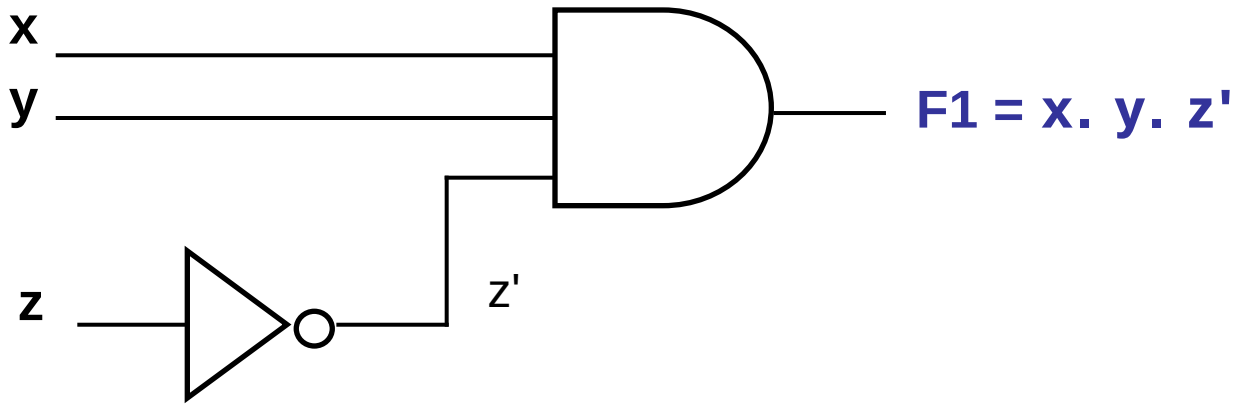
Logic Gates and Circuits

- Drawing Logic Circuit
- Analysing Logic Circuit
- Universal Gates: NAND and NOR
  - NAND Gate
  - NOR Gate
- Implementation using NAND Gates
- Implementation using NOR Gates
- Implementation of SOP Expressions
- Implementation of POS Expressions
- Positive and Negative Logic
- Integrated Circuit Logic Families

Drawing Logic Circuit (1/2)

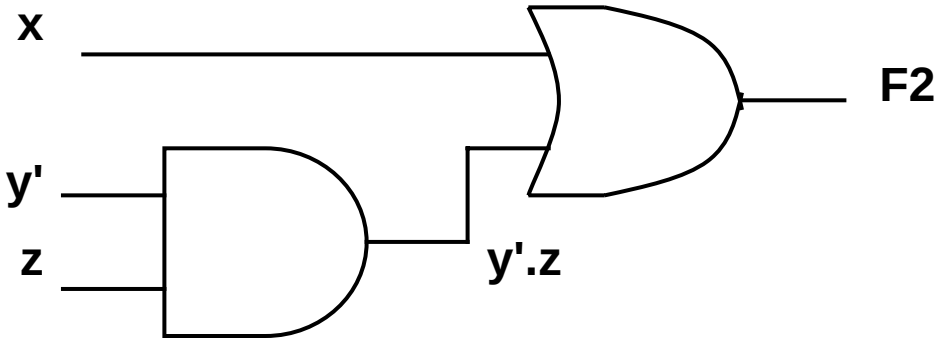
- When a Boolean expression is provided, we can easily draw the logic circuit.
- Examples:

  (i) F1 = x.y.z' (note the use of a 3-input AND gate)

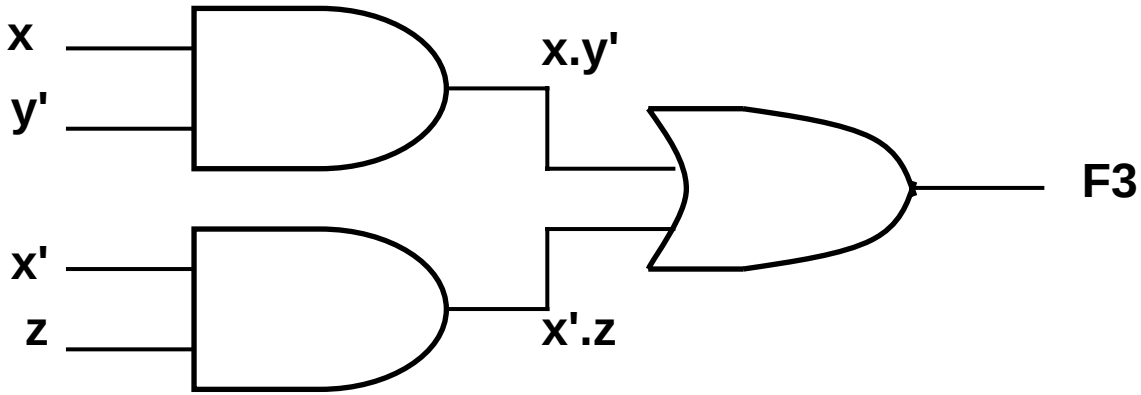


Drawing Logic Circuit (2/2)

