COMBINATIONAL LOGIC DESIGN

Lesson 7

Digital Logic

Junying Chen

ELSEVIER

COMBINATIONAL LOGIC DESIGN

Karnaugh Maps (K-Maps)

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

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 (C \ AB)	00	01	11	10
0	**1**	0	0	0
1	**1**	0	0	0

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

ELSEVIER

K-Map

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

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

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

3-Input K-Map

Y: C \ AB	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	$A\bar{B}C$

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

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

3-Input K-Map

Y: C \ AB	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	$A\bar{B}C$

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**

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

K-Map

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

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

K-Map Definitions

- **Complement:** variable with a bar over it

  $\bar{A}, \bar{B}, \bar{C}$

- **Literal:** variable or its complement

  $\bar{A}, A, \bar{B}, B, C, \bar{C}$

- **Implicant:** product of literals

  $A\bar{B}C, \bar{A}C, BC$

- **Prime implicant:** implicant corresponding to the largest circle in a K-map

K-Map Circling 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

ELSEVIER

4-Input K-Map

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

CD \ *AB*	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

4-Input K-Map

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

CD \ *AB*	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

4-Input K-Map

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

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

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

$B\bar{C}\bar{D}$

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

K-Maps 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

Y

CD \ *AB*	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

COMBINATIONAL LOGIC DESIGN

K-Maps 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

Y

CD \ AB	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

K-Maps 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

$AB\bar{C}D$

$AB\bar{C}\bar{D}$

Y

CD \ AB	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

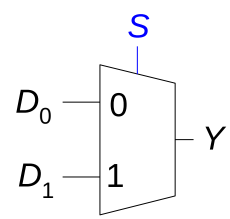
$$Y = A + \bar{B}\bar{D} + C$$

Combinational Building Blocks

- Multiplexers
- Decoders

Multiplexer (Mux)

- Selects between one of N inputs to connect to output
- $\log_2 N$-bit select input – control input
- **Example:** **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

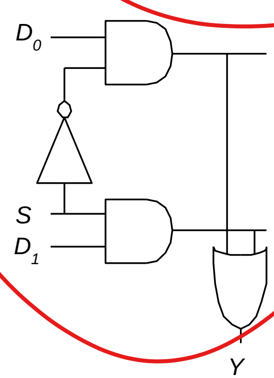
S	Y
0	D_0
1	D_1

Multiplexer Implementations

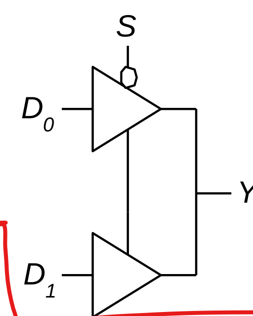
- **Logic gates**
  - Sum-of-products form

S \ D_0D_1	00	01	11	10
0	0	0	1	1
1	0	1	1	0

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



- **Tristates**
  - For an N-input mux, use N tristates
  - Turn on exactly one to select the appropriate input

