Georgia Tech College of Engineering
School of Electrical and Computer Engineering

ECE 2020

Circuit Timing and Number Systems

Instructor: Samuel Talkington

September 24, 2024

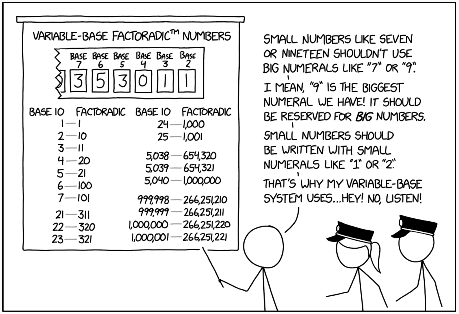
Logistics

- **Now available:**
  - **HW1 revision opportunity**: Due tonight, September 24th, 11:59pm.
  - **Lab report**: Due tonight, September 24th, 11:59pm.
- **Exam 1:**
  - Excellent performance, around half the class earned full points.
  - Grades released later this week.
- **Upcoming:**
  - **Midterm survey:** complete for +1 bonus point on your participation grade.
  - **HW3:** Released this week, due in $\approx$ 2 weeks.

Agenda

Agenda: next 2 weeks

- Circuit timing
- Number systems
- Encoders/decoders
- Multiplexers
- Adders and subtractors



FACTORIAL NUMBERS ARE THE NUMBER SYSTEM THAT SOUNDS MOST LIKE A PRANK BY SOMEONE WHO'S ABOUT TO BE ESCORTED OUT OF THE MATH DEPARTMENT BY SECURITY.

Source: xkcd

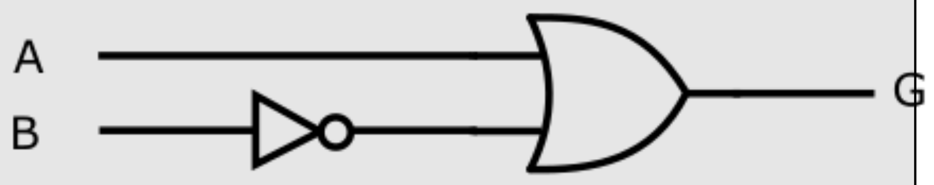
Circuit timing

Sources of delays

- Until now, we have assumed that all circuits *output* 1 *or* 0 *instantaneously*.
- However, *in real life*, it takes time for CMOS circuits to switch from $0 \rightarrow 1$ or $1 \rightarrow 0$.
- Transistors (made of **semiconductors**) need **time** to *switch* from being "**conductor**-like" to "**insulator**-like".

Timing Diagram

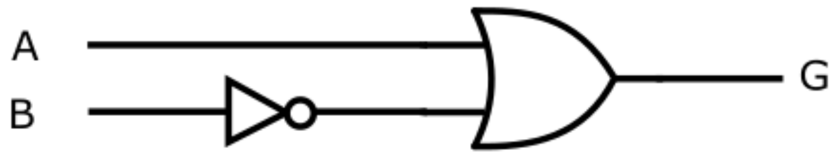
- Shows how signals change **over time**
- Consider the circuit to the right:
  - Say inverter delay = 1 ns, OR gate delay = 5 ns
- We can create a timing diagram
  - Inputs must be defined FIRST!
  - The diagram shows an arbitrary behavior of inputs

