Static Logic

Additional Slides

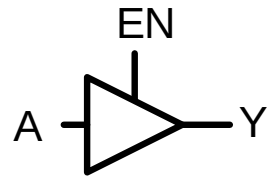
Vishal Saxena
ECE, Boise State University

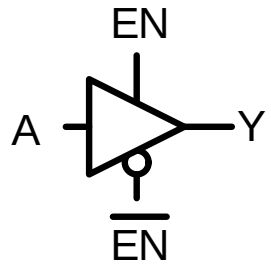
Nov 10, 2010

Multiplexers

- Tristate buffer produces Z when not enabled

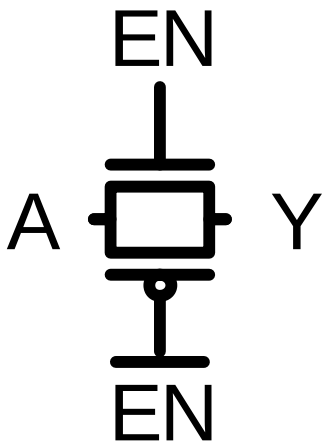
EN	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1





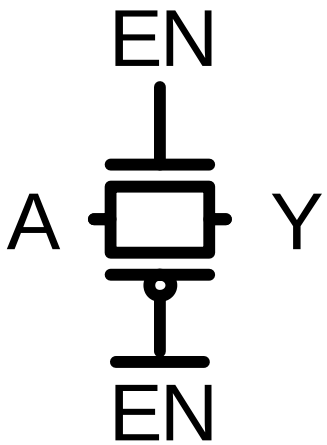
Nonrestoring Tristate

- Transmission gate acts as a tristate buffer
  - only 2 transistors
  - but non-restoring
    - noise on A is passed on to Y



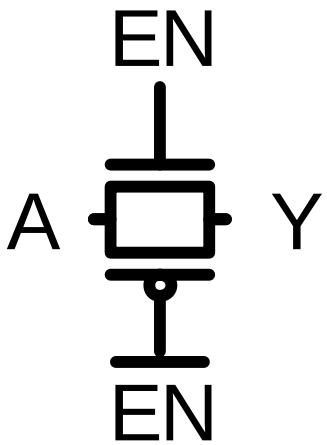
Nonrestoring Tristate

- Transmission gate acts as a tristate buffer
  - only 2 transistors
  - but non-restoring
    - noise on A is passed on to Y



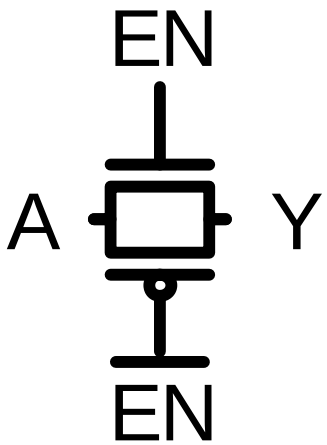
Nonrestoring Tristate

- Transmission gate acts as a tristate buffer
  - only 2 transistors
  - but non-restoring
    - noise on A is passed on to Y



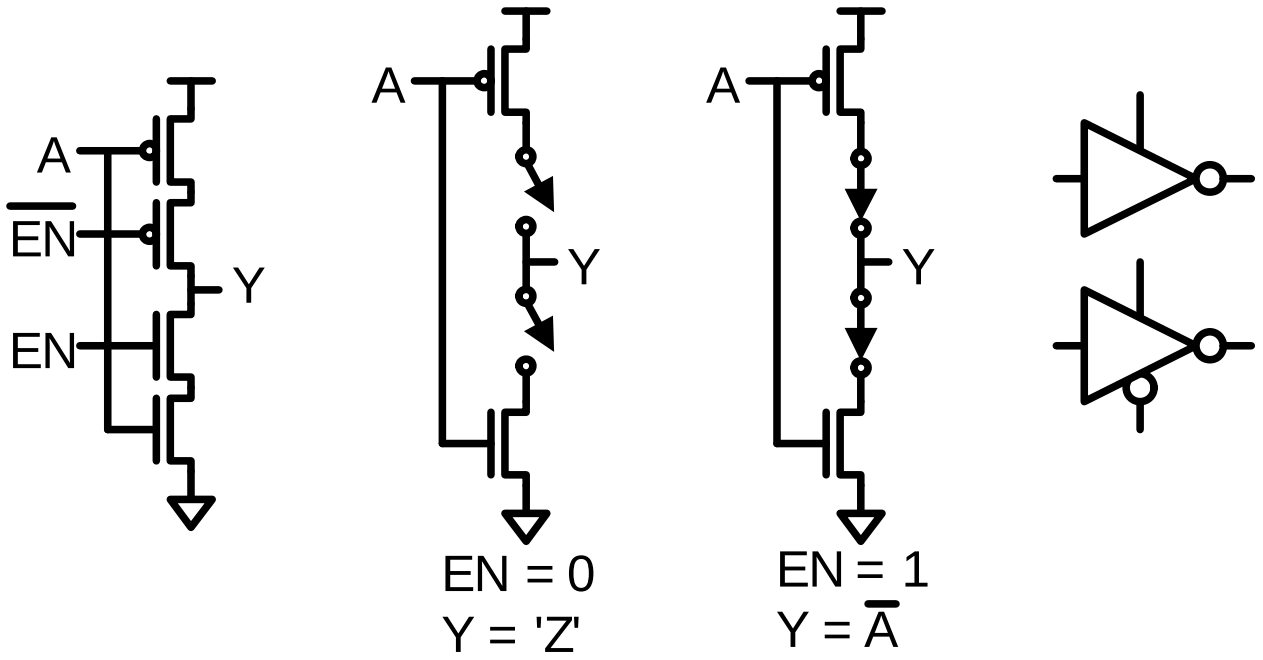
Nonrestoring Tristate

- Transmission gate acts as a tristate buffer
  - only 2 transistors
  - but non-restoring
    - noise on A is passed on to Y