(ii) F2 = x + y'.z (if we assume that variables and their complements are available)

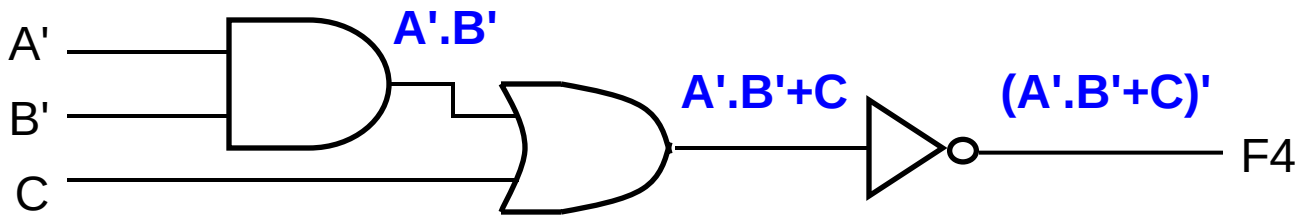


(iii) F3 = x.y' + x'.z



Analysing Logic Circuit

- When a logic circuit is provided, we can analyse the circuit to obtain the logic expression.
- Example: What is the Boolean expression of F4?



F4 = (A'.B'+C)' = (A+B).C'

Universal Gates: NAND and NOR

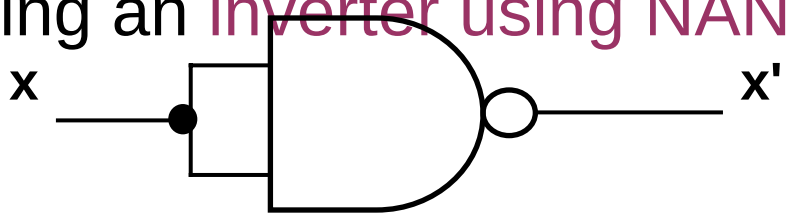
- **AND/OR/NOT** gates are sufficient for building any Boolean functions.
- We call the set {AND, OR, NOT} a complete set of logic.
- However, other gates are also used because:
  - (i) usefulness
  - (ii) economical on transistors
  - (iii) self-sufficient

NAND/NOR: economical, self-sufficient

XOR: useful (e.g. parity bit generation)

NAND Gate (1/2)

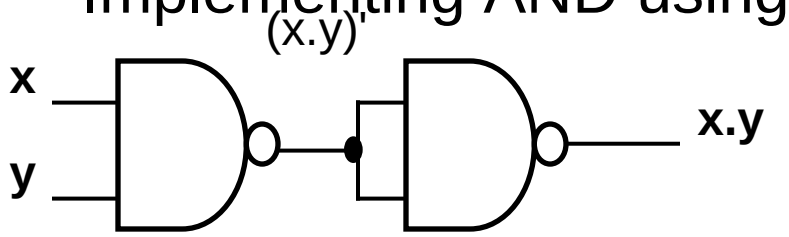
- NAND gate is self-sufficient (can build any logic circuit with it).
- Therefore, {NAND} is also a complete set of logic.
- Can be used to implement AND/OR/NOT.
- IC 7400 (4 NAND Gates)
- Implementing an inverter using NAND gate:



(x.x)' = x' (T1: idempotency)

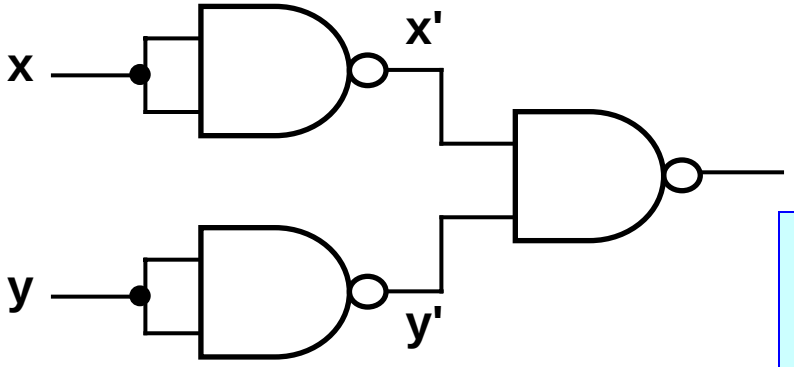
NAND Gate (2/2)

- Implementing AND using NAND gates:



