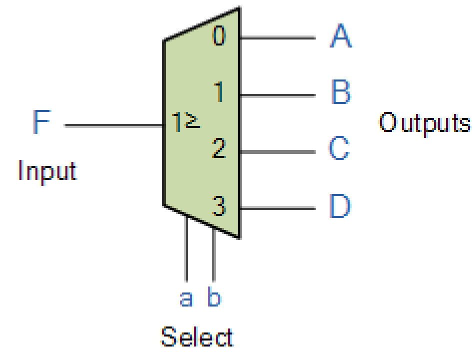
CET 141: Day 11

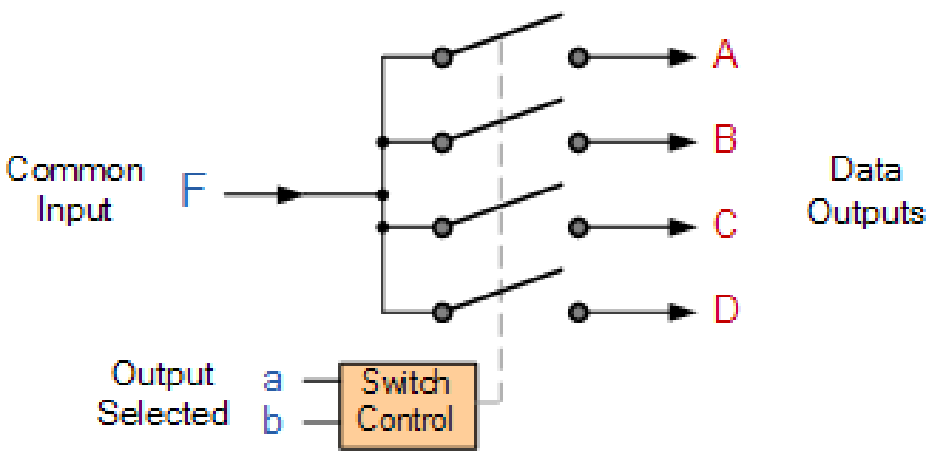
Dr. Noori KIM

Demultiplexer

- A circuit that has one input and more than one output
- Converting a serial data signal at the input to a parallel data at its output lines
- Used when a circuit wishes to send a signal to one of many devices
- A comparison with a decoder
  - Decoder: to select among many devices
  - Demultiplexer: to send a signal among many devices

- A symbol: where F is an input and a, b are selections
- Example: 1-to-4 Channel De-multiplexer





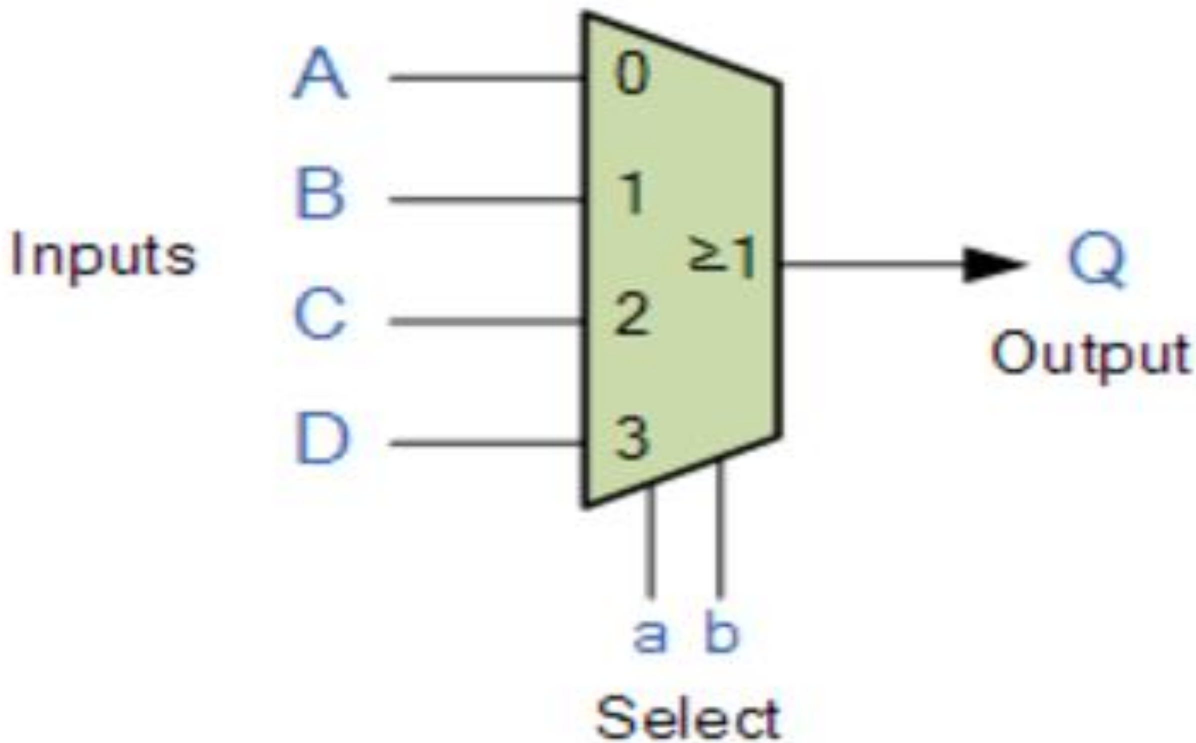
Output Select		Data Output Selected
a	b	
0	0	A
0	1	B
1	0	C
1	1	D

The Boolean expression for F $F = \bar{a}\bar{b}A + \bar{a}bB + a\bar{b}C + abD$

Multiplexer implementations

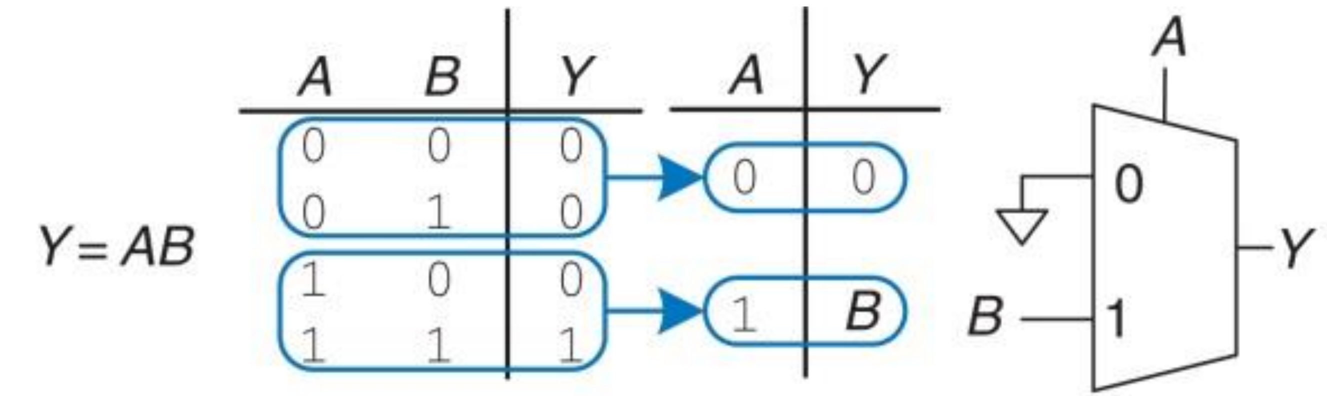
- 4:1 multiplexer implementations:

(a) two-level logic, (b) tristates, (c) hierarchical

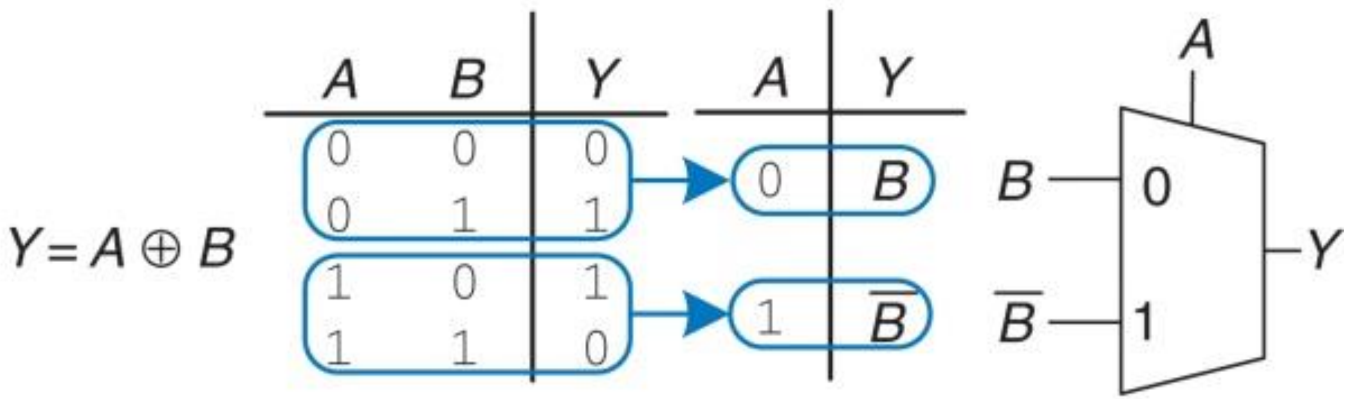


They are all multiplexers

Multiplexer logic using variable inputs



(a)



(b)

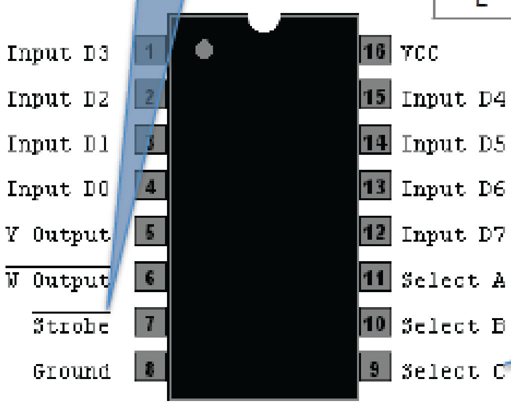
Multiplexers – IC chip

- Selects one individual data input line and then sends that data to a single output line
- We can build the device using logic gates or
- Using 74151 logic gate IC (8 to 1 multiplexer)
  - 8 (2^3) inputs (0_{10}-7_{10}), and 1 output
  - The inputs are controlled by 3 control signal.

FUNCTION TABLE

INPUTS												OUTPUTS	
$\overline{E}$	S_2	S_1	S_0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	$\overline{Y}$	Y
H	X	X	X	X	X	X	X	X	X	X	X	H	L
L	L	L	L	L	X	X	X	X	X	X	X	H	L
L	L	L	L	H	X	X	X	X	X	X	X	L	H
L	L	L	H	X	L	X	X	X	X	X	X	H	L
L	L	L	H	X	H	X	X	X	X	X	X	L	H
L	L	H	L	X	X	L	X	X	X	X	X	H	L
L	L	H	L	X	X	H	X	X	X	X	X	L	H
L	L	H	H	X	X	X	L	X	X	X	X	H	L
L	L	H	H	X	X	X	H	X	X	X	X	L	H
L	H	L	L	X	X	X	X	L	X	X	X	H	L
L	H	L	L	X	X	X	X	H	X	X	X	L	H
L	H	L	H	X	X	X	X	X	L	X	X	H	L
L	H	L	H	X	X	X	X	X	H	X	X	L	H
L	H	H	L	X	X	X	X	X	X	L	X	H	L
L	H	H	L	X	X	X	X	X	X	H	X	L	H
L	H	H	H	X	X	X	X	X	X	X	L	H	L
L	H	H	H	X	X	X	X	X	X	X	H	L	H

Strobe: the enable signal, low active.



Pin 9 is the MSB of the selection signals.

Notes on multiplexers

- Selects one individual data input line and then sends that data to a single output line
- 74151 logic gate IC (8 to 1 multiplexer)
  - 8 (2^3) inputs (0_{10}-7_{10}), and 1 output
  - The inputs are controlled by 3 control signals.
- 74153 logic gate IC (dual 4 to 1 multiplexer)
  - 4 (2^2) inputs (0_{10}-3_{10}), and 1 output
  - The inputs are controlled by 2 control signals

$$F(A, I_2, I_1, I_0) = \sum m(2,4,5,7,10,14)$$

; Min terms
; for SOP
; High outputs

$$F(A, I_2, I_1, I_0) = \prod M(0,1,3,6,8,9,11,12,13,15)$$

; Max terms
; for POS
; Low outputs

$$F(A, I_2, I_1, I_0) = \sum m(2,4,5,7,10,14)$$

$F(A, I_2, I_1, I_0) = \prod M(0,1,3,6,8,9,11,12,13,15)$; if there is no don't care terms

A (MSB)	I_2	I_1	I_0	Decimal	Y
0	0	0	0		
0	0	0	1		
0	0	1	0	2	1
0	0	1	1		
0	1	0	0	4	1
0	1	0	1	5	1
0	1	1	0		
0	1	1	1	7	1
1	0	0	0		
1	0	0	1		
1	0	1	0	10	1
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0	14	1
1	1	1	1		

$$F(A, I_2, I_1, I_0) = \sum m(2,4,5,7,10,14)$$

Let’s implement this SOP using a 8*1 multiplexer!