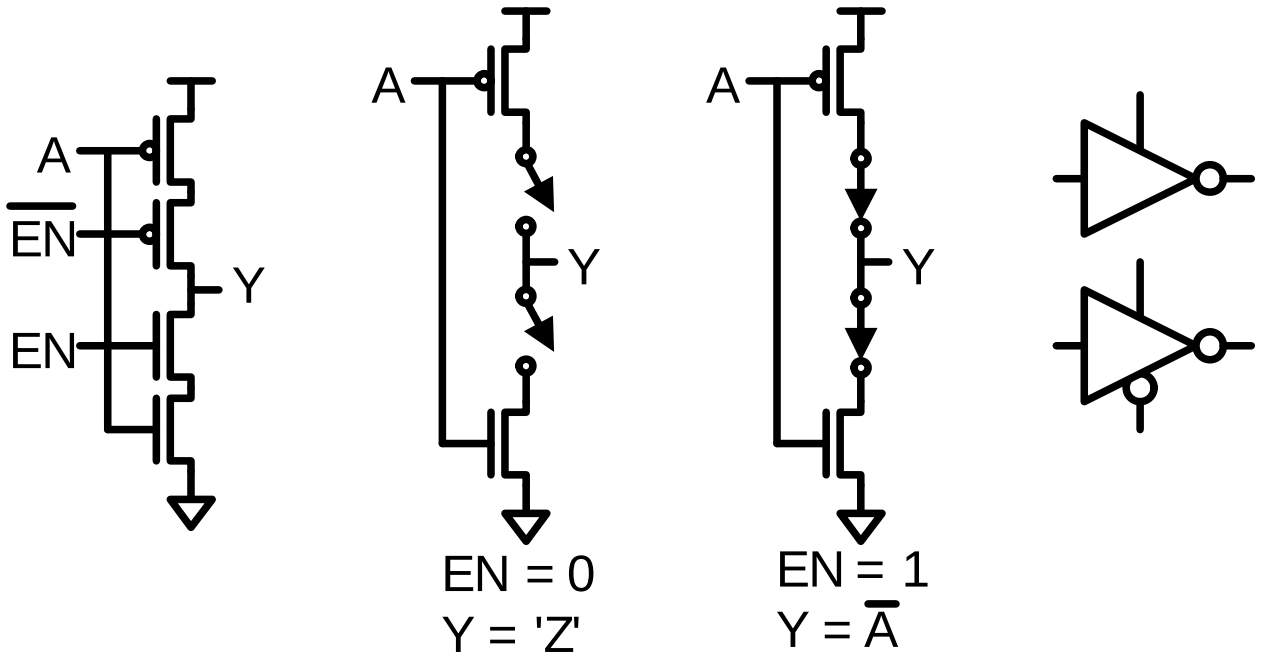
Tristate Inverter

- Tristate inverter produces restored output
  - violates conduction complement rule
    - because we want a Z output



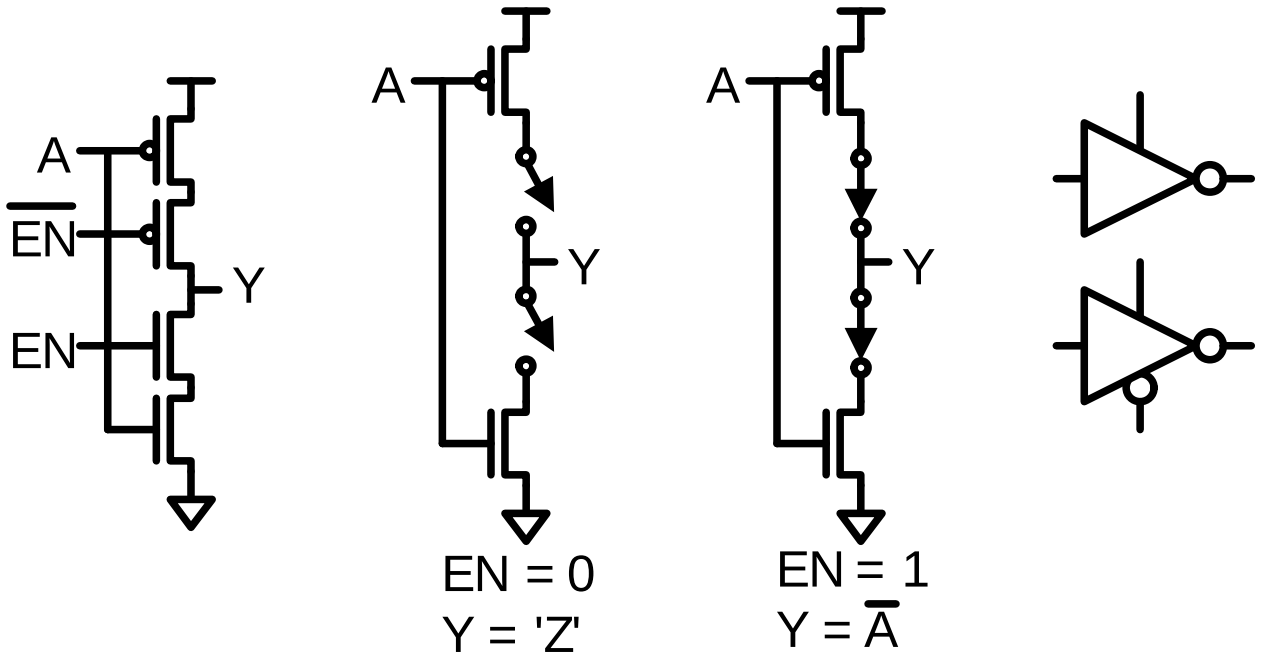
Tristate Inverter

- Tristate inverter produces restored output
  - violates conduction complement rule
    - because we want a Z output



Tristate Inverter

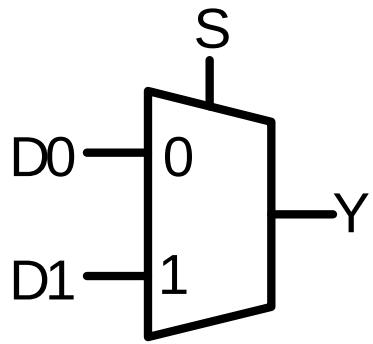
- Tristate inverter produces restored output
  - violates conduction complement rule
    - because we want a Z output



Multiplexers

- 2X1 multiplexer chooses between two inputs

S	D1	D0	Y
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1

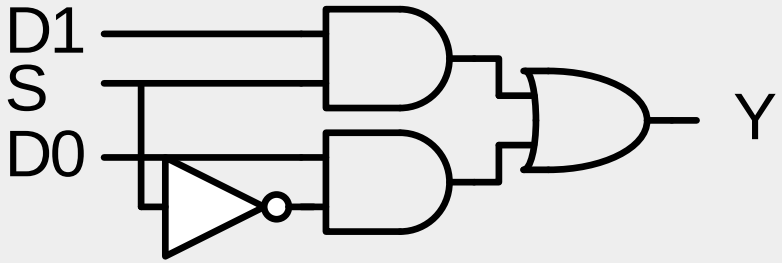


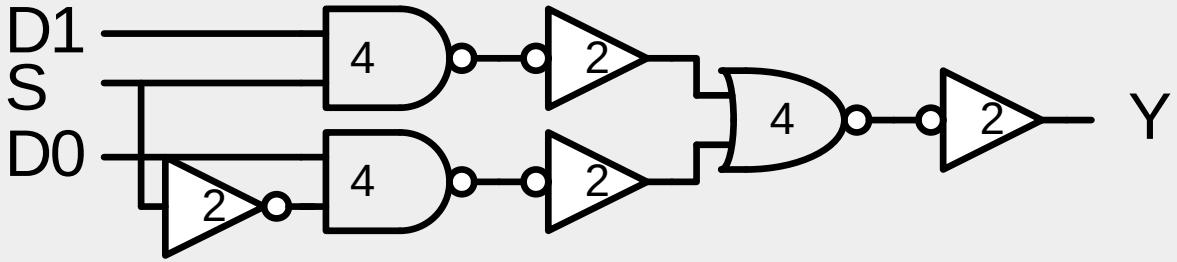
Gate-Level Mux Design

- $Y = SD_1 + \overline{S}D_0$
- How many transistors are needed?

Gate-Level Mux Design

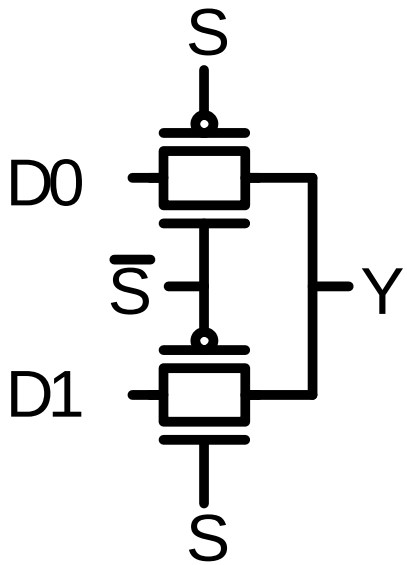
- $Y = SD_1 + \overline{S}D_0$
- How many transistors are needed?





