§2.1 INTRODUCTION

CIRCUIT

(1) one or more discrete-valued input terminals

(2) " output "

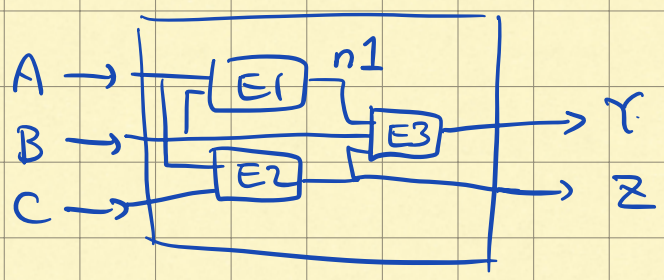
(3) a functional specification : relationship between inputs & outputs

(4) a timing specification :

delay between inputs changing and output responding.

Circuits are composed of elements and nodes

→ A element: circuits
→ A node : wire — input
output
internal

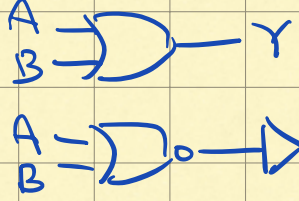


Elements : E1, E2, and E3 Nodes : A, B, C, Y, Z, and n1.

Combinational → outputs depend only on the current values of the inputs.
memoryless

Sequential → outputs depend on both current and previous values of the inputs
has memory

indicates combinational logic.



There may exists several implementations for one logic circuit.

/ represents "bus,"
3 inputs and 2 outputs in this case.

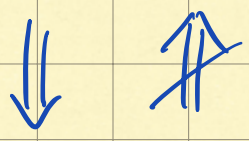
The rule of combinational composition:

- Every circuit element is itself combinational.
- Every node of the circuit is either designated as an input to the circuit or connects to exactly one output terminal

of a circuit element.

- The circuit contains no cyclic paths.

The rule of combinational composition



combinational logic circuit.

Connection to 3-Y's (e.g. microprocessor)

(1) well-defined function and interface.
=> Abstraction & Modularity.

(2) Building from smaller circuit
=> Hierarchy.

(3) The rule of combinational composition
=> Discipline or Regularity.

§2.2 BOOLEAN EQUATIONS

2.2.1 Terminology

$\overline{A}$ is a complement of A.

$A, \overline{A}, B, \overline{B}, \ldots$: literals.

A true form, $\overline{A}$ complementary form

AND: product or Implicant
minterm is a product involving all inputs.
OR: sum
maxterm is a sum involving all inputs.

2.2.2 Sum-of-Products 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	AB	m_3

Derive Boolean equation:

(SOP) → Add all minterms that makes $Y = 1$.

↑ $Y = \bar{A}B$

A	B	Y	minterm	name.
0	0	0	$\bar{A}\bar{B}$	m_0
0	1	1	$\bar{A}B$	m_1
1	0	0	$A\bar{B}$	m_2
1	1	1	AB	m_3

↑ $Y = \bar{A}B + AB$

⇒ <u>sum-of-products (SOP) canonical form.</u>

Also,

$$F(A, B) = \Sigma(m_1, m_3)$$
$$F(A, B) = \Sigma(1, 3)$$

2.2.3 Product-of-Sums Form

POS canonical form.

maxterm is sum of literals which makes FALSE for that row.

A	B	Y	maxterm.
0	0	→	$A+B$
0	1	→	$A+\bar{B}$
1	0	→	$\bar{A}+B$
1	1.	→	$\bar{A}+\bar{B}$

example

A	B	Y	Maxterm	Name
0	0	(0)	$A+B$	M_0 ✓
0	1	1	$A+\bar{B}$	M_1
1	0	(0)	$\bar{A}+B$	M_2 ✓
1	1	1	$\bar{A}+\bar{B}$	M_3