	D0	D1	D2	D3	D4	D5	D6	D7	
A’									Row 1
A									Row 2
I/P to MUX									

$$F(A, I_2, I_1, I_0) = \sum m(2,4,5,7,10,14)$$

← Sum of Inputs that make output (Y) high

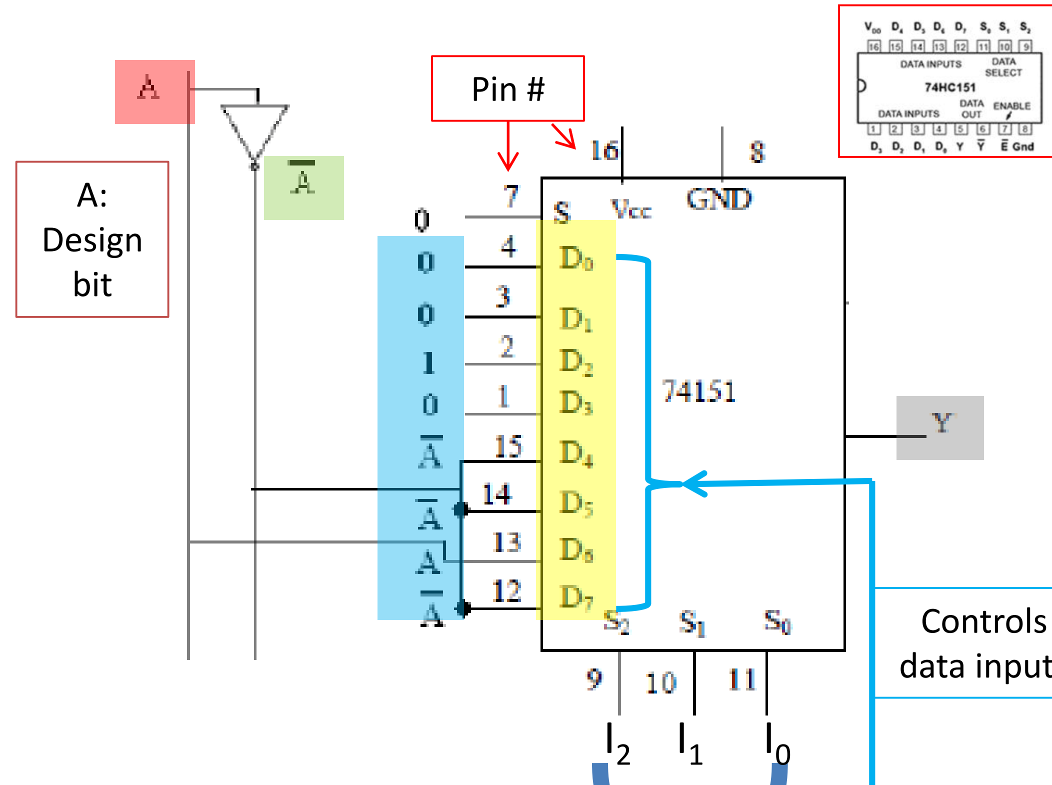
Index of data inputs are controlled by I2, I1,I0 (control bits)

$A'I_2I_1I_0$

$AI_2I_1I_0$

	D0	D1	D2	D3	D4	D5	D6	D7	
$\bar{A}$	0	1	2	3	4	5	6	7	Row 1
A	8	9	10	11	12	13	14	15	Row 2
I/P To MUX	0	0	1	0	$\bar{A}$	$\bar{A}$	A	$\bar{A}$	

A (MSB)	I_2	I_1	I_0
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



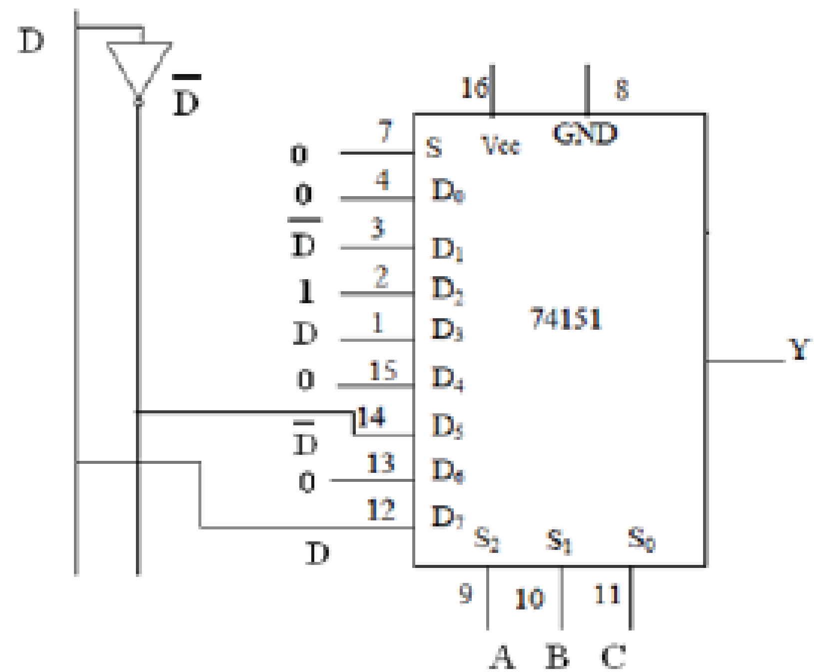
Controls data inputs

Control bits

F (A, B, C, D) = ∑m (2, 4, 5, 7, 10, 14)

Design Using MSB Bit D:

	$\overline{D}$	D	
D0	0	1	0
D1	2	3	$\overline{D}$
D2	4	5	1
D3	6	7	D
D4	8	9	0
D5	10	11	$\overline{D}$
D6	12	13	0
D7	14	15	$\overline{D}$



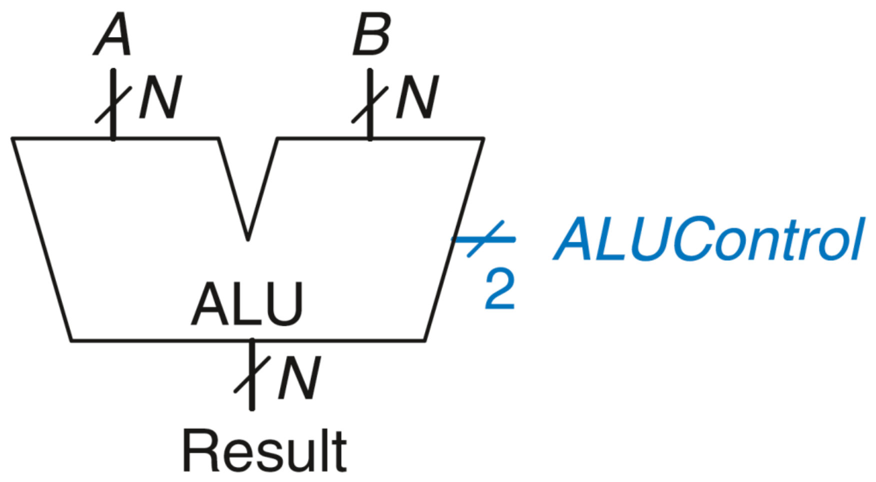
ALU
(as an application of multiplexer use)

Arithmetic Logic Unit (ALU): A subunit within a computer's CENTRAL PROCESSING UNIT that performs mathematical operations

Implement math operation circuits and put them together
https://www.allaboutcircuits.com/projects/how-to-build-your-own-discrete-4-bit-alu/

ALU: Arithmetic Logic Unit

$ALUControl_{1:0}$	Function
00	Add
01	Subtract
10	AND
11	OR



Example: Perform *A* + *B*

$$ALUControl = 00$$

$$Result = A + B$$

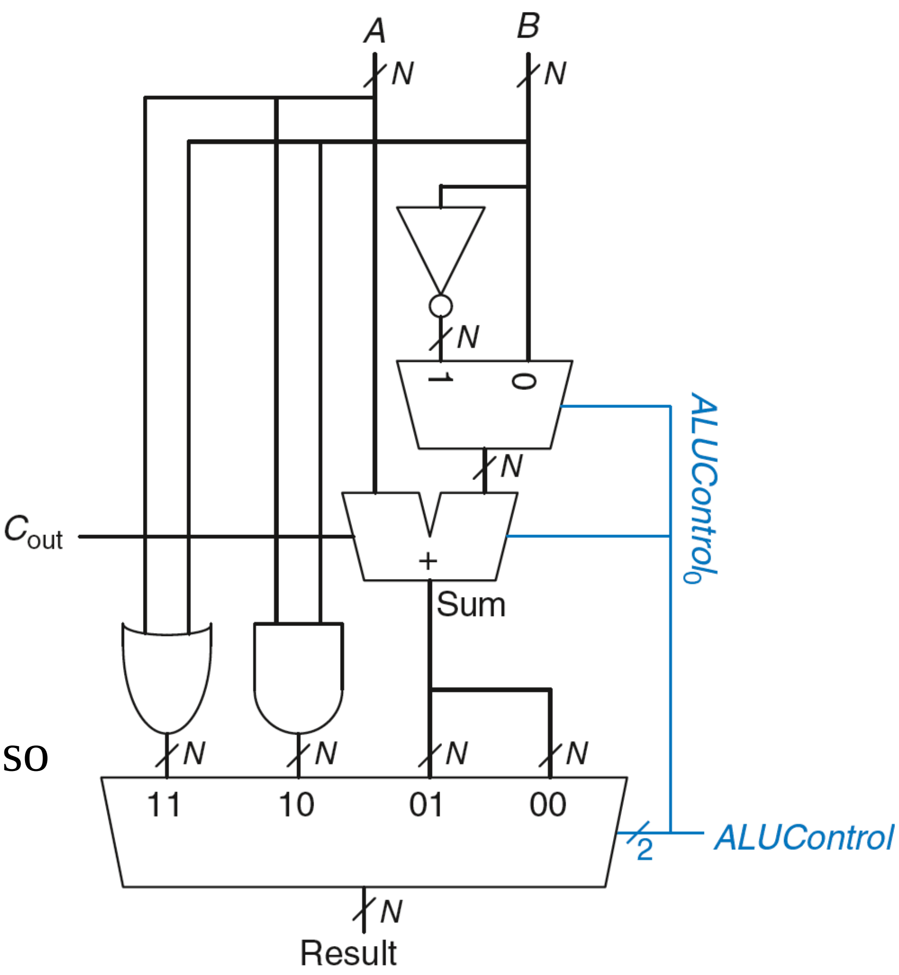
$ALUControl_{1:0}$	Function
00	Add
01	Subtract
10	AND
11	OR