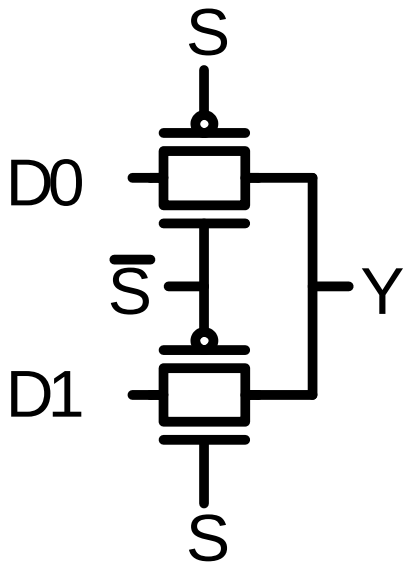
Transmission Gate (TG) Mux

- Nonrestoring mux uses two transmission gates
  - only 4 transistors



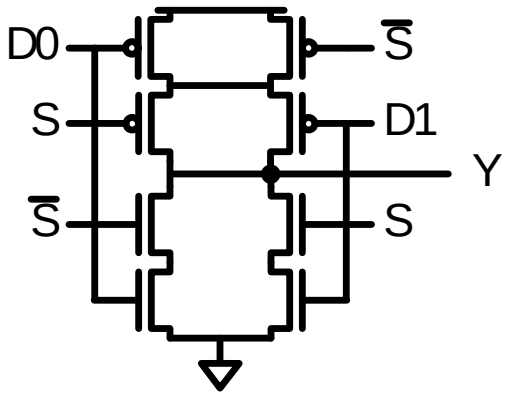
Transmission Gate (TG) Mux

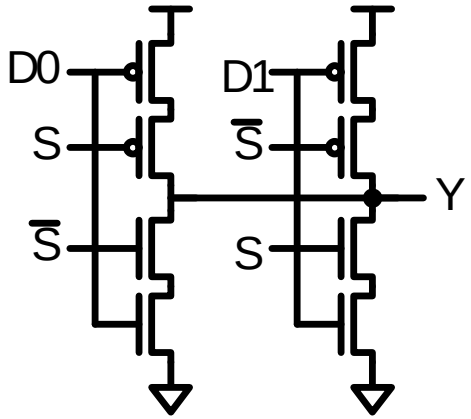
- Nonrestoring mux uses two transmission gates
  - only 4 transistors

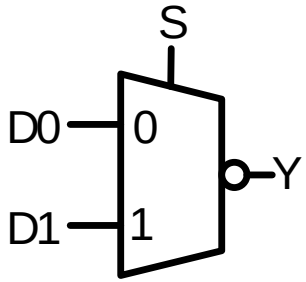


Inverting Mux

- Inverting multiplexer
  - use compound AOI22
  - or pair of tristate inverters
  - essentially the same thing
- Noninverting multiplexer adds an inverter

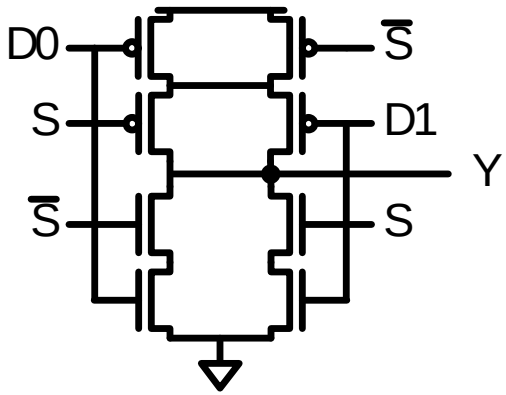


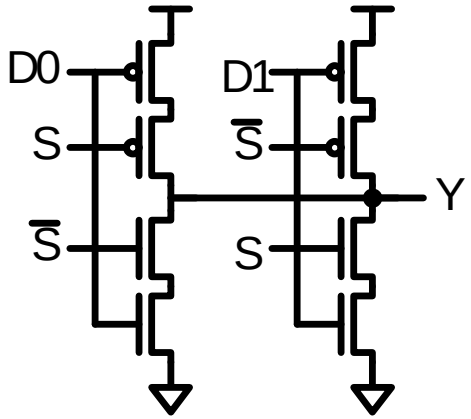


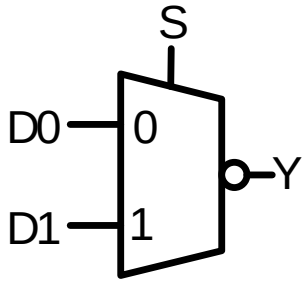


Inverting Mux

- Inverting multiplexer
  - use compound AOI22
  - or pair of tristate inverters
  - essentially the same thing
- Noninverting multiplexer adds an inverter

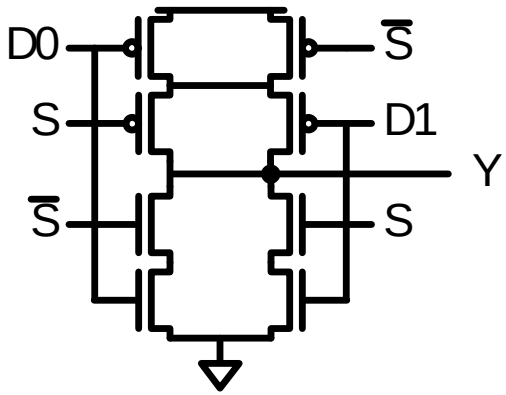


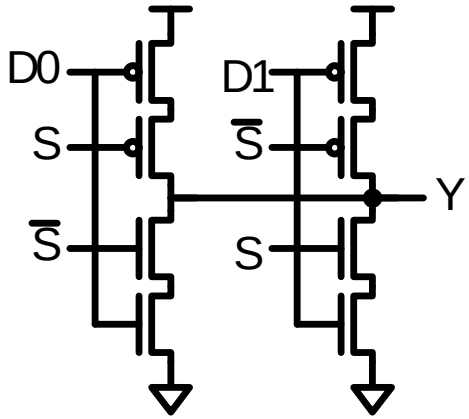


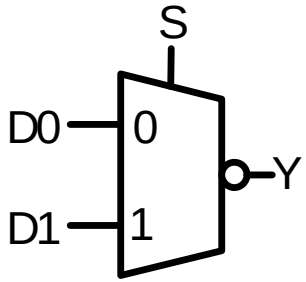


Inverting Mux

- Inverting multiplexer
  - use compound AOI22
  - or pair of tristate inverters
  - essentially the same thing
- Noninverting multiplexer adds an inverter