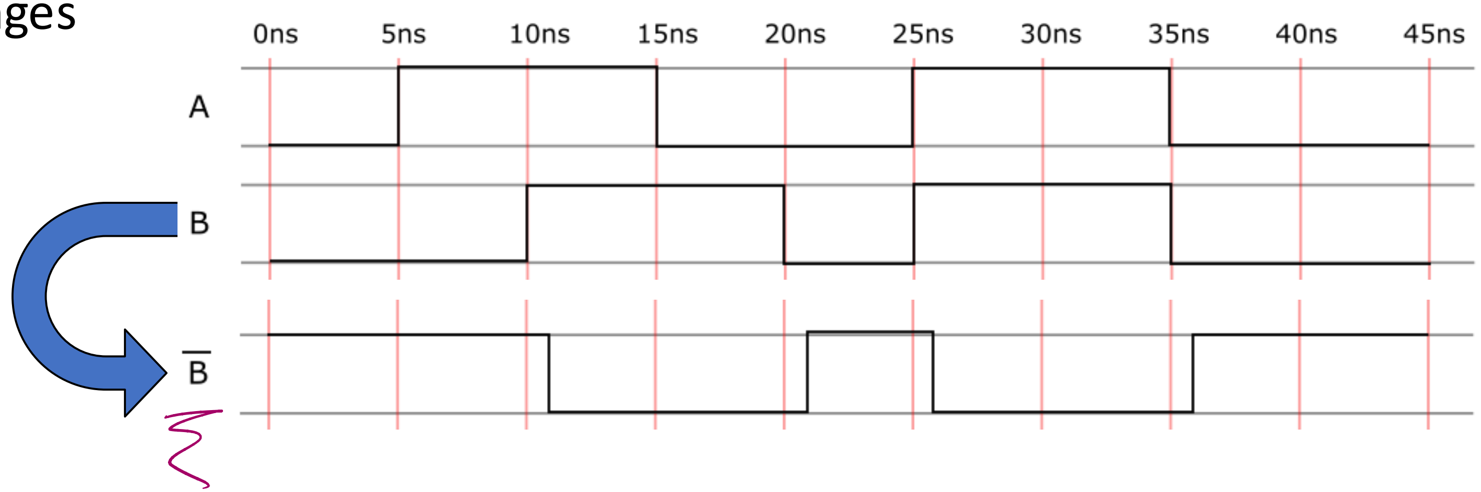


$$\mathbf{G} = \mathbf{A} + \mathbf{B}'$$

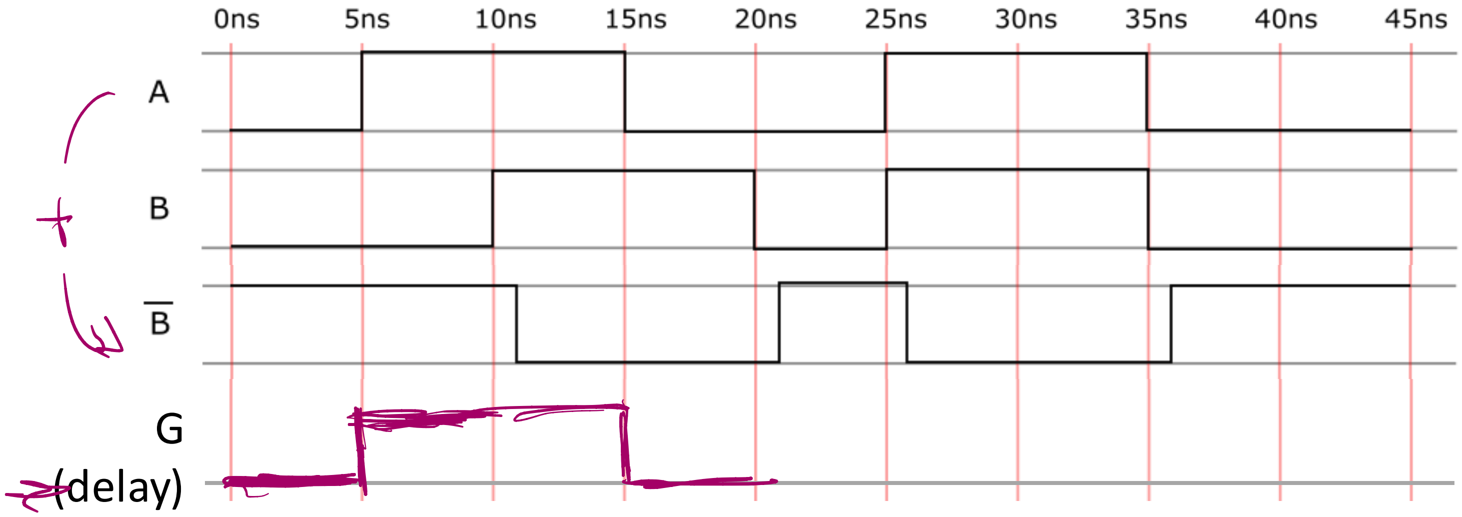
Timing Diagram

- To find how output, G will change, we first need to find how B’ changes
    - Inverter propagation delay = 1 ns.
    - ∴ NOT(B) changes 1 ns after B changes



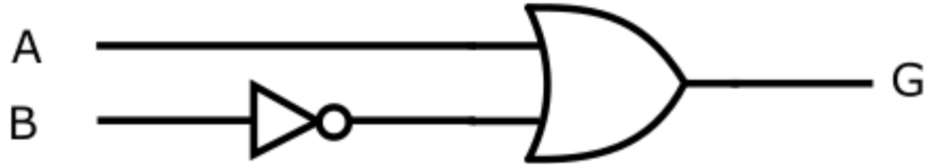


- Now, we can find how G will change
    - OR gate delay = 5 ns
    - i.e., OR gate output changes 5 ns after latest switch
    - **Tip: Look for trigger/edge points for A & B’**

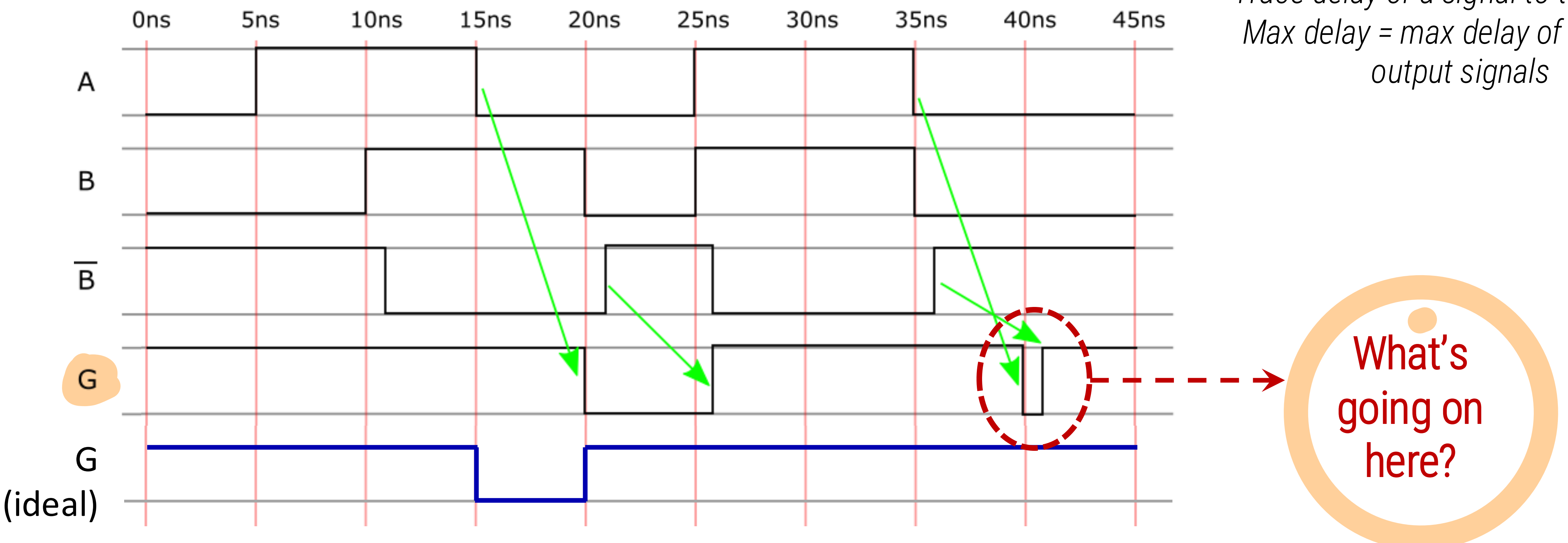


Timing Diagram

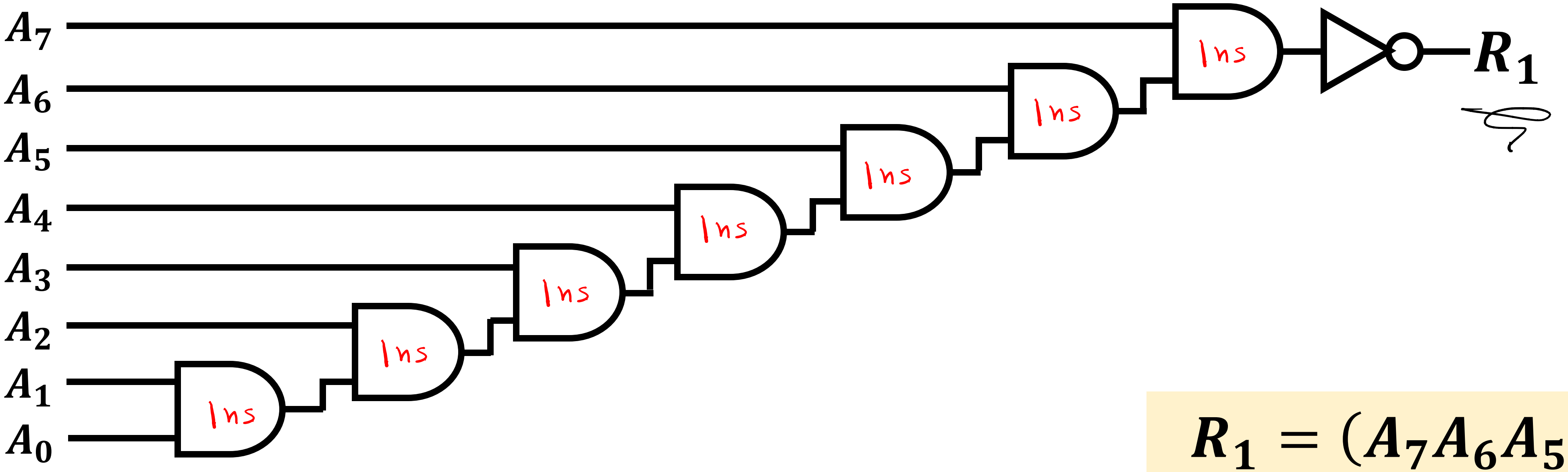
- Only a change in inputs can cause an output change
  - **Effect**: Consider only how changes propagate through a circuit
  - **Caveat**: Only for Combinational circuits (*i.e.*, no feedback paths / loops)
- Comparing ideal to actual timing response