Example: Perform *A* OR *B*

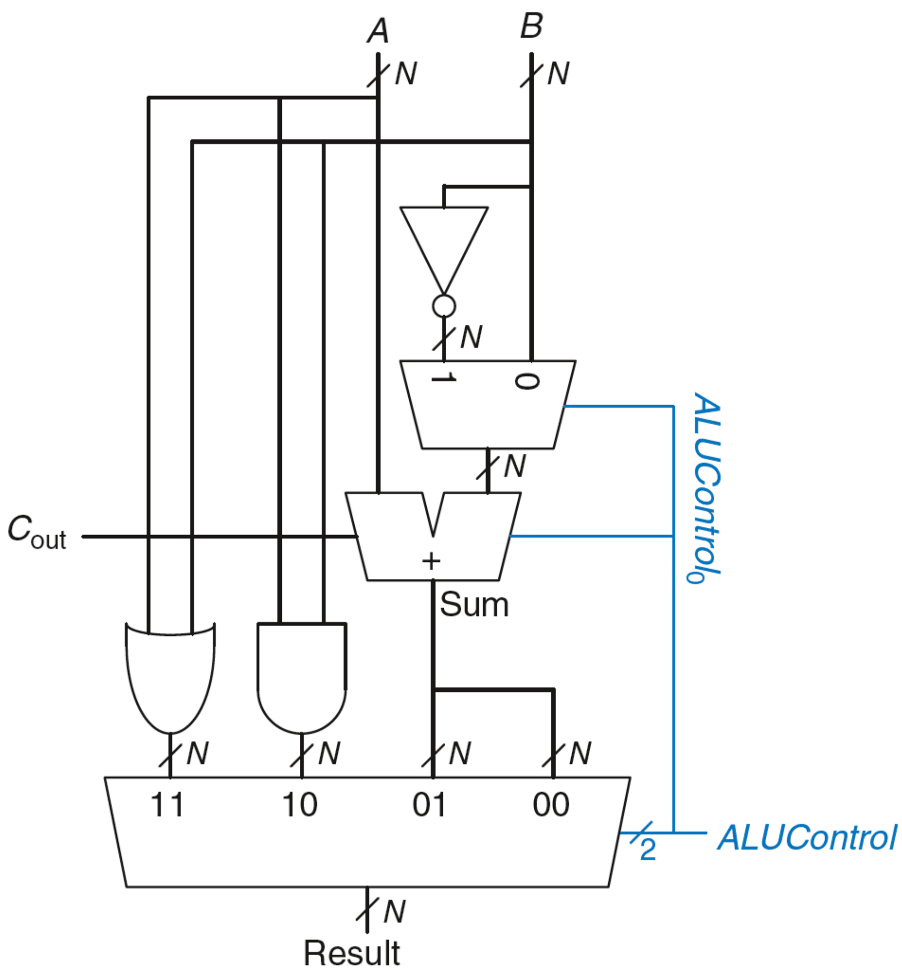
$ALUControl_{1:0} = 11$

Mux selects output of OR gate as *Result*, so

Result* = *A* OR *B



$ALUControl_{1:0}$	Function
00	Add
01	Subtract
10	AND
11	OR



Example: Perform *A* + *B*

$ALUControl_{1:0}$ = 00

$ALUControl_0$ = 0, so:

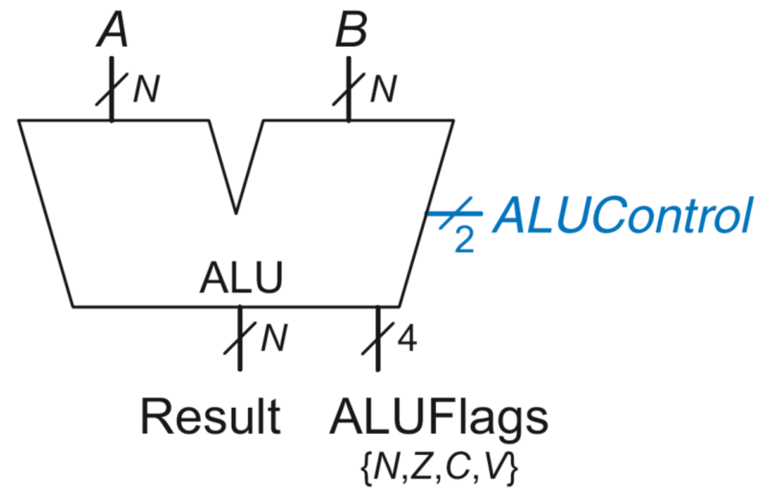
- Cin to adder = 0
- 2nd input to adder is *B*

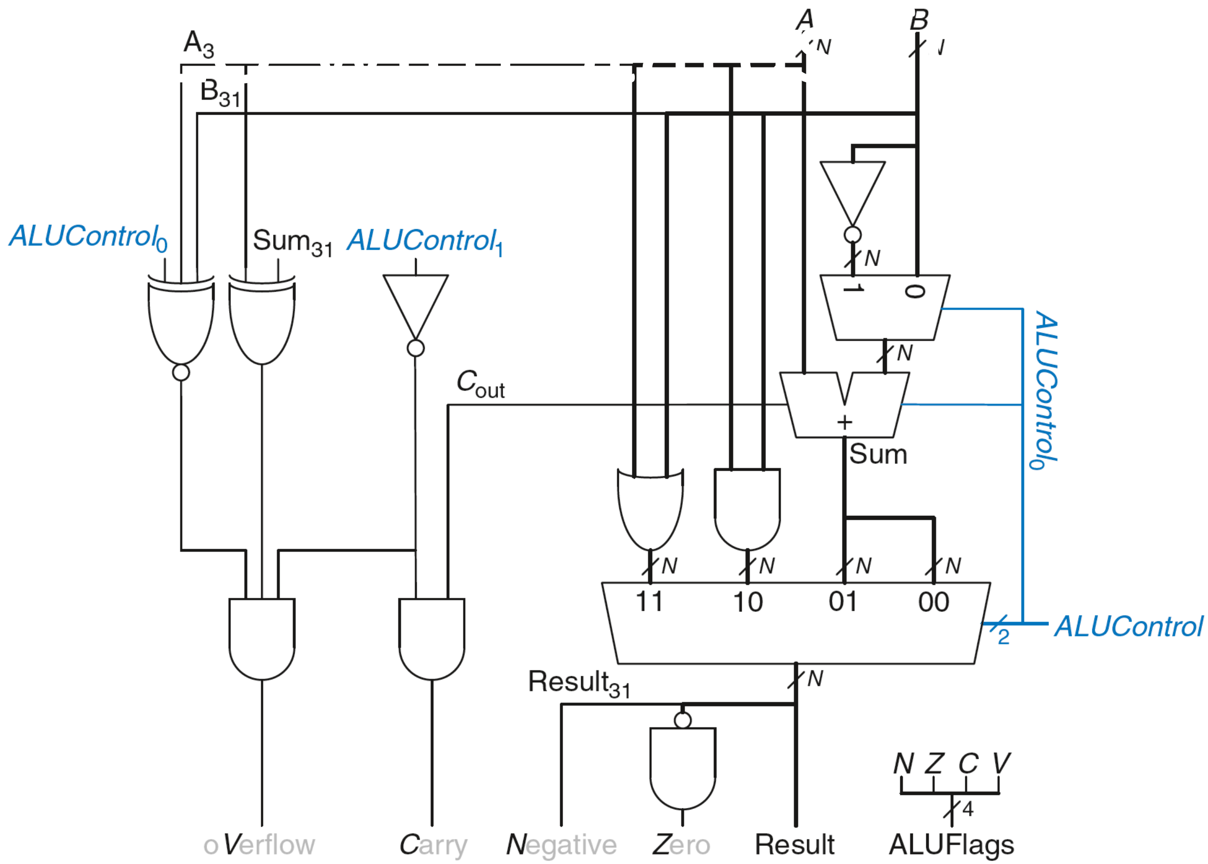
Mux selects *Sum* as *Result*, so

- ***Result* = *A* + *B***

ALU with Status Flags

Flag	Description
N	*Result* is **N**egative
Z	*Result* is **Z**ero
C	Adder produces **C**arry out
V	Adder o**V**erflowed





N = 1 if:

Result is **negative**

So, *N* is connected to most significant bit of *Result*

Z = 1 if:

all of the bits of *Result* are 0

C = 1 if:

C_{out} of Adder is 1

AND

ALU is adding or subtracting (ALUControl is 00 or 01)

V = 1 if:

The addition of 2 **same-signed numbers** produces a result with the **opposite sign**

Agenda

- Lecture: Timing in Comb. logics
  - Propagation delay
  - Contamination delay
  - Timing analysis

- Hazards in Combinational Logic may be caused due to **DELAY** in every logic gate chip
- Accumulated **Combinational Delays** from each logic gate (critical path vs. shortest path)→ (may cause) Glitch (which do not exist in an ideal case)→(may cause) circuit **Hazards**
- **Glitches** can be observed via
    - **(Pen&pencil) Logic gate delay analysis:** Timing diagram
    - **(hands-on) IC chip implementations:** Your oscilloscope

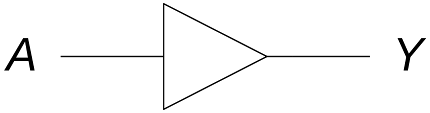
In another words…

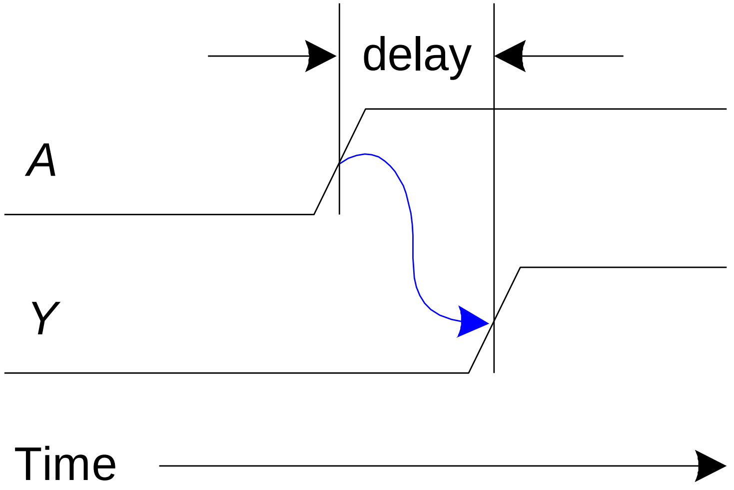
- When inputs of a combinational circuit changes
  - **unwanted switching variations** may appear in the output.
- These variations occur when
  - different paths from the input to output have **different delays** (accumulated delays combination)
- But!! The combination of delays *that produce a glitch* **may or may not occur** in the implementation of the circuit
- + not every glitch is hazardous.

Let’s talk about a little bit details..

Delay

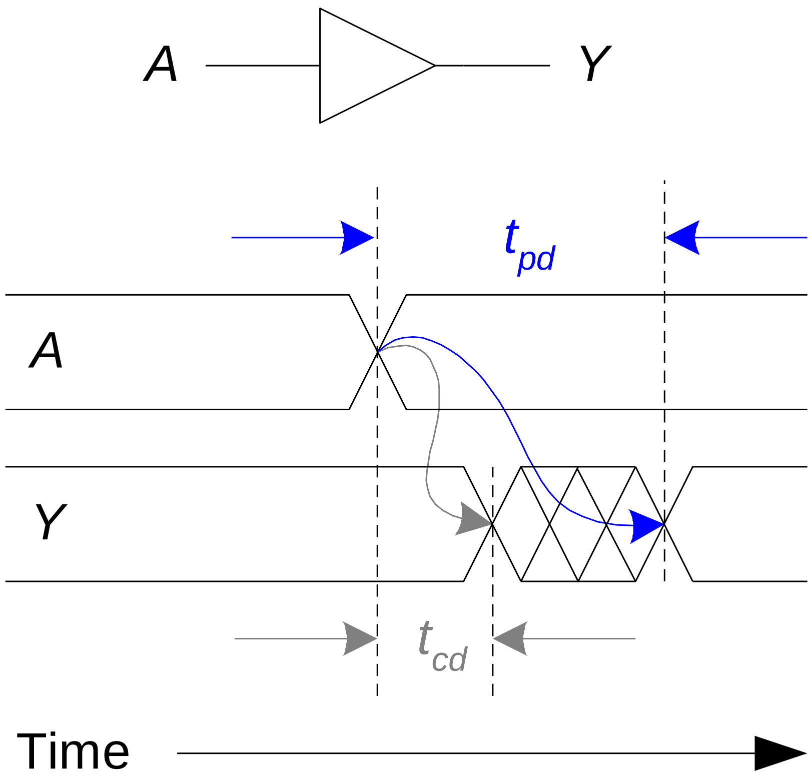
- **Delay:** time between input change and output changing
- How to build fast circuits?





