Propogation Delay and State

Pick up
A HANDOUT

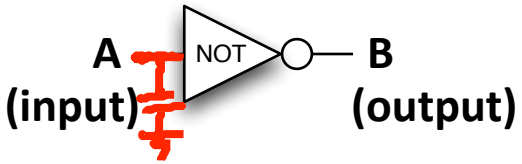
Today's lecture

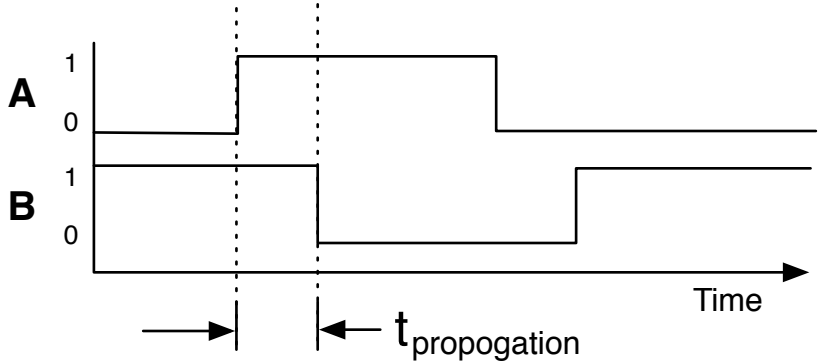
- **Propogation Delay**
  - Timing diagrams.
  - Delay of ALU32
- **Storing State**
  - SR Latch
- **Synchronous Design**
  - Clocks
  - D Flip Flops

Propogation Delay

- **Real gates don't switch instantaneously**
  - There is a latency between when the input changes and the output changes.
  - We call this latency the propagation delay