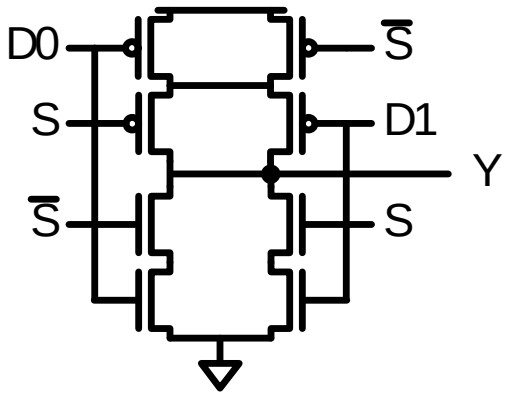


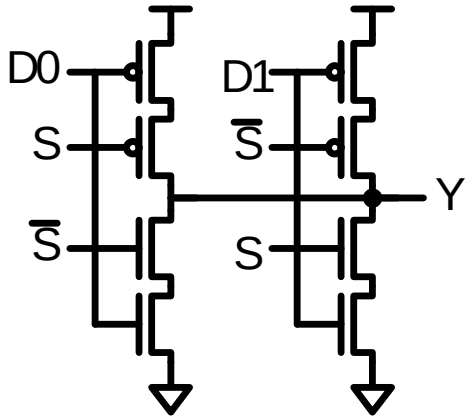


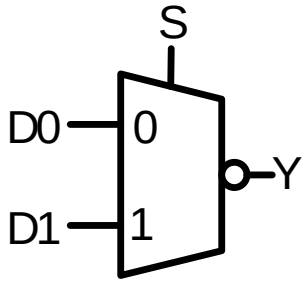


Inverting Mux

- Inverting multiplexer
  - use compound AOI22
  - or pair of tristate inverters
  - essentially the same thing
- Noninverting multiplexer adds an inverter

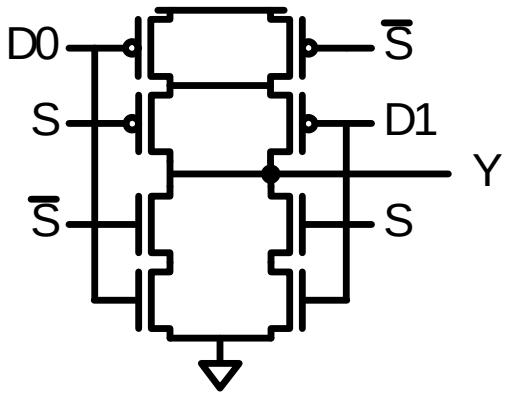


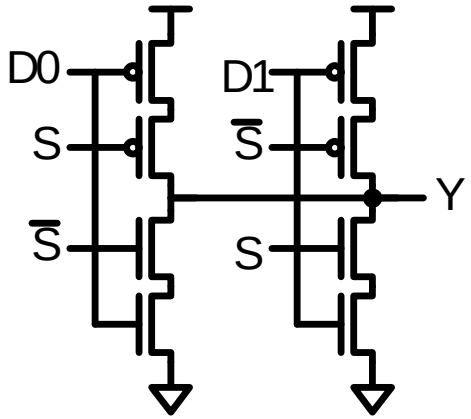


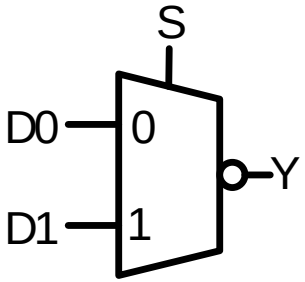


Inverting Mux

- Inverting multiplexer
  - use compound AOI22
  - or pair of tristate inverters
  - essentially the same thing
- Noninverting multiplexer adds an inverter

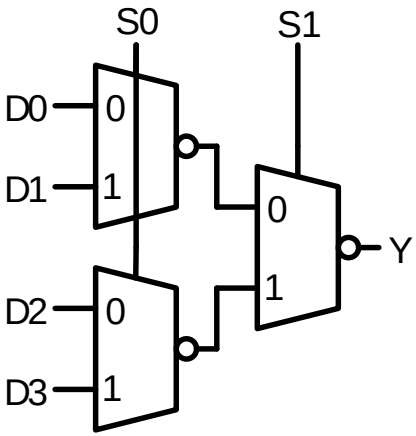


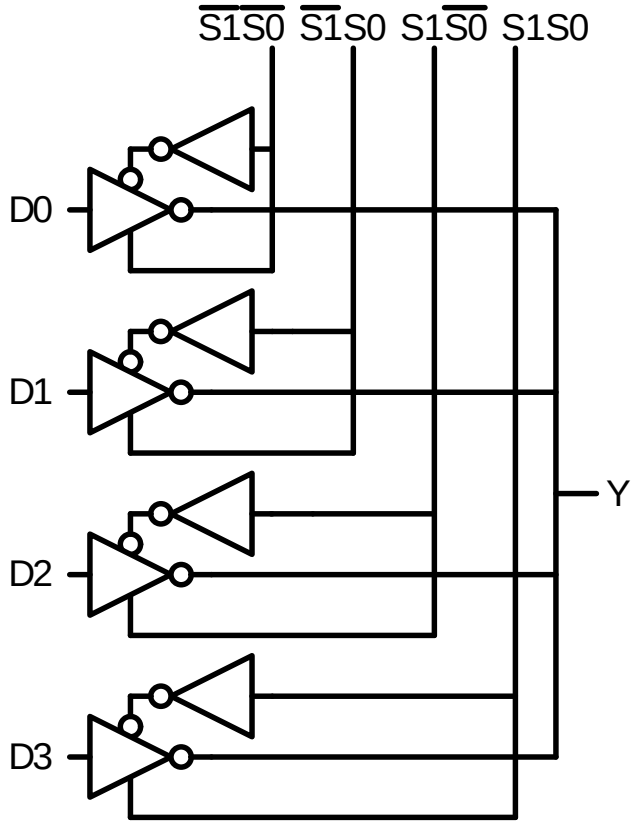




4X1 Multiplexer

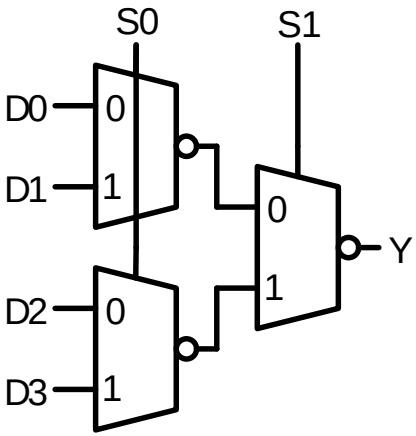
- 4X1 Mux chooses one of 4 inputs using two selects
  - two levels of 2X1 Muxes
  - or four tri-states