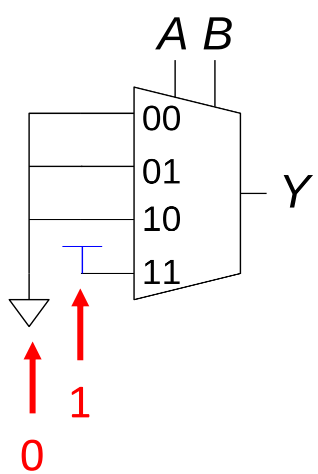


Logic using Multiplexers

- Using the mux as a lookup table

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

$Y = AB$



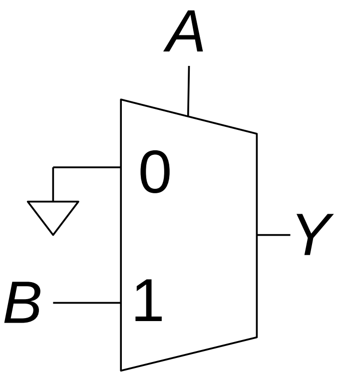
Logic using Multiplexers

- Reducing the size of the mux

$Y = AB$

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

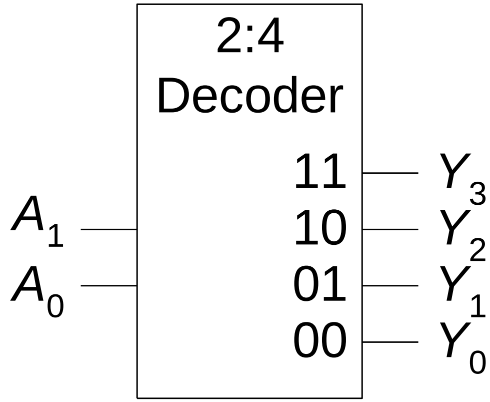
A	Y
0	0
1	B



COMBINATIONAL LOGIC DESIGN

Decoders

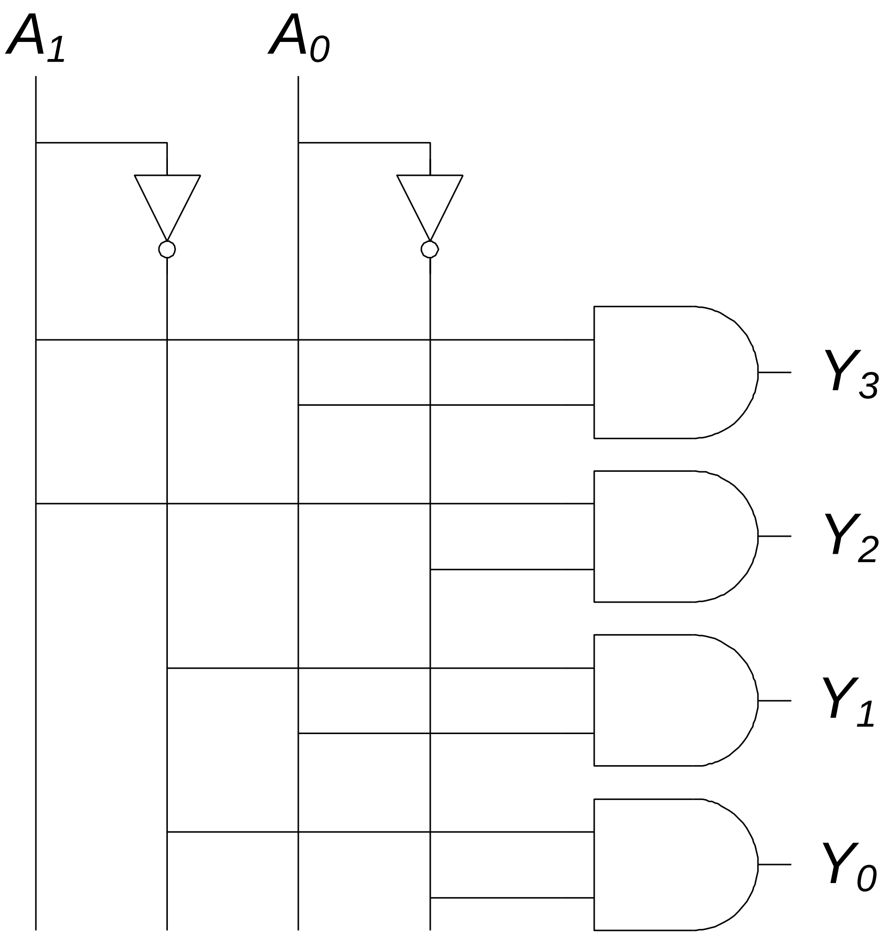
- N inputs, 2^N outputs
- One-hot outputs: only one output HIGH at once