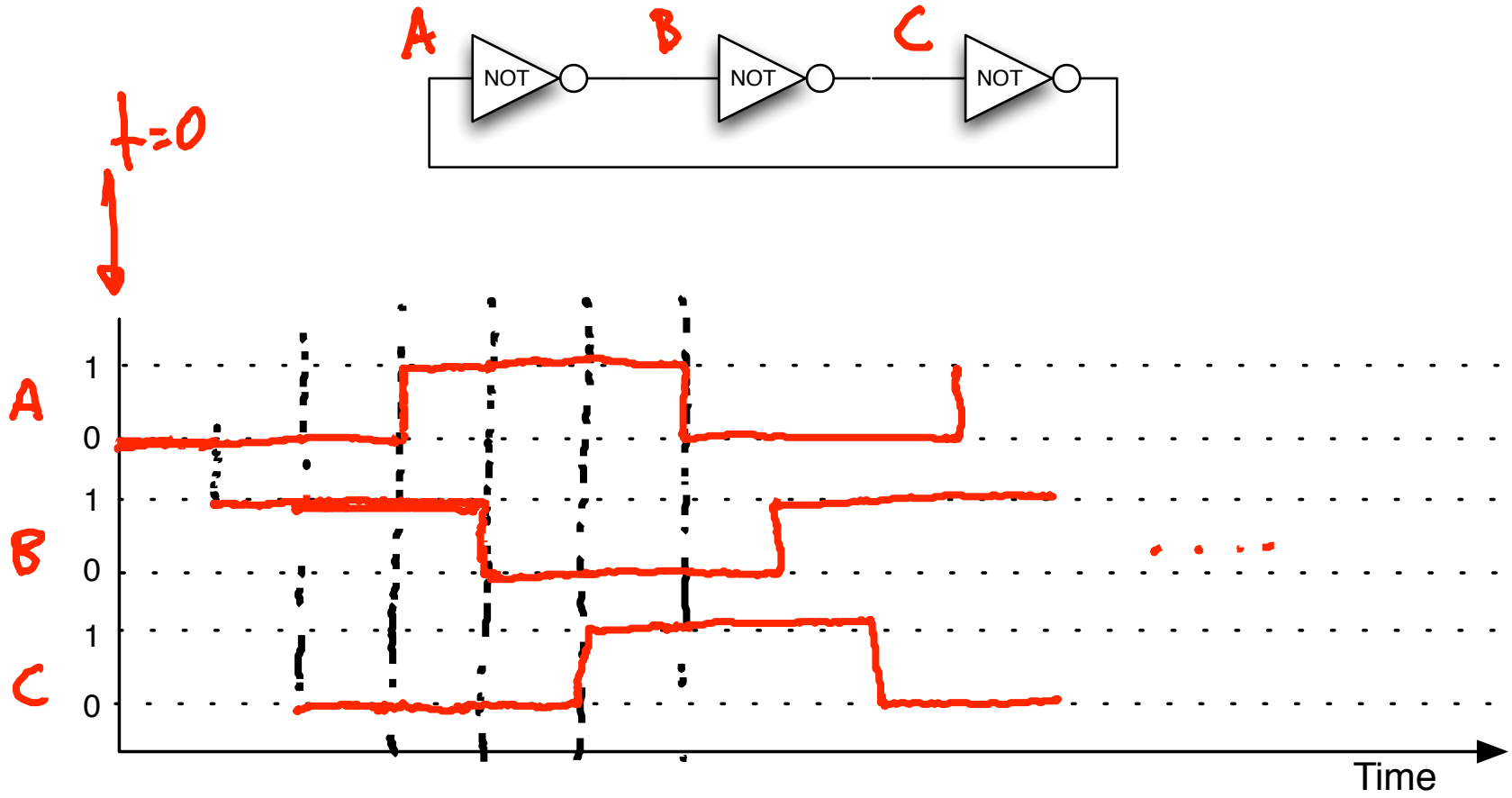
- **Consider an inverter:**



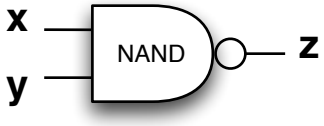


Ring Oscillator

- What happens when you connect an odd number of inverters in a circle?



Timing analysis

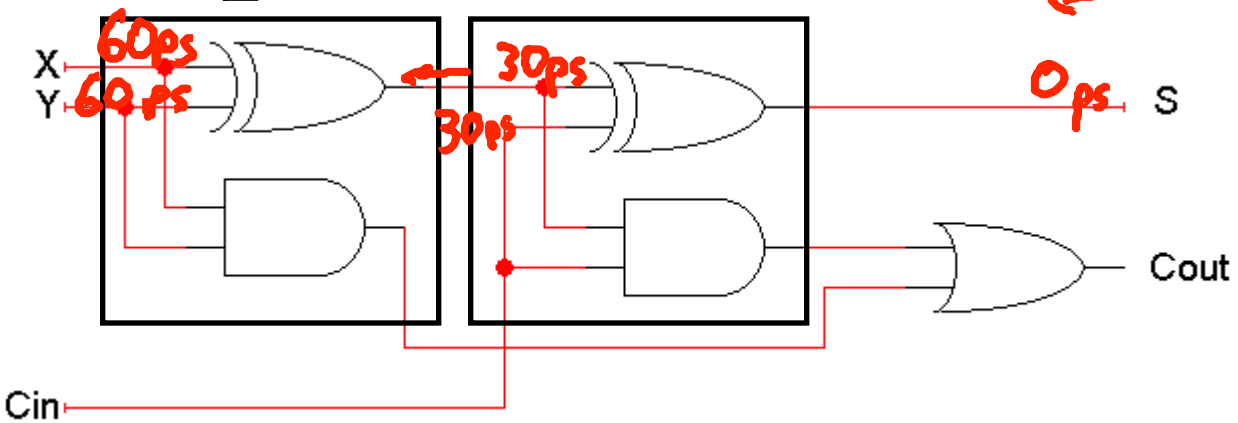


- **In reality, timing is very complicated**
  - The delay from **x** to **z** can be different on an 0➔1 transition than it is for a 1➔0 transition.
  - The delay from **x** to **z** can differ from the delay from **y** to **z**.
  - The number of gates connected to the same output (the fanout), the longer it will take to switch. (capacitance)
  - Long wires between gates slow things down as well. (resistance)
- **In this class, we'll use simplifying assumptions:**
  - **Delay is a constant from any input to the output.**
  - **We'll ignore fanout and wire delay**

Analyzing propagation delay of circuits

- **Assume:**
  - Inverter: 20ps delay
  - 2-input gate: 30ps delay
  - 4-input gate: 50ps delay