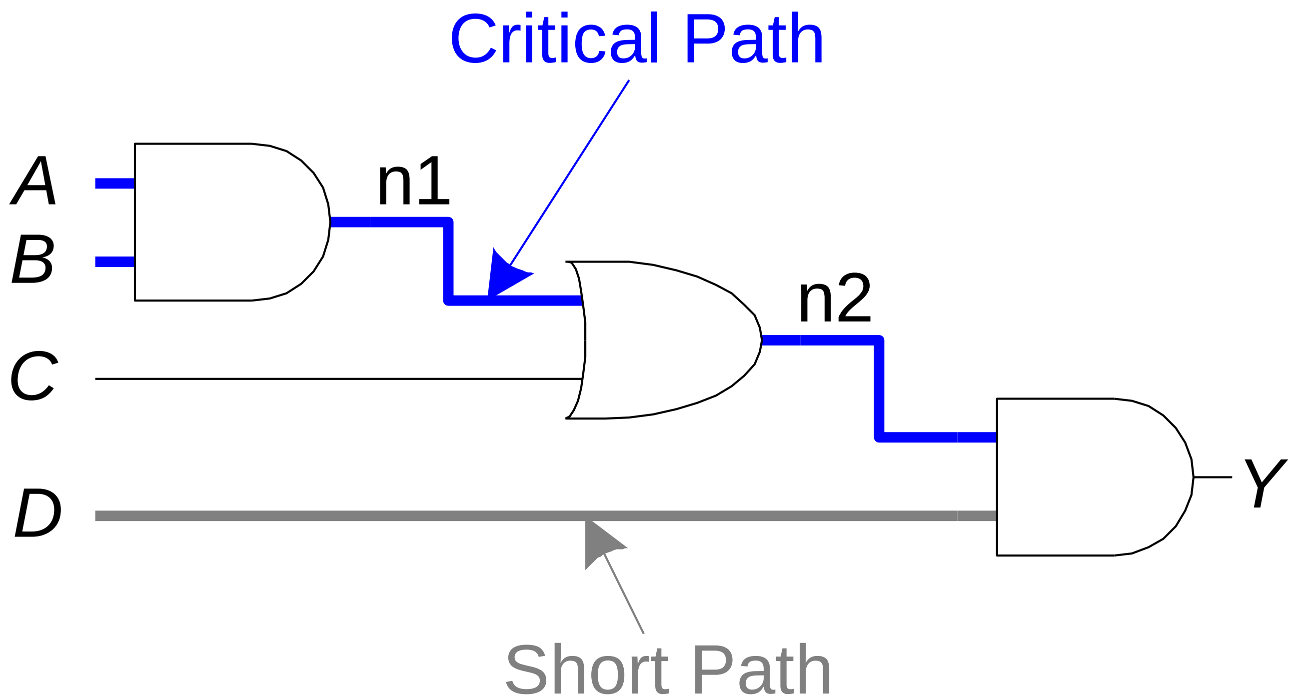
Propagation & Contamination Delay

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



- 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

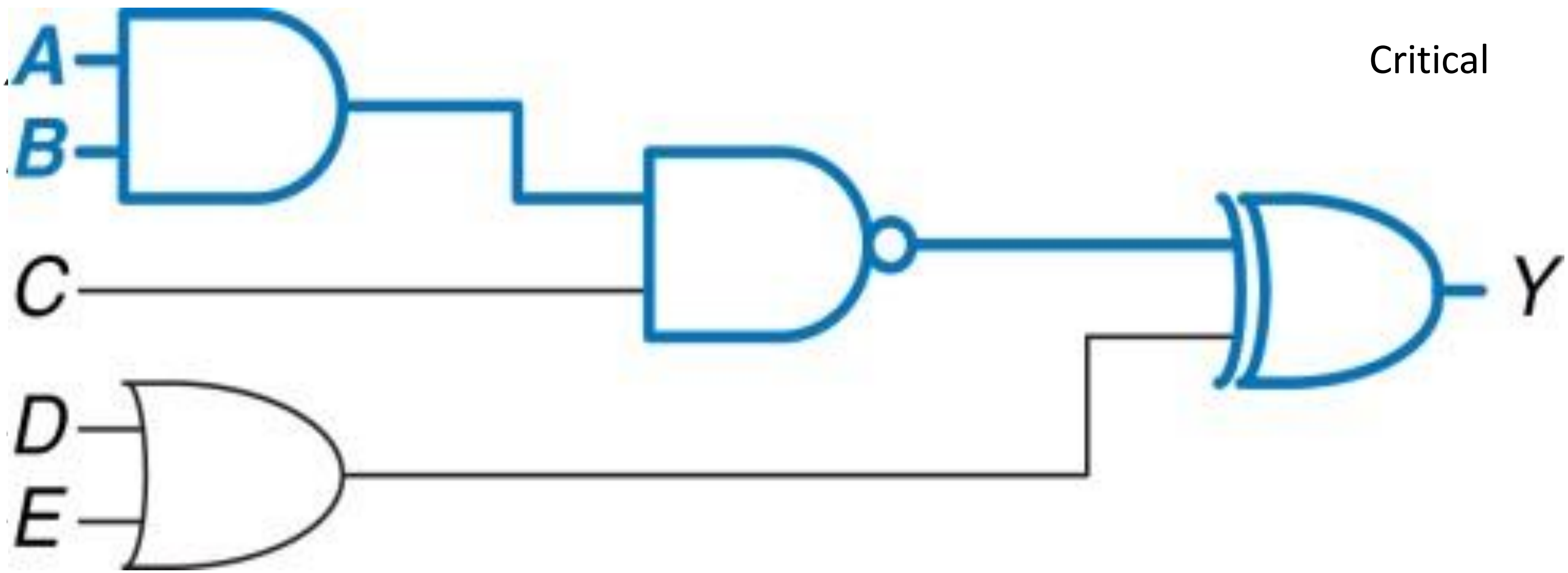


- **Critical (Long) Path:** $t_{pd} = 2t_{pd\_\mathrm{AND}} + t_{pd\_\mathrm{OR}}$

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

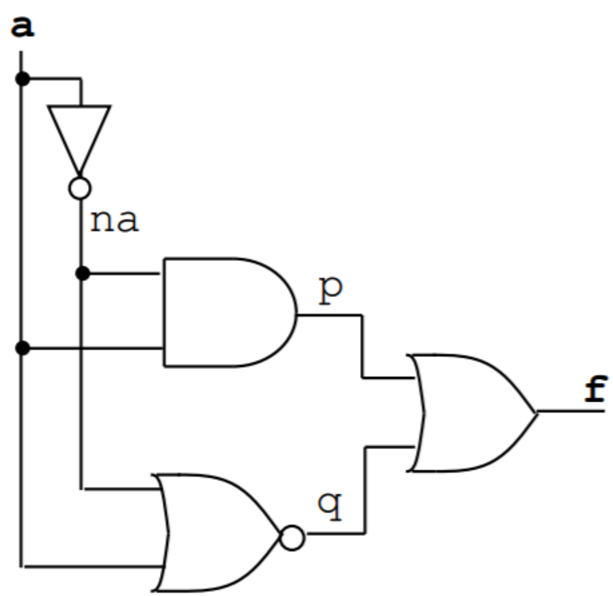
An Example

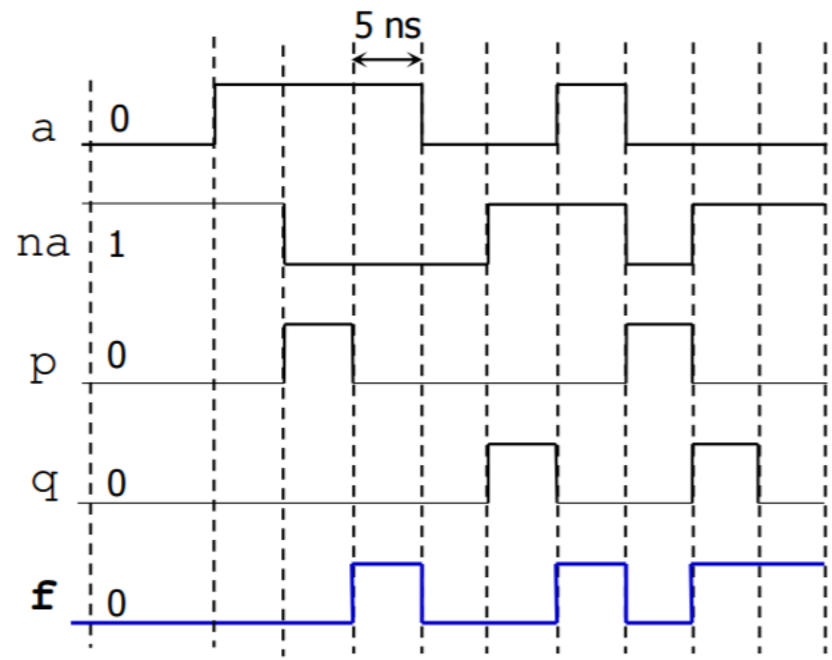
- Find a critical path and short path assuming that all gates have same delay



An example

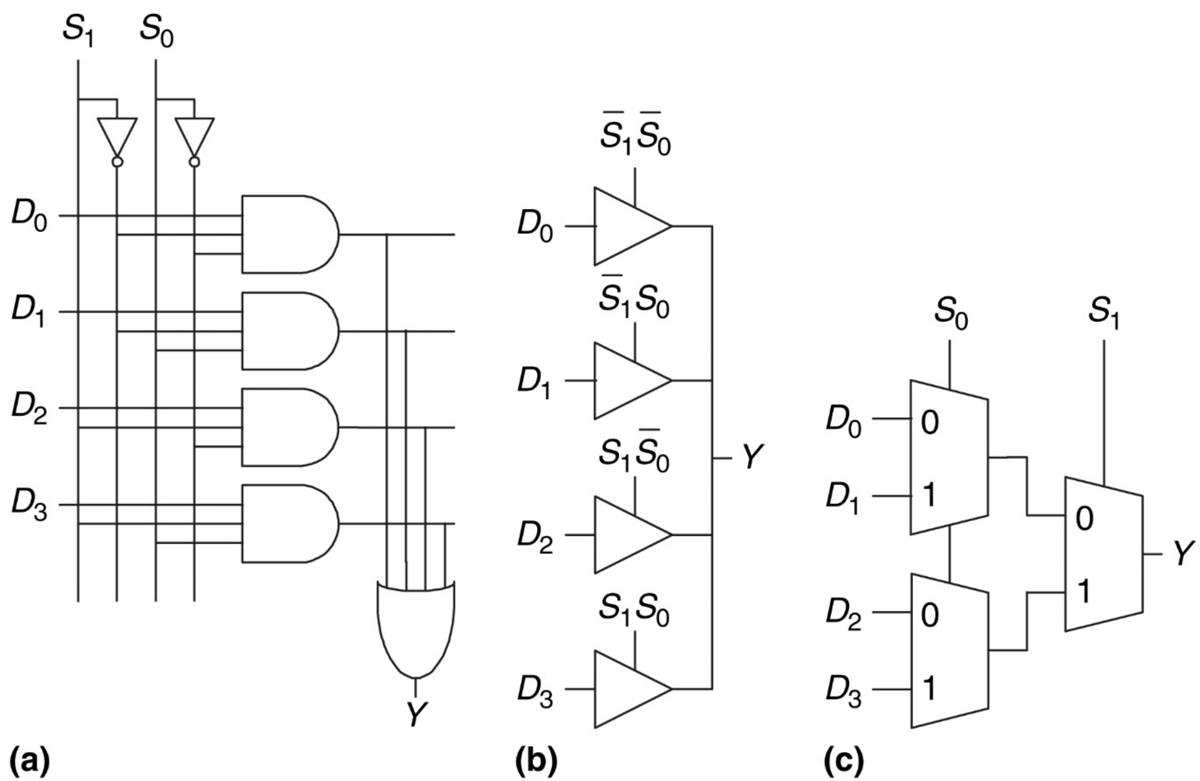
- Complete the timing diagram, assume that the propagation delay of <u>every gate is 5 ns</u>.





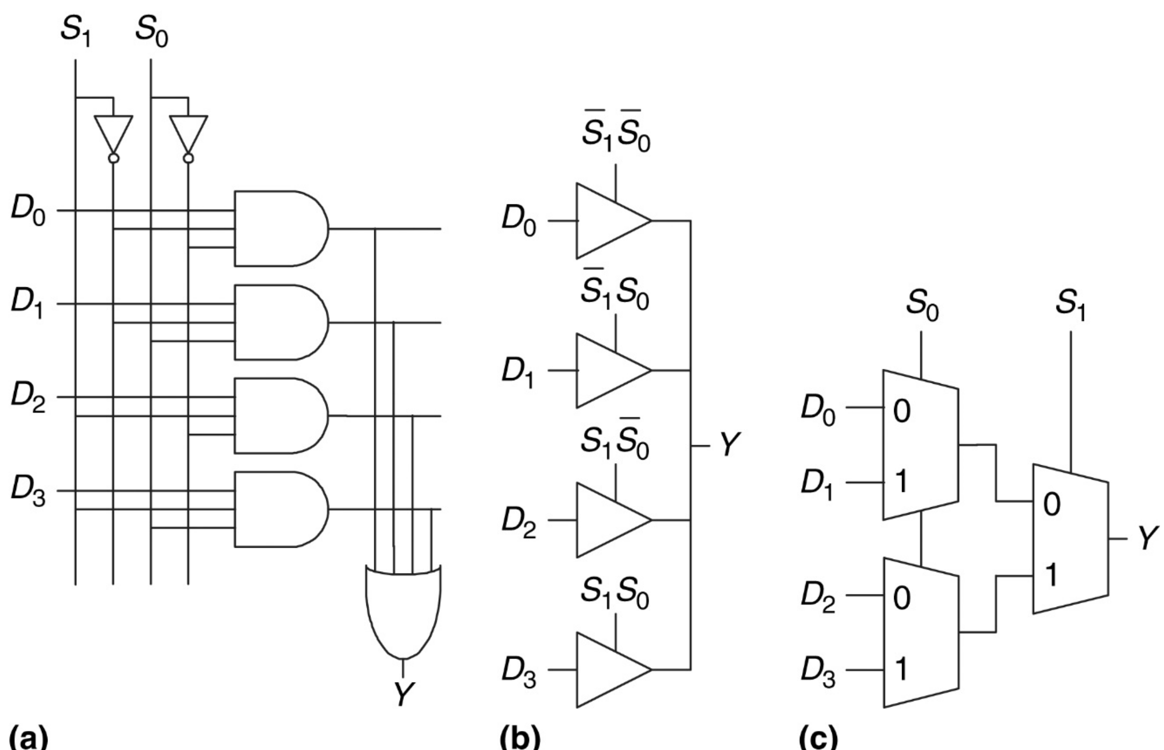
An example

- Recall our multiplexer implementation study.
- There are at least three different ways to make a multiplexer



- By following pd (propagation delay) for each logic gate, compute and compare pd for each implementation method (both for S and D).