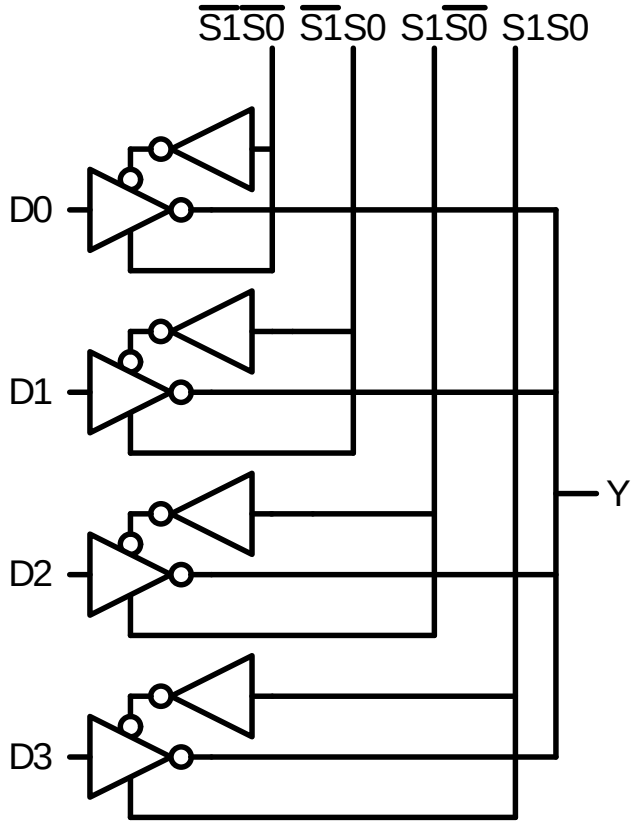




4X1 Multiplexer

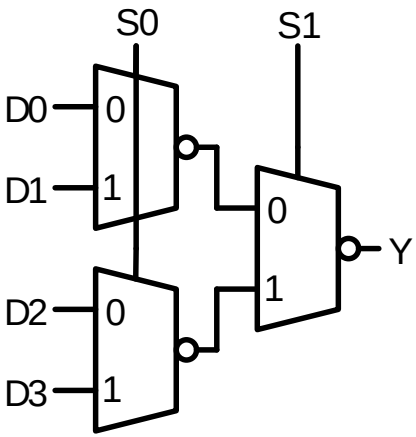
- 4X1 Mux chooses one of 4 inputs using two selects
  - two levels of 2X1 Muxes
  - or four tri-states

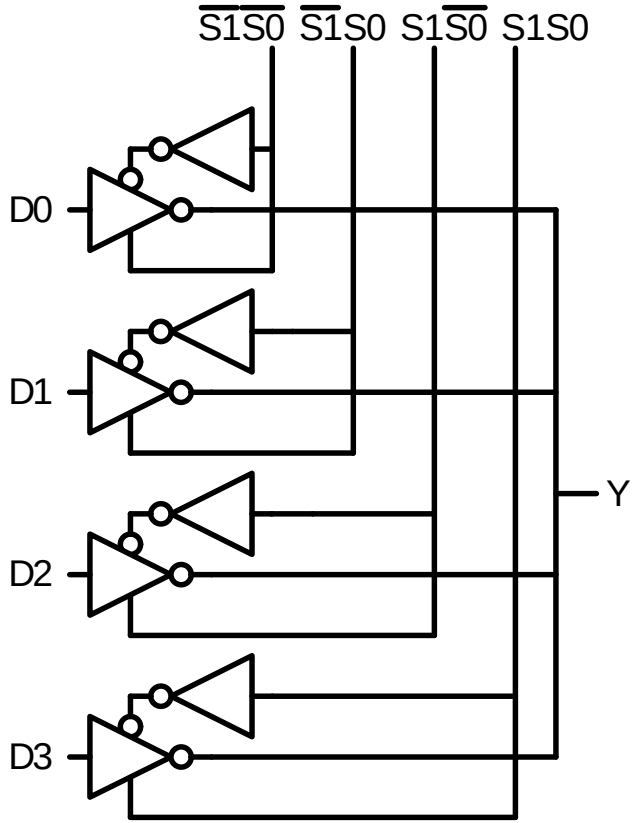




4X1 Multiplexer

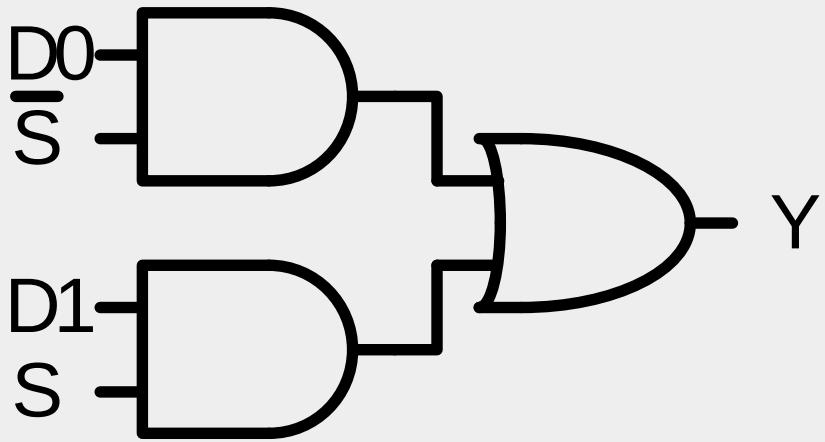
- 4X1 Mux chooses one of 4 inputs using two selects
  - two levels of 2X1 Muxes
  - or four tri-states





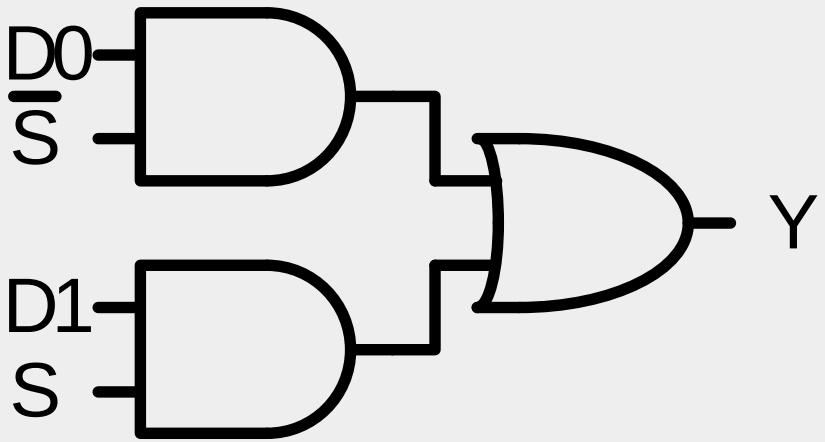
Bubble Pushing: Example 1

- MUX: $Y = \overline{S}D_0 + SD_1$
- 1) Sketch a design using AND, OR, and NOT gates



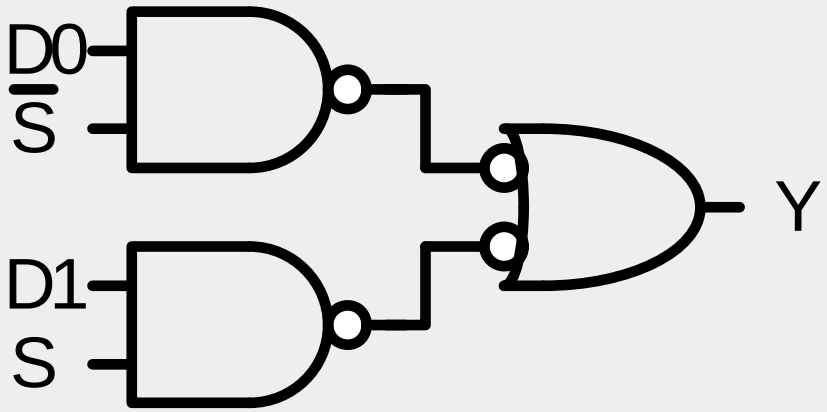
Bubble Pushing: Example 1

- MUX: $Y = \overline{S}D_0 + SD_1$
- 1) Sketch a design using AND, OR, and NOT gates



