Slide Set 6

for ENEL 353 Fall 2019

Steve Norman, PhD, PEng

Electrical & Computer Engineering
Schulich School of Engineering
University of Calgary

Fall Term, 2019

Contents

Outline of Slide Set 6

Multiplexers

Multiplexer is rather a long word, so *mux* is often used as an abbreviation.

Another quite descriptive name for *multiplexer* is *selector*.

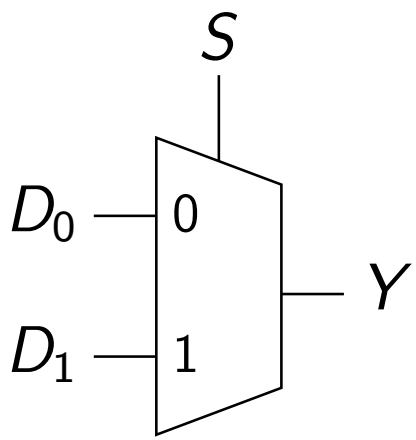
A multiplexer circuit has

- two or more *data inputs*;
- one or more bits of *select input*;
- an output.

The job of a multiplexer is to **copy one of the data inputs to the output**. The data input selected for copying is **chosen by the select input.**

The 2:1 multiplexer ("two-to-one mux")

A circuit symbol and truth table:

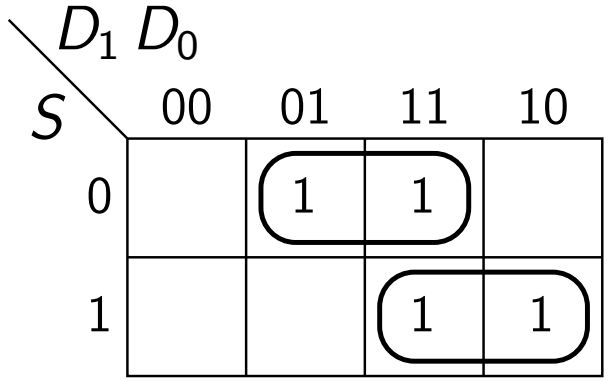


D_1	D_0	S	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Let's write out a few sentences to describe exactly what this circuit does.

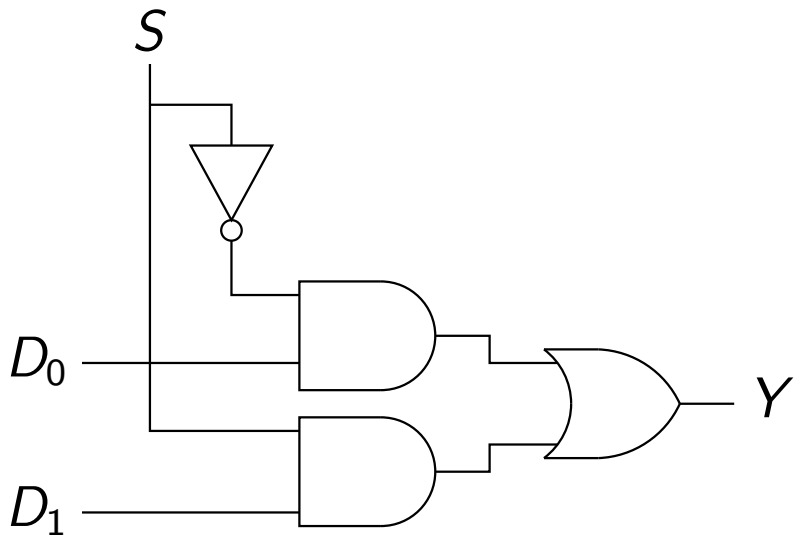
2:1 mux — built from NOT, AND and OR gates

The truth table from the previous slide results in this K-map and minimal SOP expression ...



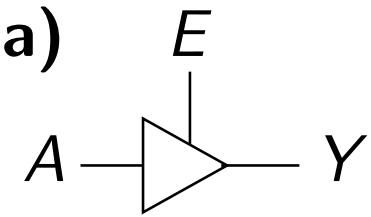
$$Y = D_0\bar{S} + D_1S$$

A circuit for this made from NOT, AND, and OR gates ...



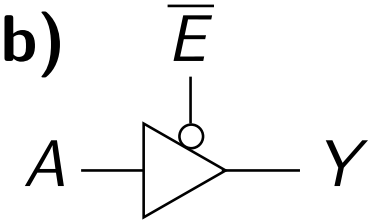
Variations on the tristate buffer

(a)



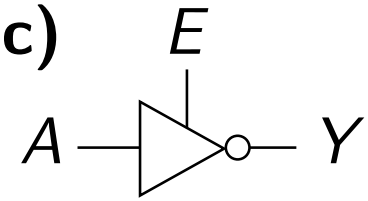
non-inverting,
active high enable

(b)



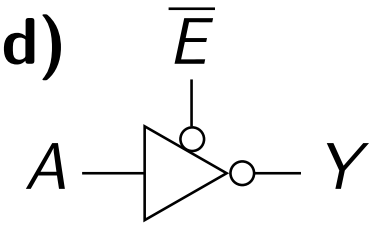
non-inverting,
active low enable

(c)



inverting,
active high enable

(d)

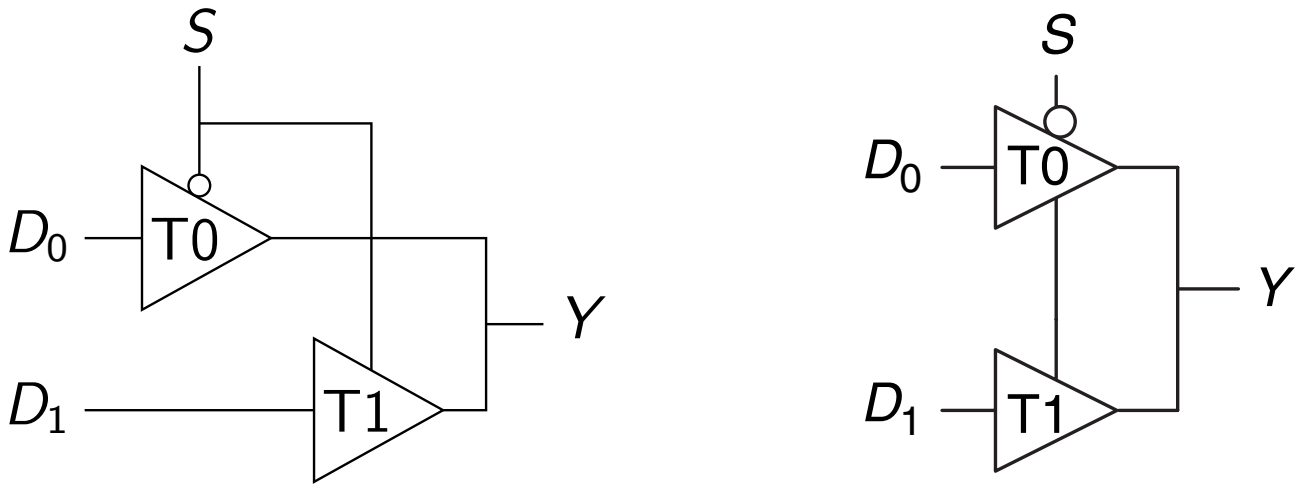


inverting,
active low enable

Version (a) is the one we have already looked at. But each of (b), (c), and (d) is sometimes useful as well. *Let's make tables to describe all four circuits.*