$F(A,B) = (A+B)(\bar{A}+B)$

$F(A,B) = \Pi(M_0, M_2)$

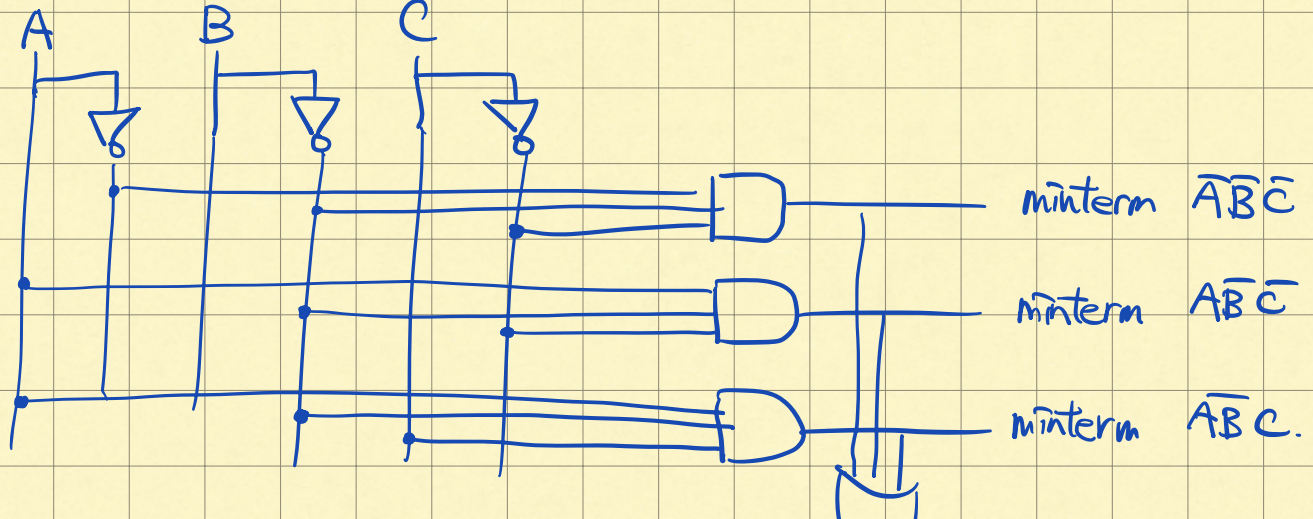
$= \Pi(0,2)$

§ 2.3 BOOLEAN ALGEBRA

§ 2.4 FROM LOGIC TO GATES

A schematic of

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



This style is called programmable logic array (PLAs)

§2.5 MULTILEVEL COMBINATIONAL LOGIC

SoP circuits were two-level logic.

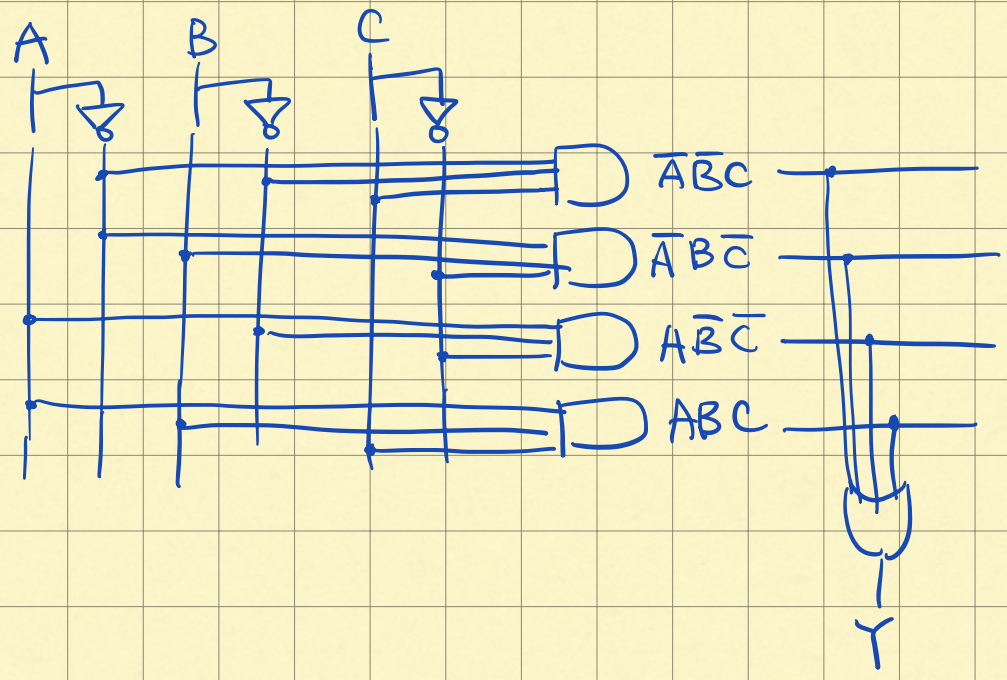
2.5.1 Hardware Reduction

Example 3-input XOR.

A	B	C	Y	minterm.
0	0	0	0	
0	0	1	1	$\bar{A}\bar{B}C$
0	1	0	1	$\bar{A}B\bar{C}$
0	1	1	0	
1	0	0	1	$A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

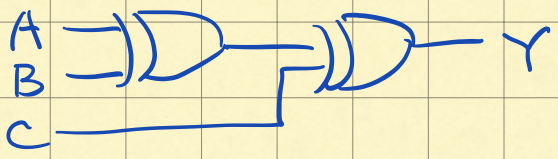
$$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC.$$

"Two-level logic"



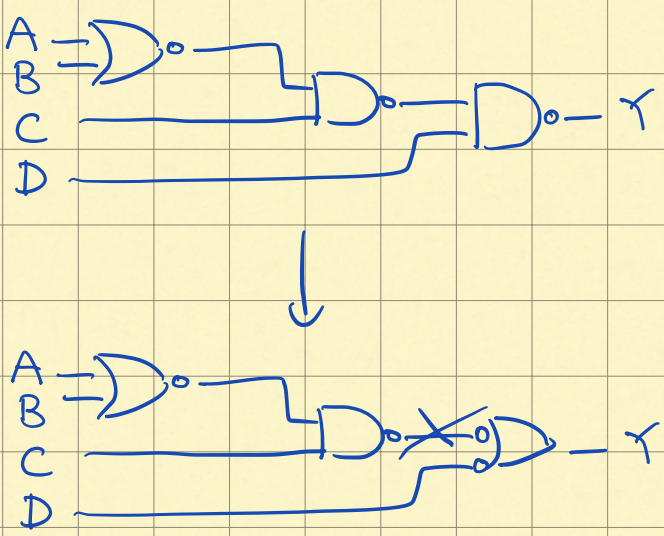
Instead,

$$A \oplus B \oplus C = (A \oplus B) \oplus C$$



In CMOS, NANDs and NORs are more efficient.