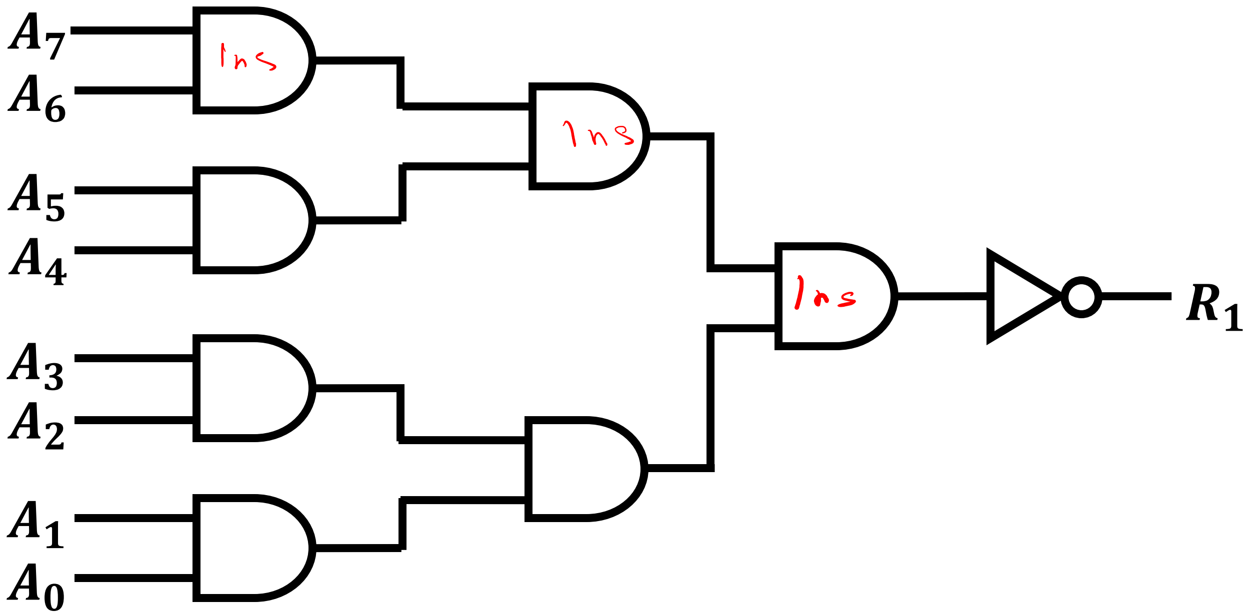
Trace delay of a signal to the output
Max delay = max delay of all input-output signals



Why does timing matter?



$$R_1 = (A_7A_6A_5A_4A_3A_2A_1A_0)'$$

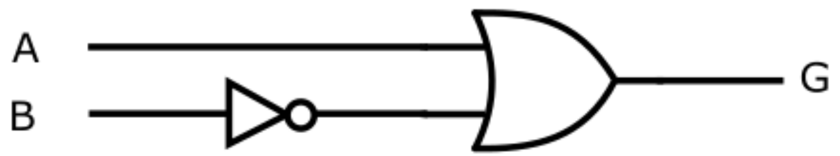


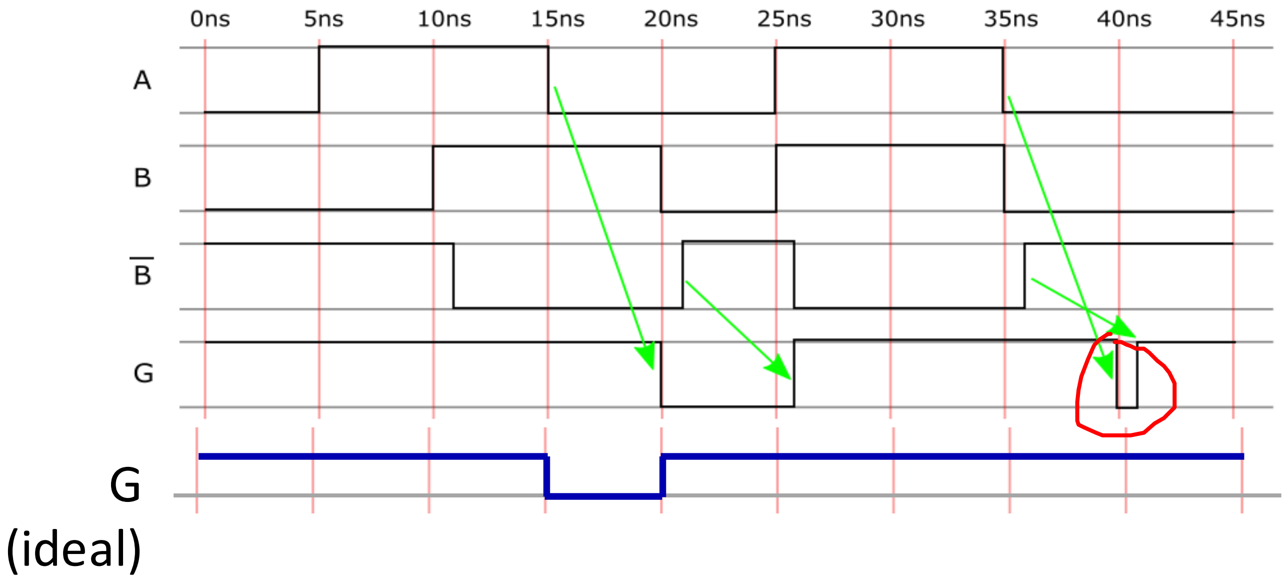
- Two possible configurations:
  - Many others possible.
- Which is preferred to reduce delay?
- What about glitches?
- Timing design in Digital Design

Timing hazards

Hazards

- **Recap:**
  - Real circuits have delays
  - Propagation delay in real circuits can cause ‘weirdness’ when inputs change