2:1 mux implemented with tristate buffers

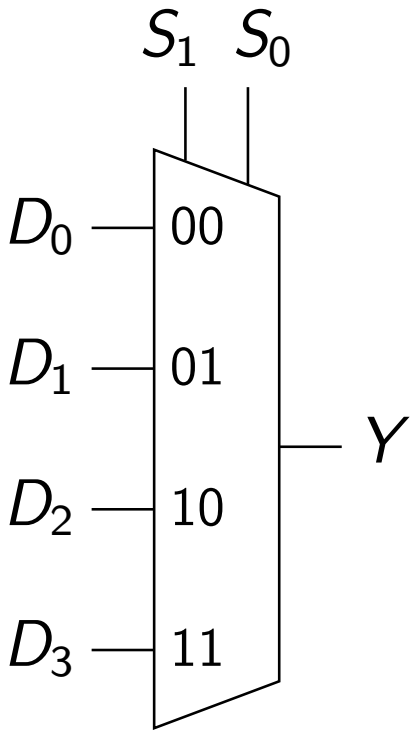
The two schematics describe **exactly the same design**. The one on the right is from the textbook, and uses a compact notation for showing a common control input wire for the tristate buffers.



Two gate outputs are wired together! Is that a problem in this design?

Image on right is from Figure 2.56 from Harris D. M. and Harris S. L., *Digital Design and Computer Architecture, 2nd ed.*, © 2013, Elsevier, Inc.

The 4:1 multiplexer ("four-to-one mux")



How many rows would a truth table for this circuit have?

Let's describe the circuit in a table that is more compact than a truth table.

Let's make some notes on these three designs for 4:1 mux circuits.

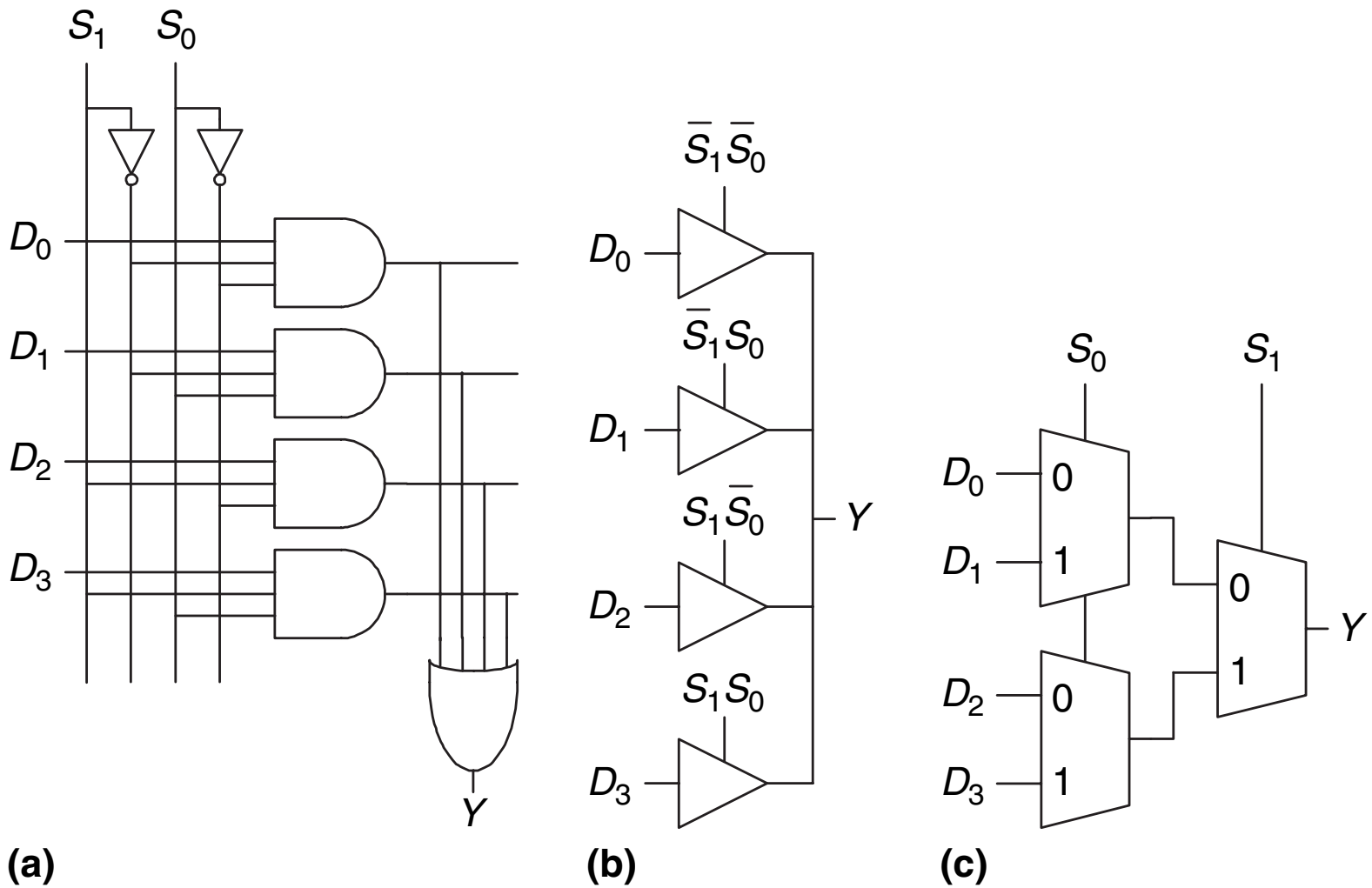


Image is Figure 2.58 from Harris D. M. and Harris S. L., *Digital Design and Computer Architecture, 2nd ed.*, © 2013, Elsevier, Inc.

Base two logarithms: What does $\log_2 N$ mean?

Base two logarithms are often useful in discussion of digital systems (and also in discussion of software algorithms and data structures).

You should already be familiar with **natural** (base e) and **common** logarithms, defined as follows:

- If $y = e^x$, then $\ln y = x$.
- If $y = 10^x$, then $\log_{10} y = x$.