((x.y)‘ (x.y)')' = ((x.y)')' idempotency
= (x y) involution

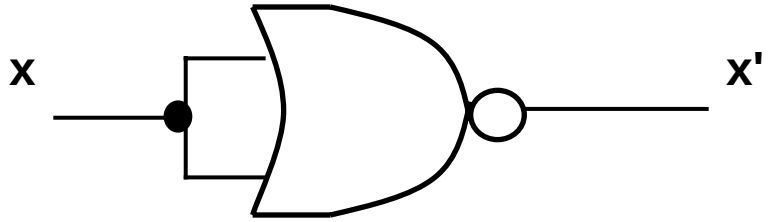
- Implementing OR using NAND gates:



((x.x)‘ (y.y)')' = (x'.y')' idempotency
= x''+y'' DeMorgan
= x+y involution

NOR Gate (1/2)

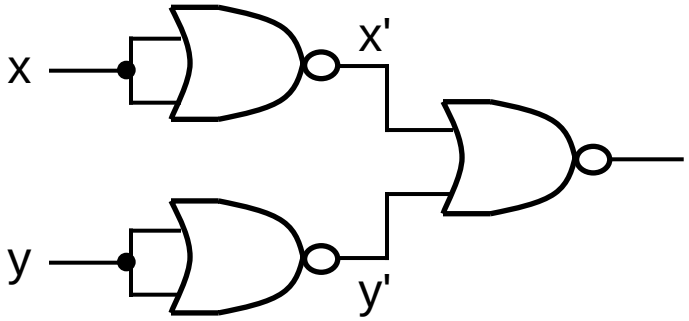
- NOR gate is also self-sufficient.
- Therefore, {NOR} is also a complete set of logic
- Can be used to implement AND/OR/NOT.
- Implementing an inverter using NOR gate:



(x+x)' = x' (T1: idempotency)

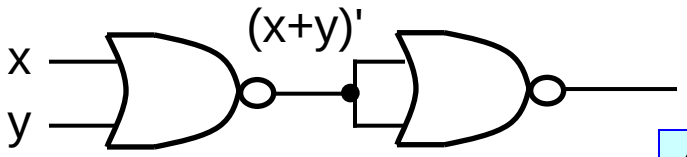
NOR Gate (2/2)

- Implementing AND using NOR gates:



((x+x)'+(y+y)')'=(x'+y')' idempotency
= x''.y'' DeMorgan
= x.y involution

- Implementing OR using NOR gates:



((x+y)'+(x+y)')' = ((x+y)')' idempotency
= (x+y) involution

Implementation using NAND gates (1/2)

- Possible to implement any Boolean expression using NAND gates.

  Procedure:

  (i) Obtain sum-of-products Boolean expression:

  e.g. F3 = x.y'+x'.z

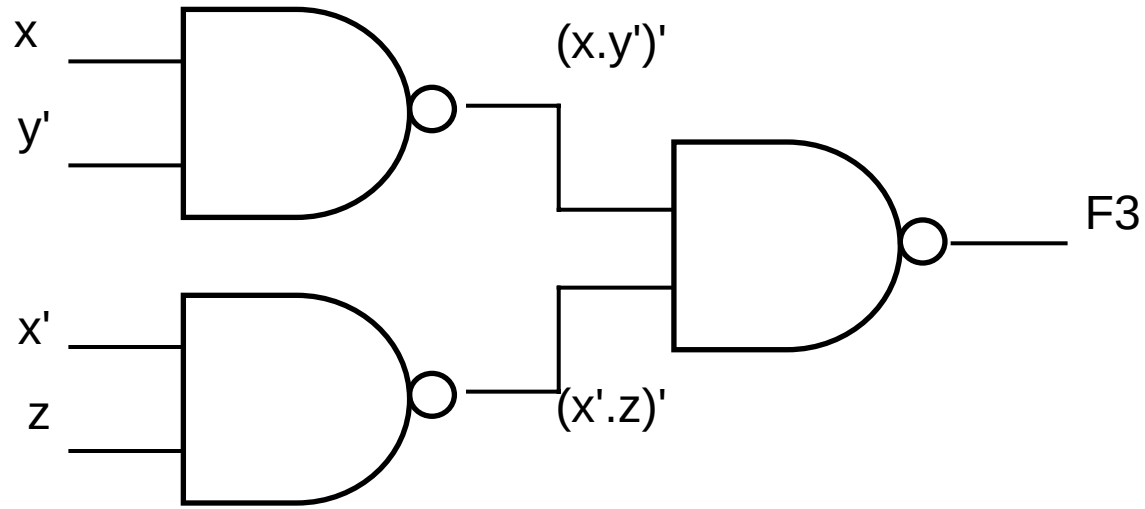
  (ii) Use DeMorgan theorem to obtain expression using 2-level NAND gates

  e.g. F3 = x.y'+x'.z

  = (x.y'+x'.z)' ' involution

  = ((x.y')' . (x'.z)')' DeMorgan

Implementation using NAND gates (2/2)



F3 = ((x . y')'. (x'.z)') ' = x . y' + x'.z

Implementation using NOR gates (1/2)

- Possible to implement any Boolean expression using NOR gates.