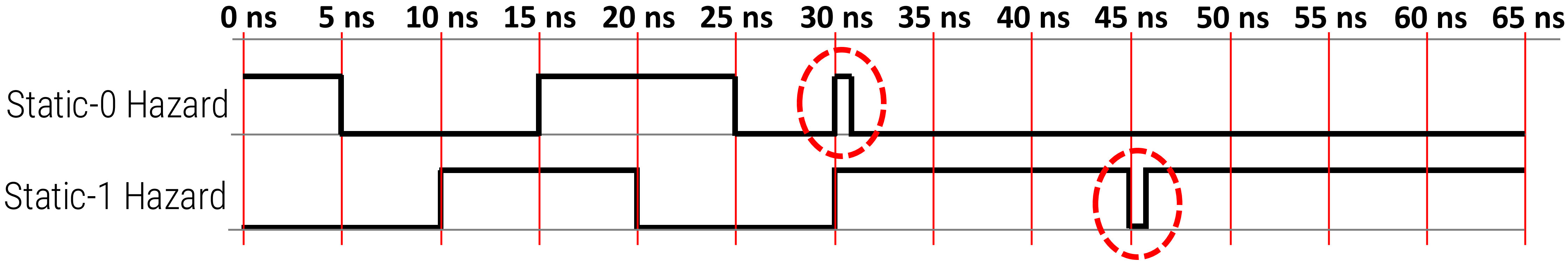


- We call these ‘glitches’ → ***‘Timing Hazards’***

Hazard Types

- **Static Hazards:**
  - When you expect the output to stay the same ('static'), but it glitches.
  - **Static-0 hazard:** Output should stay at **0**, but glitches to '1' before returning to '0'
  - **Static-1 hazard:** Output should stay at **1**, but glitches to '0' before returning to '1'

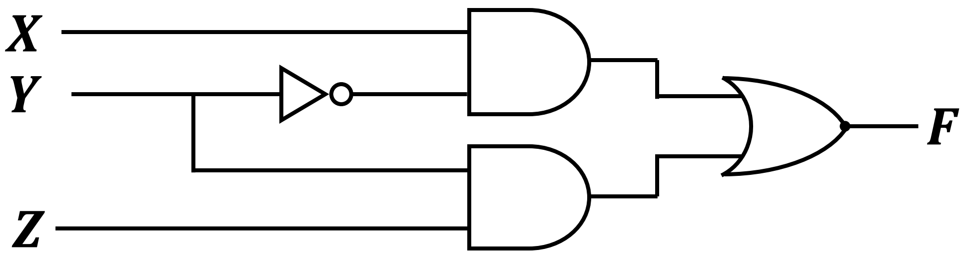
- More accurately, static hazards refer to pairs of input combinations that cause these glitches.
  - On a K-Map, these occur when there is a transition between adjacent prime implicants



- **Dynamic Hazards:**
  - Output is expected to 'cleanly' change (0→1 , or 1→0), but a short 'oscillation' occurs before output 'settles'
  - Out of course scope.

Static-1 Hazard

- Pair of input combinations that:
  - Differ in only one input variable, and
  - Both give a 1-output which can momentarily go to a ‘0’ during the transition.

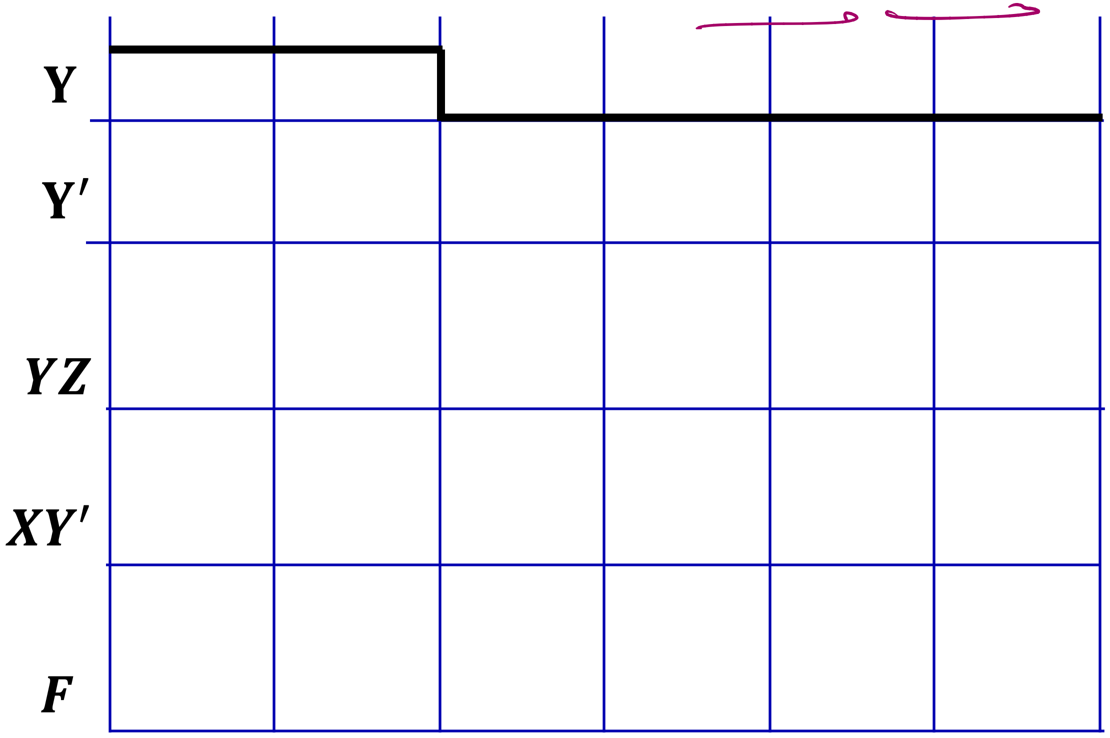


SOP Form: $F = XY' + YZ$

Z \ XY	00	01	11	10
0	0	0	0	1
1	0	1	1	1

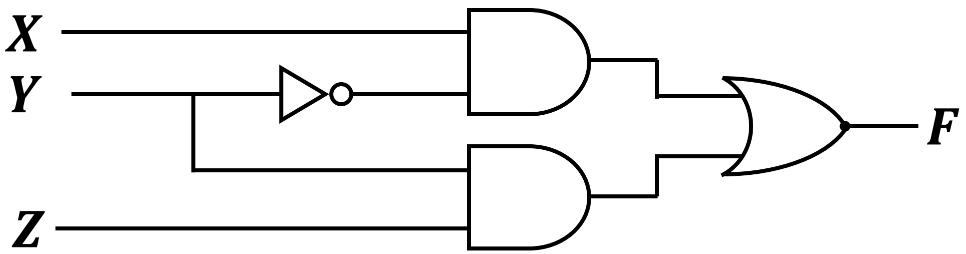
- What happens when XYZ goes from 111 to 101, with some delay in the NOT gate?