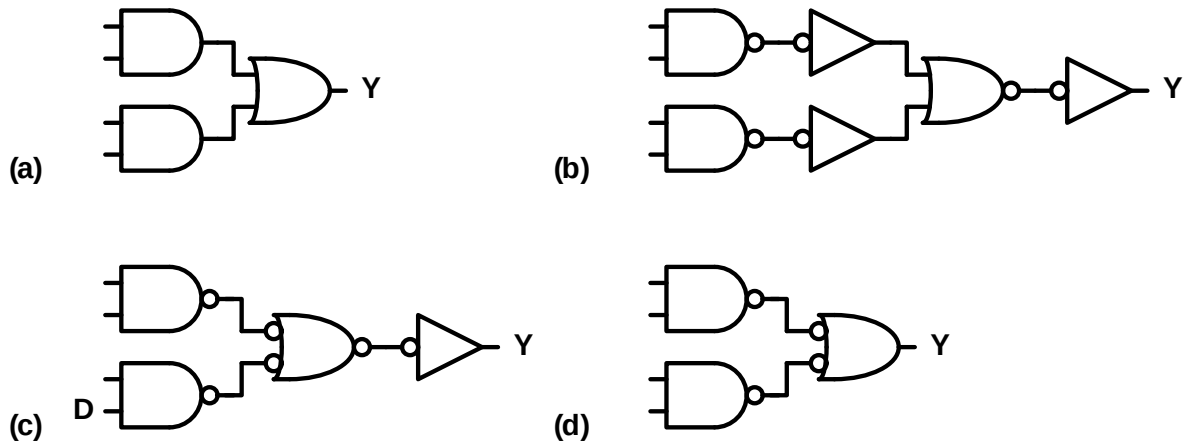
Example 2

- 2) Sketch a design using NAND, NOR, and NOT gates
  - assume $\overline{S}$ is available



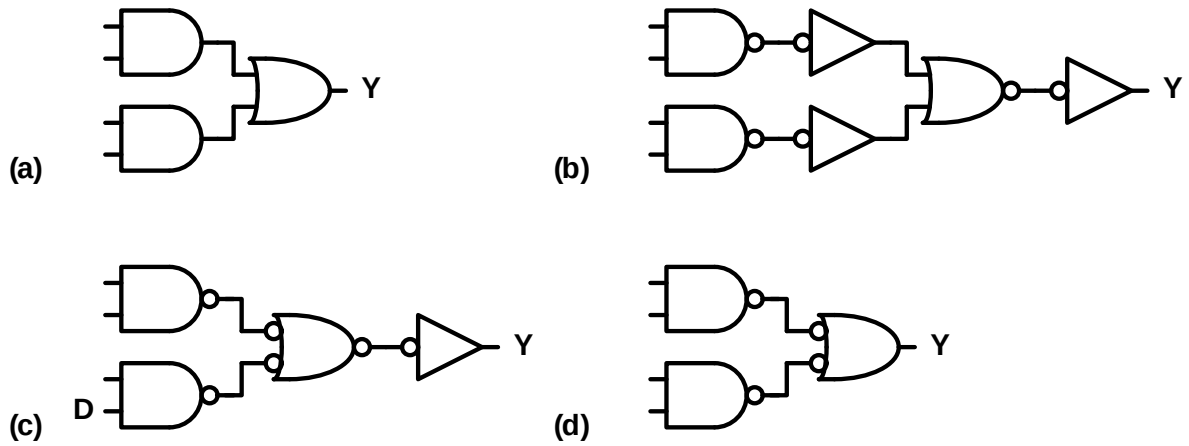
Bubble Pushing

- Start with network of AND / OR gates
- Convert to NAND / NOR + inverters
- Push bubbles around to simplify logic
  - Remember DeMorgan's Law



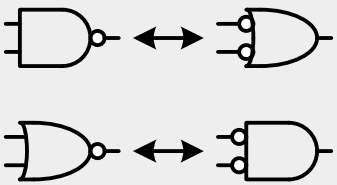
Bubble Pushing

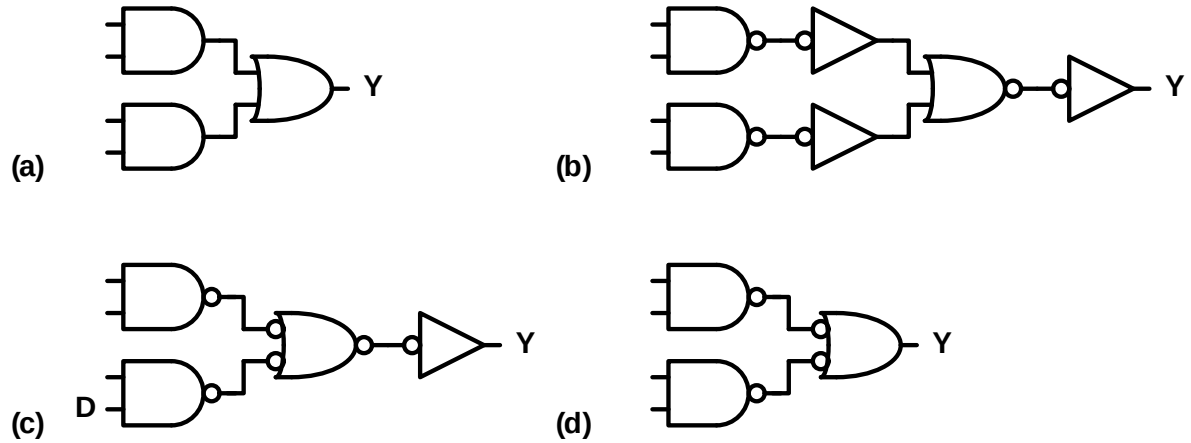
- Start with network of AND / OR gates
- Convert to NAND / NOR + inverters
- Push bubbles around to simplify logic
  - Remember DeMorgan's Law



Bubble Pushing

- Start with network of AND / OR gates
- Convert to NAND / NOR + inverters
- Push bubbles around to simplify logic
  - Remember DeMorgan's Law