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

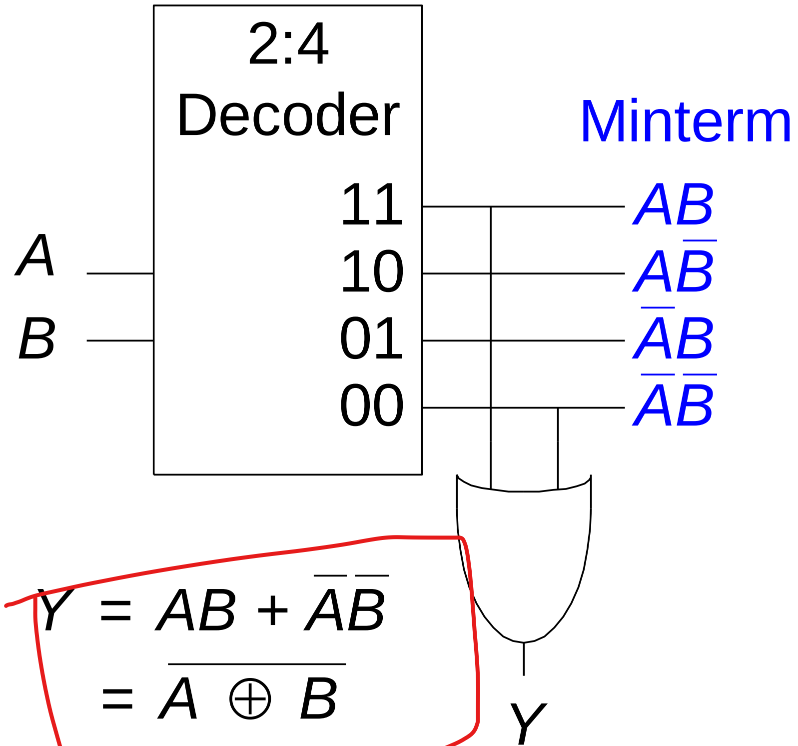
ELSEVIER

Decoder Implementation



Logic Using Decoders

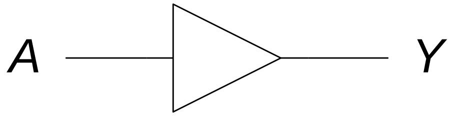
- OR minterms

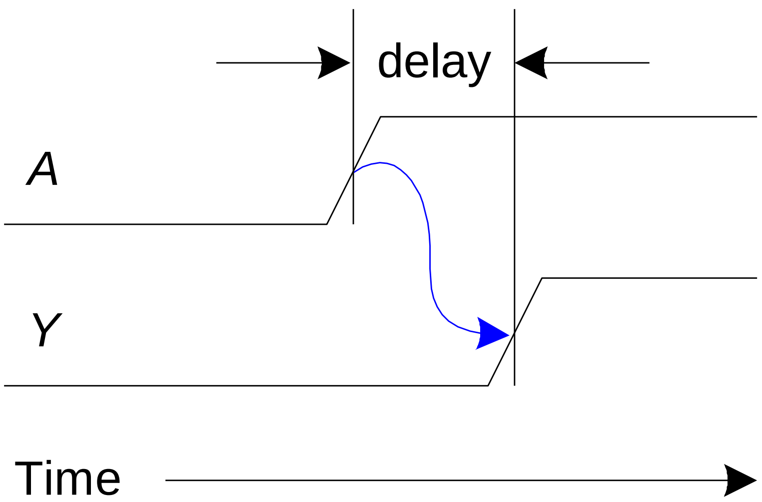


$$Y = AB + \overline{A}\,\overline{B}$$
$$= \overline{A \oplus B}$$

Timing

