Boolean Algebra

- **Boolean values/expressions** -> **all values are either true or false**
- **Problems:** *Electronic System Design, Marketing, Control Program Flow, (PASTA)*
- **What is needed** -> *Simplify the expression:*
  - ***Minimize resource utilization***
  - ***Extract Dependency***
- **Solutions** -> ***Minterms, Maxterms (BOOK)*** *gives you a standard, unambiguous form tied directly to the truth table*

What is Next?????

Boolean Algebra Laws

Law	Version +	Version ·
Commutative	$A+B=B+A$	$A \cdot B=B \cdot A$
Distributive	$A \cdot (B+C)=(A \cdot B)+(A \cdot C)$	$A+(B \cdot C)=(A+B) \cdot (A+C)$
Idempotent	$A+A=A$	$A \cdot A=A$
Associative	$(A+B)+C=A+(B+C)$	$(A \cdot B) \cdot C=A \cdot (B \cdot C)$
Involution	$(A')'=A$	
Absorption	$A+(A \cdot B)=A$	$A \cdot (A+B)=A$
De Morgan	$(A+B)'=A' \cdot B'$	$(A \cdot B)'=A'+B'$
Identity	$A+0=A$	$A \cdot 1=A$
Complement	$A+A'=1$	$A \cdot A'=0$
Null/Annihilator	$A+1=1$	$A \cdot 0=0$

Implementation

- **How do we implement boolean operations (BO) in electrical circuits?**

- **Example: NANDs**

- **Do Minterms help with simplifying BO with NANDs? -> Yes**
  minimized **Sum of Products (SOP)** drops straight into a two-level NAND–NAND implementation:
  - **Level 1:** one NAND per product term (it naturally gives you the inverted product).
  - **Level 2:** a single NAND that “ORs” those (via De Morgan):
- **How?** -> **De Morgan’s Law:**
  - not (A or B) = (not A) and (not B)
  - not (A and B) = (not A) or (not B)

NAND Representation

- Example: convert to all NANDs $ab+c$

  $\overline{\overline{ab+c}}$ (double negation)

  $\overline{\overline{a}\overline{b}*\overline{c}}$ (De Morgan)

  NAND(ab,c)=NAND(NAND(a,b),NAND(c,c))

- BUT WHY? -> One amazing example from the Verilator tool

```
174       SLIDEUP_SETUP: begin
175         if (!alu_valid_i) next_state = ERR;  // expected VL - OFFS
176         // Skip trivial case where OFFS > VL
177         // NOTE: the following expression triggers a bug in Verilator 4.210
178         //   slide_offs_ovf_i || (slideup_vl_ovf_i & (!byte_num_max_i))
179         // so we apply De Morgan's law to rewrite it as:
180         else if (!((!slide_offs_ovf_i) && (!(slideup_vl_ovf_i && !byte_num_max_i))))
181           next_state = WAIT_COMMIT;
182         else next_state = SLIDE_READ_FIRST;
183       end
```

Time to Think

- Point 3 from “Representation” Slide
- Why NAND and not NOR?
  - Are there alternatives?
- Prove De Morgan's law (just to practice)