Computer Architecture

Digital logic: Part 2

Combinational logic

Agenda

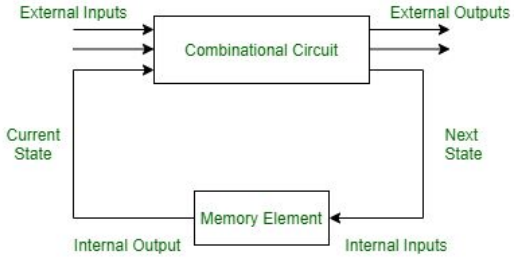
- Two types of logic
- Combinational Logic
  - Decoder
  - Multiplexer
    - Two-level logic and PLA
- Timing

Two types of logic in digital systems

Logic: In a digital system, it refers to the set of rules and structures that govern how binary values (0s and 1s) are processed to produce a specific outcome. (i.e., the implementation of logic expressions in Boolean algebra.)

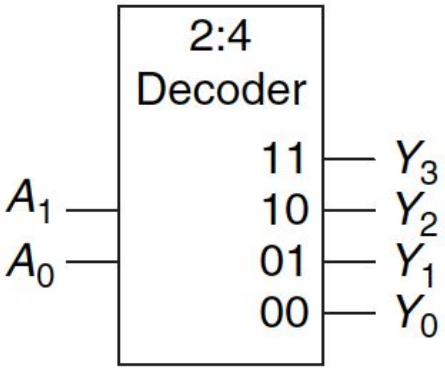
Two types of logic:

- **Combinational logic**: Output depends solely on the current inputs, without any memory of past states. It's used for operations that need immediate calculation, like arithmetic and data selection.
- **Sequential logic**: Output depends on both current inputs and past states, providing memory through components like flip-flops and registers. Sequential logic enables systems to remember information and operate in stages, crucial for tasks that need state retention and timing control.



Circuits for combinational logic

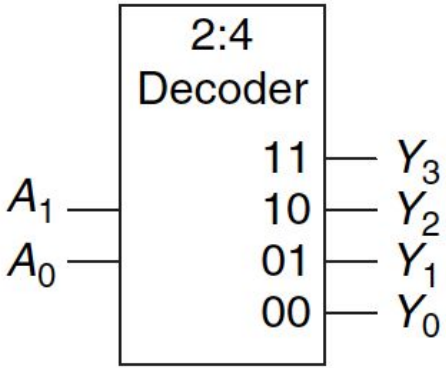
Decoders

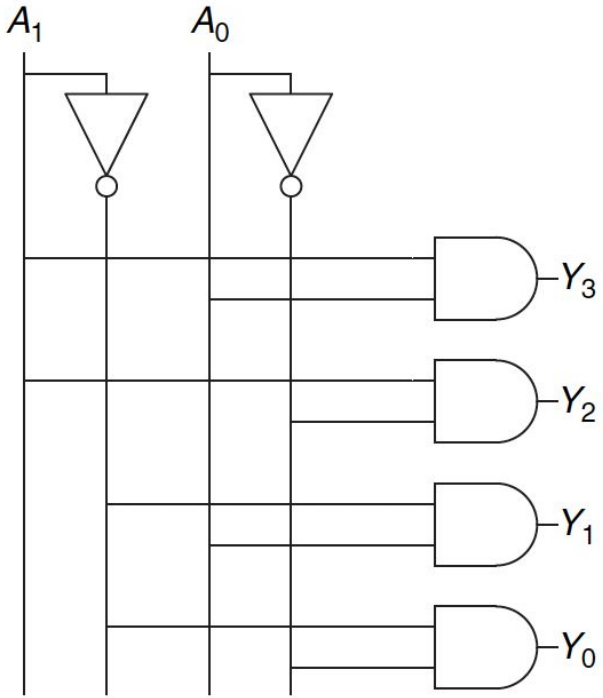


A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

- A decoder is a lookup table
- Each pattern of the inputs A_1A_0 is mapped to a pattern of the outputs

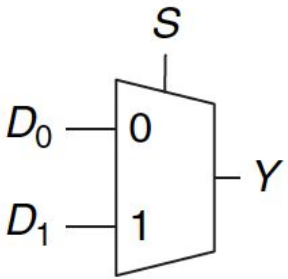
Implementation of the 2:4 decoder





Multiplexor (mux)

- It is a selector ⇒ the output is one of the inputs that is selected by a control.

