. Jihui (Aimee) Zhang based on the original version by Dr. Nicolo Malagutti

Contact:

Amy Bastine
School of Engineering
ANU College of Engineering, Computing and Cybernetics
The Australian National University

Office: B147, Brian Anderson Building (Bldg. 115)
Email: amy.bastine@anu.edu.au

Australian National University