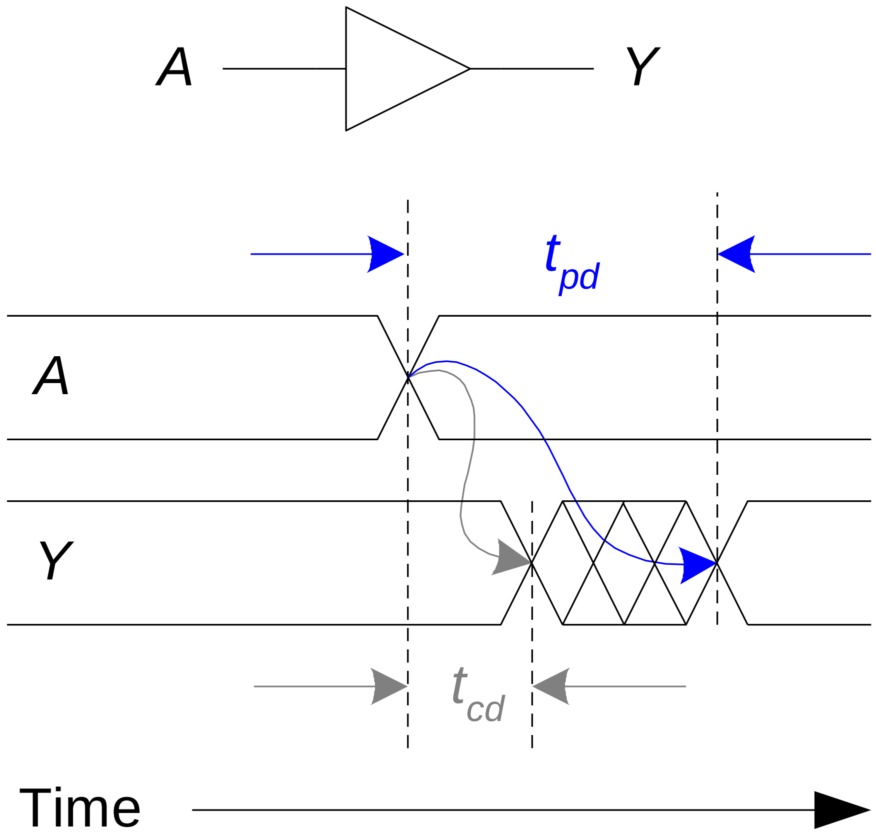
- Delay between input change and output change
- How to build fast circuits?





Propagation & Contamination Delay

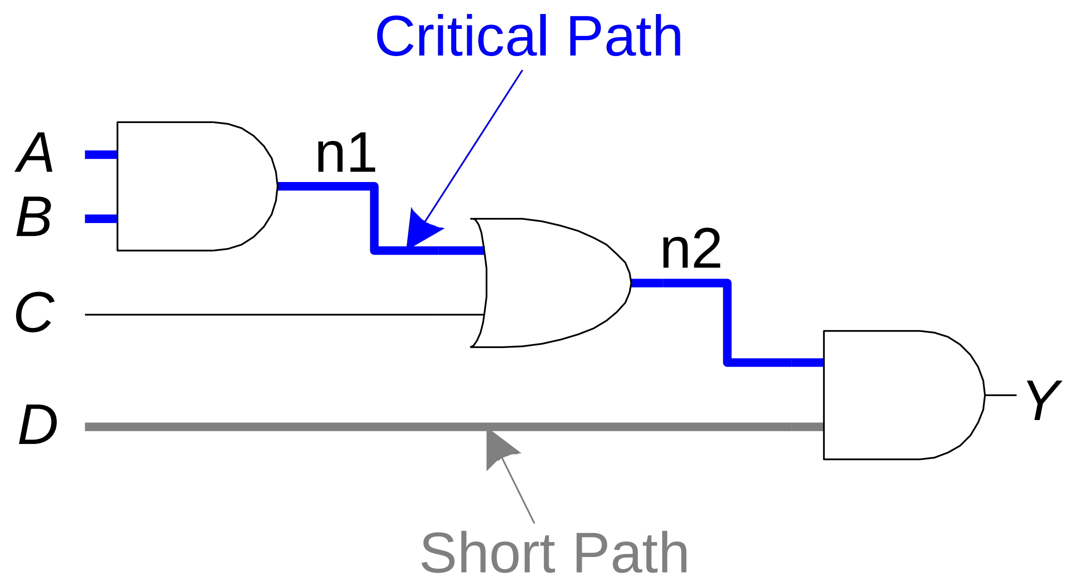
- **Propagation delay:** t_{pd} = max delay from input to output
- **Contamination delay:** t_{cd} = min delay from input to output



