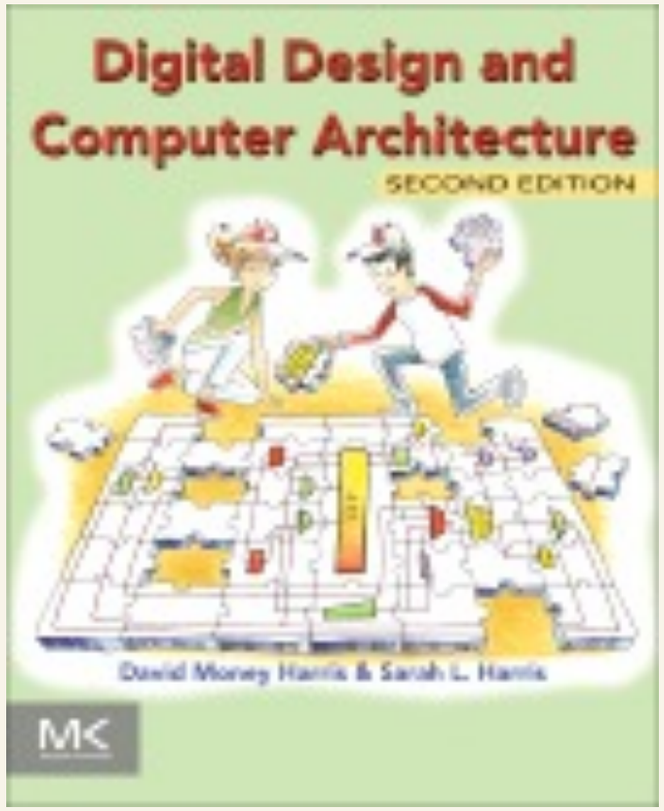
Combinational Logic Design

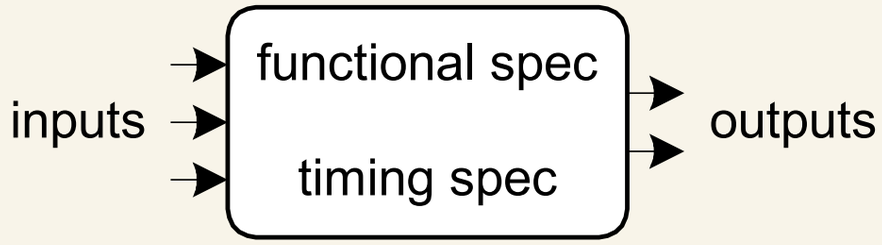
- Introduction
- Boolean Equations
- Boolean Algebra
- From Logic to Gates
- From Gates to Transistors
- X's and Z's
- Karnaugh Maps
- Combinational Building Blocks
- Timing

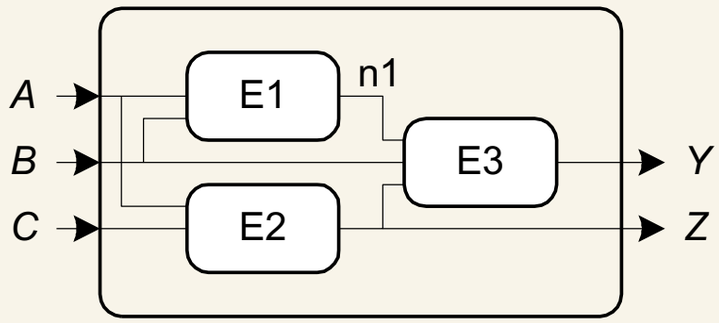


Chap 2

Introduction

- A logic circuit is composed of :
  - Inputs
  - Outputs
  - Functional specification
  - Timing specification

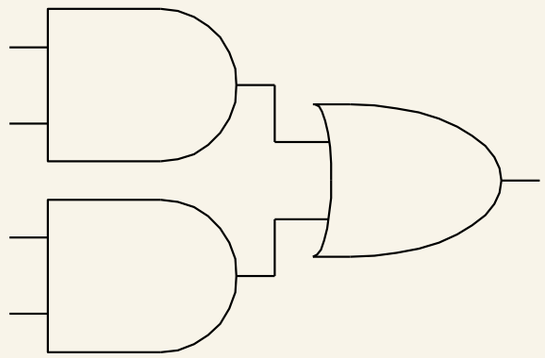




Nodes
- *Inputs: A, B, C*
- *Outputs: Y, Z*
- *Internal: n1*

Circuit elements
- *E1, E2, E3*
- *Each a circuit*

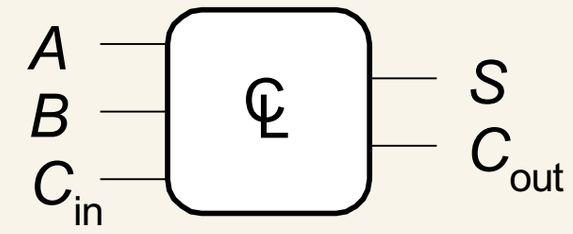
- Types of Logic Circuits
  - Combinational Logic
    - » Memoryless
    - » Outputs determined by current values of inputs
  - Sequential Logic
    - » Has memory
    - » Outputs determined by previous and current values of inputs



Example of combinational composition
- *Every element is combinational*
- *Every node is either an input or connects to exactly one output*
- *The circuit contains no cyclic paths*

Boolean Equations

- Functional specification of outputs in terms of inputs



$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

- Some definitions
  - **Complement** : variable with a bar over it
    - » $\overline{A}, \overline{B}, \overline{C}$
  - **Literal** : variable or its complement
    - » $A, \overline{A}, B, \overline{B}, C, \overline{C}$
  - **Implicant** : product of literals
    - » $A\overline{B}C, AC, \overline{B}C$
  - **Minterm** : product that includes all input variables
    - » $A\overline{B}C, \overline{A}\overline{B}C, ABC$

- Sum-of-Products (SOP) Form

  - All equations can be written in SOP form
  - Each row has a minterm
  - A minterm is a product (AND) of literals
  - Each minterm is TRUE for that row (and only that row)
  - Form function by ORing minterms where output is 1
  - Thus, a sum (OR) of products (AND terms)

A	*B*	*Y*	minterm	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

$$Y = \mathrm{F}(A, B) = \overline{A}B + AB = \Sigma(1, 3)$$

Boolean Algebra

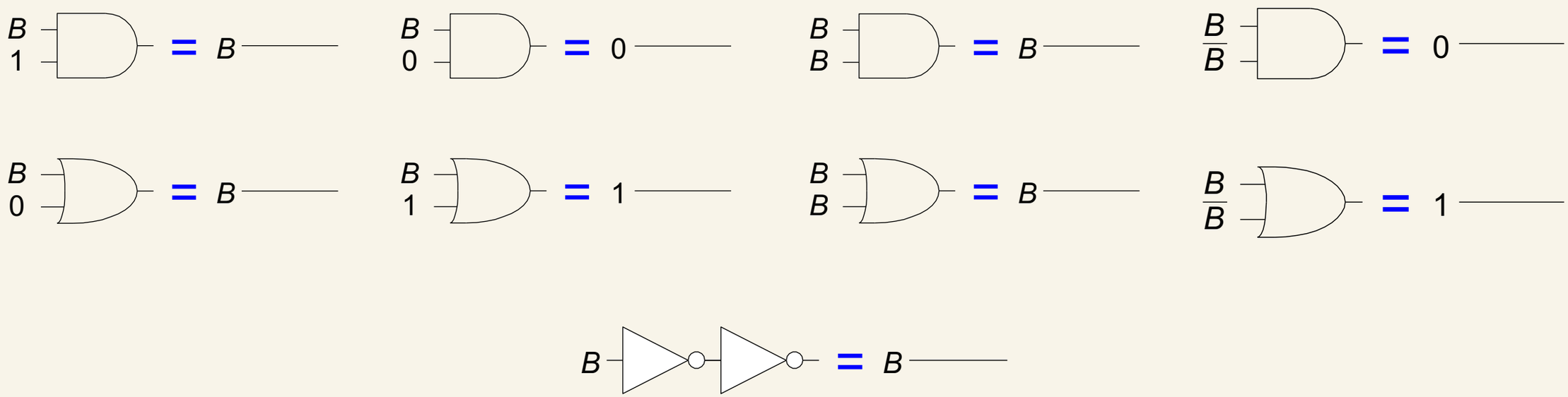
- Axioms and theorems to **simplify** Boolean equations
- Like regular algebra, but simpler: variables have only two values (1 or 0)
- **Duality** in axioms and theorems:
  - ANDs and ORs, 0's and 1's interchanged