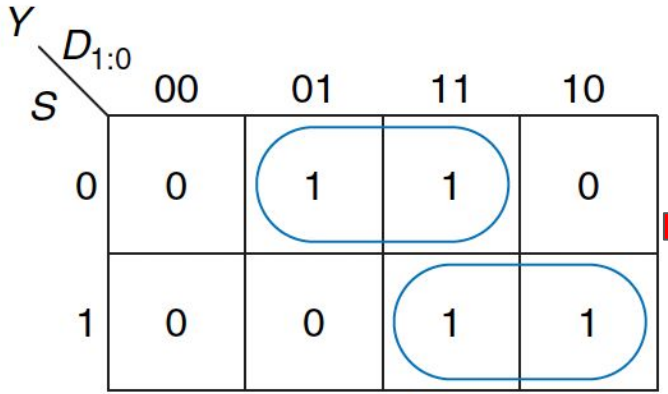


S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

sum of products:

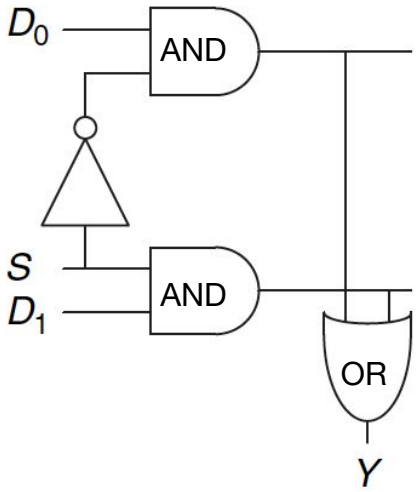
$$Y = \bar{S}\bar{D}_1 D_0 + \bar{S} D_1 D_0 + S D_1 \bar{D}_0 + S D_1 D_0$$

Simplifying the expression using K-map

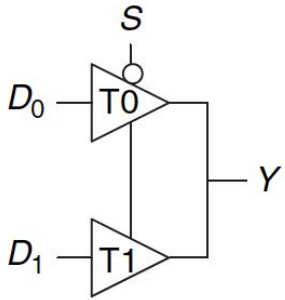


$$Y = \bar{S}D_0 + SD_1$$

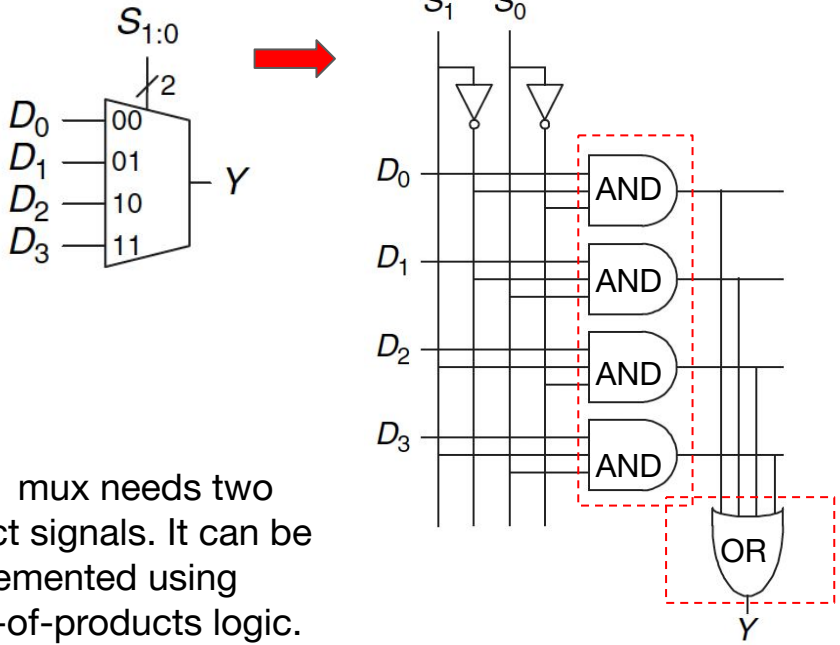
using logic gates



Or, using tristate buffer



4:1 multiplexers



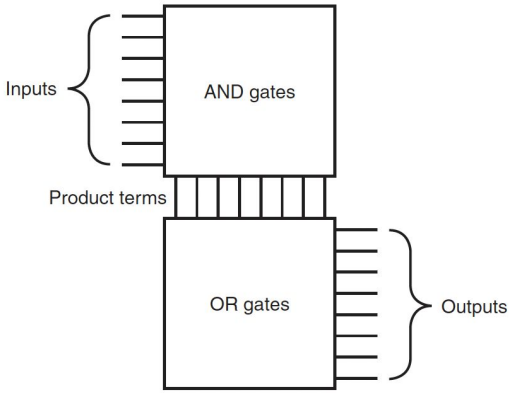
A 4:1 mux needs two select signals. It can be implemented using sum-of-products logic.