Propagation & Contamination Delay

- Delay is caused by
  - Capacitance and resistance in a circuit
  - Speed of light limitation
- Reasons why t_{pd} and t_{cd} may be different:
  - Different rising and falling delays
  - Multiple inputs and outputs, some of which are faster than others
  - Circuits slow down when hot and speed up when cold

Critical (Long) & Short Paths



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

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

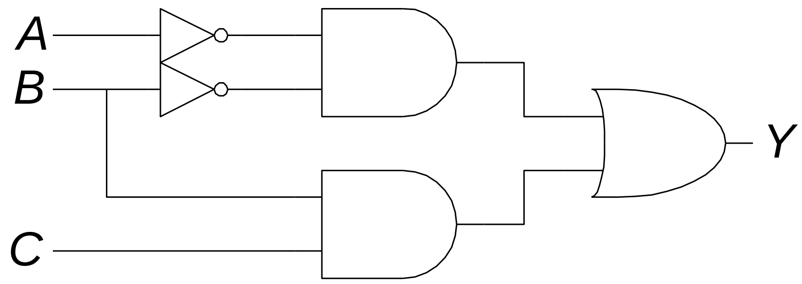
After-class reading: Example 2.15 and 2.16 in Textbook

Glitches

- When a single input change causes an output to change multiple times

Glitch Example

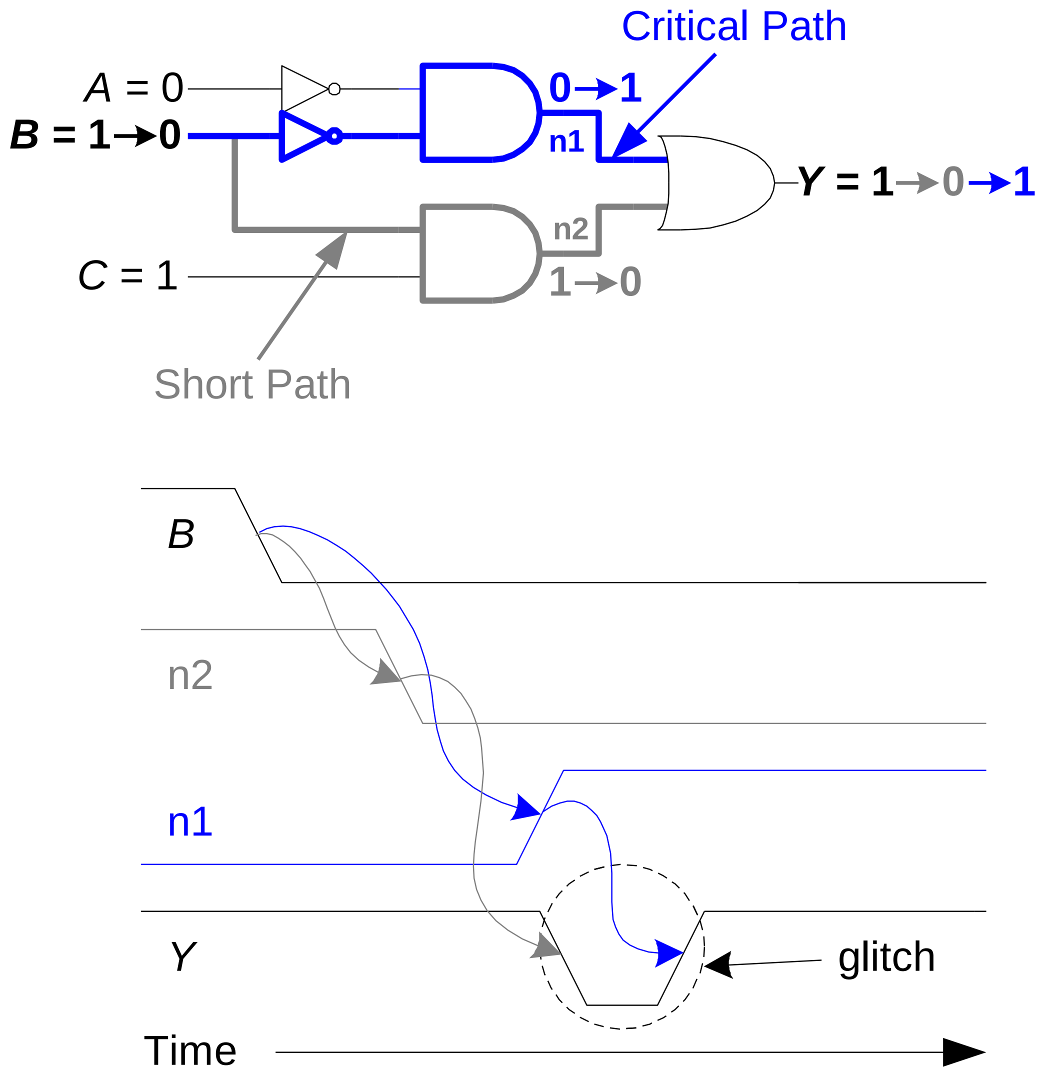
- What happens when A = 0, C = 1, B falls?