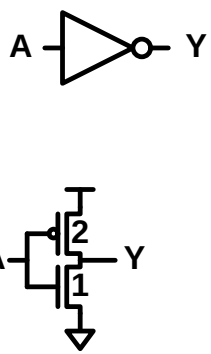


Compound Gates

- Sizing of compound gates

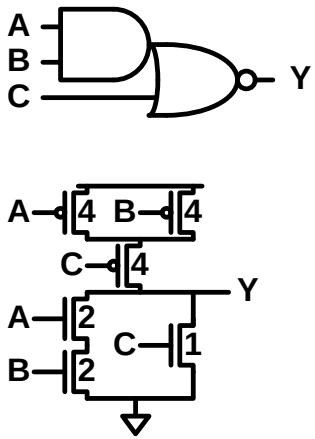
unit inverter

$Y = A$



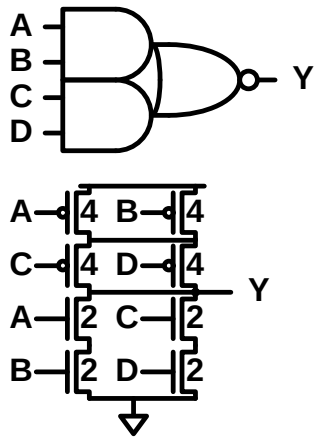
AOI21

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



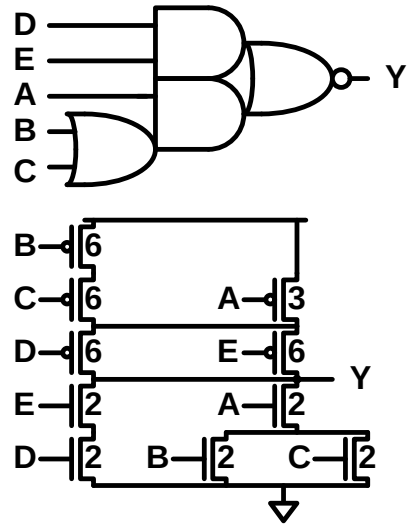
AOI22

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



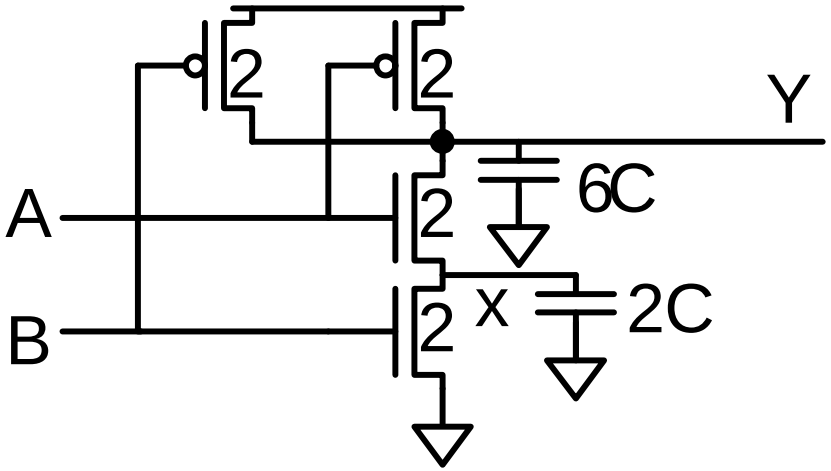
Complex AOI

$Y = \overline{A \bullet (B + C) + D \bullet E}$



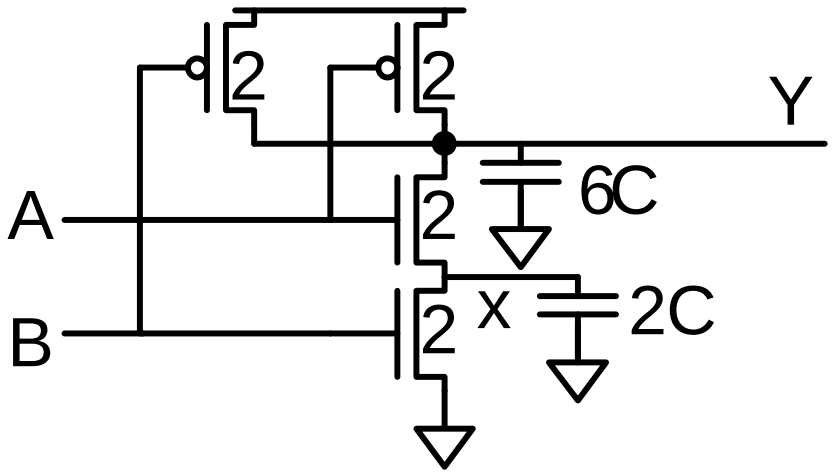
Input Order

- Calculate parasitic delay for Y falling
  - If A arrives latest?
    - $6RC = 2\tau$
  - If B arrives latest?
    - 2.33τ



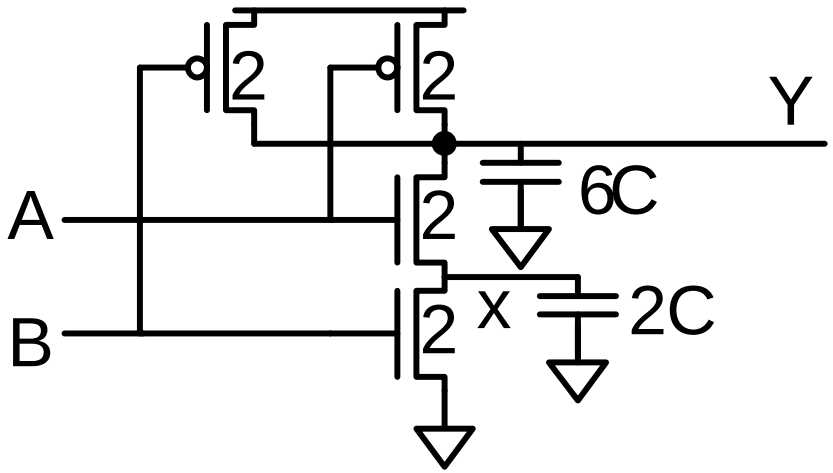
Input Order

- Calculate parasitic delay for Y falling
  - If A arrives latest?
    - $6RC = 2\tau$
  - If B arrives latest?
    - 2.33τ



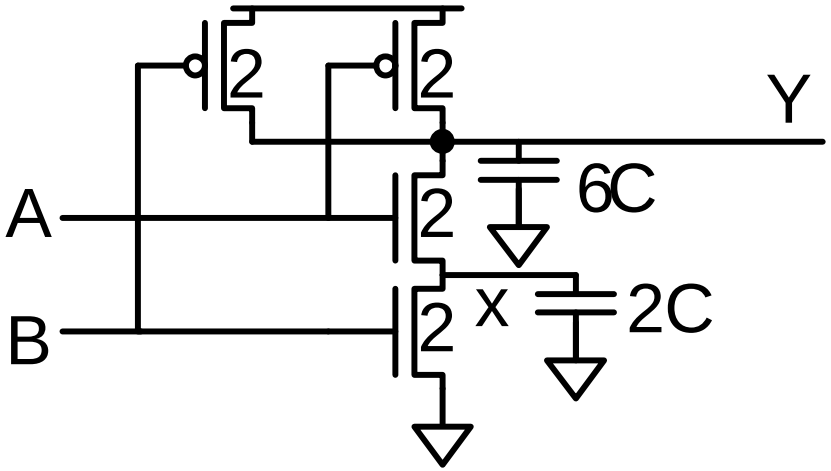
Input Order

- Calculate parasitic delay for Y falling
  - If A arrives latest?
    - $6RC = 2\tau$
  - If B arrives latest?
    - 2.33τ



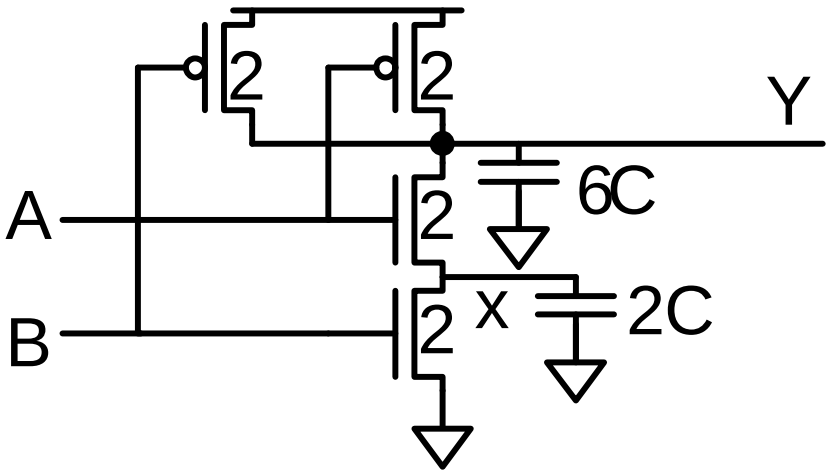
Input Order

- Calculate parasitic delay for Y falling
  - If A arrives latest?
    - $6RC = 2\tau$
  - If B arrives latest?
    - 2.33τ