- Axioms

Number	Axiom	Dual	Name
A1	B = 0 if B ≠ 1	B = 1 if B ≠ 0	Binary Field
A2	0 = 1	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

- Theorems of One 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



- Theorems of Several Variables

#	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

- Prove
  - Example : T9 - Covering

Number	Theorem	Name
T9	B• (B+C) = B	Covering

» Method 1: Perfect Induction

B	*C*	*(B+C)*	*B(B+C)*
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

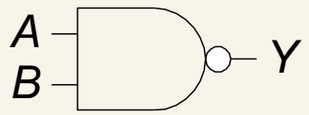
» Method 2: Prove true using other axioms and theorems

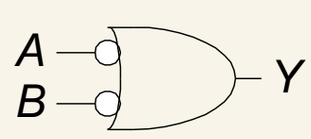
$$
\begin{aligned}
B \bullet (B+C) &= B \bullet B + B \bullet C && \text{T8: Distributivity} \\
&= B + B \bullet C && \text{T3: Idempotency} \\
&= B \bullet (1 + C) && \text{T8: Distributivity} \\
&= B \bullet (1) && \text{T2: Null element} \\
&= B && \text{T1: Identity}
\end{aligned}
$$

- De Morgan's Theorem

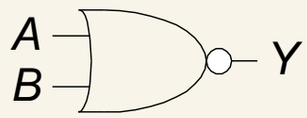
#	Theorem	Dual	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots$	DeMorgan's Theorem

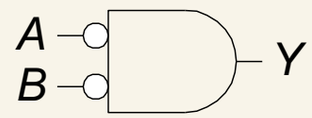
- $Y = \overline{AB} = \overline{A} + \overline{B}$





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





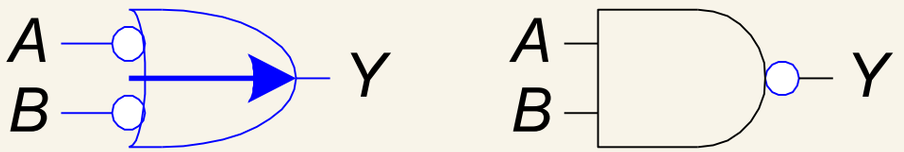
- De Morgan's Theorem Examples

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

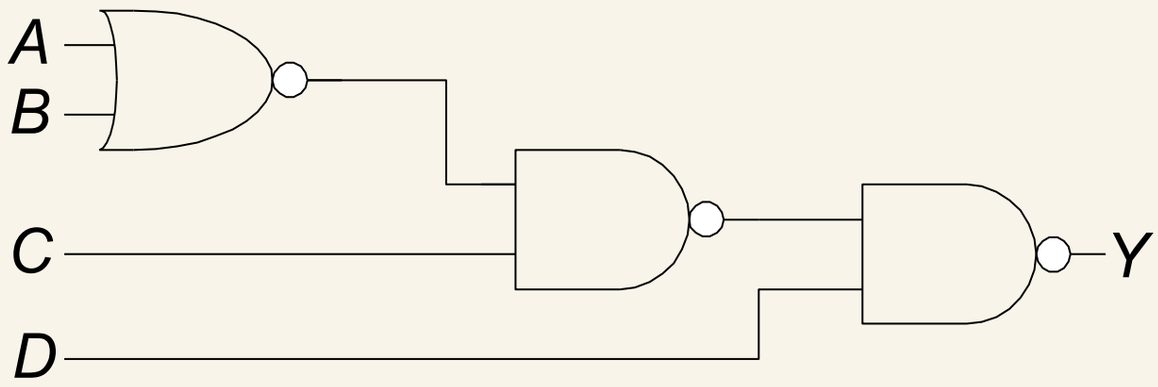
$$Y = \overline{(\overline{\overline{AC}E+\overline{D}) + B}}$$
$$= (\overline{\overline{\overline{AC}E+\overline{D}}}) \bullet \overline{B}$$
$$= (\overline{\overline{\overline{AC}E} \bullet \overline{\overline{D}}}) \bullet \overline{B}$$
$$= ((\overline{\overline{AC}}+\overline{E}) \bullet D) \bullet \overline{B}$$
$$= ((AC+\overline{E}) \bullet D) \bullet \overline{B}$$
$$= (ACD + D\overline{E}) \bullet \overline{B}$$
$$= A\overline{B}CD + \overline{B}D\overline{E}$$

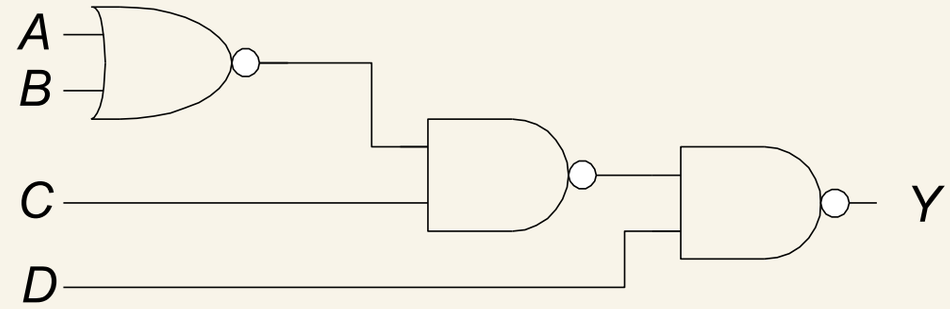
- Bubble Pushing
  - Backward:
    - » Body changes
    - » Adds bubbles to inputs

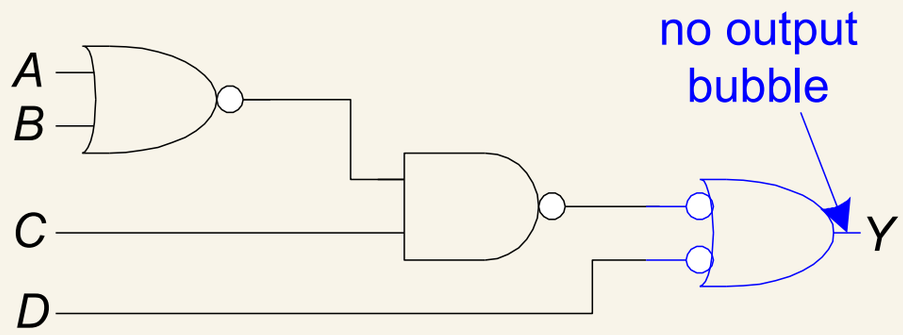
  - Forward:
    - » Body changes
    - » Adds bubble to output