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

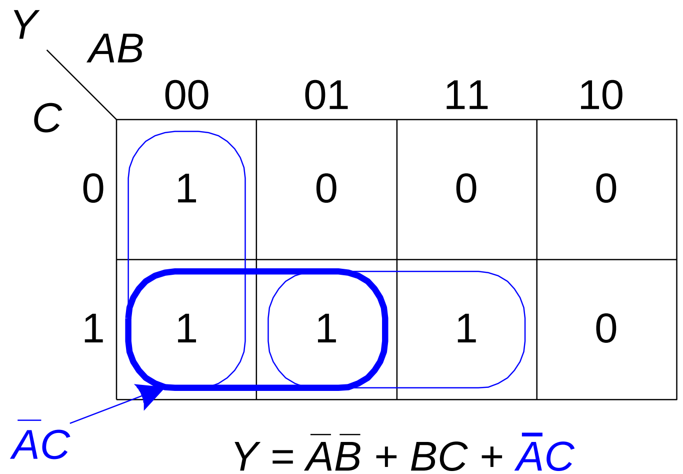
$$Y = \bar{A}\bar{B} + BC$$

Glitch Example (cont.)

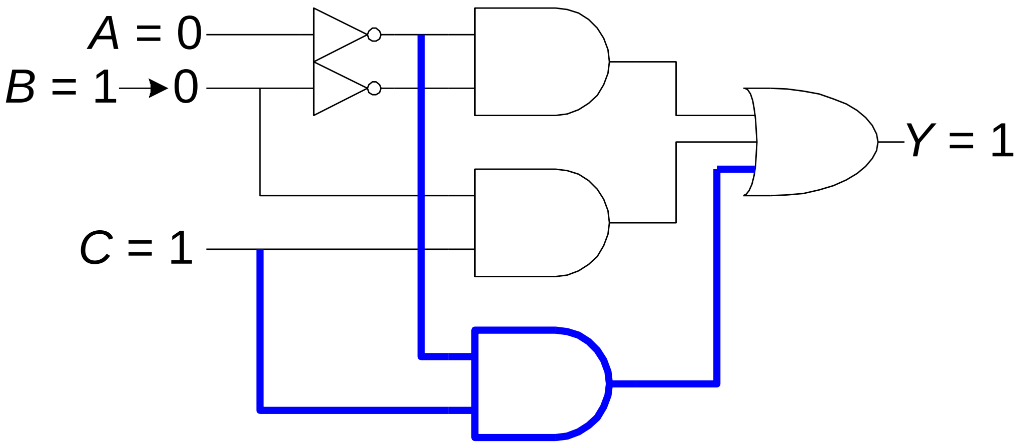


ELSEVIER

Fixing the Glitch



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



Why Understand Glitches?

- Glitches don't cause problems because of **synchronous design** conventions (see Chapter 3)
- It's important to **recognize** a glitch: in simulations or on oscilloscope
- Can't get rid of all glitches – simultaneous transitions on multiple inputs can also cause glitches