2.5.2 Bubble Pushing

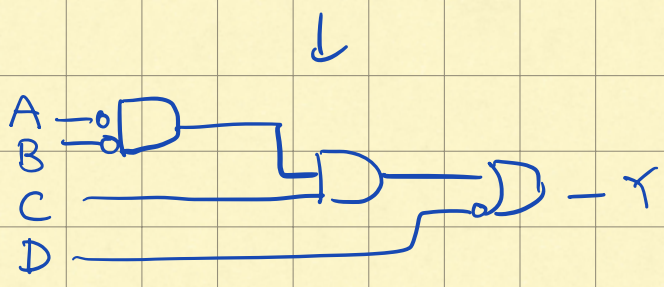


$$\{[(A+B)' \cdot C]' \cdot D\}' = Y$$

$$(XY)'$$

$$\downarrow$$

$$X'+Y'$$

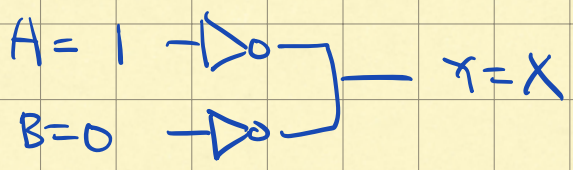


$$Y = \overline{A} \cdot \overline{B} \cdot C + \overline{D}$$

§ 2.6 X'S AND Z'S.

2.6.1 Illegal Value: X

- Unknown or Illegal value. "in circuit"



contention → unexpected value, often, in a forbidden zone.

- Uninitialized values. "in simulator"
- Don't care. "in truth table"

2.6.2 Floating Value: Z

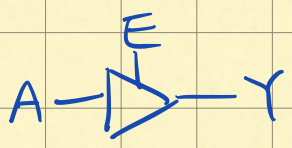
Indicates that a node is being driven neither HIGH nor LOW.

⇆ floating, high impedance, high Z.

(Forgot to connect input w or assume unconnected to be 0.

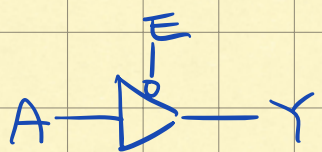
Example: tristate buffer.

① w/ Active high enable



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

② w/ Active low enable



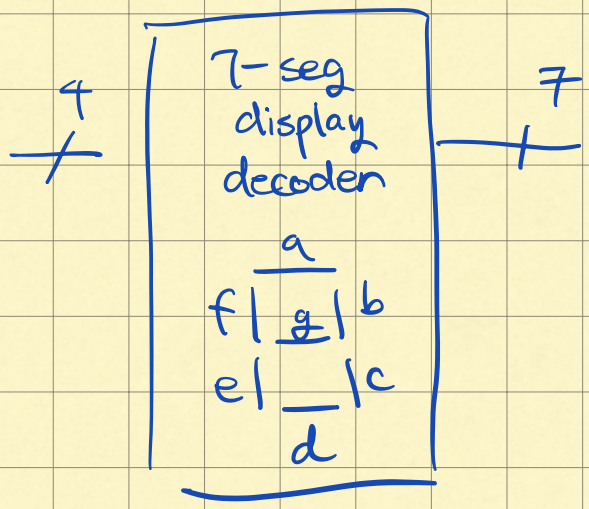
E	A	Y
0	0	0
0	1	1
1	0	Z
1	1	Z

Commonly used on busses that connect multiple chips.

In modern computers, higher speed is possible with point-to-point links. (connect directly.)

Example 2.10 SEVEN-SEGMENT DISPLAY DECODER

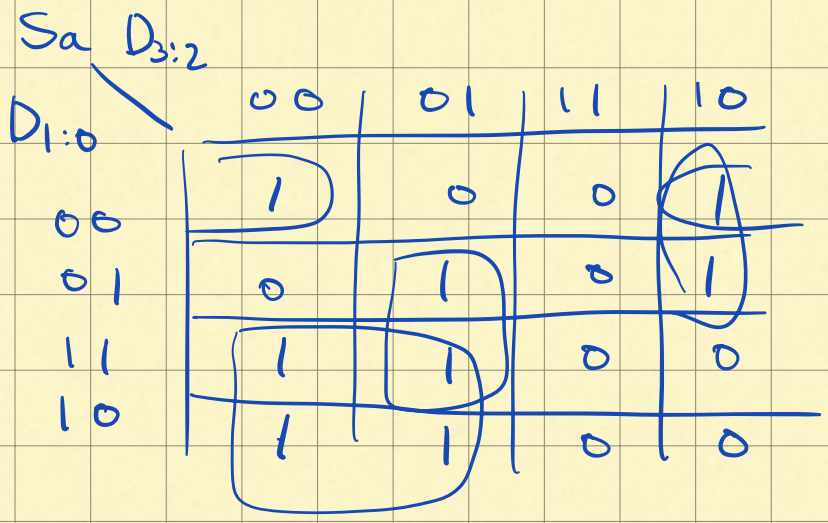
Takes a 4-bit data input $D_{3:0}$ and produces seven outputs.