Two-level logic:

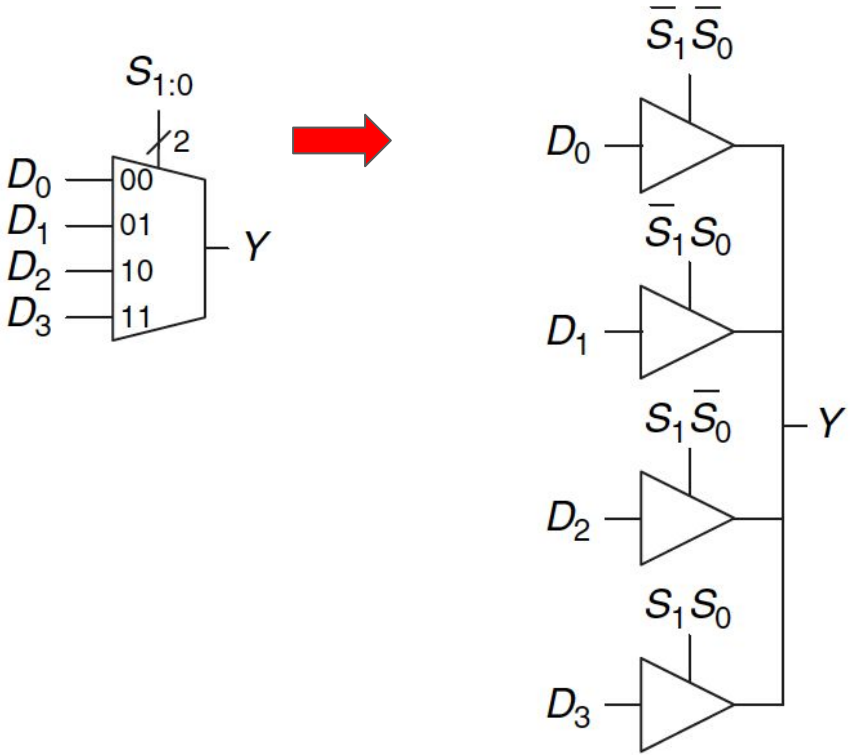
- Sum-of-products form is called two-level logic
- It consists of literal connected to a level of **AND** gates connected to a level of **OR** gates.
- The circuits can be simplified using techniques like bubble pushing and K-map.

Two-level logic and PLAs

- The sum-of-products representation corresponds to a common structured-logic implementation called a programmable logic array (PLA).
- A PLA has a set of inputs and corresponding input complements (which can be implemented with a set of inverters), and two stages of logic.
    - The first stage is an array of AND gates that form a set of product terms (also, called minterms); each product term can consist of any of the inputs or their complements.
    - The second stage is an array of OR gates, each of which forms a logical sum of any number of the product terms.



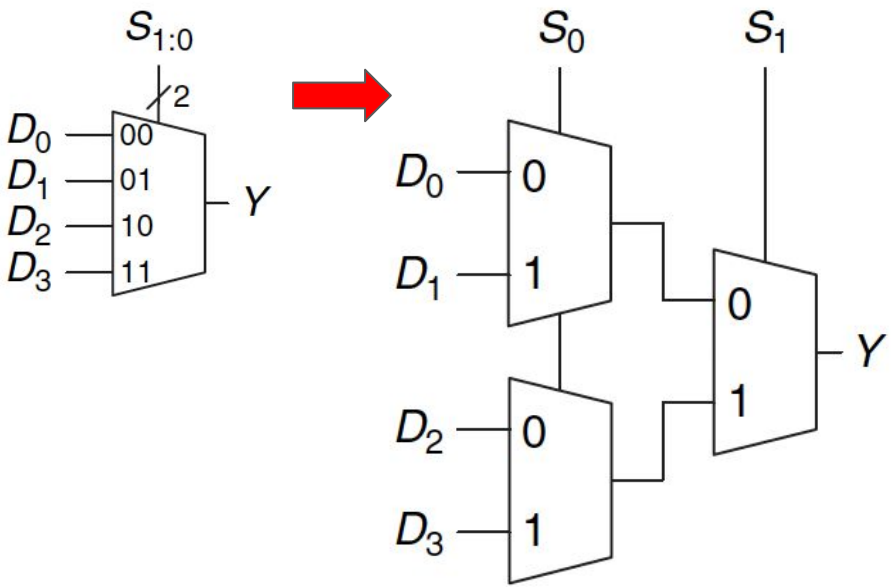
Implementation of 4:1 multiplexer



Tristates implementation

- The enable signal (S_1S_0) can be formed using AND gates and inverters.

