ECE 2300 Digital Logic and Computer Organization Fall 2025

Topic 3: Boolean Algebra

School of Electrical and Computer Engineering
Cornell University

revision: 2025-09-11-11-01

1 From Logic Gates to Boolean Equations **3**

2 From Truth Tables to Boolean Equations **6**

2.1. Sum-of-Products Canonical Form 6

2.2. Table-Gate-Equation Abstraction Triangle 9

3 Boolean Algebra **10**

3.1. Boolean Axioms, Theorems, and Proofs 10

3.2. Simplifying with Boolean Algebra 15

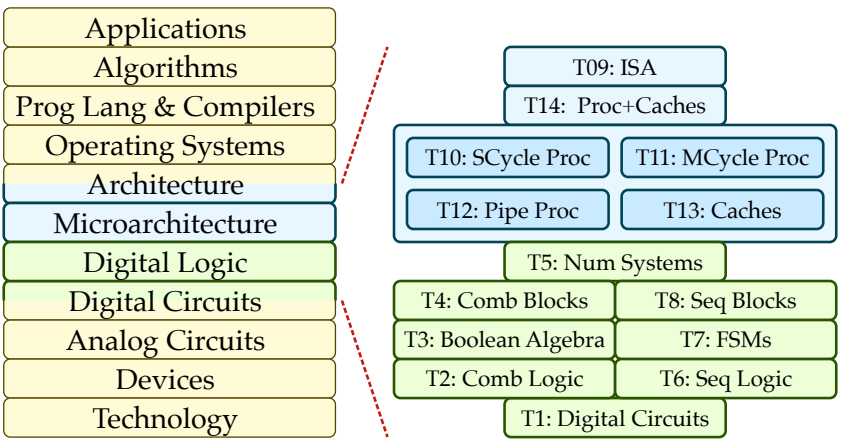
3.3. Simplifying with Karnaugh Maps 17

4 Verilog Modeling of Boolean Equations **24**

4.1. Writing Boolean Equations in Verilog 24

4.2. Implementing SOP Form in Verilog 25

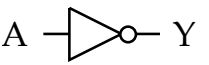
5 Summary of Abstractions **26**



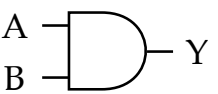
1. From Logic Gates to Boolean Equations

- We can represent our primitive logic gates as Boolean equations
  - Higher level of abstraction makes it easier to understand, manipulate, and optimize gate-level networks

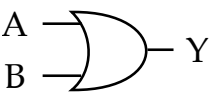
NOT: $Y = \overline{A} = A'$



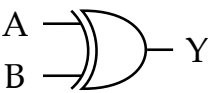
$Y = A \bullet B = AB$



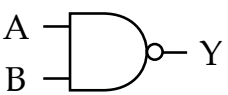
$Y = A + B$



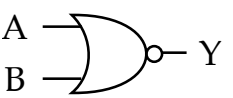
$Y = A \oplus B$



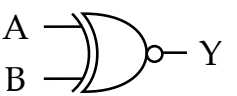
$Y = \overline{A \bullet B} = \overline{AB}$