	$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
(0)	0000	1	1	1	1	1	1	0
(1)	0001	0	1	1	0	0	0	0
(2)	0010	1	1	0	1	1	0	1
(3)	0011	1	1	1	1	0	0	1
(4)	0100	0	1	1	0	0	1	1
(5)	0101	1	0	1	1	0	1	1
(6)	0110	1	0	1	1	1	1	1
(7)	0111	1	1	1	0	0	0	0
(8)	1000	1	1	1	1	1	1	1
(9)	1001	1	1	1	0	0	1	1

others	0	0	0	0	0	0	0

K-map for...



$$S_a = \overline{D_3} D_1 + \overline{D_3} D_2 D_0 + \overline{D_2} \overline{D_1} \overline{D_0} + D_3 \overline{D_2} \overline{D_1}$$

⋮

and so on.

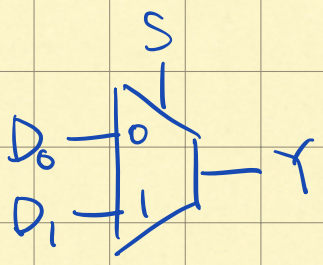
§ 2.8 COMBINATIONAL BUILDING BLOCKS

2.8.1 Multiplexers

They choose an output from among several inputs, based on the value of select signal.

⇆ mux

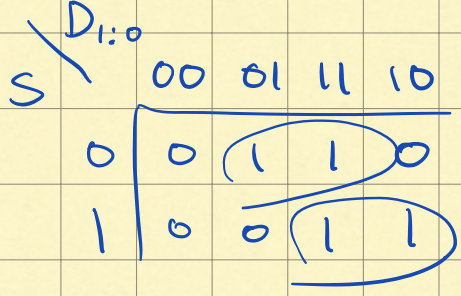
2:1 MUX



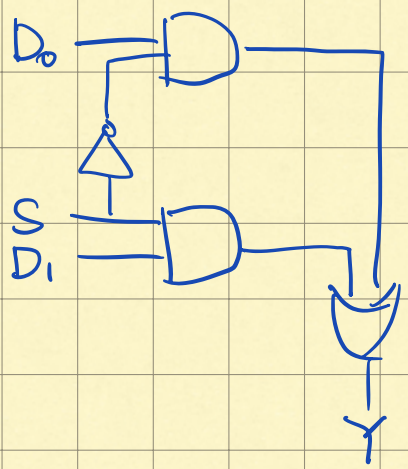
S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

S: control signal

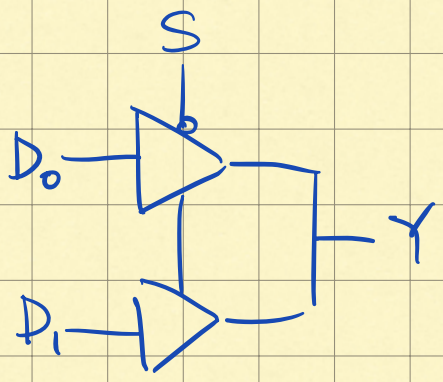
K-map:



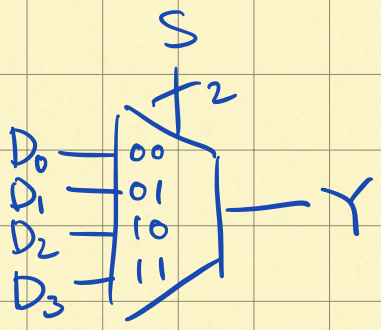
$\Rightarrow \boxed{Y = \overline{S} D_0 + S D_1}$



Alternatively, it can be built from tristate buffer.

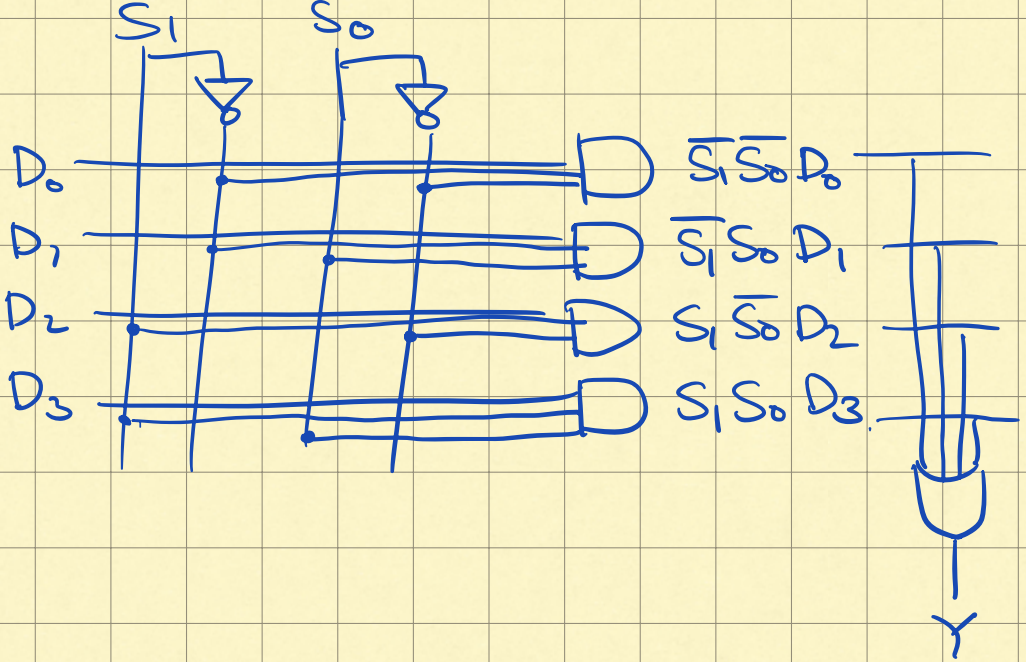


Wider MUX (4:1)