$X = 1, Z = 1$



Eliminating a Static-1 Hazard

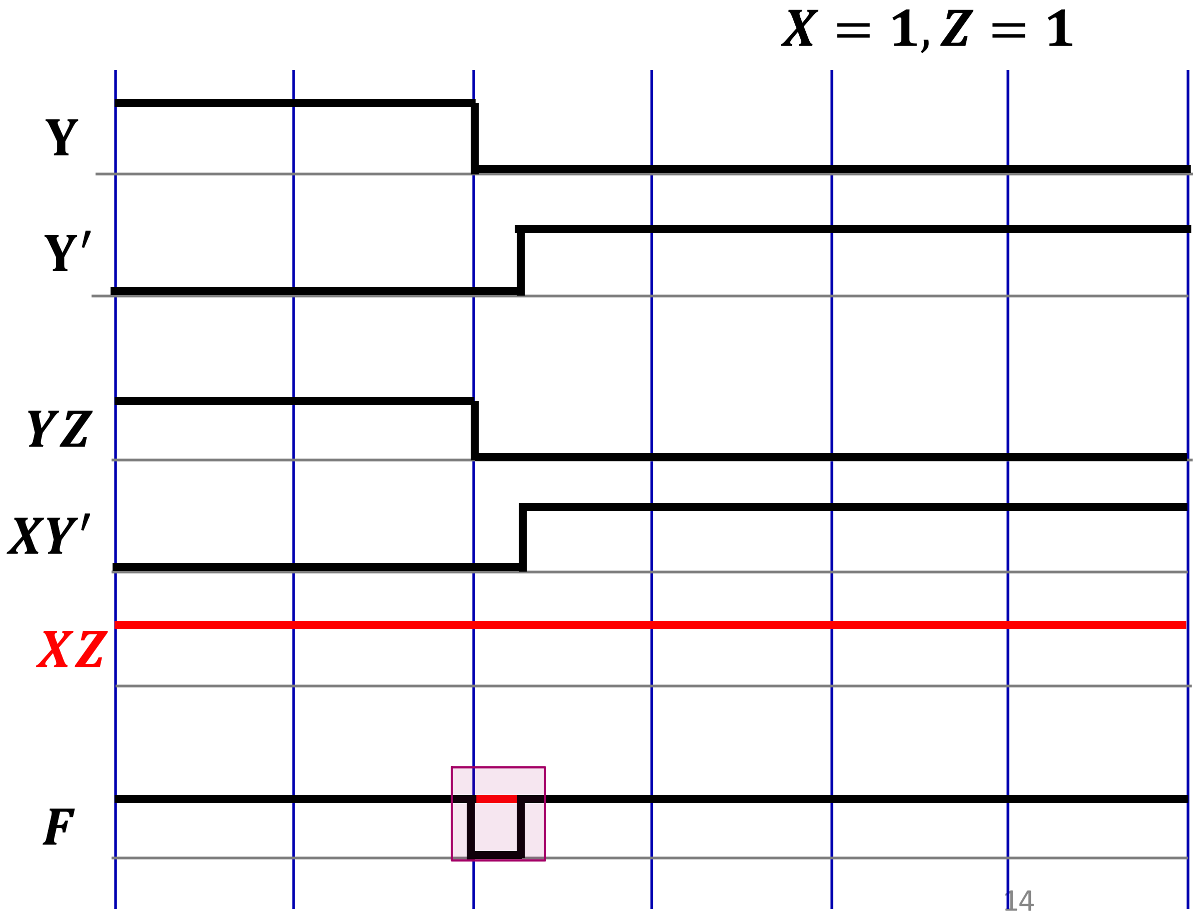
- Occurs whenever K-map has two adjacent ‘1’ cells not covered by the same product term (‘prime implicant’)
- To fix this, include extra minterms to handle transitions between non-overlapping implicants.



SOP Form: $F = XY' + YZ + XZ$

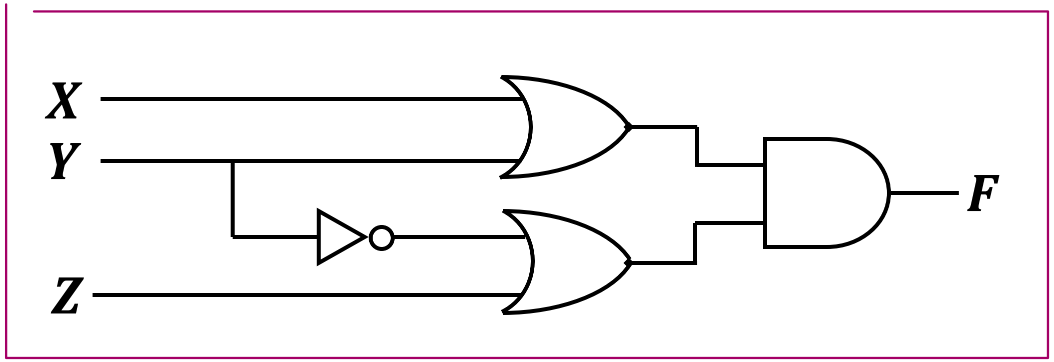
‘Consensus term’