The definition for the base two logarithm works the same way:

$$\boxed{\text{If } y = 2^x, \text{ then } \log_2 y = x.}$$

Wider multiplexers

If you understand 2:1 and 4:1 muxes, it's very easier to generalize to N:1, if N is a power of two.

The table to the right describes an 8:1 mux.

How many select bits would be needed for a 16:1 mux? For a 32:1 mux?

What would be a general formula for the number of select bits needed for an N:1 mux?

S_2	S_1	S_o	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

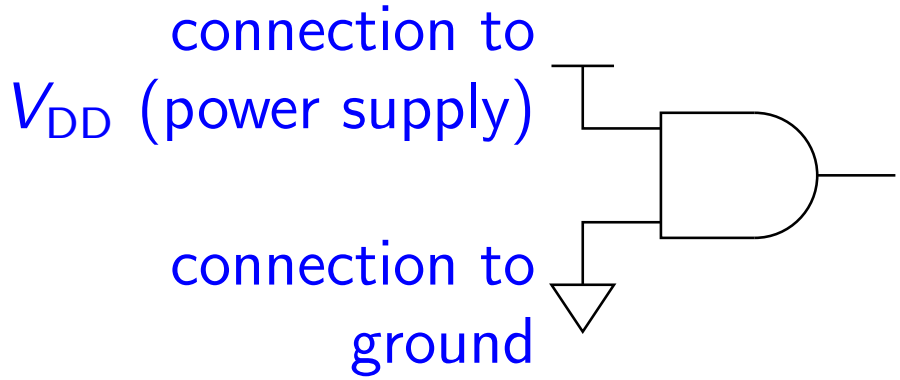
Using muxes to implement logic functions

Multiplexers have **a lot of practical applications**. We'll see some of them later in this course and others in ENCM 369 in Winter.

For now, we'll just look at one such application: Given a truth table with N rows for a function F, it's very straightforward to make a circuit for F with an N:1 multiplexer as the key component.

Textbook graphical notation for HIGH and LOW voltage connections

This schematic is an example of the symbols Harris & Harris use . . .



What is the output of the AND gate? (This is supposed to be an **easy** question.)

Example of using muxes to implement logic functions

How can we implement the given function with an 8:1 mux?

How can we implement the given function with a 4:1 mux and an inverter?

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

N:1 multiplexers with N not a power of two

Sometimes a circuit requires selection of one of N signals, where N is not a power of two. This is easy to accommodate.

For example, let's describe a 3:1 mux.

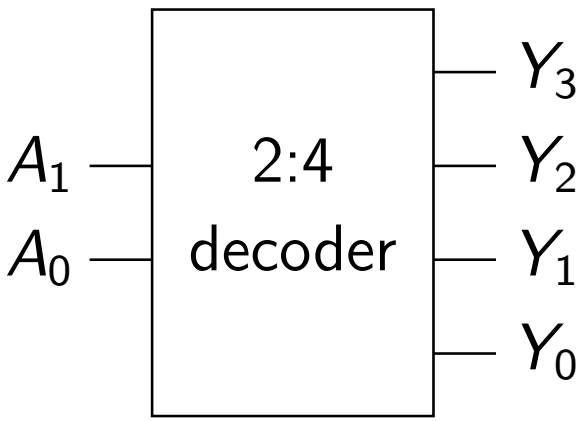
Let's build a 3:1 mux from two 2:1 muxes.

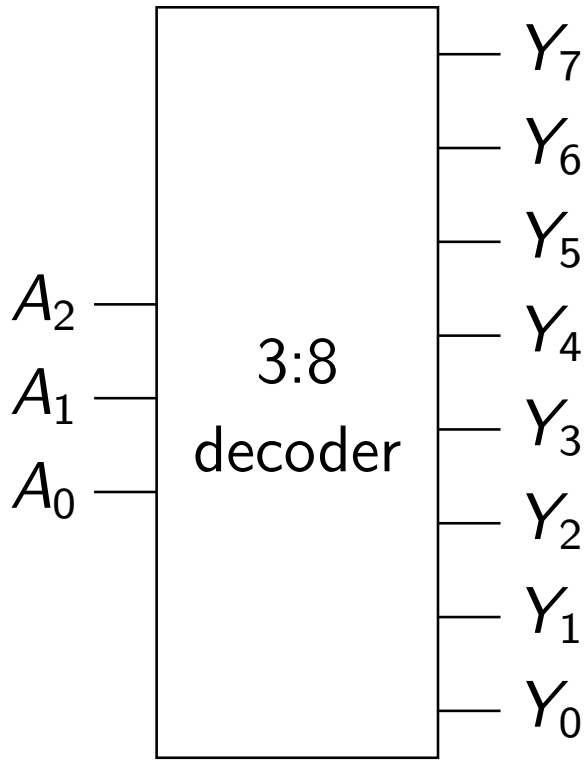
Let's build a 3:1 mux using a 4:1 mux.

Outline of Slide Set 6

Decoders

A *decoder* has N inputs and 2^N outputs. Here are examples of 2:4 and 3:8 decoders ...





Let's write a description of the 2:4 decoder, then show how to make one using inverters and AND gates.

Building logic functions with decoders

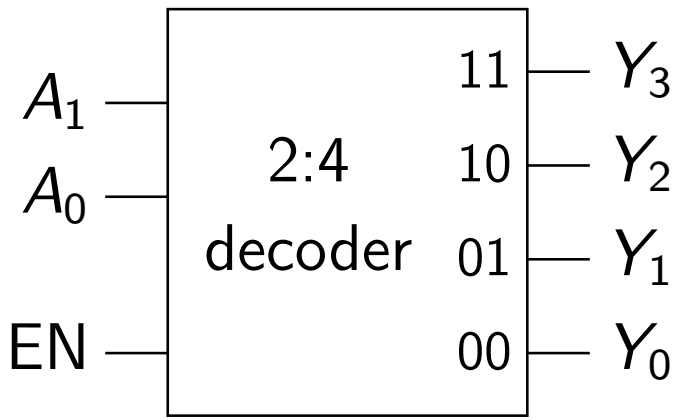
Because decoders are **minterm generators**, decoders can be used to go from truth tables to circuits in a very straightforward way.

Let's illustrate this by making a 1-bit full adder out of a 3:8 decoder and some OR gates.

Decoders with enable inputs

(This topic is **not** covered in Section 2.8 in the textbook.)

A common variation of the decoder is a decoder with an **enable** input:



This is just like the 2:4 decoder we looked at earlier, except that when EN $= 0$, **all** outputs are 0. *Let's write a truth table for this.*

Using small decoders with enable inputs to make bigger decoders

Let's build a 3:8 decoder using an inverter and two 2:4 decoder-with-enable circuits.

Let's build a 4:16 decoder-with-enable using some 2:4 decoder-with-enable circuits.

Outline of Slide Set 6

Introduction to timing of combinational logic

The definition of **combinational logic** is that the outputs of a combinational element depend only on the **current** values of its inputs.