Implementation of 4:1 multiplexer



Hierarchical implementation:

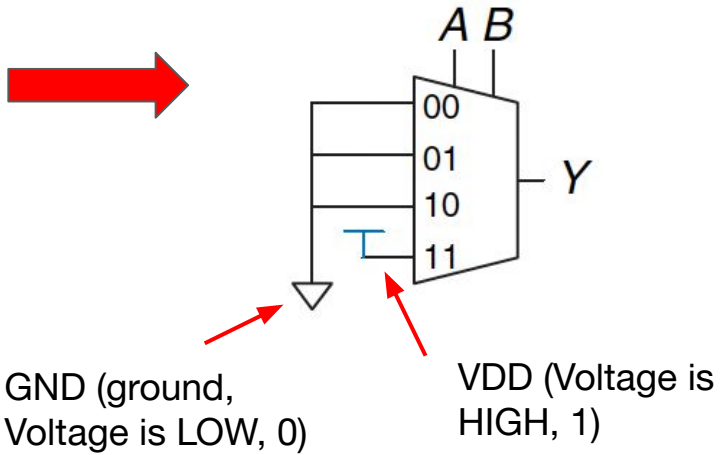
- Using two stages of 2:1 multiplexers

Logic with multiplexers

- Multiplexers can be used as lookup tables with the appropriate data inputs.
  - e.g., a 4:1 mux can be used to implement a two-input AND gate. A and B serve as select lines. (the inputs make it behave like an AND gate)

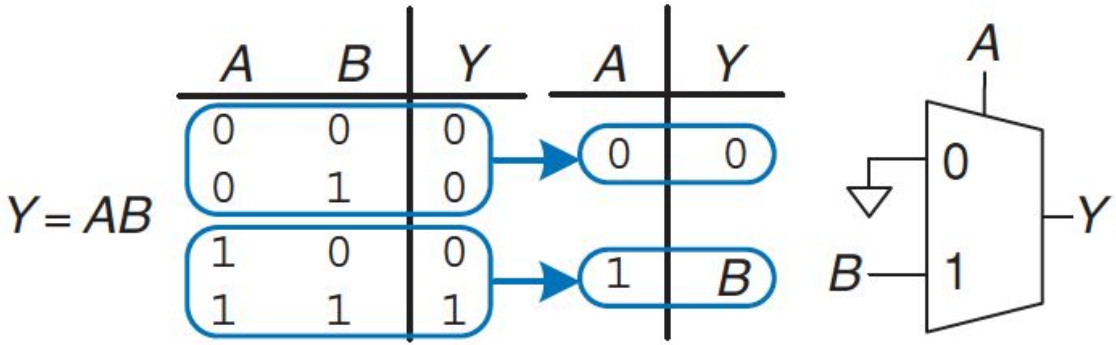
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



Logic with multiplexers

- 2:1 multiplexers can be used as AND gate.
  - Combine the rows to **eliminate the right most input variable** by expressing the output in terms of this variable. (By this way, we can represent use 2^{N-1} input multiplexer to perform any N-input logic function.)



Exercise (example 2.12 in Harris and Harris)

Alyssa P. Hacker needs to implement the function

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

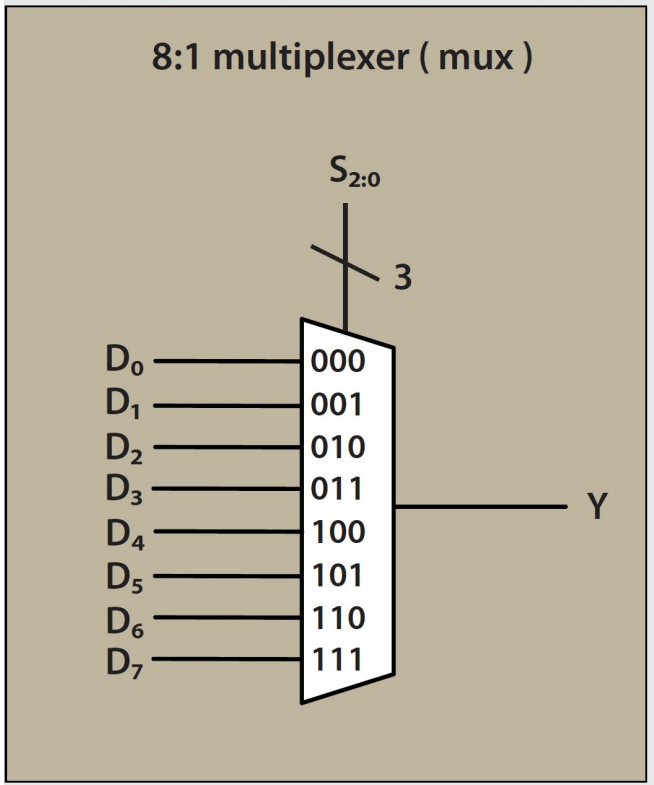
to finish her senior project, but when she looks in her lab kit, the only part she has left is an 8:1 multiplexer.

How does she implement the function?