Z \ XY	00	01	11	10
0	0	0	0	1
1	0	1	1	1



Static-0 Hazard

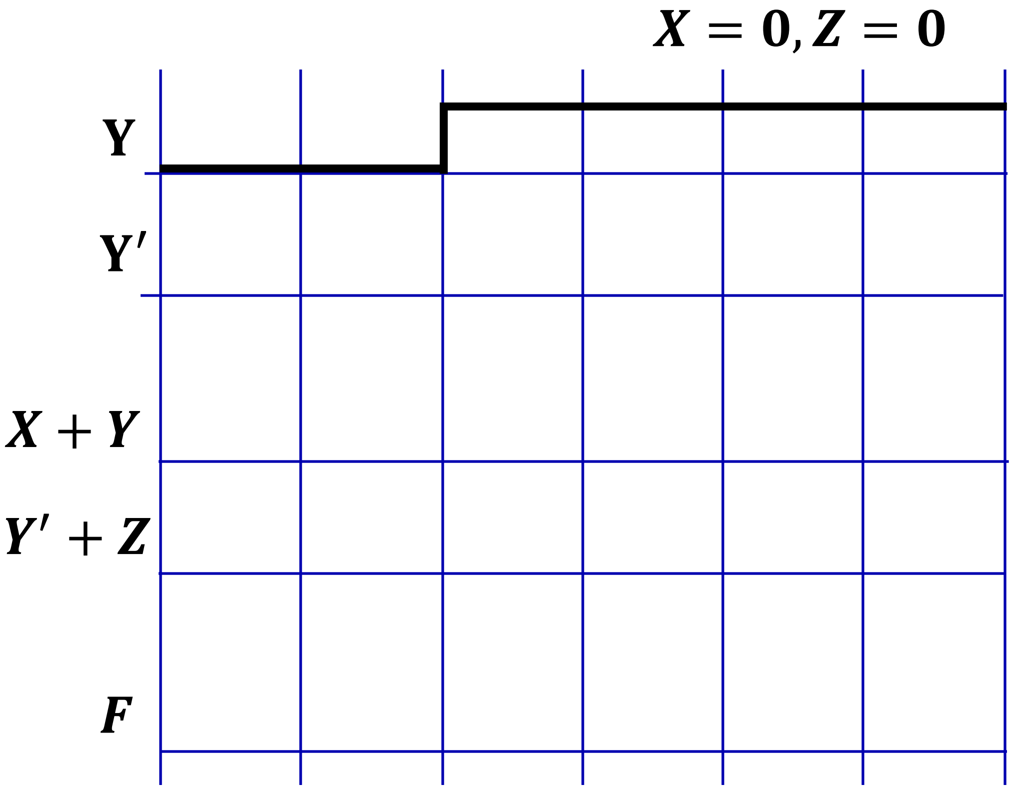
- Pair of input combinations that:
  - Differ in only one input variable, and
  - Both give a 0-output which can momentarily go to a '1' during the transition.



POS Form: $F = (X + Y) \cdot (Y' + Z)$

Z \ XY	00	01	11	10
0	0	0	0	1
1	0	1	1	1

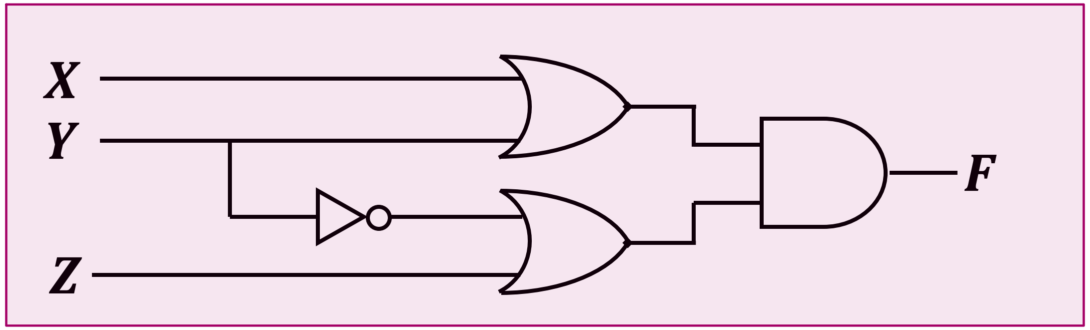
- What happens when X,Y,Z goes from 000 to 010, with some delay in the NOT gate?



Eliminating a Static-0 Hazard

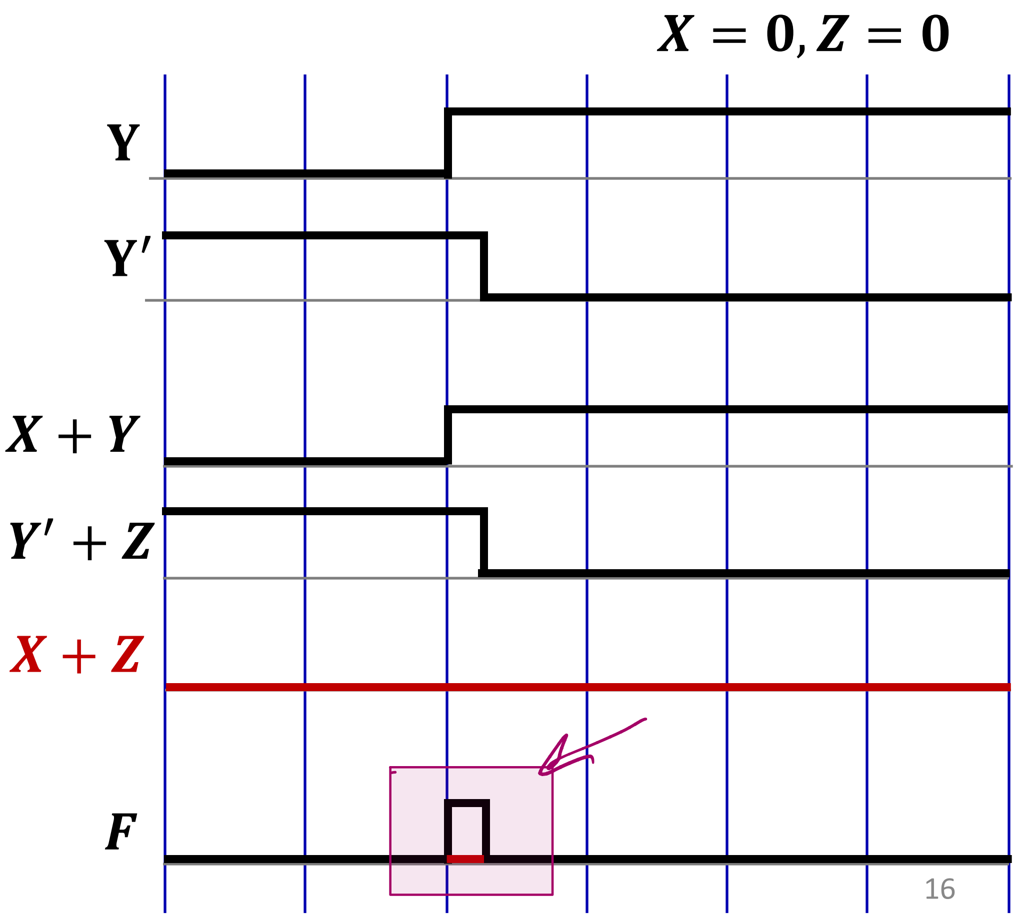
- Occurs whenever K-map has two adjacent ‘0’ cells not covered by the same sum term

- To fix this, include extra maxterms to handle transitions between non-overlapping implicants.