In reality, combinational elements have very, very short "reaction times". Changes in inputs trigger changes to outputs that are **almost but not quite instant**.

Delays in combinational logic can set important limits on how fast digital systems can operate.

We're about to study some simple methods for estimating overall delays when complex combinational elements are built from simpler combinational elements.

How short is a picosecond?

$1\,\text{ps} = 1 \times 10^{-12}\,\text{s}$.
Every second contains $10^{12} = 1$ trillion picoseconds.

For simple logic gates in today's integrated circuits, propagation delays—reaction times to changes in input values—are typically tens of picoseconds.

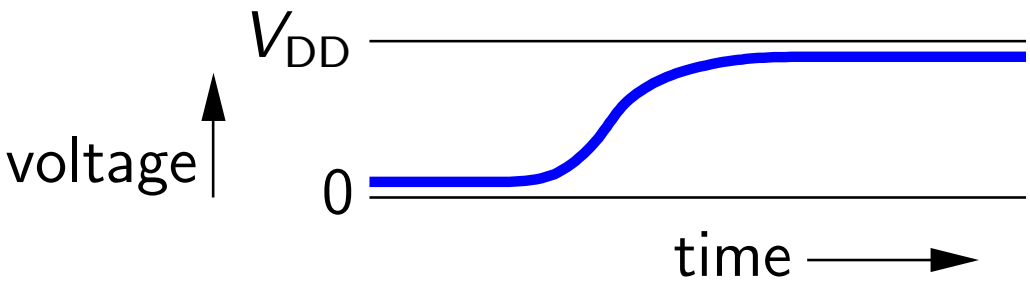
An Olympic sprinter is considered to have false-started if she or he has reacted to the starting gun in less than 0.100 seconds.

Let's compare logic gates and humans using the same units for time . . .

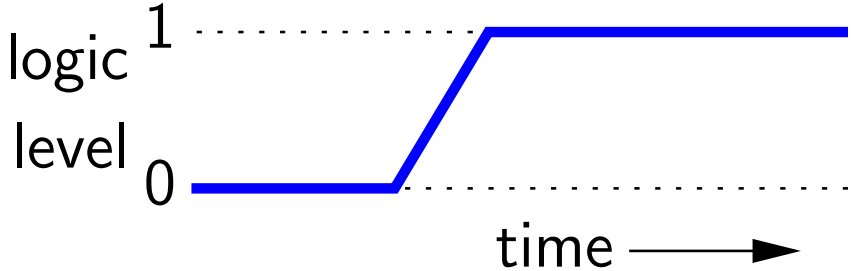
Typical AND gate reaction time:	60 ps.
Very fast human reaction time:	100,000,000,000 ps.

Sketching logic levels as functions of time

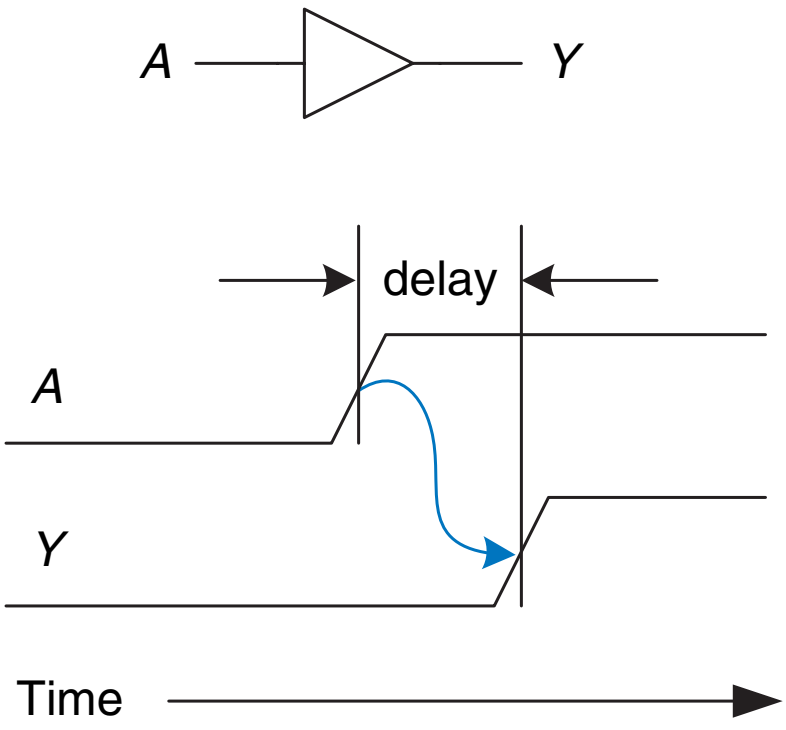
When a logic signal changes value, voltage as a function of time will follow a curve dictated by some complex physics:



In making sketches to illustrate digital circuit timing, the exact shapes of voltage/time curves are not important, and this style of drawing is often used:



Delay in a simple gate



By convention, delay is measured

- **from** the time that the **input** is **halfway** between LOW and HIGH;
- **to** the time that the **output** is **halfway** between LOW and HIGH.

Image is Figure 2.66 from Harris D. M. and Harris S. L., *Digital Design and Computer Architecture, 2nd ed.*, © 2013, Elsevier, Inc.

A combinational element will exhibit a range of delays

There is no single "reaction time" for a given combinational element. Here are some of the many reasons for this:

- HIGH-to-LOW output transitions may be faster or slower than LOW-to-HIGH transitions, depending on the design of the element.
- Circuits tend to get slower as they get warmer.
- Supposedly identical gates may perform differently due to due to variations in manufacturing.
- In elements with multiple output bits, some output bits may switch faster than others.

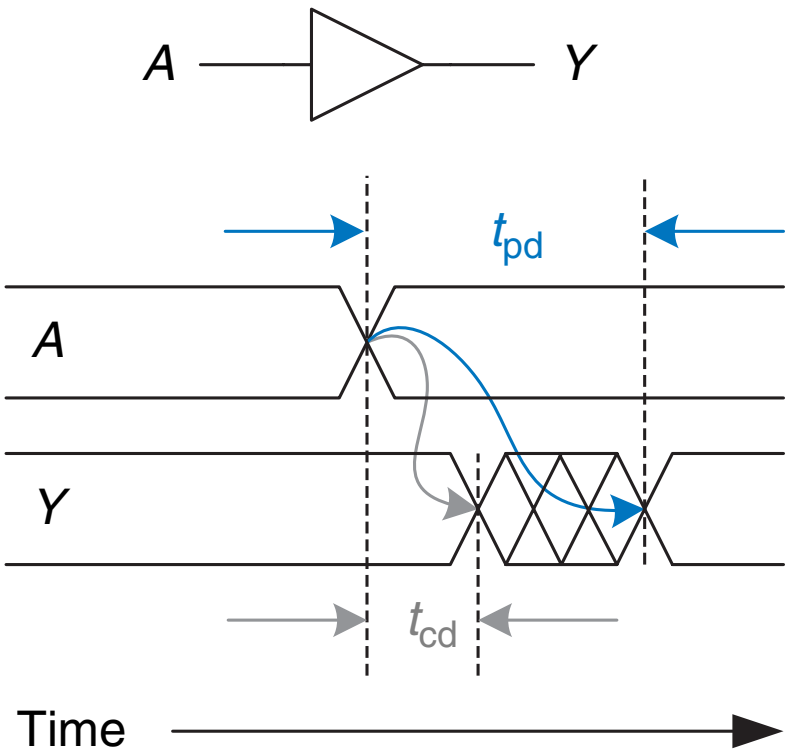
Outline of Slide Set 6

Propagation and contamination delays

Because any combinational element exhibits a **range** of delays, delay characteristics of an element are often described by **two numbers**:

