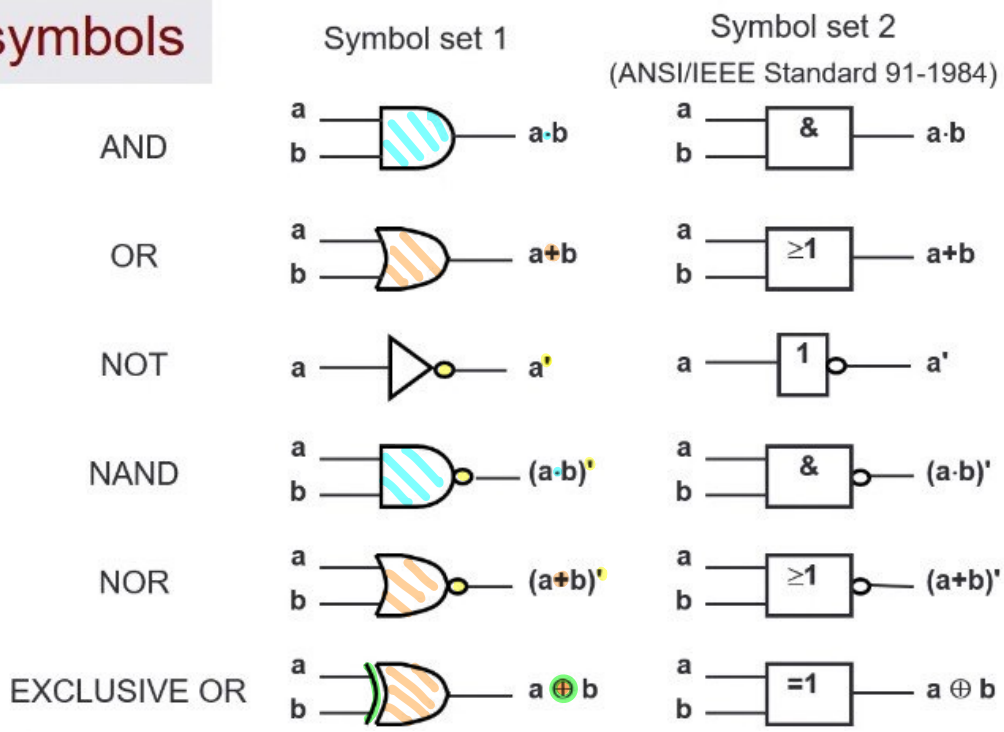
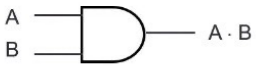
# Gate symbols




- Inverter (NOT gate)


| A | A' |
|---|---|
| 0 | 1 |
| 1 | 0 |

- AND gate



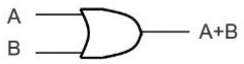

| A | B | A · B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- NAND gate


| A | B | (A · B)' |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |


NAND ~~Negative OR~~

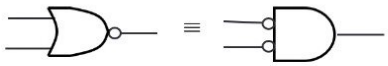
- OR gate




| A | B | A + B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

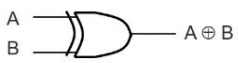
- NOR gate


| A | B | (A + B)' |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |




NOR ~~Negative AND~~

- XOR gate




| A | B | A ⊕ B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

inclusive or

A + B

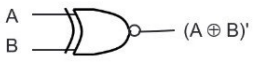
0

1

1

1

- XNOR gate




| A | B | (A ⊕ B)' |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

XNOR can be represented by ⊙
(Example: A ⊙ B)

# Logic Circuits.

- Fan-in : no. of inputs of a gate

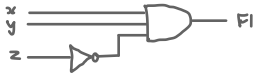
  (gate may have fan-in more than 2)

eg.

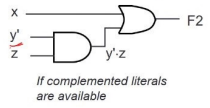

every input must be connected in a working circuit!

- May implement boolean expressions as logic circuits.

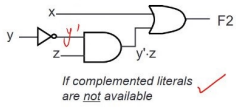
eg. F1 = x·y·z' (3 input AND gate)




- Example: F2 = x + y'·z

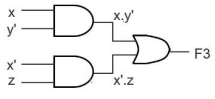


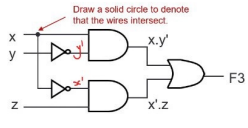

If complemented literals are available




If complemented literals are not available

- Example: F3 = x·y' + x'·z



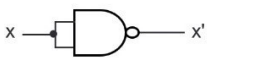

## Universal Gates

- AND/OR/NOT gates are sufficient for building any boolean function
- {AND, OR, NOT} ⇒ Complete set of logic
- Other gates are also used – usefulness, economical, self-sufficient
  - usefulness: eg. XOR for parity bit gen.
  - self-sufficient: NAND/NOR

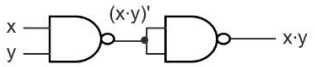
### - NAND gate

- It is a complete set of logic

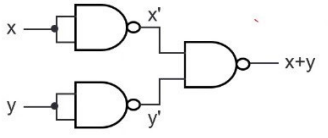
  ⇒ able to build NOT/AND/OR w just NAND.