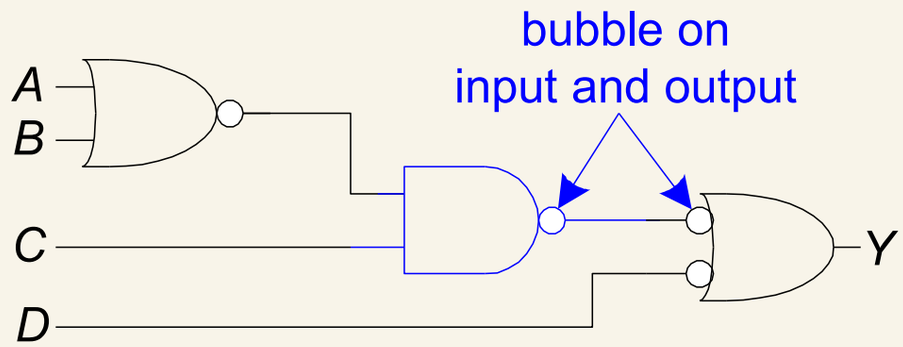


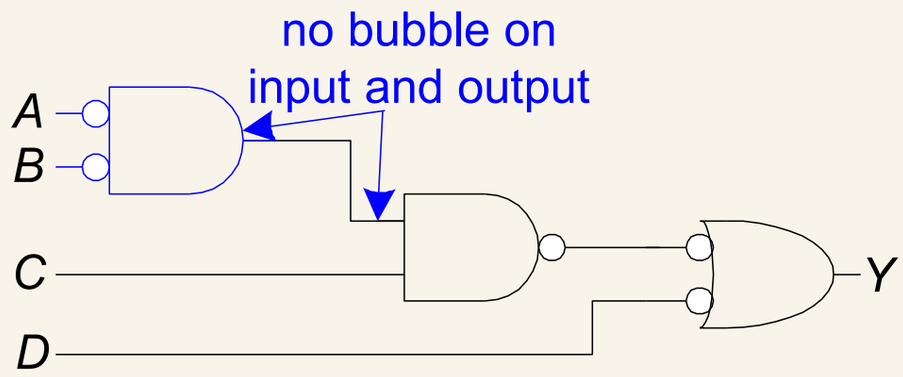
  - Rules
    - » Begin at output, then work toward inputs
    - » Push bubbles on final output back
    - » Draw gates in a form so bubbles cancel











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

- Simplifying Equations
  - Reducing an equation to the **fewest number of implicants**, where each implicant has the **fewest literals**
  - Trial and Error approach

Recall: A' = $\overline{A}$

  - Examples :

Y = A(AB + ABC)

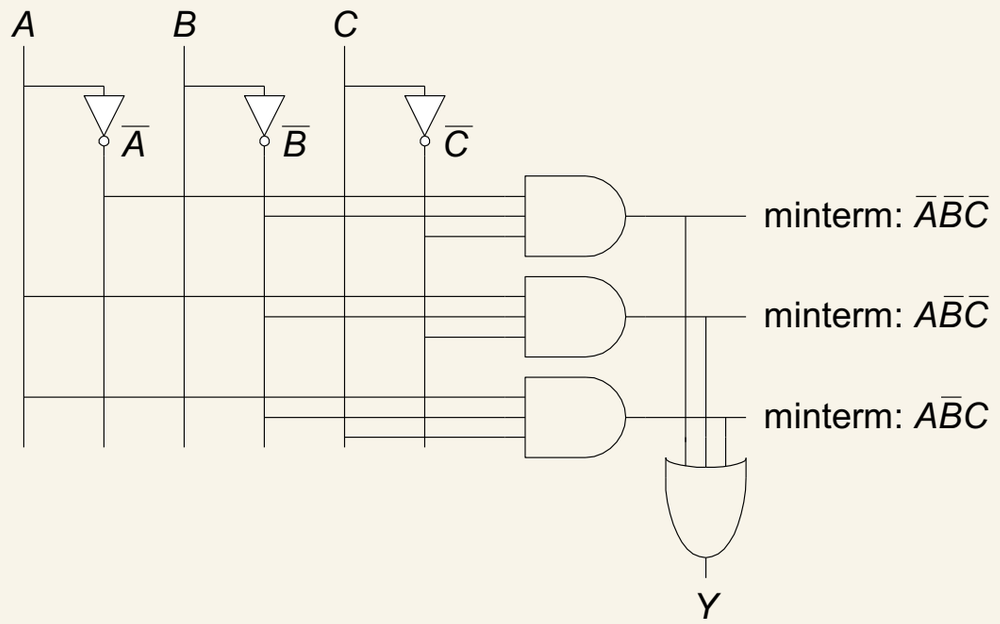
= *A*(*AB*(1 + *C*))	T8: Distributivity
= *A*(*AB*(1))	T2': Null Element
= *A*(*AB*)	T1: Identity
= (*AA*)*B*	T7: Associativity
= *AB*	T3: Idempotency

Y = AB'C + ABC + A'BC

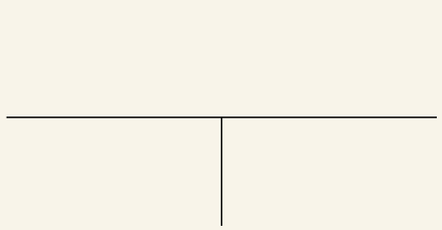
= AB'C + **ABC + ABC** + A'BC	T3': Idempotency
= (AB'C+ABC) + (ABC+A'BC)	T7': Associativity
= AC + BC	T10: Combining

From Logic to Gates

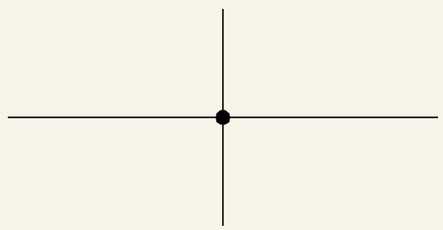
- Two-level logic: ANDs followed by ORs
- Example: $Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$