- $t_{\mathbf{pd}}$, the *propagation delay*. This is the **maximum** possible delay under the expected operating conditions for the element.
- $t_{\mathbf{cd}}$, the *contamination delay*. This is the **minimum** possible delay under the expected operating conditions for the element.

t_{pd} and t_{cd} illustrated in a timing diagram



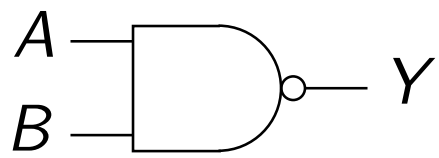
This is a relatively simple timing diagram, but there is still a lot going on here!

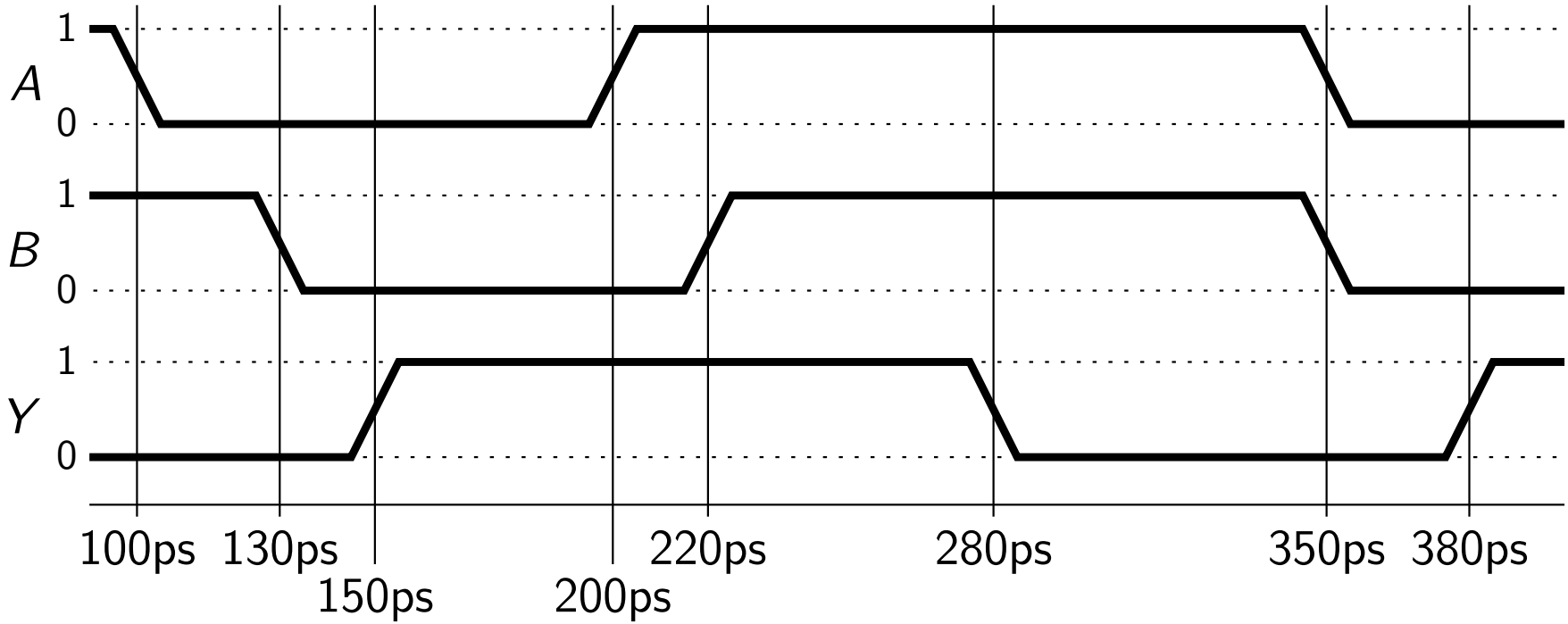
Let's make some notes about how to read this diagram.

Image is Figure 2.67 from Harris D. M. and Harris S. L., *Digital Design and Computer Architecture, 2nd ed.*, © 2013, Elsevier, Inc.

Example propagation and contamination delays

A detailed simulation of a 2-input NAND gate design produces the data shown in the sketch below ...

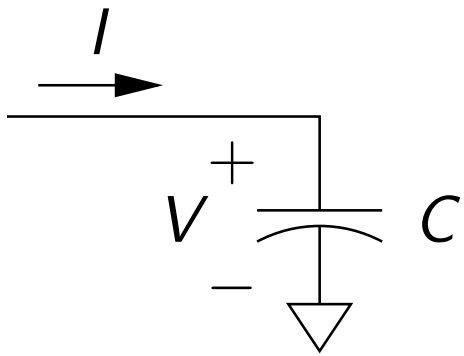




What does the data tell us about t_{pd} and t_{cd} for this NAND gate design?

What are the causes of delays?

One major cause is the fact that a node in a logic circuit acts as a **capacitor**. That puts a limit on the rate of change of voltage at a node.