- **Find longest paths from input to output**

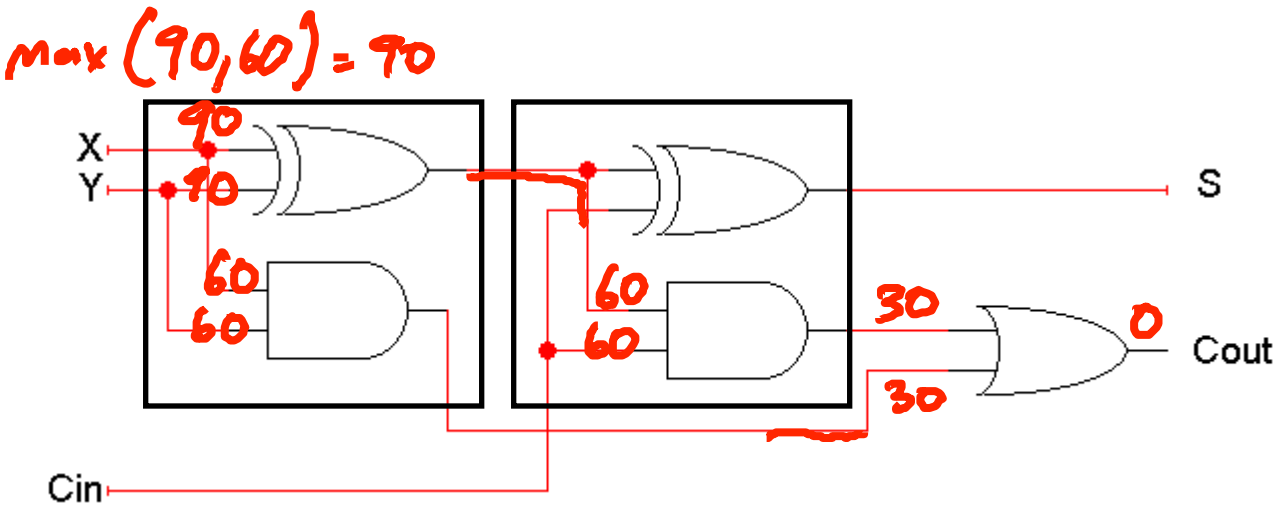


In	Out	Delay
X	S	60ps
Y	S	60ps
Cin	S	30ps
X	Cout	
Y	Cout	
Cin	Cout	

Analyzing propagation delay

- **What is the propogation delay from X to Cout?**
- **Assume:**
  - Inverter: 20ps delay
  - 2-input gate: 30ps delay

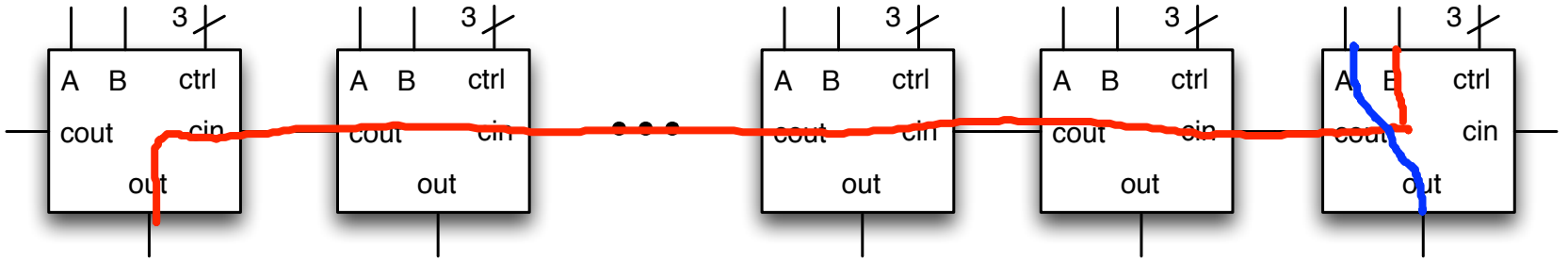
A: 30ps
B: 60ps
C: 90ps
D: 120ps



In	Out	Delay
X	S	60ps
Y	S	60ps
Cin	S	30ps
X	Cout	90ps
Y	Cout	90ps
Cin	Cout	60ps

Computing the delay of alu32

- What is likely to be the longest path through this circuit?
  - From any data input (A, B) bit to any output bit