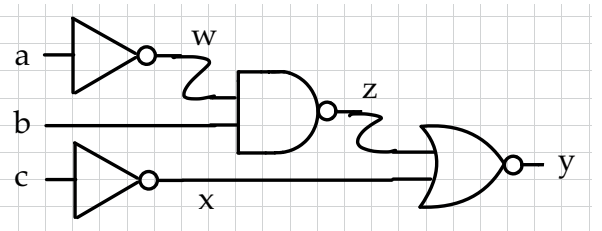
$Y = \overline{A + B}$



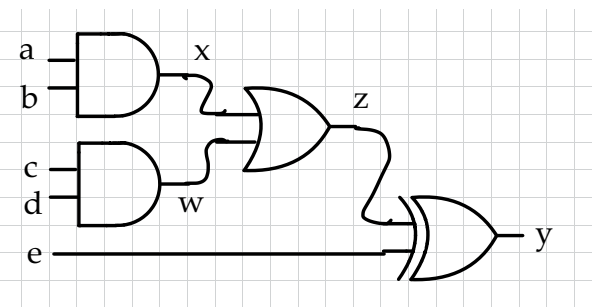
$Y = \overline{A \oplus B}$



- Let's define some terminology
  - _____ : equation where all variables are TRUE or FALSE
  - _____ : inverse of a variable (complement of A is $\overline{A}$)
  - _____ : variable or its complement (A, B, $\overline{A}$, $\overline{B}$)
  - _____ : AND of one or more literals ($\overline{A}B$, $A\overline{B}\,\overline{C}$, B)
  - _____ : another name for a product
  - _____ : product involving all inputs to function ($A\overline{B}C$)
  - _____ : OR of one or more literals ($A + \overline{B}$, B)
  - _____ : sum involving all inputs to function ($A + \overline{B} + C$)



Corresponding Boolean equation



Corresponding Boolean equation

From Boolean Equations to Logic Gates

$$Y = (\overline{A + B}) \bullet \overline{C}$$

$$Y = \overline{B}\,\overline{C} + A\,\overline{B}$$

2. From Truth Tables to Boolean Equations

- Truth table for N inputs has 2^N rows, one for every possible set of input values
- We can systematically transform any truth table into a corresponding Boolean equation

2.1. Sum-of-Products Canonical Form

- Each row in the truth table is associated with a minterm
- We can write a Boolean equation for any truth table as the OR of the minterms for which the output is TRUE
- Called sum-of-products (SOP) canonical form

A	B	Y	minterm	name
0	0	0	$\overline{A}\,\overline{B}$	m_0
0	1	1	$\overline{A}\,B$	m_1
1	0	0	$A\,\overline{B}$	m_2
1	1	0	$A\,B$	m_3

Corresponding Boolean equation

A	B	Y	minterm	name
0	0	0	$\overline{A}\,\overline{B}$	m_0
0	1	1	$\overline{A}\,B$	m_1
1	0	0	$A\,\overline{B}$	m_2
1	1	1	$A\,B$	m_3

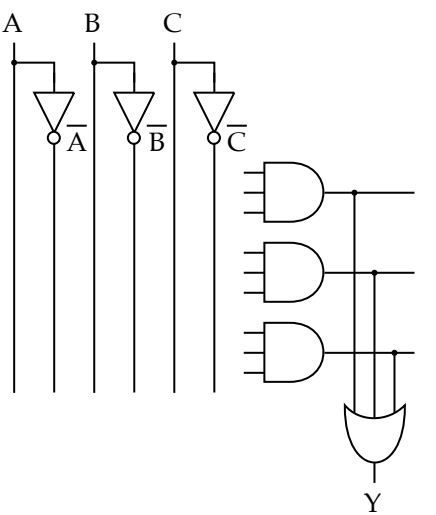
Corresponding Boolean equation

Truth Table

A	B	C	Y	minterm	name
0	0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	m_0
0	0	1	0	$\overline{A}\,\overline{B}C$	m_1
0	1	0	1	$\overline{A}B\overline{C}$	m_2
0	1	1	1	$\overline{A}BC$	m_3
1	0	0	0	$A\overline{B}\,\overline{C}$	m_4
1	0	1	0	$A\overline{B}C$	m_5
1	1	0	0	$AB\overline{C}$	m_6
1	1	1	1	ABC	m_7