Solution

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

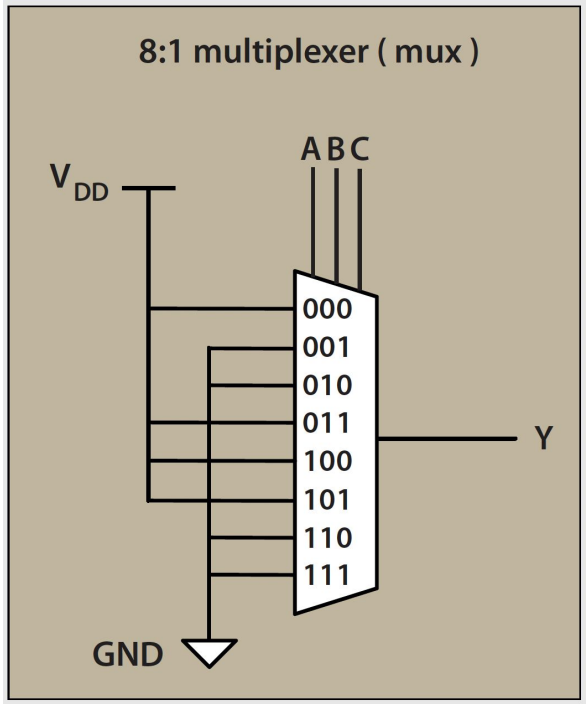
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



Solution

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

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

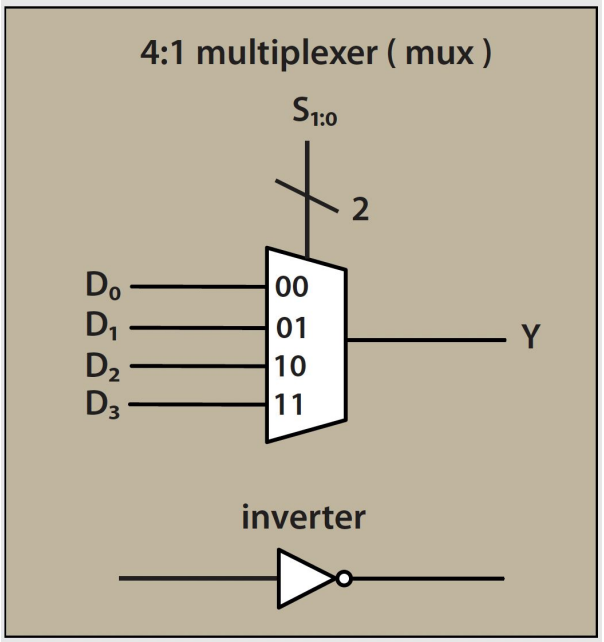


Exercise

Can you construct the same boolean function with a 4:1 multiplexer and an inverter?

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

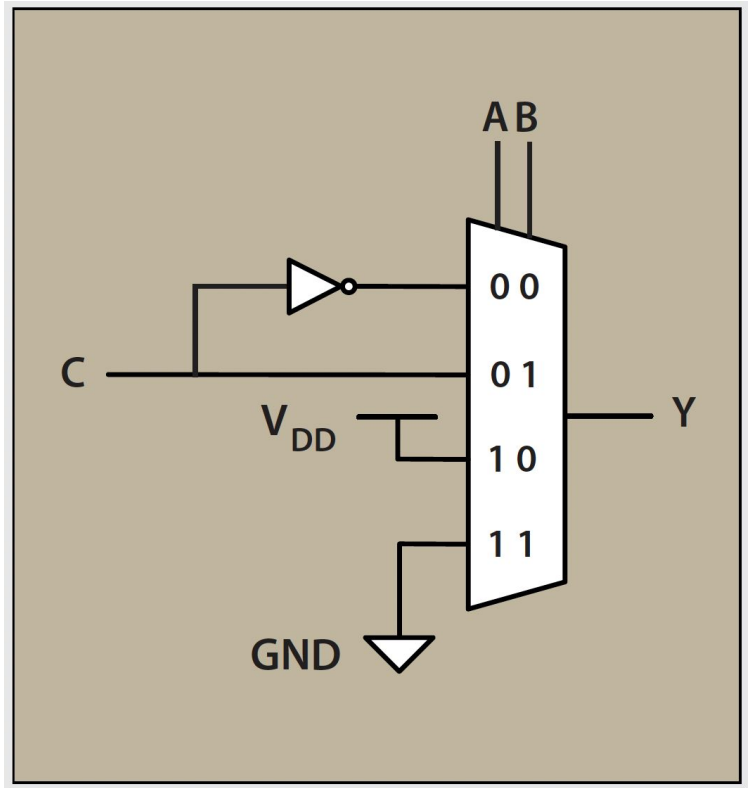
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