$$I = C\frac{dV}{dt}, \quad \text{so } \frac{dV}{dt} = \frac{I}{C}.$$

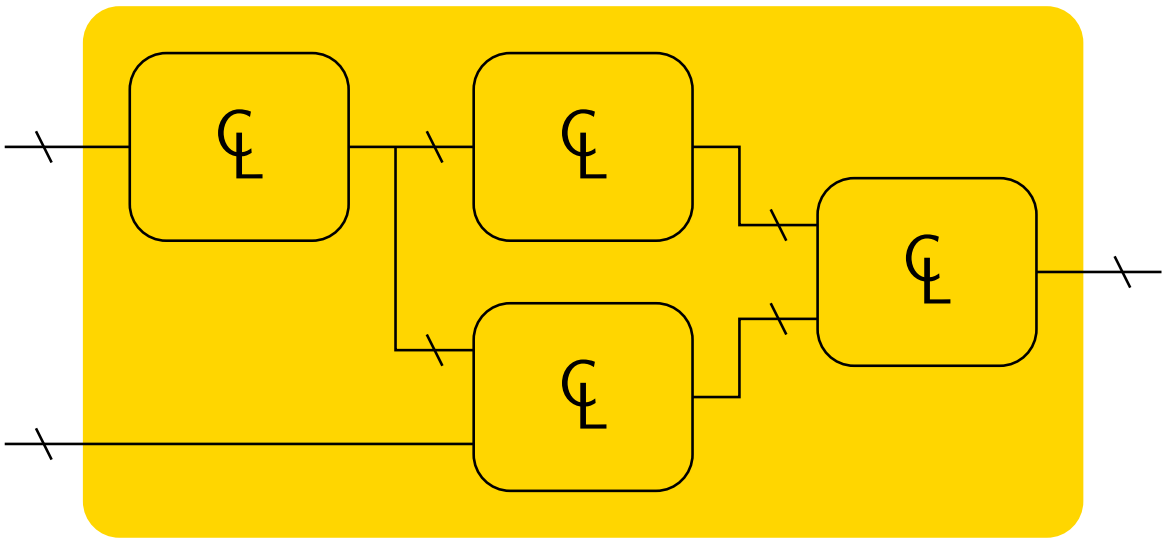
Another important cause is **wire delay**—it takes a small amount of time for a voltage change to get from one end of a wire to the other, even for the tiny wires within integrated circuits.

We won't study the physical causes of delay in ENEL 353. It's an important topic in **ENCM 467** (Digital Electronics).

Outline of Slide Set 6

Overall t_{pd} and t_{cd} calculations

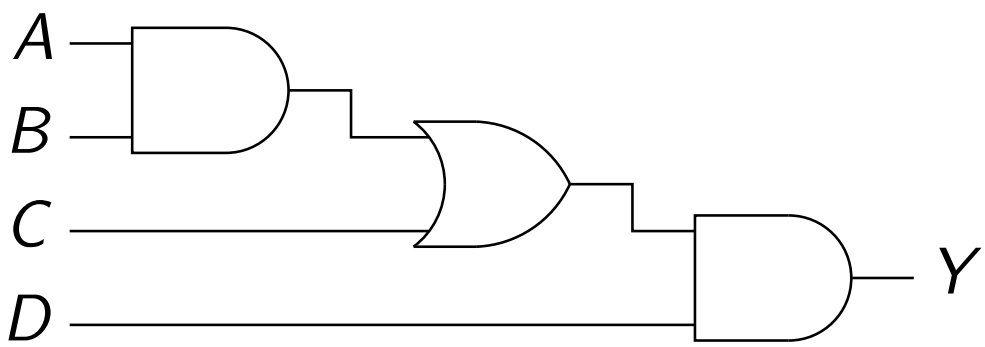
Suppose a combinational system is built by wiring together some combinational elements.



If we have t_{pd} and t_{cd} data for each of the elements, how can we find **overall** values of t_{pd} and t_{cd} for the **system as a whole**?

We'll see that solving this problem involves concepts called the **critical path** and the **short path**.

A simple example of t_{pd} and t_{cd} calculations

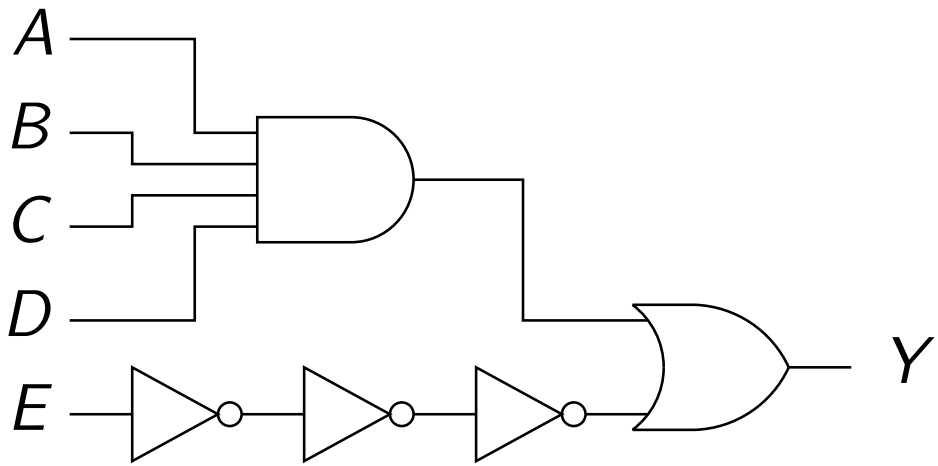


gate	t_{pd}	t_{cd}
AND	50	35
OR	60	45

(Times given in ps.)

What is the critical path for this circuit? What is the short path? What is the overall t_{pd}? What is the overall t_{cd}?

Another simple example of t_{pd} and t_{cd} calculations



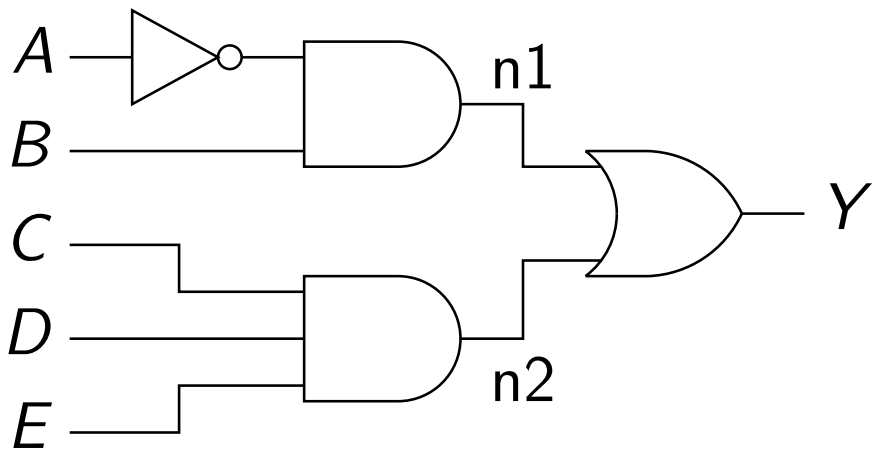
Timing data in ps . . .

gate	t_{pd}	t_{cd}
NOT	15	10
4-input AND	50	25
2-input OR	30	22

What is the critical path for this circuit? What is the short path? What are the overall t_{pd} and t_{cd}?

What important point is being made in this example?

A third simple example of t_{pd} and t_{cd} calculations



Timing data in ps . . .

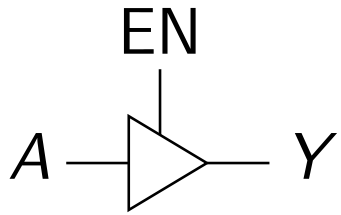
gate	t_{pd}	t_{cd}
NOT	15	10
2-input AND	31	25
3-input AND	40	30
2-input OR	42	32

What are the overall t_{pd} and t_{cd}?