(x·x)' = x' (idempotency) — NOT

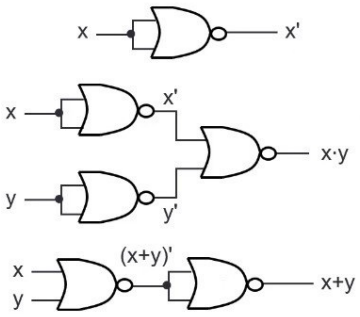



((x·y)'·(x·y)')' = ((x·y)')' (idempotency)
= x·y (involution) — AND



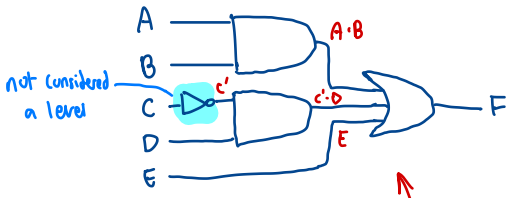

((x·x)'·(y·y)')' = (x'·y')' (idempotency)
= (x')'+(y')' (DeMorgan)
= x+y (involution) — OR

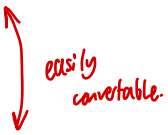
- NOR gates
  - Also a complete set of logic.



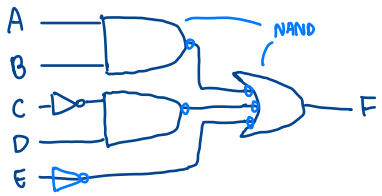

| | |
|---|---|
| (x+x)' = x' (idempotency) | NOT |
| ((x+x)'+(y+y)')' = (x'+y')' (idempotency)<br>= (x')'·(y')' (DeMorgan)<br>= x·y (involution) | AND |
| ((x+y)'+(x+y)')' = ((x+y)')' (idempotency)<br>= x+y (involution) | OR. |

- SOP expression can be easily implemented using
  eg. F = A·B + C'·D + E
  - 2-level AND-OR circuit

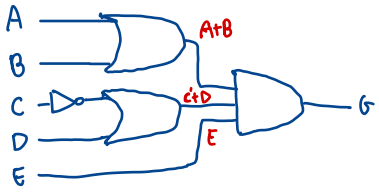


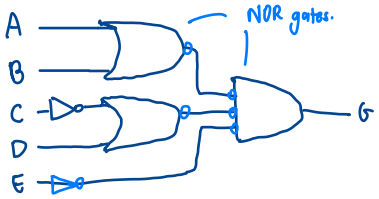

  - 2-level NAND circuit.




- POS w NOR
  - eg. G = (A+B)·(C'+D)·E
  - 2 level OR-AND circuit

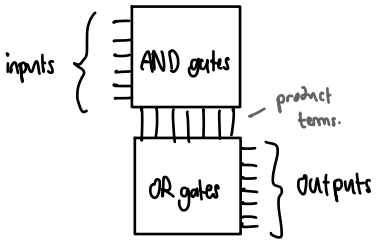



  - 2 level NOR circuit



# Programming Logic Array (PLA)

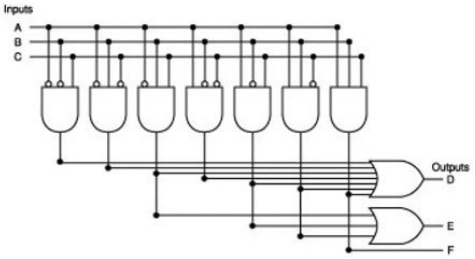
- A programmable integrated circuit — implements sum-of-products circuits — allow multiple outputs.
- 2 stages
  - AND gates = product terms
  - OR gates = outputs

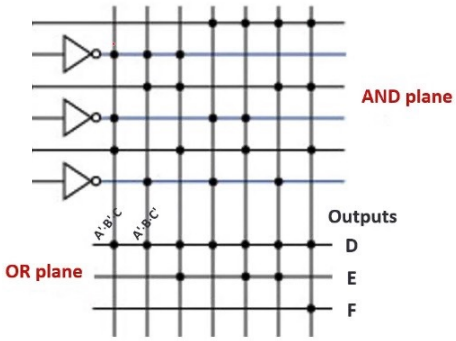



- Connection btw inputs and the planes can be "burned."

eg.

| Inputs | | | Outputs | | |
|---|---|---|---|---|---|
| A | B | C | D | E | F |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |




- Read Only Memory (ROM)
  - Similar to PLA
    - Set of inputs (addr.), set of outputs
    - programmable mapping btw inputs & outputs.
  - Fully decoded — able to implement any mapping.
  - PLAs may not be able to implement a given mapping due to not having enough minterms.