Boolean Equation

Gate-Level Network

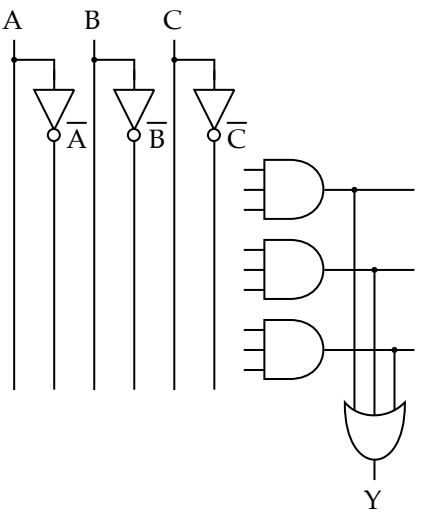


Truth Table

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

Boolean Equation

Gate-Level Network



2.2. Table-Gate-Equation Abstraction Triangle

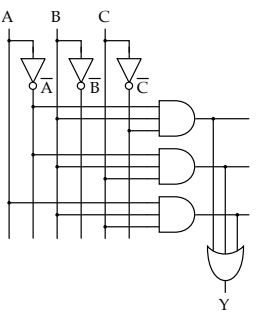
Truth Tables

A	B	C	Y	minterm	name
0	0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	m_0
0	0	1	0	$\overline{A}\,\overline{B}C$	m_1
0	1	0	1	$\overline{A}B\overline{C}$	m_2
0	1	1	1	$\overline{A}BC$	m_3
1	0	0	0	$A\overline{B}\,\overline{C}$	m_4
1	0	1	0	$A\overline{B}C$	m_5
1	1	0	0	$AB\overline{C}$	m_6
1	1	1	1	ABC	m_7

Boolean Equations

$$Y = \overline{A}B\overline{C} + \overline{A}BC + ABC$$

Gate-Level Networks



3. Boolean Algebra

- Algebra enables logically manipulating mathematical equations
- Boolean algebra enables logically manipulating Boolean equations

3.1. Boolean Axioms, Theorems, and Proofs

- The five axioms of Boolean algebra and their duals define Boolean variables and meaning of NOT, AND, OR

Number	Axiom	Dual	Name
A1	B = 0 if B ≠ 1	B = 1 if B ≠ 0	Binary Field
A2	$\overline{0} = 1$	$\overline{1} = 0$	NOT
A3	0 • 0 = 0	1 + 1 = 1	AND/OR
A4	1 • 1 = 1	0 + 0 = 0	AND/OR
A5	0 • 1 = 1 • 0 = 0	1 + 0 = 0 + 1 = 1	AND/OR

- Following theorems use the five axioms to describe equivalent equations involving one Boolean variable

Number	Theorem	Dual	Name
T1	B • 1 = B	B + 0 = B	Identity
T2	B • 0 = 0	B + 1 = 1	Null Element
T3	B • B = B	B + B = B	Idempotency
T4	$\overline{\overline{B}} = B$		Involution
T5	B • $\overline{B}$ = 0	B + $\overline{B}$ = 1	Complements

- Following theorems use the previous axioms/theorems to describe equivalent equations involving more than one Boolean variable

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	(B•C) + (B•$\overline{C}$) = B	(B+C) • (B+$\overline{C}$) = B	Combining
T11	(B•C) + ($\overline{B}$•D) + (C•D) = (B•C) + ($\overline{B}$•D)	(B+C) • ($\overline{B}$+D) • (C+D) = (B+C) • ($\overline{B}$+D)	Consensus
T12	$\overline{B•C•D\ldots}$ = $\overline{B}$+$\overline{C}$+$\overline{D}$…	$\overline{B+C+D\ldots}$= $\overline{B}$•$\overline{C}$•$\overline{D}$…	De Morgan's

- Two methods to prove a theorem
  - Method 1: Perfect induction
  - Method 2: Use other theorems and axioms to make one side of the equation look like the other