  Procedure:

  (i) Obtain product-of-sums Boolean expression:

  e.g. F6 = (x+y').(x'+z)

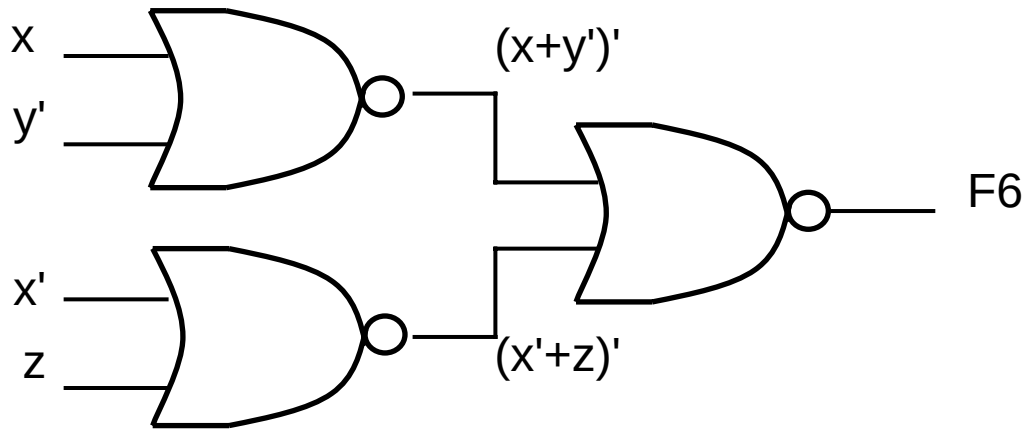
  (ii) Use DeMorgan theorem to obtain expression using 2-level NOR gates.

  e.g. F6 = (x+y').(x'+z)

  = ((x+y').(x'+z))' ' involution

  = ((x+y')'+(x'+z)')' DeMorgan

Implementation using NOR gates (2/2)

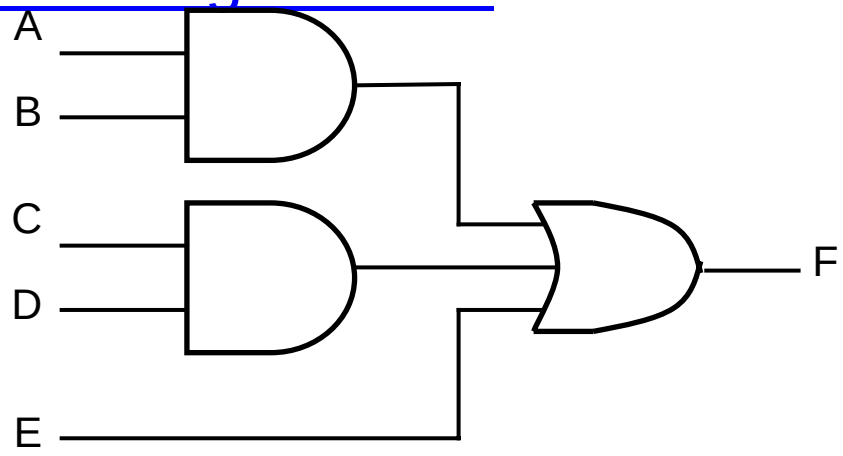


F6 = ((x+y')'+(x'+z)')' = (x+y').(x'+z)

Implementation of SOP Expressions (1/2)

- Sum-of-Products expressions can be implemented using:
  - 2-level AND-OR logic circuits
  - 2-level NAND logic circuits