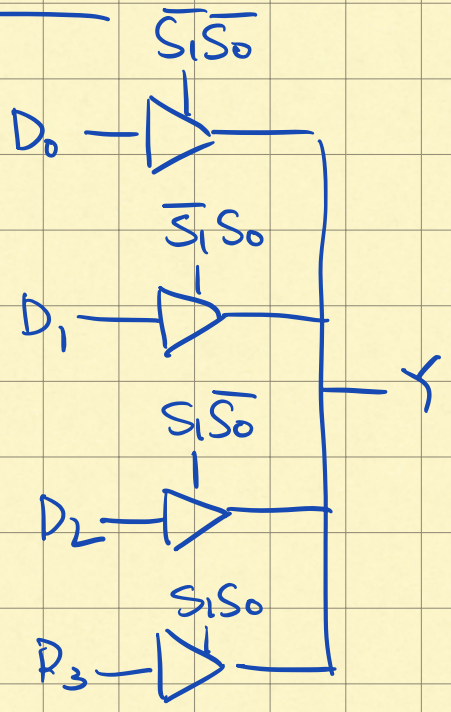


Implementation:

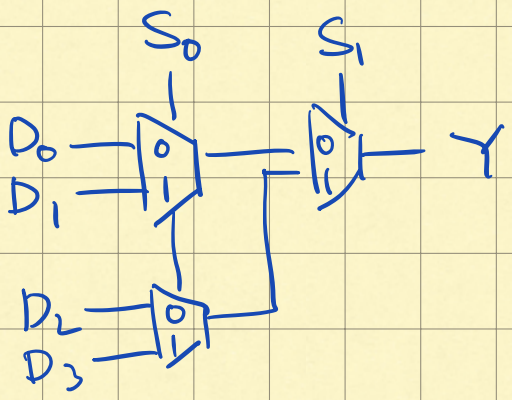
Two-level Logic:



Tristate:



Hierarchical:

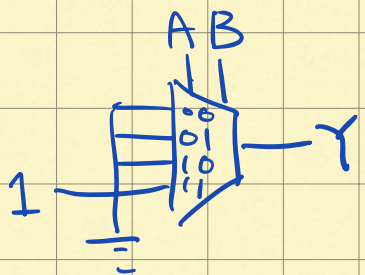


Multiplexer Logic

Can be used as LUT

By changing the data inputs, it can be reprogrammed.

ex) 4:1 MUX implementation of two-input AND



We can cut the MUX size in half using a 2^{N-1}-input MUX to perform N-input logic function.

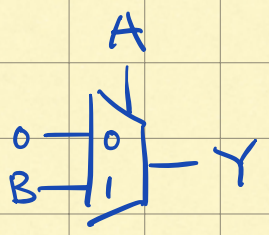
"AND"

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

→

A	Y
0	0
1	B

→



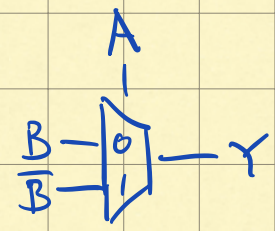
"XOR"

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

→

A	Y
0	B
1	$\overline{B}$

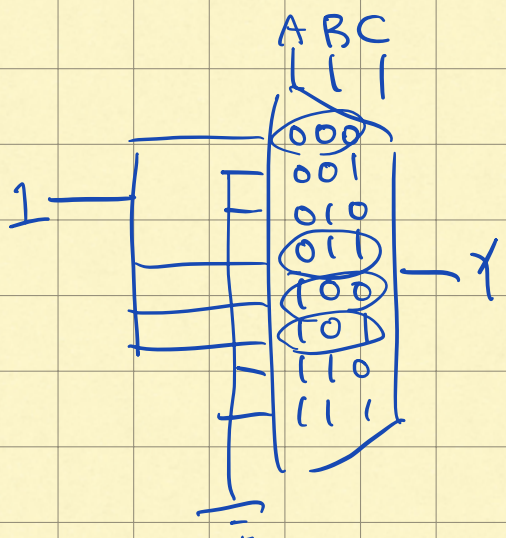
→



Example 2.12 LOGIC WITH MUX.

$Y = A\overline{B} + \overline{B}\overline{C} + \overline{A}BC$ w/ 8:1 MUX.

$10X \quad X00 \quad 011$

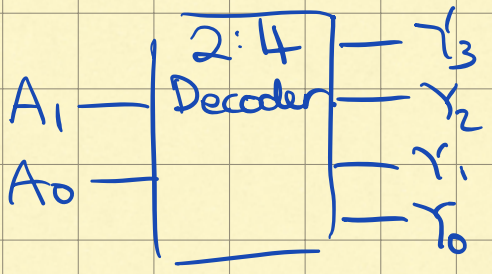


2.8.2 Decoders

N inputs → 2^N outputs.