Solution

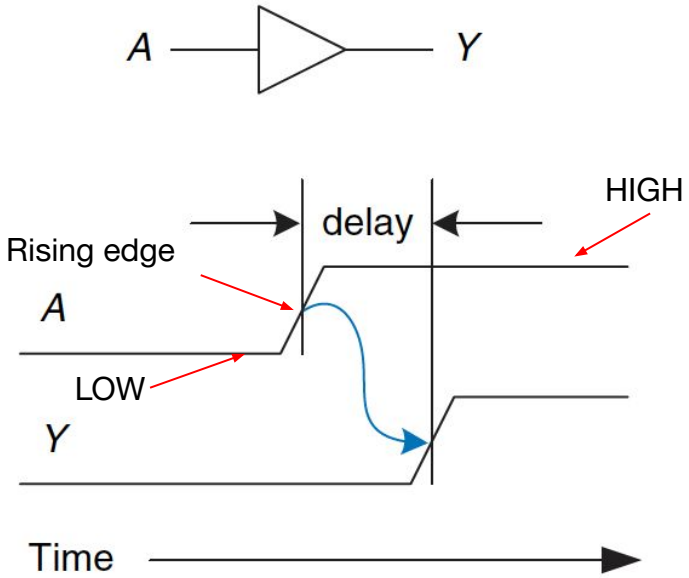
$$Y = A\overline{B} + \overline{B}\overline{C} + \overline{A}BC$$

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



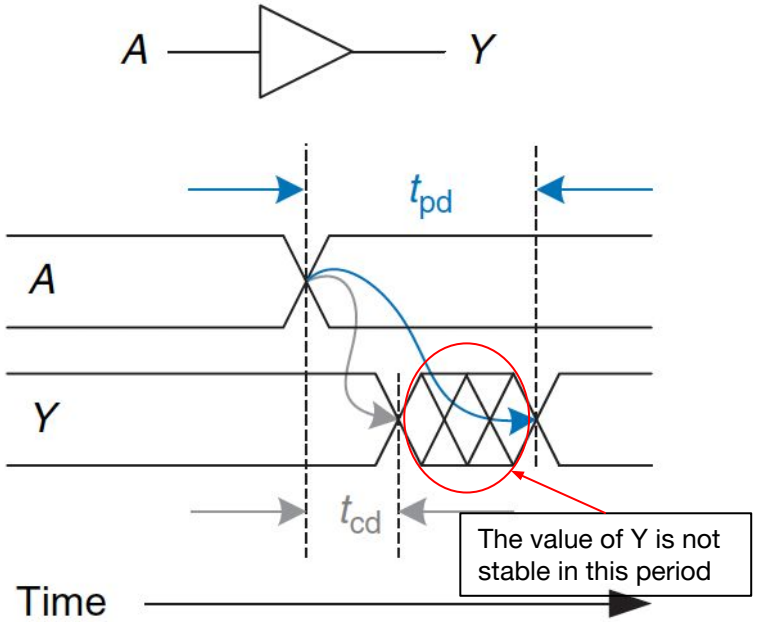
Timing

- An output takes time to change in response to an input change.



- The figure (called timing diagram) shows the transient response of the buffer circuit when the input changes.
- The transition from LOW to HIGH is called rising edge.
- The transition from HIGH to LOW is called falling edge.
- The blue arrow indicates the rising edge of Y is caused by the rising edge of A.

Propagation and contamination delay



The figure shows that A is initially either HIGH or LOW and changes to the other state at a particular time; we are interested only in the fact that it changes, not what value it has. In response, Y changes some time later.

- $\mathbf{t_{pd}}$ **:** the **propagation delay**, the maximum time from when an input changes until the output or outputs reach their final value.
- $\mathbf{t_{cd}}$ **:** the **contamination delay**, the minimum time from when an input changes until any output starts to change its value.

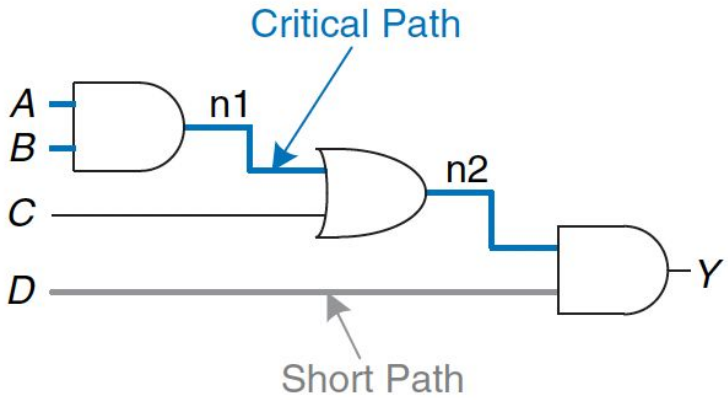
Why there are delays?

- Time required for capacitance
- The speed of light

Critical path and short path

- A **path** refers to a sequence of interconnected components (such as logic gates, multiplexers, etc.) through which a signal travels from an input to an output.