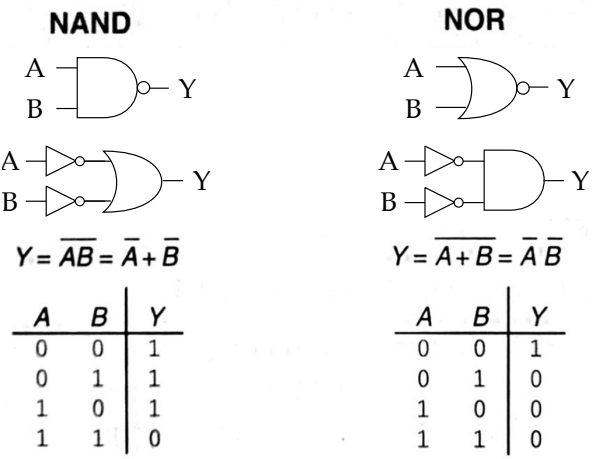
T9: $B \bullet (B + C) = B$

T9′: $B + (B \bullet C) = B$

T10: $(B \bullet C) + (B \bullet \overline{C}) = B$

De Morgan's Theorem

T12: $\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$



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

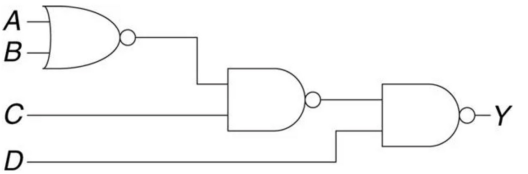
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

- Use De Morgan's Theorem to derive Y in canonical SOP form

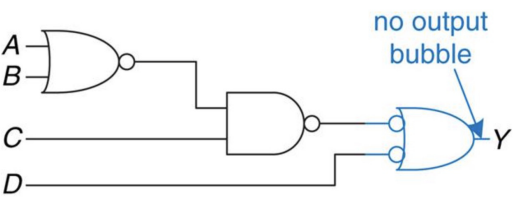
$$Y = \overline{(A+B)(A+\overline{B})(B+\overline{C})}$$

Bubble pushing

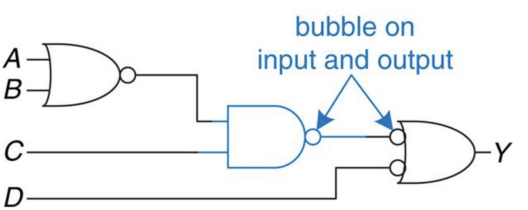
$$Y = \overline{\overline{(\overline{A+B})C}D}$$



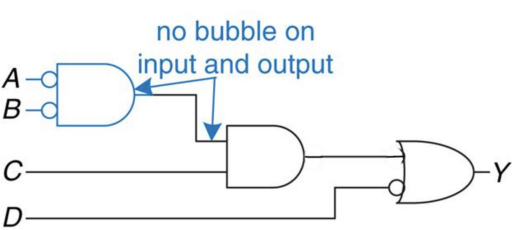
$$Y = \overline{\overline{(\overline{A+B})C}} + \overline{D}$$



$$Y = (\overline{A+B})C + \overline{D}$$



$$Y = \overline{A}\,\overline{B}C + \overline{D}$$



3.2. Simplifying with Boolean Algebra

- Consider two logically equivalent Boolean equations in SOP form
  - The *simpler* equation is the one fewer implicants
  - If both equations have the same number of implicants, then the *simpler* equation is the one with fewer literals

$$Y = \overline{A}\,B\,\overline{C} + \overline{A}BC + ABC$$
$$Y = \overline{A}B + ABC$$
$$Y = \overline{A}B + BC$$

- Simplifying Boolean equations involves using Boolean algebra to incrementally transform one equation into a simpler equation
- A Boolean equation in SOP form is *minimized* if there are no additional opportunities to simplify the equation
- Key to simplifying Boolean equations in SOP form is using the *combining* theorem

$Y = \overline{A}B + AB$	
$Y = B$	T10: Combining

$Y = \overline{A}B + AB$	
$Y = B(A + \overline{A})$	T8: Distributivity
$Y = B(1)$	T5′: Complements
$Y = B$	T1: Identity

- Often need to apply other axioms or theorems first (possibly making the equation more complex!) before applying the *combining* theorem

$$Y = \overline{A}B\overline{C} + \overline{A}BC + ABC$$

Attempt #1

Attempt #2

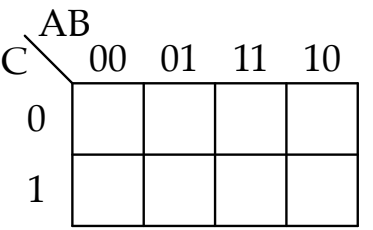
3.3. Simplifying with Karnaugh Maps

- K-maps are an abstraction of Boolean algebra for simplification

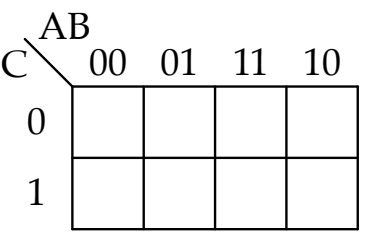
$$Y = \overline{A}B\overline{C} + \overline{A}BC + ABC$$