- AND-OR logic circuit



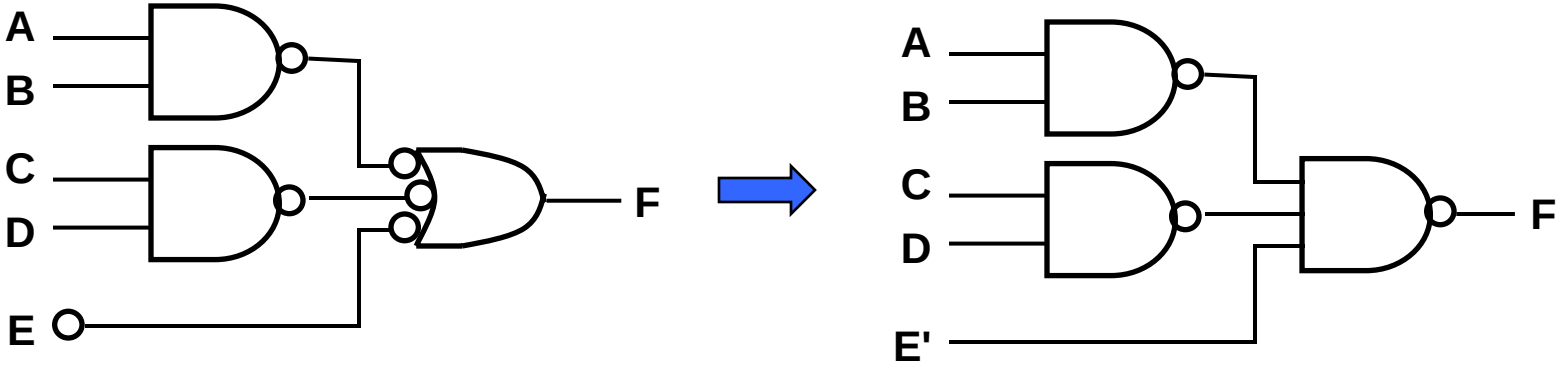
F = A.B + C.D + E

Implementation of SOP Expressions (2/2)

- NAND-NAND circuit (by circuit transformation)

  a) add double bubbles

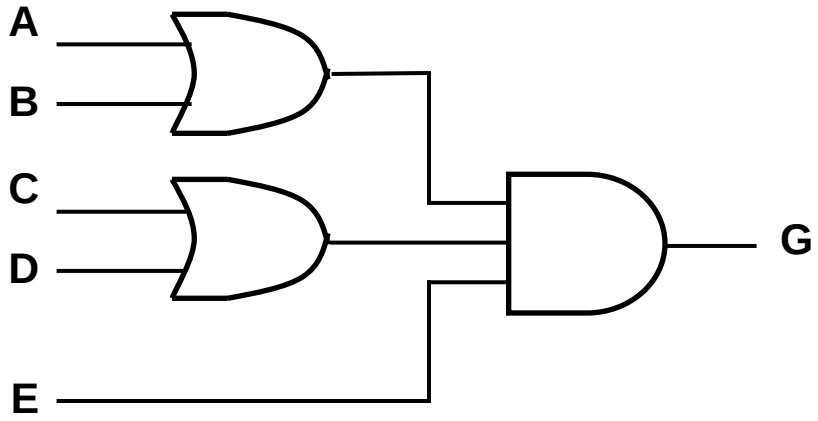
  b) change OR-with-inverted-inputs to NAND
  & bubbles at inputs to their complements



NAND ≡ Negative-OR

Implementation of POS Expressions (1/2)

- Product-of-Sums expressions can be implemented using:
  - 2-level OR-AND logic circuits
  - 2-level NOR logic circuits
- OR-AND logic circuit



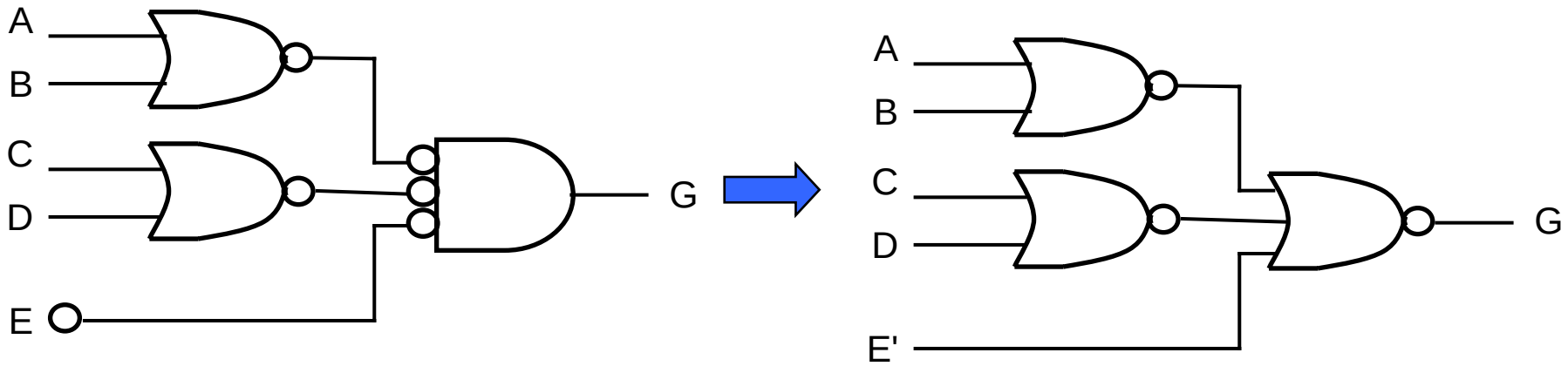
G = (A+B).(C+D).E

Implementation of POS Expressions (2/2)