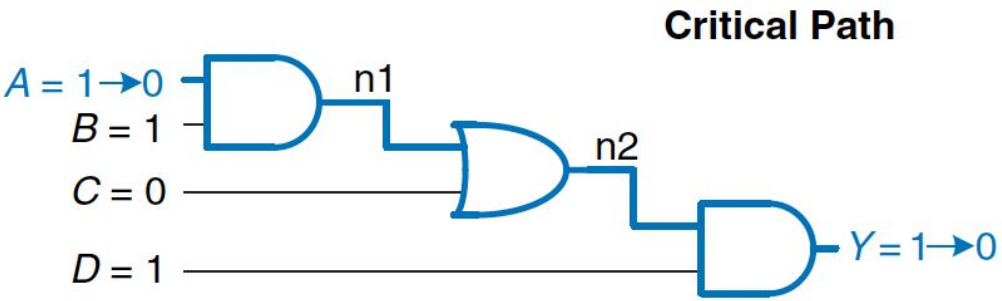
- The **critical path** is the **longest** path that signals must traverse in a digital circuit. (also, the slowest path since it is the longest one.)
- The **short path** is the **shortest** path through the circuit. (also, the fastest one.)

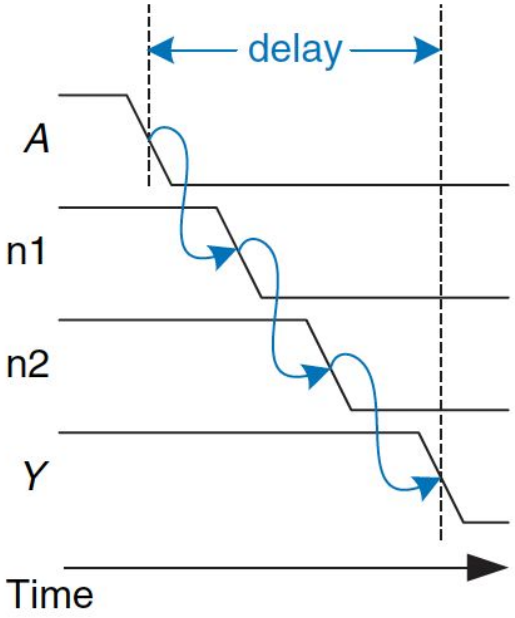


Calculating the delay of a circuit

- **worst-case value**: generally, designers take the worst-case value (i.e., the propagation delay of the critical path) as the delay of the circuit.
- Manufacturers normally supply data sheets specifying these delays for each gate.
  - Calculating t_{pd} and t_{cd} requires delving into the lower levels of abstraction beyond the scope of this course. But we use the data sheets from the manufactures.

Critical path waveform

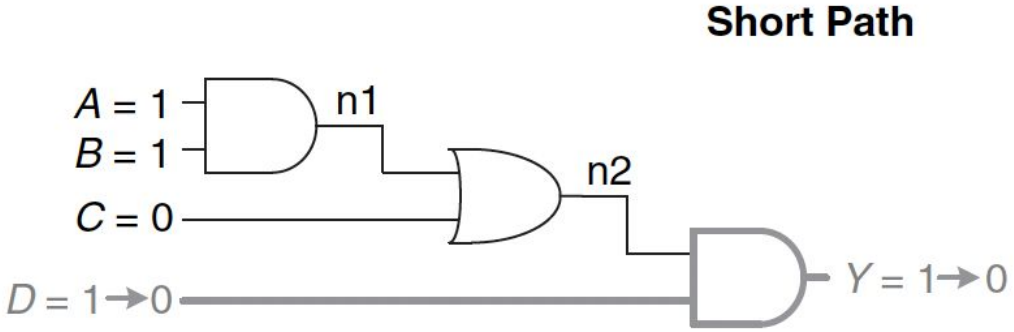


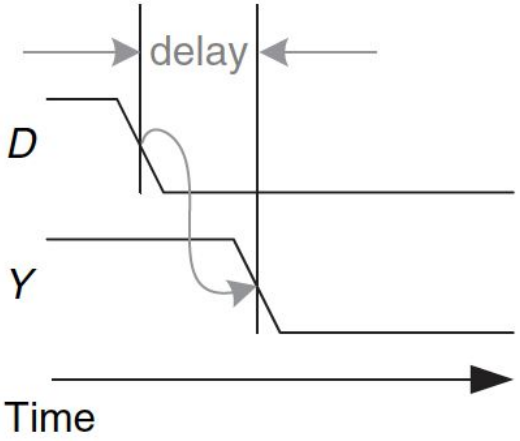


So, when A changes from 1 to 0, the delay of this circuit is:

$$t_{pd} = 2t_{pd\_\mathrm{AND}} + t_{pd\_\mathrm{OR}}$$

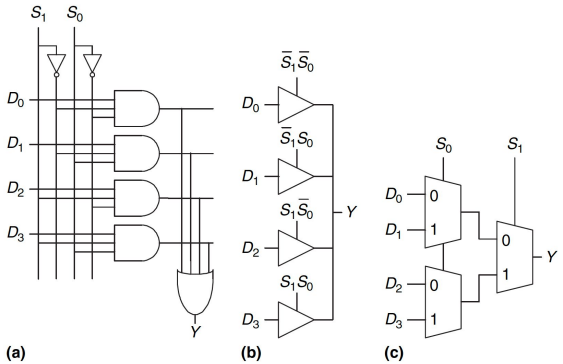
Short path waveform





Exercise (example 2.16 in Harris and Harris)

Given the propagation delays for the components given below, compare the worst-case timing of the three four-input multiplexer designs. What is the critical path for each design?

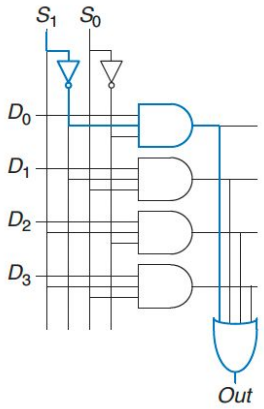


Given your timing analysis, why might you choose one design over the other?

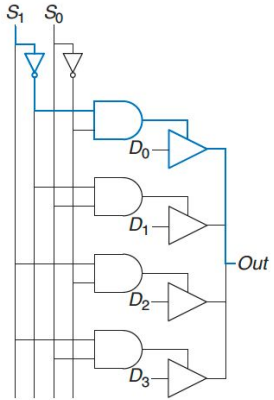
Gate	t_{pd} (ps)
NOT	30
2-input AND	60
3-input AND	80
4-input OR	90
tristate (A to Y)	50
tristate (enable to Y)	35

Exercise (example 2.16 in Harris and Harris)

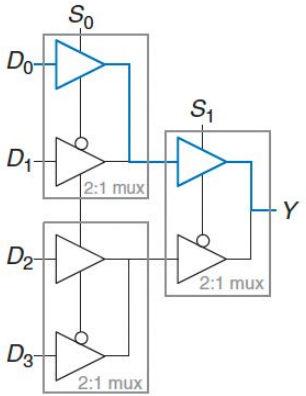
Critical path:



(a) 1 NOT + 1 AND + 1 OR



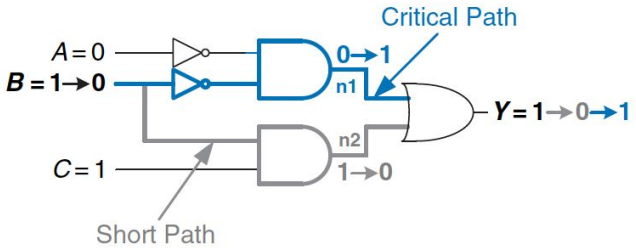
(b) 1 NOT + 1 AND + Transite (enable to Y)



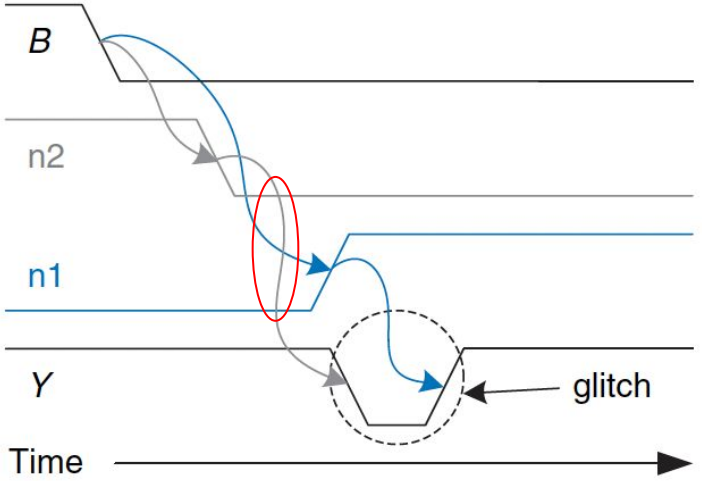
(c) 2 * Transit(D to Y)

Glitch

A glitch refers to an unintended and temporary fluctuation in the output signal.



- shortest path: going through two gates;
- critical path: going through three gates;
- when B transitions from 1 to 0, n2 falls before n1 can rise.
- until n1 rise, the two inputs to the OR gate are 0, and the output Y drops to 0.
- When n1 eventually rises, Y returns to 1.



Glitch

- Glitches usually don’t cause problems, but it is important to recognize them in timing diagrams.
- We can eliminate glitches by adding redundant implicants.