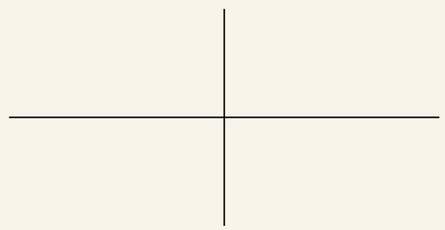
wires connect at a T junction



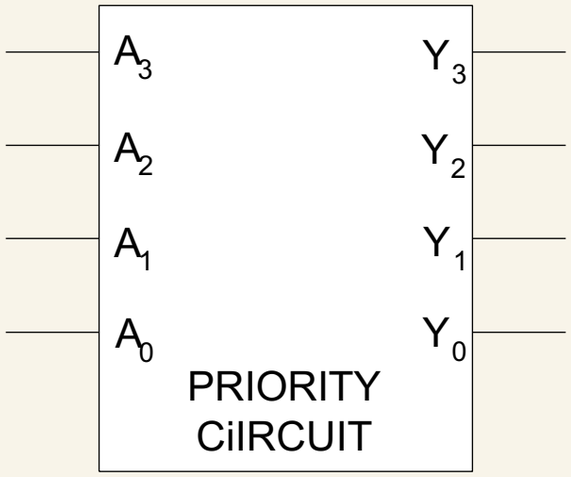
wires connect at a dot



wires crossing without a dot do not connect



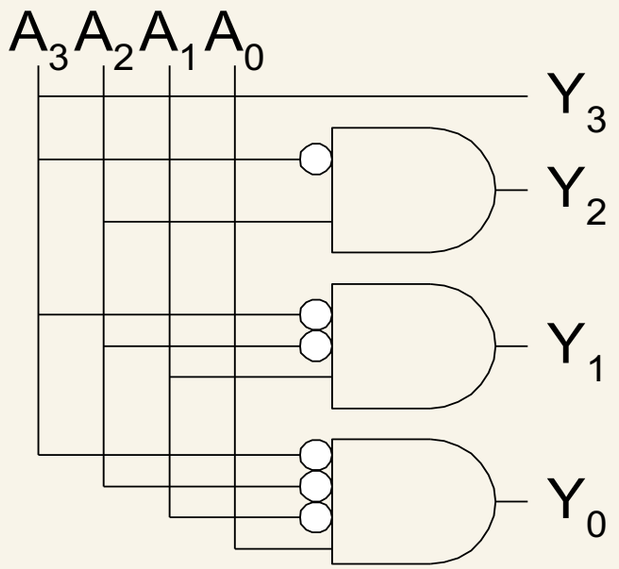
- Example : Priority Circuit



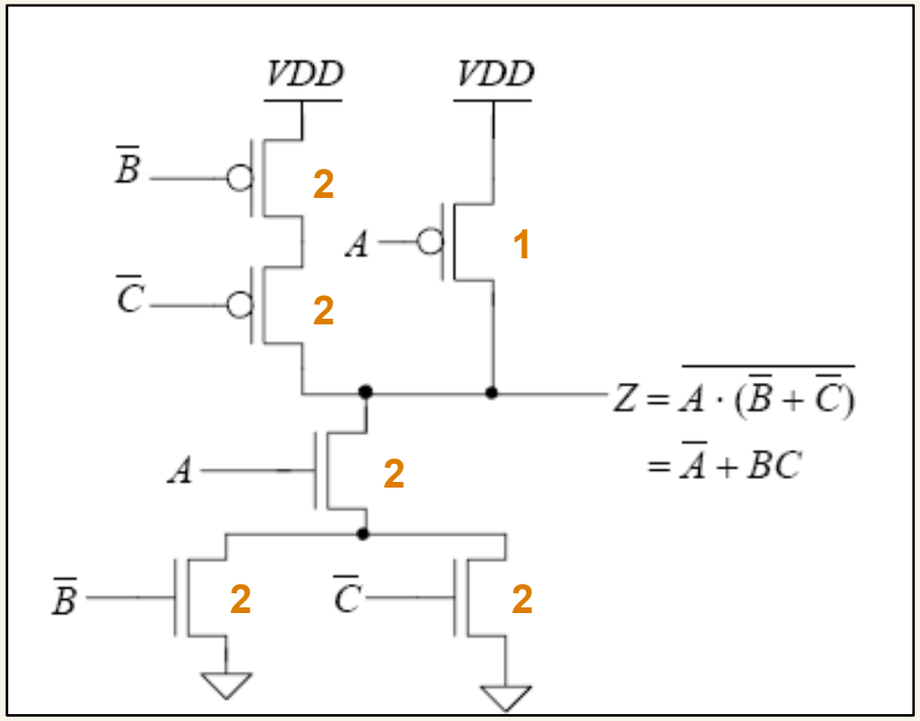
A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	0	**0**	**0**	**1**	**0**
0	0	1	1	**0**	**0**	**1**	**0**
0	1	0	0	**0**	**1**	**0**	**0**
0	1	0	1	**0**	**1**	**0**	**0**
0	1	1	0	**0**	**1**	**0**	**0**
0	1	1	1	**0**	**1**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**0**
1	0	0	1	**1**	**0**	**0**	**0**
1	0	1	0	**1**	**0**	**0**	**0**
1	0	1	1	**1**	**0**	**0**	**0**
1	1	0	0	**1**	**0**	**0**	**0**
1	1	0	1	**1**	**0**	**0**	**0**
1	1	1	0	**1**	**0**	**0**	**0**
1	1	1	1	**1**	**0**	**0**	**0**

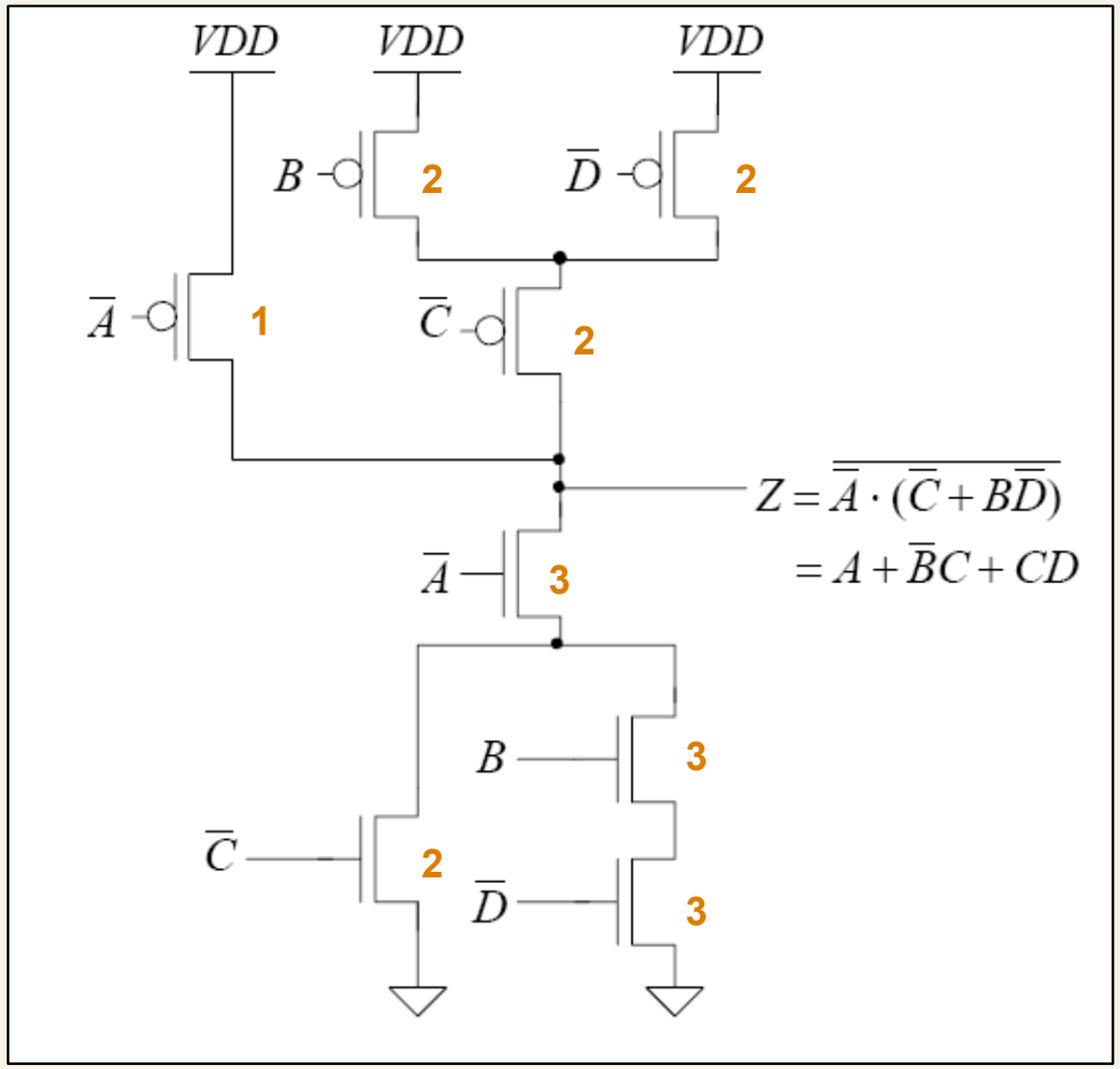
– Don’t Cares

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	x	**0**	**0**	**1**	**0**
0	1	x	x	**0**	**1**	**0**	**0**
1	x	x	x	**1**	**0**	**0**	**0**



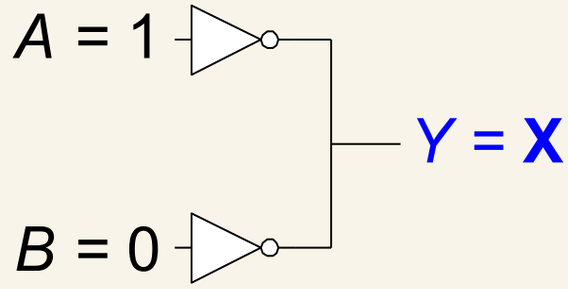
From Gates to Transistor





Contention : X

- **Contention**: circuit tries to drive output to 1 and 0
  - Actual value somewhere in between
  - Could be 0, 1, or in forbidden zone
  - Might change with voltage, temperature, time, noise
  - Often causes excessive power dissipation

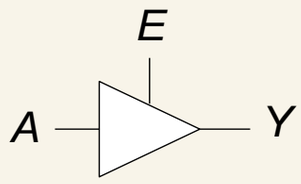


- Warnings :
  - Contention usually indicates a **bug**.
  - **X** is used for "**don't care**" and **contention**
    - » Look at the context to tell them apart.