Timing data for textbook 4:1 mux examples

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

Sometimes a gate can respond faster to one of its inputs than to another. The tristate buffer is an example of that.



All of the data is **made up** for the purpose of setting up the mux design examples. (That's also true about other examples in the textbook and in lecture slides.) Real timing depends on dimensions and chemical composition of transistors, layout of gates, and other factors.

For these 4:1 mux designs, find t_{pd} from the S inputs to the output, and also from the D inputs to the output.

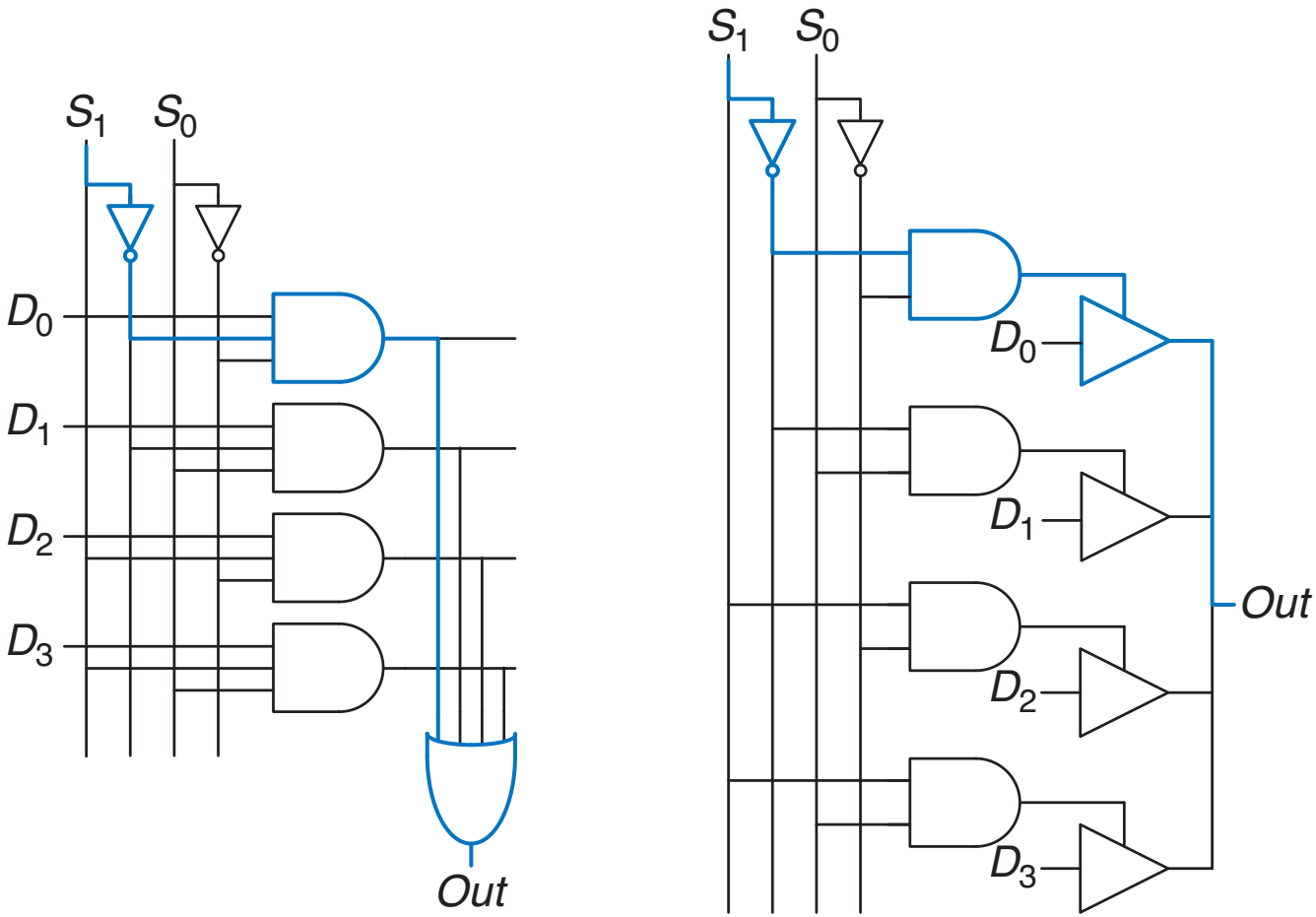
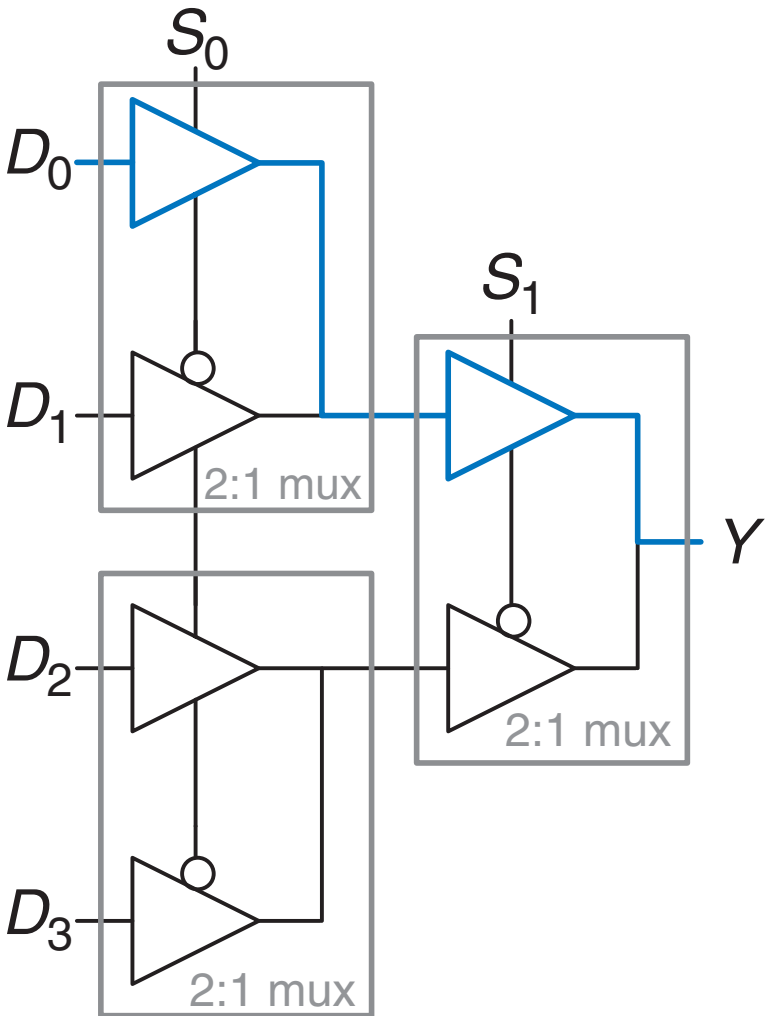


Image is taken from Figure 2.73 from Harris D. M. and Harris S. L., *Digital Design and Computer Architecture, 2nd ed.*, © 2013, Elsevier, Inc.