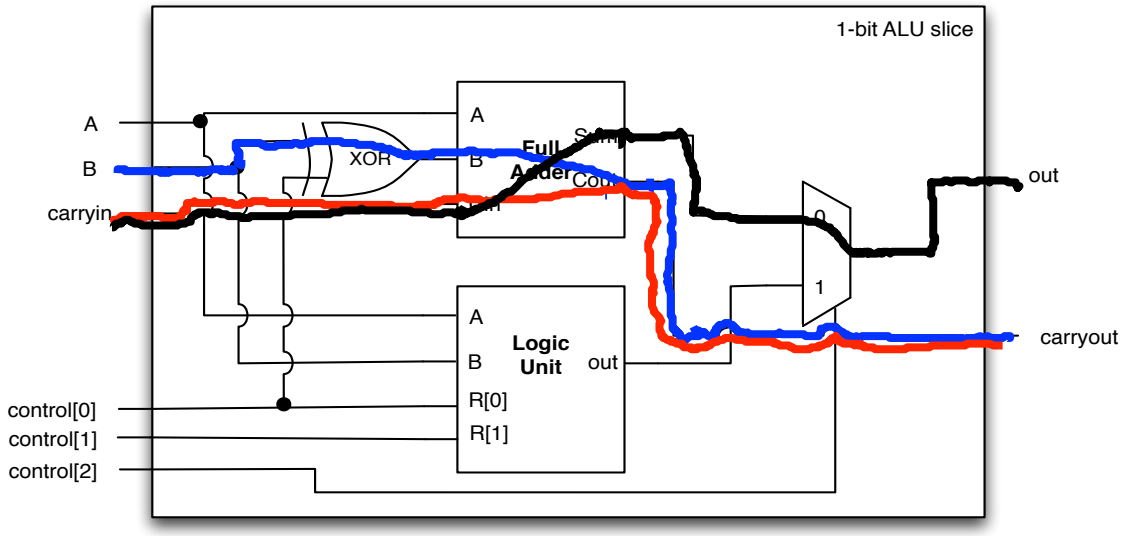


$T_{cin \to out}$ +

$30(T_{cin \to cout}) + T_{B \to Cout}$

- How long will it take?

Computing components from ALU1



XOR gate

In	Out	Delay
A,B	out	30ps

Full Adder

In	Out	Delay
A,B	Sum	60ps
Cin	Sum	30ps
A,B	Cout	90ps
Cin	Cout	60ps

Logic Unit

In	Out	Delay
A,B	out	110ps
R	out	10ps

2-to-1 Multiplexor

In	Out	Delay
A,B	out	60ps
R	out	80ps

$t_{PropBCarryout} + 30(t_{PropCarryinCarryout}) + t_{PropCarryinOut} =$

$(30ps + 90ps) + 30(60ps) + (30ps + 60ps) =$

$\sim 2000ps = 2ns$

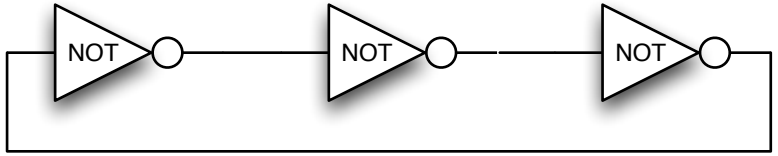
$\frac{1}{2ns} = 500\,MHz$

Thinking about ALU32's delay

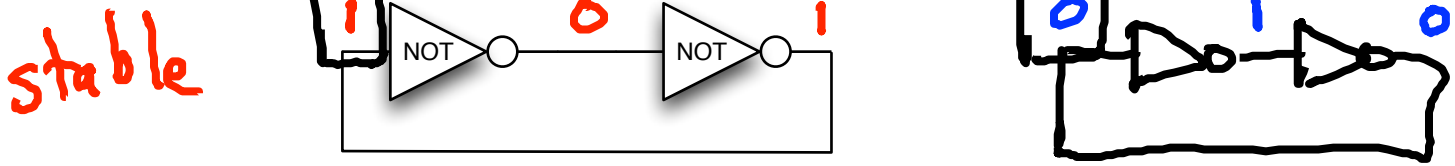
- **That is really bad. Really, really bad.**
- **Processors don't really implement ADDs this way**
  - This is what is called a "ripple carry adder"
  - It has latency ***O(n)*** where ***n*** is the # bits being added
- **There are much smarter ways to handle carries**
  - E.g., "carry lookahead adder" (Google it)
  - Has latency ***O(log$_2$(N))*** and only slightly larger
- **But, we won't cover them in this class.**

Rings of Inverters

- We saw an odd number of inverters creates a ring oscillator:



- What happens if you have an even number of inverters?