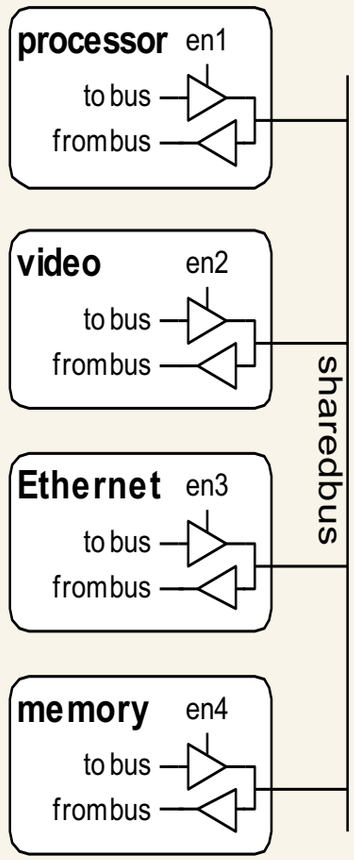
Floating : Z

- Floating, high impedance, open, high Z
- Floating output might be 0, 1, or somewhere in between

Tristate Buffer



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1



- Floating nodes are used in tristate busses
  - Many different drivers
  - Exactly one is active at once

Karnaugh Maps (K-Maps)

- Boolean expressions can be minimized by combining terms
- K-maps minimize equations graphically
- $PA + P\overline{A} = P$

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

Y \ AB / C	00	01	11	10
0	1	0	0	0
1	1	0	0	0

Y \ AB / C	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Y \ AB / C	00	01	11	10
0	1	0	0	0
1	1	0	0	0

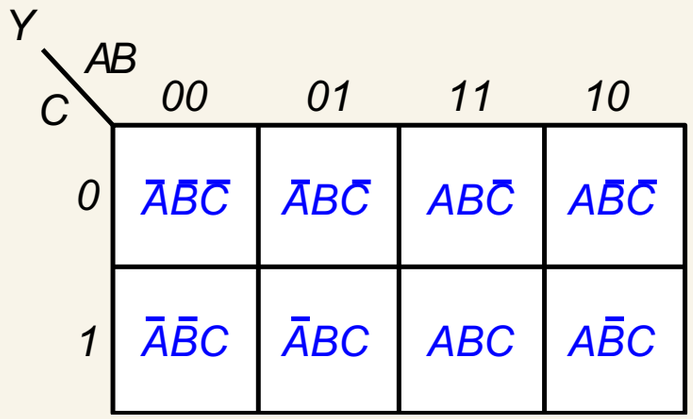
$$Y = \bar{A}\bar{B}$$

- Circle 1's in adjacent squares
- In Boolean expression, include only literals whose true and complement form are not in the circle

- K-Map Rules

  - » Every 1 must be circled at least once
  - » Each circle must span a power of 2 (i.e. 1, 2, 4) squares in each direction
  - » Each circle must be as large as possible
  - » A circle may wrap around the edges
  - » A “don't care” (X) is circled only if it helps minimize the equation

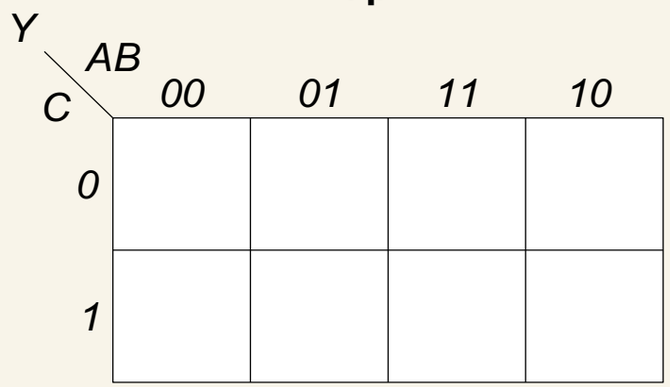
- Example



Truth Table

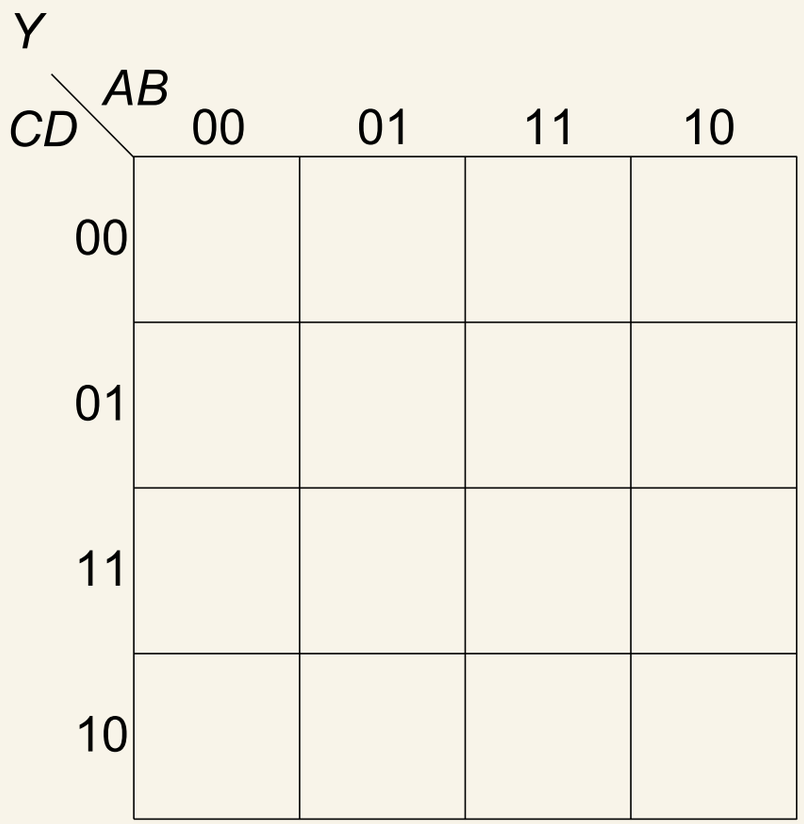
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	**1**
0	1	1	**1**
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	**1**