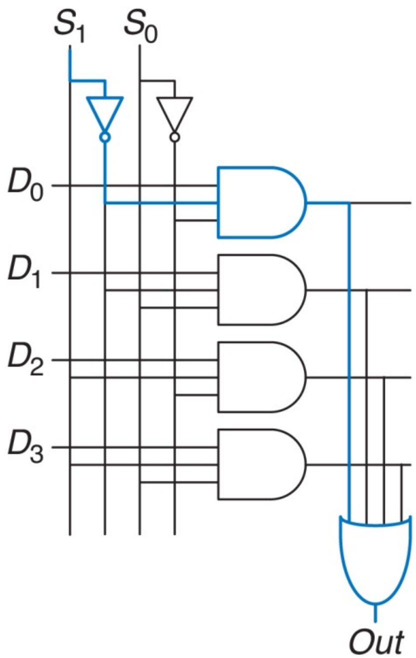
	t_{pd}[ps]
Inverter	30
3-input AND	80
2-input AND	60
4-input OR	90
Tri-state (A to Y)	50
Tri-state (enable to Y)	35

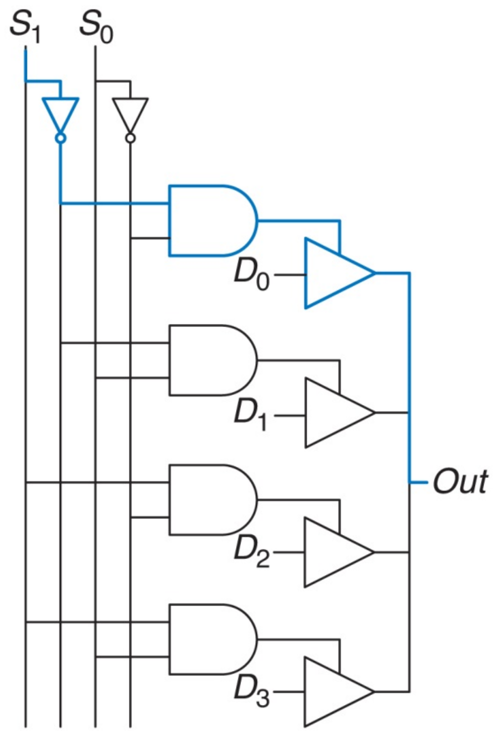


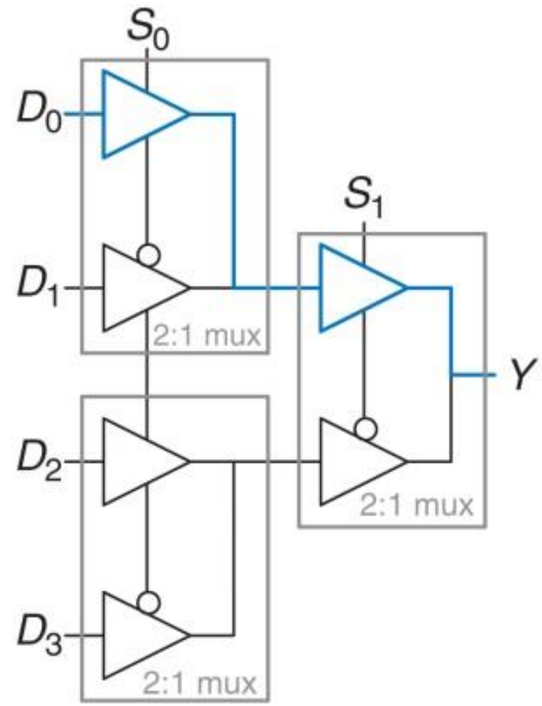
Two logic level

Tri-state

Hierarchical using 2:1 multiplexers







(a)

$$t_{pd\_sy} = t_{pd\_INV} + t_{pd\_AND3} + t_{pd\_OR4}$$
$$= 30\text{ ps} + 80\text{ ps} + 90\text{ ps}$$
$$= \mathbf{200\text{ ps}}$$
$$t_{pd\_dy} = t_{pd\_AND3} + t_{pd\_OR4}$$
$$= \mathbf{170\text{ ps}}$$

(b)

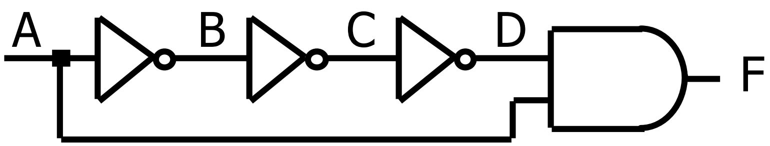
$$t_{pd\_sy} = t_{pd\_INV} + t_{pd\_AND2} + t_{pd\_TRI\_sy}$$
$$= 30\text{ ps} + 60\text{ ps} + 35\text{ ps}$$
$$= \mathbf{125\text{ ps}}$$
$$t_{pd\_dy} = t_{pd\_TRI\_ay}$$
$$= \mathbf{50\text{ ps}}$$

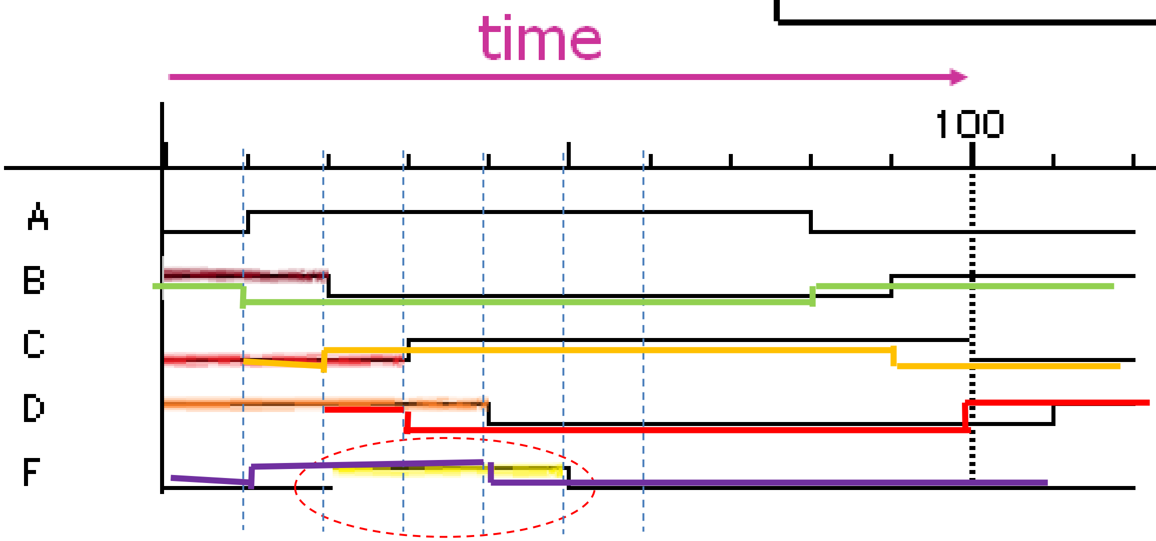
$$t_{pd\_s0y} = t_{pd\_TRI\_sy} + t_{pd\_TRI\_ay} = \mathbf{85\text{ ps}}$$
$$t_{pd\_dy} = 2\ t_{pd\_TRI\_ay} = \mathbf{100\text{ ps}}$$

Note that the best choice depends not only on the critical path through the circuit and the input arrival times, but also on the power, cost, and availability of parts.

Timing diagrams

- The idea starts from: **<u>Real gates</u>** have **<u>real delays</u>**
- Example: A''' • A = 0?





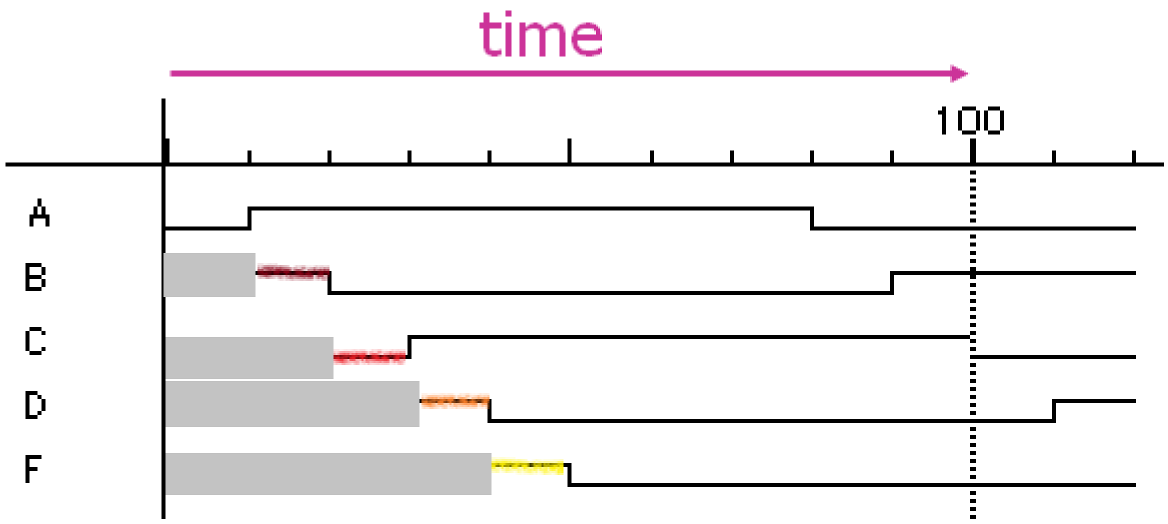
While A (input) changes 0→1, output F changes 0→1→0

- Delays cause Glitch (F supposes to be 0 but momentarily it stays as 1: a glitch): **"Static 0 hazard"**

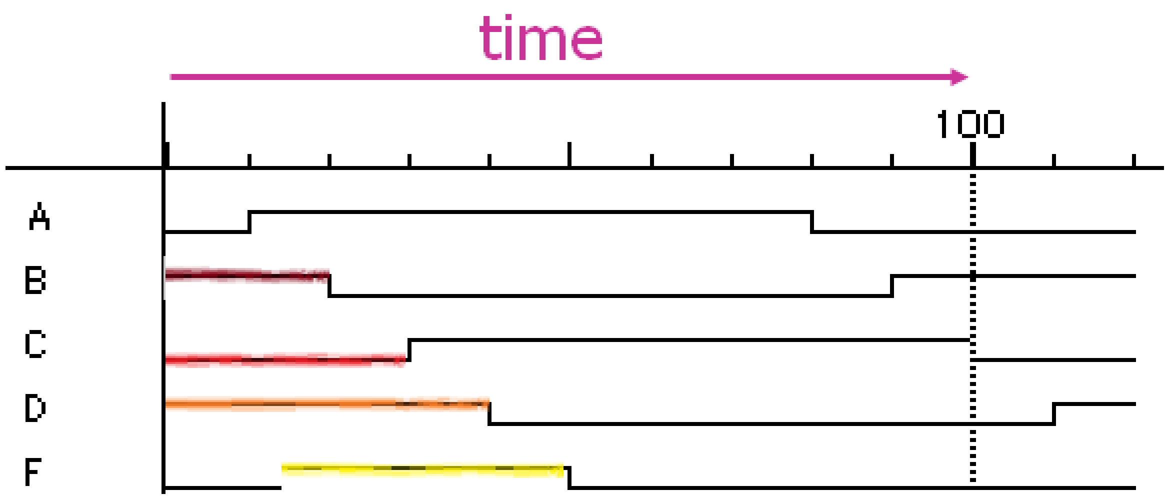
A sidebar

Difference?