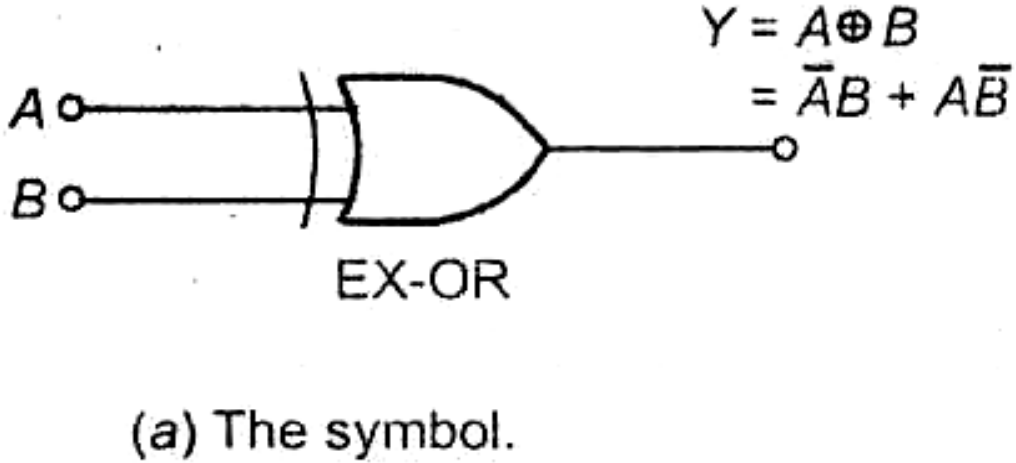
- NOR-NOR circuit (by circuit transformation):

  a) add double bubbles

  b) changed AND-with- inverted-inputs to NOR & bubbles at inputs to their complements



XOR Gate(Exclusive-OR Gate)



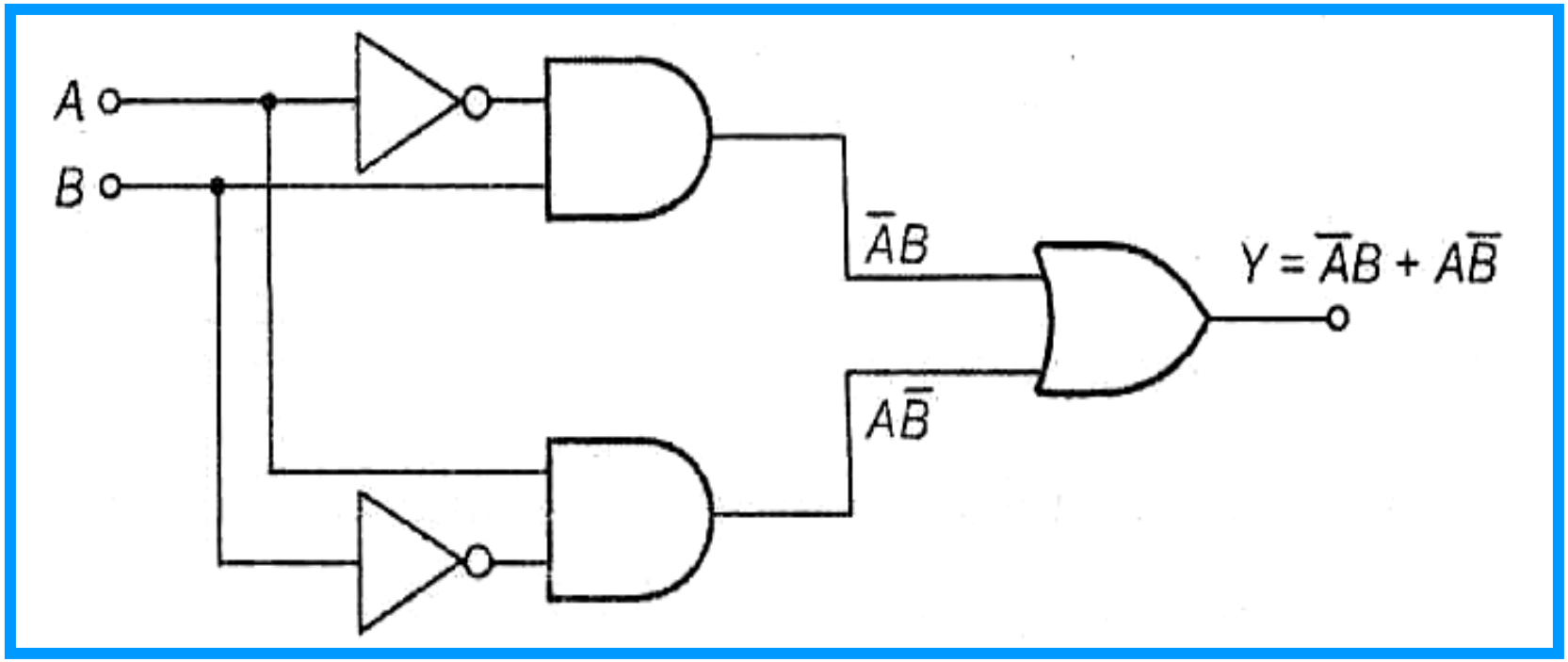
(a) The symbol.