It asserts exactly one of its outputs depending on the input combination.

2:4 Decoder



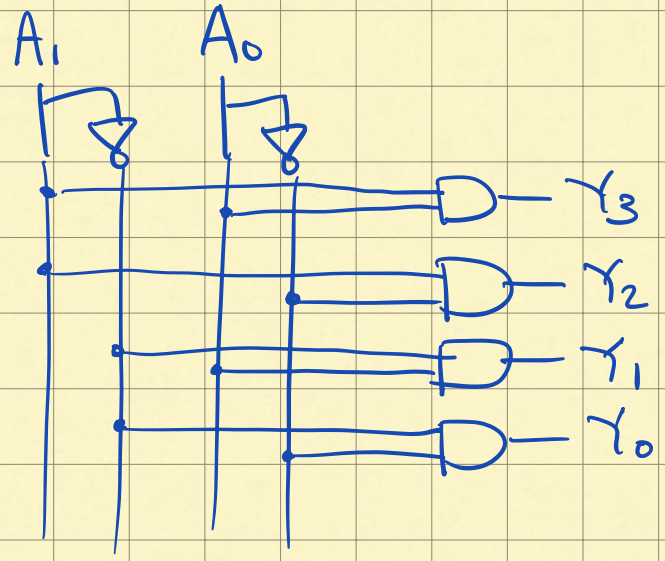
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

The outputs are called one-hot.

Example 2.14 DECODER IMPLEMENTATION

Implement 2:4 decoder w/ AND and NOT gates.

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



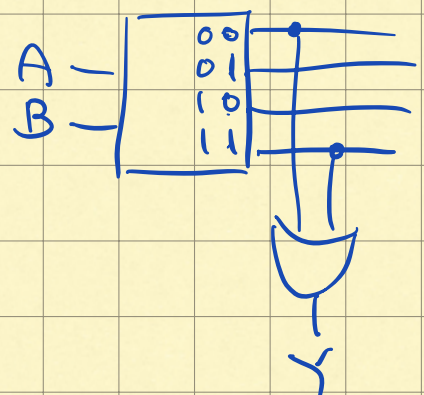
Decoder Logic

Using decoders to build logic, express as a truth table or SOP form.

N-inputs w/ M-1's in the truth table can be built w/ an $N:2^N$ decoder and M-input OR gate.

Ex) XNOR

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

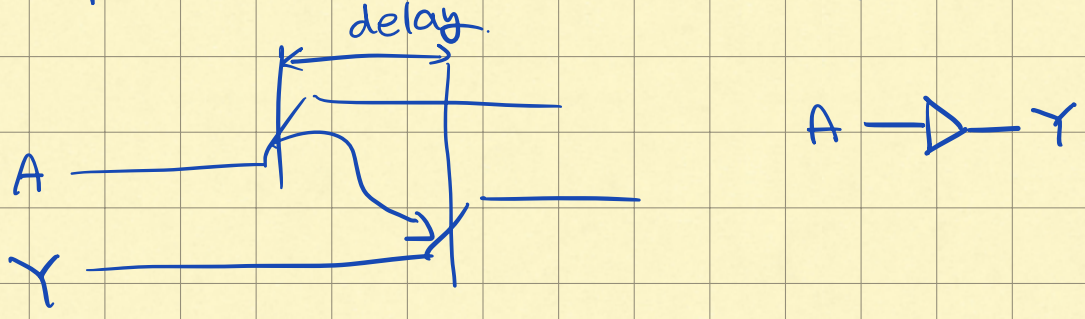


(Will be applied to the building of ROM.)

§ 2.9 TIMING

timing : making a circuit run fast.

An output takes time to change.



timing diagram

The transition from LOW to HIGH
→ the rising edge.

HIGH to LOW
→ the falling edge.

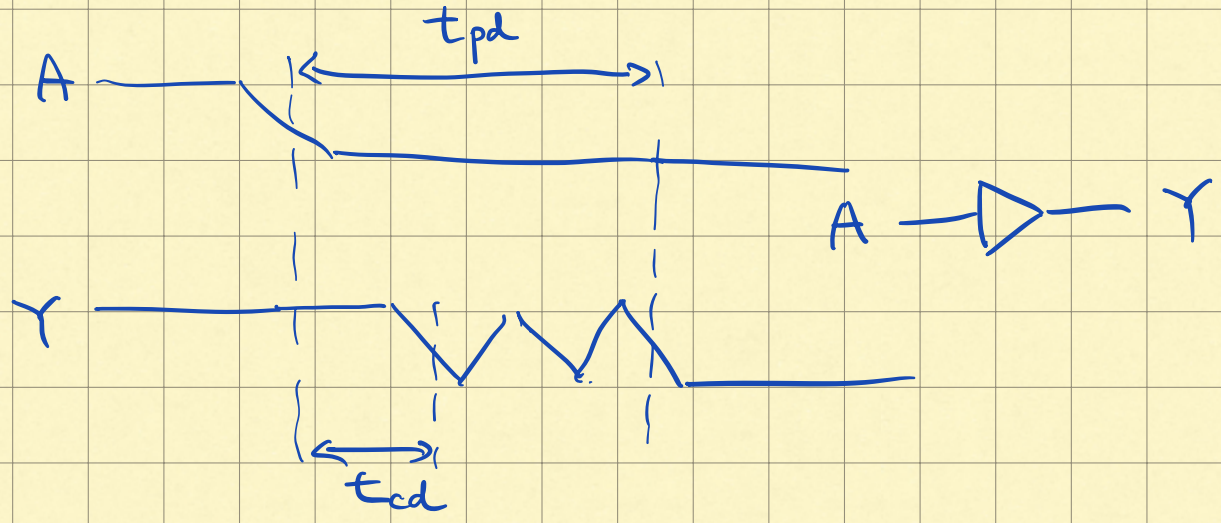
We measure delay from the 50% point of A to the 50% point of Y.