K-Map

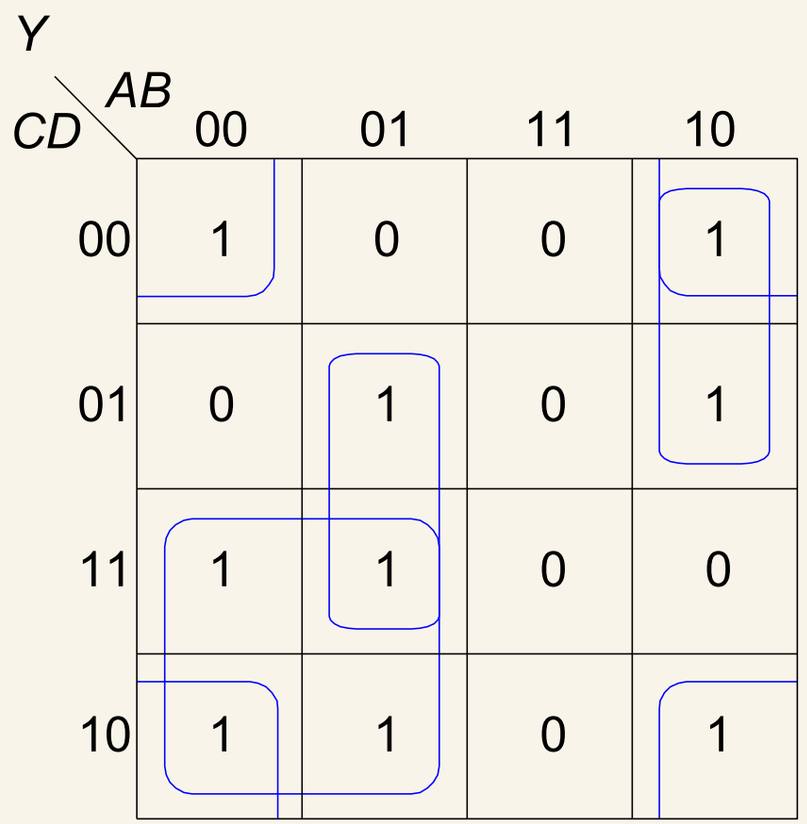


- Example of K-Map with 4 variables

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



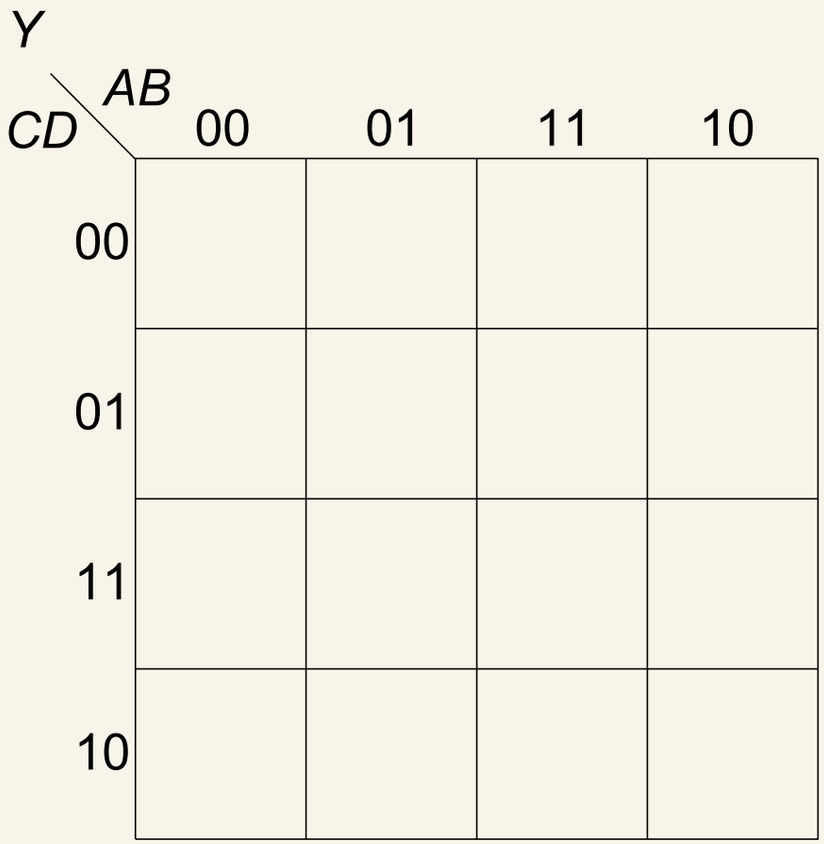
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



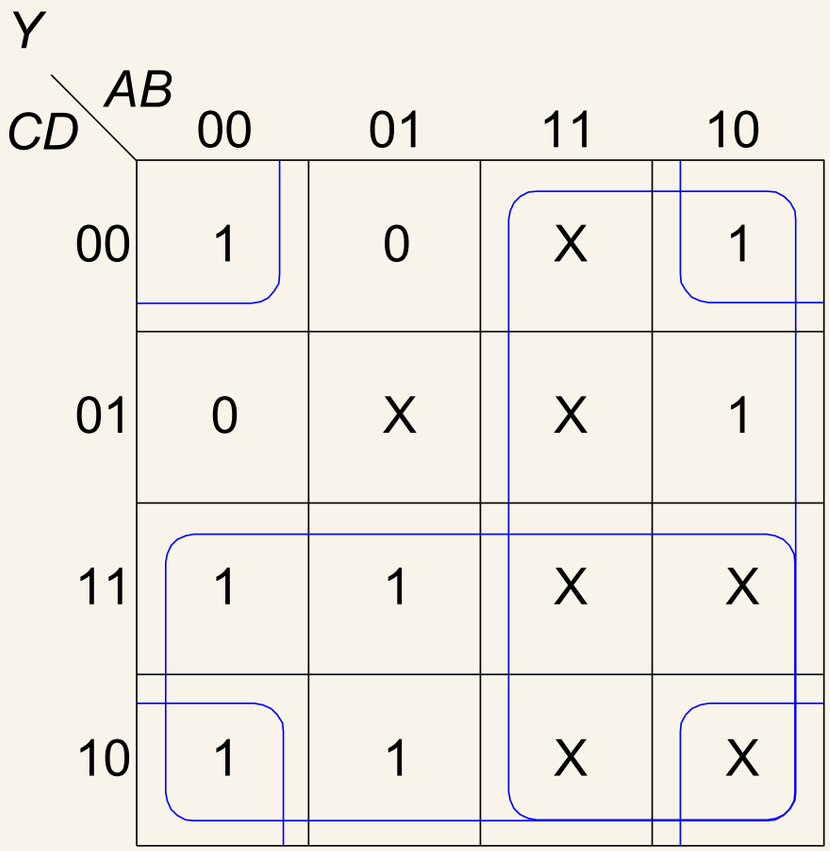
$$Y = \bar{A}C + \bar{A}BD + A\bar{B}\bar{C} + \bar{B}\bar{D}$$

- Example of K-Map with Don't Cares

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	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



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	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



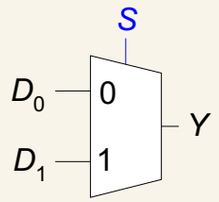
$$Y = A + \bar{B}\bar{D} + C$$

Combinational Building Blocks

- Multiplexer (Mux)
  - Selects between one of N inputs to connect to output
  - $\log_2$N-bit select input – control input

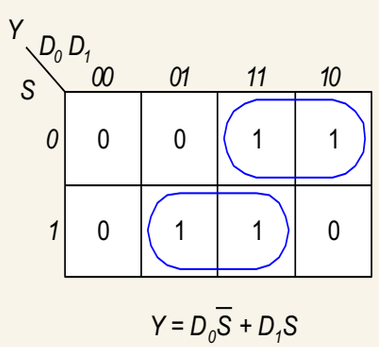
  - Multiplexer Implementations
    - » Logic gates
    - » Tristates

2:1 Mux

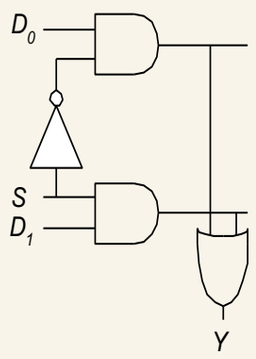


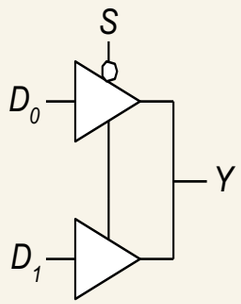
S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