POS Form: $F = (X + Y) \cdot (Y' + Z) \cdot (X + Z)$

Z \ XY	00	01	11	10
0	0	0	0	1
1	0	1	1	1



Hardware changes when removing hazards

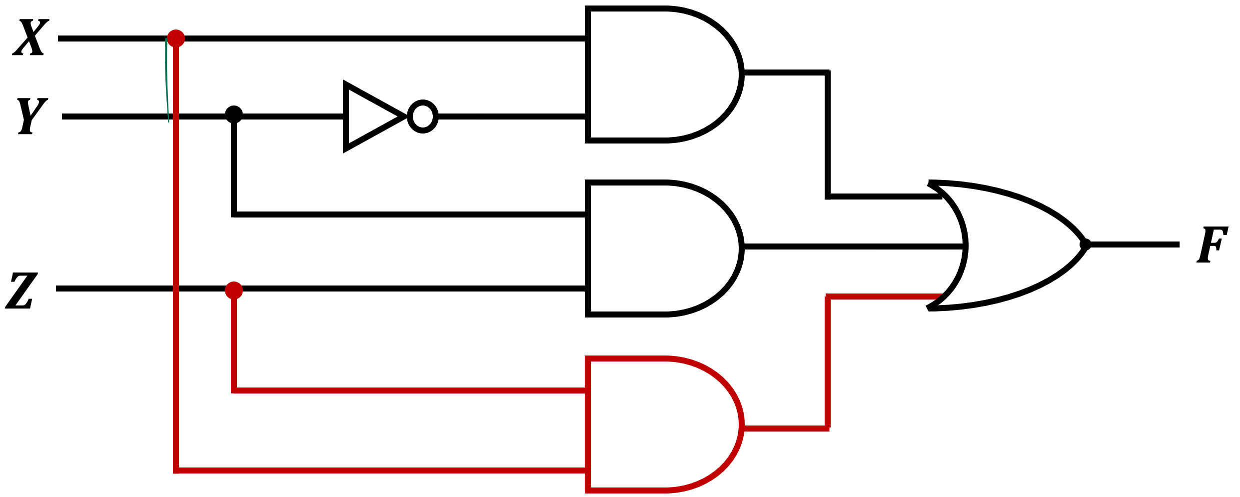
"Is the hardware unchanged if we include consensus terms?"

- Answer: Hardware **<u>IS</u>** changed when we include consensus terms.

SOP Form: $\boldsymbol{F = XY' + YZ}$ $\boldsymbol{+ XZ}$

'Consensus term'

Z \ XY	00	01	11	10
0	0	0	0	1
1	0	1	1	1



Identifying timing hazards

grouping

- **Occurs when we have <u>neighboring but not overlapping implicants</u> in a K-map**
- Mitigated by adding a consensus term into the K-map

- **Static-1 Hazard:**
  - The output should ideally stay at 1, but glitches to 0 for a short period of time
  - Occurs in SoP-form expressions / K-maps with 1's circled

- **Static-0 Hazard:**
  - The output should ideally stay at 0, but glitches to 0 for a short period of time
  - Occurs in PoS-form expressions / K-maps with 0's circled

Numbers: How do they work?

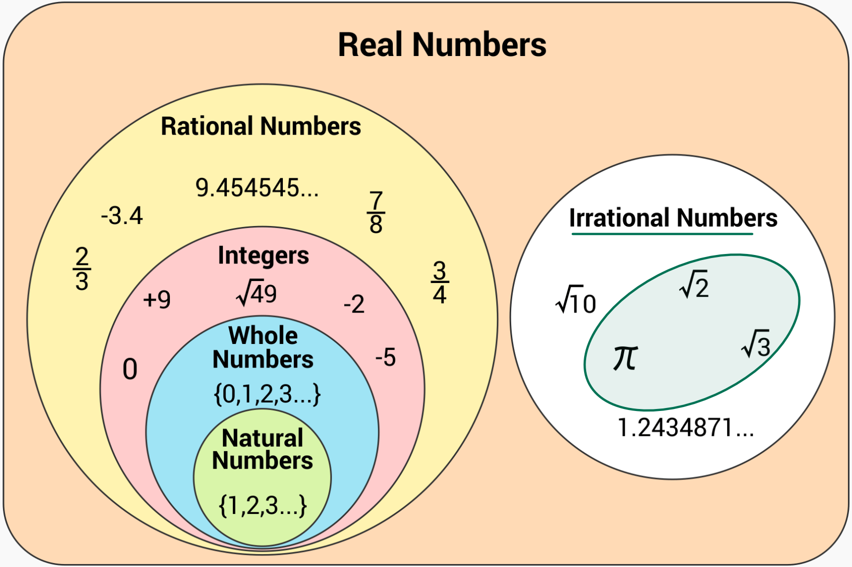


Figure 1: The subsets of the real numbers

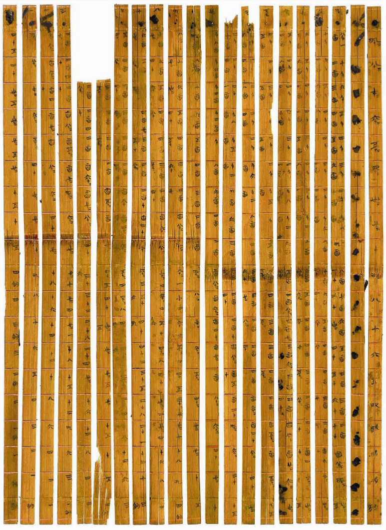


Figure 2: Oldest known base-10 multiplication table, China, c. 305 BC

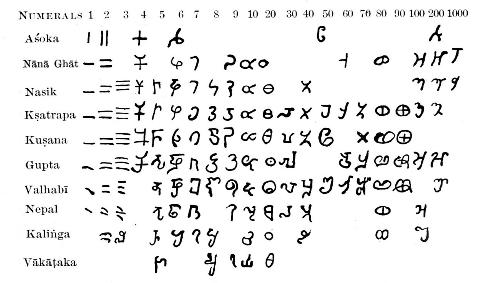


Figure 3: Evolution of Hindu-Arabic numerials, starting with Edicts of Ashoka, c. 250 BC

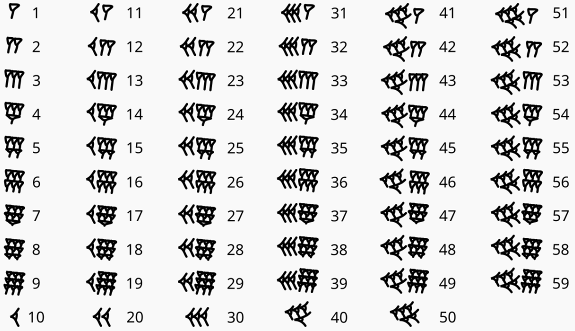


Figure 4: Babylonian cuneiform numerals, c. 2000 BC

The Babylonian cuneiform numerals, c. 2000 BC, were the first positional number system; shockingly, they were **base-60**, or *sexagesimal*. (???)

Bottom line

How we represent numbers is a *choice of human definition.*

Classroom discussion

There's an infinite number of real numbers, ("uncountably" infinite), and an infinite number of natural numbers, ("countably" infinite). Yet, we represent everything in terms of *just 10* of the natural numbers: $0, 1, \ldots, 9$.

Think about it for a second:

- How can we store numbers in computers?
- What kind of numbers would fit well into digital logic design?