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

Attempt #1



$$Y = \overline{A}B\overline{C} + \overline{A}BC + ABC$$



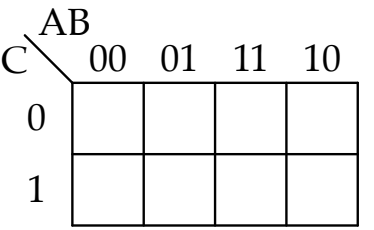
$$Y = \overline{A}B + ABC$$

$$Y = \overline{A}B\overline{C} + \overline{A}BC + ABC$$

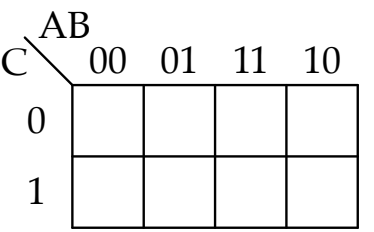
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

A *minimized* K-map is one in which all ones are "covered" with the minimal number of ovals *and* those ovals are as large as possible.

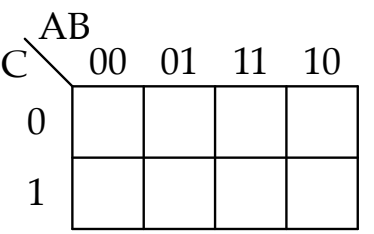
Attempt #2



$$Y = \overline{A}B\overline{C} + \overline{A}BC + ABC$$



$$Y = \overline{A}B\overline{C} + \overline{A}BC + \overline{A}BC + ABC$$

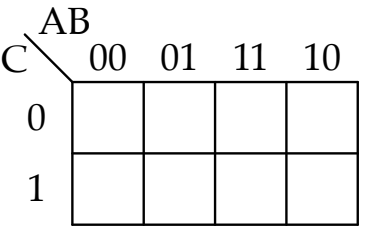


$$Y = \overline{A}B + BC$$

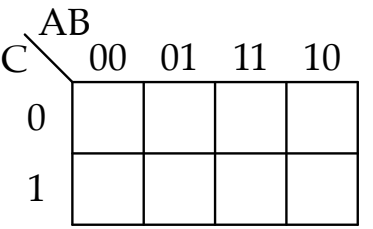
Use incremental K-maps to illustrate how to minimize the given Boolean equation. You must show each step and the associated Boolean logic equation for that step. Label each oval with the corresponding implicant in each step.

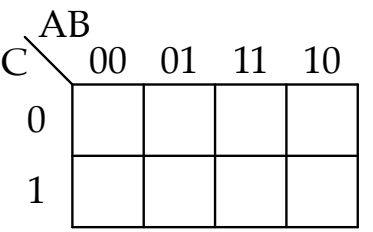
$$Y = \overline{A}B\overline{C} + \overline{A}BC + ABC$$

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

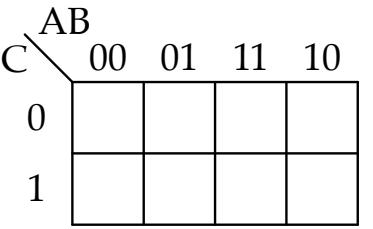


$$Y = \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$$

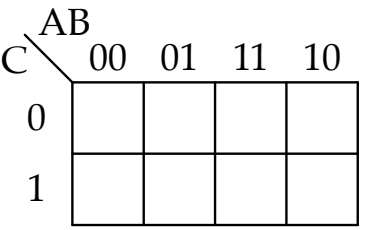




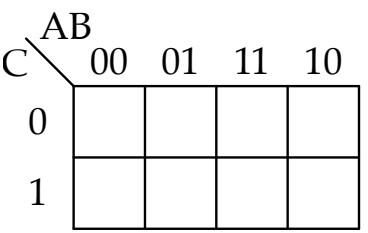
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