Output is HIGH whenever two inputs are at opposite levels.

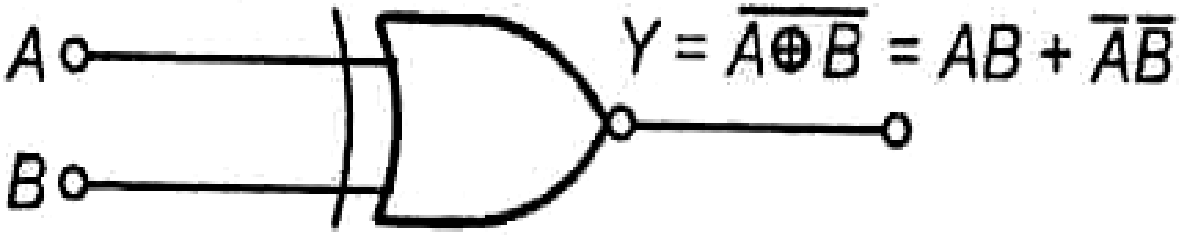
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

(b) The truth table.

Logic circuit of XOR Gate



XNOR Gate(Exclusive-NOR Gate)

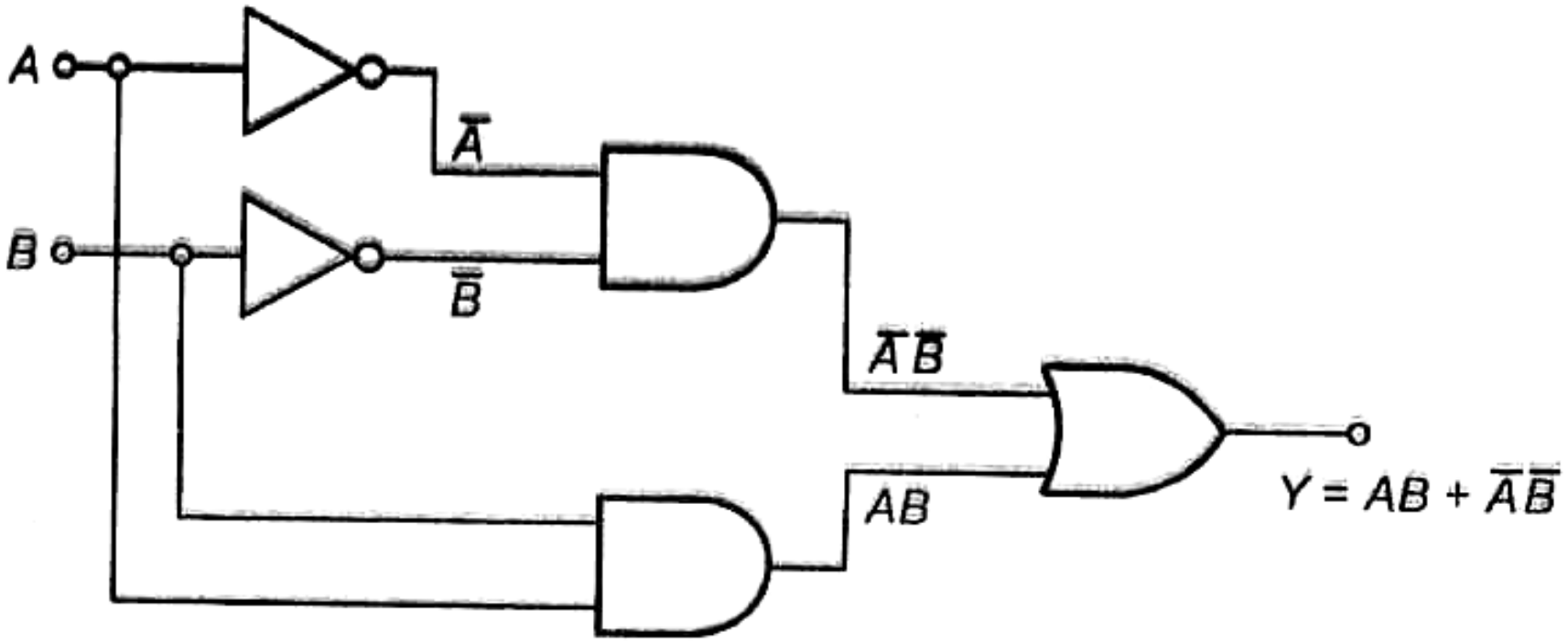


Output is HIGH whenever two inputs are at same level.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

(*b*) The truth table.