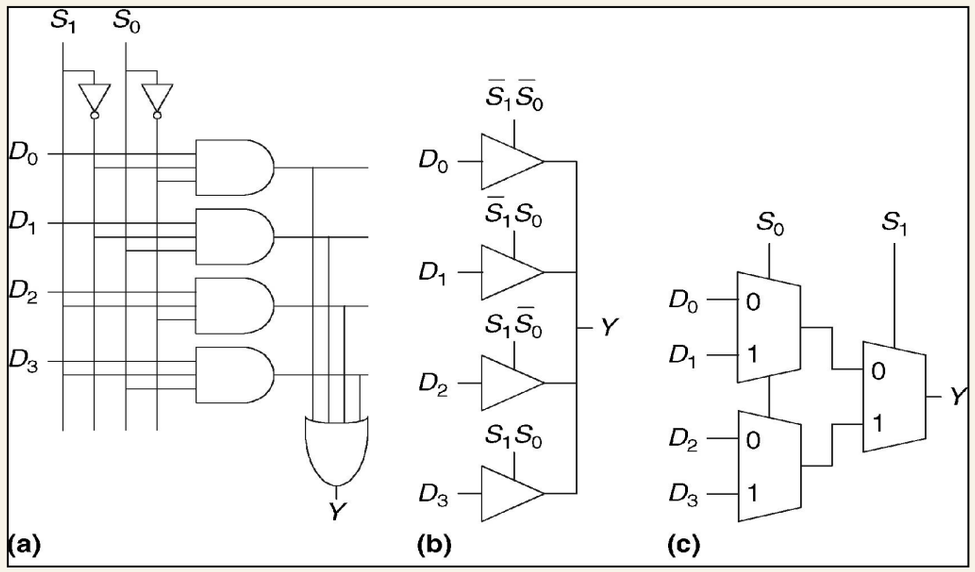
S	Y
0	D_0
1	D_1



$Y = D_0\overline{S} + D_1S$



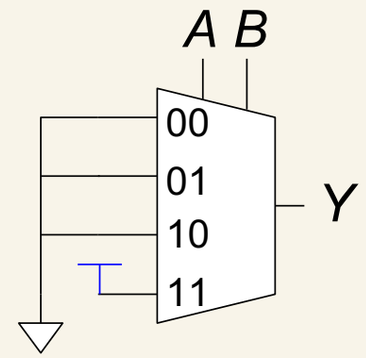




- Logic using Multiplexers
  - Using mux as a lookup table

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

$Y = AB$

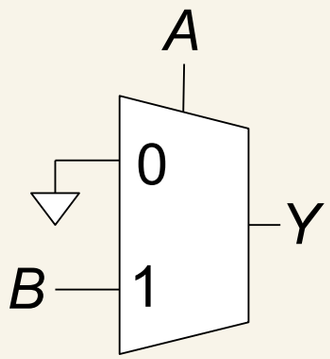


  - Reducing the size of the mux

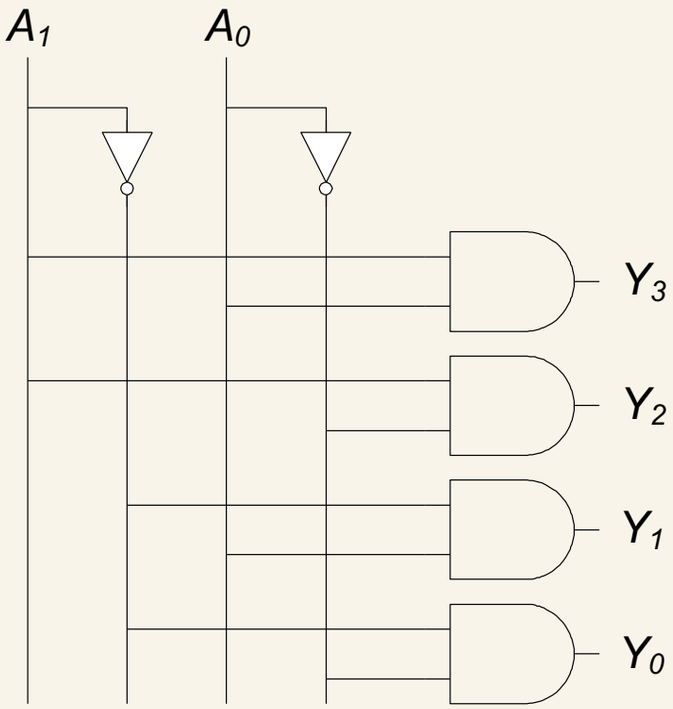
$Y = AB$

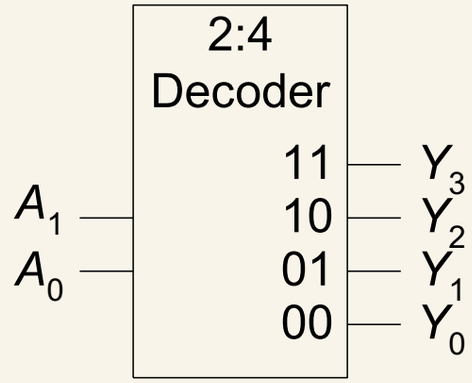
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	Y
0	0
1	B

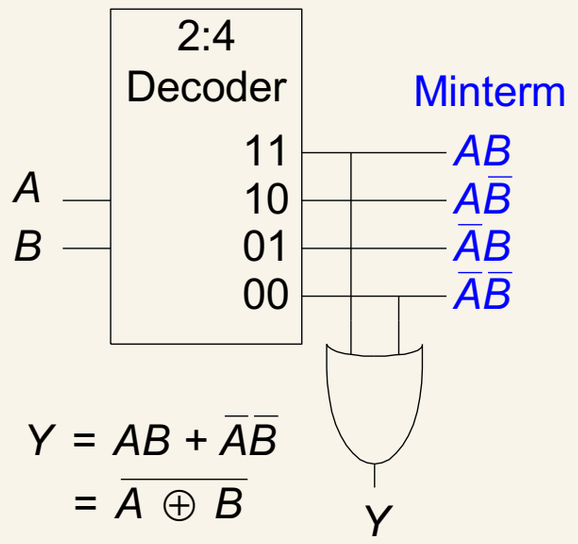


- Decoder
  - N inputs, 2^N outputs
  - One-hot outputs : only one output HIGH at once
  - Implementation





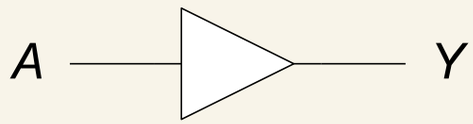
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

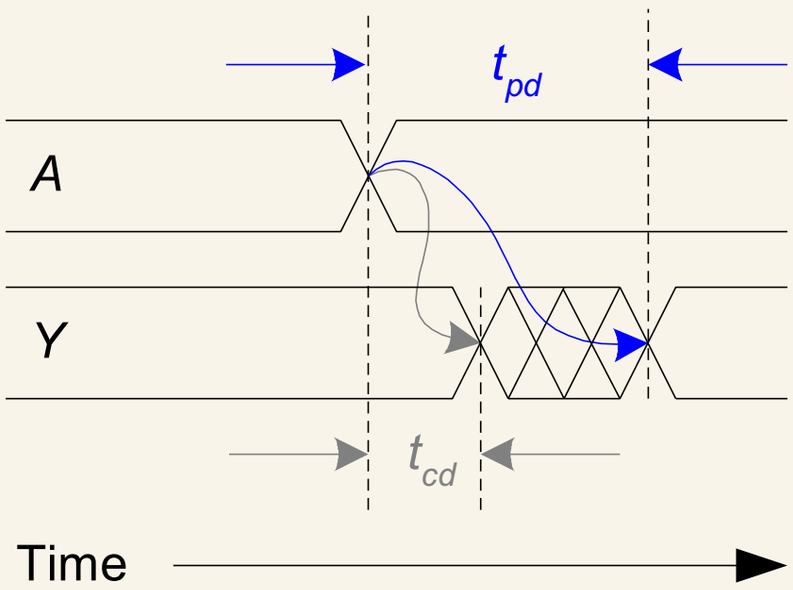


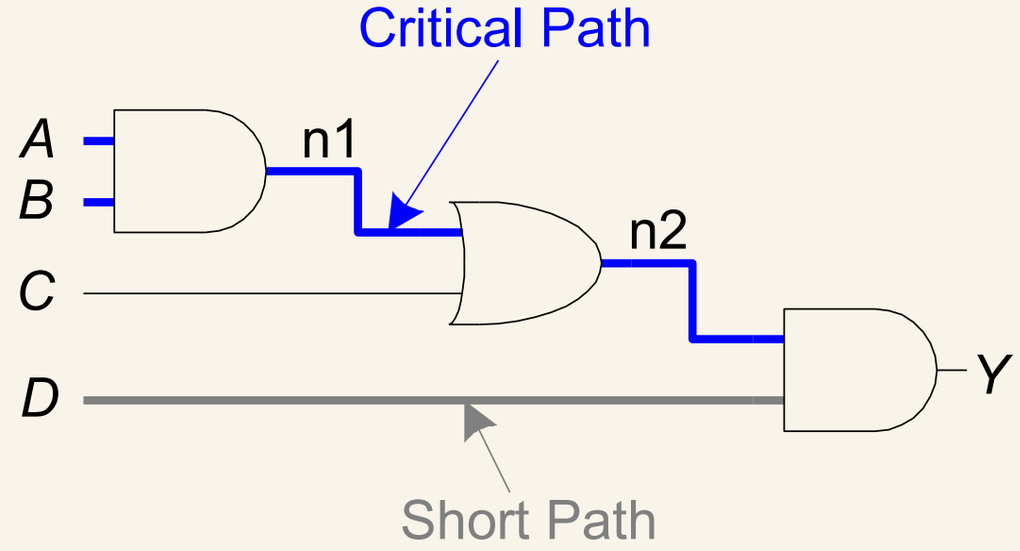
Logic Using Decoders
OR minterms

Timing

- Propagation & Contamination Delay
  - Propagation delay: t_{pd} = max delay from input to output
  - Contamination delay: t_{cd} = min delay from input to output