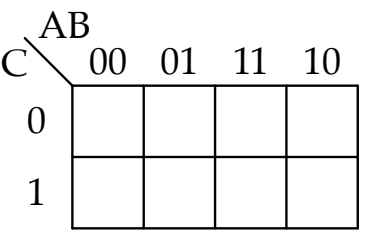
$$Y = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$$



$$Y = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + AB\overline{C} + ABC$$



$$Y = B + A\overline{C}$$

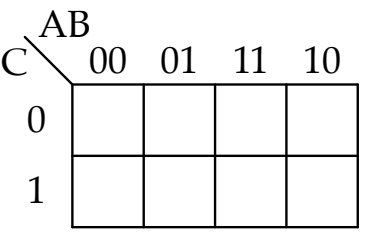


$$Y = \overline{A}B\overline{C} + \overline{A}BC + A\overline{C} + AB\overline{C} + ABC$$

- Can also go directly to the final minimized K-map

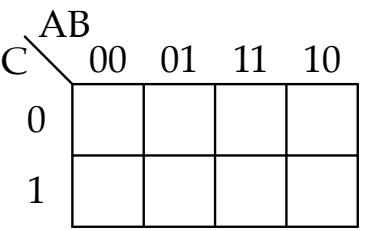
$$\begin{aligned} Y &= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} \\ &+ A\overline{B}C + AB\overline{C} \end{aligned}$$

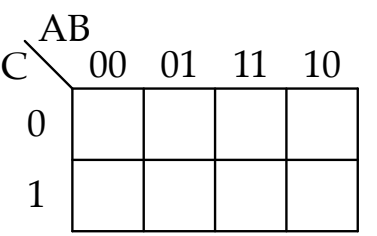
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



- An output in a truth table can be a *don't care* (X) which means the logic designer can choose whether to make that output a 0 or 1
- Don't cares enable even more logic minimization
- Can choose to either cover an X with a circle (i.e., output will be one) or to not cover an X with a circle (i.e., output will be zero)

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	X
1	0	1	1
1	1	0	0
1	1	1	0

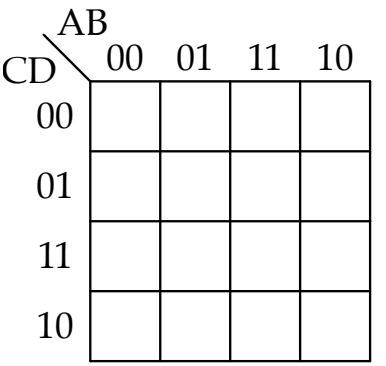




Use a K-map to minimize the following equation with four Boolean variables. Show the final minimized K-map and Boolean equation. Label each oval with the corresponding implicant in the final minimized Boolean equation.

$$\begin{aligned} Y &= \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}D \\ &+ \overline{A}\,BC\overline{D} + \overline{A}\,BCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D + A\overline{B}C\overline{D} \end{aligned}$$

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



4. Verilog Modeling of Boolean Equations

- We can raise the level of abstraction in our Verilog models by using Boolean equations to represent gate-level networks

4.1. Writing Boolean Equations in Verilog

NOT	$Y = \overline{A} = A'$	`assign y =  ~a;`
AND	$Y = A \bullet B = AB$	`assign y =  a & b;`
NAND	$Y = \overline{A \bullet B} = \overline{AB}$	`assign y = ~(a & b);`
OR	$Y = A + B$	`assign y =  a | b;`
NOR	$Y = \overline{A + B}$	`assign y = ~(a | b);`
XOR	$Y = A \oplus B$	`assign y =  a ^ b;`
XNOR	$Y = \overline{A \oplus B}$	`assign y = ~(a ^ b);`

$$
\begin{aligned}
W &= \overline{A} \\
Z &= \overline{W}B \\
X &= \overline{C} \\
Y &= \overline{Z + X} \\
\\
Y &= \overline{\overline{(\overline{A}B)} + \overline{C}}
\end{aligned}
$$