Any assumptions?



All Signal starts at t=0

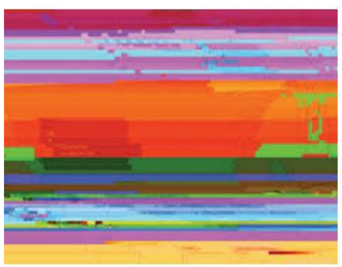
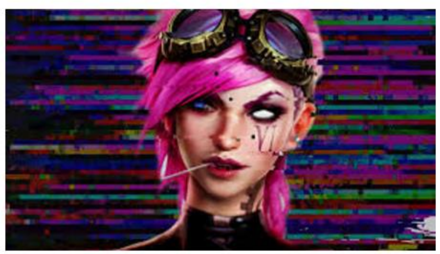


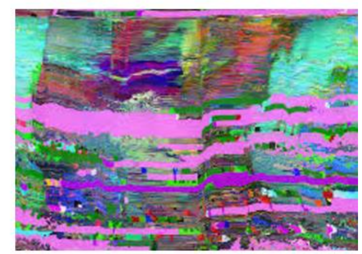
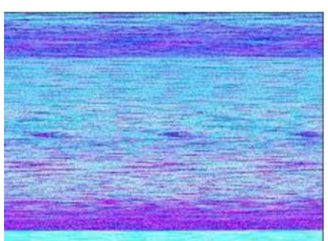
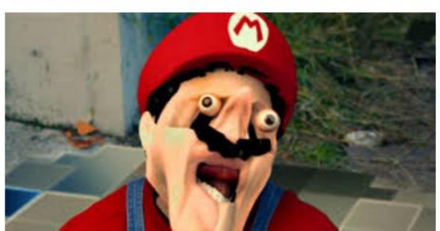
All Signal has existed before t=0

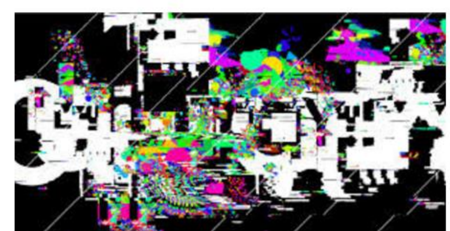
A glitch?

A glitch is a short-lived fault in a system. It is often used to describe a transient fault that corrects itself, and is therefore difficult to troubleshoot.





Page 92, Harris

4-input OR	90
tristate (A to Y)	50
tristate (enable to Y)	35

meaning
cture
ill stick
for
ions to

2.9.2 Glitches

So far we have discussed the case where a single input transition causes a single output transition. However, it is possible that a single input transition can cause *multiple* output transitions. These are called *glitches* or *hazards*. Although glitches usually don't cause problems, it is important to realize that they exist and recognize them when looking at timing diagrams. Figure 2.75 shows a circuit with a glitch and the Karnaugh map of the circuit.

The Boolean equation is correctly minimized, but let's look at what happens when $A = 0$, $C = 1$, and B transitions from 1 to 0. Figure 2.76 (see page 94) illustrates this scenario. The short path (shown in gray) goes through two gates, the AND and OR gates. The critical path (shown in blue) goes through an inverter and two gates, the AND and OR gates.

Glitches

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

Why Understand Glitches?

- As long as we wait for the pd to elapse, glitches are not a problem as the output eventually settles to the right answer.
- 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

What does cause a glitch?

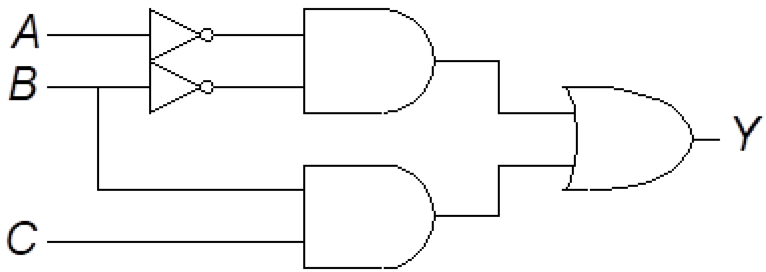
- Signal arrival time differences to destination
  - Delay
  - Critical path vs. Shortest path

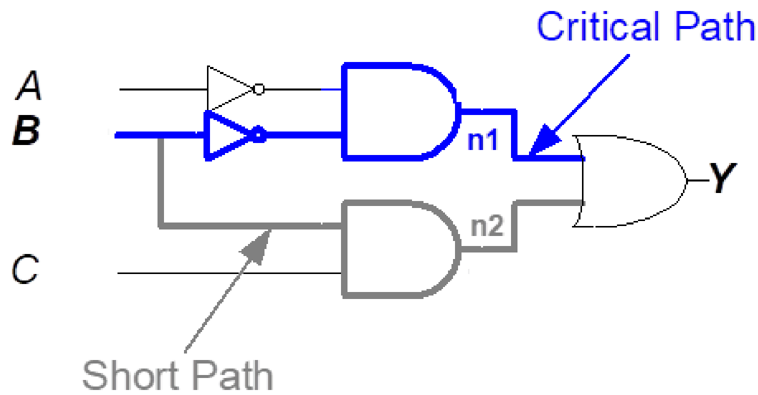
- Glitch analysis
  - (given logic gate delay) Circuit diagram + Timing diagram
- If there is a glitch (potential hazards), then fix it
  - K-map
- Types of static hazard
  - static 1-hazards: output supposes to be 1, but momentarily goes to 0
  - static 0-hazards: vice versa.

- For single-bit input changes,
  - A sum-of-products (SOP) expression or circuit can exhibit only static 1-hazards
    - SOP can never exhibit static 0-hazards
  - A product-of-sums (POS) expression or circuit can exhibit only static 0-hazards
    - POS can never exhibit static 1-hazards
- Meaning that **for one function "F"**
  - Different hazards can be introduced based on <u>**your circuit implementation**</u>
  - Recall that POS and SOP exercise=>we can have identical Boolean expressions with POS and SOP

Glitch Example

- What happens when A = 0, C = 1, B falls (transition from 1 to 0)?



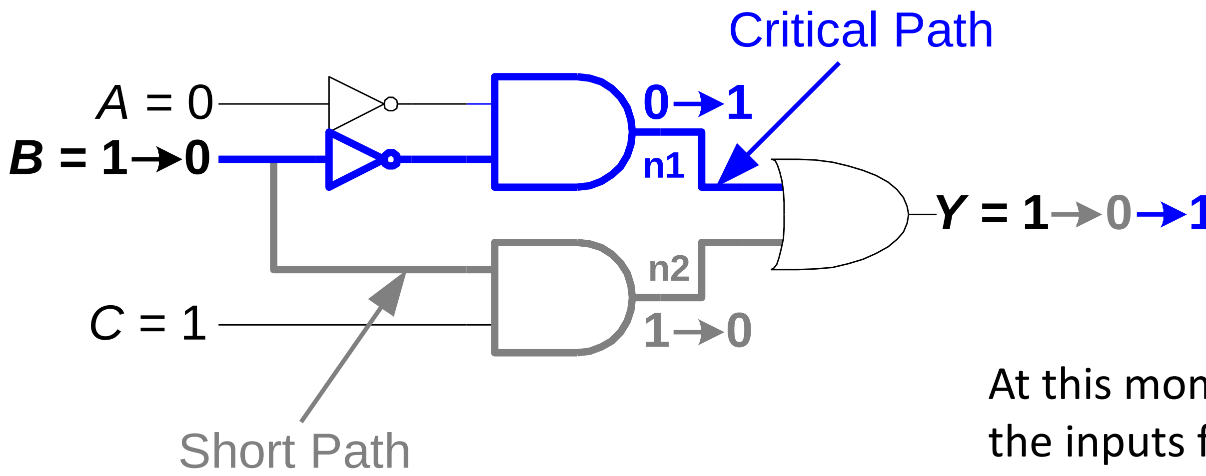


- This is an _SOP_ Implementation: (SOP or POS)
- Function Y=A’B’ + BC

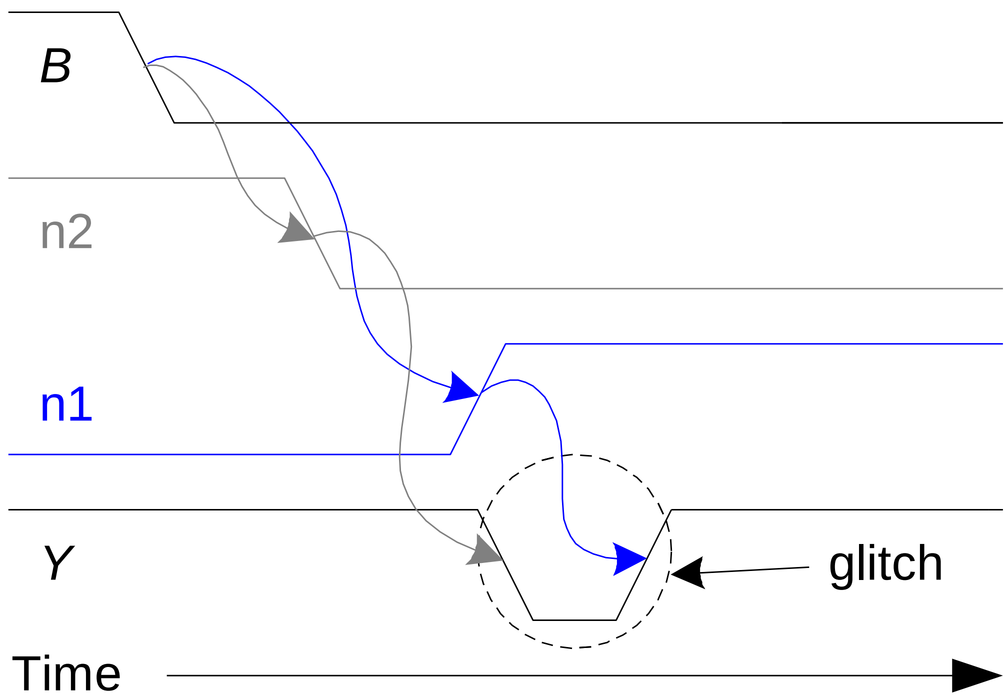
Y

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

- In terms of POS Y= (B’+C)(A’+B)
  - Another circuit implementation, but a same function



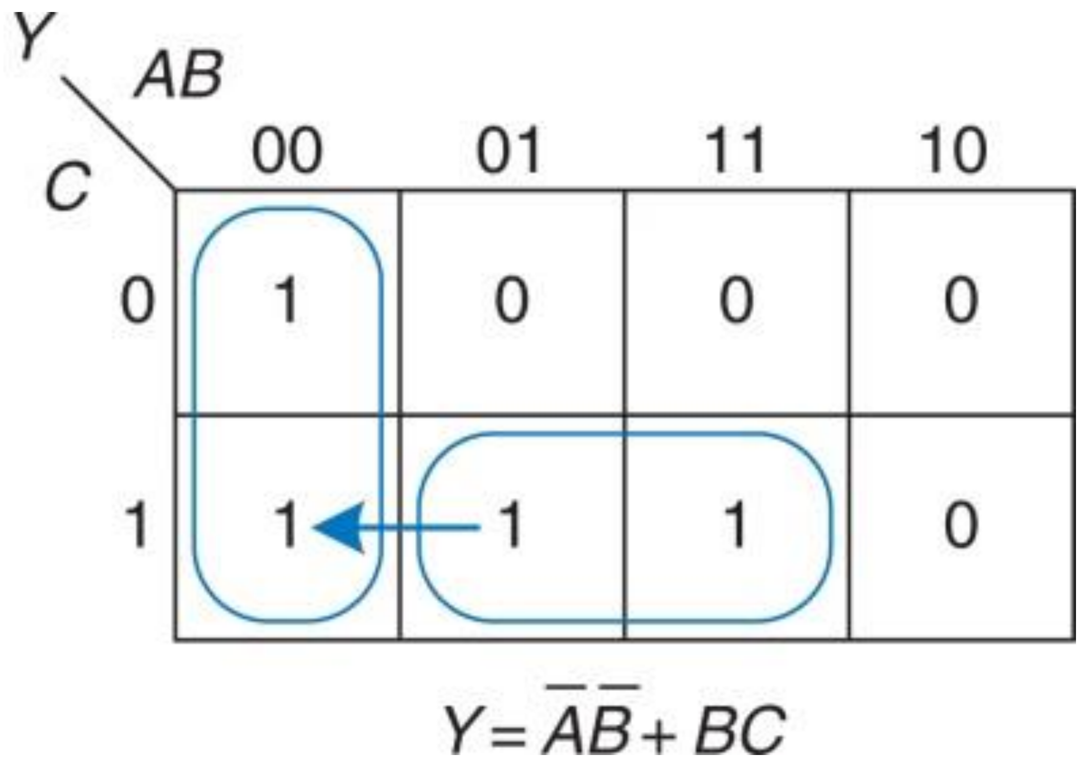
At this moment, one of the inputs for the OR gate is not stabilized yet



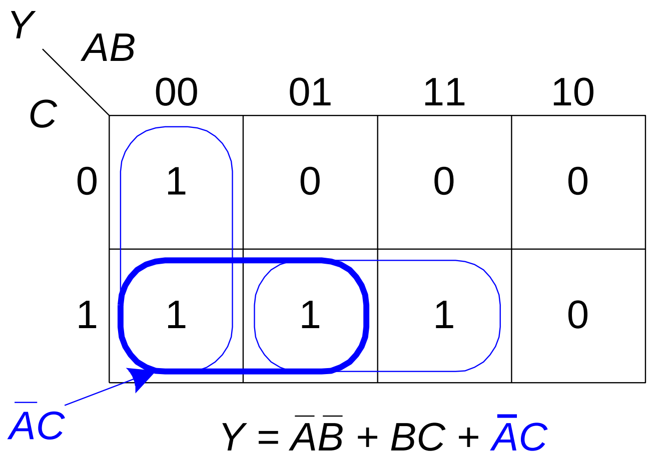
What happens when A = 0, C = 1, B falls (transition from 1 to 0?