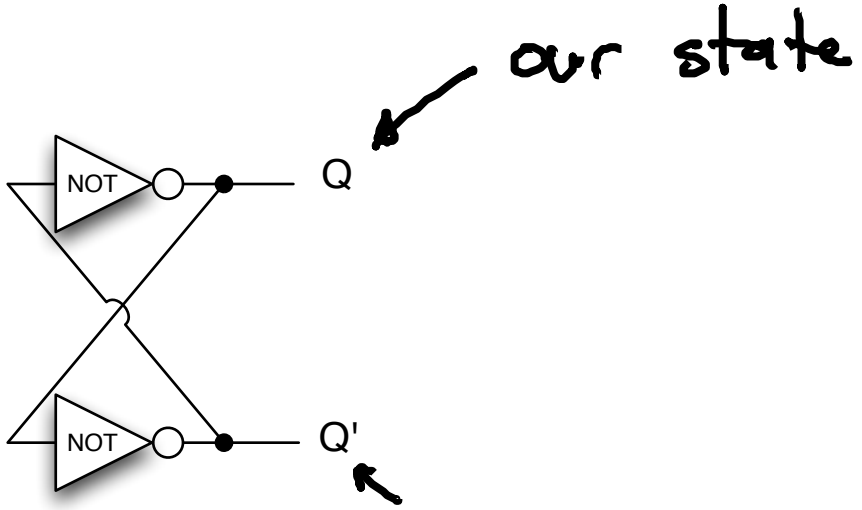


Building a useful storage mechanism

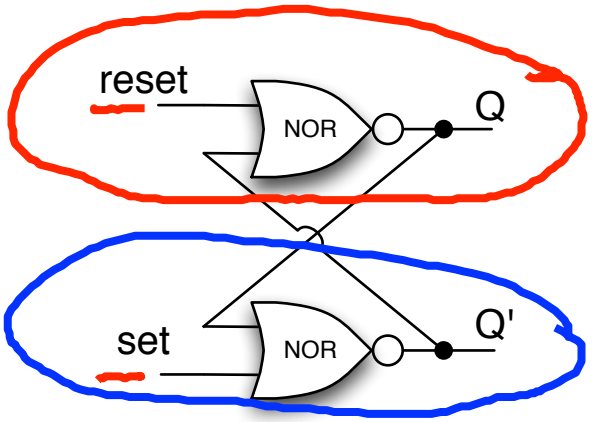
- **A memory should have at least three properties.**

  1. It should be able to hold a value. ✓
  2. You should be able to *read* the value that was saved. ✓
  3. You should be able to *change* the value that's saved. ✗

Cross-coupled inverters:



Set/Reset Latch (SR latch)



- **Cross-coupled NOR gates**
  - Two inputs:
    - **reset**: when 1, sets Q=0
    - **set**: when 1, sets Q=1
  - When **reset=set=0**, Q keeps its value
    - ***reset=set=1*** *causes bad things. Make sure this doesn't happen.*

- **This circuit has feedback, its outputs (Q, Q') are also inputs!**
  - Current values of Q, Q' depend on past values of Q, Q'

$$Q_{t=x} = (reset_{t=x-1} + Q'_{t=x-1})'$$

delay = 1 time unit

$$Q'_{t=x} = (set_{t=x-1} + Q_{t=x-1})'$$

Analyzing SR latch

$Q_{t=x} = (\text{reset} + Q'_{t=x-1})'$

$Q'_{t=x} = (\text{set} + Q_{t=x-1})'$

- **Case 1:** reset = set = 0 @ t = 0

$Q_{t=1} = (\emptyset + Q'_{t=0})' = (Q'_{t=0})' = Q_{t=0}$

$Q'_{t=1} = (\emptyset + Q_{t=0})' = (Q_{t=0})' = Q'_{t=0}$

HOLD ITS STATE

- **Case 2:** reset = 1, set = 0

$Q_{t=1} = (1 + Q'_{t=0})' = (1)' = \emptyset$

$Q'_{t=1} = (0 + Q_{t=0})' = Q'_{t=0}$

Cleared the latch

$Q = \emptyset$

$Q'_{t=2} = (0 + Q_{t=1})' = (0+0)' = 1$

- **Case 3:** reset = 0, set = 1

$Q_{t=1} =$

$Q'_{t=1} =$

Analyzing SR latch

$$Q_{t=x} = (\text{reset} + Q'_{t=x-1})'$$
$$Q'_{t=x} = (\text{set} + Q_{t=x-1})'$$

$Q_{t=2} = (0 + Q'_{t=1})'$
$= (0+0)' = (0)' = 1$

- **Case 3:** reset = 0, set = 1

$Q_{t=1} = (0 + Q'_{t=0})' = (Q'_{t=0})' = Q_{t=0}$
$Q'_{t=1} = (1 + Q_{t=0})' = (1)' = \emptyset$

@t=∅

A: 0
B: 1
C: $Q_{t=0}$
D: $Q'_{t=0}$

i>clicker

Timing diagram of an SR Latch

Initialized to Q=0

→ $Q_{t=x} = (reset + Q'_{t=x-1})'$

$Q'_{t=x} = (set + Q_{t=x-1})'$