2.9.1 Propagation and Contamination Delay.

- Characteristics of combinational logic.

① The **propagation delay** t_{pd} is the maximum time from when any input changes until the output or outputs reach their final value.

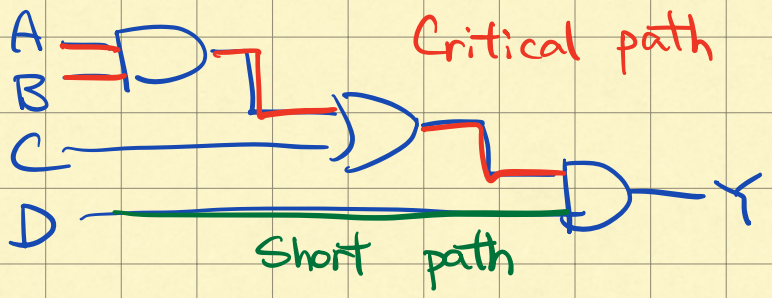
② The **contamination delay** t_{cd} is the minimum time from when any input changes until any output starts to change its value.



The underlying causes of delay:

- diff. rising and falling edges.
- multiple inputs and outputs. (diff. speed)
- temperature.

Also, determined by the path.



Critical path: longest, slowest path.
It limits the speed.

Shortest path: shortest, fastest

t_{pd} of comb. circuit is
the sum of the t_{pd} through each element on the critical path.

t_{cd} of comb. circuit is the sum of the t_{cd} through each element on the shortest path.

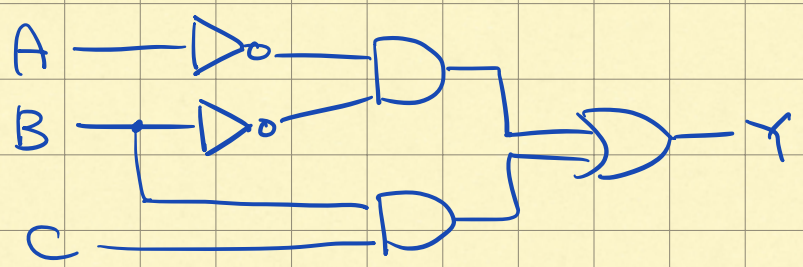
For above circuit,

$$t_{pd} = 2 \cdot t_{pd\_AND} + t_{pd\_OR}$$

$$t_{cd} = t_{pd\_AND}.$$

2.9.2 Glitches

A single input transition can cause multiple output transitions.

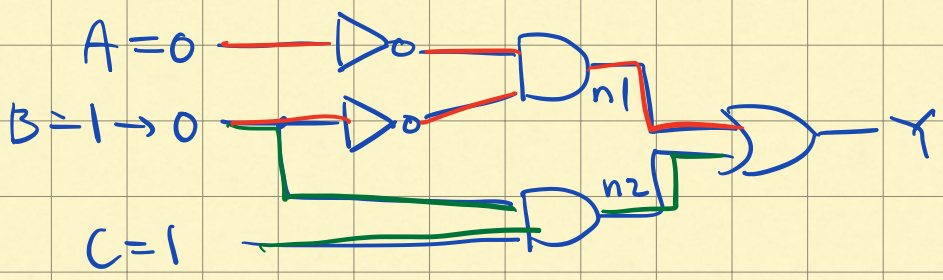


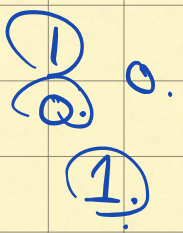
Y

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

$$Y = \overline{AB} + BC.$$

If A=0, C=1, and B=1 → 0.