➔ The response output shows a Glitch

Fixing the glitch



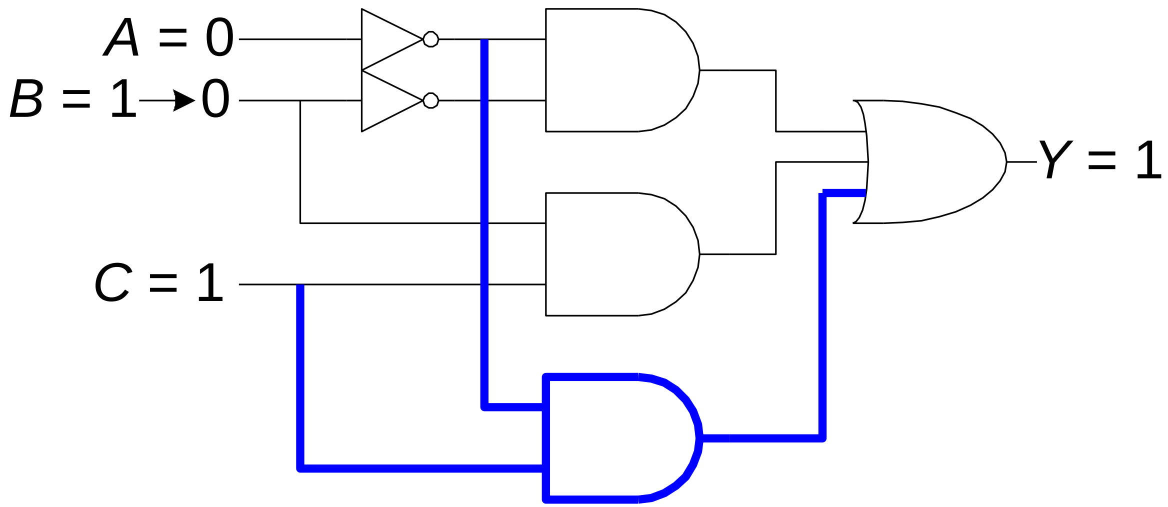
Input change crosses implicant boundary from 011 to 001.



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

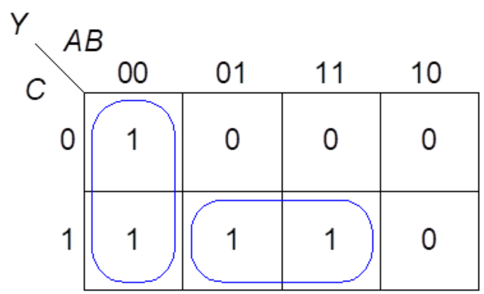
How can we prove that we fix the glitch?

- By timing diagram



How do we choose test implicants for glitch?

- Take implicants on kmap group boundaries

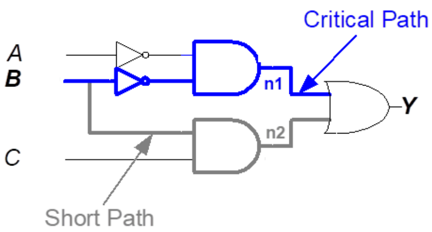


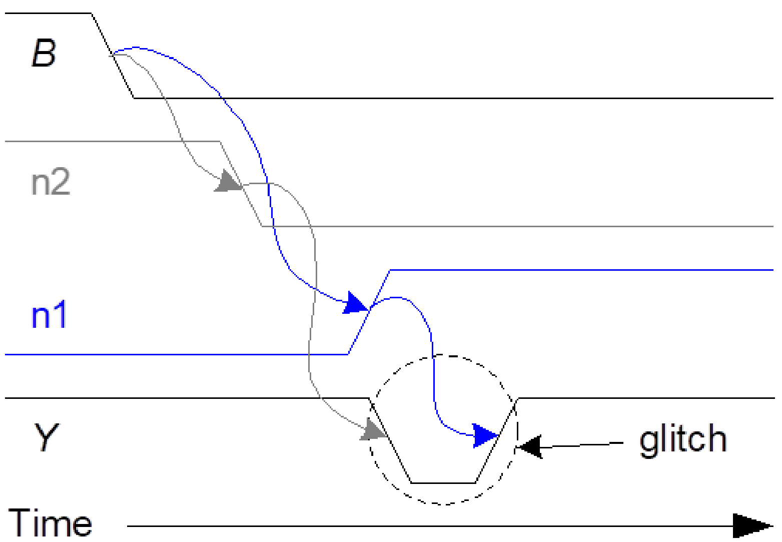
$Y = \overline{A}\overline{B} + BC$

- B changes between 0 and 1
- A and C stay same

- Do “timing diagram analysis”

- Take two edges from each transition of B (0->1 and 1->0)



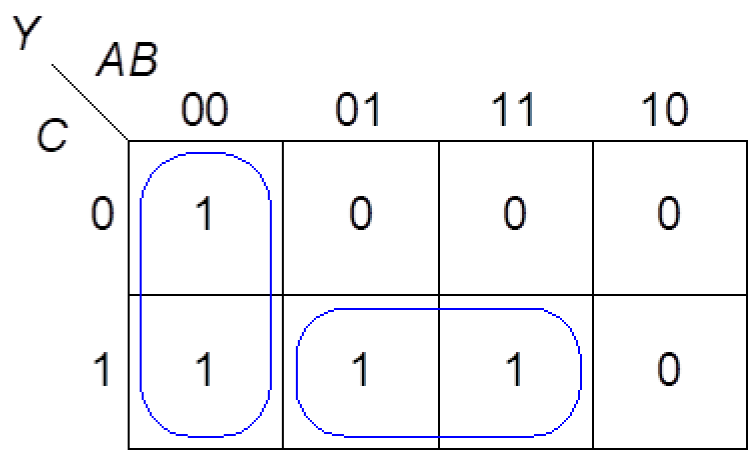


- Analyze output Y for both transitions
- Check if Y changes more than one time for each transition
  - A glitch (this case: static 1 hazard)
  - Y supposes to be 1 for both implicants (011 or 001) but momentarily goes to 0 (1→0→1)

Mostly,
1→0 input transition direction for SOP
0→1 input transition direction for POS
But, not always.

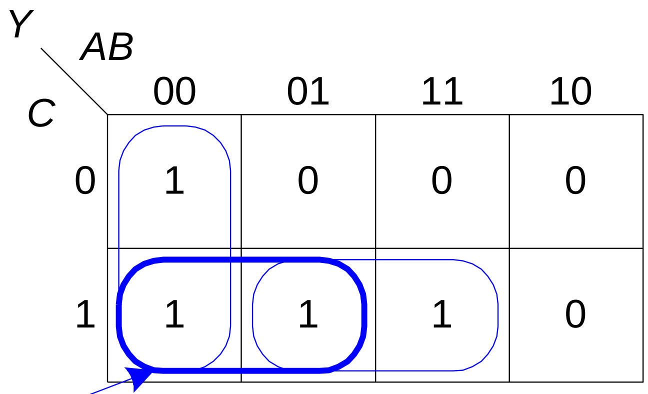
To remove a glitch

- Add Redundant Prime Implicants to Eliminate Static 1-Hazards in Sum-of-Products
- Add Redundant Prime Implicants to Eliminate Static 0-Hazards in *Product-of-Sums*

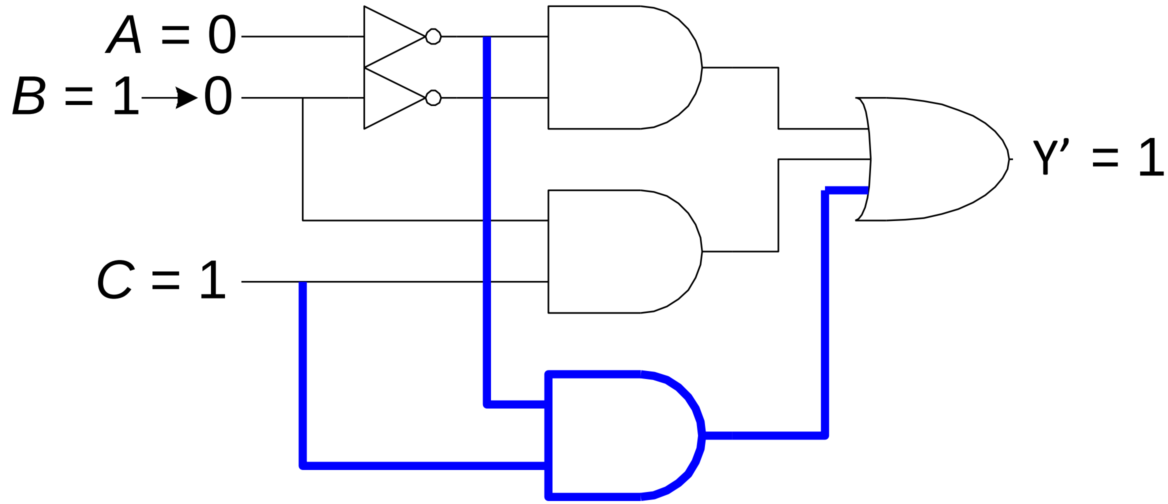


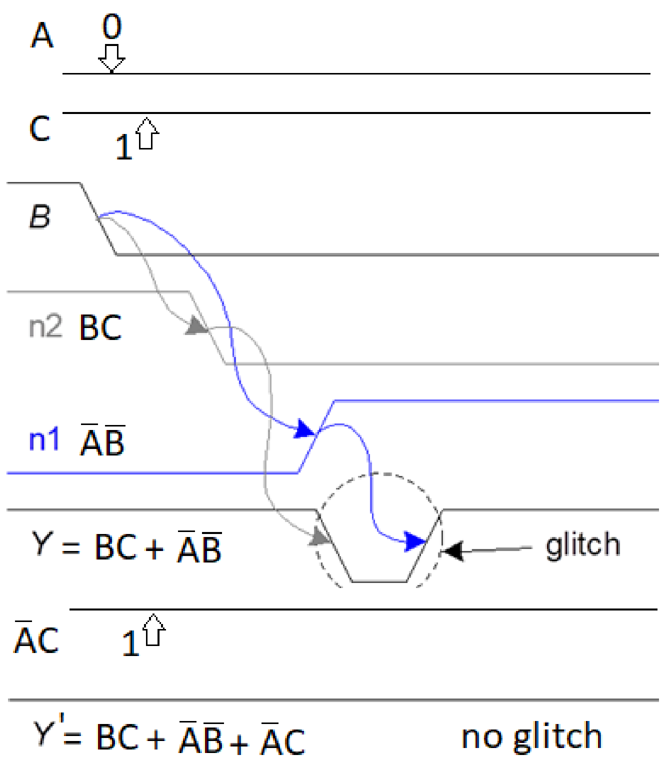
$Y = (B'+C)(A'+B)\ (A'+C)$

$Y = A'B' + BC + A'C$



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





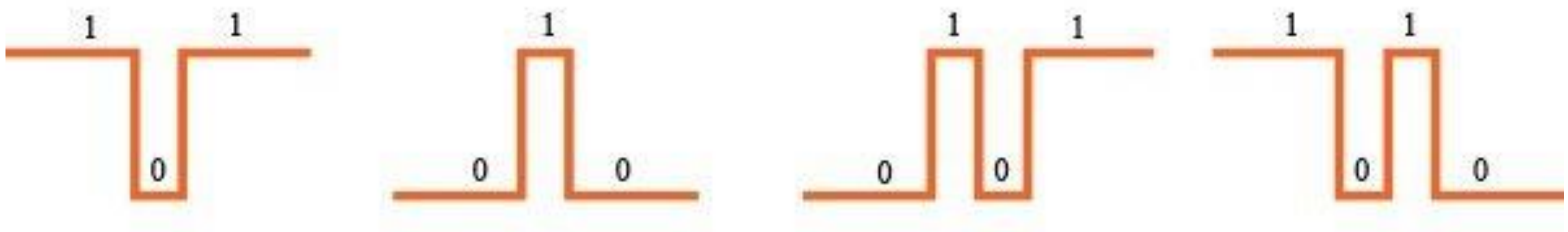
Hazards and glitches

- **A Glitch**: an unwanted output, a bump like thing
  - When a single input change causes an output to change multiple times
- A circuit with the potential for a glitch has a **hazard**.
- Glitches occur when different pathways have different delays
  - Causes circuit noise
  - Dangerous if logic makes a decision while output is unstable
- **Therefore, Hazards and glitches are closely related to each other.**