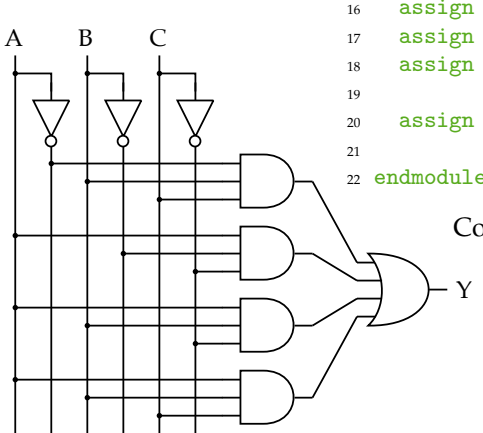
```
1  module BoolAlgebraEx1
2  (
3    input  wire a,
4    input  wire b,
5    input  wire c,
6    output wire y
7  );
8    wire w, x, z;
9
10   assign w = ~a;
11   assign z = ~( w & b );
12   assign x = ~c;
13   assign y = ~( z | x );
14
15   // assign y = ~( ~( ~a & b ) | ~c )
16
17 endmodule
```

4.2. Implementing SOP Form in Verilog

- Let's revisit a truth table from the previous topic
- Create a wire for every minterm
- Assign output as the OR of every minterm for which output is TRUE

A	B	C	Y	
0	0	0	0	m_0
0	0	1	0	m_1
0	1	0	0	m_2
0	1	1	1	m_3
1	0	0	1	m_4
1	0	1	0	m_5
1	1	0	1	m_6
1	1	1	1	m_7

```
1  module BoolAlgebraEx2
2  (
3    input  wire a,
4    input  wire b,
5    input  wire c,
6    output wire y
7  );
8    wire [7:0] m;
9
10   assign m[0] = ~a & ~b & ~c;
11   assign m[1] = ~a & ~b &  c;
12   assign m[2] = ~a &  b & ~c;
13   assign m[3] = ~a &  b &  c;
14
15   assign m[4] =  a & ~b & ~c;
16   assign m[5] =  a & ~b &  c;
17   assign m[6] =  a &  b & ~c;
18   assign m[7] =  a &  b &  c;
19
20   assign y = m[3] | m[4] | m[6] | m[7];
21
22 endmodule
```



Corresponding Boolean equation

5. Summary of Abstractions

Boolean Eqs

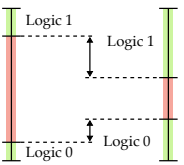
$$Y = \overline{A + B}$$

Boolean Equations & Algebra

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

Karnaugh Maps

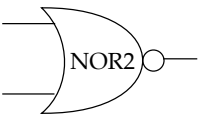
Combinational Logic



Bits

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

Truth Tables

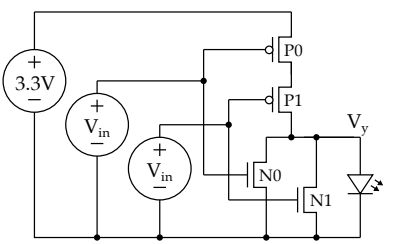


Logic Gates

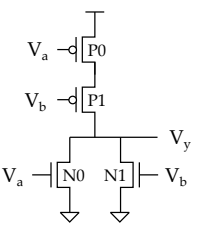
$t_{pd} = 3\tau$
$t_{cd} = 1\tau$

Delay Models

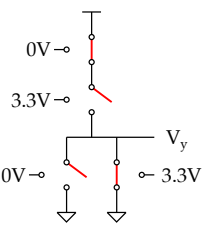
Digital Circuits



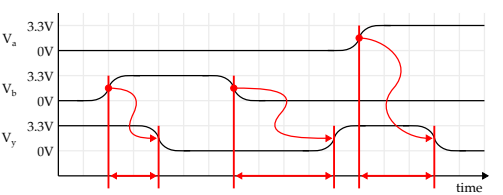
Circuit Diagrams



Transistor-Level Schematics



Switch-Level Model



Delay Models