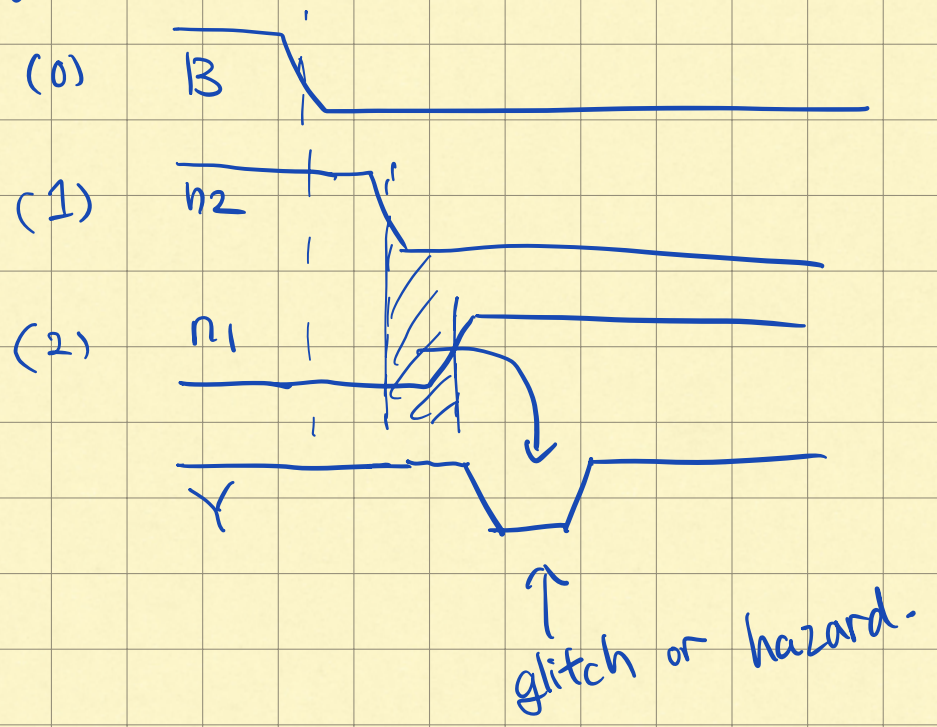




Ideally, $A=0, B=1, C=1 \Rightarrow Y=1$.

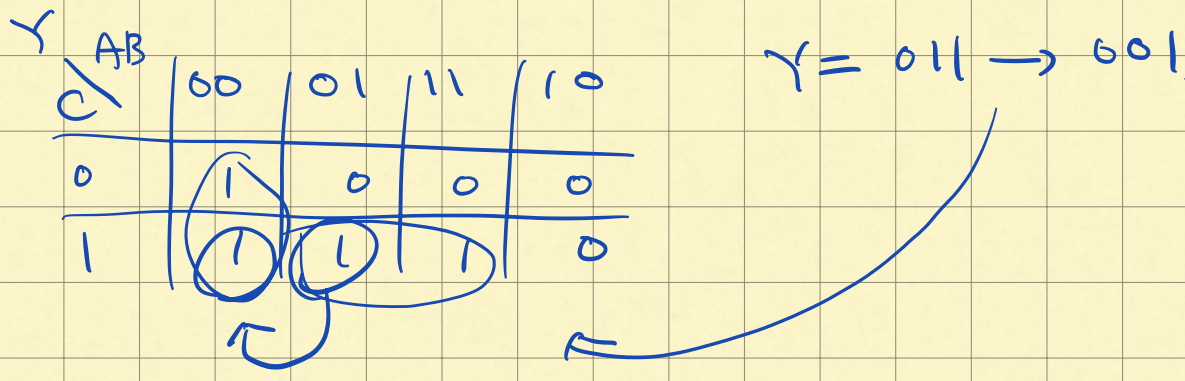
$A=0, B=0, C=1 \Rightarrow Y=1$

gates.



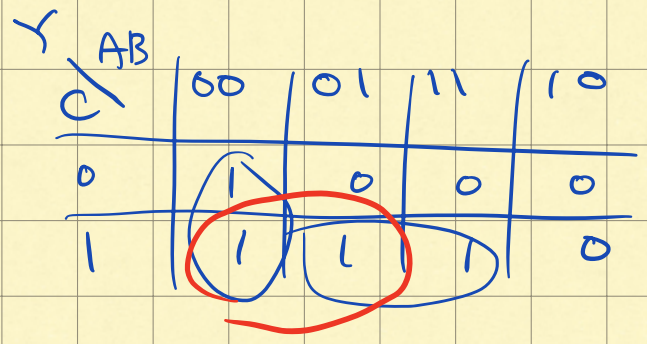
If we wait, it's not a problem.

From K-map perspective,

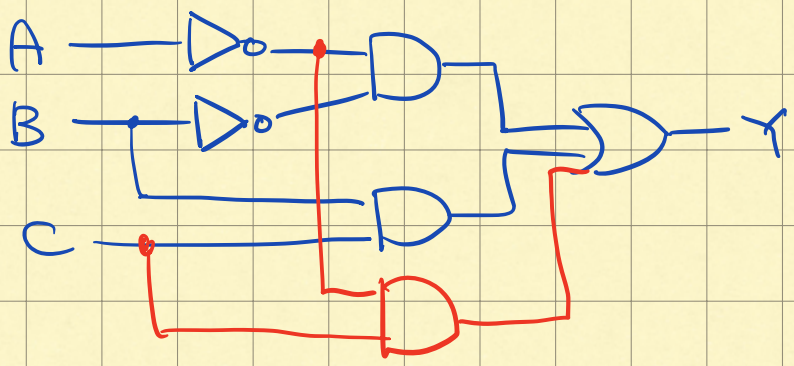


Transition from a prime implicant to another can cause a glitch.

To fix this, if we want,



add consensus (or redundant) term, $\bar{A}C$.



Simultaneous transitions on multiple inputs can also cause glitches.
Majority of circuits have them.
We don't have to eliminate them but be aware of them.