Hazards and glitches

- Solutions
  - Design hazard-free circuits (similar to our lab 4)
    - Difficult when logic is multilevel
  - Wait until signals are stable

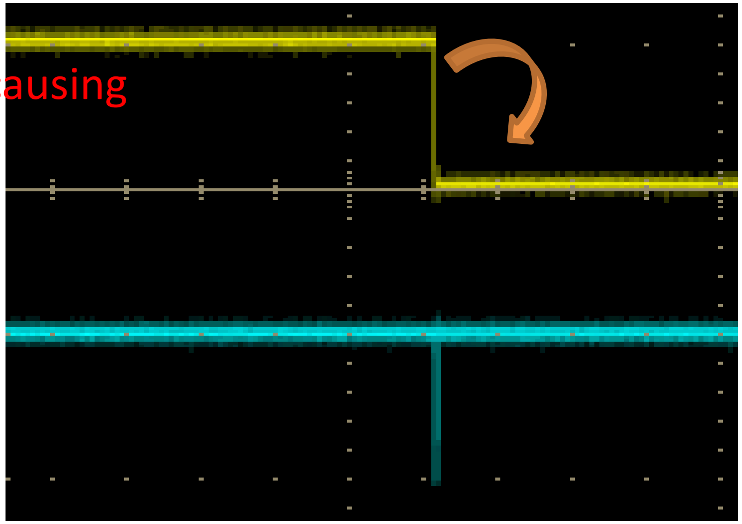
Types of hazard



(a) Static 1- hazard (b) Static 0- hazard (c) Dynamic hazards

- Static:
  - 1 hazard: output momentarily goes to 0 when it should remain a constant value of 1
  - 0 hazard: output momentarily goes to 1 when it should remain a constant value of 0
- Dynamic
  - an output may change three or more times

A hazard causing

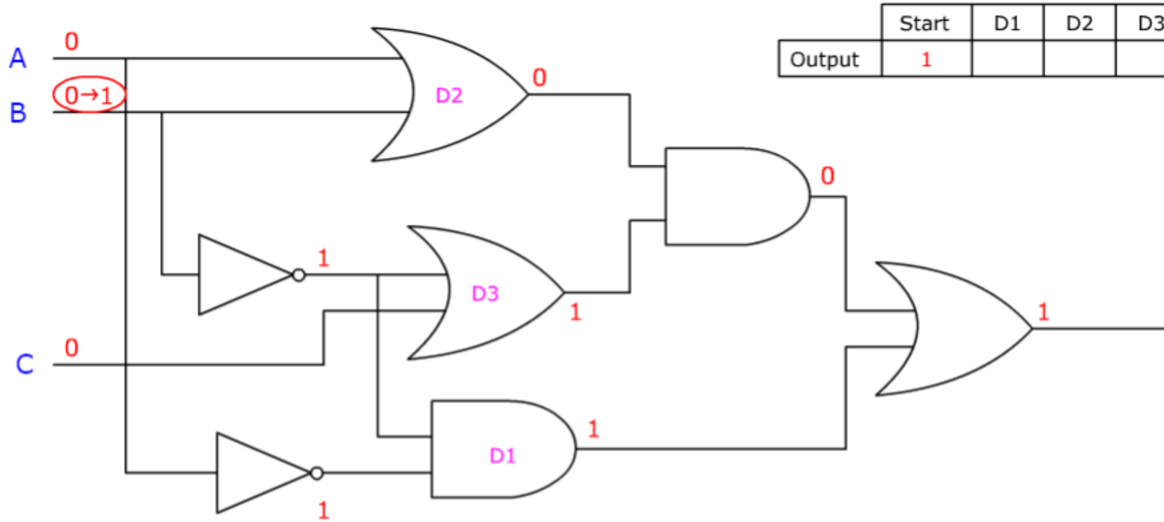


static 1 hazard

Dynamic hazard (a simple example)

Logic values at each stage

	Start	D1	D2	D3
Output	1			



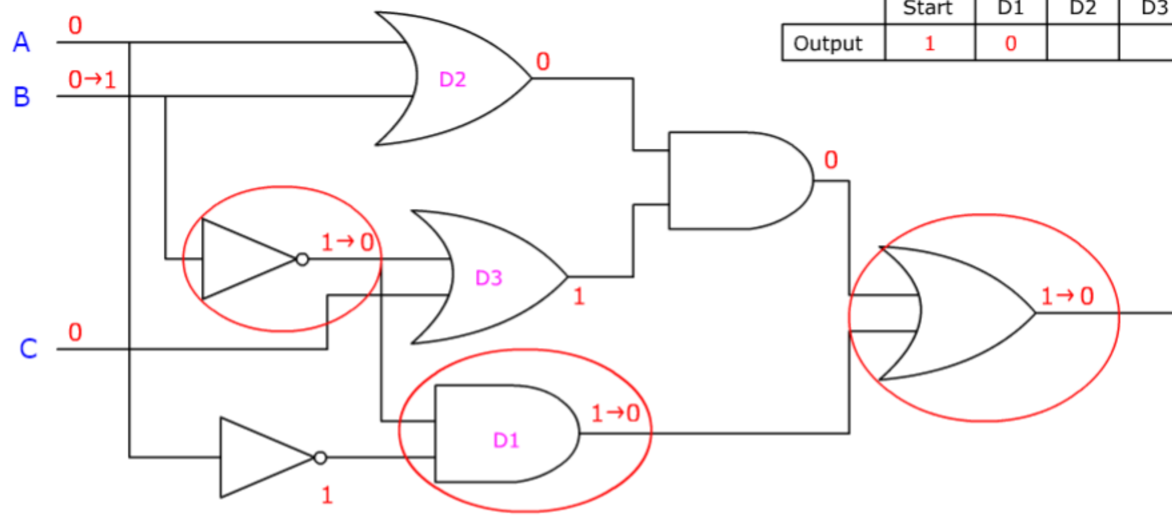
Y=((A+B)(B’+C))+(A’B’)

- ABC=000 , Y=1
- ABC=010 , Y=0

- We will see actually Y changes more than one time during this simulation

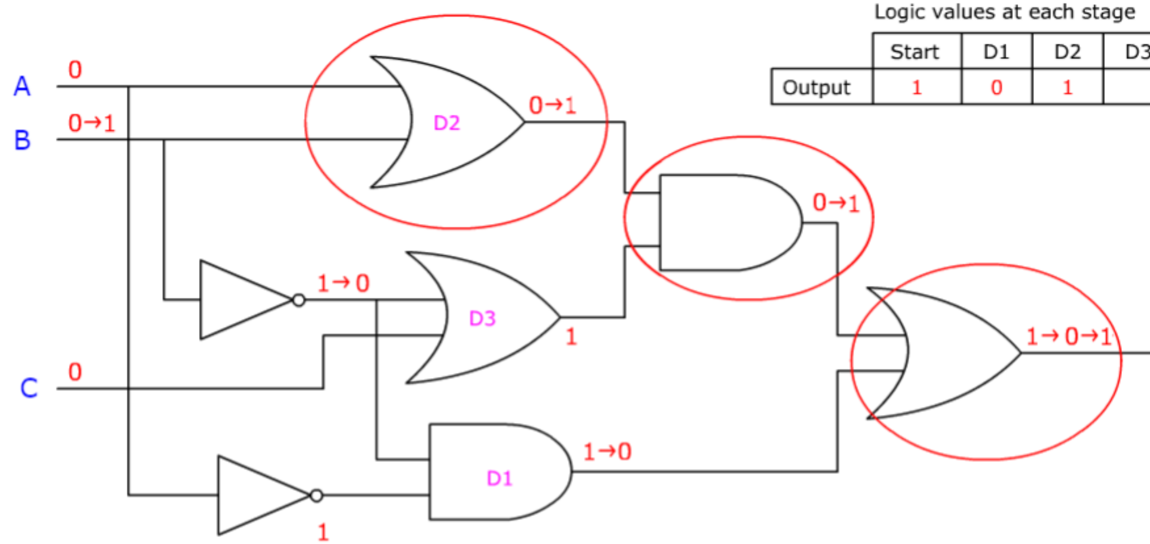
Logic values at each stage

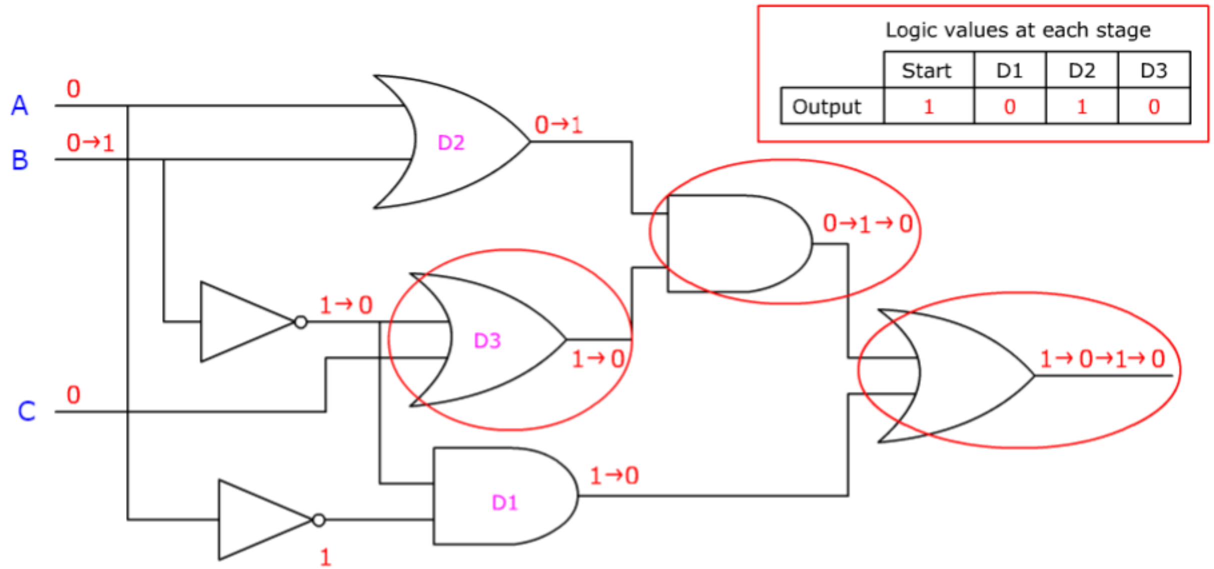
	Start	D1	D2	D3
Output	1	0		



Logic values at each stage

	Start	D1	D2	D3
Output	1	0	1	





	Start	D1	D2	D3
Output	1	0	1	0

Y=((A+B)(B'+C))+(A'B')

- ABC=000 , Y=1
- ABC=010 , Y=0

- Y supposes to change from 1 to 0 (one time, it's not static!) but as we can see Y changes 1->0->1->0 (multiple times) → a dynamic hazard!