Positional number systems

Question: How exactly do we represent a number?

Answer: We have to agree on the total number of **unique**, or **base** numbers to build our numbers from; the number of such unique numbers is called the *radix*.

$$\text{usual digits} = \underbrace{\{0, 1, 2, \ldots, 9\}}_{\#\text{digits}=\text{b},}$$

The total number of unique digits in a number system, b, is called the radix.

Positional number systems ii

How? Positional numbers work by exponentiating the radix, multiplying the value of its place, and summing all of these together.

Example:

$$(241)_{10} = \left(2 \times 10^2\right) + \left(4 \times 10^1\right) + \left(1 \times 10^0\right)$$

We can make this more general!

Definition: Base-b number system

A *base-b number system* with *radix* $b > 1$ represents any number $x \in \mathbb{R}$ as a *string of digits* a_i in n "places" $i = 0, 1, \ldots, n-1$, where each a_i is one of b possible digits in a digit set $\mathcal{D}$:

$$a_i \in \mathcal{D} = \{d_1, d_2, \ldots, d_b\}.$$

Any real number x can be represented in a base-b system as the following sum:

$$x = (a_{n-1}a_{n-2}\ldots a_1a_0)_b = \sum_{i=0}^{n-1} a_i \times b^i. \quad (1)$$

Base-10 number system

Base-10 numbers are the numbers we all know and love.

Examples:

- $\underbrace{3}_{=a_0} = 3 \times 10^0$
- $\underbrace{53}_{a_1 a_0} = 5 \times 10^1 + 3 \times 10^0$
- $\underbrace{125}_{=a_2 a_1 a_0} = \sum_{i=0}^{2} a_i \times 10^i = a_2 \times 10^2 + a_1 \times 10^1 + a_0 \times 10^0$

Base-2 (Binary) number system

Base-2 (a.k.a. binary) numbers are the numbers we are all (starting) to know and love.

$(\cdot)_{10} \equiv$ base 10 $D = \{0, 1\}$

Examples:

- $\underbrace{10}_{a_1 a_0} = 1 \times 2^1 + 0 \times 2^0 = (2)_{10}$
- $\underbrace{110}_{a_2 a_1 a_0} = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (6)_{10}$
- $\underbrace{11010}_{a_4 a_3 a_2 a_1 a_0} = \sum_{i=0}^{3} a_i \times 2^i = 2^4 + 2^3 + 0 + 2^1 + 0 = (26)_{10}$

Base-16 (Hexadecimal) number system

In **base-16** or *hexadecimal* numbers, the set of base digits are:

$$\mathcal{D} = \{0, 1, 2, \ldots, 9, A, B, C, D, E, F\},$$

where:

$$(A)_{10} = 10, \quad (B)_{10} = 11, \quad (C)_{10} = 12,$$
$$(D)_{10} = 13, \quad (E)_{10} = 14, \quad (F)_{10} = 15.$$

Why care about non-base-10?

- base-2 (binary): Digital logic, all of computing instruction are converted to this.
- base-8: 3-bit information (useful in analysis, prototyping)
- base-16: 4-bit information (**tons** of computer stuff)
  - 32-bit IP addresses are 8 digits
  - 32-bit CPU instructions are 8 digits
- base-60: Deciphering ancient Babylonian Cuneiform tablets (**essential**)

Converting between number systems

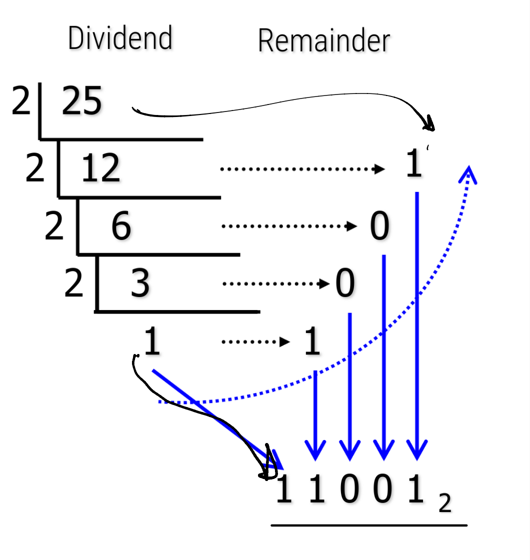


Figure 5: General procedure for converting $(25)_{10}$ to binary.

Converting between number systems i

To **convert** a base-b number to binary (base-2), you can follow these general concepts:

- Convert the number to base-10 using the appropriate number system.
- Divide the decimal number by 2.
- Note the remainder.
- Repeat the previous 2 steps for the quantient till the quotient is zero.
- Write the remainders in reverse order.

Example: converting a hexadecimal number to binary

How to convert this hexadecimal number to binary?

$$(4A)_{16} = (?)_2.$$

Solution: For the case of hex→binary, you can **individually convert** each digit into a **4-bit** binary digit.

$$(2A)_{16} = (42)_{10} = \underbrace{0010}_{=(2)_{10}}\ \underbrace{1010}_{(10)_{10}}$$

$$2A = \sum_{i=0}^{n} a_i b^i = 2 \cdot 16^1 + A \cdot 16^0 = 2 \cdot 16^1 + 10 \cdot 1 = 32 + 10 = (42)_{10}$$

$$(2A)_{16} = (0010\ 1010)_2, \quad 1 \cdot 2^3 + 1 \cdot 2^1 = 10$$

Fractional number representations

Fixed-point fractional representation i

Consider the number 5.75 in base-10:

$$\underbrace{5}_{\text{whole part}}\ \underbrace{.}_{\text{decimal point}}\ \underbrace{75}_{\text{fractional part}}$$

Equivalently, in binary, $5.75 = 101.11$:

$$\underbrace{101}_{\text{whole part}}\ \underbrace{.}_{\text{binary point}}\ \underbrace{11}_{\text{fractional part}}$$

The reason for this is because:

$$(5)_{10} = (101)_2,$$

and

$$\begin{aligned}(0.75)_{10} &= (0.5)_{10} + (0.25)_{10} \\ &= 1 \times 2^{-1} + 1 \times 2^{-2}\end{aligned}$$

Puzzle

Next time

Next time:

1. Conversion between arbitrary number systems
2. Negative binaries, signed magnitude, 2s complement
3. Building blocks

Participation puzzle

Perform these conversions:

$$(11001)_2 = (?)_{10}$$
$$(B4)_{16} = (?)_{10}$$

Due by 11:59pm tonight, password: `radix`

Bonus puzzles (to be discussed Thursday)

Perform these conversions:

$$(3A6.C)_{16} = (?)_2 = (?)_8 = (?)_{10}$$
$$(1010011100)_2 = (?)_{16} = (?)_8 = (?)_{10}$$