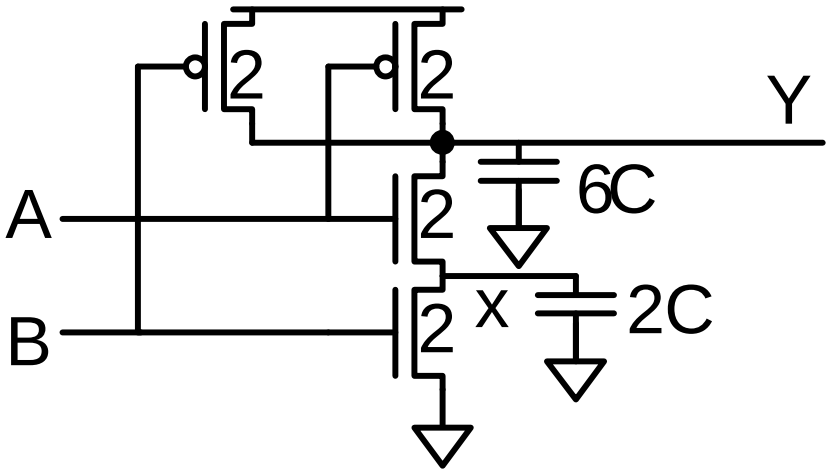
Input Order

- Calculate parasitic delay for Y falling
  - If A arrives latest?
    - $6RC = 2\tau$
  - If B arrives latest?
    - 2.33τ



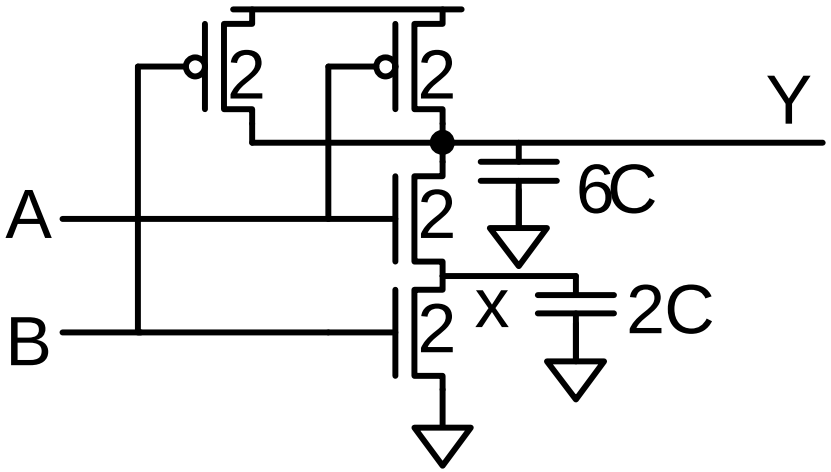
Inner and Outer Inputs

- Inner input is closest to output (A)
- Outer input is closest to rail (B)
- If input arrival time is known
  - Connect latest input to inner terminal



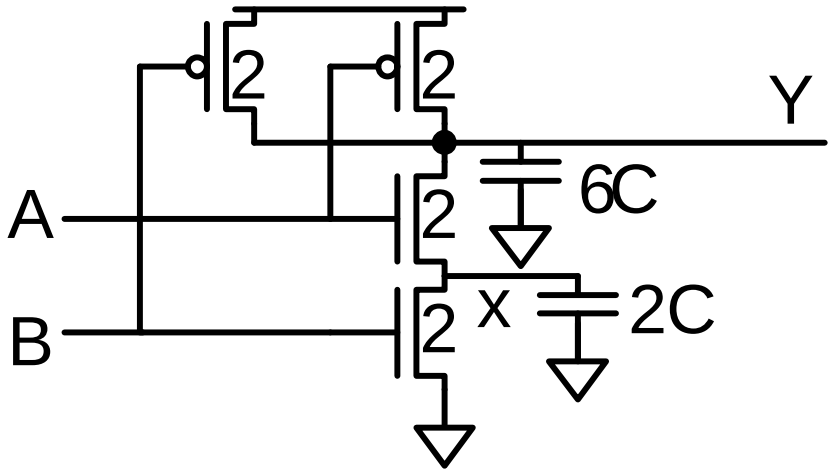
Inner and Outer Inputs

- Inner input is closest to output (A)
- Outer input is closest to rail (B)
- If input arrival time is known
  - Connect latest input to inner terminal



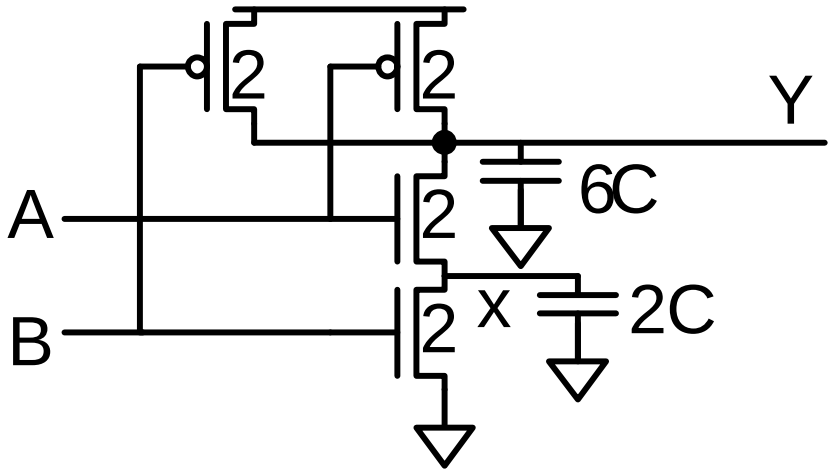
Inner and Outer Inputs

- Inner input is closest to output (A)
- Outer input is closest to rail (B)
- If input arrival time is known
  - Connect latest input to inner terminal



Inner and Outer Inputs

- Inner input is closest to output (A)
- Outer input is closest to rail (B)
- If input arrival time is known
  - Connect latest input to inner terminal



Observations

- For speed:
  - NAND vs NOR
  - Many simple stages vs. fewer high fan-in stages
  - Latest-arriving input
- For area and power:
  - Many simple stages vs. fewer high fan-in stages

Observations

- For speed:
  - NAND vs NOR
  - Many simple stages vs. fewer high fan-in stages
  - Latest-arriving input
- For area and power:
  - Many simple stages vs. fewer high fan-in stages

Observations

- For speed:
  - NAND vs NOR
  - Many simple stages vs. fewer high fan-in stages
  - Latest-arriving input
- For area and power:
  - Many simple stages vs. fewer high fan-in stages

Observations

- For speed:
  - NAND vs NOR
  - Many simple stages vs. fewer high fan-in stages
  - Latest-arriving input
- For area and power:
  - Many simple stages vs. fewer high fan-in stages

Observations

- For speed:
  - NAND vs NOR
  - Many simple stages vs. fewer high fan-in stages
  - Latest-arriving input
- For area and power:
  - Many simple stages vs. fewer high fan-in stages

Observations

- For speed:
  - NAND vs NOR
  - Many simple stages vs. fewer high fan-in stages
  - Latest-arriving input
- For area and power:
  - Many simple stages vs. fewer high fan-in stages

References I

N. Weste and D. Harris, CMOS VLSI Design: A circuits and systems perspective, 4th Ed., Addison-Wesley, 2010.

R. J. Baker, CMOS Circuit Design, Layout and Simulation, revised 2nd Edition, Wiley-IEEE, 2008.