Logic circuit of XNOR Gate



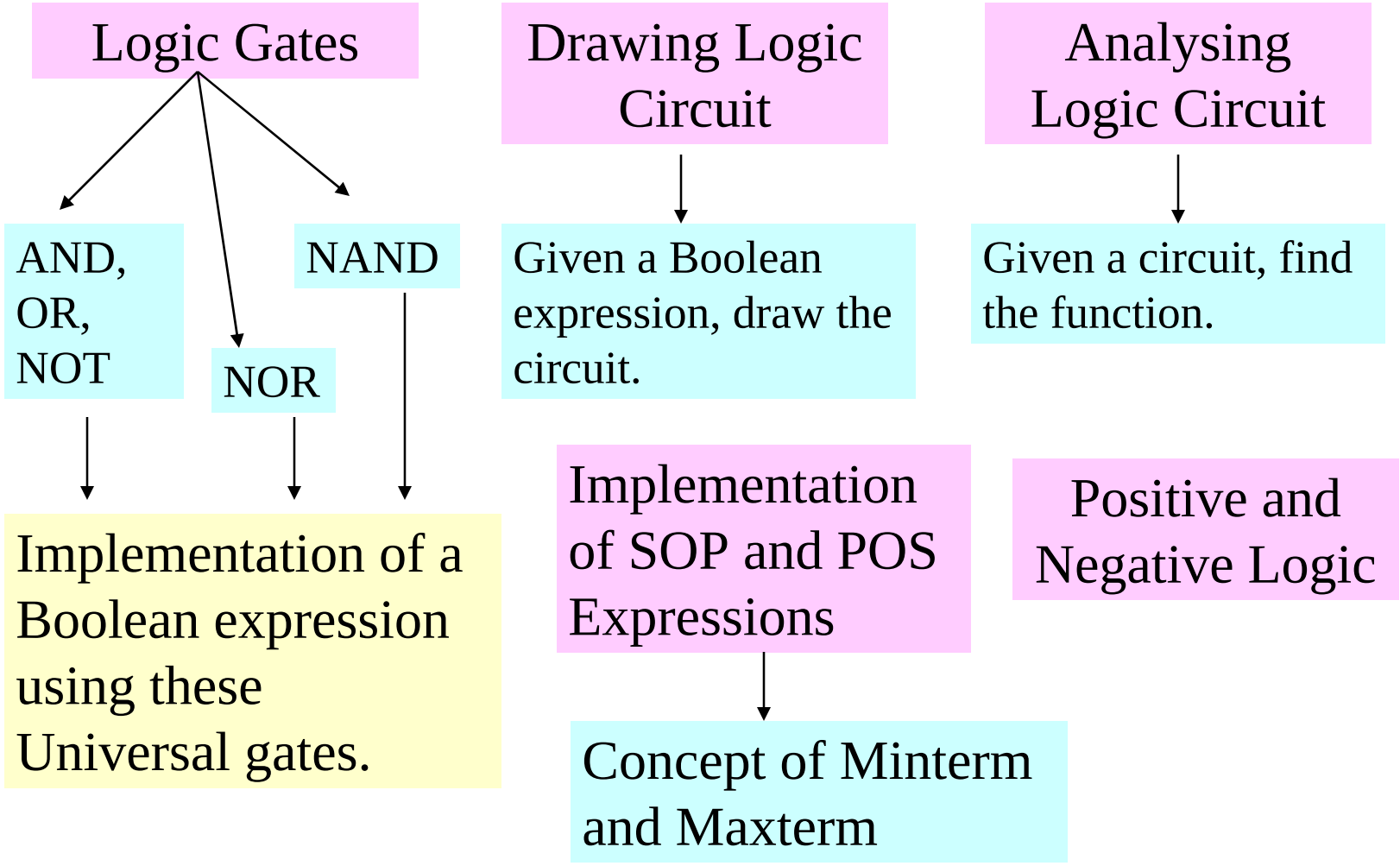
Positive & Negative Logic (1/2)

- In logic gates, usually:
  - H (high voltage, 5V) = 1
  - L (low voltage, 0V) = 0
- This convention is known as positive logic.
- However, the reverse convention, negative logic possible:
  - H (high voltage) = 0
  - L (low voltage) = 1
- Depending on convention, same gate may denote different Boolean function.

Positive & Negative Logic (2/2)

- A signal that is set to logic 1 is said to be *asserted*, or *active*, or *true*.
- A signal that is set to logic 0 is said to be *deasserted*, or *negated*, or *false*.
- Active-high signal names are usually written in uncomplemented form.
- Active-low signal names are usually written in complemented form.

Summary


Logic Gates
AND, OR, NOT
NOR
NAND
Implementation of a Boolean expression using these Universal gates.
Drawing Logic Circuit
Given a Boolean expression, draw the circuit.
Analysing Logic Circuit
Given a circuit, find the function.
Implementation of SOP and POS Expressions
Concept of Minterm and Maxterm
Positive and Negative Logic