One more 4:1 mux example



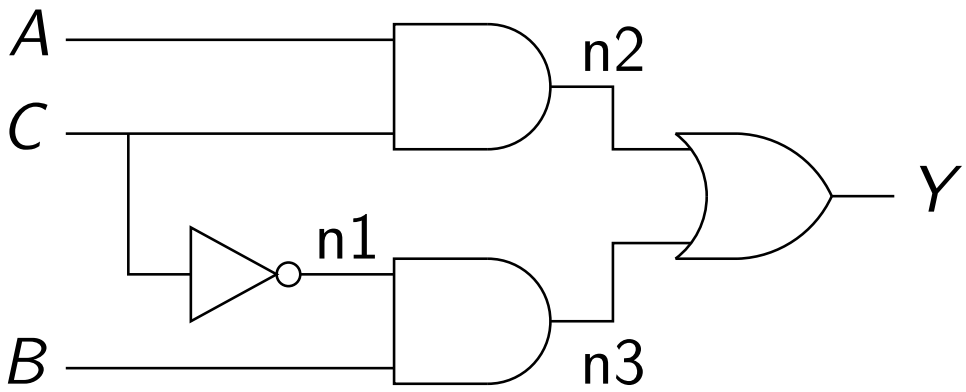
For these 4:1 mux designs, find t_{pd} from the S inputs to Y, and also from the D inputs to Y.

Image is taken from Figure 2.74 from Harris D. M. and Harris S. L., *Digital Design and Computer Architecture, 2nd ed.*, © 2013, Elsevier, Inc.

Note: The textbook gives answers in nanoseconds, but clearly they should be in picoseconds.

Outline of Slide Set 6

Glitches

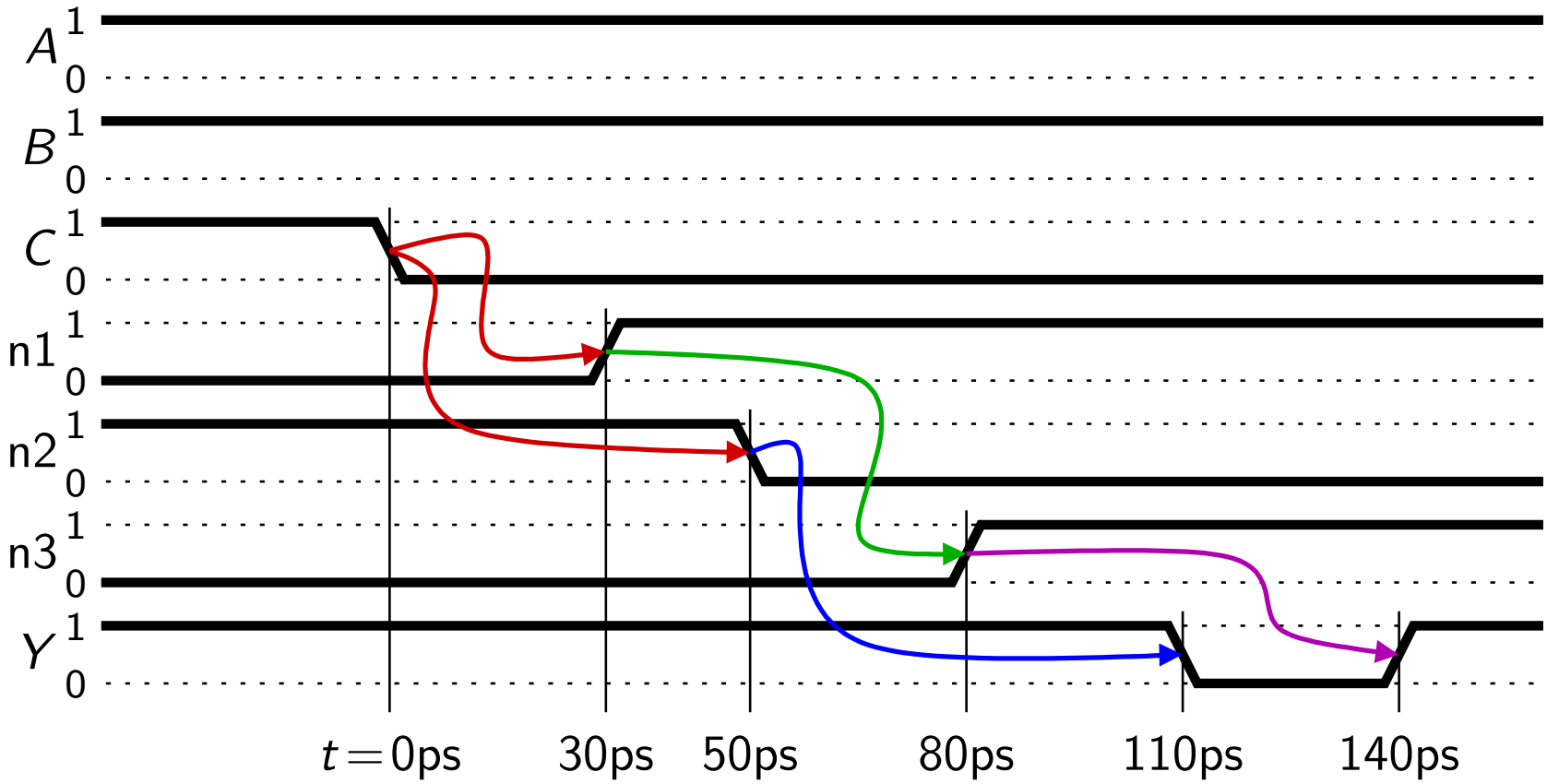


What is Y when $(A,B,C) = (1,1,1)$? What about $(A,B,C) = (1,1,0)$?

Suppose the delays are 30 ps for NOT, 50 ps for AND, and 60 ps for OR.

Let's make a timing diagram to show what happens to Y when (A,B,C) goes from $(1,1,1)$ to $(1,1,0)$.

Timing diagram for glitch example



Let's write down a few remarks about this diagram.

Are glitches bad?

In certain specialized digital design problems, avoidance of glitches in combinational outputs is very important.

Usually, though, glitches are **not** a concern, and what really matters in timing of combinational logic is making sure that **overall propagation delay** is not long.

(Sometimes **low power consumption** is even more important than small propagation delay.)

In Section 2.9.2, Harris & Harris present a method based on K-maps that can sometimes be used to make circuits glitch-free. We're not going to study that in ENEL 353.