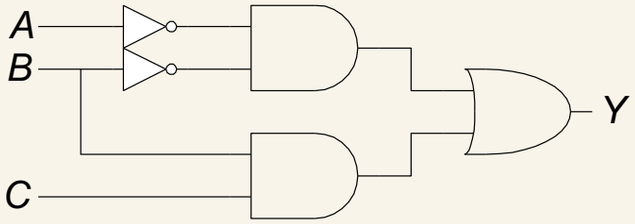


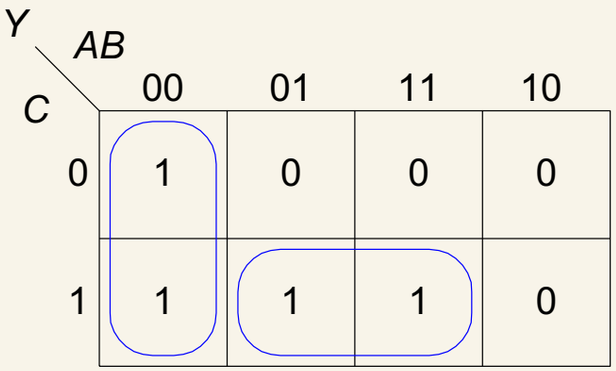


Critical (Long) Path: $t_{pd} = 2t_{pd\_\mathrm{AND}} + t_{pd\_\mathrm{OR}}$

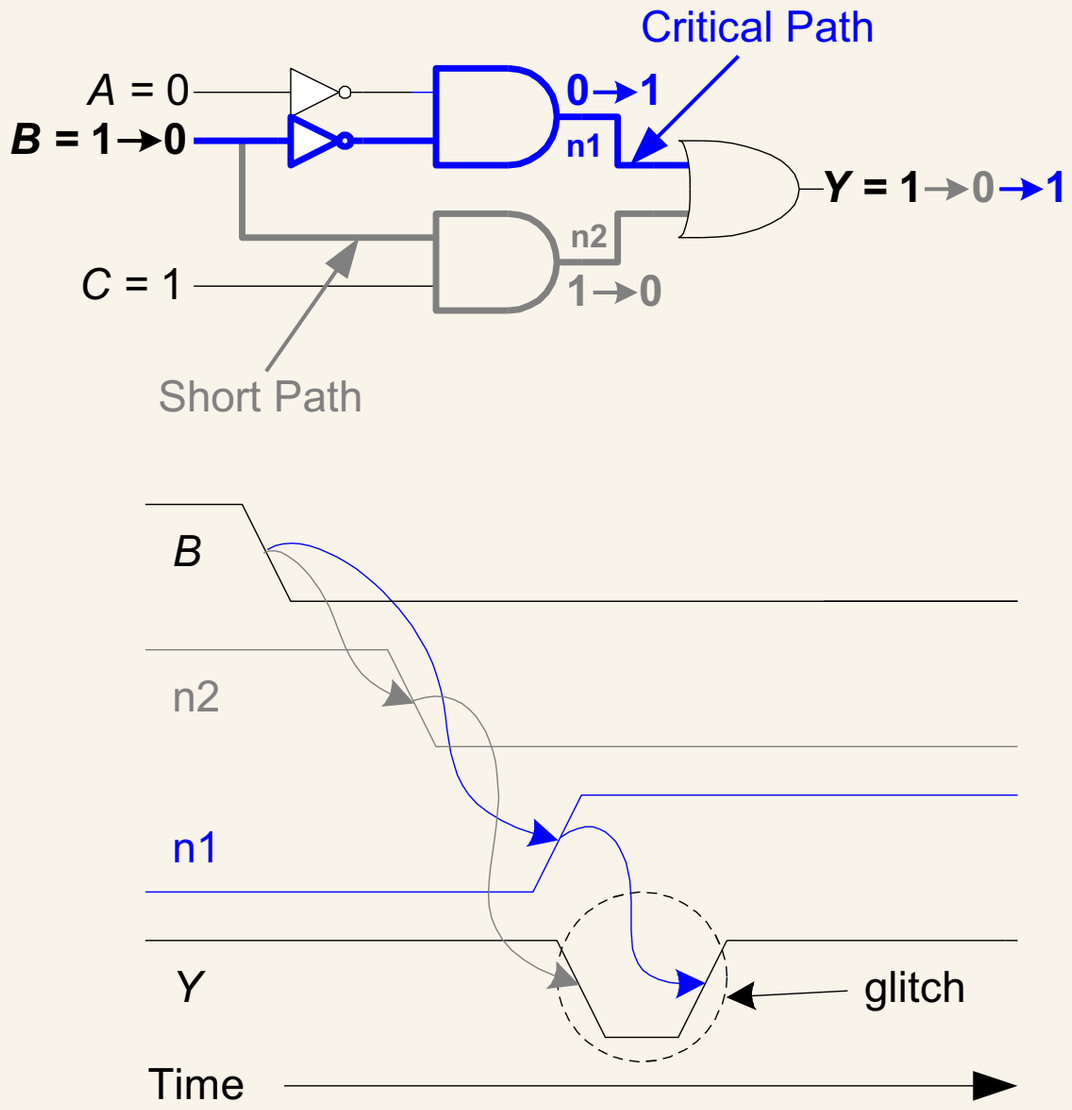
Short Path: $t_{cd} = t_{cd\_\mathrm{AND}}$

- Glitches
  - When a single (or more) input change causes an output to change multiple times

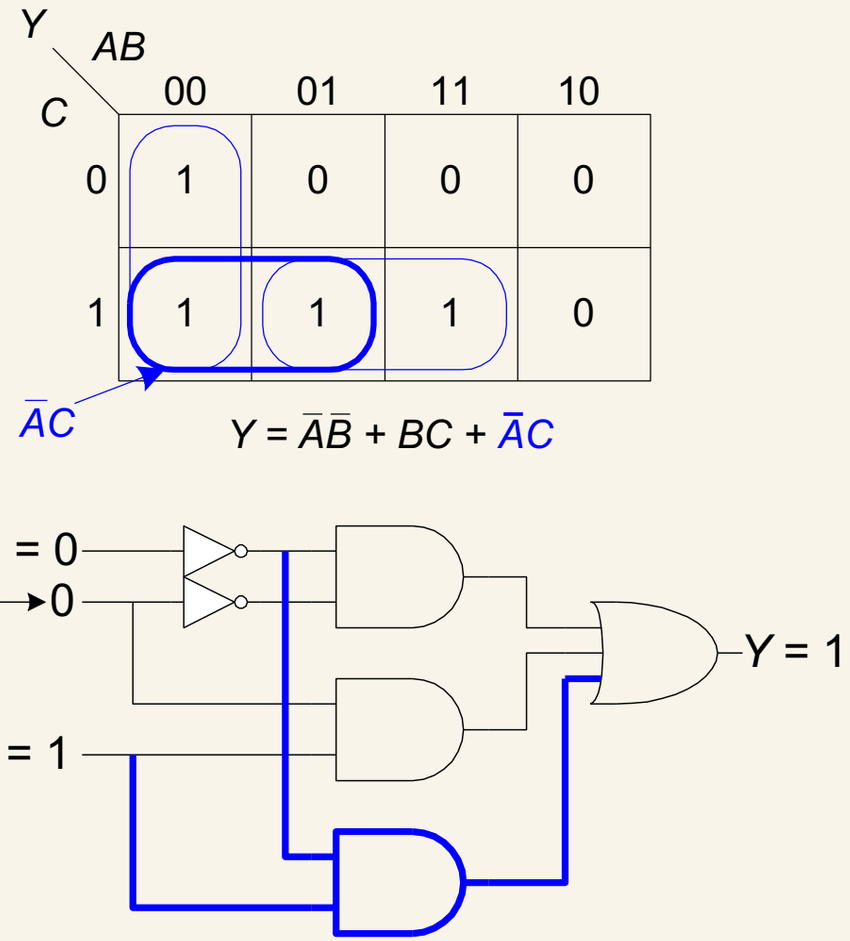




$Y = \bar{A}\bar{B} + BC$



- Fixing the glitch



- Glitches don’t cause problems because of **synchronous design** conventions (see next chapter)